1 Inne ciekawe (i trudne) wyniki

Omówimy teraz krótko ciekawsze wyniki nie przedstawione we wcześniejszych wykładach, a dotyczące problemu poprzednika bądź sortowania.

1.1 Problem poprzednika

Przypomnijmy, że kolejki van Emde Boasa i y-szybkie drzewa Willarda rozwiązują (statyczny i dynamiczny) problem poprzednika w czasie $O(\lg w)$. Przy pomocy randomizacji można uzyskać liniową pamięć.

Alternatywne rozwiązanie przy pomocy drzew spajających ma złożoność $O(\lg_k n + \lg k)$ dla dowolnego $k \leq w$. Można pozbyć się drugiego członu w tej złożoności, co daje $O(\lg_w n)$ (to wynik Ramana).

W przypadku statycznym poprzednika znana jest dokładna złożoność problemu.

Twierdzenie 1 (Beame, Fich). *Statyczny problem poprzednika ma złożoność $\Theta(\min\{\lg_w n, \frac{\lg w}{\lg \lg w}, \frac{\lg w}{\lg \frac{\lg S}{\lg n}}\})$, gdzie S jest rozmiarem pamięci używanej przez strukturę danych.*

Nie wiadomo, czy równie szybko można rozwiązać przypadek dynamiczny (oczywiście dolne granice się przenoszą). W dalszej części wykładu pokażemy proste dolne granice dla problemu poprzednika.

1.2 Sortowanie

Pozaliśmy, że posortować n liczb można w czasie $O(n \lg \lg n)$, a jeśli w jest wystarczająco duże, to nawet w czasie $O(n)$.

Najlepsze znane algorytmy sortują w czasie $O(n \lg \lg n)$ (Han) deterministycznie i $O(n\sqrt{\lg \lg n})$ (Han, Thorup) przy użyciu randomizacji. Nie są znane żadne nietrywialne dolne granice. W szczególności nie jest wykluczone, że można zawsze sortować w czasie liniowym.

1.3 Sortowanie, a kolejki priorytetowe

Wiadomo, że korzystając z szybkiej kolejki priorytetowej można szybko sortować. Okazuje się, że zybkiego algorytmu sortowania można użyć do szybkiej implementacji operacji kolejki!

Twierdzenie 2 (Thorup). *Niech $nS(n, w)$ będzie złożonością sortowania n liczb w-bitowych. Wtedy istnieje implentacja kolejki priorytetowej dla liczb w-bitowych, w której wszystkie operacje mają złożoność $O(S(n, w))$.*

Korzystając z szybkich kolejek priorytetowych można szybko rozwiązywać problemy grafowe (w przypadku całkowitych wag): minimalne drzewa rozpinające w czasie $O(m + n)$, najkrótsze ścieżki w grafach nieskierowanych w czasie $O(m + n)$, a w grafach skierowanych w czasie $O(m + n \lg \lg n)$. Są to jednak raczej wyniki o charakterze teoretycznym.

2 Dolne granice w modelu cell-probe

W ostatniej części cyklu wykładów na temat struktur danych i algorytmów operujących na liczbach całkowitych omówimy kwestię dolnych granic. Dowodzenie dolnych granic bezpośrednio w modelu RAM, czy WordRAM, byłoby bardzo trudne, bo modele te są niezwykle skomplikowane. Okazuje się jednak, że nawet w modelu dużo silniejszym niż WordRAM, tzw. modelu cell-probe, można udowodnić dla wielu problemów dolne granice, które są równe górnym granicom uzyskanym w modelu RAM. Jest to w szczególności możliwe dla problemu poprzednika. W dalszej części tego wykładu pokażemy „prostą" wersję tych rozważań.

2.1 Definicja modelu

Zacznijmy od definicji prostego modelu, w którym będziemy dowodzić nasze dolne granice. Program w w-bitowym modelu *cell-probe* jest drzewem o stopniu rozgałęzienia 2^w. Każdy wierzchołek v drzewa ma etykietę n_v będącą numerem komórki pamięci, a każda krawędź e drzewa ma dwie etykiety r_e (do zapisu), w_e (do odczytu). Zakładamy przy tym, że w krawędziach wychodzących z dowolnego wierzchołka v, każda liczba w-bitowa pojawia się jako r_e dokładnie raz.

„Wykonanie" wierzchołka polega na odczytaniu z pamięci zawartości x komórki n_v, i zejściu w dół krawędzią o etykiecie r_e równej x. Dodatkowo zawartość komórki n_v zmnieniana jest na w_e (jeśli $w_e = r_e$ to tak naprawdę nic się nie zmienia).

Zwróćmy uwagę, że w tym modelu obliczenia nic nas nie kosztują, płacimy tylko za dostęp do pamięci. Dlatego kluczowe jest tutaj przyjęcie WordRAM-owego ograniczenia na liczbę bitów pojedynczej komórki pamięci. Bez tego założenia moglibyśmy każdy problem rozwiązać w czasie potrzebnym na przeczytanie danych i wypisanie wyniku, co nie byłoby zbyt korzystne w kontekście przydatności modelu cell-probe do dowodzenia dolnych granic.

2.2 Złożoność komunikacyjna

Wiele ciekawych dolnych granic można udowadniać bezpośrednio w modelu cell-probe. W przypadku problemu poprzednika wygodniej jest skorzystać z jeszcze silniejszego modelu, tzw. *modelu komunikacyjnego*.

Niech $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{V}$ będzie dowolną funkcją. W modelu komunikacyjnym mamy dwie osoby – Alicję i Boba. Alicja dostaje $x \in \mathcal{X}$, a Bob $y \in \mathcal{Y}$. Ich zadaniem jest obliczenie $f(x, y)$. Mogą się oni porozumiewać jedynie przesyłając ustalonego rozmiaru komunikaty i oczywiście chcą ich przesłać jak najmniej. Algorytm wysyłania komunikatów nazywa się z reguły *protokołem*. Protokół, w którym Alicja wysyła komunikaty rozmiaru a, Bob wysyła komunikaty rozmiaru b, a łączna liczba komunikatów jest $\leq t$ będziemy nazywać protokołem typu $[a, b, t]$. Ponadto będziemy czasem pisać $[a, b, t]^A$ / $[a, b, t]^B$, aby podkreślić, że pierwszy komunikat wysyła Alicja/Bob.

Niech $f : \mathcal{D} \times \mathcal{Q} \to \mathcal{V}$ będzie problemem strukturodanowym. $\mathcal{D}$ jest tu zbiorem wszystkich danych jakie struktura może reprezentować, $\mathcal{Q}$ jest zbiorem wszystkich możliwych zapytań, a $\mathcal{V}$ zbiorem możliwych odpowiedzi. Ponadto dla $d \in \mathcal{D}$ and $q \in \mathcal{Q}$ wartość $f(d, q)$ jest prawidłową odpowiedzią dla danych d i zapytania q.

Lemat 3. *Przy założeniach jak powyżej, jeśli istnieje struktura danych rozmiaru S, odpowiadając na zapytania w czasie $\leq t$, to istnieje dla f protokół typu* $[\log S, w, 2t]$.

Ćwiczenie 1. *Uzasadnij ten lemat.*

W związku z tym możemy się skoncentrować na dowodzeniu ograniczeń dolnych na t dla protokołów typu $[a = \log n, b = w, t]$, rozwiązujących problem poprzednika.

2.3 Eliminacja rund

Podstawowym narzędziem w dowodzeniu dolnych ograniczeń w modelu komunikacyjnym jest tzw. lemat o eliminacji rund. Zanim go jednak sformułujemy, musimy zdefiniować jedno dodatkowe pojęcie.

Niech $f : \mathcal{X} \times \mathcal{Y} \to \mathcal{V}$ będzie jak poprzednio. k-krotnym złożeniem f będziemy nazywać problem $f^{(k)}$ zdefiniowany następująco. Alicja dostaje liczby $x_1, \ldots, x_k \in \mathcal{X}$, a Bob liczbę $y \in \mathcal{Y}$, indeks $i \in 1, \ldots, k$ i liczby $x_1, \ldots, x_{i-1}$. Ich zadaniem jest obliczenie $f(x_i, y)$.

Uwaga. Zwróćmy uwagę, że Bob zna część liczb znanych Alicji. Wydaje się to niepotrzebne, ale jest to założenie techniczne bardzo przydatne w dowodach.

Rozważmy dowolny protokół obliczający $f^{(k)}$. Przypuśćmy, że Alicja wysyła pierwszy komunikat. Chciałaby ona przesłać x_i, ale niestety nie zna wartości i. Jeśli $a \ll k$, to wydaje się, że nie jest ona w stanie przesłać zbyt wiele informacji na temat x_i (średnio a/k bitów). Poniższy lemat formalizuje tę intuicję:

Lemat 4 (Eliminacja rund)**.** *Przypuśćmy, że istnieje protokół typu $[a, b, t]^A$ rozwiązujący $f^{(k)}$ z prawdopodobieństwem błędu δ. Wtedy istnieje protokół typu $[a, b, t-1]^B$ rozwiązujący f z prawdopodobieństwem błędu $\delta + O(\sqrt{a/k})$.*

Dowód tego lematu pominiemy. Same definicje niezbędnych pojęć z teorii informacji zajęłoby nam zbyt wiele czasu. Intuicja jest jednak jasna. Aby obliczyć f w $t-1$ krokach pomijamy pierwsza wiadomość Alicji. Zamiast tego Bob losuje tę wiadomość. Skoro Alicja nie zna i, to w pierwszej wiadomości nie może przekazać Bobowi zbyt wiele informacji na temat x_i, średnio a/k bitów. Prawdopodobieństwo, że Bob źle wylosuje te bity jest równe $1-2^{-a/k}$. Protokół da błędną odpowiedź, jeśli dawał ją wcześniej, lub jeśli Bob źle odgadł. Dla małych a/k mamy $1 - 2^{-a/k} \approx a/k$, co dawałoby prawdopodobieństwo błędu $\delta + a/k$. Niestety powyższa dyskusja nie jest formalnym dowodem, ignorujemy tu pewne trudności techniczne, które powodują, że daje się udowodnić tylko nieco słabsze ograniczenie prawdopodobieństwa błędu: $\delta + \sqrt{a/k}$.

2.4 Dolne granice dla problemu poprzednika

Idea dowodu ograniczenia dolnego jest prosta. Zakładamy, że istnieje protokół dla problemu poprzednika używający t komunikatów przy ustalonych w i n. Za pomocą lematu o eliminacji rund pokażemy, że istnieje wtedy protokół korzystający z $t-1$ komunikatów dla pewnych mniejszych wartości w i n. Jeśli t jest „za małe", to w ten sposób dojdziemy do protokołu, który nie wymaga żadnej komunikacji, a mimo to rozwiązuje problem poprzednika z małym prawdopodobieństwem błędu, co jest oczywiście niemożliwe (oznaczałoby to odpowiadanie na zapytania bez zaglądania do struktury danych).

Dla uproszczenia zamiast problemu poprzednika rozważymy problem *koloru poprzednika*. W problemie tym każdy element ma jeden z dwóch kolorów: biały (0) lub czarny (1). Nie wymagamy, aby protokół znajdował poprzednika, a jedynie by znajdował kolor poprzednika. Problem ten w prosty sposób sprowadza się do problemu poprzednika.

Oczywiście nie może istnieć algorytm, który bez zaglądania do struktury danych odpowie na pytanie o kolor z prawdopodobieństwem $> 1/2$. Będziemy postępować następująco. Zaczniemy od protokołu typu $[a, b, t]^A$ z prawd. błędu δ dla problemu poprzednika z parametrami w, n. Następnie za pomocą lematu o eliminacji rund uzyskamy protokół typu $[a, b, t-1]^B$ z prawd. błędu $\delta' > \delta$ dla problemu poprzednika z parametrami $w' \leq w$, $n' \leq n$, itd. Naszym celem jest uzyskanie protokołu typu $[a, b, 0]$ z prawd. błędu $\leq \frac{1}{3}$ dla nietrywialnego problemu poprzednika. Aby uzyskać takie prawdopodobieństwo błędu będziemy za każdym razem dokonywać redukcji w taki sposób, aby prawd. błędu rosło o nie więcej niż $\frac{1}{3t}$.

Eliminacja rund Alicja → Bob Załóżmy, że istnieje protokół P typu $[a, b, T]^A$ z p.b. δ dla problemu $\mathrm{PRED}_{w,n}$ (tzn. statyczny problem poprzednika dla n w-bitowych liczb). Chcielibyśmy skorzystać z lematu o eliminacji rund. W tym celu musimy zinterpretować ten protokół jako rozwiązanie problemu $f^{(k)}$ dla pewnego f. Chcemy przy tym, żeby p.b. wzrosło o $\leq 1/3t$, więc $k = \Theta(at^2)$. Jak wybrać problem f? Okazuje się, że możemy wziąć $f = \mathrm{PRED}_{w/k,n}$. Dlaczego P rozwiązuje $f^{(k)}$?

Niech y, $x_1, \ldots, x_k$ oraz i będą jak w definicji $f^{(k)}$. A zatem Bob zna $y, x_1, \ldots, x_{i-1}$ oraz i, a Alicja $x_1, \ldots, x_k$. Zwróćmy uwagę, że liczby x_j oraz liczby w y są w/k bitowe. W protokole P Alicja dysponuje tylko jedną liczbą – $\hat{x}$. Tą liczbą niech będzie $\hat{x} = x_1 \ldots x_k$ (konkatenacja). Z kolei Bob w P dysponuje tylko danymi, oznaczmy je $\hat{y}$. Zdefiniujmy $\hat{y}$ następująco: $\hat{y} = x_1 x_2 \ldots x_{i-1} y$. Widać, że poprzednik $\hat{x}$ w $\hat{y}$ odpowiada poprzednikowi x_i w y. A zatem P rozwiązuje $\mathrm{PRED}^{(k)}_{w/k,n}$. Z lematu o eliminacji rund dostajemy, że istnieje protokół typu $[a, b, T-1]^B$ z p.b. $\delta + 1/3t$ rozwiązujący problem $\mathrm{PRED}_{w/k,n}$.

Uwaga. Można dość intuicyjnie wytłumaczyć to co się tutaj dzieje. Alicja dysponuje pewną liczbą x natomiast dane Boba y mają (dzięki naszej złośliwości) tę własność, że do pewnego miejsca zgadzają się z x a potem wszystkie na bardzo małej (w/k) liczbie bitów rozjeżdzają się z x. Alicja nie wie w którym miejscu to następuje, wiec nie jest w stanie wysłać sensownego komunikatu.

Eliminacja rund Bob → Alicja Teraz rozpatrujemy przypadek, w którym Bob wysyła pierwszy komunikat (czyli struktura danych mówi coś bez pytania :)). Załóżmy, że istnieje protokół P typu $[a, b, T]^B$ z p.b. δ dla problemu $\mathrm{PRED}_{w,n}$. Chcemy znów skorzystać z lematu o eliminacji rund, więc musimy pokazać, że P rozwiązuje jakiś problem postaci $f^{(k)}$. Tym razem $k = \Theta(bt^2)$ (role Boba i Alicji są zamienione).

Wystarczy wziąć $f = \mathrm{PRED}_{w'=w-\log k, n/k}$. Pokażemy teraz, że P rozwiązuje $f^{(k)}$. Niech $y_1, \ldots, y_k$ będą zestawami danych Boba. Alicja dysponuje liczbą i, zestawami $y_1, \ldots, y_{i-1}$ i zapytaniem x. W protokole P Bob ma mieć tylko jeden zestaw danych $\hat{y}$. Zdefiniujmy $\hat{y} = (0)_2 y_1 \cup (1)_2 y_2 \cup \ldots (k-1)_2 y_k$, tzn. każdy z zestawów y_j poprzedzamy unikalnym prefiksem. Z kolei zapytanie $\hat{x}$ Alicji definiujemy $\hat{x} = (i-1)_b x$. Znów, łatwo sprawdzić, że poprzednik $\hat{x}$ w $\hat{y}$ odpowiada poprzednikowi x w y_i.

Uwaga. Tu także można intuicyjnie wytłumaczyć to co się dzieje. Wyobraźmy sobie, że dzielimy uniwersum na k części, i że dane Boba mają z każdą częścia przecięcie tej samej mocy. Liczba x Alicji jest w jednej z części i to tam należy szukać poprzednika, ale Bob nie wie, która to część, więc nie jest w stanie wysłać dobrego komunikatu.

Podsumowanie Każde dwa zastosowania lematu o eliminacji rund bt^2-krotnie zmniejszają n i at^2-krotnie w (w zmniejsza się też o $\log(bt^2)$, ale dla dużych w jest to nieistotne). Eliminację musimy przerwać, gdy $w = O(\lg(bt^2))$ lub $n \leq 2$. Taka sytuacja musi nastąpić, w przeciwnym przypadku uzyskalibyśmy protokół typu $[a, b, 0]$, co jak wspomnieliśmy wcześniej, jest niemożliwe. A zatem $t = \Omega(\min\{\lg_{at^2} w, \lg_{bt^2} n\})$. Mamy, $t = O(\lg n)$, $t = O(\lg w)$, $a \geq \lg n$, $b = w$, stąd podstawy logarytmów są odpowiednio pomiędzy a i a^3 oraz pomiędzy b i b^3. Stąd $t = \Omega(\min\{\lg_a w, \lg_b n\})$.

„Czyste" ograniczenia dolne dla wielomianowej pamięci Jeśli założymy, że nasza struktura danych mieści się w pamięci wielomianowej względem n, to $a = \Theta(\lg n)$ i dostajemy $t = \Omega(\min\{\lg_{\lg n} w, \lg_w n\})$. Spróbujmy uzyskać jak najlepsze ograniczenia „czyste", tzn. takie w których pojawia się tylko n bądź tylko w. Dla ustalonego n lub w z najgorszą sytuacją mamy do czynienia wtedy, gdy

$$\lg_{\lg n} w = \lg_w n \Rightarrow \frac{\lg w}{\lg \lg n} = \frac{\lg n}{\lg w} \Rightarrow \lg^2 w = \lg n \lg \lg n.$$

Przy ustalonym n dostajemy $w = \sqrt{\lg n \lg \lg n}$, a więc $t = \Omega\left(\sqrt{\frac{\lg n}{\lg \lg n}}\right)$. Przy ustalonym w dostajemy $\lg \lg w = \Theta(\lg \lg n)$, a zatem $t = \Omega\left(\frac{\lg w}{\lg \lg w}\right)$.