Zagadnienia złożoności obliczeniowej

http://zajecia.jakubw.pl/nai

ZŁOŻONOŚĆ - PRZYPOMNIENIE

- Złożoność czasowa
  - liczba „podstawowych operacji", jakie musi wykonać program komputerowy rozwiązujący zadanie
- Złożoność pamięciowa
  - liczba „podstawowych jednostek pamięci", które zajmuje program podczas pracy
- Złożoność jest funkcją wielkości problemu

Złożoność obliczeniową można szacować jako złożoność pesymistyczną (dla najgorszych możliwych danych), lub średnią (dla danych „losowych"). Tak rozumiana <u>złożoność jest cechą algorytmu</u>, a nie problemu; można jednak czasem udowodnić, że dany problem nie może zostać rozwiązany algorytmem o zbyt niskiej złożoności.

ZŁOŻONOŚĆ - PROSTE PRZYKŁADY

Sortowanie n obiektów:
• algorytm bąbelkowy - $O(n^2)$
• algorytm szybki - $O(n \log n)$
• sprawdzenie wszystkich możliwości - $O(n!)$

Wielkość problemu: n (liczba obiektów).

Sprawdzenie, czy liczba naturalna n jest pierwsza:
• algorytm dzielący n przez wszystkie liczby mniejsze od n - $O(2^k)$
• znaleziony niedawno algorytm ma złożoność $O(k^{12})$

Uwaga! Wielkość problemu to k = log n (liczba bitów), a nie n.

ZADANIA ŁATWE I TRUDNE

Zadania "łatwe"

- Sortowanie
- Szukanie pierwiastków wielomianów
- Szukanie maksimum funkcji ciągłej i różniczkowalnej
- Mnożenie macierzy
- Sprawdzenie, czy w grafie istnieje cykl Eulera
- ...

Znamy efektywne algorytmy dające dokładne rozwiązania.

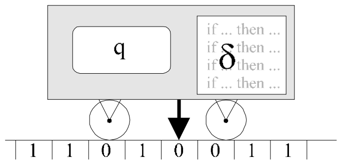
Zadania "trudne"

- Szukanie maksimum funkcji nieciągłej, nieróżniczkowalnej, zaszumionej, zmieniającej się w czasie
- Szukanie najkrótszej postaci danej formuły logicznej
- Rozkładanie liczb na czynniki pierwsze
- Sprawdzenie, czy w grafie istnieje cykl Hamiltona

Znane algorytmy dokładne mają wysoką (np. wykładniczą) złożoność czasową. Musimy szukać metod przybliżonych.

DETERMINISTYCZNA MASZYNA TURINGA (DTM)

Formalnie: $\{Q, \Sigma, \delta, q_0, F\}$, gdzie:



Q - zbiór ***stanów sterowania*** maszyny,
Σ - ***alfabet*** (zbiór symboli) taśmy,
δ – ***funkcja przejścia****:*

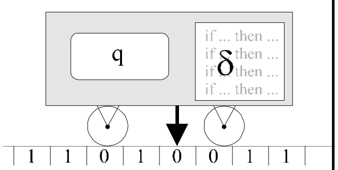
$$\delta: Q \times \Sigma \rightarrow Q \times \Sigma \times \{R, L, N\}$$

q_0 - ***początkowy stan*** sterowania,
F - zbiór ***końcowych stanów*** sterowania.

DTM - DZIAŁANIE

Działanie maszyny:
- Startujemy z pewnego miejsca na taśmie i ze stanu sterowania $\mathbf{q_0}$.
- Czytamy symbol **s** z taśmy.
- Na podstawie tych dwóch danych (stan **q** = $\mathbf{q_0}$, symbol **s**) za pomocą funkcji **δ** obliczamy:
  - nowy stan **q'**,
  - nowy symbol **s'**,
  - który zapisujemy na taśmie, oraz jeden z symboli: **R**, **L** lub **N**, odpowiadający kierunkowi przemieszczenia się czytnika na taśmie.
- Operację powtarzamy do momentu, gdy maszyna znajdzie się w stanie sterowania należącym do zbioru **F**.

$\{Q, \Sigma, \delta, q_0, F\}$

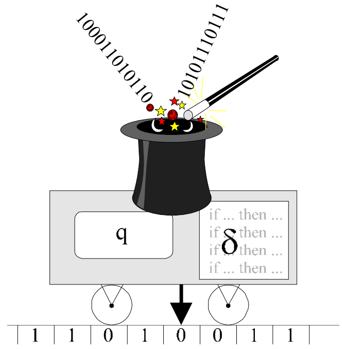


NIEDETERMINISTYCZNA MASZYNA TURINGA (NDTM)

$\{Q, \Sigma, \delta, q_0, F\}$,

Definicja jest analogiczna do DTM, jednak funkcja przejścia $\delta(q,s)$ może mieć kilka różnych wartości.

Wynik obliczeń jest pozytywny, jeśli choć jedna z możliwych dróg działania maszyny doprowadzi do sukcesu.



Innymi słowy: NDTM podczas wykonywania "programu" potrafi w magiczny sposób przewidzieć, jakiego dokonać wyboru (np. czy zapisać na taśmie 1, czy 0), by doprowadzić do pozytywnego wyniku (o ile jest to w ogóle możliwe).

KLASA *P* ORAZ *NP*

Maszyna Turinga jako ścisły model matematyczny umożliwia precyzyjne definiowanie pojęć związanych ze złożonością obliczeniową.
Czas działania = liczba kroków maszyny.

Problem należy do *klasy złożoności czasowej **P***, gdy istnieje **DTM** rozwiązująca ten problem **w czasie wielomianowym** względem rozmiaru danych wejściowych.

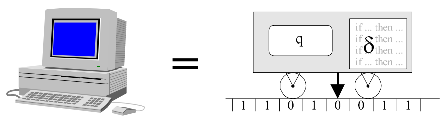
Problem należy do *klasy złożoności czasowej **NP***, gdy istnieje **NDTM** rozwiązująca ten problem **w czasie wielomianowym** względem rozmiaru danych wejściowych.

Intuicja: problem ma złożoność NP, jeśli znając rozwiązanie jesteśmy w stanie sprawdzić w czasie wielomianowym, czy jest ono poprawne.

KLASY P I NP - UWAGI

Uwaga 1: Ten sam problem można zakodować na różne sposoby - jeżeli kodowanie będzie "nieoszczędne", możemy uzyskać wielomianową szybkość działania, kosztem wykładniczej (w stosunku do optymalnej) reprezentacji.

Uwaga 2: (*Teza Churcha*) Możemy DTM uważać za model dowolnej klasycznej sekwencyjnej maszyny cyfrowej, więc w definicji klasy P możemy napis "DTM" zastąpić słowami "algorytm sekwencyjny".



Uwaga 3: Analogią NDTM w informatyce mógłby być język programowania ze specjalną funkcją, np. *forecast(),* zwracającą wartość 0 lub 1 (zawsze w ten sposób, "żeby było dobrze").

PROBLEMY NP-ZUPEŁNE

Problem P_0 jest ***NP-zupełny***, gdy:

a) P_0 należy do klasy NP,

b) każdy problem z klasy NP da się sprowadzić w czasie wielomianowym do problemu P_0.

Czyli np. znając rozwiązanie problemu P_0 w czasie wielomianowym na DTM, moglibyśmy w czasie wielomianowym rozwiązać każdy problem z klasy NP.
Czyli wówczas byłoby P = NP.

Problem ***NP-trudny*** spełnia tylko punkt b) powyższej definicji.

(Problemy NP-zupełne mają postać pytania "czy istnieje...", a problemy NP-trudne to zwykle ich optymalizacyjne wersje - "znajdź najmniejszy...")

SAT JEST NP-ZUPEŁNY

Sprawdzenie, czy formuła jest spełnialna (problem SAT), należy do klasy NP.

Zarys dowodu: używamy funkcji *forecast()*, by znaleźć wartościowanie spełniające formułę. W szybki (wielomianowy) sposób sprawdzamy, że rzeczywiście formuła jest spełniona.

Do problemu SAT da się sprowadzić dowolny problem z klasy NP.

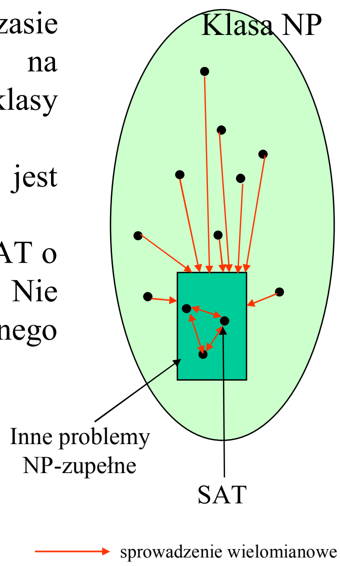
Zarys dowodu: każdą maszynę Turinga rozwiązującą konkretny problem z klasy NP (wraz z danymi wejściowymi) można opisać pewną skomplikowaną formułą logiczną, która jest spełnialna wtedy i tylko wtedy, gdy maszyna da wynik pozytywny. Zamiast konstruować maszynę, możemy więc znaleźć odpowiednią formułę i sprawdzić, czy jest spełnialna.

P=NP ?

- Istnieje problem “uniwersalny” (SAT), tzn. taki, że jego rozwiązanie w czasie wielomianowym pozwalałoby na rozwiązanie <u>wszystkich</u> problemów z klasy NP w czasie wielomianowym.
- Takich problemów NP-zupełnych jest więcej!
- Nie znamy algorytmu rozwiązującego SAT o złożoności mniejszej, niż wykładnicza. Nie znamy też takiego algorytmu dla żadnego innego problemu NP-zupełnego.

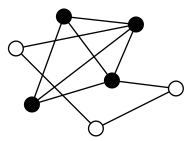
- Problem otwarty:

$\boxed{\text{Czy P = NP ?}}$



KLIKI W GRAFIE

Niech G = (V, E) - dany graf.
Kliką nazywamy zbiór wierzchołków grafu G połączonych "każdy z każdym".



Czy w danym grafie istnieje klika rzędu *k*?

Problem istnienia kliki jest NP-zupełny

Sprowadzimy 3-SAT do problemu kliki.
Każdy literał a_i kodujemy jako jeden wierzchołek w grafie. Wierzchołki łączymy krawędzią, jeśli odpowiednie dwa literały należą do różnych klauzul i nie są wzajemnie sprzeczne (tzn. nie łączymy zmiennej i jej zaprzeczenia).

Niech *k* - liczba klauzul. Wtedy klika rzędu *k* w tak skonstruowanym grafie odpowiada wartościowaniu spełniającemu formułę.