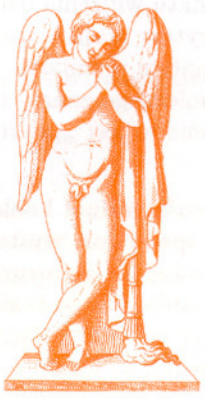
Czy $P \neq NP$? – wyzwanie dla algorytmików XXI wieku

Krzysztof DIKS

Nasze rozważania na temat problemu „Czy $P \neq NP$?", w skrócie problemu PNP, rozpoczniemy od podania algorytmu rozwiązującego następujące zadanie:

Dane: Formuła boolowska ϕ w postaci koniunkcji zmiennych lub ich negacji.

Pytanie: Czy formuła ϕ jest spełnialna, tzn. czy istnieje takie wartościowanie zmiennych, dla którego formuła ϕ przyjmuje wartość 1 (prawda)?



Przykład: Formuła $x_1 \wedge \neg x_2 \wedge x_3 \wedge x_1$ *przyjmuje wartość* 1 *tylko dla wartościowania* $x_1 = 1, x_2 = 0, x_3 = 1$.

Łatwo zauważyć, że formuła ϕ (w postaci z powyższego zadania) jest spełnialna wtedy i tylko wtedy, gdy nie występują w niej jednocześnie zmienna i jej negacja. Ta obserwacja pozwala sformułować bardzo prosty algorytm rozwiązujący nasze zadanie:

> Dla każdej zmiennej x z ϕ sprawdź, czy w ϕ występują jednocześnie x i $\neg x$. Jeśli taka zmienna nie istnieje, to ϕ jest spełnialna. W przeciwnym razie ϕ nie jest spełnialna.

Sprawny programista bardzo szybko zaprogramuje powyższy algorytm. Musimy tylko sprecyzować, co oznacza sformułowanie *Dana jest formuła boolowska*. Innymi słowy, musimy podać *rozsądny* sposób kodowania formuł. Jednym z możliwych może być następujący sposób kodowania: Przyjmujemy, że zmienne występujące w formule są ponumerowane kolejno $1, 2, \ldots$ Formułę kodujemy jako ciąg liczb oddzielonych średnikami. Pierwszą liczbą w ciągu jest liczba n równa liczbie literałów w formule. Po niej występuje n liczb ze zbioru $\{-n, -(n-1), \ldots, -1, 1, 2, \ldots, n\}$. Jeśli i-tym literałem jest zmienna x_k, to za i-tą spośród tych liczb bierzemy k, jeśli zaś i-tym literałem jest $\neg x_k$, to i-tą liczbą będzie $-k$.

Literał to wystąpienie zmiennej lub jej negacji. W formule z przykładu mamy cztery literały.

Kodem przykładowej formuły jest $4; 1; -2; 3; 1$.

W każdym rozsądnym kodowaniu przyjmuje się ponadto, że do kodowania liczb stosujemy dowolny zapis o podstawie co najmniej 2 (najczęściej właśnie o podstawie 2). Zauważmy, że przy takim kodowaniu długość kodu formuły wynosi co najwyżej $cn \log n$, dla pewnej stałej c. Długość kodu danych będziemy nazywali *rozmiarem* zadania. W przedstawionym powyżej algorytmie literały są porównywane między sobą. Nawet przy bardzo naiwnej implementacji tego algorytmu liczba takich porównań wyniesie co najwyżej n^2. Jeśli uwzględnimy, że porównywanie kodów dwóch literałów wymaga $\log n$ porównań bitów, to liczbę wszystkich operacji da się ograniczyć przez $n^2 \log n$, a idąc dalej możemy powiedzieć, że liczbę operacji wykonywanych przez algorytm da się ograniczyć przez r^k, gdzie r jest rozmiarem danych, a k stałą całkowitą większą od 0. (W naszym przypadku za k można wziąć 3.) W takim przypadku mówimy, że algorytm rozwiązuje zadanie w czasie wielomianowym.

> Klasą P nazywamy zbiór tych zadań algorytmicznych, dla których istnieją algorytmy rozwiązujące je w czasie wielomianowym.

Utrudnijmy teraz nasze wyjściowe zadanie. Powiemy, że formuła boolowska jest *w postaci koniunkcyjno-normalnej* (z ang. *w postaci CNF*), jeśli jest koniunkcją formuł (klauzul), z których każda jest alternatywą zmiennych lub ich negacji, być może zdegenerowaną do jednego literału. Formuła jest w *postaci k-CNF*, jeśli w każdej klauzuli występuje co najwyżej k literałów.

Proszę wykazać, że każdą formułę boolowską można przekształcić do równoważnej (ze względu na spełnialność) formuły w postaci 3-*CNF* o długości tylko wielomianowo większej od długości formuły wyjściowej.

Oto przykład formuły w postaci 2-*CNF:*

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

Proszę spróbować znaleźć algorytm, który rozwiązuje to zadanie w czasie liniowym, tj. w czasie proporcjonalnym do długości formuły.

Pokazaliśmy, że zadanie spełnialności formuł w postaci 1-*CNF* można rozwiązać w czasie wielomianowym. To stwierdzenie pozostaje w mocy także dla formuł w postaci 2-*CNF*. Sytuacja zmienia się diametralnie, gdy weźmiemy $k \geq 3$. Nie są znane algorytmy, które rozwiązywałyby to zadanie w czasie wielomianowym, nawet dla wielomianów bardzo dużego stopnia, np. 1000. Najlepsze znane algorytmy wymagają czasu co najmniej c^r, gdzie c jest stałą większą od 1, a r jest długością kodu formuły. (Myślę, że Czytelnik nie będzie miał żadnego problemu ze znalezieniem rozsądnego kodowania formuł w postaci koniunkcyjno-normalnej.) O takich algorytmach mówimy, że działają w czasie wykładniczym. Z praktycznego punktu widzenia oznacza to, że nawet na współczesnych komputerach takie algorytmy mają szanse dać wynik w rozsądnym czasie tylko dla danych o bardzo małym rozmiarze. Można zaryzykować stwierdzenie, że jeżeli dla zadania algorytmicznego znamy tylko rozwiązania działające w czasie wykładniczym, to zadanie to jest „praktycznie algorytmicznie nierozwiązywalne".

Jaką interesującą własność ma jeszcze zadanie spełnialności formuł boolowskich? Gdyby ktoś chciał przekonać nas, że dana formuła jest spełnialna, wystarczy, żeby podał odpowiednie wartościowanie zmiennych. Zauważmy, że rozmiar takiego wartościowania nie jest większy od długości formuły. Mając takie wartościowanie, w czasie wielomianowym można obliczyć odpowiadającą mu wartość logiczną formuły. Jeśli tą wartością jest 1 (prawda), to formuła jest spełnialna. Wartościowanie, dla którego formuła jest spełnialna, nazywamy *świadectwem* spełnialności. Algorytm, który sprawdza spełnialność formuły dla danego wartościowania, nazywamy *algorytmem weryfikacji*. Innymi słowy, algorytmu weryfikacji można użyć do wykazania w czasie wielomianowym, że dana formuła jest spełnialna, jeżeli tylko istnieje i dane jest odpowiednie świadectwo.

Skróty P i NP pochodzą z angielskiego, odpowiednio, *polynomial time* i *nondeterministic polynomial time*. Niedeterminizm dotyczy pochodzenia świadectwa, ponieważ nie żąda się podania metody jego konstrukcji.

> Klasą NP nazywamy zbiór tych zadań algorytmicznych, które można weryfikować w czasie wielomianowym.

Łatwo zauważyć, że każde zadanie z P należy do NP, ponieważ zadanie takie można zawsze rozwiązać w czasie wielomianowym i do weryfikacji nie potrzebujemy żadnego świadectwa. Zatem $P \subseteq NP$, a problem PNP to problem

$$\text{Czy } P \supseteq NP?$$

Do klasy NP należy tysiące ważnych, praktycznych zadań algorytmicznych, o których nie wiadomo, czy należą do P. Przyjrzyjmy się jeszcze jednemu takiemu zadaniu.

Dane: Nieskierowany graf $G = (V, E)$ oraz liczba naturalna k.

Pytanie: Czy w G istnieje klika rozmiaru k, tzn. czy w G istnieje podgraf k-wierzchołkowy, w którym każda para różnych wierzchołków jest połączona krawędzią?

Problem kliki z pewnością należy do NP. Dla danego k-elementowego podzbioru wierzchołków (świadectwa) można łatwo w czasie wielomianowym, zależnym tylko od rozmiaru grafu, sprawdzić, czy wierzchołki te tworzą klikę. Wykażemy teraz, że gdybyśmy w czasie wielomianowym potrafili rozwiązać zadanie spełnialności, to także w czasie wielomianowym można by rozwiązać zadanie kliki. W tym celu zadanie kliki sprowadzimy do zadania formuł boolowskich. (Poniższą konstrukcję powtarzamy za [5].)

Dla każdego wierzchołka $v \in V$ wprowadzamy k zmiennych boolowskich $x_1^v, x_2^v, \ldots, x_k^v$. Zmienna x_i^v intuicyjnie mówi, że wierzchołek v jest i-tym wierzchołkiem w poszukiwanej klice. Skonstruujemy formułę ϕ, która jest koniunkcją trzech formuł ϕ_1, ϕ_2 i ϕ_3. Oto intuicyjne znaczenie i formalne definicje tych formuł:

- ϕ_1 = „Dla każdego i, $1 \leq i \leq k$, istnieje co najmniej jeden wierzchołek $u \in V$, który jest i-tym wierzchołkiem w klice."

$$\phi_1 = \bigwedge_{i=1}^{k} \left(\bigvee_{v \in V} x_i^v \right).$$

- ϕ_2 = „Dla każdego i, $1 \leq i \leq k$, żadne dwa wierzchołki nie są jednocześnie i-tymi wierzchołkami w klice".

$$\phi_2 = \bigwedge_{i=1}^{k} \bigwedge_{u,v \in V, u \neq v} (\neg x_i^u \vee \neg x_i^v).$$

- ϕ_2 = „Dla każdej pary u, v, jeśli u–v nie jest krawędzią w grafie, to u i v nie są jednocześnie w klice".

$$\phi_3 = \bigwedge_{u-v \notin E} \bigwedge_{1 \leq i,j \leq k} (\neg x_i^u \vee \neg x_j^v).$$

Wystarczy wykazać wielomianową redukcję z zadania spełnialności do zadania kliki.

Pozostawiamy Czytelnikowi wykazanie, że formuła ϕ jest spełnialna wtedy i tylko wtedy, gdy w grafie G istnieje klika rozmiaru k. Łatwo zauważyć, że rozmiar powstałej formuły jest wielomianowo zależny od rozmiaru grafu, a samą formułę można skonstruować w czasie wielomianowym. W tym przypadku mówimy, że zadanie kliki jest *redukowalne w czasie wielomianowym* do zadania spełnialności. W 1971 roku R. Cook [2] udowodnił, że każde zadanie z NP można w czasie wielomianowym zredukować do zadania spełnialności pewnej formuły boolowskiej. Zadanie, które należy do klasy NP i do którego można zredukować w czasie wielomianowym każde inne zadanie z NP, nazywamy zadaniem NP-zupełnym. W tym sensie zadanie spełnialności jest NP-zupełne. Pozostawiamy Czytelnikom wykazanie, że zadanie kliki jest też NP-zupełne. Rozwiązanie dowolnego zadania NP-zupełnego w czasie wielomianowym pozwalałoby rozwiązywać w czasie wielomianowym każde zadanie z NP.

Pojęcie klasy P wprowadzili niezależnie Cobham [1] i Edmonds [3] w połowie lat sześćdziesiątych. Edmonds wprowadził też pojęcie klasy NP i jako pierwszy sformułował pytanie, czy $P \neq NP$. Metoda redukcji pochodzi od Karpa [4], który za jej pomocą wykazał, że wiele ważnych zadań kombinatorycznych i optymalizacyjnych jest NP-zupełnych. Czytelnika zainteresowanego problemem PNP odsyłamy do lektury 36. rodziału książki T.C. Cormena, C.E. Leisersona, R.L. Rivesta, *Wprowadzenie do algorytmów*, WNT, 1998. Dowód twierdzenia Cooka można znaleźć w książce A.V. Aho, J.E. Hopcrofta, J.D. Ullmana, *Projektowanie i analiza algorytmów komputerowych*, PWN, 1983.

BIBLIOGRAFIA

[1] A. Cobham. The intrinsic computational complexity difficulty of functions, w *Proceedings of the 1964 Congress for Logic, Methodology, and the Philosophy of Science*, ss. 24–30. North-Holland, 1964.

[2] S. Cook. The complexity of theorem proving procedures, w *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ss. 151–158, 1971.

[3] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17: ss. 449–467, 1971.

[4] R.M. Karp. Reducibility among combinatorial problems, w Raymond E. Miller and James W. Thatcher, editors, *Complexity of Computer Computations*, ss. 85–103. Plenum Press, 1972.

[5] D.C. Kozen. The design and analysis of algorithms. Springer-Verlag, 1991.

Pod koniec lat 70. potwierdzone zostało występowanie cyklów aktywności magnetycznej na gwiazdach typów widmowych F i G. Stało się to możliwe dzięki wieloletnim (trwającym od 1966 r.) i systematycznym obserwacjom linii widmowych (absorpcyjnych) zjonizowanego wapnia (Ca II H+K) prowadzonych w obserwatorium Mount Wilson. Linie te są bardzo czułym wskaźnikiem aktywności. W ich środkach pojawiają się linie emisyjne, których natężenie zależy od stopnia aktywności gwiazdy. Zaobserwowane wahania natężenia emisji wskazują na fakt, że wiele gwiazd przechodzi okresy zmniejszonej i nasilonej aktywności, podobnie jak Słońce. Długości odkrytych dotychczas cyklów są wprawdzie zbliżone do typowej długości cyklu słonecznego (wynoszą od siedmiu do kilkunastu lat), jednak obserwacje trwają zbyt krótko, by wykluczyć występowanie cyklów o okresach dwudziestu lat i dłuższych.

W 1883 r. astronom Królewskiego Obserwatorium Greenwich, Edward W. Maunder, wysunął hipotezę podważającą przekonanie o niezmienności cyklu słonecznego. Zauważył mianowicie, że w publikacjach astronomicznych z końca XVII i początku XX w. praktycznie nie ma opisów plam słonecznych, co miało świadczyć – jego zdaniem – o zaniku aktywności plamotwórczej Słońca na około 70 lat. Swoje spostrzeżenia opublikował po raz pierwszy w 1894 roku i ponownie w 1922 roku. Żadna z tych publikacji nie została przyjęta poważnie, gdyż jak (zresztą słusznie) uważano: „brak raportów o plamach nie musi świadczyć o braku plam". Dopiero w 1976 roku John E. Eddy potwierdził przypuszczenia Maundera. Nie tylko udokumentował jego hipotezę, lecz wskazał także na istnienie innych, wcześniejszych okresów zaniku aktywności Słońca.