Klasy problemów, NP-zupełność

Klasa P

Problemy decyzyjne – takie problemy, dla których rozwiązaniem jest decyzja **TAK** / **NIE** (np. problem: sprawdzić, czy dany graf jest spójny).

Przez **P** oznaczamy klasę (zbiór) problemów dla których istnieje algorytm wielomianowy, tzn. algorytm o złożoności czasowej $O(n^c)$ dla pewnej stałej c. Nie wszystkie problemy są w klasie P.

Przykład problemu poza klasą P: Problem stopu.
Dane: program w Pascalu
Problem: stwierdzić, czy program się zatrzyma.

Twierdzenie. *Nie istnieje żaden algorytm rozwiązujący problem stopu.*

Problem SAT

Dana jest formuła logiczna ϕ, zbudowana poprawnie ze zmiennych $(\alpha, \beta, \gamma, \ldots)$, spójników logicznych $(\vee, \wedge, \neg)$ i nawiasów.

Przykład. $\phi_0 = (\alpha \wedge \beta) \vee (\beta \wedge \neg\gamma)$.

Formuła ϕ jest *spełnialna*, jeśli można tak podstawić wartości logiczne pod zmienne z ϕ, żeby formuła ϕ miała wartość **True**. Układ wartości logicznych, które podstawiamy pod zmienne nazywamy *wartościowaniem.*

Przykład.

- ϕ_0 jest spełnialna; wartościowanie: $\{\alpha \leftarrow False, \beta \leftarrow True, \gamma \leftarrow True\}$,
- $\alpha \wedge \neg\alpha$: nie jest spełnialna.
- $(\alpha \vee \beta) \wedge \neg\alpha \wedge \neg\beta$: nie jest spełnialna.

Problem SAT: Rozstrzygnąć czy dana formuła ϕ jest spełnialna.

Algorytm dla problemu SAT. Algorytm sprawdza wszystkie możliwe wartościowania zmiennych z formuły ϕ. Dla każdego wartościowania oblicza wartość formuły ϕ. Jeśli w pewnym momencie dostanie wartość True, odpowiada TAK. Jeśli dla każdego wartościowania wartość formuły była False, odpowiada NIE.

Fakt 1. Oznaczmy przez $\#_v(\phi)$ liczbę zmiennych w formule ϕ, przez $|\phi|$ długość formuły ϕ. Powyższy algorytm dla problemu SAT działa w czasie $O(2^{\#_v(\phi)} \cdot |\phi|)$

Uzasadnienie. W przypadku pesymistycznym (gdy formuła nie jest spełnialna) musimy sprawdzić wszystkie wartościowania. Każda zmienna logiczna może przyjąć dwie wartości, a więc wszystkich wartościowań jest $2^{\#_v(\phi)}$. Każde z nich sprawdzamy w czasie $O(|\phi|)$. Łączny czas wynosi więc $O(2^{\#_v(\phi)} \cdot |\phi|)$. □

Dla każdego zestawu danych dla problemu SAT (formuły ϕ), która jest **spełnialna** weźmy takie wartościowanie zmiennych z ϕ, żeby ϕ miała wartość True. To wartościowanie nazwiemy *świadectwem* dla formuły ϕ.

Przykład. Dla formuły $\phi = (\alpha \vee \neg\beta) \wedge (\beta \vee \gamma)$ możemy wziąć świadectwo $S(\phi) = \{\alpha \leftarrow True, \beta \leftarrow False, \gamma \leftarrow True\}$.

Zauważmy, że możemy skonstruować algorytm wielomianowy, który:

- dostaje na wejściu formułę ϕ oraz pewne wartościowanie zmiennych z ϕ (nazwijmy je s),
- **sprawdza**, czy s jest jest świadectwem dla ϕ.

Algorytm ten działa w czasie $O(|\phi|)$. Taki algorytm nazywamy *weryfikatorem*.

Klasa NP

Dla dowolnego problemu *weryfikator* to algorytm, który dostaje na wejściu:

- dane d,
- świadectwo s (być może fałszywe), że odpowiedź dla danych d powinna być TAK,

a następnie sprawdza czy s jest dobrym świadectwem dla danych d. Ponadto:

- świadectwo ma wielomianowy rozmiar w stosunku do rozmiaru danych, tzn. $|s| = O(|d|^c)$ dla pewnej stałej c niezależnej od danych.
- weryfikator działa w czasie wielomianowym.

Problemy, dla których można utworzyć taki weryfikator tworzą klasę **NP**.

Definicja (Definicja klasy NP). Niech R będzie problemem decyzyjnym. $R \in NP$, gdy istnieje dwuparametrowy wielomianowy algorytm W taki, że dla dowolnych danych d do problemu R rozwiązaniem problemu R dla d jest „TAK” wtedy i tylko wtedy gdy istnieje świadectwo $S(d)$, $|S(d)| = O(|d|^c)$ takie, że $W(d, S(d)) =$ TAK (c – stała niezależna od d).

Klasa NP, inaczej:

- Weryfikator dostaje dane i „uzasadnienie”, że odpowiedź powinna być TAK,
- Dla każdych danych dla których rozwiązaniem dla problemu R jest TAK, istnieje uzasadnienie, które weryfikator akceptuje
- Dla każdych danych dla których rozwiązaniem dla problemu R jest NIE, weryfikator zawsze odrzuca (nie daje się oszukać fałszywym uzasadnieniem)

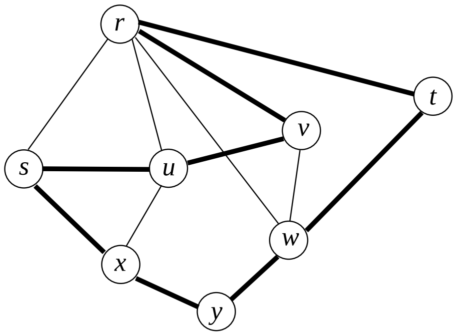
$\mathbf{P} \subseteq \mathbf{NP}$

Zauważmy, że dla dowolnego problemu z klasy **P** możemy wziąć **puste** świadectwa: weryfikator rozwiązuje problem w czasie wielomianowym stwierdzając, czy odpowiedź powinna być TAK (bez pomocy uzasadnienia). Dlatego:

$$\mathrm{P} \subseteq \mathrm{NP}$$
$$(\text{jeśli } R \in \mathbf{P}, \text{ to } R \in \mathbf{NP}).$$

Problem cyklu Hamiltona

Dla danego grafu G sprawdzić, czy istnieje cykl prosty zawierający wszystkie wierzchołki G (czy można obejść wszystkie wierzchołki i wrócić do początkowego tak, aby każdy wierzchołek odwiedzić dokładnie raz? Patrz Rys. 1.



Rysunek 1: Graf zawierający cykl Hamiltona (pogrubiony)

Fakt 2. Problem cyklu Hamiltona jest w **NP**.

Uzasadnienie. Jako świadectwo bierzemy kolejne wierzchołki tworzące cykl. Weryfikator łatwo sprawdza w czasie wielomianowym, czy podane wierzchołki są połączone krawędziami i czy każdy wierzchołek grafu wystąpił w ciągu dokładnie raz. □

Problem komiwojażera (TSP)

Dane:

- n miast ponumerowanych od 1 do n;
- $c(i, j)$ – koszt podróży z miasta i do miasta j;
- liczba rzeczywista k.

Problem: Komiwojażer mieszka w mieście 1. Chce odwiedzić wszystkie miasta i wrócić do miasta 1 tak, aby całkowity koszt podróży był $\leq k$. Czy jest to możliwe?

Fakt 3. TSP $\in$ **NP**.

Uzasadnienie. Świadectwem jest ciąg numerów miast na trasie komiwojażera. Weryfikator sprawdza, czy wszystkie miasta są odwiedzone, oblicza sumę kosztów przejazdów między kolejnymi miastami i porównuje ją z k. □

Sprowadzanie jednych problemów do drugich

Niech A – algorytm rozwiązujący problem komiwojażera. Mamy graf $G = (V, E)$, $V = \{1, 2, \dots n\}$ i chcemy rozwiązać problem cyklu Hamiltona. Ustalamy

$$c(i,j) = \begin{cases} 0 & \text{gdy } i\text{–}j \in E, \\ 1 & \text{w przeciwnym przypadku.} \end{cases}$$

Uruchamiamy A dla $k = 0$. Jeśli A odpowie TAK, to w G jest cykl Hamiltona, jeśli odpowie NIE, to nie ma.

Pokazaliśmy, że w czasie wielomianowym można *sprowadzić* problem cyklu Hamiltona do problemu TSP, czyli że problem cyklu Hamiltona „nie jest trudniejszy" niż problem TSP.

Mówimy, że problem R_1 można *sprowadzić* do problemu R_2, jeśli na podstawie dowolnych dancyh d_1 dla problemu R_1 możemy skonstruować takie dane d_2 do problemu R_2, że rozwiązaniem problemu R_1 dla danych d_1 jest TAK wtedy i tylko wtedy gdy rozwiązaniem problemu R_2 dla danych d_2 jest TAK.

NP-zupełność

Twierdzenie (Cooka). *Każdy problem z klasy **NP** da się sprowadzić do problemu SAT w czasie wielomianowym.*

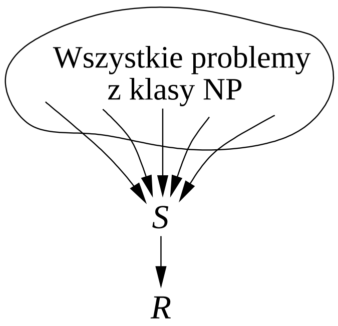
Wniosek 1. *Jeśli rozwiążemy SAT w czasie wielomianowym, to rozwiążemy też **wszystkie** problemy z **NP** w czasie wielomianowym.*

Definicja. Powiemy, że problem R jest **NP**-zupełny, jeśli

- $R \in$ **NP**,
- każdy problem z klasy **NP** można sprowadzić w czasie wielomianowym do R

Wiemy, że problem SAT jest **NP**-zupełny. Co zrobić, żeby pokazać, że jakiś problem R jest **NP**-zupełny? Udowodnić nowe twierdzenie Cooka?

Chcemy pokazać, że pewien problem $R \in$ **NP** jest **NP**-zupełny. Wystarczy znaleźć jakiś problem **NP**-zupełny S i pokazać, że S sprowadza się w czasie wielomianowym do R; np $S =$ SAT (patrz Rys 2). Wiadomo (ktoś udowodnił), że problem problem cyklu Hamiltona jest NP-zupełny. My pokazaliśmy, że problem TSP jest NP-zupełny (poprzez sprowadzenie problemu cyklu Hamiltona).



Rysunek 2: Dowodzenie, że problem R jest **NP**-zupełny: sprowadzenie **NP**-zupełnego problemu S do problemu R

Czy P=NP?

Intuicyjnie możemy sobie wyobrażać, że problemy NP-zupełne są najtrudniesze w klasie NP. Ale czy są trudniejsze niż problemy z klasy P? To znaczy, czy można je rozwiązać w czasie wielomianowym? Wydawałoby się, że nie można, bo chyba trudniej jest *rozwiązać* jakiś problem (np. sprawdzić czy istnieje cykl Hamiltona), niż *sprawdzić*, czy jakieś rozwiązanie jest dobre (np. czy dany ciąg wierzchołków tworzy cykl Hamiltona). A jednak wiadomo tylko, że:

- dla żadnego problemu NP-zupełnego nie znaleziono algorytmu wielomianowego
- gdyby rozwiązano jeden z nich w czasie wielomianowym, możnaby rozwiązać **wszystkie** (wtedy **P**=**NP**),
- nikt nie umie udowodnić, że jakiś problem **NP**-zupełny nie może być rozwiązany w czasie wielomianowym (!!!).

A więc **nie wiadomo**, czy choć jeden problem z klasy **NP** jest poza klasą **P** (bo nie wiadomo tego nawet dla najtrudniejszych problemów w klasie **NP**).

Ważny problem: Czy **P**=**NP**?