Ważne przykłady w teorii złożoności

Szkoła Matematyki Poglądowej, Siedlce, sierpień 1997

Damian Niwiński
Uniwersytet Warszawski

Prolog. Słynny *Entscheidungsproblem* D. Hilberta (1928) można sprowadzić do następującego zagadnienia: znaleźć metodę, która dla danej formuły matematycznej (jak np. hipoteza Goldbacha), rozstrzyga, czy jest ona prawdziwa, czy też nie? Inny, bardziej szczegółowy problem postawiony przez Hilberta brzmiał: znaleźć metodę, która dla danego równania Diofantycznego znajduje rozwiązanie lub stwierdza, że rozwiązania nie ma. Dzięki pracom A. Turinga (1936) i J. Matijasiewicza (1970) wiemy, że żaden z tych problemów nie ma pozytywnego rozwiązania: hipoteza istnienia takich metod prowadzi do sprzeczności.

Znacznie prostszy problem: *wskazać metodę rozstrzygającą, czy dana formuła rachunku zdań jest tautologią* ma powszechnie znane rozwiązanie, a jednak od blisko 30 lat stanowi wyzwanie dla teoretycznej informatyki. Metoda "siłowa" przeszukiwania wszystkich wartościowań prowadzi bowiem do tak długiego czasu obliczenia, że wyklucza praktyczne realizacje. Jednak żadna istotnie lepsza metoda nie jest znana. Co więcej, nie znamy również dowodu, że taka metoda nie istnieje.

Teoria złożoności próbuje wyjaśniać dlaczego pewne problemy okazują się odporne na próby znalezienia dobrych algorytmicznych rozwiązań. W tym sensie jest komplementarna w stosunku do *algorytmiki* – która takich rozwiązań poszukuje i je tworzy. Ujmując rzecz pozytywnie (*think positive!*), teoria złożoności dąży do określenia *trudności* problemów algorytmicznych i klasyfikuje je ze względu na stopień trudności.

Świat problemów algorytmicznie rozwiązywalnych nie jest przypadkowym katalogiem zadań wynikłych z rozmaitych kontekstów praktycznych, ale ma swoją strukturę, którą dopiero poznajemy. Na tej "mapie" są pewne obszary "latwe" i "trudne", a także "szare". Są także pewne punkty orientacyjne ważne dla struktury całej "mapy".

Zgodnie z tematem przewodnim aktualnej szkoły matematyki poglądowej, właśnie te punkty znajdą się w centrum naszego zainteresowania.

Podstawowe pojęcie. Bez zmniejszenia ogólności możemy przyjąć, że *algorytmiczny problem funkcyjny* dany jest jako funkcja $f : \Sigma^* \to \Sigma^*$, a *algorytmiczny problem decyzyjny* jako podzbiór $L \subseteq \Sigma^*$ (lub, równoważnie, jako funkcja *charakterystyczna* L), gdzie Σ^* oznacza zbiór wszystkich słów nad pewnym alfabetem Σ.

Problem decyzyjny da się więc sprowadzić do pytania, "czy dane słowo w należy do języka L ?". Niektóre problemy już mają taką postać, inne można do niej sprowadzić poprzez odpowiednie "zakodowanie", np. czy dane słowo $w \in \{0,1\}^*$ jest binarnym przedstawieniem liczby pierwszej lub też czy dane słowo jest macierzą incydencji grafu posiadającego cykl Hamiltona.

Zauważmy, że problem nad alfabetem o $k \geq 2$ symbolach możemy łatwo sprowadzić do problemu nad alfabetem $\{0, 1\}$ powiększając długość słów $\lceil \log_2 k \rceil$ razy.

Jak wspomnieliśmy, interesować nas będzie trudność problemów algorytmicznych, którą będziemy mierzyć ilością "zasobów" zużywanych przez różne modele obliczeń do rozwiązania problemu. Zasoby mogą być różne: ilość stanów wewnętrznych maszyny, czas rozwiązywania (jako funkcja rozmiaru danych wejściowych), pamięć (inaczej: przestrzeń) robocza, liczba procesorów (przy realizacjach równoległych) i wiele innych.

Języki regularne – mikrokosmos obliczalności

Języki regularne stanowią klasę najprostszych języków obliczalnych, z grubsza mówiąc, rozpoznawalnych po jednokrotnym przeczytaniu słowa i przy użyciu stałej pamięci.

Automat skończony[1] możemy opisać jako układ $\langle \Sigma, Q, q_0, \delta, F \rangle$, gdzie Σ jest skończonym alfabetem, Q skończonym zbiorem stanów z wyróżnionym stanem początkowym q_0 i zbiorem stanów *akceptujących* $F \subseteq Q$, a $\delta \subseteq Q \times \Sigma \times Q$ jest zbiorem możliwych *przejść*. Ciąg przejść postaci (p_0, σ_1, p_1), (p_1, σ_2, p_2), ..., (p_{m-1}, σ_m, p_m) nazwiemy *obliczeniem* automatu na słowie $\sigma_1\sigma_2 \ldots \sigma_m$. Jeśli wyobrazimy sobie automat jako graf, którego wierzchołkami są stany, a krawędzie etykietowane są symbolami z Σ, przy czym krawędź z p do q etykietowana σ istnieje wtedy, gdy $(p, \sigma, q) \in \delta$, to obliczenia automatu odpowiadają w naturalny sposób wędrówkom po takim grafie. Mówimy, że automat *akceptuje* słowo w, jeśli istnieje obliczenie na tym słowie startujące ze stanu początkowego q_0 i kończące się w jednym ze stanów z F.

Przykład: podzielność. Zbiór słów nad alfabetem $\{0, 1, \ldots, 9\}$ przedstawiających w systemie dziesiętnym liczby podzielne przez 7 jest rozpoznawalny przez automat o stanach $0, 1, \ldots, 6$ i przejściach $(i, k, (10 \cdot i + k) \bmod 7)$, ze stanem początkowym i jedynym stanem akceptującym równym 0.

Powyższy automat jest *deterministyczny* w tym sensie, że dla każdego stanu q i symbolu–etykiety σ istnieje dokładnie jeden stan p taki, że $(q, \sigma, p) \in \delta$. W ogólności każdy automat niedeterministyczny można zastąpić równoważnym deterministycznym o zbiorze stanów $\wp(Q)$ i przejściach postaci $(X, \sigma, \{y : (\exists x)\,(x, \sigma, y) \in \delta\})$. Za stan początkowy przyjmujemy $\{q_0\}$, a za stany akceptujące $\{X : X \cap F \neq \emptyset\}$.

Ta użyteczna konstrukcja nie jest bardzo ekonomiczna, ale wykładniczego wzrostu liczby stanów przy determinizacji nie można w ogólności uniknąć.

Przykład: determinizm, niedeterminizm. Niech $k < \omega$ i niech $Last_k$ będzie zbiorem słów nad alfabetem $\{0, 1\}$ o długości $\geq k$, takich, że na k–tej od końca pozycji jest 1. Nietrudno zbudować niedeterministyczny automat o $k + 1$ stanach akceptujący język $Last_k$. Natomiast jakikolwiek deterministyczny automat rozpoznający ten język musi mieć co najmniej 2^k stanów. Żeby się o tym przekonać, przypuśćmy, że dwa słowa v i w o długości dokładnie k różnią się na i–tej (licząc od początku) pozycji, powiedzmy $v_i = 1$ a $w_i = 0$. Jeśli teraz "dopełnimy" je *w taki*

[1] Prezentowany automat nazywa się *niedeterministycznym*, choć należałoby raczej mówić o automacie "niekoniecznie deterministycznym". Automat deterministyczny jaki definiujemy poniżej jest więc szczególnym przypadkiem niedeterministycznego.

sam sposób tak, by i–ta pozycja stała się w obu słowach k–tą pozycją od końca, np. $v0^i$ i $w0^i$, to widzimy, że pierwsze słowo jest w $Last_k$, a drugie nie. A zatem, po przeczytaniu słów v i w automat musi znajdować się w dwóch różnych stanach (dlaczego?).

Można wykazać, że 2^k stanów *wystarczy* do akceptacji języka $Last_k$.

Dla tak prostego modelu obliczeń jak automat skończony, rozmiar automatu można uważać za pewną miarę złożoności akceptowanego języka.[2] Rozważmy pytanie, jak zachowuje się ta wielkość przy konstrukcjach teorio-mnogościowych. Na przykład, mając dane automaty $A_1, \ldots, A_k$ rozpoznające języki $L_1, \ldots, L_k$, nietrudno jest skonstruować automat rozpoznający język $L_1 \cap \ldots \cap L_k$. Punktem wyjścia jest przyjęcie za zbiór stanów nowego automatu*produktu Kartezjańskiego* zbiorów stanów automatów $A_1, \ldots, A_k$.

Jak widzimy, liczba stanów przy tej konstrukcji wzrasta wykładniczo w zależności od k. I tu również, tego niekorzystnego zjawiska (zwanego "eksplozją stanów", ang. *state explosion*) w ogólności nie da się uniknąć.

Przykład – eksplozja stanów. Niech $\Sigma = \{0, 1, \ldots, k\}$ i niech dla $m = 1, \ldots, k$, L_m będzie zbiorem słów v nad Σ o następującej własności:

> m występuje w v, ale przed pierwszym i pomiędzy każdymi dwoma kolejnymi wystąpieniami m, $m-1$ występuje co najmniej dwa razy.

Nietrudno jest skonstruować (deterministyczny) automat o 4 stanach rozpoznający L_m. Z drugiej strony, najkrótsze słowo w $L_1 \cap \ldots \cap L_k$ ma długość $1 + 2 + \ldots + 2^k$, a zatem każdy automat rozpoznający ten język musi mieć co najmniej tyleż stanów (dlaczego?).

Jak możemy oczekiwać, już całkiem proste własności mogą leżeć poza zasięgiem automatów skończonych.

Przykład – nieregularność. Język $\{0^n 1^n : n < \omega\}$ nie jest rozpoznawany przez żaden automat skończony. Istotnie, przypuśćmy, że jest rozpoznawany przez automat o M stanach. Wówczas w obliczeniu akceptującym na słowie $0^M 1^M$, dla pewnych $0 \leq i < j \leq M$, po 0^i i 0^j wystąpi *ten sam* stan. Ale wtedy moglibyśmy skonstruować akceptujące obliczenie również dla słowa $0^{M-(j-i)} 1^M$, sprzeczność!

Z podobnych powodów automaty skończone nie potrafią akceptować zbioru palindromów, zbioru poprawnie uformowanych ciągów nawiasów itp.

Wspomniane języki można by jednak akceptować przy pomocy automatu dodatkowo wyposażonego w strukturę *stosu*, na który automat może wkładać bądź z niego zdejmować symbole, w zależności od aktualnego stanu i symbolu widocznego na szczycie stosu.

Jednak już tak prostego języka jak $\{0^n 1^n 0^n : n < \omega\}$ nie da się akceptować nawet przy pomocy automatu ze stosem. (Dowód jest nieco trudniejszy.)

[2] Czytelnik zaznajomiony z pojęciem *uniwersalnej* maszyny Turinga zgodzi się zapewne, że miara taka miałaby znikomą przydatność dla maszyn Turinga.

Granice obliczalności — funkcja Ackermana

Jest to klasyczny przykład zadania wprawdzie obliczalnego, ale bardzo trudnego; przykład znany na długo przed początkami teorii złożoności.

Funkcję Ackermana $A : N \times N \times N \to N$, możemy określić równaniami rekurencyjnymi

$$\begin{aligned} A(0,0,x) &= x \\ A(0,y+1,x) &= A(0,y,x)+1 \\ A(1,0,x) &= 0 \\ A(z+2,0,x) &= 1 \\ A(z+1,y+1,x) &= A(z,A(z+1,y,x),x) \end{aligned}$$

Nietrudno sprawdzić, że $A(0,y,x) = y + x$, $A(1,y,x) = yx$, $A(2,y,x) = x^y$; w pewnym sensie "i tak dalej"... (Pomyślmy z jaką szybkością rośnie funkcja jednej zmiennej $A(n,n,n)$.)

O funkcji Ackermana można wykazać, że jest obliczalna, ale nie należy do klasy funkcji pierwotnie rekurencyjnych, określonej jako najmniejsza rodzina PR zawierająca stałą 0, następnik i rzutowanie $P_{n,i}(x_1,\ldots,x_n) = x_i$, dla każdych $n \in N$, $1 \leq i \leq n$, oraz zamknięta na złożenie i operację pierwotnej rekursji: tzn. jeśli funkcje $g(y_1,\ldots,y_k)$ i $h(x,z,y_1,\ldots,y_k)$ są w PR, to również funkcja $f(x,y_1,\ldots,y_k)$, określona równaniami

$$\begin{aligned} f(0,y_1,\ldots,y_k) &= g(y_1,\ldots,y_k) \\ f(n+1,y_1,\ldots,y_k) &= h(n,f(n,y_1,\ldots,y_k),y_1,\ldots,y_k) \end{aligned}$$

jest w PR.

Paradygmat efektywnej obliczalności: złożoność wielomianowa

Matematyczną idealizacją klasy problemów praktycznie rozwiązywalnych przez komputery jest tzw. klasa P problemów rozwiązywalnych w *czasie wielomianowym*. Mówiąc ogólnie, problem jest w tej klasie, jeśli może być rozwiązany przez algorytm, który dla danych wejściowych rozmiaru n wykonuje nie więcej niż n^k kroków dla pewnego k. Ścisła definicja wymaga sprecyzowania modelu obliczeń, jednak klasa P jest bardzo elastyczna i "odporna" na modyfikacje definicji (poniekąd ze względu na własności samych wielomianów). Dlatego też Czytelnik może posługiwać się powyższym intuicyjnym określeniem lub lubianym przez siebie modelem (jak np. programy w Pascalu), pod warunkiem, że model ten nie będzie używał niedeterminizmu, losowości ani równoległości.[3]

Dla zainteresowanych Czytelników podajemy jednak dokładniejszą definicję, w opraciu o pojęcie *maszyny Turinga*.

Pojęcie to możemy uważać za rozszerzenie automatu skończonego o kilka istotnych elementów. Technicznie, deterministyczną maszynę Turinga określamy

$$M = \langle \Sigma, \Gamma, B, Q, q_0, q_a, \delta \rangle$$

[3]Równoległy model obliczeń, to taki, gdzie pewna liczba jednostek (np. procesorów) może pracować niezależnie, przy czym liczba tych jednostek może wzrastać w zależności od danych wejściowych.

gdzie Σ jest alfabetem wejściowym, $\Gamma \supseteq \Sigma$ alfabetem roboczym, $B \in \Gamma - \Sigma$ wyróżnionym symbolem *blank*, Q zbiorem stanów, q_0 stanem początkowym, q_a stanem *akceptującym*, a δ funkcją częściową z $Q \times \Gamma$ w $Q \times \Gamma \times \{L, R, Z\}$. (W niedeterministycznej maszynie δ może być relacją.)

Intuicyjna interpretacja jest następująca: maszyna wyposażona jest w *taśmę*, która stanowi nieskończony ciąg komórek, z których każda może pomieścić jeden symbol alfabetu Γ. W chwili startowej początkowe komórki taśmy wypełnione są kolejnymi symbolami słowa wejściowego (napisanego w alfabecie Σ), a pozostałe komórki zawierają symbol *blank* (intuicyjnie: są puste). Począwszy od tej chwili czas płynie dyskretnie i w każdej chwili maszyna znajduje się w pewnym stanie, a jej *głowica* ogląda jedną komórkę taśmy. W chwili startowej głowica ogląda symbol w *pierwszej* komórce taśmy, a maszyna znajduje się w stanie początkowym q_0. Dalej, jeśli w danej chwili maszyna znajduje się w stanie q, a oglądana komórka zawiera symbol γ, to sytuacja w chwili następnej określona jest przez wartość funkcji $\delta(q, \gamma) = \langle p, \rho, X\rangle$. Mianowicie maszyna zmienia stan z q na p, zastępuje[4] symbol γ przez symbol ρ, po czym przesuwa głowicę o jedną komórkę w lewo, w prawo, lub też pozostawia ją na miejscu, w zależności od wartości $X = R$, L lub Z, odpowiednio. Globalna sytuacja jest więc w pełni określona przez zawartość taśmy, numer oglądanej komórki i stan maszyny. *A priori* mamy przeliczalnie[5] wiele możliwych sytuacji. Sytuacja początkowa, określona przez słowo wejściowe, w pełni determinuje ciąg następujących po sobie sytuacji, który może być nieskończony lub skończony (jeśli, dla pewnych q i γ, wartość $\delta(q, \gamma)$ nie jest określona lub też "ruch w lewo" nie da się wykonać). Taki ciąg nazywamy *obliczeniem*.

Uważamy, że maszyna *akceptuje* słowo $w \in \Sigma^*$, jeśli obliczenie, w którym w jest słowem wejściowym, prowadzi do sytuacji, w której maszyna osiąga stan akceptujący q_a.

Mówimy, że maszyna pracuje w czasie ograniczonym przez funkcję $\varphi : N \to N$, jeśli dla każdego $w \in \Sigma^*$, obliczenie ze słowem wejściowym w kończy się po nie więcej niż $\varphi(|w|)$ krokach.

Problem $L \subseteq \Sigma^*$ zaliczamy do klasy P wtedy, gdy składa się on dokładnie ze słów akceptowanych przez pewną deterministyczną maszynę Turinga pracującą w czasie ograniczonym przez wielomian.

Przynależność problemu funkcyjnego $f : \Sigma^* \to \Sigma^*$ do klasy P określamy podobnie, z tym, że tym razem wymagamy, by istniała deterministyczna maszyna pracująca w czasie ograniczonym przez wielomian, taka, że, dla każdego słowa $w \in \Sigma^*$, w sytuacji kończącej obliczenie na taśmie znajduje się słowo $f(w)$ (stany akceptujące nie odgrywają w tej definicji żadnej roli). Poniżej przedstawiamy kilka problemów w klasie P.

Przykład – osiągalność w grafie. Dany graf skierowany o zbiorze wierzchołków[6] V i zbiorze krawędzi $E \subseteq V \times V$ oraz wyróżnione wierzchołki $s, t \in V$. Znaleźć, o ile istnieje, ścieżkę z s do t o najkrótszej długości.

Przeglądanie wszystkich możliwych ścieżek zajęłoby nam czas wykładniczy. Możemy jednak postępować sprawniej. W kolejnych etapach obliczamy pewien (jak zobaczymy, niezbyt duży!) zbiór ścieżek *Path* i, pomocniczo, zbiór stanów *Reach*.

Startujemy z $Path = \{\langle s\rangle\}$ i $Reach = \{s\}$. Jeśli na jakimś etapie *Path* zawiera ścieżkę z s do t, ścieżka ta stanowi odpowiedź. W przeciwnym razie próbujemy rozszerzyć wszystkie elementy *Path* o jedną krawędź, prowadzącą do wierzchołka w $V-$ *Reach*. Jeśli to się nie uda dla żadnej ścieżki, kończymy działanie z odpowiedzią, że ścieżki nie ma. Jeśli niektóre ścieżki dadzą się rozszerzyć, dokładamy nowo osiągnięte wierzchołki do *Reach*. Nowa wartość *Path* składa się natomiast ze wszystkich udanych rozszerzeń, przy czym, jeśli do jakiegoś wierzchołka prowadzi więcej niż jedna ścieżka, pozostawiamy tylko jedną, a resztę wyrzucamy.

Zauważmy, że w ten sposób zawsze $|Path| \leq |V|$; co więcej liczba ścieżek "badanych na możliwość rozszerzenia" w ciągu całego działania algorytmu nie przekracza $|V|$.

---

[4]Nie wykluczamy, oczywiście, sytuacji gdzie $p = q$ lub $\rho = \gamma$.

[5]Zauważmy, że w każdej chwili prawie wszystkie komórki taśmy zawierają *blank*.

[6]Wszystkie grafy rozważane w naszych przykładach są *skończone*.

Dowód poprawności i oszacowanie czasu działania algorytmu pozostawiamy Czytelnikowi. (Wykładnik wielomianu może zależeć od sposobu reprezentacji grafu i przyjętego modelu obliczeń.)

Przykład – obwody logiczne. *Obwód logiczny* możemy przedstawić jako skierowany graf bez pętli, którego wierzchołki (tzw. bramki) są etykietowane przez $\vee$, $\wedge$, $\neg$, $\mathbf{0}$, $\mathbf{1}$ lub zmienne $x, y, z, \ldots$, przy czym wierzchołki etykietowane $\neg$ mają stopień wejścia 1, a wiechołki etykietowane $\mathbf{0}$, $\mathbf{1}$ lub zmienną, stopień wejścia 0 (tzn. wierzchołek nie jest końcem żadnej krawędzi). Dla danego wartościowania zmiennych występujących w obwodzie, określamy indukcyjnie *wartość logiczną* dowolnego wierzchołka w oczywisty sposób. Zwykle wyróżnia się także jeden wierzchołek "wyjściowy", którego wartość logiczna określana jest jako wartość całego obwodu.[7]

Obwód logiczny *per se* może być uważany za pewien model obliczeń, jakkolwiek o skończonym zasięgu: obwód ze zmiennymi $x_1, \ldots, x_n$ oblicza funkcję Boole'owską $\{0,1\}^n \to \{0,1\}$. (Sposób działania obwodu logicznego przypomina pod pewnymi względami działanie komórek neuronowych.) Model ten można modyfikować wprowadzając inne typy bramek niż operacje logiczne, np. bramki typu *majority* (wierzchołek ma wartość logiczną $\mathbf{1}$ jeśli *większość* poprzedników ma wartość $\mathbf{1}$) lub bramki zliczające liczbę jedynek *modulo* p. Zależności pomiędzy różnymi typami bramek są jeszcze w znacznym stopniu nie wyjaśnione.[8]

Jeśli ograniczymy się do obwodów logicznych *bez zmiennych*, to nietrudno jest podać algorytm, który w czasie wielomianowym oblicza wartość danego obwodu logicznego.

Problem ten jest w pewnym sensie, który można ściśle zdefiniować, *uniwersalny* w klasie P.

Pomijając formalne definicje, podamy główną ideę. Załóżmy, że dana jest deterministyczna maszyny Turinga M działająca w czasie ograniczonym przez $p(n)$ i dowolne słowo w. Twierdzimy, że, *nie* wykonując symulacji działania M, można *łatwo* skonstruować obwód logiczny, który ma wartość $\mathbf{1}$ wtedy i tylko wtedy, gdy M przyjmując na wejściu słowo w, kończy obliczenie w stanie akceptującym. Rozważmy wszystkie czwórki postaci (t, k, γ, p), gdzie $t, k \leq p(|w|)$, γ jest symbolem alfabetu roboczego maszyny M, a p jest stanem M lub, powiedzmy, $\emptyset$. W naszym obwodzie, każdej takiej czwórce będzie odpowiadała pewna bramka, przy czym jej wartość będzie $\mathbf{1}$ wtedy, gdy

> w chwili t w komórce k znajduje się symbol γ, a ponadto, jeśli $p \neq \emptyset$, to głowica maszyny ogląda tę właśnie komórkę i maszyna jest w stanie p.

Zauważmy, że zawartość komórki k w chwili $t > 0$ jest całkowicie zdeterminowana przez zawartość komórek $k-1$, k i $k+1$ w chwili $t-1$, a zawartość w chwili $t = 0$ jest wyznaczona przez słowo wejściowe. Ta obserwacja pozwala na konstrukcję żądanego obwodu.

(Ze względów technicznych będzie on zawierał jeszcze inne bramki niż te opisane powyżej, ale liczba wszystkich bramek pozostanie proporcjonalna do $p(|w|)^2$.)

Istotą tej konstrukcji jest, że obliczenie odpowiedniego obwodu na podstawie M i w może być wykonane jeszcze *bardziej efektywnie* niż po prostu w czasie wielomianowym.[9] W tym sensie (który można uściślić) można powiedzieć, że każdy problem w P da się sprowadzić do zagadnienia ewaluacji obwodów logicznych.

---

[7] Oczywiście każda formuła rachunku zdań może być utożsamiona z pewnym obwodem; na odwrót, każdy obwód może zostać "rozwinięty" do równoważnej formuły. Pytanie, czy przy tej ostatniej transformacji rozmiar formuły *musi* rosnąć wyładniczo w stosunku do rozmiaru obwodu okazuje się być głębokie i pozostaje otwarte.

[8] Wiadomo, że bramka typu *majority* może łatwo symulować wszystkie bramki mod$n$. R. Smolensky wykazał w 1987 r., że bramka typu mod$p$ nie może symulować mod$q$ jeśli p i q są różnymi liczbami pierwszymi; w konsekwencji mod$p$ dla żadnego pierwszego p nie może symulować *majority*. Analogiczne pytanie dla bramki typu mod6 jest otwarte.

[9] Na przykład przez maszynę Turinga pracującą w pamięci zaledwie logarytmicznej.

Przykład – doskonałe skojarzenia. Rozważmy teraz graf $\langle V, E\rangle$ szczególnej postaci. Zbiór wierzchołków jest sumą dwóch rozłącznych równolicznych zbiorów, powiedzmy $V = B \cup G$, gdzie $B = \{b_1, \ldots, b_n\}$ (*boys*) i $G = \{g_1, \ldots, g_n\}$ (*girls*), a $E \subseteq B \times G$ (relacja znajomości lub, powiedzmy, sympatii).

Pytanie: czy istnieje relacja $f \subseteq E$ stanowiąca *bijekcję* B na G? (Taką relację nazywamy "doskonałym skojarzeniem", ang. *perfect matching*.)

Wielomianowy algorytm rozstrzygający ten problem jest nieco mniej oczywisty niż w dwóch poprzednich przykładach. Poszukiwany zbiór f obliczamy etapami. Załóżmy, że do tej pory obliczyliśmy f, które jest bijekcją ze swojej dziedziny $\mathrm{dom}\, f$ w swój zbiór wartości $\mathrm{Im}\, f$, ale $\mathrm{dom}\, f$ nie jest jeszcze całym B. Rozważmy graf otrzymany z $\langle V, E\rangle$ przez zastąpienie każdej krawędzi $(b, g) \in f$ przez krawędź przeciwną (g, b) (przypomnijmy, że w E były wyłącznie krawędzie z B do G). W tym grafie poszukujemy ścieżki z jakiegoś $b \notin \mathrm{dom}\, f$ do jakiegoś $g \notin \mathrm{Im}\, f$. (Algorytm z poprzedniego przykładu.) Jeśli znajdziemy taką ścieżkę, powiedzmy P z b_0 do g_0, to nowa wartość f, powiedzmy f' jest sumą f i P minus wszystkie pary krawędzi (b, g), (g, b), gdzie $(g, b) \in P$. Zauważmy, że f' jest bijekcją (jeśli z jakiegoś b wychodzi "nowa" strzałka, to "stara" została usunięta!), a ponadto $\mathrm{dom}\, f' = \mathrm{dom}\, f \cup \{b_0\}$.

Dla dowodu poprawności algorytmu, przypuśćmy, że częściowa bijekcja f nie daje się rozszerzyć w powyższy sposób *pomimo, że* istnieje doskonałe skojarzenie h. Wybierzmy dowolne $b_0 \notin \mathrm{dom}\, f$ i poprowadźmy z tego wierzchołka ścieżkę, która używa wyłącznie krawędzi z $h - f$ lub odwróconych krawędzi z f. Jest to możliwe tylko na jeden sposób i nietrudno wykazać, że ta ścieżka osiągnie w końcu wierzchołek w $G - \mathrm{Im}\, f$. A zatem *było możliwe* rozszerzenie f, wbrew przypuszczeniu.

Uwaga. Opisany wyżej algorytm *mutatis mutandis* może być również zastosowany do rozwiązania problemu optymalizacyjnego *znalezienia* skojarzenia o maksymalnej możliwej mocy.

Problemy NP zupełne — wyzwanie dla algorytmiki

Poprzedni przykład jest szczególnym przypadkiem dość typowego schematu: zadanie polega na znalezieniu jakiegoś obiektu zależnego od danych wejściowych (w tamtym przypadku – doskonałego skojarzenia w danym grafie), przy czym liczba potencjalnych kandydatów wyklucza możliwość efektywnego wyczerpującego przeszukiwania. Z drugiej strony, rozmiar poszukiwanego obiektu jest wielomianowy w stosunku do rozmiaru danych wejściowych, a co więcej, sprawdzenie, czy dany kandydat jest poszukiwanym obiektem może również być wykonane efektywnie (w naszym przykładzie, sprawdzenie, czy jakaś relacja jest bijekcją z B na G zawartą w E jest trywialne). Znalezienie rozwiązania, czy choćby rozpoznanie, czy rozwiązanie istnieje, wymaga więc jakiegoś "tricku".

Podamy teraz kilka przykładów pasujących do tego schematu, dla których jednak żaden algorytm pracujący w czasie wielomianowym nie jest znany.

Przykład – spełnialność formuł. Dana formuła rachunku zdań (napisana z użyciem spójników logicznych $\vee$, $\wedge$, $\neg$ i zmiennych). Pytanie: czy ta formuła jest *spełnialna*, tj. czy istnieje wartościowanie zmiennych w zbiorze wartości logicznych $\{0, 1\}$, przy którym formuła ma wartość 1. (Tzw. problem SAT.)

Problem pozostaje trudny również po ograniczeniu pytania do formuł danych w szczególnej postaci: mianowicie jako koniunkcje tzw. *klauzul*, gdzie każda klauzula jest z kolei alternatywą pewnej liczby tzw. *literałów*, gdzie literał jest zmienną lub jej negacją (problem *CNF SAT*).

Co więcej, problem pozostaje trudny, jeśli nałożyć dodatkowe ograniczenie, że w każdej klauzuli występują co najwyżej *trzy* literały (problem *3–CNF SAT*), jak np. w formule

$$(x_1 \vee \neg x_2 \vee x_3) \ \wedge \ (\neg x_1 \vee \neg x_2 \vee x_3) \ \wedge \ (x_1 \vee \neg x_2 \vee \neg x_3)$$

Natomiast, jeśli ograniczyć pytanie do koniunkcji klauzul o jedynie *dwóch* literałach, problem jest wielomianowo rozstrzygalny; podobnie po ograniczeniu do tzw. *klauzul Horna* (tj. klauzul postaci $x_1 \wedge \ldots \wedge x_k \Rightarrow y$, k bez ograniczeń).

Analogiczny problem dla formuł będących *alternatywami* koniunkcji literałów jest w łatwy sposób wielomianowo rozstrzygalny (dlaczego?).

Przykład – grafy Hamiltona. Dany graf skierowany. Pytanie, czy istnieje *cykl Hamiltona* tj. pętla, na której każdy wierzchołek występuje dokładnie raz.

W ogólniejszej wersji tego problemu krawędziom pzyporządkowane są wagi i pytanie brzmi: czy istnieje cykl Hamiltona o łącznej sumie wag niewiększej niż dana liczba k. (Tzw. problem komiwojażera; ang. *Traveling Salesman Problem.*) W wersji funkcyjnej (optymalizacyjnej) chcemy *znaleźć* cykl Hamiltona o *najmniejszej* wadze.

Przykład – kolorowalność. Ustalone $k \geq 3$, dany graf. Pytamy, czy wierzchołki grafu da się pokolorować k kolorami tak, by każde dwa wierzchołki połączone krawędzią miały różne kolory. Dla $k = 3$, pytanie jest trudne nawet dla grafów *planarnych*.[10]

(Dla $k \geq 4$ i grafów planarnych problem jest rozwiązany przez Twierdzenie o Czterech Barwach.)

Powyższe problemy należą do klasy złożoności *NP*, którą możemy opisać w następujący sposób. Powiemy, że relacja $r \subseteq \Sigma^* \times \Sigma^*$ jest *wielomianowo zrównoważnona*, jeśli istnieje wielomian $p(n)$, taki że, dla każdych $w, v \in \Sigma^*$, $(w, v) \in r \Rightarrow |v| \leq p(|w|)$. Relacja r jest w klasie P, jeśli język $\{w\sharp v : (w, v) \in r\}$ jest w P (gdzie $\sharp$ jest symbolem spoza Σ). Powiemy, że język L jest w klasie NP, jeśli istnieje wielomianowo zrównoważona relacja r w klasie P, taka, że

$$L \ = \ \{w : (\exists v)\,(w, v) \in r\}$$

(Równoważną definicję można podać w oparciu o pojęcie *niedeterministycznej* maszyny Turinga pracującej w czasie wielomianowym.)

Problemy z powyższych przykładów są w klasie NP.

Przykład – tautologie. Rozważmy wspomniany na wstępie problem, czy dana formuła φ jest tautologią rachunku zdań. Czy ten problem jest w *NP*, tj. czy możemy dobrać odpowiednią relację r ? Ktoś mógłby zaproponować pary (φ, p), gdzie p jest *dowodem* formuły φ w jakimś formalnym systemie dowodzenia dla rachunku zdań. Istotnie, dla rozsądnych systemów relacja taka jest w P; nie znamy jednak żadnego systemu, w którym długość dowodu można ograniczyć wielomianem od długości formuły, tak więc nie wiemy, czy relację taką można by wielomianowo zrównoważyć.

W istocie, przypuszcza się, że problem tautologii *nie* jest w NP.

[10] Pytanie, czy dany graf $\langle V, E\rangle$ jest planarny jest wielomianowo rozstrzygalne dzięki twierdzeniu Kuratowskiego.

Oczywiście $P \subseteq NP$, ale nie wiemy, czy $P \neq NP$.

Okazuje się, że większość znanych problemów w klasie *NP* ma pewną własność uniwersalności, zwaną *NP–zupełnością*. Jej najważniejszą konsekwencją jest, że *gdybyśmy* znali algorytm działający w czasie wielomianowym dla przynajmniej jednego problemu *NP*–zupełnego, to potrafilibyśmy zbudować podobny algorytm *dla każdego* problemu w NP, w szczególności otrzymalibyśmy $P = NP$.

Pojęcie problemu *NP*–zupełnego określamy następująco. Funkcja $f : \Sigma^* \to \Gamma^*$ *wielomianowo redukuje* problem $A \subseteq \Sigma^*$ do problemu $B \subseteq \Gamma^*$, jeśli f jest w P oraz $w \in A \Leftrightarrow f(w) \in B$, dla wszystkich $w \in \Sigma^*$. Problem L jest *NP–zupełny* jeśli jest w klasie *NP*, a ponadto każdy problem z tej klasy redukuje się wielomianowo do L.

(Dokładniej, jest to tzw. *NP*–zupełność *w sensie Karpa*. Pojęcie *NP*–zupełności można zawęzić zawężając klasę dopuszczalnych redukcji.)

Wszystkie wspomniane do tej pory problemy, które są w *NP*, a o których nie wiadomo, czy są w P, są *NP*–zupełne. Wkrótce zobaczymy jednak, że może być inaczej.

Czytelnik może jako ćwiczenie udowodnić najpierw *NP*–zupełność problemu spełnialności dla układów logicznych, używając techniki przekształcania obliczenia maszyny Turinga w obwód logiczny, wspomnianej przy okazji omawiania przykładu obwodów logicznych. Z kolei, ten problem można zredukować do SAT, dowodząc *NP*–zupełności problemu SAT. (Uwaga: nie należy, dla danego obwodu, konstruować równoważnej formuły, lecz formułę, której *spełnialność* jest równoważna spełnialności obwodu.)

Problemy *NP* ∩ *co–NP*. Liczby pierwsze

Można przypuszczać się, że jeśli język $L \subseteq \Sigma^*$ jest *NP*–zupełny, to jego uzupełnienie $\bar{L} = \Sigma^* - L$ nie jest w tej klasie. (Na przykład uzupełnienie problemu SAT można łatwo zredukować do problemu tautologii, który, jak wspomnieliśmy, *przypuszczalnie* nie jest w *NP*.) Oczywiście, jeśli język jest w P, to jego uzupełnienie również jest w P. Znamy zaledwie kilka przykładów takich języków L, że L i $\bar{L}$ są w *NP*, choć nie wiemy, czy L jest w P. Jednym z najciekawszych jest problem liczb pierwszych.

Przykład – złożoność, pierwszość. Dana liczba n *w postaci binarnej*. Pytanie: czy n jest liczbą złożoną? Oczywiście ten problem jest w *NP*, wystarczy bowiem "zgadnąć" nietrywialny dzielnik n. Innymi słowy, za relację r można przyjąć zbiór par (n, d), gdzie d jest dzielnikiem n różnym od 1 i od n (plus para $(1, 1)$). Własność, czy d dzieli n jest sprawdzalna w czasie wielomianowym (względem *długości* zapisów liczb n i d w postaci binarnej) za pomocą szkolnego algorytmu.

Okazuje się, że również problem *czy dana liczba n w (postaci binarnej) jest pierwsza?* jest w NP. Zagadnienie wydaje się trudniejsze, bo w pierwszej chwili nie widać, jaki obiekt mógłby pełnić rolę "świadka" pierwszości liczby n (tj. rolę analogiczną do tej, jaką w poprzednich przykładach pełniły wartościowanie, cykl Hamiltona, kolorowanie lub dzielnik). Jednak V. Pratt (w 1975 r.) podał argument, wykorzystujący Małe Twierdzenie Fermata.

Twierdzenie. Liczba nieparzysta $p > 2$ jest pierwsza wtedy i tylko wtedy, gdy zbiór $\{1, 2, \ldots, p-1\}$ z mnożeniem $\pmod p$ tworzy grupę. Jeśli tak jest, to w grupie tej istnieje element rzędu $p - 1$.

Twierdzenie sugeruje, że należałoby poszukiwać $x < p$, takiego że $x^{p-1} \equiv 1 \pmod{p}$, ale $x^i \not\equiv 1 \pmod{p}$, dla $0 < i < p-1$. Czy jednak, mając takie x, potrafimy w czasie wielomianowym (od rozmiaru p tj. $\log p$) *sprawdzić*, że ma ono żądane własności?

Zauważmy najpierw, że dla danego x, pierwszy warunek potrafimy sprawdzić w czasie proporcjonalnym do $(\log p)^3$. (Uwaga: nie należy wykonywać $p-1$ mnożeń, lecz obliczyć pomocniczo $x^2 \bmod p$, $x^4 \bmod p$, $x^8 \bmod p$, itd. i pomnożyć przez siebie mod$p$ odpowiednie potęgi, wykonując w ten sposób $O(\log p)$ mnożeń.)

Większa trudność tkwi w warunku negatywnym. Nietrudno jednak wykazać, że jeśli istnieje $0 < i < p-1$, dla którego $x^i \equiv 1 \pmod{p}$, to można znaleźć takie i pośród dzielników $p-1$. Dlatego, przy sprawdzaniu tego warunku, można się ograniczyć do dzielników liczby $p-1$, co więcej, można się ograniczyć do dzielników "maksymalnych" tj. postaci $\frac{p-1}{p_j}$, gdzie p_j jest czynnikiem pierwszym liczby $p-1$. (Bo jeśli $x^i \equiv 1 \pmod{p}$, to również $x^{ki} \equiv 1 \pmod{p}$.)

Tych ostatnich jest już niewiele (nie więcej niż $\log_2 p$), jak jednak mamy się przekonać, że p_j istotnie jest *pierwsze*? Wydaje się, że wpadliśmy w błędne koło, bo właśnie ten problem chcieliśmy rozwiązać. Tak jednak nie jest, bo $p_j < p$.

Ta ostatnia uwaga sugeruje postać *krótkiego certyfikatu pierwszości liczby* p zaproponowanego przez Pratta.

Możemy przedstawić go jako drzewo, którego każdy wierzchołek ma etykietę postaci

$$\langle q, x, p_1^{\alpha_1} \cdot \ldots \cdot p_k^{\alpha_k} \rangle$$

gdzie q jest pewną liczbą pierwszą, x jest elementem rzędu $q-1$ z Twierdzenia Fermata, a $q-1 = p_1^{\alpha_1} \cdot \ldots \cdot p_k^{\alpha_k}$ stanowi rozkład $q-1$ na czynniki pierwsze. W korzeniu drzewa, $q = p$.

Wierzchołek o etykiecie jak powyżej będzie miał k synów (następników), których etykiety rozpoczynają się od $p_1, \ldots, p_k$, odpowiednio. (Dla $q = 2$, $k = 0$ i wierzchołek jest liściem.)

Na przykład, dla $p = 67$, istnieje certyfikat o następujących wierzchołkach (pozostawiamy Czytelnikowi "rozrysowanie" grafu): $\langle 67, 2, 2 \cdot 3 \cdot 11 \rangle$, $\langle 2, 1, 1 \rangle$ (ten wierzchołek występuje 4 razy), $\langle 3, 2, 2 \rangle$, $\langle 11, 8, 2 \cdot 5 \rangle$, $\langle 5, 3, 2 \cdot 2 \rangle$.

Nietrudno jest sprawdzić, że certyfikat dla liczby p ma nie więcej niż $3 \log_2 p$ wierzchołków, a informacja zawarta w etykiecie każdego wierzchołka ma długość nie większą niż $5 \log_2 p$. Tak więc rozmiar certyfikatu można oszacować wielomianem od $\log p$ (a więc od rozmiaru p). Na mocy powyższych rozważań, sprawdzenie, czy dane słowo *jest* poprawnym certyfikatem pierwszości liczby p może być również wykonane w czasie ograniczonym przez wielomian od $\log p$.

Ta uwaga kończy dowód przynależności problemu liczb pierwszych do klasy NP, a zatem także do $NP \cap co\text{–}NP$, gdzie $co\text{–}\mathcal{K}$ oznacza klasę uzupełnień problemów z klasy $\mathcal{K}$.

Istnienie rozwiązania a liczba rozwiązań

Jeśli $r \subseteq \Sigma^* \times \Sigma^*$ jest wielomianowo zrównoważoną relacją w klasie P, to dla danego $w \in \Sigma^*$ możemy pytać nie tylko o istnienie, ale także o *liczbę* takich v, że $(w, v) \in r$. Na przykład, możemy pytać o liczbę wartościowań spełniającą daną formułę, o liczbę cyklów Hamiltona lub 3–kolorowań w grafie, o liczbę dzielników lub certyfikatów pierwszości danej liczby (zauważmy, że certyfikaty mogą się różnić doborem elementów z tw. Fermata). Jeśli relacja r jest zrównoważona poprzez wielomian $p(n)$, to dla danego w poszukiwana liczba jest nie większa od $|\Sigma|^{p(|w|)}$, a zatem jej długość w postaci binarnej jest proporcjonalna "zaledwie" do $p(|w|)$. *A priori* mógłby więc istnieć algorytm, który dla danego w w czasie wielomianowym oblicza tę liczbę. Jednak, jak możemy oczekiwać, dla większości naszych przykładów taki algorytm nie jest znany. Co więcej, problem znalezienia liczby rozwiązań wydaje się być istotnie trudniejszy od pytania, czy istnieje rozwiązanie.

Również i w tej kategorii możemy wskazać problemy w pewnym sensie zupełne. Np. można wykazać, że dla każdej wielomianowo zrównoważonej relacji $r \subseteq \Sigma^* \times \Sigma^*$ w klasie P, można znaleźć wielomianowo obliczalną funkcję (zależną od r) $w \mapsto \varphi_w$, gdzie φ_w jest formułą rachunku zdań, taką, że liczba wartościowań spełniających formułe φ_w jest równa liczbie v, takich, że $(w, v) \in r$.

Tak więc, pytanie o liczbę rozwiązań dla problemu r da się sprowadzić do pytania o liczbę wartościowań spełniających daną formułę rachunku zdań.

Dość zadziwiającym odkryciem dokonanym przez G.L. Valianta (w 1979 r.) było, że tę samą własność mogą mieć również problemy "łatwiejsze", leżące w klasie P.

Przypomnijmy "grafy sympatii" rozważane w przykładzie o doskonałych skojarzeniach. Okazuje się, że problem znalezienia dla takiego grafu *liczby wszystkich* doskonałych skojarzeń jest równie trudny jak problem znalezienia liczby wartościowań spełniających formułę. Tymczasem, jak pamiętamy, pytanie, czy *istnieje* skojarzenie może być rozstrzygnięte w czasie wielomianowym.

W nieco ogólniejszej wersji, zagadnienie to jest znane jako problem obliczania *permanentu macierzy*, gdyż liczbę doskonałych skojarzeń w grafie $\langle V, E\rangle$, gdzie $V = B \cup G$, $B = \{b_1, \ldots, b_n\}$, $G = \{g_1, \ldots, g_n\}$ i $E \subseteq B \times G$, można przedstawić

$$\text{perm } A^E \quad = \quad \sum_{\pi} \prod_{i=1}^{n} A^E_{i,\pi(i)}$$

gdzie π przebiega wszystkie permutacje zbioru $\{1, \ldots, n\}$, a A^E jest macierzą $n \times n$ określoną

$$\begin{aligned} A^E_{i,j} &= 1 \text{ gdy } (b_i, g_j) \in E \\ &= 0 \text{ w przec. przyp.} \end{aligned}$$

Należy więc przypuszczać, że obliczenie permanentu macierzy jest zadaniem trudnym. Dla kontrastu warto przypomnieć, że *wyznacznik* macierzy

$$\text{perm } A^E \;=\; \sum_{\pi} \sigma(\pi) \cdot \prod_{i=1}^{n} A^E_{i,\pi(i)}$$

może być obliczony w czasie wielomianowym np. znaną metodą Gaussa.

Rozważmy jeszcze pytanie, jak problem obliczania liczby rozwiązań ma się do zagadnienia *znajdowania* (jakiegoś) rozwiązania. Okazuje się, że problemy te są w ogólności nieporównywalne.

W jednym kierunku, przykładu dostarcza omówiony przed chwilą problem skojarzeń. Jak bowiem już zauważyliśmy, wielomianowy algorytm dla tego problemu nie tylko rozstrzyga o istnieniu, ale także *znajduje* pewne konkretne rozwiązanie (o ile istnieje).

Z kolei zobaczymy przykład odwrotny, gdzie znajdowanie rozwiązania jest trudne, a liczba rozwiązań może być obliczona łatwo.

Otóż M. R. Fellows i N. Koblitz wykazali (w 1992 r.), że można tak zmodyfikować pojęcie certyfikatu pierwszości, by każda liczba pierwsza miała *dokładnie jeden* certyfikat. Wykazali oni mianowicie, że na podstawie znajomości rozkładu na czynniki pierwsze liczby $p-1$, można już w czasie wielomianowym orzec, czy liczba p jest pierwsza. Rozważmy problem znajdowania dla liczby n (danej w postaci binarnej) jej rozkładu na czynniki pierwsze, dane wraz ze swoimi certyfikatami pierwszości. (Bez obecności certyfikatów, nie byłoby jasne, że relacja $\langle$ liczba, rozkład $\rangle$ jest w P.) Wyznaczenie liczby rozwiązań jest w tym problemie trywialne, bo każda liczba naturalna ma dokładnie jeden taki rozkład!

Z drugiej strony, żaden wielomianowy algorytm dla problemu faktoryzacji nie jest znany. (Hipoteza nieistnienia takiego algorytmu leży u podstaw wiary w bezpieczeństwo systemów kryptograficznych z jawnym kluczem.)

Literatura

C.H. Papadimitriou, *Computational complexity*, Addison–Wesley, 1995.
J.E. Hopcroft, J.D. Ullman, *Wprowadzenie do teorii automatów, języków i obliczeń*; tłum. Beata Konikwska, PWN, Warszawa 1994 (Addison–Wesley, 1979).
A. Hodges, *Alan Turing: The enigma of intelligence*, Unwin, London, 1983.
R. Penrose, *Nowy umysł cesarza*, tłum. P. Amsterdamski, PWN, Warszawa 1995 (Oxford University Press 1989).
N. Koblitz, *Wykład z teorii liczb i kryptografii*, tłum. W. Guzicki, WNT, Warszawa, 1995.

Adres autora: Damian Niwiński, Instytut Informatyki Uniwersytetu Warszawskiego, ul. Banacha 2, 02 097 Warszawa
e-mail: niwinski@mimuw.edu.pl