NP-zupełność

Kilka pojęć...

- „czas działania alg” vs „złożoność czasowa alg”

  Uwaga: złożoność to funkcja rozmiaru danych na we (instancji problemu) !!

  Co to jest rozmiar danych? Zależy od reprezentacji i sposobu kodowania...

  Np.: alg A jest „algorytmem grafowym” i działa w czasie O(n^5)

  Na we A mamy graf G reprezentowany przez macierz sąsiedztwa,

  która można zapisać na n^2 bitach;

  złożoność czasowa A = O((n^2)^(5/2)) = O(|G|^5/2)

- „złożoność problemu”:

  złożoność najlepszego alg, który go rozwiązuje

- „problem decyzyjny” vs „język formalny”

  Problem decyzyjny: „czy graf G ma cykl Hamiltona” (odp TAK/NIE)

  Język formalny HAM_CYCLE = {<G>: G zawiera cykl Hamiltona}

  gdzie <G> oznacza ciąg bitów reprezentujący graf G

- problemy decyzyjne:

  1) nierozstrzygalne, 2) wielomianowe, 3) nie-wielomianowe

  1) funkcja odp TAK/NIE jest nieobliczalna (patrz maszyny Turinga)

  2) i 3) funkcja jest obliczalna, chodzi o czas działania...

NP-zupełność

Kilka pojęć...

- „problem abstrakcyjny Q”:

  relacja na zb. instancji problemu (egzemplarzy) oraz rozwiązań;
  szersza definicja niż „problem decyzyjny” (gdzie rozw to TAK/NIE)
  dlaczego relacja a nie funkcja ?? (może być wiele rozw)
  Np. problem: „najkrótsza ścieżka w grafie G między wierz u i v”
    Instancja: <G, u, v>
    Rozwiązanie: ścieżka w grafie G (ciąg wierz)
  Np. problem: „największa klika w grafie G” (podgraf pełny)
    Instancja: <G>
    Rozwiązanie: zbiór wierz w G
  Np. wersja decyzyjna problemu największej kliki:
   „czy istnieje klika o rozmiarze =k”
    Instancja: <G, k>
    Rozwiązanie: TAK/NIE

Uwaga: 1 i 2 przykład to problemy optymalizacyjne;
  zawsze można dla „problemu opt” podać „problem decyzyjny”
    (patrz przykład z kliką ...)
  jeśli problem decyzyjny jest „trudny obliczeniowo” to opt też...
  w dalszej części wykładu ograniczamy się do problemów decyzyjnych

NP-zupełność

- problemy wielomianowe: klasa problemów **P**
- zmierzamy do zdef klasy problemów **NP** (ang. nondeterministic polynomial)
- **algorytm weryfikacji A(x,y)**

  Algorytm A(x,y) gdzie „x" to <u>instancja</u> problemu a „y" to <u>świadectwo.</u>

  A(x,y)=1 gdy rozwiązanie problemu jest na TAK

  Algorytm powinien działać w czasie wielomianowym wzg |x| i |y|

  (|x| oznacza liczbę bitów „x")

  Np. problem „czy graf ma cykl Hamiltona"

  A(x,y), gdzie x to reprezentacja grafu G, y to ciąg wierz

  A(x,y)=1 jeśli y jest permutacją wsz wierz i wszystkie sąsiednie pary to kraw G

- **def NP**: jest to klasa problemów decyzyjnych, które:
  1. posiadają algorytm weryfikacji A(), działający w czasie wielomianowym
  2. dla każdej instancji problemu istnieje świadectwo (2 arg A())

     o rozmiarze wielomianowo większym niż rozmiar instancji
- **def NP** (nieco bardziej szczegółowa)

  Język formalny L należy do klasy NP jeśli istnieje A(,) i stała c, takie, że:

  L = {x є {0,1}^* : istnieje świadectwo y, |y|=O(|x|^c), że A(x,y)=1)}

NP-zupełność

- zmierzamy do zdef klasy problemów **NP**-zupełnych
- *dlaczego to ważne ????*
  jeśli pewien problem **NP-zupełny** ma rozwiązanie wielomianowe
  to wszystkie problemy klasy NP mają rozw wielomianowe (czyli P=NP)
  Ale jest to mało prawdopodobne...
    jeśli udowodnimy, że nasz problem jest NP-zupełny
      to nie ma sensu szukać algorytmu wielomianowego..
- **redukowalność** w czasie wielomianowym: **L_1 <P= L_2**
  „język L_1 redukuje się wielomianowo do języka L_2”

def formalna: L_1 <P= L_2

jeśli istnieje funkcja obliczalna wielomianowo f(), f: L_1 -> L2
taka, że x ϵ L_1 <=> f(x) ϵ L_2

- **def języka NP-zupełnego L** (zawierającego słowa {0,1}^*)
  1. L ϵ NP
  2. dla każdego L**'** ϵ NP: L**'** <P= L
- **def języka NP-trudnego L**
  dla każdego L**'** ϵ NP: L**'** <P= L

Uwaga: słowo „język” można zastąpić słowem „problem decyzyjny”

NP-zupełność

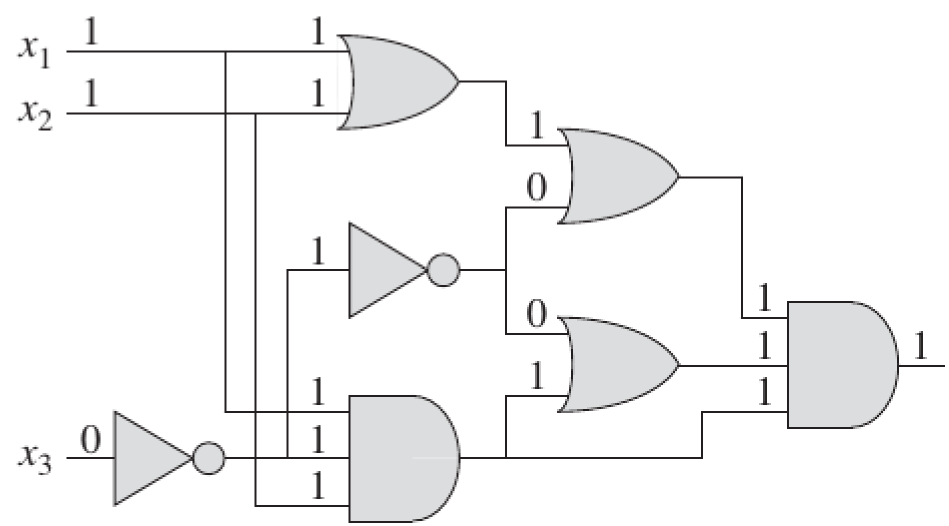
Kilka tw o NP-zupełności...

Twierdzenie 36.4 [Cormen, str 1039]
Jeśli jakikolwiek problem NP-zupełny jest rozwiązywalny w czasie wielomianowym, to P=NP.
Jeśli jakikolwiek problem w NP nie jest rozwiązywalny w czasie wielomianowym, to żaden problem NP-zupełny nie jest rozwiązywalny w czasie wielomianowym.

Lemat 36.8 [Cormen, str 1046]
Jeśli L jest językiem takim, że L' <P=L dla pewnogo NP-zupełnego L',
to L jest NP-trudny. Jeśli, dodatkowo, L jest w NP to L jest NP-zupełny.

Można udowodnić, że następujące problemy są NP-zupełne:

CIRCUIT-SAT =
problem spełnialności układów logicznych
Czy istnieją wartości 0/1 na we,
takie że na wy mamy 1.



NP-zupełność

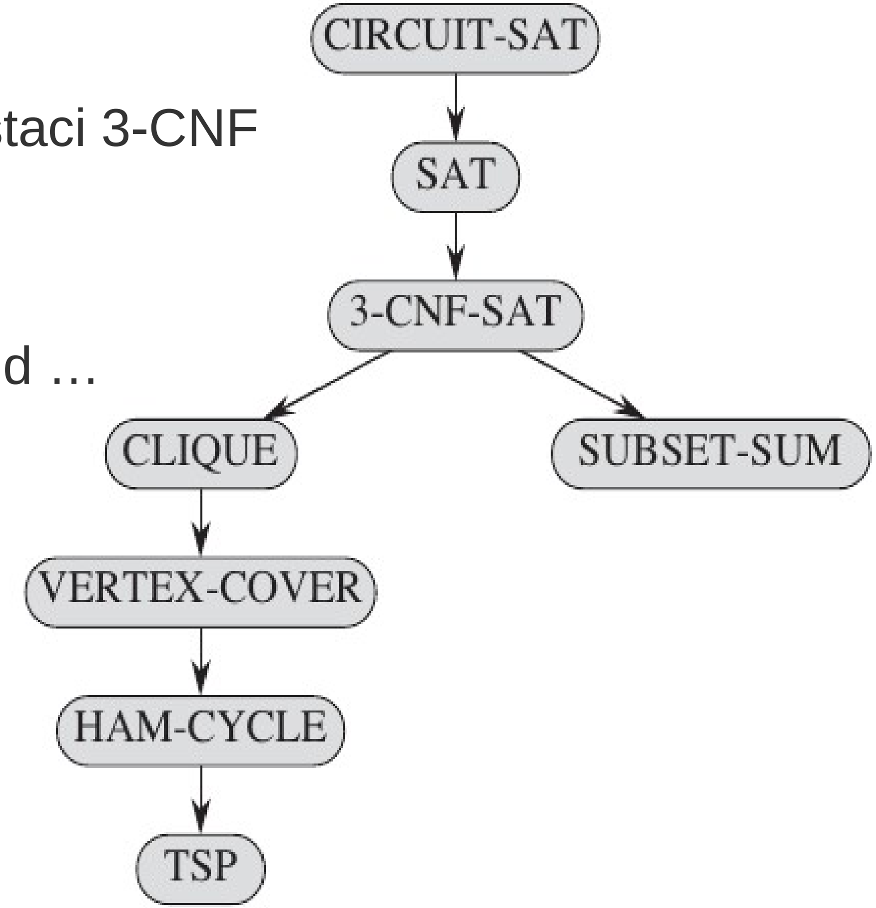
Można udowodnić, że następujące problemy są NP-zupełne (c.d.)

SAT =
problem spełnialności formuł logicznych:
czy istnieją wartości zmiennych x_i ze zbioru {0,1} takie że formuła zwraca 1
przykład formuły: (x1 and x2 or not x3) imp (x3 lub x4)
spójniki logiczne: and, or, imp (implikuje) i inne...

3-CNF-SAT =
problem spełnialności formuł logicznych w postaci 3-CNF
CNF = postać koniunkcyjna normalna
3-CNF składa się z klauzul,
które składają się z 3 literałów (np. x1, not x1, x2)
(x1 lub not x1 lub x2) and (x2 lub x3 lub x4) and …

CLIQUE =
czy istnieje klika o rozmiarze =k

Pozostałe problemy rozwiązuje się
poprzez redukcję wielomianową
z 3-CNF-SAT do problemu...

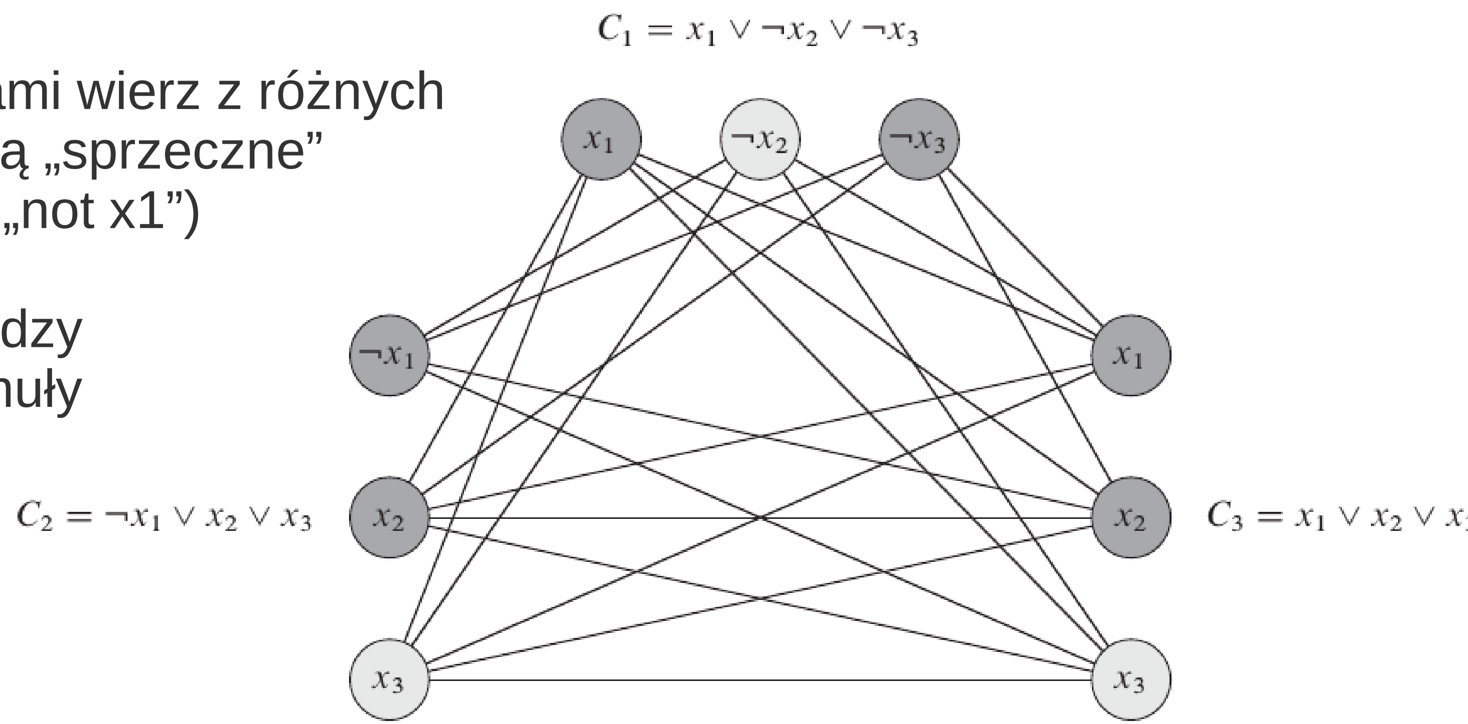


NP-zupełność

Tw 36.11 [Cormen, str 1055]
problem kliki (CLIQUE) jest NP-zupełny
Dowód:
Najpierw pokazuje się, że CLUQUE należy do NP (świadkiem jest podzbiór wierz, trzeba sprawdzić, czy między każdą parą jest kraw w grafie).
Potem pokazujemy, że 3-CNF-SAT <P= CLUQUE;
W tym celu dla formuły 3-cnf-sat „C1 and C2 and C3” budujemy graf:

łączymy krawędziami wierz z różnych klauzul, które nie są „sprzeczne” (<u>nie</u> łączymy „x1” i „not x1”)

widać związek między spełnialnością formuły a istnieniem kliki...



NP-zupełność

VERTEX-COVER = pokrycie wierzchołkowe (minimalne)

Jest to zbiór wierz V' taki, że każda kraw ma 1 lub 2 końce w V'

„czy istnieje VERTEX-COVER o mocy =k” (wersja decyzyjna)

Tw 36.12 [Cormen, str 1058]

Problem VERTEX-COVER jest NP-zupełny

Dowód:

Instancja: <G,k>, świadectwo: zbiór wierz

Oczywiście problem należy do NP.

Pokażemy, że CLIQUE <P= VERTEX-COVER...

Niech D(G) oznacza dopełnienie grafu G

({u,v} in D(G) <=> {u,v} not in G)

Zauważmy, że „G zawiera klikę o rozmiarze =k”

<=> „D(G) zawiera pokrycie wierz o rozmiarze |V|-k”