Uniwersytet Warszawski

Wydział Matematyki,
Informatyki i Mechaniki

Anna Urbańska

Kombinatoryczne Algorytmy dla Algebraicznych Problemów Związanych z Macierzami

Rozprawa Doktorska

Promotor rozprawy
prof. dr hab. Wojciech Rytter
Instytut Informatyki

Styczeń 2009

Oświadczenie promotora rozprawy

Niniejsza rozprawa jest gotowa do oceny przez recenzentów.

Data Podpis promotora rozprawy

Oświadczenie autora rozprawy

Oświadczam, że niniejsza rozprawa została napisana przeze mnie samodzielnie.

Data Podpis autora rozprawy

Streszczenie

Metoda kombinatoryczna liczenia problemów macierzowych to sprowadzanie problemów natury numerycznej do prostych problemów teorii grafów i kombinatoryki. Kilka lat temu Mahajan i Vinay w sposób nowatorski zapoczątkowali algorytmikę tego typu. Szczególną specyfiką ich metod jest unikanie dzielenia. Wyeliminowanie dzielenia jest istotnym czynnikiem podczas pracy z pierścieniem przemiennym, który nie jest ciałem, na przykład, kiedy elementy pierścienia są liczbami całkowitymi, wielomianami czy też jeszcze bardziej skomplikowanymi wyrażeniami. Takie obliczenia pojawiają się w wielu kombinatorycznych problemach.

Celem pracy jest przede wszystkim przedstawienie nowego podejścia i dogłębniejsze zbadanie algorytmów Mahajana i Vinaya: uproszczenie algorytmów, interpretacja w terminach problemu plecakowego oraz pokazanie możliwości znacznego ich przyspieszenia.

W niniejszej pracy prezentujemy algorytmy nie wykonujące operacji dzielenia rozwiązujące problemy obliczania wyznacznika, Pfaffianu, współczynników wielomianu charakterystycznego oraz macierzy dopełnień algebraicznych dla dowolnych macierzy w czasie $\tilde{O}(n^{3.03})$ oraz dla macierzy definiujących grafy planarne w czasie $O(n^{2.5})$. Nasze algorytmy w przypadku dowolnych macierzy posiadają bardzo proste implementacje w czasie $O(n^{3.5})$ a jedynym nietrywialnym elementem algorytmów działających w czasie $\tilde{O}(n^{3.03})$ jest algorytm szybkiego mnożenia macierzy.

Zaprezentowane przez nas algorytmy są relatywnie prostsze w porównaniu z wcześniejszymi rozwiązaniami i wykorzystują algorytmy rozwiązywania znanych problemów kombinatorycznych.

Słowa kluczowe: Algorytm, Wyznacznik, Kombinatoryka, Graf, Graf Planarny, Macierz, Pfaffian.

Klasyfikacja tematyczna ACM: F.2.1, F.2.2, G.2.1, G.2.2.

Abstract

Combinatorial method of evaluation of algebraic functions means reducing numerical problems to simple graph problems and combinatorics. Several years ago Mahajan and Vinay began algorithmics of such type. The attractive feature of their methods is that they are division-free. Avoiding divisions seems attractive when working over a commutative ring which is not a field, for example when the entries are integers, polynomials, or rational or even more complicated expressions. Such computations arise in combinatorial problems.

The main topic of this dissertation is to exploit new possibilities and show new aspects of beautiful combinatorial algorithms of Mahajan and Vinay: simplification of algorithms, showing a relation to a pseudo-polynomial dynamic-programming algorithm for the knapsack problem and showing the possibility of significant speeding up.

In this work, we present division-free algorithms for the evaluation of algebraic functions such as a matrix determinant, matrix Pfaffian, characteristic polynomial and matrix adjoint in general and planar matrices. The algorithms for the general case have $\tilde{O}(n^{3.03})$ time complexity and the algorithms for the planar case have $O(n^{2.5})$ time complexity. Our algorithms for the general case have very simple implementations in time $O(n^{3.5})$ and the only non-trivial element of the $\tilde{O}(n^{3.03})$ algorithms is the fast matrix multiplication algorithm.

Our algorithms are simpler than previously known algorithms and they are related to well known combinatorial problems.

Keywords: Algorithm, Determinant, Combinatorics, Graph, Matrix, Planar Graph, Pfaffian.

ACM Classification: F.2.1, F.2.2, G.2.1, G.2.2.

Spis treści

1 Wstęp **9**

2 Pojęcia Podstawowe, Oznaczenia i Założenia **13**
2.1 Grafy 13
2.1.1 Podstawowe Definicje 13
2.1.2 Grafy Planarne 14
2.1.3 Skojarzenia w Grafach 15
2.2 Problem Plecakowy 15
2.3 Algebra Liniowa 15
2.3.1 Podstawowe Definicje 15
2.3.2 Algebra Macierzy 16
2.3.3 Macierze Skośno-Symetryczne 18
2.4 Mnożenie Macierzy 19

3 Wyznacznik **21**
3.1 Wprowadzenie 21
3.2 Nowe Spojrzenie na Algorytm MV 22
3.2.1 Zmodyfikowany Problem Plecakowy 22
3.2.2 Algebraiczny Problem Plecakowy 22
3.2.3 Skośne Domknięcie Macierzy 23
3.2.4 Ciągi Pseudocykli i Twierdzenie Mahajana-Vinaya 24
3.2.5 Algorytm MV 26
3.2.6 Nasza Wersja Algorytmu MV 27
3.2.7 Obliczanie $Knapsack(A)$ 28
3.3 Przyspieszenie Algorytmu Obliczania Wyznacznika 29
3.3.1 Obliczanie $SkewClosure(A)$ 29
3.3.2 Wykorzystanie Algorytmu Szybkiego Mnożenia Macierzy . . . 34

4 Pfaffian **39**
4.1 Wstęp 39
4.2 Naprzemienne Ciągi Pseudocykli 40
4.3 Nasza Wersja Algorytmu MSV 42

5 Grafy Planarne **45**
5.1 Wstęp 45
5.2 Grafy Planarne i Mały Separator 45
5.3 Algorytm Numerujący 46
5.4 Obliczanie $SkewClosure(A)$ 47
5.5 Szybszy Algorytm Obliczania $Knapsack(A)$ 50

6 Inne Zastosowania **53**
6.1 Wielomian Charakterystyczny 53
6.2 Macierz Dopełnień Algebraicznych 55

7 Znane Algorytmy Obliczania Wyznacznika **57**
7.1 Wstęp 57
7.2 Metoda Samuelsona 58
7.2.1 Algorytm Używający Ciągów Pseudocykli z Własnością Prefiksów 60
7.2.2 Nasza Wersja Algorytmu Samuelsona 62
7.3 Algorytm Chistova 63
7.3.1 Algorytm Używający Tablic Ciągów Marszrut 65
7.3.2 Nasza Wersja Algorytmu Chistova 66
7.4 Algorytm Csanky'ego 67

Rozdział 1

Wstęp

W niniejszej pracy przedstawimy zastosowanie metod kombinatorycznych w konstrukcji efektywnych algorytmów rozwiązujących wybrane problemy algebry liniowej. Metody kombinatoryczne okazują się bardzo przydatne w wielu problemach algebraicznych, takich jak obliczanie wyznacznika, Pfaffianu, współczynników wielomianu charakterystycznego czy macierzy dopełnień algebraicznych. Konstruowane w ten sposób algorytmy wykorzystują jako procedury operacje grafowe. Dzięki temu zyskują zazwyczaj na czytelności.

Głównym tematem mojej rozprawy będą algorytmy kombinatoryczne rozwiązujące problemy algebraiczne bez używania dzielenia. Wyeliminowanie dzielenia jest istotnym czynnikiem podczas pracy z pierścieniem przemiennym, który nie jest ciałem, na przykład, kiedy elementy pierścienia są liczbami całkowitymi, wielomianami czy też jeszcze bardziej skomplikowanymi wyrażeniami. Takie obliczenia pojawiają się w wielu kombinatorycznych problemach [27].

Tło

Pierwsze próby zastosowania technik kombinatorycznych do dowodzenia poprawności lub tworzenia nowych algorytmów obliczających funkcje macierzowe zostały podjęte przez Valianta [52] w 1992 roku. Valiant analizował algorytm Samuelsona obliczania współczynników wielomianu charakterystycznego i zinterpretował te obliczenia w sposób kombinatoryczny. Przedstawił on kombinatoryczne twierdzenie posługując się terminologią zamkniętych ścieżek w grafie, poprawność którego wynikała wprost z poprawności algorytmu Samuelsona. Zainspirowani tym oraz czysto kombinatorycznym oraz wyjątkowo eleganckim dowodem twierdzenia Cayleya-Hamiltona podanym przez Rutherforda [39], Mahajan i Vinay [35] zaprezentowali pierwszy kombinatoryczny al-

gorytm obliczania wielomianu charakterystycznego. Dowód poprawności ich algorytmu odwoływał się jedynie do argumentów kombinatorycznych bez korzystania z algebry liniowej lub arytmetyki na wielomianach.

Nasze Rezultaty

Jednym z klasycznych problemów algebry liniowej jest problem obliczania wyznacznika. Najbardziej znanym algorytmem obliczania wyznacznika jest algorytm eliminacji Gaussa. Wymaga on $O(n^3)$ (lub $O(n^{2.38})$, jeśli użyty zostanie algorytm szybkiego mnożenia macierzy) operacji dodawania, odejmowania, mnożenia oraz *dzielenia*. Z drugiej strony, definicja wyznacznika jako sumy $n!$ iloczynów pokazuje, że może on zostać obliczony *bez* wykonywania operacji dzielenia. Mahajan i Vinay w swojej pracy z 1997 roku [34] przedstawili zupełnie nową kombinatoryczną interpretację wyznacznika redukując problem jego obliczania do problemu sumowania ścieżek w pewnym acyklicznym grafie a następnie bazując na tej charakteryzacji podali algorytm obliczania wyznacznika w czasie $O(n^4)$, który nie wykorzystywał operacji dzielenia.

W oparciu o artykuł [50], zaprezentowany na konferencji ISAAC 2007, przedstawimy nowe spojrzenie na algorytm Mahajana i Vinaya: relację z pseudo-wielomianowym algorytmem programowania dynamicznego dla problemu plecakowego. Główna faza algorytmu Mahajana i Vinaya może być zinterpretowana jako obliczenia algebraicznej wersji problemu plecakowego, co jest alternatywą dla podejścia opartego na teorii grafów użytego w oryginalnym algorytmie. Głównym rezultatem [50, 51] będzie pokazanie jak zaimplementować algorytm Mahajana i Vinaya bez korzystania z operacji dzielenia w czasie $\tilde{O}(n^{3.03})$. Zaprezentowany przez nas algorytm posiada bardzo prostą implementację w czasie $O(n^{3.5})$ a jedynym nietrywialnym elementem algorytmu działającego w czasie $\tilde{O}(n^{3.03})$ jest algorytm szybkiego mnożenia macierzy.

Jak się okazuje technika wprowadzona przez Mahajana i Vinaya może być wykorzystana do wyznaczania *wszystkich* współczynników wielomianu charakterystycznego [35]. Opisane w niniejszej pracy rezultaty [50] dla przypadku wielomianu charakterystycznego będą analogiczne do tych dla wyznacznika.

Dzięki technice wprowadzonej przez Baura i Strassena [4] pokazującej jak obliczać wszystkie pochodne cząstkowe otrzymujemy [50] nie wykonujący dzielenia algorytm obliczania macierzy dopełnień algebraicznych działający w tym samym czasie $\tilde{O}(n^{3.03})$.

Rozważamy również Pfaffian macierzy skośno-symetrycznych ściśle związany z pojęciem wyznacznika. Pfaffian pojawia się w badaniach nad skojarzeniami w grafach [30]: Pfaffian zorientowanego grafu zdefiniowany jest jako suma przebiegająca po wszystkich możliwych skojarzeniach doskonałych, przy czym każde skojarzenie ma

przypisany znak zależny od orientacji. To wskazuje już na podobieństwa do wyznacznika. Gdyby nie obecność znaku w tej definicji Pfaffian można by wykorzystać do wyznaczania liczby skojarzeń doskonałych. Wiadomo, że istnieją specjalne grafy, które można zorientować w taki sposób, aby wszystkie skojarzenia doskonałe miały dodatni znak. To oczywiście oznacza, że żadne skracanie się składników w sumie nie będzie miało miejsca a stąd Pfaffian da dokładnie liczbę skojarzeń doskonałych. Pfaffian możemy wyznaczyć wykonując procedurę eliminacji dla macierzy skośno-symetrycznych, która jest podobna do eliminacji Gaussa (i wykorzystuje operację dzielenia). Alternatywnie, pierwiastek kwadratowy z wyznacznika daje Pfaffian z dokładnością do znaku. Żadne z tych podejść nie może jednak zostać użyte, jeśli operacje dzielenia i pierwiastkowania będą musiały zostać wyeliminowane. Korzystając z techniki kombinatorycznej opracowanej przez Mahajana, Subramanya i Vinaya [33] dostajemy również i dla przypadku Pfaffianu rezultaty analogiczne do tych jak dla wyznacznika.

Problem obliczania wyznacznika wydaje się być znacznie prostszy dla macierzy określonych przez grafy planarne. Na samym początku takie macierze są rzadkie, mają one jedynie $O(n)$ niezerowych współrzędnych. Jednak możemy powiedzieć znacznie więcej. Lipton, Rose oraz Tarjan [31] pokazali, że istnienie rodziny $O(\sqrt{n})$-separatorów dla tej klasy grafów daje możliwość wykonania eliminacji Gaussa w czasie $O(n^{3/2})$ (lub $O(n^{1.19})$, jeśli użyty zostanie algorytm szybkiego mnożenia macierzy). Ale działanie tego algorytmu ciągle wymaga dzielenia. W pracy prezentujemy algorytmy, opisane w naszym kolejnym artykule [49], obliczania wyznacznika, wielomianu charakterystycznego oraz macierzy dopełnień algebraicznych dla przypadku grafów planarnych działające w czasie $O(n^{2.5})$ i nie wykonujące operacji dzielenia.

Pokażemy również jak możemy wykorzystać nasze wyniki do implementacji znanych algorytmów obliczania wyznacznika: algorytmu Samuelsona oraz algorytmu Chistova w czasie $\tilde{O}(n^{3.03})$.

Organizacja Niniejszej Pracy

Rozpoczniemy od przypomnienia podstawowych definicji i faktów w rozdziale 2. W rozdziale 3 przedstawimy nowe spojrzenie na algorytm Mahajana i Vinaya oraz pokażemy jak można go zaimplementować bez korzystania z operacji dzielenia w czasie $\tilde{O}(n^{3.03})$. W rozdziale 4 rozważymy Pfaffian macierzy skośno-symetrycznych ściśle związany z pojęciem wyznacznika oraz przedstawimy algorytm jego wyznaczania w czasie $\tilde{O}(n^{3.03})$. Rozdział 5 poświęcimy na opisanie algorytmu znajdowania wyznacznika dla macierzy określonych przez grafy planarne w czasie $O(n^{2.5})$. W

rozdziale 6 zaprezentujemy rozszerzenia naszych algorytmów z rozdziałów 3 oraz 5 do algorytmów obliczania wszystkich współczynników wielomianu charakterystycznego oraz macierzy dopełnień algebraicznych. Ostatecznie w rozdziale 7 przedstawimy interpretacje kombinatoryczne znanych algorytmów obliczania wyznacznika: algorytmu Samuelsona, Chistova, Csanky'ego. Pokażemy również jak możemy wykorzystać nasze wcześniejsze wyniki do implementacji algorytmu Samuelsona oraz algorytmu Chistova w czasie $\tilde{O}(n^{3.03})$.

Podziękowania

Serdecznie dziękuje mojemu promotorowi, profesorowi Wojciechowi Rytterowi za opiekę nad tą pracą.

Rozdział 2

Pojęcia Podstawowe, Oznaczenia i Założenia

W tej części przypomnimy podstawowe definicje i fakty z teorii grafów, algebry liniowej oraz mnożenia macierzy. Dla wygody czytelnika, na końcu rozprawy zamieszczono indeks zawierający wszystkie zdefiniowane pojęcia wraz z numerami stron, na których można odszukać stosowne definicje.

2.1 Grafy

2.1.1 Podstawowe Definicje

Grafy Niezorientowane

Graf niezorientowany G definiujemy jako parę skończonych zbiorów $G = (V, E)$. Elementy zbioru V nazywamy *wierzchołkami* grafu G. Zbiór E zawiera nieuporządkowane pary wierzchołków, tzn.

$$E \subseteq \{\{u, v\} : u, v \in V\}.$$

Elementy zbioru E nazywamy *krawędziami* grafu G. Wierzchołki u i v nazywamy *końcami* krawędzi $\{u, v\}$. Mówimy też, że wierzchołek u jest *incydentny* z krawędzią $\{u, v\}$ (podobnie wierzchołek v) oraz że u jest *sąsiadem* wierzchołka v. Zbiór sąsiadów wierzchołka u w grafie G oznaczamy przez $N_G(u)$. Liczbę krawędzi incydentnych z wierzchołkiem u nazywamy *stopniem* u w grafie G i oznaczamy przez $deg_G(u)$. Będziemy używać oznaczeń $V(G)$ i $E(G)$ na zbiory wierzchołków i krawędzi grafu G. Przyjmujemy także konwencję, że $n = |V|$ oraz $m = |E|$, kiedy oznaczenia

te będą jednoznacznie określone. *Grafem pustym* nazwiemy graf o pustym zbiorze wierzchołków.

Grafy Zorientowane

Graf $G = (V, E)$, w którym krawędziom przypisano kierunki nazywamy *grafem zorientowanym* lub krótko *digrafem*. Zbiór krawędzi E grafu zorientowanego zawiera uporządkowane pary wierzchołków, tzn.

$$E \subseteq \{(u, v) : u, v \in V\}.$$

Jeśli $e = (u, v)$ jest krawędzią w grafie zorientowanym $G = (V, E)$, to mówimy, że krawędź e *wychodzi* z u i *wchodzi* do v.

Ścieżką (*zorientowaną*) π *długości* k z wierzchołka v do wierzchołka w w grafie (zorientowanym) $G = (V, E)$ nazywamy ciąg wierzchołków $\pi = (v_0, v_1, \ldots, v_k)$ taki, że $v = v_0$, $w = v_k$ oraz $(v_{i-1}, v_i) \in E$, dla każdego $i = 1, 2, \ldots, k$. Długość ścieżki π jest liczbą jej krawędzi i oznaczamy ją przez $|\pi|$. Ścieżkę, w której $v_0 = v_k$ nazywamy *zamkniętą ścieżką* lub *pseudocyklem*. Zamkniętą ścieżkę w której dodatkowo wierzchołki $v_0, v_1, \ldots, v_{k-1}$ są różne nazywamy *cyklem* (*zorientowanym*). *Pokryciem cyklami* grafu G nazywamy ciąg rozłącznych cykli których całkowita liczba krawędzi jest równa $|V|$.

Graf, którego każdy wierzchołek jest połączony bezpośrednio krawędzią z każdym innym wierzchołkiem nazywamy *grafem pełnym*. Graf pełny o n wierzchołkach oznaczamy przez K_n.

Mówimy, że graf $G' = (V', E')$ jest *podgrafem* grafu $G = (V, E)$, jeśli $V' \subseteq V$ oraz $E' \subseteq E$. Dla danego zbioru $V' \subseteq V$ podgrafem G' grafu $G = (V, E)$ *indukowanym* przez V' jest graf $G' = (V', E')$, gdzie

$$E' = \{(u, v) \in E : u, v \in V'\}.$$

Grafem ważonym nazywamy graf $G = (V, E)$ wraz z funkcją, która każdej krawędzi z E przyporządkowuje pewną liczbę, nazywaną *wagą krawędzi*.

2.1.2 Grafy Planarne

Rysunkiem grafu w płaszczyznę nazywamy dowolne różnowartościowe odwzorowanie wierzchołków grafu w punkty płaszczyzny wraz z odwzorowaniem każdej krawędzi $\{u, v\}$ w łamaną zwyczajną na płaszczyźnie o końcach w obrazach punktów u i v. Jeżeli obrazy dwóch krawędzi przecinają się w punkcie x, który nie jest obrazem

wspólnego końca tych krawędzi to punkt x nazywamy *przecięciem*. Rysunek grafu na płaszczyźnie nie zawierający przecięć, nazywamy *rysunkiem płaskim*. Jeżeli istnieje rysunek płaski grafu G, to G nazywamy *grafem planarnym*. Zdarza się, że rysunek płaski grafu planarnego nazywa się także *grafem płaskim*. *Ściany* grafu płaskiego to maksymalne spójne podzbiory płaszczyzny, rozłączne z rysunkiem płaskim grafu.

Zachodzi następujące twierdzenie:

Twierdzenie 2.1.1. *Niech $G = (V, E)$ będzie grafem planarnym, $n = |V|$ oraz $m = |E|$. Jeśli $n \geq 3$ wówczas*

$$m \leq 3n - 6.$$

2.1.3 Skojarzenia w Grafach

Skojarzeniem $\mathcal{M}$ w grafie $G = (V, E)$ nazywamy zbiór krawędzi taki, że żadne dwie krawędzie z $\mathcal{M}$ nie mają wspólnego końca. Mówimy, że wierzchołek v jest *skojarzony* przez $\mathcal{M}$ jeśli $\mathcal{M}$ zawiera krawędź incydentną z v. Skojarzenie $\mathcal{M}$ nazywamy *doskonałym* jeśli kojarzy wszystkie wierzchołki grafu G.

2.2 Problem Plecakowy

Problem plecakowy (ang. *knapsack problem*) jest jednym z najczęściej poruszanych problemów optymalizacyjnych. Nazwa zagadnienia pochodzi od maksymalizacyjnego problemu wyboru przedmiotów, tak by ich sumaryczna wartość była jak największa i jednocześnie mieściły się w plecaku. Przy podanym zbiorze elementów o podanej wadze i wartości, należy wybrać taki podzbiór by suma wartości była możliwie jak największa, a suma wag była nie większa od danej pojemności plecaka.

2.3 Algebra Liniowa

2.3.1 Podstawowe Definicje

W całej pracy posługujemy się terminologią *ciał* i *pierścieni* (*przemiennych*). W ciele mamy określone dwa działania dwuargumentowe, dodawanie i mnożenie, i dwie stałe 0 i 1 ($0 \neq 1$), przy czym spełnione są naturalne prawa łączności, przemienności dodawania i mnożenia, rozdzielności mnożenia względem dodawania, neutralności 0 i 1 względem odpowiednio dodawania i mnożenia, istnienia elementu przeciwnego (względem dodawania) dla każdego elementu i elementu odwrotnego (względem mnożenia)

dla każdego elementu różnego od 0. W pierścieniu natomiast nie wymagamy istnienia elementów odwrotnych, pozostałe aksjomaty są analogiczne jak dla ciał.

Przez $\mathbb{R}$ ($\mathbb{Q}$, $\mathbb{C}$) oznaczamy ciało liczb rzeczywistych (wymiernych, zespolonych), przez $\mathbb{Z}$ oznaczamy pierścień liczb całkowitych a przez $\mathbb{N}$ zbiór liczb naturalnych (z 0). Dla dowolnego pierścienia R, przez $R[u]$ oznaczamy pierścień wielomianów o współczynnikach w R zmiennej u, przez $R[[u]]$ pierścień formalnych szeregów potęgowych, elementy $R[[u]]$ są postaci

$$a_0 + a_1 u + a_2 u^2 + \ldots + a_i u^i + \ldots,$$

gdzie $a_0, a_1, a_2, \ldots, a_i, \ldots \in R$. $R^n[[u]]$ oznacza pierścień formalnych szeregów potęgowych modulo u^{n+1}.

2.3.2 Algebra Macierzy

Dla dowolnego pierścienia R, $R^{m\times n}$ oznacza zbiór wszystkich macierzy o m wierszach i n kolumnach o współczynnikach w R. Dla macierzy $A \in R^{m\times n}$ zakładamy, że zbiór indeksów wierszy jest równy $\{1, 2, \ldots, m\}$ a zbiór indeksów kolumn $\{1, 2, \ldots, n\}$. Współczynnik znajdujący się w wierszu o numerze i i kolumnie o numerze j w macierzy A oznaczamy przez $A[i, j]$.

Niech $A \in R^{m\times n}$. Dla $R \subseteq \{1, 2, \ldots, m\}$ oraz $C \subseteq \{1, 2, \ldots, n\}$, przez $A_{R,C}$ oznaczamy podmacierz macierzy A składającą się z jej wierszy o indeksach zawartych w R oraz kolumn o indeksach z C a przez A_R podmacierz macierzy A zawierającą wiersze z R, to znaczy $A_R = A_{R,\{1,2,\ldots,n\}}$. Dla macierzy kwadratowej $A \in R^{n\times n}$ przez $A^{i,j}$ oznaczamy macierz rozmiaru $(n-1)\times(n-1)$ powstałą z macierzy A przez usunięcie wiersza o numerze i i kolumny o numerze j. Przez A^T oznaczamy *transpozycję* macierzy A, gdzie $A^T[i, j] = A[j, i]$. Przez A_h oznaczamy macierz powstałą z macierzy A poprzez wyzerowanie wszystkich wierszy i kolumn o indeksach nie większych niż h, to znaczy $A_h[i, j] = A[i, j]$, jeśli $i > h$ oraz $j > h$, 0 w przeciwnym wypadku, natomiast przez a_h (a^h) wiersz (kolumnę) macierzy A o numerze h z wyzerowanymi h pierwszymi współrzędnymi.

Niech A będzie macierzą kwadratową o boku n. *Wyznacznik* macierzy A, $det(A)$, zdefiniowany jest jako

$$det(A) = (-1)^n \cdot \sum_{\sigma} sgn(\sigma) \cdot weight(\sigma, A), \tag{2.1}$$

gdzie sumowanie odbywa się po wszystkich permutacjach σ grupy permutacji zbioru $\{1, 2, \ldots, n\}$, S_n, *znak* permutacji σ, $sgn(\sigma)$, równy jest $(-1)^k$, gdzie k jest liczbą

cykli w rozkładzie na cykle permutacji σ, natomiast *waga* permutacji σ wynosi

$$weight(\sigma, A) = A[1, \sigma(1)] \cdot A[2, \sigma(2)] \cdot \ldots \cdot A[n, \sigma(n)].$$

Permanent macierzy A, $perm(A)$, definiujemy jako

$$perm(A) = \sum_{\sigma} weight(\sigma, A). \tag{2.2}$$

Zauważmy, że jedyna różnica pomiędzy definicją permanentu a definicją wyznacznika jest taka, że w permanencie wszystkie składniki sumy mają ten sam znak. Problem obliczania wartości permanentu jest jednak nieporównywalnie trudniejszy od problemu obliczania wyznacznika.

Z wyznacznikiem blisko związane jest pojęcie *wielomianu charakterystycznego* (w rzeczywistości większość algorytmów obliczania wyznacznika oblicza od razu wielomian charakterystyczny, z którego wyznacznik już jest bardzo łatwo odczytać). Wielomianem charakterystycznym macierzy A nazywamy wielomian

$$\Phi_A(\lambda) = det(\lambda \cdot I_n - A) = c_0 \cdot \lambda^n + c_1 \cdot \lambda^{n-1} + \ldots + c_{n-1} \cdot \lambda + c_n, \tag{2.3}$$

gdzie I_n oznacza macierz identycznościową rozmiaru $n \times n$. Mamy

$$c_n = (-1)^n \cdot det(A),$$

natomiast $c_1 = -tr(A)$, gdzie $tr(A)$ oznacza *ślad macierzy* A

$$tr(A) = A[1,1] + A[2,2] + \ldots + A[n,n].$$

Pierwiastki wielomianu 2.3 nazywamy *wartościami własnymi* macierzy A. Oznaczamy je zazwyczaj przez $\lambda_1, \lambda_2, \ldots, \lambda_n$. Mamy

$$\lambda_1 + \lambda_2 + \ldots + \lambda_n = tr(A),$$

$$\lambda_1 \cdot \lambda_2 \cdot \ldots \cdot \lambda_n = det(A).$$

Macierzą dopełnień algebraicznych macierzy A, $adj(A)$, nazywamy macierz rozmiaru $n \times n$, gdzie

$$adj(A)[i,j] = (-1)^{i+j} det(A^{i,j}).$$

2.3.3 Macierze Skośno-Symetryczne

Macierz A rozmiaru $n \times n$ nazywamy macierzą *skośno-symetryczną* jeśli $A = -A^T$, to znaczy $A[i,j] = -A[j,i]$, dla wszystkich $i, j \in \{1, 2, \ldots, n\}$.

Wyznacznik macierzy skośno-symetrycznej jest pierwiastkiem innego wyrażenia, które nazywamy *Pfaffianem* [30]. Formalnie, Pfaffian skośno-symetrycznej macierzy A o parzystej liczbie n wierszy i kolumn zdefiniowany jest jako

$$Pf(A) = \sum_{\mathcal{M}} sgn(\mathcal{M}) \cdot weight(\mathcal{M}, A), \tag{2.4}$$

gdzie sumowanie odbywa się po wszystkich skojarzeniach doskonałych $\mathcal{M}$ grafu pełnego K_n. Skojarzenie doskonałe $\mathcal{M}$ zapisujemy jako podział zbioru $\{1, 2, \ldots, n\}$ na $m = n/2$ nieuporządkowanych par

$$\mathcal{M} = \{\{i_1, j_1\}, \{i_2, j_2\}, \ldots, \{i_m, j_m\}\},$$

gdzie $i_k < j_k$ dla każdego $k = 1, 2, \ldots, m$ oraz $i_1 < i_2 < \ldots < i_m$. *Znak* skojarzenia $\mathcal{M}$, $sgn(\mathcal{M})$, definiujemy jako znak permutacji

$$\sigma_{\mathcal{M}} = \begin{pmatrix} 1 & 2 & 3 & 4 & \cdots & n-1 & n \\ i_1 & j_1 & i_2 & j_2 & \cdots & i_m & j_m \end{pmatrix}.$$

Waga skojarzenia $\mathcal{M}$ jest równa

$$weight(\mathcal{M}, A) = A[i_1, j_1] \cdot A[i_2, j_2] \cdot \ldots \cdot A[i_m, j_m].$$

Przykład 2.3.1. Weźmy macierz

$$A = \begin{bmatrix} 0 & a \\ -a & 0 \end{bmatrix}$$

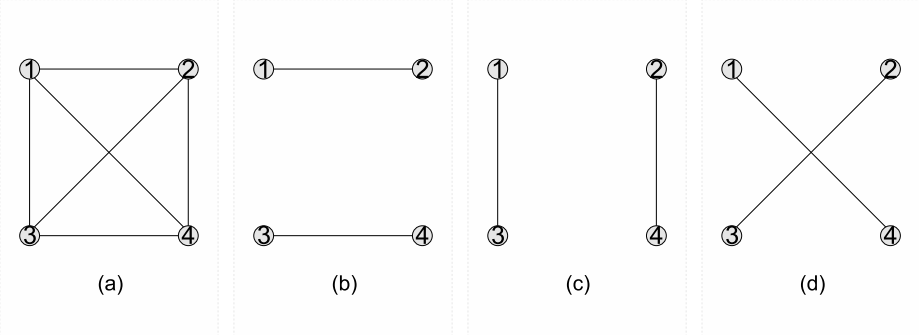
Graf K_2 ma jedno skojarzenie doskonałe $\mathcal{M} = \{\{1, 2\}\}$, złożone z jego jedynej krawędzi. Mamy więc $Pf(A) = a$.

Weźmy teraz macierz

$$A = \begin{bmatrix} 0 & a & b & e \\ -a & 0 & c & 0 \\ -b & -c & 0 & d \\ -e & 0 & -d & 0 \end{bmatrix}$$

Graf K_4 ma 3 skojarzenia doskonałe

$$\{\{1, 2\}, \{3, 4\}\},\ \{\{1, 3\}, \{2, 4\}\},\ \{\{1, 4\}, \{2, 3\}\},$$



Rysunek 2.1: Graf pełny K_4 – (a) oraz jego skojarzenia doskonałe – (b), (c), (d).

pokazane na rysunku 2.1. Mamy

$$sgn\begin{pmatrix}1 & 2 & 3 & 4\\ 1 & 2 & 3 & 4\end{pmatrix} = 1,$$

$$sgn\begin{pmatrix}1 & 2 & 3 & 4\\ 1 & 3 & 2 & 4\end{pmatrix} = -1,$$

$$sgn\begin{pmatrix}1 & 2 & 3 & 4\\ 1 & 4 & 2 & 3\end{pmatrix} = 1.$$

Zatem

$$Pf(A) = a \cdot d - b \cdot 0 + e \cdot c = a \cdot d + e \cdot c.$$

2.4 Mnożenie Macierzy

W naszych algorytmach używać będziemy szybkiego mnożenia macierzy prostokątnych. Niech $\omega(a, b, c)$ oznacza wykładnik w algorytmie mnożenia macierzy rozmiaru $n^a \times n^b$ przez macierz $n^b \times n^c$. Dla $a = b = c = 1$ dostajemy wykładnik w algorytmie mnożenia macierzy kwadratowych $\omega = \omega(1, 1, 1)$.

Strassen [44] pokazał, że macierze kwadratowe mogą zostać przemnożone w czasie $o(n^3)$. To ograniczenie było wielokrotnie poprawiane. Aktualnie najlepszy wynik został podany przez Coppersmitha i Winograda [11], $\omega \leq 2.3755$. Ich rezultat został później rozszerzony przez Coppersmitha [10], który pokazał, że jest możliwe przemnożenie macierzy rozmiaru $n \times n^{0.294}$ przez macierz $n^{0.294} \times n$ przy użyciu jedynie $\tilde{O}(n^2)$ operacji arytmetycznych.

Notacja $f(n) = \tilde{O}(g(n))$ jest skróconym zapisem dla $f(n) = O(g(n)log^k g(n))$ dla pewnej stałej k.

Niech

$$\zeta = sup\{a : \omega(1, a, 1) = 2 + o(1)\}.$$

Wartość ζ jest nie większa niż 0.294.

Przez połączenie ograniczeń na ζ i ω, Huang i Pan [23] udowodnili następujące ograniczenie na $\omega(a, b, c)$.

Lemat 2.4.1 (Huang i Pan [23]). *Niech* $1 \geq a \geq b$. *Wówczas*

$$\omega(1, a, b) \leq \hat{\omega}(1, a, b) = \begin{cases} 1 + a & : \ 0 \leq b \leq \zeta a, \\ 1 + \eta a + \theta b & : \ \zeta a \leq b \leq 1, \end{cases}$$

gdzie $\eta = \frac{1+\zeta-\zeta\omega}{1-\zeta} \leq 0.844$ *oraz* $\theta = \frac{\omega-2}{1-\zeta} \leq 0.533$.

Zauważmy, że $\eta + \theta = \omega - 1$.

Wniosek 2.4.1. *Dla dowolnego* $0 \leq a \leq 1$

$$\omega(1, a, a) \leq a(\omega - 1) + 1.$$

To ograniczenie można również udowodnić przez podział macierzy $n \times n^a$ na macierze rozmiaru $n^a \times n^a$ i użycie algorytmu Coppersmitha i Winograda do ich przemnożenia.

W dalszej części pracy będziemy używać skrótów algorytm MV dla algorytmu Mahajana i Vinaya obliczania wyznacznika oraz algorytm MSV dla algorytmu Mahajana, Subramanya i Vinaya obliczania Pfaffianu.

Rozdział 3

Wyznacznik

3.1 Wprowadzenie

Tło. Obliczanie wyznacznika jest jednym z klasycznych problemów algebry liniowej. Najbardziej znanym algorytmem obliczania wyznacznika jest eliminacja Gaussa. Wymaga ona $O(n^3)$ (lub $O(n^{2.38})$, jeśli użyty zostanie algorytm szybkiego mnożenia macierzy) operacji dodawania, odejmowania, mnożenia oraz *dzielenia*. Z drugiej strony, definicja wyznacznika jako sumy $n!$ iloczynów pokazuje, że może on zostać obliczony *bez* wykonywania operacji dzielenia.

Wyeliminowanie dzielenia jest istotnym czynnikiem podczas pracy z pierścieniem przemiennym, który nie jest ciałem, na przykład, kiedy elementy pierścienia są liczbami całkowitymi, wielomianami czy tez jeszcze bardziej skomplikowanymi wyrażeniami. Takie obliczenia pojawiają się w wielu kombinatorycznych problemach [27].

Strassen [45] podał ogólną metodę modyfikacji algorytmów używających dzielenia na algorytmy nie wykorzystujące tej operacji. Tak zmodyfikowany algorytm obliczania wyznacznika zaproponowany przez Strassena ma czas działania $O(n^5)$ (lub $O(n^{3.38})$, jeśli użyte zostaną algorytm szybkiego mnożenia macierzy oraz algorytm szybkiej transformaty Fouriera).

Algorytm kombinatoryczny. Algorytm zaproponowany przez Mahajana i Vinaya [34] oblicza wyznacznik w całkowicie nieklasyczny, kombinatoryczny sposób poprzez redukcję tego problemu do problemu sumowania ścieżek w pewnym acyklicznym grafie. Algorytm działa w czasie $O(n^4)$ a jego ważną zaletą jest to, że nie wykorzystuje on operacji dzielenia.

Nasze wyniki. W tym rozdziale przedstawimy nowe spojrzenie na algorytm Mahajana i Vinaya: relację z pseudo-wielomianowym algorytmem programowania dynamicznego dla problemu plecakowego. Główna faza algorytmu Mahajana i Vinaya może być zinterpretowana jako obliczenia algebraicznej wersji problemu plecakowego, co jest alternatywą dla podejścia opartego na teorii grafów użytego w oryginalnym algorytmie. Głównym rezultatem [50, 51] jest pokazanie jak zaimplementować algorytm Mahajana i Vinaya bez korzystania z operacji dzielenia w czasie $\tilde{O}(n^{3.03})$. Zaprezentowany przez nas algorytm posiada bardzo prostą implementację w czasie $O(n^{3.5})$ a jedynym nietrywialnym elementem algorytmu działającego w czasie $\tilde{O}(n^{3.03})$ jest algorytm szybkiego mnożenia macierzy.

3.2 Nowe Spojrzenie na Algorytm MV

3.2.1 Zmodyfikowany Problem Plecakowy

Wprowadzamy *zmodyfikowany problem plecakowy*: mamy przedmioty, z których każdy jest jednego z n możliwych typów, dla każdego typu (i) mamy m przedmiotów typu (i). Przedmiot $Item_{i,j}$ ma typ i, wagę j oraz wartość $A[i,j]$. Musimy wybrać podzbiór

$$\{Item_{i_1,j_1}, Item_{i_2,j_2}, \ldots, Item_{i_k,j_k}\}$$

zbioru wszystkich przedmiotów, przy czym każdy typ powinien być reprezentowany co najwyżej jeden raz. Całkowita wartość wybranego podzbioru powinna być maksymalna, natomiast całkowita jego waga powinna być równa n.

Innymi słowy wybieramy z każdego wiersza macierzy A co najwyżej jeden element w taki sposób, aby suma $j_1 + j_2 + \ldots + j_k$ indeksów wybranych kolumn była równa n oraz suma $A[i_1, j_1] + A[i_2, j_2] + \ldots + A[i_k, j_k]$ wartości wybranych elementów była maksymalna.

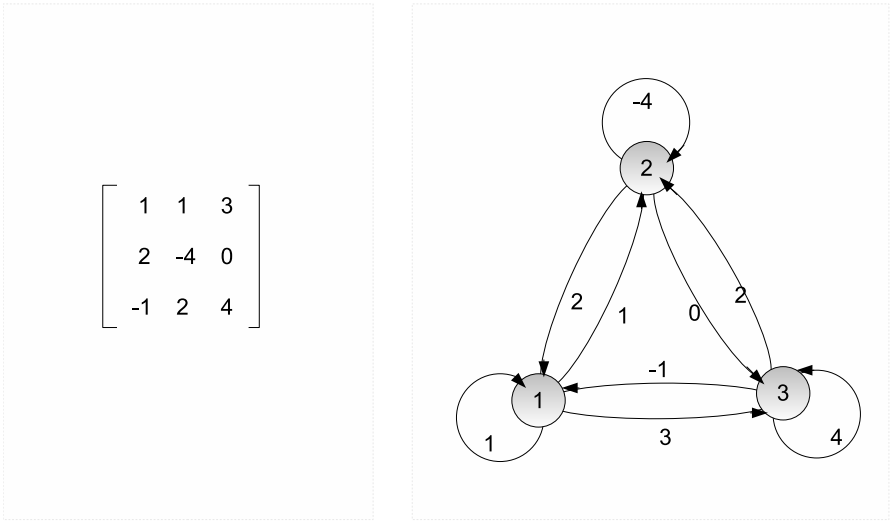
3.2.2 Algebraiczny Problem Plecakowy

Zmodyfikowany problem plecakowy możemy sformułować w bardziej algebraicznych terminach. Załóżmy, że mamy dwie operacje $\oplus$, $\otimes$ w pierścieniu S. Dla prostokątnej macierzy A rozmiaru $n \times m$ oznaczamy przez $\mathcal{K}_t(A)$ sumę (za względu na operację $\oplus$) wszystkich iloczynów postaci

$$A[i_1, j_1] \otimes A[i_2, j_2] \otimes \ldots \otimes A[i_k, j_k],$$

gdzie

$$1 \leq i_1 < i_2 < \ldots < i_k \leq n,\ 1 \leq j_1, j_2, \ldots, j_k \leq m,\ j_1 + j_2 + \ldots + j_k \ = \ t.$$



Rysunek 3.1: Macierz A oraz odpowiadający jej graf G_A.

Jeśli $\oplus = \max$ i $\otimes = +$, wówczas $\mathcal{K}_t(A)$ jest równe maksymalnej wartości spośród wartości wszystkich podzbiorów zbioru przedmiotów spełniających ograniczenia, że całkowita suma wag ich elementów jest równa t oraz wartości przedmiotów zadane są przez macierz A. A to jest dokładnie zmodyfikowany problem plecakowy.

Jeśli A jest macierzą o współczynnikach całkowitych rozmiaru $n \times n$ oraz operacje $\oplus$, $\otimes$ to klasyczne operacje arytmetyczne $+$, $\cdot$ wówczas oznaczamy

$$Knapsack(A) = \mathcal{K}_n(A).$$

W naszej wersji algorytmu MV [50, 51] mamy dwie fazy, pierwsza odpowiada wariacji domknięcia przechodniego i nazywana jest w pracy skośnym domknięciem. Druga faza odpowiada algebraicznemu uogólnieniu problemu plecakowego.

3.2.3 Skośne Domknięcie Macierzy

Niech A będzie macierzą o współczynnikach całkowitych rozmiaru $n \times n$. Możemy myśleć o macierzy A jako o zorientowanym pełnym grafie G_A z wagami na krawędziach w którym wierzchołki są etykietowane liczbami ze zbioru $\{1, 2, \ldots, n\}$ a krawędź (i, j) ma wagę $A[i, j]$, $i, j = 1, 2, \ldots, n$.

Niech $\pi = (i_0, i_1, \ldots, i_k)$ będzie ścieżką długości k w grafie G_A. *Wagą* ścieżki π nazywamy iloczyn wag tworzących ją krawędzi

$$weight(\pi, A) \;=\; A[i_0, i_1] \cdot A[i_1, i_2] \cdot \ldots \cdot A[i_{k-1}, i_k].$$

Powiemy, że π jest *pseudocyklem* z *liderem* h jeśli $i_0 = i_k = h$ oraz $i_t > h$ dla każdego $0 < t < k$. Lidera pseudocyklu π oznaczamy przez $h(\pi)$. Przez $\Pi_{h,k}$ oznaczamy zbiór wszystkich pseudocykli długości k z liderem h w grafie G_A a przez $\hat{A}[h,k]$ sumę wag wszystkich pseudocykli ze zbioru $\Pi_{h,k}$

$$\hat{A}[h,k] \ = \sum_{\pi \in \Pi_{h,k}} weight(\pi, A).$$

Tak zdefiniowaną macierz $\hat{A}$ rozmiaru $n \times n$ nazywamy *skośnym domknięciem* macierzy A i oznaczamy przez $SkewClosure(A)$.

3.2.4 Ciągi Pseudocykli i Twierdzenie Mahajana-Vinaya

Ciągiem pseudocykli $\mathcal{C}$ nazywamy uporządkowany ciąg pseudocykli

$$\mathcal{C} = (C_1, C_2, \ldots, C_k),$$

których całkowita *długość*, $|\mathcal{C}|$, równa sumie długości tworzących go pseudocykli, wynosi n i których liderzy są w ściśle rosnącym porządku.

Wagą (ze względu na macierz A) ciągu pseudocykli $\mathcal{C}$ nazywamy iloczyn wag tworzących go pseudocykli

$$weight(\mathcal{C}, A) \ = \prod_{i=1}^{k} weight(C_i, A).$$

Znak ciągu pseudocykli $\mathcal{C}$, $sgn(\mathcal{C})$, jest równy $(-1)^k$, gdzie k jest liczbą pseudocykli w $\mathcal{C}$.

Zauważmy, że suma $(-1)^n \cdot sgn(\mathcal{C}) \cdot weight(\mathcal{C}, A)$ przebiegająca po tylko tych ciągach pseudocykli, które są jednocześnie pokryciami cyklami grafu G_A jest równa dokładnie wyznacznikowi macierzy A.

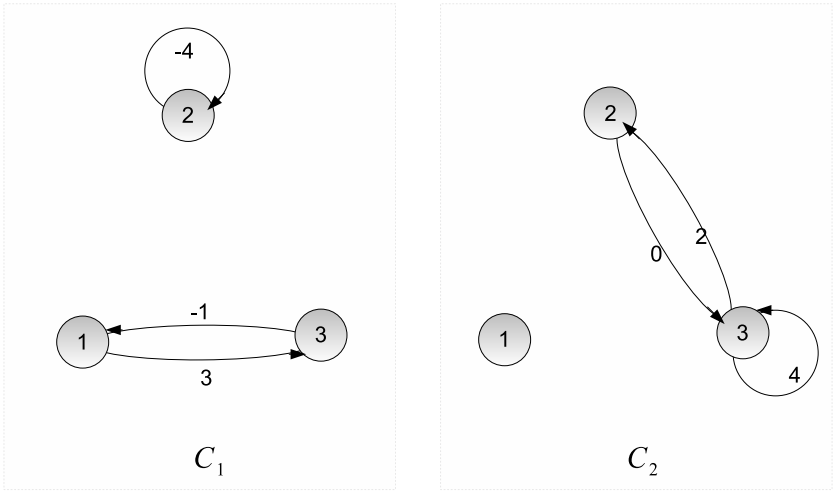
Mahajan i Vinay [34] pokazali, że w istocie możemy zastąpić sumowanie po zbiorze pokryć cyklami grafu przez sumowanie po znacznie większej klasie ciągów pseudocykli.

Twierdzenie 3.2.1 (Mahajan-Vinay)**.**

$$det(A) = (-1)^n \cdot \sum_{\mathcal{C}} sgn(\mathcal{C}) \cdot weight(\mathcal{C}, A),$$

gdzie sumowanie przebiega po wszystkich *ciągach pseudocykli* $\mathcal{C}$*.*

Dowód. Pokażemy, że wszystkie „złe" ciągi pseudocykli, które nie są pokryciami cyklami, wzajemnie się redukują, ponieważ odpowiadające im wagi występują dokładnie tyle samo razy z dodatnim jak i ujemnym znakiem. Zrobimy to parując złe

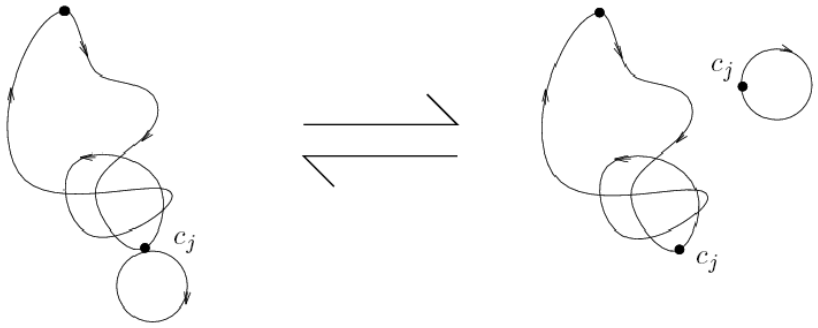


Rysunek 3.2: Dwa ciągi pseudocykli $\mathcal{C}_1 = ((1,3,1),(2,2))$ oraz $\mathcal{C}_2 = ((2,3,3,2))$ w grafie G_A przedstawionym na rysunku 3.1, $weight(\mathcal{C}_1) = 12$, $sgn(\mathcal{C}_1) = 1$, $weight(\mathcal{C}_2) = 0$ oraz $sgn(\mathcal{C}_2) = -1$. Ciąg pseudocykli $\mathcal{C}_1$ jest jednocześnie pokryciem cyklami grafu G_A, ale już ciąg pseudocykli $\mathcal{C}_2$ nie jest pokryciem cyklami: wierzchołek 1 nie występuje w żadnym z pseudocykli składających się na ciąg $\mathcal{C}_2$ z kolei wierzchołek 3 pojawia się w ciągu $\mathcal{C}_2$ dwukrotnie.

ciągi pseudocykli w taki sposób, by wagi skojarzonych ciągów były takie same i aby różniły się one tylko znakiem jak w lemacie 3.2.1. Wykorzystana tu technika dowodzenia jest typowa dla wielu kombinatorycznych dowodów twierdzeń algebry liniowej [38, 24, 42, 39, 46, 57]. Poniższy lemat został podany przez Valianta [52] a jego szczegółowy dowód można znaleźć w pracach [35, 36].

Lemat 3.2.1. *Niech $\mathcal{C}$ będzie ciągiem pseudocykli zawierającym powtórzony element. Wówczas możemy znaleźć ciąg pseudocykli $\overline{\mathcal{C}}$ o tej samej wadze i przeciwnym znaku. Co więcej takie odwzorowanie pomiędzy $\mathcal{C}$ a $\overline{\mathcal{C}}$ jest odwzorowaniem wzajemnie jednoznacznym, to znaczy jeśli skonstruujemy ciąg pseudocykli dla $\overline{\mathcal{C}}$ otrzymamy wyjściowy ciąg $\mathcal{C}$.*

Dowód. Niech $\mathcal{C} = (C_1, C_2, \ldots, C_k)$. Dodajemy kolejne pseudocykle $C_k, C_{k-1} \ldots$ dopóki nie nastąpi powtórzenie jakiegoś elementu. Przypuśćmy, że $C_{m+1}, C_{m+2}, \ldots, C_k$ jest ciągiem rozłącznych cykli ale ciąg $C_m, C_{m+1} \ldots, C_k$ zawiera już powtórzony element, gdzie $1 \leq m \leq k$. Niech $C_m = (c_0, c_1, \ldots, c_i)$ i niech c_j będzie pierwszym elementem na pseudocyklu C_m, który jest albo (1) równy poprzedzającemu go elementowi c_l na C_m $(1 \leq l < j)$ albo (2) równy elementowi występującemu w cyklu C_s



Rysunek 3.3: Parowanie ciągów pseudocykli o przeciwnych znakach.

będącym jednym z cykli $C_{m+1}, C_{m+2}, \ldots, C_k$. Dokładnie jeden z tych dwóch przypadków musi mieć miejsce.

W przypadku (1) usuwamy cykl $(c_l = c_j, c_{l+1}, \ldots, c_{j-1})$ z pseudocyklu C_m i wkładamy go jako osobny cykl w $\mathcal{C}$ na właściwą pozycję tak, aby zachowane było ściśle rosnące uporządkowanie liderów.

W przypadku (2) przenosimy cykl C_s do pseudocyklu C_m tuż za miejscem pojawienia się elementu c_j.

Łatwo sprawdzić, że operacje wykonywane w przypadkach (1) i (2) są operacjami wzajemnie odwrotnymi. Zmieniają one liczbę pseudocykli o jeden, co powoduje zmianę znaku na przeciwny. To dowodzi prawdziwości lematu. □

Teraz już wprost z lematu otrzymujemy, że całkowity wkład w sumę składników nie będących pokryciami cyklami jest równy zero, co kończy dowód twierdzenia 3.2.1. □

3.2.5 Algorytm MV

Pomysłem na efektywny algorytm jest konstrukcja ciągów pseudocykli przyrostowo, krawędź po krawędzi. Wykorzystujemy programowanie dynamiczne do systematycznego wyznaczania następujących wielkości:

> $[l, c, c_0, s]$ jest sumą wag wszystkich częściowych ciągów pseudocykli długości l, kończących się w wierzchołku c, których liderem aktualnie budowanego pseudocyklu jest c_0, a znak $s = \pm 1$ określa parzystość liczby dotychczas zakończonych pseudocykli.

Częściowy ciąg pseudocykli jest początkowym fragmentem ciągu pseudocykli. Składa się on z pewnej liczby pseudocykli oraz jednego niekompletnego pseudocyklu. Aby uzupełnić ten ciąg pseudocykli musimy jedynie znać lidera oraz ostatni wierzchołek na jego niekompletnym pseudocyklu. Musimy również znać parzystość całkowitej liczby pseudocykli, pamiętamy więc wartość $(-1)^k$, gdzie k jest liczbą dotychczas skompletowanych pseudocykli.

Nasz algorytm programowania dynamicznego można w czytelny sposób opisać za pomocą dynamicznie konstruowanego grafu. Nasz graf ma $O(n^3)$ wierzchołków odpowiadających wielkością $[l, c, c_0, s]$ dla $1 \leq l \leq n, 1 \leq c_0 \leq c \leq n$ oraz $s = \pm 1$.

Częściowy ciąg pseudocykli może być rozbudowany albo przez przedłużenie aktualnego (niekompletnego) pseudocyklu lub przez zakończenie aktualnego pseudocyklu i rozpoczęcie nowego.

Odpowiednio, nasz dynamicznie konstruowany graf posiada dwa rodzaje wychodzących krawędzi z wierzchołka $[l, c, c_0, s]$: krawędzie do wierzchołków $[l + 1, c', c_0, s]$ o koszcie $A[c, c']$ dla wszystkich $c' > c_0$ oraz krawędzie o koszcie $A[c, c_0]$ do wierzchołków $[l + 1, c'_0, c'_0, -s]$, dla wszystkich $c'_0 > c_0$.

Drugi typ krawędzi odpowiada rozpoczęciu nowego pseudocyklu z liderem c'_0 i nie zawierającego jeszcze żadnej krawędzi. Dodajemy ponadto do naszego grafu pomocnicze wierzchołki startowe postaci $[0, c_0, c_0, 1]$, dla $1 \leq c_0 \leq n$, z których wychodzą krawędzie do wierzchołków $[1, c, c_0, 1]$, dla wszystkich $c > c_0$, o koszcie $A[c_0, c]$ oraz dwa wierzchołki końcowe $[n, n+1, n+1, s]$, dla $s = \pm 1$, do których wchodzą krawędzie z wierzchołków $[n - 1, c, c_0, -s]$, dla wszystkich $1 \leq c_0 \leq c \leq n$, o koszcie $A[c, c_0]$.

Suma wag wszystkich ścieżek z wierzchołków startowych do wierzchołków końcowych liczonych z odpowiednim znakiem w zależności od s jest równa wyznacznikowi macierzy A.

Suma wag wszystkich ścieżek w tym acyklicznym grafie może być obliczona w czasie proporcjonalnym do liczby krawędzi tego grafu, która jest równa $O(n^4)$: mamy $O(n)$ krawędzi wychodzących z każdego spośród $O(n^3)$ wierzchołków.

3.2.6 Nasza Wersja Algorytmu MV

Naszą wersję algorytmu MV [50, 51] możemy sformułować w następujący sposób.

Algorytm 3.2.1. *ComputeDeterminant*(A)
$\hat{A} := SkewClosure(A)$;
return $(-1)^n \cdot Knapsack(-\hat{A})$;
end;

Jak już widzieliśmy Mahajan i Vinay pokazali, że wyznacznik macierzy A można zdefiniować w terminach sum wag ścieżek w pewnym zorientowanym acyklicznym grafie. A oto sformułowanie algebraiczne twierdzenia Mahajana i Vinaya.

Twierdzenie 3.2.2. *Algorytm ComputeDeterminant*(A) *jest poprawny:*

$$det(A) = (-1)^n \cdot Knapsack(-SkewClosure(A)). \tag{3.1}$$

Dowód. Grupujemy ciągi pseudocykli bazując na liderach oraz na długościach poszczególnych pseudocykli. W ciągu pseudocykli $\mathcal{C}$, jeśli długość pseudocyklu C z liderem h jest równa l, wówczas dowolny pseudocykl z liderem h i długości l może zastąpić C w $\mathcal{C}$ wciąż dając ciąg pseudocykli.

Ponieważ $\hat{A}[h,l]$ równe jest całkowitej wadze wszystkich pseudocykli, które mają wierzchołek h jako lidera oraz długość l, otrzymujemy, że wartość $(-1)^n \cdot det(A)$ jest sumą wszystkich iloczynów postaci

$$(-1)^k \cdot \hat{A}[h_1,l_1] \cdot \hat{A}[h_2,l_2] \cdot \ldots \cdot \hat{A}[h_k,l_k],$$

gdzie

$$1 \leq h_1 < h_2 < \ldots < h_k \leq n,\ 1 \leq l_1, l_2, \ldots, l_k \leq n,\ l_1 + l_2 + \ldots + l_k = n,$$

a to jest dokładnie $Knapsack(-\hat{A}) = Knapsack(-SkewClosure(A))$. □

3.2.7 Obliczanie $Knapsack(A)$

Niech $A_{[k]}$ będzie podmacierzą macierzy A złożoną z jej k pierwszych wierszy. Klasyczny algorytm wyznaczania $\mathcal{K}_t(A)$ jest algorytmem programowania dynamicznego i używa tablicy $Table[i,j] = \mathcal{K}_j(A_{[i]})$.

Inicjujemy początkowe wartości $Table[1,j] = A[1,j]$, dla wszystkich $1 \leq j \leq n$, a pozostałe elementy tablicy $Table[i,j]$, dla $2 \leq i \leq n$ oraz $1 \leq j \leq n$, wyznaczamy zgodnie ze wzorem

$$Table[i,j] = \oplus_{0<p<j} \left(Table[i-1,p] \otimes A[i,j-p] \right) \oplus Table[i-1,j] \oplus A[i,j].$$

W konsekwencji otrzymujemy:

Lemat 3.2.2. *Jeśli A jest macierzą rozmiaru $n \times n$, wówczas wszystkie wartości $\mathcal{K}_1(A), \mathcal{K}_2(A), \ldots, \mathcal{K}_n(A) = Knapsack(A)$ możemy wyznaczyć w czasie $O(n^3)$ korzystając jedynie z operacji dodawania oraz mnożenia.*

3.3 Przyspieszenie Algorytmu Obliczania Wyznacznika

3.3.1 Obliczanie $SkewClosure(A)$

Najpierw opiszemy prostszą wersję naszego algorytmu obliczającego $SkewClosure(A)$ działającą w czasie $O(n^{3.5})$. Wykorzystujemy w niej klasyczny algorytm mnożenia macierzy, który do wymnożenia macierzy rozmiaru $s \times n$ przez macierz rozmiaru $n \times t$ wymaga czasu $O(s \cdot n \cdot t)$.

Musimy wyznaczyć współczynniki macierzy $\hat{A} = SkewClosure(A)$, gdzie $\hat{A}[h,k]$ jest równe sumie wag wszystkich pseudocykli długości k i z liderem h.

Zgodnie z przyjętymi przez nas oznaczeniami w rozdziale 2 przez A_h oznaczamy macierz powstałą z macierzy A poprzez wyzerowanie wszystkich wierszy i kolumn o indeksach nie większych niż h, odpowiadającą podgrafowi grafu G_A indukowanemu przez wierzchołki $h+1, h+2, \ldots, n$. Przez a_h (a^h) oznaczamy wiersz (kolumnę) macierzy A o numerze h z wyzerowanymi h pierwszymi współrzędnymi.

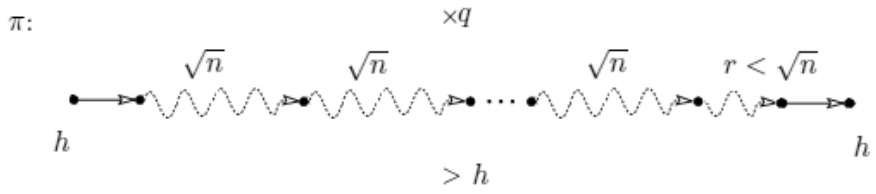
Mamy, że $\hat{A}[h,1] = A[h,h]$. Dla $k \geq 2$ pseudocykl długości k z liderem h startując z wierzchołka h musi najpierw odwiedzić pewien wierzchołek $i > h$ następnie wykonać drogę długości $k-2$ przez wierzchołki większe niż h do pewnego wierzchołka $j > h$ i wrócić do h. Ponieważ współczynnik znajdujący się w i-ym wierszu i j-ej kolumnie macierzy A_h^{k-2} równy jest dokładnie wadze wszystkich ścieżek z i do j długości $k-2$ przez wierzchołki większe niż h otrzymujemy, że

$$\hat{A}[h,k] = a_h \cdot A_h^{k-2} \cdot a^h, \tag{3.2}$$

dla $h = 1, 2, \ldots, n$ oraz $k = 2, 3, \ldots, n$.

Dla uproszczenia notacji załóżmy, że n jest kwadratem pewnej liczby naturalnej, tzn. $\lceil \sqrt{n} \rceil = \sqrt{n}$. Ścieżkę w której wszystkie wewnętrzne wierzchołki są większe niż h będziemy nazywać krócej *h-ścieżką*. Przez $A_h^k(i,j)$ oznaczamy sumę wag wszystkich h-ścieżek długości k z wierzchołka i do j.

Na początku nasz algorytm obliczał będzie $A_h^r(i,h)$ - sumę wag wszystkich h-ścieżek długości r z i do h oraz $A_{h-1}^r(h,j)$ - sumę wag wszystkich $(h-1)$-ścieżek długości r z h do j, dla $h = 1, 2, \ldots, n$, $r = 1, 2, \ldots, \sqrt{n}$, oraz $i, j = h, h+1, \ldots, n$.



Rysunek 3.4: Rozkład ścieżki π długości k z wierzchołka h do h przez wierzchołki większe niż h na dwie ścieżki długości 1, q ścieżek długości $\sqrt{n}$ oraz ścieżkę długości r, gdzie $k = q \cdot \sqrt{n} + r + 2$, $0 \leq r < \sqrt{n}$.

Następnie obliczamy sumę wag wszystkich h-ścieżek długości $\sqrt{n}$: odpowiadającą potędze $\sqrt{n}$ macierzy A_h, tzn. $A_h^{\sqrt{n}}$, $h = n, n-1, \ldots, 1$. Do obliczenia wartości $A_h^{\sqrt{n}}$ używać będziemy wyznaczonych wcześniej $A_{h+1}^{\sqrt{n}}$ oraz $A_h^k(i, h+1)$, $A_h^{\sqrt{n}-k}(h+1, j)$, dla $k = 0, 1, \ldots, \sqrt{n}$, $i, j = h+1, h+2, \ldots, n$. $A_n^{\sqrt{n}}$ jest macierzą zerową oraz

$$A_h^{\sqrt{n}}(i,j) \;=\; A_{h+1}^{\sqrt{n}}(i,j) \;+\; \sum_{k=0}^{\sqrt{n}} A_{h+1}^k(i, h+1) \cdot A_h^{\sqrt{n}-k}(h+1, j).$$

Poprawność użytego w tym kroku naszego algorytmu wzoru pokazujemy w dowodzie lematu 3.3.1.

W kroku 3, znając już wartości $A_h^{\sqrt{n}}$ obliczamy $A_h^{q\sqrt{n}+1}(h,i)$ - sumę wag wszystkich h-ścieżek długości $q \cdot \sqrt{n} + 1$ z h do i, dla $h = 1, 2, \ldots, n$, $i = h+1, h+2, \ldots, n$ oraz $q = 1, 2, \ldots, \sqrt{n}$. Każda h-ścieżka długości $q \cdot \sqrt{n} + 1$ z h do i jest złożeniem dwóch ścieżek: ścieżki długości $(q-1) \cdot \sqrt{n} + 1$ z wierzchołka h do pewnego wierzchołka $j > h$ oraz ścieżki długości $\sqrt{n}$ z wierzchołka j do i, tak więc

$$A_h^{q\sqrt{n}+1}(h,i) = \sum_{j=h+1}^{n} A_h^{(q-1)\sqrt{n}+1}(h,j) \cdot A_h^{\sqrt{n}}(j,i).$$

Na końcu wyznaczamy wartość $\hat{A}[h,k] = A_h^k(h,h)$, $h, k = 1, 2, \ldots, n$, korzystając z faktu, że każdą h-ścieżkę π długości $k = q \cdot \sqrt{n} + r + 2$, $q, r = 0, 1, \ldots, \sqrt{n} - 1$, z wierzchołka h do h możemy podzielić na dwie ścieżki: ścieżkę długości $q \cdot \sqrt{n} + 1$ z wierzchołka h do pewnego wierzchołka $i > h$ oraz ścieżkę długości $r+1$ z wierzchołka i do h, jak zostało to pokazane na rysunku 3.4. Suma wag ścieżek pierwszego rodzaju to $A_h^{q\sqrt{n}+1}(h,i)$ a drugiego $A_h^{r+1}(i,h)$, mamy więc

$$\hat{A}[h, q \cdot \sqrt{n} + r + 2] = \sum_{i=h+1}^{n} A_h^{q\sqrt{n}+1}(h,i) \cdot A_h^{r+1}(i,h).$$

SCHEMAT ALGORYTMU

KROK 1 oblicz $A_h^r(i,h)$, $A_{h-1}^r(h,j)$ **dla** $1 \le r \le \sqrt{n},\ 1 \le h \le i,j \le n$

KROK 2 niech $A_n^{\sqrt{n}}$ będzie macierzą zerową
oblicz $A_h^{\sqrt{n}}(i,j)$, $h = n-1, n-2, \ldots, 1$ **dla** $h < i,j \le n$
skorzystaj ze wzoru

$$A_h^{\sqrt{n}}(i,j) \;=\; A_{h+1}^{\sqrt{n}}(i,j) \;+\; \sum_{k=0}^{\sqrt{n}} A_{h+1}^{k}(i,h+1) \cdot A_h^{\sqrt{n}-k}(h+1,j)$$

KROK 3 oblicz $A_h^{q\sqrt{n}+1}(h,i)$ **dla** $1 \le q \le \sqrt{n},\ 1 \le h < i \le n$

KROK 4 oblicz $A_h^k(h,h) = \hat{A}[h,k]$ **dla** $1 \le h,k \le n$
skorzystaj ze wzoru

$$\hat{A}[h, q\cdot\sqrt{n} + r + 2] = \sum_{i=h+1}^{n} A_h^{q\sqrt{n}+1}(h,i) \cdot A_h^{r+1}(i,h)$$

W dalszej części wprowadzimy bardziej formalne oznaczenia dla obiektów występujących w schemacie algorytmu, w ten sposób skrócimy dokładny zapis algorytmu i będziemy lepiej przygotowani na dalsze przyspieszanie stosując metody algebraiczne.

Oznaczamy przez

$v_h^{[r]}[i]$ - sumę wag wszystkich h-ścieżek długości $r+1$ z wierzchołka i do h,

$$v_h^{[r]} = A_h^r \cdot a^h = A_h \cdot v_h^{[r-1]},$$

$w_h^{[r]}[i]$ - sumę wag wszystkich $(h-1)$-ścieżek długości $r+1$ z wierzchołka h do i,

$$w_h^{[r]} = a'_h \cdot A_{h-1}^r = w_h^{[r-1]} \cdot A_{h-1},$$

gdzie a'_h jest równy wierszowi macierzy A o numerze h z wyzerowanymi $h-1$ pierwszymi współrzędnymi,

$Z_h[i,j]$ - sumę wag wszystkich h-ścieżek długości $\sqrt{n}$ z wierzchołka i do j,

$$Z_h = A_h^{\sqrt{n}},$$

$u_h^{[q]}[i]$ - sumę wag wszystkich h-ścieżek długości $q \cdot \sqrt{n} + 1$ z wierzchołka h do i,

$$u_h^{[q]} = a_h \cdot A_h^{q \cdot \sqrt{n}} = u_h^{[q-1]} \cdot Z_h.$$

Przyjmijmy ponadto

$$v_h^{[-1]}[i] = w_h^{[-1]}[i] = 1,$$

jeśli $i = h$, 0 w przeciwnym wypadku, Z_n jest macierzą zerową.

Macierz $\hat{A}$ obliczamy w następujący sposób.

Algorytm 3.3.1. *SkewClosure*(A)

for $h = n-1, n-2, \ldots, 1$ **do**

KROK 1 **for** $r = 0, 1, \ldots, \sqrt{n}$ **do**

$v_h^{[r]} := A_h \cdot v_h^{[r-1]}$;

$w_h^{[r]} := w_h^{[r-1]} \cdot A_{h-1}$;

KROK 2 $Z_h := Z_{h+1} + \sum_{k=0}^{\sqrt{n}} (v_{h+1}^{[k-1]} \cdot w_{h+1}^{[\sqrt{n}-k-1]})$;

KROK 3 **for** $q = 1, 2, \ldots, \sqrt{n}$ **do**

$u_h^{[q]} := u_h^{[q-1]} \cdot Z_h$;

KROK 4 **for** $r = 0, 1, \ldots, \sqrt{n} - 1$ **do**

for $q = 0, 1, \ldots, \sqrt{n} - 1$ **do**

$\hat{A}[h, q \cdot \sqrt{n} + r + 2] := u_h^{[q]} \cdot v_h^{[r]}$;

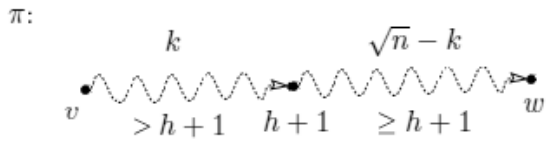
return $\hat{A}$;

end;

Lemat 3.3.1. *Algorytm SkewClosure*(A) *jest poprawny.*

Dowód. Przede wszystkim musimy pokazać poprawność kroku 2. Mamy, że $Z_h = A_h^{\sqrt{n}}$, i wartość $Z_h[v, w]$ jest równa sumie wag wszystkich ścieżek π długości $\sqrt{n}$ z wierzchołka v do w przez wierzchołki *większe* niż h.

Ścieżki długości $\sqrt{n}$ z wierzchołka v do w przez wierzchołki *większe* niż h możemy podzielić na dwa rodzaje ścieżek, mianowicie ścieżki długości $\sqrt{n}$ z wierzchołka v do w przez wierzchołki *większe* niż $h + 1$, których suma wag jest równa $Z_{h+1}[v, w]$, oraz ścieżki *zawierające* wierzchołek $h + 1$.



Rysunek 3.5: Jednoznaczny rozkład ścieżki π długości $\sqrt{n}$ z wierzchołka v do w przez wierzchołki większe niż h ale zawierającej wierzchołek $h+1$ na ścieżkę długości k z wierzchołka v do $h+1$ przez wierzchołki *większe* niż $h+1$ oraz ścieżkę długości $\sqrt{n}-k$ z wierzchołka $h+1$ do w przez wierzchołki *większe bądź równe* $h+1$.

Ścieżki drugiego rodzaju dzielimy ze względu na pozycję pierwszego wystąpienia w nich wierzchołka $h+1$, jak pokazano na rysunku 3.5. Ponieważ

$$v_{h+1}^{[k-1]}[v] = (A_{h+1}^{k-1} \cdot a^{h+1})[v]$$

jest równe sumie wag wszystkich ścieżek długości k z wierzchołka v do $h+1$ przez wierzchołki *większe* niż $h+1$ oraz

$$w_{h+1}^{[\sqrt{n}-k-1]}[w] = (a'_{h+1} \cdot A_h^{\sqrt{n}-k-1})[w]$$

jest równe sumie wag wszystkich ścieżek długości $\sqrt{n}-k$ z wierzchołka $h+1$ do w przez wierzchołki *większe bądź równe* $h+1$, otrzymujemy, że $(v_{h+1}^{[k-1]} \cdot w_{h+1}^{[\sqrt{n}-k-1]})[v,w]$ jest równe sumie wag wszystkich h-ścieżek długości $\sqrt{n}$ z wierzchołka v do w ale zawierających wierzchołek $h+1$, co więcej pierwsze wystąpienie wierzchołka $h+1$ jest na pozycji k. Stąd

$$Z_h[v,w] \;=\; Z_{h+1}[v,w] + \sum_{k=0}^{\sqrt{n}}((v_{h+1}^{[k-1]} \cdot w_{h+1}^{[\sqrt{n}-k-1]})[v,w]),$$

$$Z_h \;=\; Z_{h+1} + \sum_{k=0}^{\sqrt{n}}(v_{h+1}^{[k-1]} \cdot w_{h+1}^{[\sqrt{n}-k-1]}).$$

Dowiedliśmy tym samym poprawności kroku 2.

Aby zakończyć dowód naszego lematu zauważmy, że $\hat{A}[h,k] = a_h \cdot A_h^{k-2} \cdot a^h$, dla $h = 1,2,\ldots,n$ oraz $k = 2,3,\ldots,n$ ($\hat{A}[h,1] = A[h,h]$) i niech $k = q \cdot \sqrt{n} + r + 2$, dla $0 \le r < \sqrt{n}$.

Każdą h-ścieżkę π długości $q \cdot \sqrt{n} + r + 2$ z wierzchołka h do h możemy podzielić na dwie ścieżki: ścieżkę długości $q \cdot \sqrt{n} + 1$ z wierzchołka h do pewnego wierzchołka

$i > h$ oraz ścieżkę długości $r+1$ z wierzchołka i do h, jak zostało to pokazane na rysunku 3.4. Ostatecznie dostajemy, że

$$\hat{A}[h,k] = \hat{A}[h, q \cdot \sqrt{n} + r + 2] = a_h \cdot A_h^{q\sqrt{n}} A_h^r \cdot a^h = u_h^{[q]} \cdot v_h^{[r]}.$$

□

Przeanalizujmy nasz algorytm. Najbardziej kosztownymi krokami są krok 1, krok 2 oraz krok 3 i każdy z nich działa w czasie $O(n^{2.5})$. Krok 4 wymaga jedynie czasu $O(n^2)$ i jego koszt jest zdominowany przez koszty pozostałych kroków. Najbardziej zewnętrzna pętla **for** wykonuje się $O(n)$ razy, zatem złożoność czasowa naszego algorytmu to $O(n^{3.5})$.

3.3.2 Wykorzystanie Algorytmu Szybkiego Mnożenia Macierzy

Jak już zostało powiedziane we wstępie, wykorzystanie algorytmu szybkiego mnożenia macierzy pozwala polepszyć teoretyczną złożoność czasową naszego algorytmu [50, 51]. Niech ω oznacza wykładnik w algorytmie szybkiego mnożenia macierzy. Coppersmith i Winograd [11] pokazali, że możemy przyjąć $\omega = 2.3755$. Wartość γ tutaj będzie nie większa niż $\lceil n^\epsilon \rceil$, $\eta = \lceil n/\gamma \rceil$, a ϵ będzie stałą z przedziału $(0,1)$, której dokładną wartość podamy później. Rozważania w tej części mają czysto teoretyczny charakter.

Zauważmy, że w kroku 1 algorytmu 3.3.1 obliczającego $SkewClosure(A)$ wektor kolumnowy $v_h^{[r]} = A_h \cdot v_h^{[r-1]}$ jest równy wektorowi $A \cdot v_h^{[r-1]}$ z wyzerowanymi h pierwszymi współrzędnymi. Oznaczmy przez $V^{[r]}$ macierz rozmiaru $n \times n$ o kolumnach $v_1^{[r]}, v_2^{[r]}, \ldots, v_n^{[r]}$, dla $r = 0, 1, \ldots, \gamma$ oraz przez $\nabla(M)$ macierz powstałą z macierzy M poprzez wyzerowanie jej głównej przekątnej wraz z elementami znajdującymi się powyżej niej. Mamy wówczas

$$V^{[0]} = \nabla(A); \quad V^{[r+1]} = \nabla(A \cdot V^{[r]}).$$

Podobnie, wektor wierszowy $w_h^{[r]} = w_h^{[r-1]} \cdot A_{h-1}$ jest równy wektorowi $w_h^{[r-1]} \cdot A$ z wyzerowanymi $h-1$ pierwszymi współrzędnymi. Niech teraz $W^{[r]}$ będzie macierzą rozmiaru $n \times n$ o wierszach $w_1^{[r]}, w_2^{[r]}, \ldots, w_n^{[r]}$, dla $r = 0, 1, \ldots, \gamma$ a $\Delta(M)$ macierzą powstałą z macierzy M poprzez wyzerowanie elementów znajdujących się poniżej jej głównej przekątnej. Wówczas

$$W^{[0]} = \Delta(A); \quad W^{[r+1]} = \Delta(W^{[r]} \cdot A).$$

Koszt wyznaczenia wszystkich macierzy $V^{[r]}$ oraz $W^{[r]}$, dla $r = 0, 1, \ldots, \gamma$ jest równy $O(\gamma \cdot n^\omega)$.

Całkowita zmiana macierzy Z_h w kroku 2 to

$$v_{h+1}^{[-1]} \cdot w_{h+1}^{[\gamma-1]} + v_{h+1}^{[0]} \cdot w_{h+1}^{[\gamma-2]} + v_{h+1}^{[1]} \cdot w_{h+1}^{[\gamma-3]} + \ldots + v_{h+1}^{[\gamma-1]} \cdot w_{h+1}^{[-1]} = V_{h+1} \cdot W_{h+1},$$

gdzie V_{h+1} jest macierzą rozmiaru $n \times \gamma$ o kolumnach $v_{h+1}^{[-1]}, v_{h+1}^{[0]}, \ldots, v_{h+1}^{[\gamma-1]}$ a W_{h+1} jest macierzą rozmiaru $\gamma \times n$ o wierszach $w_{h+1}^{[\gamma-1]}, w_{h+1}^{[\gamma-2]}, \ldots, w_{h+1}^{[-1]}$.

Iloczyn $V_{h+1} \cdot W_{h+1}$ moglibyśmy wyznaczyć wykorzystując algorytm szybkiego mnożenia macierzy, ale jak się zaraz okaże nie wszystkie iloczyny tej postaci będą nam potrzebne. Wyznaczymy jedynie macierze Z_h dla okrojonego zbioru indeksów $h = n, n-\eta, \ldots, n-(\gamma-1)\cdot\eta$. Możemy zrobić to w prosty sposób w czasie $O(\gamma \cdot n^\omega)$ ponieważ

$$Z_h = Z_{h+\eta} + V_{h+\eta} \cdot W_{h+\eta} + V_{h+\eta-1} \cdot W_{h+\eta-1} + \ldots + V_{h+1} \cdot W_{h+1}$$

$$= Z_{h+\eta} + [V_{h+\eta}|V_{h+\eta-1}|\ldots|V_{h+1}] \cdot \begin{bmatrix} W_{h+\eta} \\ W_{h+\eta-1} \\ \vdots \\ W_{h+1} \end{bmatrix} = Z_{h+\eta} + V_{(h)} \cdot W_{(h)},$$

gdzie $V_{(h)}$ oraz $W_{(h)}$ są obie macierzami rozmiaru $n \times n$.

Do poprawienia czasu działania kroku 3 możemy wykorzystać podobną technikę jak dla kroku 1. Mianowicie, dla $i = 1, 2, \ldots, \eta$

$$u_{h+i}^{[q]} = u_{h+i}^{[q-1]} \cdot Z_{h+i}$$

$$= u_{h+i}^{[q-1]} \cdot Z_{h+\eta} + u_{h+i}^{[q-1]} \cdot [V_{h+\eta}|V_{h+\eta-1}|\ldots|V_{h+i+1}] \cdot \begin{bmatrix} W_{h+\eta} \\ W_{h+\eta-1} \\ \vdots \\ W_{h+i+1} \end{bmatrix}$$

a drugi składnik tej sumy jest równy wektorowi $u_{h+i}^{[q-1]} \cdot V_{(h)}$ z wyzerowanymi $i \cdot \gamma$ ostatnimi współrzędnymi i przemnożonemu przez macierz $W_{(h)}$.

Oznaczmy przez $U_h^{[q]}$, dla $h = n-\eta, n-2\cdot\eta, \ldots, n-\gamma\cdot\eta$ oraz $q = 0, 1, \ldots, \eta$, macierz rozmiaru $\eta \times n$ o wierszach $u_{h+\eta}^{[q]}, u_{h+\eta-1}^{[q]}, \ldots, u_{h+1}^{[q]}$ a przez $\nabla_\star(M)$ macierz rozmiaru $\eta \times n$ taką, że

$$\nabla_\star(M)[i,j] = M[i,j],$$

jeśli $j \leq \gamma \cdot (i-1)$, 0 w przeciwnym wypadku. Otrzymujemy, że

$$U_h^{[q]} = U_h^{[q-1]} \cdot Z_{h+\eta} + \nabla_\star(U_h^{[q-1]} \cdot V_{(h)}) \cdot W_{(h)}.$$

Koszt obliczenia macierzy $U_h^{[q]}$ jest proporcjonalny do kosztu przemnożenia macierzy rozmiaru $\eta \times n$ przez macierz rozmiaru $n \times n$.

Wykorzystamy teraz szybką metodę mnożenia macierzy prostokątnych opisaną w [10], która pozwala przemnożyć macierz rozmiaru $n \times n$ przez macierz rozmiaru $n \times \nu$ wykonując jedynie $\tilde{O}(n^{\omega-\theta} \cdot \nu^{\theta})$ operacji arytmetycznych, gdzie $\theta = (\omega - 2)/(1 - \zeta)$ dla $\zeta = 0.2946289$ (realizowane jest to poprzez podział macierzy rozmiaru $n \times n$ na bloki wielkości $t \times t$ a macierzy $n \times \nu$ na bloki wielkości $t \times t^{\zeta}$ w taki sposób, aby $n/t = \nu/t^{\zeta}$ oraz aby wyznaczanie iloczynów poszczególnych bloków wymagało jedynie $\tilde{O}(t^2)$ operacji arytmetycznych). Możemy zatem obliczyć wszystkie macierze $U_h^{[q]}$, dla $h = n - \eta, n - 2 \cdot \eta, \ldots, n - \gamma \cdot \eta$ oraz $q = 0, 1, \ldots, \eta$, w czasie $\tilde{O}(n^{1+\omega-\theta} \cdot \eta^{\theta})$.

Algorytm 3.3.2. *FasterSkewClosure*(A)

KROK 1 **for** $r = 0, 1, \ldots, \gamma - 1$ **do**
$V^{[r+1]} := \nabla(A \cdot V^{[r]})$;
$W^{[r+1]} := \Delta(W^{[r]} \cdot A)$;

KROK 2 **for** $h = n - \eta, n - 2 \cdot \eta, \ldots, n - (\gamma - 1) \cdot \eta$ **do**
$Z_h := Z_{h+\eta} + V_{(h)} \cdot W_{(h)}$;

KROK 3 **for** $h = n - \eta, n - 2 \cdot \eta, \ldots, n - \gamma \cdot \eta$ **do**
for $q = 0, 1, \ldots, \eta$ **do**
$U_h^{[q]} := U_h^{[q-1]} \cdot Z_{h+\eta} + \nabla_{\star}(U_h^{[q-1]} \cdot V_{(h)}) \cdot W_{(h)}$;

KROK 4 **for** $h = 1, 2, \ldots, n$ **do**
for $r = 0, 1, \ldots, \gamma - 1$ **do**
for $q = 0, 1, \ldots, \eta - 1$ **do**
$\hat{A}[h, q \cdot \gamma + r + 2] := u_h^{[q]} \cdot v_h^{[r]}$;
return $\hat{A}$;
end;

Całkowita złożoność czasowa kroku 4 jest zdominowana przez złożoności pozostałych kroków. W konsekwencji czas działania algorytmu *FasterSkewClosure*(A) wynosi $\tilde{O}(\gamma \cdot n^{\omega} + n^{1+\omega-\theta} \cdot \eta^{\theta})$ i dla $\gamma = \lceil n^{0.65} \rceil$, $\eta = \lceil n/\gamma \rceil$ jest równy $\tilde{O}(n^{3.03})$.

Pokazaliśmy więc:

Lemat 3.3.2. *Skośne domknięcie macierzy rozmiaru $n \times n$ może być obliczone w czasie $\tilde{O}(n^{3.03})$ bez wykonywania operacji dzielenia.*

Z lematów 3.2.2, 3.3.2 oraz z twierdzenia 3.2.2 otrzymujemy:

Twierdzenie 3.3.1. *Wyznacznik macierzy A rozmiaru $n \times n$ może być obliczony w czasie $\tilde{O}(n^{3.03})$ bez wykonywania operacji dzielenia.*

Rozdział 4

Pfaffian

4.1 Wstęp

Pfaffian pojawia się w sposób naturalny w badaniach nad skojarzeniami w grafach, *Pfaffian* zorientowanego grafu jest równy sumie przebiegającej po wszystkich skojarzeniach doskonałych wag tych skojarzeń, przy czym każde skojarzenie ma przypisany znak zależny od orientacji. To wskazuje już na podobieństwa do wyznacznika. Gdyby nie obecność znaku w tej definicji Pfaffian można by wykorzystać do wyznaczania liczby skojarzeń doskonałych. Wiadomo, że istnieją specjalne grafy, które można zorientować w taki sposób, aby wszystkie skojarzenia doskonałe miały dodatni znak. To oczywiście oznacza, że żadne skracanie się składników w sumie nie będzie miało miejsca a stąd Pfaffian da dokładnie liczbę skojarzeń doskonałych. Takie orientacje grafów nazywa się *p-orientacjami*.

Pfaffian możemy wyznaczyć wykonując procedurę eliminacji dla macierzy skośnosymetrycznych, która jest podobna do eliminacji Gaussa (i wykorzystuje operację dzielenia). Alternatywnie, pierwiastek kwadratowy z wyznacznika daje Pfaffian z dokładnością do znaku. Żadne z tych podejść nie może jednak zostać użyte, jeśli operacje dzielenia i pierwiastkowania będą musiały zostać wyeliminowane. Knuth [25] przedstawił krótką historię Pfaffianu i doszedł do wniosku, że pojęcie Pfaffianu w wielu problemach jest bardziej fundamentalne od pojęcia wyznacznika, z którym jest ściśle związane.

W tym rozdziale przedstawimy algorytm obliczania Pfaffianu działający w czasie $\tilde{O}(n^{3.03})$ i nie wykonujący operacji dzielenia.

4.2 Naprzemienne Ciągi Pseudocykli

Kombinatoryczne podejście do wyznacznika opisane w rozdziale 3 może zostać zmodyfikowane dla Pfaffianu. Rozważamy iloczyn Pfaffianów dwóch macierzy skośno-symetrycznych A i B, obie rozmiarów $n \times n$ (n jest parzyste), gdzie

$$B[2j-1, 2j] = 1, \ B[2j, 2j-1] = -1,$$

dla wszystkich $j = 1, 2, \ldots, n/2$ oraz $B[i, j] = 0$ w przeciwnym wypadku. Macierz B reprezentuje skojarzenie

$$\mathcal{M}_0 = \{\{1, 2\}, \{3, 4\}, \ldots, \{n-1, n\}\},$$

które nazywamy *skojarzeniem bazowym*. Mamy

$$Pf(B) = \sum_{\mathcal{M}} sgn(\mathcal{M}) \cdot weight(\mathcal{M}, B) = sgn(\mathcal{M}_0) \cdot weight(\mathcal{M}_0, B) = 1$$

oraz

$$Pf(A) = \sum_{\mathcal{M}} sgn(\mathcal{M}) \cdot weight(\mathcal{M}, A).$$

Rozwijając definicję iloczynu Pfaffianów macierzy A i B dostajemy

$$Pf(A) = Pf(A) \cdot Pf(B)$$

$$= \sum_{\mathcal{M}} (sgn(\mathcal{M}) \cdot weight(\mathcal{M}, A) \cdot sgn(\mathcal{M}_0) \cdot weight(\mathcal{M}_0, B)),$$

co prowadzi do operacji „nałożenia" na siebie dwóch skojarzeń, dowolnego skojarzenia $\mathcal{M}_A$ z bazowym skojarzeniem $\mathcal{M}_0$. Ta suma skojarzeń $\mathcal{M}_A$ i $\mathcal{M}_0$ rozpada się na sumę naprzemiennych cykli o całkowitej liczbie krawędzi równej n:

$$\mathcal{M}_A \uplus \mathcal{M}_0 = \mathcal{C} = C_1 \cup C_2 \cup \ldots.$$

Naprzemiennym cyklem (lub $\mathcal{M}_0$-*naprzemiennym cyklem*) $C = (c_0, c_1, \ldots, c_i)$ nazywamy cykl parzystej długości, którego krawędzie należą na przemian do $\mathcal{M}_A$ i $\mathcal{M}_0$, musi zachodzić $c_1 = c_2 - 1$, jeśli c_2 jest parzyste oraz $c_1 = c_2 + 1$, jeśli c_2 jest nieparzyste i ta sama relacja ma miejsce pomiędzy c_3 i c_4, ... , c_{i-1} i c_i, jak pokazano na rysunku 4.1.

Krawędzie, które jednocześnie zawarte są w obu skojarzeniach $\mathcal{M}_A$ i $\mathcal{M}_0$, są rozróżnialne w sumie $\mathcal{M}_A \uplus \mathcal{M}_0$: dla każdej wspólnej krawędzi, $\mathcal{M}_A \uplus \mathcal{M}_0$ zawiera dwie równoległe krawędzie tworzące naprzemienny cykl długości 2.

Rysunek 4.1: Konstrukcja naprzemiennych cykli.

Naprzemiennemu cyklowi możemy przypisać dwie orientacje. Ustalamy więc orientację poprzez wyróżnienie wierzchołka o najmniejszym numerze jako *lidera* naszego cyklu i przyjęcie założenia, że pierwsza krawędź musi należeć do skojarzenia $\mathcal{M}_A$.

Waga naprzemiennego cyklu $C = (c_0, c_1, \ldots, c_i)$ z $c_0 = c_i$ oraz $c_t > c_0$, dla wszystkich $0 < t < i$, jest równa

$$weight_B(C, A) = A[c_0, c_1] \cdot B[c_1, c_2] \cdot A[c_2, c_3] \cdot B[c_3, c_4] \cdot \ldots \cdot A[c_{i-2}, c_{i-1}] \cdot B[c_{i-1}, c_i]$$

pamiętając, że dla macierzy B mamy $B[2j-1, 2j] = 1$, $B[2j, 2j-1] = -1$. *Rodzina naprzemiennych cykli* $\mathcal{C}$ jest zbiorem rozłącznych naprzemiennych cykli. Jej *długość* jest równa całkowitej liczbie krawędzi w tworzących ją cyklach. Jeżeli długość rodziny naprzemiennych cykli jest równa n mówimy o *naprzemiennym pokryciu cyklami*. *Waga* naprzemiennego pokrycia cyklami $\mathcal{C}$ jest iloczynem wag tworzących go cykli, a jego *znak*, $sgn(C)$, jest równy $(-1)^k$, gdzie k jest liczbą naprzemiennych cykli w $\mathcal{C}$.

Uogólniając twierdzenie 3.2.1 ze strony 24, Mahajan, Subramanya i Vinay [33] rozszerzyli notację naprzemiennych pokryć cyklami do naprzemiennych ciągów pseudocykli.

Naprzemienny pseudocykl $(c_0, c_1, \ldots, c_i)$ długości i jest zdefiniowany jak pseudocykl, z dodatkowymi ograniczeniami, że i musi być parzyste oraz co druga krawędź (c_1, c_2), (c_3, c_4), $\ldots$, (c_{i-1}, c_i) musi należeć do $\mathcal{M}_0$. Mówimy, że h jest *liderem* naprzemiennego pseudocyklu $(c_0, c_1, \ldots, c_i)$, jeśli $c_0 = c_i = h$ oraz $c_t > h$, dla każdego $0 < t < i$. Waga naprzemiennego pseudocyklu jest zdefiniowana jak dla naprzemiennego cyklu. Naprzemienny ciąg pseudocykli, jego długość, waga oraz znak są zdefiniowane jak dla pseudocykli oraz rodziny naprzemiennych cykli.

To prowadzi do następującego twierdzenia, którego szczegółowy dowód można znaleźć w pracach [38, 33].

Twierdzenie 4.2.1.

$$Pf(A) \;=\; \sum_{\mathcal{C}} sgn(\mathcal{C}) \cdot weight_B(\mathcal{C}, A),$$

gdzie sumowanie przebiega po wszystkich naprzemiennych ciągach pseudocykli $\mathcal{C}$ długości n.

Algorytm programowania dynamicznego obliczania Pfaffianu poprzez sumowanie po wszystkich częściowych naprzemiennych ciągach pseudocykli w porządku ich rosnących długości możemy z łatwością sformułować analogicznie jak w części 3.2.5. Można go również znaleźć w pracy [33].

4.3 Nasza Wersja Algorytmu MSV

Niech $Shrink(A)$ będzie macierzą rozmiaru $n/2 \times n/2$ powstałą z macierzy A poprzez usunięcie z niej wszystkich wierszy o indeksach parzystych oraz kolumn o indeksach większych niż $n/2$, to znaczy

$$Shrink(A)[i,j] = A[2i-1, j],$$

dla wszystkich $1 \leq i, j \leq n/2$.

Przez $ExchColumns(A)$ oznaczmy permutację macierzy A polegającą na zamianie sąsiadujących kolumn, dokładniej, kolumnę pierwszą zamieniamy miejscami z kolumną drugą, kolumnę trzecią z kolumną czwartą itd.

Przykład 4.3.1. Weźmy macierz

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

Mamy $Shrink(A) = \begin{bmatrix} 1 & 2 \\ 9 & 10 \end{bmatrix}$ oraz

$$ExchColumns(A) = \begin{bmatrix} 2 & 1 & 4 & 3 \\ 6 & 5 & 8 & 7 \\ 10 & 9 & 12 & 11 \\ 14 & 13 & 16 & 15 \end{bmatrix}$$

Będziemy potrzebować również niewielkiej modyfikacji macierzy $SkewClosure(A)$. Przez $SkewClosure'(A)$ oznaczamy macierz rozmiaru $n \times n$, gdzie

$$SkewClosure'(A)[h,k] = \sum_{\pi \in \Pi'_{h,k}} weight(\pi, A)$$

oraz $\Pi'_{h,k}$ jest zbiorem wszystkich pseudocykli $\pi = (i_0, i_1, \ldots, i_k)$ z $\Pi_{h,k}$ spełniających dodatkowy warunek, że $i_t > h+1$ dla każdego $0 < t < k$. Pseudocykle spełniające powyższy warunek nazywamy *p-pseudocyklami.*

Ciąg p-pseudocykli, jego długość, waga oraz znak są zdefiniowane jak dla pseudocykli.

Algorytm wyznaczający macierz $SkewClosure'(A)$ jest tylko drobną modyfikacją algorytmu 3.3.2 ze strony 36 obliczającego macierz $SkewClosure(A)$.

Nasza wersja algorytmu MSV [50, 51] może zostać sformułowana następująco.

Algorytm 4.3.1. *ComputePfaffian*(A)
/* A jest macierzą skośno-symetryczną */
$A_\star := ExchColumns(A)$;
for $i, j = 1, 2, \ldots, n$ **do**
$A_\star[i,j] := (-1)^j \; A_\star[i,j]$;
$\hat{A}_\star := SkewClosure'(A_\star)$;
return $Knapsack(-Shrink(\hat{A}_\star))$;
end;

Twierdzenie 4.3.1. *Algorytm ComputePfaffian*(A) *jest poprawny:*

$$Pf(A) \;=\; Knapsack(-Shrink(SkewClosure'(A_\star))). \tag{4.1}$$

Dowód. Zbiór wszystkich naprzemiennych pseudocykli długości i z liderem h reprezentujemy przez bijektywny z nim zbiór wszystkich p-pseudocykli długości $i/2$ z liderem h. P-pseudocykl $C = (c_0, c_2, c_4 \ldots, c_{i-2}, c_i)$ odpowiada naprzemiennemu pseudocyklowi $C_+ = (c_0, c_1, c_2 \ldots, c_{i-1}, c_i)$, gdzie $c_1 = c_2 - 1$, jeśli c_2 jest parzyste oraz $c_1 = c_2 + 1$, jeśli c_2 jest nieparzyste i ta sama zależność ma miejsce pomiędzy c_3 i c_4, ... , c_{i-1} i c_i.

Zauważmy, że powyższe warunki wymuszają, że h musi być nieparzyste (w przeciwnym wypadku mielibyśmy $c_{i-1} = c_i - 1 < h$) oraz, że wykorzystujemy tu nasze założenie $c_{2t} > h+1$, dla każdego $0 < t < i/2$ (dla $c_{2t} = h+1$ w C byłoby $c_{2t-1} = h$ w C_+). Mamy również

$$weight_B(C_+, A) = weight(C, A_\star),$$

ponieważ

$$A_\star[i, 2j] = A[i, 2j-1] = A[i, 2j-1] \cdot B[2j-1, 2j],$$

$$A_\star[i, 2j-1] = -A[i, 2j] = A[i, 2j] \cdot B[2j, 2j-1].$$

$A_\star$ tutaj jest $A_\star$ po modyfikacji w linii 5 algorytmu *ComputePfaffian*(A).

W konsekwencji dostajemy, że

$$Pf(A) = \sum_{\mathcal{C}} sgn(\mathcal{C}) \cdot weight(\mathcal{C}, A_\star),$$

gdzie suma przebiega po wszystkich ciągach p-pseudocykli $\mathcal{C}$, których długość jest równa $n/2$ oraz liderzy są nieparzyści. Tak jak w przypadku wyznacznika grupujemy ciągi p-pseudocykli bazując na liderach oraz na długościach poszczególnych p-pseudocykli. Ponieważ $\hat{A}_\star[h, l]$ równe jest całkowitej wadze (ze względu na macierz $A_\star$) wszystkich p-pseudocykli, które mają wierzchołek h jako lidera oraz długość l, otrzymujemy, że wartość $Pf(A)$ jest sumą wszystkich iloczynów postaci

$$(-1)^k \cdot \hat{A}_\star[h_1, l_1] \cdot \hat{A}_\star[h_2, l_2] \cdot \ldots \cdot \hat{A}_\star[h_k, l_k],$$

gdzie $h_1, h_2, \ldots, h_k$ są nieparzyste,

$$1 \leq h_1 < h_2 \ldots < h_k \leq n,\ 1 \leq l_1, l_2, \ldots l_k \leq n/2,\ l_1 + l_2 \ldots + l_k \ = \ n/2,$$

a to jest dokładnie

$$Knapsack(-Shrink(\hat{A}_\star)) \ = \ Knapsack(-Shrink(SkewClosure'(A_\star))).$$

□

Udowodniliśmy więc:

Twierdzenie 4.3.2. *$Pf(A)$ może zostać obliczony w czasie $\tilde{O}(n^{3.03})$ bez wykonywania operacji dzielenia.*

Rozdział 5

Grafy Planarne

5.1 Wstęp

Problem obliczania wyznacznika wydaje się być znacznie prostszy dla macierzy określonych przez grafy planarne. Na samym początku takie macierze są rzadkie, mają one jedynie $O(n)$ niezerowych współrzędnych. Jednak możemy powiedzieć znacznie więcej. Lipton, Rose oraz Tarjan [31] pokazali, że istnienie rodziny $O(\sqrt{n})$-separatorów dla tej klasy grafów daje możliwość wykonania eliminacji Gaussa w czasie $O(n^{3/2})$ (lub $O(n^{1.19})$, jeśli użyty zostanie algorytm szybkiego mnożenia macierzy). Ale działanie tego algorytmu ciągle wymaga dzielenia.

W tym rozdziale przedstawimy algorytm obliczania wyznacznika dla przypadku grafów planarnych działający w czasie $O(n^{2.5})$ i nie wykonujący operacji dzielenia.

5.2 Grafy Planarne i Mały Separator

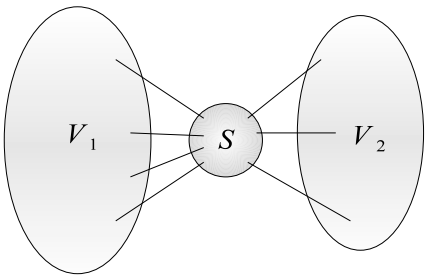
Powiemy, że graf $G = (V, E)$ posiada *rodzinę $s(n)$-separatorów* (ze względu na pewną stałą n_0), jeśli $|V| \leq n_0$ lub poprzez usunięcie pewnego podzbioru S zbioru wierzchołków V, takiego, że

$$|S| \leq s(|V|),$$

dostajemy podział grafu G na dwa rozłączne podgrafy posiadające rodzinę $s(n)$-separatorów o zbiorach wierzchołków V_1 i V_2, takich, że

$$|V_1| \leq 2/3|V|,\ |V_2| \leq 2/3|V|.$$

Zbiór S występujący w tej definicji nazywamy *$s(n)$-separatorem* grafu G. *Małym separatorem* nazywamy $O(\sqrt{n})$-separator.



Rysunek 5.1: Podział grafu G ze względu na separator S. Każda ścieżka z wierzchołka $v \in V_1$ do dowolnego wierzchołka $w \notin V_1$ musi zawierać wierzchołek $s \in S$.

Podział grafu G powstały przez rekurencyjne zastosowanie powyższej definicji nazywamy *drzewem $s(n)$-separatorów*.

Następujące twierdzenie udowodnione przez Liptona i Tarjana [32] daje ważny przykład grafów posiadających rodzinę $O(\sqrt{n})$-separatorów.

Twierdzenie 5.2.1. *Grafy planarne posiadają rodzinę $O(\sqrt{n})$-separatorów. Co więcej, drzewo $O(\sqrt{n})$-separatorów dla grafów planarnych można znaleźć w czasie $\tilde{O}(n)$.*

5.3 Algorytm Numerujący

Niech G będzie grafem posiadającym rodzinę $O(\sqrt{n})$-separatorów i niech T_G będzie drzewem $O(\sqrt{n})$-separatorów dla G. Nasz rekurencyjny algorytm numeruje wierzchołki grafu G tak, że jeśli l wierzchołków ma już przypisany numer, każdy z nich mniejszy niż a (stała, którą zdefiniujemy później), wówczas naszym celem jest ponumerowanie pozostałych wierzchołków G kolejnymi liczbami od a do b.

Jeśli G zawiera nie więcej niż $n_0 = 9$ wierzchołków, numerujemy nie ponumerowane jeszcze wierzchołki dowolnie liczbami od a do b. W przeciwnym wypadku, niech S będzie $O(\sqrt{n})$-separatorem grafu G (znajdującym się na najwyższym poziomie w drzewie T_G) oraz niech G_1, G_2 będą dwoma rozłącznymi podgrafami grafu G o zbiorach wierzchołków V_1 i V_2 powstałymi poprzez usunięcie zbioru S z G i odpowiadającymi poddrzewom T_{G_1} oraz T_{G_2} w T_G. Numerujemy wierzchołki zbioru S dowolnie liczbami od a do $a + |S| - 1$.

Wywołujemy nasz algorytm rekurencyjnie do podgrafu G_1 i poddrzewa T_{G_1} numerując nie ponumerowane jeszcze wierzchołki w V_1 liczbami od $a + |S|$ do $a + |S| + |V_1| - 1$. Wywołujemy nasz algorytm rekurencyjnie do podgrafu G_2 i poddrzewa T_{G_2} numerując nie ponumerowane jeszcze wierzchołki w V_2 liczbami od $a + |S| + |V_1|$ do b.

Algorytm 5.3.1. $Numbering(G, T_G, a, b)$

if $|V| \leq 9$ **then**
$NumberArbitrarily(V, a, b)$;
else
$NumberArbitrarily(S, a, a + |S| - 1)$;
$Numbering(G_1, T_{G_1}, a+|S|, a+|S|+|V_1|-1)$;
$Numbering(G_2, T_{G_2}, a + |S| + |V_1|, b)$;
end;

Na początku, kiedy wszystkie wierzchołki grafu G nie mają przypisanych numerów, wówczas wywołanie naszego algorytmu dla G z parametrami $a = 1$, $b = n$ spowoduje ponumerowanie wierzchołków G liczbami od 1 do n.

Czas działania opisanego algorytmu numerującego jest równy $O(n)$.

5.4 Obliczanie $SkewClosure(A)$

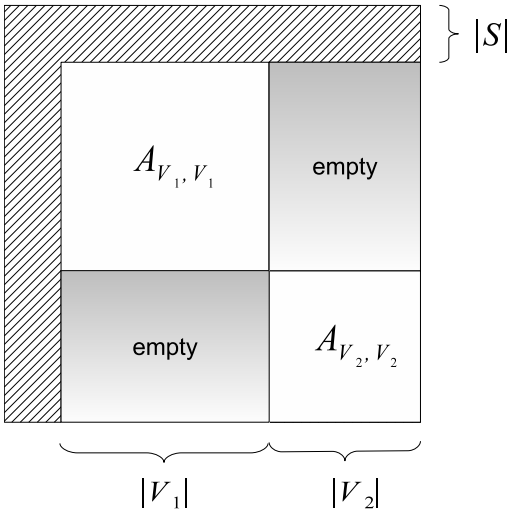
Zakładamy teraz, że graf G_A jest grafem planarnym oraz, że wiersze i kolumny macierzy A są spermutowane przy użyciu drzewa $O(\sqrt{n})$-separatorów T_{G_A} grafu G_A przez algorytm 5.3.1 opisany w części 5.3. Wierzchołki separatora S znajdującego się na najwyższym poziomie w drzewie T_{G_A} odpowiadają pierwszym $|S|$ wierszom i pierwszym $|S|$ kolumnom, itd.

Aby znaleźć skośne domknięcie, $SkewClosure(A)$, macierzy A rozmiaru $n \times n$ najpierw znajdujemy tę część macierzy $SkewClosure(A)$, która odpowiada małemu separatorowi S grafu G_A, tak jak podmacierz $\hat{A}_S$ rozmiaru $|S| \times n$. Następnie rozwiązujemy problem oddzielnie dla dwóch rozłącznych podgrafów o zbiorach wierzchołków V_1 i V_2 powstałych poprzez usunięcie zbioru S z grafu G_A wyznaczając tym samym podmacierze $\hat{A}_{V_1}$ rozmiaru $|V_1| \times n$ oraz $\hat{A}_{V_2}$ rozmiaru $|V_2| \times n$ odpowiednio.

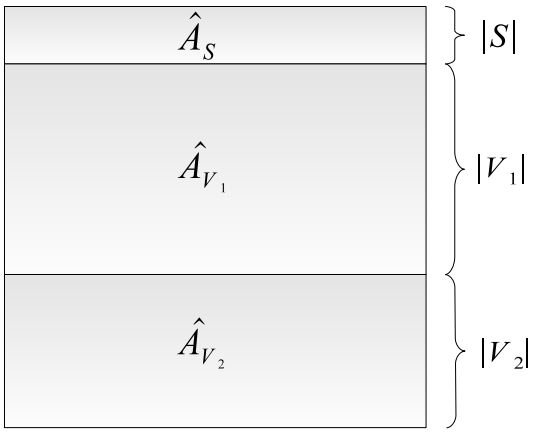
Dla macierzy A rozmiaru $n \times n$ przez $SkewClosure''(A, N)$ oznaczmy macierz rozmiaru $n \times N$ taką, że $SkewClosure''(A, N)[h, k]$ jest równe sumie wag wszystkich pseudocykli w grafie G_A długości k z liderem h, dla $h = 1, 2, \ldots, n$ oraz $k = 1, 2, \ldots, N$. Mamy, że

$$SkewClosure(A) = SkewClosure''(A, n).$$

Macierz $SkewClosure''(A, N)$ obliczamy następująco.



Rysunek 5.2: Podział macierzy A rozmiaru $n \times n$ ze względu na mały separator S grafu G_A i dwa rozłączne podgrafy o zbiorach wierzchołków V_1 i V_2 powstałe przez usunięcie zbioru S z G_A. Macierze A_{V_1,V_2} oraz A_{V_2,V_1} są macierzami zerowymi, ponieważ nie mamy żadnej krawędzi pomiędzy zbiorami wierzchołków V_1 i V_2 w grafie G_A.



Rysunek 5.3: Podział macierzy $\hat{A}$ rozmiaru $n \times n$ ze względu na mały separator S grafu G_A i dwa rozłączne podgrafy o zbiorach wierzchołków V_1 i V_2 powstałe przez usunięcie zbioru S z grafu G_A.

Algorytm 5.4.1. $SkewClosure''(A, N)$

/* Niech S, V, V_1 oraz V_2 będą jak wyżej. */
if $|V| \leq 9$ **then**
$\hat{A}_V := SeparatorSkewClosure(A, V, N)$;
else
$\hat{A}_S := SeparatorSkewClosure(A, S, N)$;
$\hat{A}_{V_1} := SkewClosure''(A_{V_1,V_1}, N)$;
$\hat{A}_{V_2} := SkewClosure''(A_{V_2,V_2}, N)$;
return $\hat{A}_V$;
end;

Algorytm $SeparatorSkewClosure(A, S, N)$ dla macierzy A rozmiaru $n \times n$ oblicza macierz $\hat{A}_S$ rozmiaru $|S| \times N$ w czasie $O(|S| \cdot N \cdot n^\theta)$, gdzie n^θ jest czasem potrzebnym do przemnożenia macierzy rozmiaru $1 \times n$ przez macierz A, przy czym zakładamy, że jest nie mniejsze niż $O(n)$ (czas potrzebny do przemnożenia macierzy rozmiaru $1 \times n$ przez macierz rozmiaru $n \times 1$).

Algorytm 5.4.2. $SeparatorSkewClosure(A, S, N)$

for $h = 1, 2, \ldots, |S|$ **do**
$\hat{A}_S[h, 1] := A[h, h]$;
$a_h^{[0]} := a_h$;
for $k = 1, 2, \ldots, N-1$ **do**
$a_h^{[k]} := a_h^{[k-1]} \cdot A_h$;
$\hat{A}_S[h, k+1] := a_h^{[k-1]} \cdot a^h$;
return $\hat{A}_S$;
end;

Najpierw zauważmy, że algorytm $SkewClosure''(A, N)$ jest poprawny. Każda ścieżka z wierzchołka $v \in V_1$ do dowolnego wierzchołka $w \notin V_1$ musi zawierać wierzchołek $s \in S$, ale wierzchołki z S są ponumerowane liczbami, z których każda jest mniejsza niż numer przypisany wierzchołkowi v. Analogicznie dla ścieżek z $v \in V_2$.

Stąd nie istnieje żaden pseudocykl z liderem $h \in V_1$ zawierający wierzchołek $w \notin V_1$ oraz nie istnieje żaden pseudocykl z liderem $h \in V_2$ zawierający wierzchołek $w \notin V_2$.

Przeanalizujemy teraz nasz algorytm. G_A jest grafem planarnym, więc macierz A rozmiaru $n \times n$ ma jedynie $O(n)$ niezerowych współczynników, zatem czas potrzebny do przemnożenia macierzy rozmiaru $1 \times n$ przez macierz A to tylko $O(n)$, co razem z $|S| = O(\sqrt{n})$ daje, że złożoność algorytmu $SeparatorSkewClosure(A, S, N)$ jest równa $O(N \cdot n^{1.5})$. Rozmiar każdego ze zbiorów wierzchołków $|V_i|$, $i = 1, 2$, jest co najwyżej równy $2/3|V|$ i $|V_1|+|V_2| \leq |V|$, więc całkowity czas potrzebny do obliczenia macierzy $SkewClosure(A) = SkewClosure''(A, n)$ dla macierzy A rozmiaru $n \times n$ spełnia zależność rekurencyjną

$$T(n) = n \cdot T_1(n),$$

gdzie

$$T_1(n) \leq T_1(n/3) + T_1(2/3n) + O(n^{1.5}).$$

Korzystając ze standardowych argumentów [3] dostajemy, że rozwiązaniem tej rekurencji jest $T_1(n) = O(n^{1.5})$ oraz $T(n) = O(n^{2.5})$.

W konsekwencji pokazaliśmy:

Lemat 5.4.1. *Jeśli G_A jest grafem planarnym wówczas skośne domknięcie macierzy A rozmiaru $n \times n$ możemy obliczyć w czasie $O(n^{2.5})$ bez wykonywania operacji dzielenia.*

5.5 Szybszy Algorytm Obliczania $Knapsack(A)$

Niech $\Theta_i(u)$, dla $i = 1, 2, \ldots, n$, będzie wielomianem stopnia n zdefiniowanym następująco

$$\Theta_i(u) = 1 + A[i, 1] \cdot u + A[i, 2] \cdot u^2 + \ldots + A[i, n] \cdot u^n.$$

Ponieważ $\mathcal{K}_t(A)$ jest równe sumie wszystkich iloczynów postaci

$$A[i_1, j_1] \cdot A[i_2, j_2] \cdot \ldots \cdot A[i_k, j_k],$$

gdzie

$$1 \leq i_1 < i_2 < \ldots < i_k \leq n,\ 1 \leq j_1, j_2, \ldots, j_k \leq n,\ j_1 + j_2 + \ldots + j_k = t$$

otrzymujemy, że

$$\Theta_1(u) \cdot \Theta_2(u) \cdot \ldots \cdot \Theta_n(u) = 1 + \mathcal{K}_1(A) \cdot u + \mathcal{K}_2(A) \cdot u^2 + \ldots + \mathcal{K}_n(A) \cdot u^n + u^{n+1} \cdot F(u),$$

gdzie $F(u)$ jest wielomianem stopnia $n^2 - n - 1$.

Wszystkie obliczenia możemy wykonywać w pierścieniu reszt modulo u^{n+1}. Zakładając, że iloczyn dwóch wielomianów stopnia n możemy wyznaczyć wykonując jedynie $\tilde{O}(n)$ operacji dodawania oraz mnożenia [41] dostajemy, że:

Lemat 5.5.1. *Jeśli A jest macierzą rozmiaru $n \times n$, wówczas wszystkie wartości $\mathcal{K}_1(A), \mathcal{K}_2(A), \ldots, \mathcal{K}_n(A) = Knapsack(A)$ możemy wyznaczyć w czasie $\tilde{O}(n^2)$ korzystając jedynie z operacji dodawania oraz mnożenia.*

Ostatecznie, z lematów 5.4.1, 5.5.1 oraz z twierdzenia 3.2.2 otrzymujemy:

Twierdzenie 5.5.1. *Jeśli G_A jest grafem planarnym wówczas wyznacznik $det(A)$ macierzy A rozmiaru $n \times n$ możemy obliczyć w czasie $O(n^{2.5})$ bez wykonywania operacji dzielenia.*

Rozdział 6

Inne Zastosowania

6.1 Wielomian Charakterystyczny

Technika wprowadzona przez Mahajana i Vinaya może być wykorzystana do wyznaczania *wszystkich* współczynników wielomianu charakterystycznego macierzy A

$$\Phi_A(\lambda) = det(\lambda \cdot I_n - A) = c_0 \cdot \lambda^n + c_1 \cdot \lambda^{n-1} + \ldots + c_{n-1} \cdot \lambda + c_n.$$

Mamy $\lambda \cdot I_n - A =$

$$\begin{bmatrix} \lambda - A[1,1] & -A[1,2] & -A[1,3] & \ldots & -A[1,n] \\ -A[2,1] & \lambda - A[2,2] & -A[2,3] & \ldots & -A[2,n] \\ -A[3,1] & -A[3,2] & \lambda - A[3,3] & \ldots & -A[3,n] \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -A[n-1,1] & -A[n-1,2] & -A[n-1,3] & \ldots & -A[n-1,n] \\ -A[n,1] & -A[n,2] & -A[n,3] & \ldots & \lambda - A[n,n] \end{bmatrix}$$

Zapisując $det(\lambda \cdot I_n - A)$ w terminach pokryć cyklami zobaczymy, że c_{n-r}, współczynnik w wielomianie $\Phi_A(\lambda)$, może zostać obliczony jako suma przebiegająca po wszystkich permutacjach σ zawierających co najmniej r punktów stałych wag tych permutacji liczonych poza tymi punktami stałymi

$$c_{n-r} = (-1)^n \sum_{S \subseteq \{1,2,\ldots,n\}, |S|=r} \sum_{\sigma \in S_n, j \in S \Rightarrow \sigma(j)=j} sgn(\sigma) \prod_{i \notin S} (-A[i, \sigma(i)]).$$

Niech k będzie liczbą cykli w rozkładzie na cykle permutacji σ, wtedy

$$(-1)^n sgn(\sigma) \prod_{i \notin S} (-A[i, \sigma(i)]) = (-1)^{2n+k-r} \prod_{i \notin S} A[i, \sigma(i)] = (-1)^{k-r} \prod_{i \notin S} A[i, \sigma(i)].$$

Jeżeli permutacja σ jest stała na wszystkich punktach ze zbioru S, wówczas przez $sgn(\sigma|S)$ oznaczamy parzystość liczby cykli w rozkładzie na cykle permutacji σ, nie licząc punktów stałych ze zbioru S, tzn.

$$sgn(\sigma|S) = \begin{cases} 1 & : \text{ jeśli liczba cykli jest parzysta,} \\ -1 & : \text{ jeśli liczba cykli jest nieparzysta.} \end{cases}$$

Mamy teraz

$$c_{n-r} = \sum_{S\subseteq\{1,2,\ldots,n\},|S|=r} \sum_{\sigma\in S_n, j\in S\Rightarrow\sigma(j)=j} sgn(\sigma|S) \prod_{i\notin S} A[i,\sigma(i)].$$

Ale każdy składnik tej sumy jest wagą pewnego częściowego pokrycia cyklami, pokrywającego dokładnie $n-r$ wierzchołków. Aby wyznaczyć tę sumę rozważamy *częściowe ciągi pseudocykli.*

l-ciągiem pseudocykli nazywamy uporządkowany ciąg pseudocykli, którego całkowita liczba krawędzi jest równa dokładnie l (licząc powtarzające się krawędzie) i którego liderzy są w ściśle rosnącym porządku.

Waga l-ciągu pseudocykli $\mathcal{C}$, $weight(\mathcal{C}, A)$, jest równa iloczynowi wag tworzących go krawędzi a jego *znak*, $sgn(\mathcal{C}) = (-1)^k$, gdzie k jest liczbą pseudocykli w $\mathcal{C}$.

Mahajan i Vinay [35] zauważyli, że odwzorowanie skonstruowane w dowodzie twierdzenia 3.2.1 ze strony 24 odwzorowuje l-ciągi pseudocykli w l-ciągi pseudocykli otrzymując tym samym, że

$$c_l = \sum_{\mathcal{C}} sgn(\mathcal{C}) \cdot weight(\mathcal{C}, A),$$

gdzie sumowanie przebiega po *wszystkich* l-ciągach pseudocykli $\mathcal{C}$.

Podobnie jak poprzednio możemy poprawić algorytm Mahajana i Vinaya obliczający wielomian charakterystyczny [50, 51]. Omijamy w tym przypadku dowód poniższego rezultatu, który przebiega w sposób analogiczny do dowodu twierdzenia 3.2.2.

Twierdzenie 6.1.1. *Współczynniki wielomianu charakterystycznego $\Phi_A(\lambda)$ macierzy A spełniają równanie*

$$c_l \;=\; \mathcal{K}_l(-SkewClosure(A))$$

i mogą one zostać obliczone w czasie $\tilde{O}(n^{3.03})$ bez wykonywania operacji dzielenia.

Dodatkowo z lematów 5.4.1, 5.5.1 dostajemy:

Twierdzenie 6.1.2. *Jeśli G_A jest grafem planarnym wówczas współczynniki wielomianu charakterystycznego $\Phi_A(\lambda)$ macierzy A możemy obliczyć w czasie $O(n^{2.5})$ bez wykonywania operacji dzielenia.*

6.2 Macierz Dopełnień Algebraicznych

Dzięki wynikowi Baura i Strassena [4] pokazującemu jak obliczać wszystkie pochodne cząstkowe otrzymujemy, że dowolna metoda obliczania wyznacznika może zostać automatycznie skonwertowana do metody obliczającej macierz dopełnień algebraicznych, przy czym złożoność wzrasta o nie więcej niż stały czynnik. W konsekwencji dostajemy:

Twierdzenie 6.2.1. *Macierz dopełnień algebraicznych macierzy A może zostać obliczona w czasie $\tilde{O}(n^{3.03})$ bez wykonywania operacji dzielenia.*

Twierdzenie 6.2.2. *Jeśli G_A jest grafem planarnym wówczas macierz dopełnień algebraicznych macierzy A możemy obliczyć w czasie $O(n^{2.5})$ bez wykonywania operacji dzielenia.*

Rozdział 7

Interpretacje Kombinatoryczne Znanych Algorytmów Obliczania Wyznacznika

7.1 Wstęp

Jak już widzieliśmy, obliczanie wyznacznika jest równoważne z obliczaniem sumy wag pokryć cyklami w odpowiednim grafie, przy czym każde pokrycie ma przypisany znak. Metoda zaproponowana przez Mahajana i Vinaya opisana w 3.2.4 pokazuje, że w istocie możemy zastąpić sumowanie po zbiorze pokryć cyklami grafu przez sumowanie po znacznie większej klasie ciągów pseudocykli. Pokazaliśmy, że wszystkie „złe" ciągi pseudocykli, które nie są pokryciami cyklami, wzajemnie się redukują, ponieważ odpowiadające im wagi występują dokładnie tyle samo razy z dodatnim jak i ujemnym znakiem.

W tym rozdziale przedstawimy interpretacje kombinatoryczne znanych algorytmów obliczania wyznacznika, które jak się przekonamy odpowiadają dalszemu zwiększaniu klasy sumowanych obiektów i podobnie jak w przypadku algorytmu Mahajana i Vinaya wszystkie nadmiarowe składniki sum wzajemnie się będą redukować: metoda Samuelsona [40] używa ciągów pseudocykli z własnością prefiksów, metoda Chistova [9] używa tablic ciągów marszrut, a algorytm Csanky'ego ściśle związany jest z lematem Leveriera [18], który możemy zinterpretować używając pętli i częściowych pokryć cyklami.

Pokażemy również jak możemy wykorzystać nasze wcześniejsze wyniki do implementacji algorytmu Samuelsona oraz algorytmu Chistova w czasie $\tilde{O}(n^{3.03})$.

7.2 Metoda Samuelsona

Powodem przejścia w algorytmie Mahajana i Vinaya od sumy wag pokryć cyklami do sumy wag ciągów pseudocykli było to, że wyznaczanie sumy wag ciągów pseudocykli można było wykonać efektywnie, przy czym końcowy wynik nie ulegał zmianie. Aczkolwiek istnieją zbiory ciągów pseudocykli, które możemy wyeliminować z powyższej sumy bez utraty tej efektywności. Jeden z takich zbiorów wynika z poniższych rozważań.

W pokryciu cyklami każdy wierzchołek jest pokryty dokładnie jeden raz. Przypuśćmy, że numerujemy wierzchołki w kolejności w jakiej są one odwiedzane w pokryciu cyklami (przyjmując jako pierwszy wierzchołek cyklu jego lidera). Jeśli wierzchołek h pojawia się jako lider cyklu wówczas wszystkie wierzchołki w tym oraz kolejnych cyklach są większe niż h. Zatem wierzchołki o numerach niższych od h musiały już zostać odwiedzone w poprzednich cyklach. Wynika z tego, że co najmniej $h-1$ wierzchołków, stąd i $h-1$ krawędzi, musiało już być pokrytych.

Możemy więc żądać aby nasze ciągi pseudocykli również posiadały tę własność. Sformalizujmy *własność prefiksów*: powiemy, że ciąg pseudocykli $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ spełnia własność prefiksów jeśli dla $1 \leq r \leq k$ całkowita długość ciągów pseudocykli $C_1, C_2, \ldots, C_{r-1}$ wynosi co najmniej $h(C_r) - 1$. Podobnie własność prefiksów możemy zdefiniować dla częściowych pokryć cyklami i częściowych ciągów pseudocykli: l-ciąg pseudocykli $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ spełnia własność prefiksów jeśli dla $1 \leq r \leq k$

$$\sum_{t=1}^{r-1} |C_t| \geq h(C_r) - 1 - (n - l).$$

Okazuje się, że odwzorowanie skonstruowane w dowodzie twierdzenia 3.2.1 dla ciągów pseudocykli działa nawet dla tego ograniczonego zbioru.

Twierdzenie 7.2.1.

$$c_l = \sum_{\mathcal{C}} sgn(\mathcal{C}) \cdot weight(\mathcal{C}, A),$$

gdzie sumowanie przebiega po wszystkich *l-ciągach pseudocykli $\mathcal{C}$ spełniających własność prefiksów.*

Dowód. W [52] Valiant zaobserwował, że ciągi pseudocykli spełniające własność prefiksów są wyrażeniami obliczanymi w metodzie Samuelsona obliczania wielomianu charakterystycznego $\Phi_A(\lambda)$. Stąd poprawność powyższego twierdzenia wynika z poprawności metody Samuelsona. Natomiast poprawność metody Samuelsona tradycyjnie dowodzona jest przy użyciu algebry liniowej.

Przedstawimy tu prosty alternatywny dowód kombinatoryczny. Zauważmy, że odwzorowanie skonstruowane w dowodzie twierdzenia 3.2.1 ze strony 24 odwzorowuje ciągi pseudocykli spełniające własność prefiksów w ciągi pseudocykli spełniające własność prefiksów.

Niech $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ będzie l-ciągiem pseudocykli spełniającym własność prefiksów. Dodajemy kolejne pseudocykle $C_k, C_{k-1} \ldots$ dopóki nie nastąpi powtórzenie jakiegoś elementu. Przypuśćmy, że C_{m+1}, C_{m+2}, ..., C_k jest ciągiem rozłącznych cykli ale ciąg $C_m, C_{m+1} \ldots, C_k$ zawiera już powtórzony element, gdzie $1 \leq m \leq k$. Niech $C_m = (c_0, c_1, \ldots, c_i)$ i niech c_j będzie pierwszym elementem na pseudocyklu C_m, który jest albo

(1) równy poprzedzającemu go elementowi c_l na C_m $(1 \leq l < j)$ albo

(2) równy elementowi występującemu w cyklu C_s będącym jednym z cykli C_{m+1}, C_{m+2}, ..., C_k.

Dokładnie jeden z tych dwóch przypadków musi mieć miejsce.

W przypadku (1) usuwamy cykl $C = (c_l = c_j, c_{l+1}, \ldots, c_{j-1})$ z pseudocyklu C_m i wkładamy go jako osobny cykl w $\mathcal{C}$ na właściwą pozycję (niech to będzie pozycja s) tak, aby zachowane było ściśle rosnące uporządkowanie liderów. Ta operacja nie powoduje zmiany wartości $h(C_m)$. Niech $\mathcal{C}' = \mathcal{D} = (D_1, D_2, \ldots, D_{k+1})$ będzie powstałym l-ciągiem pseudocykli, $D_t = C_t$ dla $t = 1, 2, \ldots, m-1$ oraz dla $t = m+1, m+2, \ldots, s-1$, $D_m = C_m - C$, $D_s = C$ a $D_{t+1} = C_t$ dla $t = s, s+1, \ldots, k$. Musimy pokazać, że $\mathcal{D}$ spełnia własność prefiksów.

Dla $r = 1, 2, \ldots, m$ oraz $r = s+1, s+2, \ldots, k+1$ warunek $\sum_{t=1}^{r-1} |D_t| \geq h(D_r) - 1 - (n-l)$ zachodzi, ponieważ $\mathcal{C}$ posiadał własność prefiksów.

Niech $r = m+1, m+2, \ldots, s$. Z konstrukcji C_m i ponieważ $m+1 \leq r$ wiemy, że $D_r, D_{r+1}, \ldots, D_{k+1}$ tworzą częściowe pokrycie cyklami, których liderzy są w ściśle rosnącym porządku. Zatem wierzchołki pokryte przez $D_r, D_{r+1}, \ldots, D_{k+1}$ muszą być różne oraz wszystkie mieć numery nie mniejsze niż $h(D_r)$.

Ale wierzchołków spełniających te ograniczenia mamy jedynie $n - h(D_r) + 1$. Stąd

$$\sum_{t=r}^{k+1} |D_t| \leq n - h(D_r) + 1, \; l - \sum_{t=1}^{r-1} |D_t| \leq n - h(D_r) + 1, \; \sum_{t=1}^{r-1} |D_t| \geq h(D_r) - 1 - (n-l)$$

i $\mathcal{D}$ spełnia własność prefiksów.

W przypadku (2) przenosimy cykl C_s do pseudocyklu C_m tuż za miejscem pojawienia się elementu c_j. Ponieważ długość pseudocyklu C_m jedynie zwiększa się, zatem własność prefiksów pozostaje spełniona.

Zatem podczas sumowania po wszystkich l-ciągach pseudocykli spełniających własność prefiksów jedynymi l-ciągami pseudocykli które się nie redukują wzajemnie są l-pokrycia cyklami, co kończy dowód naszego twierdzenia. □

7.2.1 Algorytm Używający Ciągów Pseudocykli z Własnością Prefiksów

Aby obliczyć współczynnik c_l wielomianu $\Phi_A(\lambda)$ musimy sumować po wszystkich l-ciągach pseudocykli spełniających własność prefiksów. Moglibyśmy zmodyfikować algorytm dynamiczny opisany w 3.2.5. Zamiast tego przedstawimy jednak inny algorytm działający również w czasie $O(n^4)$, powód stanie się za chwilę jasny.

Przyjmiemy teraz konwencję, że pseudocykle mogą mieć długość 0. Wówczas każdy l-ciąg pseudocykli $\mathcal{C} = (C_1, C_2, \ldots, C_k)$ zawiera dokładnie jeden pseudocykl C_h z liderem h, dla każdego $h = 1, 2, \ldots, n$. Możemy więc napisać $\mathcal{C} = (C_1, C_2, \ldots, C_n)$.

Wagą ze znakiem pseudocyklu C nazwiemy $sweight(C, A) = -weight(C, A)$, jeśli C posiada niezerową długość, oraz $sweight(C, A) = 1$ w przeciwnym wypadku. *Wagą ze znakiem* l-ciągu pseudocykli $\mathcal{C} = (C_1, C_2, \ldots, C_n)$ nazwiemy wyrażenie

$$sweight(\mathcal{C}, A) = \prod_{i=1}^{n} sweight(C_i, A).$$

Wówczas

$$sgn(\mathcal{C})weight(\mathcal{C}, A) = sweight(\mathcal{C}, A)$$

i z twierdzenia 7.2.1

$$c_l = \sum_{\mathcal{C}} sweight(\mathcal{C}, A),$$

gdzie sumowanie przebiega po wszystkich l-ciągach pseudocykli $\mathcal{C}$ spełniających własność prefiksów.

Powiemy, że ciąg nieujemnych liczb całkowitych $l_1, l_2, \ldots, l_n$ spełnia *własność l-prefiksów* jeśli

1. $\sum_{t=1}^{n} l_t = l$,
2. $\sum_{t=1}^{r-1} l_t \geq r - 1 - (n - l)$ dla $r = 1, 2, \ldots, n$.

Takie ciągi są „dopuszczalne” jako długości pseudocykli w ciągach pseudocykli, które będziemy konstruować.

Grupujemy ciągi pseudocykli z własnością prefiksów bazując na długościach poszczególnych pseudocykli. W ciągu pseudocykli z własnością prefiksów $\mathcal{C}$, jeśli długość pseudocyklu C_h z liderem h jest równa l_h, wówczas dowolny pseudocykl z liderem h i długości l_h może zastąpić C_h w $\mathcal{C}$ wciąż dając ciąg pseudocykli z własnością prefiksów. Przez $z(h,p)$ oznaczmy całkowitą wagę ze znakiem wszystkich pseudocykli, które mają wierzchołek h jako lidera oraz długość p. Wówczas

$$c_l = \sum_{l_1,l_2,\ldots,l_n} \prod_{i=1}^{n} z(i,l_i),$$

gdzie sumowanie przebiega po wszystkich ciągach nieujemnych liczb całkowitych $l_1, l_2, \ldots, l_n$ spełniających własność l-prefiksów.

Aby wyznaczać współczynniki c_l efektywnie ustawiamy wartości $z(h,p)$ we właściwe miejsca w ciąg macierzy $B_1, B_2, \ldots, B_n$. Macierz B_k zawiera współczynniki postaci $z(k,p)$. Ponieważ rozważamy jedynie ciągi spełniające własność l-prefiksów wystarcza brać $z(k,p)$ dla $p \leq n-k+1$. Macierz B_k jest wymiaru $(n-k+2)\times(n-k+1)$ i zawiera $z(k,p)$ na p-ej dolnej przekątnej jak pokazano poniżej.

$$B_k = \begin{bmatrix} z(k,0) & 0 & 0 & \ldots & 0 & 0 \\ z(k,1) & z(k,0) & 0 & \ldots & 0 & 0 \\ z(k,2) & z(k,1) & z(k,0) & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ z(k,n-k) & z(k,n-k-1) & z(k,n-k-2) & \ldots & z(k,1) & z(k,0) \\ z(k,n-k+1) & z(k,n-k) & z(k,n-k-1) & \ldots & z(k,2) & z(k,1) \end{bmatrix}$$

Teraz z równości na c_l dostajemy, że

$$c_l = \sum_{l+1=j_0\geq j_1\geq\ldots\geq j_n=1} \prod_{i=1}^{n} B_i[j_{i-1}, j_i] = (\prod_{i=1}^{n} B_i)[l+1, 1],$$

gdzie ciągi $j_0 - j_1, j_1 - j_2, \ldots, j_{n-1} - j_n$ spełniają własność l-prefiksów. Równoważnie możemy zapisać powyższą równość jako

$$[c_0, c_1, \ldots, c_n]^T = \prod_{i=1}^{n} B_i.$$

Zgodnie z oznaczeniami stosowanymi w pracy dostajemy, że dla $p > 1$

$$z(k,p) = -\hat{A}[k,p] = -a_k A_k^{p-2} a^k,$$

gdzie macierz $\hat{A}$ oznacza skośne domknięcie macierzy A zdefiniowane w części 3.2.3. Ostatecznie B_k ma więc postać:

$$B_k = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ -A[k,k] & 1 & 0 & \dots & 0 & 0 \\ -a_k a^k & -A[k,k] & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -a_k A_k^{n-k-2} a^k & -a_k A_k^{n-k-3} a^k & -a_k A_k^{n-k-4} a^k & \dots & -A[k,k] & 1 \\ -a_k A_k^{n-k-1} a^k & -a_k A_k^{n-k-2} a^k & -a_k A_k^{n-k-3} a^k & \dots & -a_k a^k & -A[k,k] \end{bmatrix}$$

Przedstawiona metoda obliczania wielomianu $\Phi_A(\lambda)$ jest dokładnie metodą Samuelsona [40, 18, 6, 52]. Samuelson otrzymał powyższy wzór używając twierdzenia Laplace dla macierzy $\lambda \cdot I_n - A$, podczas gdy my dostajemy go posługując się ciągami pseudocykli z własnością prefiksów.

Taka interpretacja algorytmu Samuelsona-Berkowitza pochodzi od Valianta [52], natomiast kombinatoryczny dowód poprawności (dowód twierdzenia 7.2.1) znaleźć można w [36].

7.2.2 Nasza Wersja Algorytmu Samuelsona

Przedstawimy tutaj nową wersję algorytmu Samuelsona obliczania wyznacznika działającą w czasie $\tilde{O}(n^{3.03})$, która będzie korzystała z algorytmu obliczania macierzy $\hat{A} = SkewClosure(A)$ opisanego w części 3.3.

Rozpoczniemy od wprowadzenia ograniczeń do definicji $Knapsack(A)$, tak aby uwzględniała ona własność prefiksów. Przez $PrefixKnapsack(A)$ oznaczamy sumę wszystkich iloczynów postaci

$$A[h_1, j_1] \cdot A[h_2, j_2] \cdot \ldots \cdot A[h_k, j_k],$$

gdzie

$$1 \leq h_1 < h_2 < \ldots < h_k \leq n,\ 1 \leq j_1, j_2, \ldots, j_k \leq n,\ \sum_{t=1}^{k} j_t = n$$

oraz $\sum_{t=1}^{r-1} j_t \geq h_r - 1$ dla $r = 1, 2, \ldots, k$.

Algorytm obliczania wartości $PrefixKnapsack(A)$ jest tylko niewielką modyfikacją algorytmu dynamicznego obliczania $Knapsack(A)$ opisanego w części 3.2.7 ze strony 28 działającego w czasie $O(n^3)$. Używamy tablicy $Table[h, j]$ takiej, że

$$PrefixKnapsack(A) = Table[n, n].$$

Inicjujemy początkowe wartości $Table[1,j] = A[1,j]$, dla wszystkich $1 \leq j \leq n$, a pozostałe elementy tablicy $Table[h,j]$, dla $2 \leq h \leq n$ oraz $1 \leq j \leq n$, wyznaczamy zgodnie ze wzorem

$$Table[h,j] \;=\; (\sum_{p=h-1}^{j-1} Table[h-1,p] \cdot A[h,j-p]) + Table[h-1,j].$$

W powyższym wzorze p nie przebiega już jak w przypadku $Knapsack(A)$ od 0, ale od $h-1$ ponieważ własność prefiksów wymusza na nas aby w chwili pojawienia się wierzchołka h jako lidera w ciągu pseudocykli $\mathcal{C}$ co najmniej $h-1$ wierzchołków, stąd i $h-1$ krawędzi, było już zawartych w $\mathcal{C}$.

Ostatecznie dostajemy poniższe twierdzenie, którego poprawność wynika wprost z poprawności twierdzenia 7.2.1 oraz z lematu 3.3.2.

Twierdzenie 7.2.2. *Wyznacznik macierzy A spełnia równanie*

$$det(A) \;=\; (-1)^n \cdot PrefixKnapsack(-SkewClosure(A)).$$

i może on zostać obliczony w czasie $\tilde{O}(n^{3.03})$ bez wykonywania operacji dzielenia.

7.3 Algorytm Chistova

Zmienimy teraz podejście zaprezentowane w części 7.2 i uogólnimy pojęcie ciągów pseudocykli.

Po pierwsze eliminujemy warunek, że lider pseudocyklu ma być odwiedzony tylko jeden raz. To daje nam bardziej ogólne pojęcie pseudocykli które nazywać będziemy *marszrutami*. Ustalamy kanoniczną reprezentację w której żądamy aby pierwszym wierzchołkiem marszruty był jej lider. Ponieważ lider może pojawiać się wielokrotnie w marszrucie dwie reprezentacje rozpoczynające się od jego różnych wystąpień uważamy za różne. Na przykład, marszrutę $(2,5,3,2,4,6)$ uważamy za różną od $(2,4,6,2,5,3)$.

Po drugie rozważamy już nie tylko ciągi marszrut ale uporządkowane listy, bądź *tablice*, takich ciągów. Wewnątrz ciągu liderzy marszrut są uporządkowani w ściśle rosnącym porządku. Ale nie ma ograniczenia na uporządkowanie ciągów wewnątrz tablic. Oznacza to, że dla tego samego multizbioru ciągów marszrut różne ich uporządkowania dają nam tablice, które uważamy za różne.

Po trzecie parzystość tablicy ciągów marszrut zależeć będzie od liczby tworzących ją ciągów, nie natomiast od liczby marszrut. Ciąg pseudocykli jest więc tablicą ciągów

marszrut taką, że każdy ciąg zawiera pojedynczą marszrutę będącą pseudocyklem oraz wewnątrz tablicy liderzy ciągów są uporządkowani w ściśle rosnącym porządku.

Zapiszemy to bardziej formalnie. *Marszrutą* długości k nazwiemy uporządkowany ciąg wierzchołków $\pi = (v_0, v_1, \ldots, v_k)$ taki, że $v_0 = v_k = min\{v_0, v_1, \ldots, v_k\}$ oraz $(v_{i-1}, v_i) \in E$, dla każdego $i = 1, 2, \ldots, k$. Wierzchołek v_0 nazywamy *liderem* marszruty π i oznaczamy go przez $h(\pi)$. *Długość* marszruty π jest liczbą jej krawędzi i oznaczamy ją przez $|\pi|$. *Wagą* marszruty π nazywamy iloczyn wag tworzących ją krawędzi

$$weight(\pi, A) \;=\; A[v_0, v_1] \cdot A[v_1, v_2] \cdot \ldots \cdot A[v_{k-1}, v_k].$$

j-ciągiem marszrut $\mathcal{T}$ nazywamy uporządkowany ciąg marszrut $\mathcal{T} = (T_1, T_2, \ldots, T_k)$ taki, że

$$h(T_1) < h(T_2) < \ldots < h(T_k),\ |T_1| + |T_2| + \ldots + |T_k| = j.$$

Wagą ciągu marszrut nazywamy iloczyn wag tworzących go marszrut

$$weight(\mathcal{T}, A) = \prod_{i=1}^{k} weight(T_i, A)$$

a jego długość $|\mathcal{T}| = j$.

l-tablica ciągów marszrut $\mathcal{F}$ jest uporządkowanym ciągiem ciągów marszrut $\mathcal{F} = (\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r)$ takim, że $|\mathcal{T}_1| + |\mathcal{T}_2| + \ldots + |\mathcal{T}_r| = l$.

Wagą tablicy ciągów marszrut $\mathcal{F}$ jest iloczyn wag tworzących ją ciągów

$$weight(\mathcal{F}, A) = \prod_{i=1}^{r} weight(\mathcal{T}_i, A)$$

a jej *znak* równy jest $sgn(\mathcal{F}) = (-1)^r$.

Następujące twierdzenie pokazuje, że nawet tablice ciągów marszrut mogą być użyte do policzenia wielomianu charakterystycznego macierzy A.

Twierdzenie 7.3.1.

$$c_l = \sum_{\mathcal{F}} sgn(\mathcal{F}) \cdot weight(\mathcal{F}, A),$$

gdzie sumowanie przebiega po wszystkich *l-tablicach ciągów marszrut* $\mathcal{F}$.

W [36] znaleźć można kombinatoryczny dowód poprawności twierdzenia 7.3.1, polegający na konstrukcji odwzorowania określonego na zbiorze l-tablic ciągów marszrut którego jedynymi punktami stałymi są l-ciągi pseudocykli. Pozostałe l-tablice ciągów marszrut odwzorowywane są w l-tablice ciągów marszrut o tej samej wadze ale

przeciwnym znaku. Ponieważ wiemy, że l-ciągi pseudocykli, które nie są pokryciami cyklami również mają zerowy wkład w powyższą sumę, zatem suma przebiegająca po wszystkich l-tablicach ciągów marszrut jest dokładnie równa współczynnikowi c_l wielomianu charakterystycznego macierzy A.

7.3.1 Algorytm Używający Tablic Ciągów Marszrut

Pokażemy, jak zgrupować l-tablice ciągów marszrut aby dostać wyrażenie proste do obliczenia. Algorytm, który teraz opiszemy działał będzie w czasie $O(n^4)$. Musimy wyznaczyć wkład w sumę wszystkich l-tablic ciągów marszrut. Będzie to prostsze do osiągnięcia jeśli podzielimy ten wkład na l grup w zależności od tego jak wiele krawędzi znajduje się w pierwszym ciągu marszrut w tablicy. Grupa j zawiera te l-tablice ciągów marszrut które są postaci $\mathcal{F} = (\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r)$, gdzie $|\mathcal{T}_1| = j$.

Wtedy $\mathcal{F}' = (\mathcal{T}_2, \mathcal{T}_3, \ldots, \mathcal{T}_r)$ tworzy $(l-j)$-tablice ciągów marszrut,

$$sgn(\mathcal{F}) = -sgn(\mathcal{F}'),\ weight(\mathcal{F}, A) = weight(\mathcal{T}_1, A) weight(\mathcal{F}', A).$$

Zatem wkład tej grupy w c_l, który oznaczamy przez $c_l(j)$ można przedstawić jako

$$c_l(j) = \sum_{\mathcal{T}, \mathcal{F}'} -sgn(\mathcal{F}') weight(\mathcal{F}', A) weight(\mathcal{T}, A)$$

$$= -(\sum_{\mathcal{T}} weight(\mathcal{T}, A))(\sum_{\mathcal{F}'} sgn(\mathcal{F}') weight(\mathcal{F}', A)) = -d_j c_{l-j},$$

gdzie sumowanie przebiega po wszystkich j-ciągach marszrut $\mathcal{T}$ oraz $(l-j)$-tablicach ciągów marszrut $\mathcal{F}'$ a d_j jest sumą wag wszystkich j-ciągów marszrut.

Musimy zatem wyznaczyć d_j. Łatwo zobaczyć, że $A^l[1,1]$ jest równe sumie wag wszystkich marszrut długości l z liderem 1. Aby wyznaczyć w podobny sposób tę sumę dla lidera k musimy rozważyć podgraf grafu G_A indukowany przez wierzchołki $k, k+1, \ldots, n$, który będzie odpowiadał macierzy A_{k-1}. Niech $y(l,k)$ oznacza sumę wag wszystkich l-marszrut z liderem k. Wówczas $y(l,k) = A^l_{k-1}[k,k]$.

Waga j-ciągu marszrut $\mathcal{T}$ może zostać podzielona na n części, gdzie k-ta część ma wartość 1 jeśli $\mathcal{T}$ nie zawiera marszruty z liderem k w przeciwnym wypadku jest ona równa wadze tej marszruty. Mamy

$$d_j = \sum_{0 \le l_i \le j:\ l_1 + \ldots + l_n = j} \prod_{i=1}^{n} y(l_i, i) = \sum_{0 \le l_i \le j:\ l_1 + \ldots + l_n = j} \prod_{i=1}^{n} A^{l_i}_{i-1}[i,i].$$

Zdefiniujmy szereg potęgowy $D(x) = \sum_{j=0}^{\infty} d_j x^j$. Wtedy, używając powyższego wzoru na d_j możemy zapisać

$$D(x) = (\sum_{l=0}^{\infty} x^l A_0^l[1,1])(\sum_{l=0}^{\infty} x^l A_1^l[2,2]) \ldots (\sum_{l=0}^{\infty} x^l A_{n-1}^l[n,n]).$$

Ponieważ jesteśmy zainteresowani jedynie tymi wartościami d_j dla których $j \leq n$ możemy zignorować jednomiany stopnia wyższego niż n. To wymaga od nas wyznaczenia pierwszych $n+1$ współczynników $D(x)$ co możemy osiągnąć używając potęgowania macierzy oraz posługując się arytmetyką na wielomianach. Teraz już wartość c_l może być obliczona indukcyjnie korzystając z równości

$$c_l = \sum_{j=1}^{l} c_l(j) = \sum_{j=1}^{l} -d_j c_{l-j} \tag{7.1}$$

Ale zaprezentowany tu algorytm odpowiada już algorytmowi Chistova [9]. Jedyna różnica jest taka, że Chistov rozpoczął od kilku algebraicznych równości i manipulując nimi używając arytmetyki wielomianów dostał powyższe sformułowanie, podczas gdy tutaj rozpoczynamy od tablic ciągów marszrut, które mają kombinatoryczne definicje i grupując je we właściwy sposób osiągamy dokładnie ten sam rezultat.

7.3.2 Nasza Wersja Algorytmu Chistova

Podobnie jak w przypadku algorytmu Samuelsona i tutaj możemy wykorzystać algorytm obliczania macierzy $\hat{A} = SkewClosure(A)$ aby dostać implementację algorytmu Chistova obliczania wielomianu charakterystycznego w czasie $\tilde{O}(n^{3.03})$.

Najpierw wyznaczymy wartości $y(l,h)$, czyli sumę wag wszystkich l-marszrut z liderem h. Niech $Tour(\hat{A})$ będzie macierzą powstałą z macierzy $\hat{A}$ taką, że

$$Tour(\hat{A})[h,l] = \sum_{0 \leq k_i \leq l:\ k_1+k_2+\ldots+k_n=l} \prod_{i=1}^{n} \hat{A}[h,k_i],$$

gdzie jak pamiętamy $\hat{A}[h,k]$ to suma wag wszystkich pseudocykli z liderem h długości k. Ponieważ każda l-marszruta z liderem h jest złożeniem pewnej liczby pseudocykli z liderem h i sumarycznej długości l mamy, że

$$y(l,h) = Tour(\hat{A})[h,l] = \sum_{0 \leq k_i \leq l:\ k_1+k_2+\ldots+k_n=l} \prod_{i=1}^{n} \hat{A}[h,k_i]. \tag{7.2}$$

Niech teraz $\mathcal{A}_h(x)$, dla $h = 1, 2, \ldots, n$, będzie wielomianem stopnia n zdefiniowanym następująco

$$\mathcal{A}_h(x) = 1 + \hat{A}[h,1] \cdot x + \hat{A}[h,2] \cdot x^2 + \ldots + \hat{A}[h,n] \cdot x^n.$$

Wówczas używając wzoru 7.2 na $y(l,h)$ otrzymujemy, że

$$(\mathcal{A}_h(x))^n = 1 + y(1,h) \cdot x + y(2,h) \cdot x^2 + \ldots + y(n,h) \cdot x^n + x^{n+1} \cdot F(x),$$

gdzie $F(x)$ jest wielomianem stopnia $n^2 - n - 1$.

Ponieważ jesteśmy zainteresowani jedynie tymi wartościami $y(l,h)$ dla których $l \leq n$ wszystkie obliczenia możemy wykonywać w pierścieniu reszt modulo x^{n+1}. Iloczyn dwóch wielomianów stopnia n wyznaczamy korzystając z klasycznego algorytmu w czasie $O(n^2)$. Używając algorytmu szybkiego potęgowania pierwsze $n+1$ współczynników wielomianu $(\mathcal{A}_h(x))^n$ obliczymy w czasie $\tilde{O}(n^2)$. A zatem wszystkie n^2 wartości $y(l,h)$, dla $l, h = 1, 2, \ldots, n$ możemy wyznaczyć w czasie $\tilde{O}(n^3)$.

Do obliczenia wartości d_j, sumy wag wszystkich j-ciągów marszrut, $j = 1, 2, \ldots, n$ skorzystamy z algorytmu dynamicznego działającego czasie $O(n^3)$ i opisanego w części 3.2.7. Mamy bowiem

$$d_j = \mathcal{K}_j(Tour(\hat{A})).$$

Ostatecznie dostajemy poniższe twierdzenie, którego poprawność wynika wprost z poprawności twierdzenia 7.3.1, lematów 3.2.2, 3.3.2 oraz równości 7.1.

Twierdzenie 7.3.2. *Współczynniki wielomianu charakterystycznego $\Phi_A(\lambda)$ macierzy A spełniają równanie*

$$c_l \;=\; -\sum_{j=1}^{l} c_{l-j} \cdot \mathcal{K}_j(Tour(SkewClosure(A)))$$

i mogą one zostać obliczone w czasie $\tilde{O}(n^{3.03})$ bez wykonywania operacji dzielenia.

7.4 Algorytm Csanky'ego

Teraz rozważmy kolejne uogólnienie pojęcia cyklu: osłabiamy nasz warunek, że marszruta musi rozpoczynać się od pojawienia się wierzchołka o minimalnym numerze. Jedyne czego będziemy oczekiwać po marszrucie to aby była ona zamkniętą ścieżką, takie zamknięte ścieżki nazywać będziemy *pętlami*. Formalnie, pętlą długości k w wierzchołku v nazywamy zamkniętą ścieżkę długości k z v do v, tzn. uporządkowany

ciąg wierzchołków $\pi = (v_0, v_1, \ldots, v_k)$ taki, że $v = v_0 = v_k$ oraz $(v_{i-1}, v_i) \in E$, dla każdego $i = 1, 2, \ldots, k$.

Wagą pętli π nazywamy iloczyn wag tworzących ją krawędzi

$$weight(\pi, A) \;=\; A[v_0, v_1] \cdot A[v_1, v_2] \cdot \ldots \cdot A[v_{k-1}, v_k].$$

Mając tak osłabione pojęcie pętli wprowadzimy teraz ograniczenia na sposoby w jakie pętle mogą być organizowane w ciągi. Otóż pętle będą mogły być łączone jedynie z częściowymi pokryciami cyklami. Pokażemy następujący wzór na współczynniki wielomianu $\Phi_A(\lambda)$.

Twierdzenie 7.4.1. *Dla każdego $k = 1, 2, \ldots, n$*

$$kc_k + \sum_{j=1}^{k} c_{k-j}(\sum_{L} weight(L, A)) = 0,$$

gdzie sumowanie przebiega po wszystkich pętlach L długości j w grafie G_A.

Łatwo zobaczyć, że $A^j[i, i]$ to suma wag wszystkich ścieżek długości j z wierzchołka i do i w grafie G_A. A takie ścieżki to oczywiście pętle, stąd $\sum_{i=1}^{n} A^j[i, i]$ to suma wag wszystkich pętli długości j w G_A. Mamy

$$\sum_{i=1}^{n} A^j[i, i] = tr(A^j) = s_j,$$

gdzie $tr(A^j)$ to ślad macierzy A^j.

A zatem powyższe twierdzenie jest po prostu innym zapisem lematu Leveriera [18, 26], który zwykle można znaleźć w poniższej postaci.

Lemat 7.4.1. *Współczynniki wielomianu charakterystycznego macierzy A spełniają następujące równanie:*

$$\begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ s_1 & 2 & 0 & \ldots & 0 & 0 \\ s_2 & s_1 & 3 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ s_{n-2} & s_{n-3} & s_{n-4} & \ldots & n-1 & 0 \\ s_{n-1} & s_{n-2} & s_{n-3} & \ldots & s_1 & n \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \\ c_n \end{bmatrix} = - \begin{bmatrix} s_1 \\ s_2 \\ s_3 \\ \vdots \\ s_{n-1} \\ s_n \end{bmatrix}$$

Dowód. Przedstawimy kombinatoryczny dowód lematu Leveriera. Rozważmy k-te równanie

$$kc_k + \sum_{j=1}^{k} s_j c_{k-j} = 0,$$

gdzie $c_0 = 1$. Składniki sumy $\sum_{j=1}^{k} s_j c_{k-j}$ odpowiadają pętlą długości j oraz częściowym pokryciom cyklami długości $k - j$. Pętle wnoszą do powyższej sumy jedynie wagę, ale nie znak, podczas gdy częściowe pokrycia cyklami zarówno wagę jak i znak. Pokażemy teraz jak następuje skracanie w tym zbiorze.

Niech S będzie pętlą długości j oraz niech $\mathcal{C}$ będzie częściowym pokryciem cyklami pokrywającym $k - j$ wierzchołków. Możliwe są dwa przypadki:

(1) pętla S jest cyklem, rozłącznym z wszystkimi cyklami z $\mathcal{C}$,

(2) S i $\mathcal{C}$ nie mogą zostać złączone w częściowe pokrycie cyklami długości k.

W przypadku (1) S może zostać złączone z $\mathcal{C}$ tworząc częściowe pokrycie cyklami $\mathcal{C}'$ długości k o wadze $weight(S, A)weight(\mathcal{C}, A)$ i znaku $-sgn(\mathcal{C})$. A ten składnik skróci się z jedną kopią $\mathcal{C}'$ pochodzącą z części kc_k.

Pozostaje nam jeszcze $k-1$ kopii $\mathcal{C}'$. Zauważmy, że jeśli $\mathcal{C}' = (C_1, C_2, \ldots, C_l)$, wówczas każdy z cykli C_i długości l_i może zostać wybrany do utworzenia podziału na pętlę $S = C_i$ i częściowe pokrycie cyklami $\mathcal{C} = \mathcal{C}' - C_i$, które będzie odpowiadało składnikowi z $s_{l_i} c_{k-l_i}$.

Co więcej każdy cykl C_i można zapisać jako pętlę na l_i sposobów w zależności od wierzchołka startowego. A zatem częściowe pokrycie cyklami $\mathcal{C}'$ może zostać przedstawione na $\sum_{i=1}^{l} l_i = k$ różnych sposobów jako para pętla-częściowe pokrycie cyklami.

W przypadku (2) gdy S i $\mathcal{C}$ nie mogą zostać złączone w częściowe pokrycie cyklami długości k rozpoczynamy przeglądanie pętli S dopóki nie nastąpi jedna z następujących możliwości: (a) pętla S zawiera wierzchołek v z $\mathcal{C}$ lub (b) natrafiamy na drugie wystąpienie wierzchołka v w S. Dokładnie jedna z tych dwóch możliwości nastąpi jako pierwsza.

Przypuśćmy, że ma miejsce (a) i niech v znajduje się w cyklu C długości l zawartym w pokryciu $\mathcal{C}$. Rozważmy nową parę $S', \mathcal{C}'$, która powstanie w wyniku usunięcia cyklu C z $\mathcal{C}$ i dołączenia go do S za pierwszą pozycją wierzchołka v. Taka para wnosi do $s_{j+l} c_{k-j-l}$ składnik o takiej samej wadze ale przeciwnym znaku niż para $S, \mathcal{C}$ a zatem składniki im odpowiadające w sumie wzajemnie się zredukują.

A teraz niech zachodzi (b), S zawiera cykl C długości l który jest rozłączny z wszystkimi cyklami z $\mathcal{C}$. Rozważmy nową parę $S', \mathcal{C}'$, która powstanie w wyniku usunięcia cyklu C z S i wstawienia go na właściwą pozycję do $\mathcal{C}$. Taka para wnosi do $s_{j-l} c_{k-j+l}$ składnik o takiej samej wadze ale przeciwnym znaku niż para $S, \mathcal{C}$ zatem tak jak poprzednio nastąpi ich redukcja. Łatwo sprawdzić, że operacje wykonywane w przypadkach (a) i (b) są operacjami wzajemnie odwrotnymi. □

Każdy algorytm obliczania współczynników wielomianu $\Phi_A(\lambda)$ używający lematu Leveriera w istocie wykorzystuje opisane powyżej skracanie pomiędzy pętlami i częściowymi pokryciami cyklami do wyznaczenia sumy, która odpowiada dokładnie sumie wag częściowych pokryć cyklami długości l. Wprost nasuwającym się sekwencyjnym algorytmem jest algorytm, który najpierw dla każdego j wyznacza sumę s_j, wag wszystkich pętli długości j w G_A używając do tego celu potęgowania macierzy lub programowania dynamicznego a później oblicza $c_1, c_2, \ldots, c_n$ w tej kolejności posługując się opisaną powyżej rekurencją. Implementacja Csanky'ego [13] korzysta z odwrotności macierzy aby wyznaczyć współczynniki c_j równolegle znając wartości s_i.

Bibliografia

[1] A. V. Aho, J. E. Hopcroft and J. D. Ullman, *The Design and Analysis of Computer Algorithms,* Addison-Wesley, Reading, MA (1974)

[2] A. C. Aitken, *Determinants and Matrices,* Oliver and Boyd, Edinburgh (1946)

[3] M. Akra and L. Bazzi, *On the Solution of Linear Recurrence Equations,* Computational Optimization and Applications 10(**2**) (1998) 195–210

[4] W. Baur and V. Strassen, *The Complexity of Partial Derivatives,* Theoretical Comput. Sci. **22** (1983) 317–330

[5] C. Berge, *Two Theorems in Graph Theory,* Proc. Nat. Acad. Sci. USA (1957) 842–844

[6] S. J. Berkowitz, *On Computing the Determinant in Small Parallel Time Using a Small Number of Processors,* Information Processing Letters **18** (1984) 147–150

[7] A. Borodin and I. Munro, *The Computational Complexity of Algebraic and Numeric Problems,* Elsevier, New York (1975)

[8] R. A. Brualdi and H. Ryser, *Combinatorial Matrix Theory,* Cambridge University Press, Vol. 39 in Encyclopedia of Mathematics and its Applications (1991)

[9] A. L. Chistov, *Fast Parallel Calculation of the Rank of Matrices over a Field of Arbitrary Characteristic,* In Proc Int. Conf. Foundations of Computation Theory, LNCS 199 Springer (1985) 63–69

[10] D. Coppersmith, *Rectangular Matrix Multiplication Revisited,* J. Complex. **13(1)** (1997) 42–49

[11] D. Coppersmith and S. Winograd, *Matrix Multiplication via Arithmetic Progressions,* In Proceedings of the nineteenth Annual ACM Conference on Theory of Computing (1987) 1–6

[12] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Wprowadzenie do Algorytmów,* Wydawnictwa Naukowo-Techniczne, Warszawa (2004)

[13] L. Csanky, *Fast Parallel Inversion Algorithm,* SIAM J of Computing **5** (1976) 818–823

[14] C. Damm, $DET = L^{(\#L)}$, Technical Report Informatik-Preprint 8, Fachbereich Informatik der Humboldt-Universitat zu Berlin (1991)

[15] R. Diestel, *Graph Theory,*

[16] P. Doubilet, *On the Foundations of Combinatorial Theory,* VII: symmetric functions through the theory of distribution and occupancy, Stud. Appl. Math. Vol. LI No. 4 (1972) 377.396, reprinted in: Gian-Carlo Rota on Combinatorics, J.P. Kung Ed., Birkh¨auser (1995) 403–422.

[17] S. Even, *Graph Algorithms,* Computer Science Press, Rockville (1979)

[18] D. Fadeev and V. Fadeeva, *Computational Methods in Linear Algebra,* Freeman, San Francisco (1963)

[19] J. Geelen, *An Algebraic Matching Theory,* Combinatorica **20** (2000) 61–70

[20] R. Graham, D. Knuth and O. Patashnik, *Concrete Mathematics,* Addison-Wesley (1994)

[21] D. H. Greene and D. E. Knuth, *Mathematics for the Analysis of Algorithms,* Birkhauser (1982)

[22] F. Harary, *Graph Theory,* Reading, Mass. Addison-Wesley (1972)

[23] X. Huang and V. Y. Pan, *Fast Rectangular Matrix Multiplication and Applications,* Journal of complexity, **14(2)** (1998) 257–299

[24] M. Karpiński and W. Rytter, *Fast Parallel Algorithms for Graph Matching Problems,* Oxford University Press, Oxford (1998)

[25] D. Knuth, *Overlapping Pfaffians,* Electron. J. Comb. 3, 1996, No. 2, article R5, 13 pp. Printed version: J. Comb. 3, **2** (1996) 147–159

[26] D. Kozen, *The Design and Analysis of Algorithms,* Springer-Verlag, New York (1992)

[27] C. Krattenthaler, *Determinant Calculus,* Séminaire Lotharingien de Combinatoire B42 (1999) 67

[28] C. H. Little, *Kasteleyn's Theorem and Arbitrary Graphs,* Canadian Journal of Mathematics **25** (1973) 758–764

[29] L. Lovász, *On Determinants, Matchings and Random Algorithms,* In L. Budach, editor, Fundamentals of Computation Theory, Akademie-Verlag (1979) 565–574

[30] L. Lovász and M. Plummer, *Matching Theory,* Ann. Discr. Math, Vol. 29. North-Holland Mathematics Studies, Vol. 121, Amsterdam (1986)

[31] R. J. Lipton, D. J. Rose and R. E. Tarjan, *Generalized Nested Dissection,* SIAM J. Num. Anal. **16** (1979) 346–358

[32] R. J. Lipton and R. E. Tarjan, *A Separator Theorem for Planar Graphs,* SIAM J. Applied Math. (1979) 177–189

[33] M. Mahajan, P. Subramanya and V. Vinay, *A Combinatorial Algorithm for Pfaffians,* In Computing and Combinatorics. Proc. Fifth Annual International Conference. (COCOON '99), Tokyo, July 1999, ed. Takao Asano et al. Lecture Notes Computer Science 1627, Springer-Verlag (1999) 134–143

[34] M. Mahajan and V. Vinay, *A Combinatorial Algorithm for the Determinant,* In Proceedings of the Eight Annual ACM-SIAM Symposium on Discrete Algorithms, SODA97

[35] M. Mahajan and V. Vinay, *Determinant: Combinatorics, Algorithms and Complexity,* Chicago Journal of Theoretical Computer Science **5** (1997)

[36] M. Mahajan and V. Vinay, *Determinant: Old algorithms, New Insights,* SIAM J. Discrete Math. **12** (1999) 474–490

[37] S. Micali and V. V. Vazirani, *An $O(\sqrt{|V|}|E|)$ Algorithm for Finding Maximum Matching in General Graphs,* In Proceedings of the twenty first annual IEEE Symposium on Foundations of Computer Science (1980) 17–27

[38] G. Rote, *Division-Free Algorithms for the Determinant and the Pfaffian: Algebraic and Combinatorial Approaches,* In H. Alt, editor, Computational Discrete Mathematics: Advanced Lectures, volume LNCS 2122, Springer (2001) 119–135

[39] D. E. Rutherford, *The Cayley-Hamilton Theorem for Semi-Rings,* Proc. Roy. Soc. Edinburgh, Sect. A **66** (1964) 211–215

[40] P. A. Samuelson, *A Method of Determining Explicitly the Coefficients of the Characteristic Polynomial,* Ann. Math. Stat. **13** (1942) 424–429

[41] A. Schonhage and V. Strassen, *Schnelle Multiplikation grosser Zahlen,* Computing **7** (1971) 281–292

[42] D. Stanton and D. White, *Constructive Combinatorics,* Springer-Verlag, New York (1986)

[43] V. Strassen, *Algebraic Complexity Theory,* Handbook of Theoretical Computer Science, ed. J. van Leeuwen, volume A, Elsevier, Amsterdam (1990) 633–672

[44] V. Strassen, *Gaussian Elimination is Not Optimal,* Numerische Mathematik, **13** (1969) 354–356

[45] V. Strassen, *Vermeidung von Divisionen,* Journal of Reine U. Angew Math. **264** (1973) 182–202

[46] H. Straubing, *A Combinatorial Proof of the Cayley-Hamilton Theorem,* Discrete Math. **43** (1983) 273–279

[47] H. N. Tempereley, *Graph Theory and Applications,* Ellis Horwood, Chichester (1981)

[48] S. Toda, *Counting Problems Computationally Equivalent to the Determinant,* Manuscript (1991)

[49] A. Urbańska, *Application of Graph Separators to the Efficient Division-Free Computation of Determinant,* Praca przyjęta na konferencję SOFSEM 2009

[50] A. Urbańska, *Faster Combinatorial Algorithms for Determinant and Pfaffian,* ISAAC 2007, LNCS 4835 (2007) 599–608

[51] A. Urbańska, *Faster Combinatorial Algorithms for Determinant and Pfaffian,* Praca przyjęta do publikacji w czasopiśmie Algorithmica, publikacja elektroniczna dostępna na stronie: http://dx.doi.org/10.1007/s00453-008-9240-9 (2008)

[52] L. G. Valiant, *Why is Boolean Complexity Theory Difficult?* In: Boolean Function Complexity, ed. M. S. Paterson, LMS Lecture Notes Series, Vol. 169, Cambridge Univ. Press (1992) 84–94

[53] L. G. Valiant, *The Complexity of Computing the Permanent,* Theoretical Computer Science **8** (1979) 189–201

[54] V. Vinay, *Counting Auxiliary Pushdown Automata and Semi-Unbounded Arithmetic Circuits,* In Proc. 6th Structure in Complexity Theory Conference (1991) 270–284

[55] D. B. Wilson, *Determinant Algorithms for Random Planar Structures,* In Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics (1997) 258–267

[56] R. J. Wilson, *Wprowadzenie do Teorii Grafów,* PWN (2002)

[57] Doron Zeilberger, *A combinatorial Approach to Matrix Algebra,* Discrete Math. **56** (1985) 61–72

Indeks

h-ścieżka, 29
j-ciąg marszrut, 64
l-ciąg pseudocykli, 54
l-tablica ciągów marszrut, 64
$s(n)$-separator, 45
ściana, 15
ścieżka, 14
    zamknięta, 14
    zorientowana, 14
ślad macierzy, 17

algorytm
    MSV, 20
    MV, 20, 26

ciąg marszrut, 64
ciąg p-pseudocykli, 43
ciąg pseudocykli, 24
ciało, 15
cykl, 14
    naprzemienny, 40
    zorientowany, 14
częściowy ciąg pseudocykli, 54

długość
    ścieżki, 14
    ciągu marszrut, 64
    ciągu p-pseudocykli, 43
    ciągu pseudocykli, 24
    marszruty, 64
    naprzemiennego ciągu pseudocykli, 41
    rodziny naprzemiennych cykli, 41
digraf, 14
drzewo $s(n)$-separatorów, 46

graf
    niezorientowany, 13
    płaski, 15
    pełny, 14
    planarny, 15
    posiadający rodzinę $s(n)$-separatorów, 45
    pusty, 14
    ważony, 14
    zorientowany, 14
grupa permutacji S_n, 16

koniec krawędzi, 13
krawędź, 13
    wchodząca, 14
    wychodząca, 14

lider
    marszruty, 64
    naprzemiennego cyklu, 41
    naprzemiennego pseudocyklu, 41
    pseudocyklu, 24

mały separator, 45
macierz
    dopełnień algebraicznych, 17
    identycznościowa, 17
    skośno-symetryczna, 18
    transponowana, 16
marszruta, 63

naprzemienne pokrycie cyklami, 41
naprzemienny ciąg pseudocykli, 41
naprzemienny pseudocykl, 41
notacja $\tilde{O}$, 19

p-orientacja, 39
p-pseudocykl, 43
pętla, 67

permanent, 17
Pfaffian, 18, 39
pierścień, 15
    formalnych szeregów potęgowych, 16
    formalnych szeregów potęgowych modulo u^{n+1}, 16
    wielomianów, 16
podgraf, 14
    indukowany, 14
pokrycie cyklami, 14
problem plecakowy, 15
przecięcie, 15
pseudocykl, 14, 24

rodzina naprzemiennych cykli, 41
rysunek grafu, 14
rysunek płaski, 15

sąsiad, 13
skośne domknięcie, 24
skojarzenie, 15
    bazowe, 40
    doskonałe, 15
stopień, 13

tablica ciągów marszrut, 63

własność l-prefiksów, 60
własność prefiksów, 58
waga
    l-ciągu pseudocykli, 54
    ścieżki, 23
    ciągu marszrut, 64
    ciągu p-pseudocykli, 43
    ciągu pseudocykli, 24
    krawędzi, 14
    marszruty, 64
    naprzemiennego ciągu pseudocykli, 41
    naprzemiennego cyklu, 41
    naprzemiennego pokrycia cyklami, 41
    naprzemiennego pseudocyklu, 41
    pętli, 68
    permutacji, 17
    skojarzenia, 18
    tablicy ciągów marszrut, 64
waga ze znakiem
    l-ciągu pseudocykli, 60
    pseudocyklu, 60
wartości własne, 17
wielomian charakterystyczny, 17
wierzchołek, 13
    incydentny, 13
    skojarzony, 15
wyznacznik, 16

zamknięta ścieżka, 14
zmodyfikowany problem plecakowy, 22
znak
    l-ciągu pseudocykli, 54
    ciągu p-pseudocykli, 43
    ciągu pseudocykli, 24
    naprzemiennego ciągu pseudocykli, 41
    naprzemiennego pokrycia cyklami, 41
    permutacji, 16
    skojarzenia, 18
    tablicy ciągów marszrut, 64