12 NP-úplnost

v textu jsou použity podklady Ing. A. Netrvalové

třída problémů *P* (ne algoritmů)

Třída problémů složitosti *P* - množina konkrétních problémů řešitelných v polynomiálním čase (existuje pro ně polynomiální algoritmus)

třída problémů *NP*

Problémy, pro které existuje polynomiální verifikační algoritmus *A(instance problému, certifikát),* tvoří třídu složitosti NP – nedeterministicky polynomiální

Ian Stewart:

http://www.claymath.org/Popular_Lectures/Minesweeper/

„My favourite example here is a jigsaw puzzle.“

http://justjigsawpuzzles.com/

Je-li nějaký problém ve třídě ***P*****,** verifikační algoritmus ***A(instance problému, certifikát)*** ignoruje certifikát a jeho výstup je dán polynomiálním algoritmem řešení pro instatnci problému

$$P \subseteq NP$$

http://www.claymath.org/millennium/

problém X těžší než problém Y

umíme-li řešit X potom umíme řešit Y

třída problémů NP-úplné

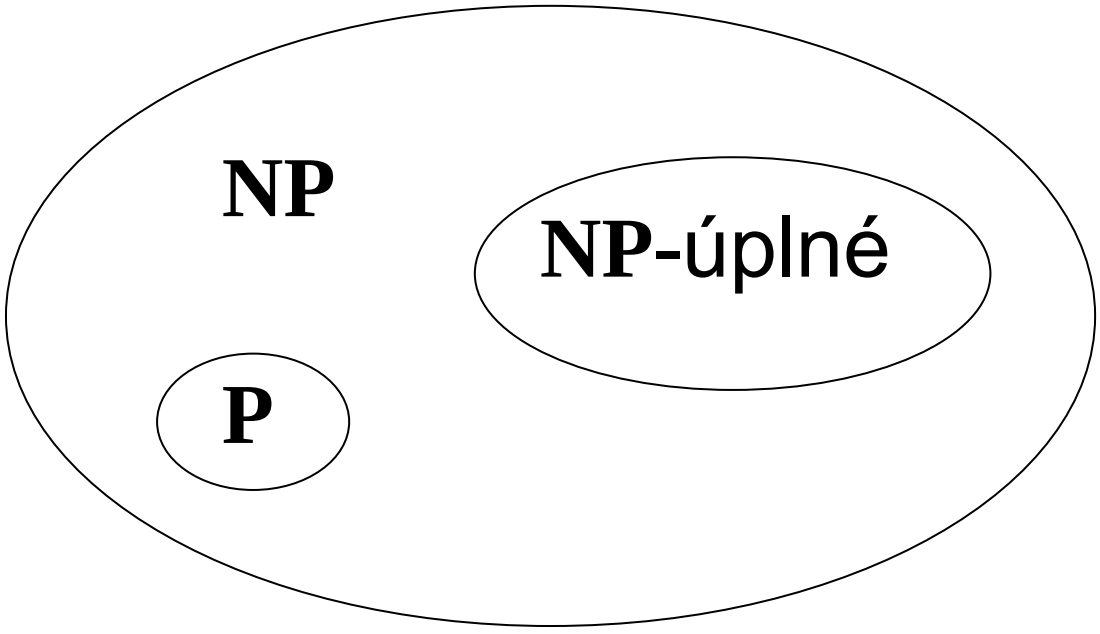
problémy třídy NP aspoň tak těžké jako jakýkoliv problém v NP (včetně ostatních NP-úplných)

může-li být některý NP-úplný problém vyřešen v polynomiálním čase, potom může být v polynomiálním čase vyřešen každý **NP** problém

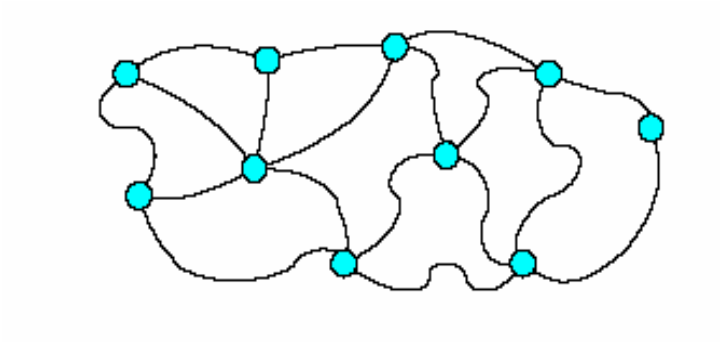
P = NP

pro žádný z asi 1000 známých NP-úplných problémů nikdo nenašel polynomiální algoritmus, čehož důsledkem by bylo **P = NP**

nikdo nedokázal neexistenci časově polynomiálního algoritmu pro některý z nich, což by znamenalo **P ≠ NP**



⁉ Někdy **dva problémy** mohou vypadat podobně, a přitom jeden z nich je řešitelný v polynomiálním čase, zatímco druhý je NP- úplný.

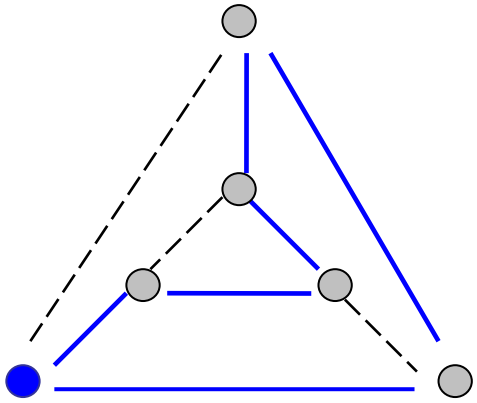


Problém 1 [P]
Vstup: Popis měst a silnic.
Výstup: Existuje okružní cesta, na které projedeme každou silnici právě jednou?

Problém 2 [NP- úplný]
Vstup: Popis měst a silnic.
Výstup: Existuje okružní cesta, na které projedeme každé město právě jednou?

Eulerovský graf. Určete, jestli v neorientovaném grafu *G(V, H)* existuje uzavřený tah (hrany se neopakují) obsahující všechny hrany, tj. mající délku $|H|$.

Hamiltonovský graf. Určete, jestli v neorientovaném grafu *G(V, H)* existuje uzavřená cesta (vrcholy se neopakují, kromě prvního a posledního) – kružnice délky $|V|$, tj. procházející všemi vrcholy.



Eulerův tah umíte najít, pokud existuje, v čase ***O(H)***. Umíte tedy i rozhodovací problém řešit v polynomiálním (lineárním) čase. Patří tedy do třídy ***P***.

Hamiltonovská cesta je ***NP***.
Certifikát - permutace vrcholů $v_1, v_2, ..., v_{|V|}$ tvořící hamiltonovskou kružnici.
Ověřovací algoritmus:
Prověříme, je-li certifikát permutací vrcholů grafu a je-li $(v_i, v_{i+1}) \in H$, $i = 1,2, ..., |V|-1$ a $(v_{|V|}, v_1) \in H$, což je možné v polynomiálním čase - problém patří do třídy NP.

Algoritmus (řešení): prověřováním všech permutací
Zakódování maticí sousednosti - počet vrcholů m je $\Omega(\sqrt{n})$, $n = |G|$
Čas výpočtu - $\Omega(m!) = \Omega(\sqrt{n}!)$, není $O(n^k)$ pro žádnou konstantu k.

Hamiltonovská cesta je ve skutečnosti *NP* – úplný problém.

Převoditelnost problému – *formalizace vztahu těžší*

- uvažujme dva problémy *S* a *T*
- umíme-li každou instanci *x* problému *S* převést na instanci *f(x)* problému *T* tak, že

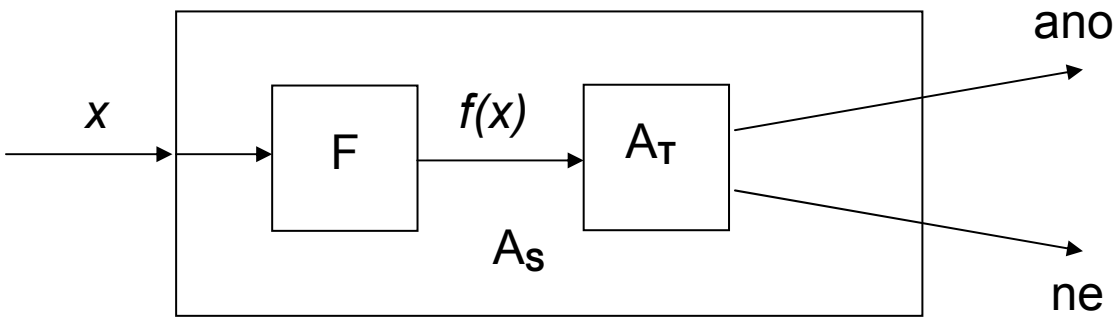
řešení problému *S* pro *x* je stejné jako řešení problému *T* pro *f(x)*

problém *S* je převeditelný na problém *T*
problém *S* není těžší než *T*

S – rovnice $a.x + b = 0$, instance [a,b]
T – rovnice $a.x^2 + b.x + c = 0$, instance [a,b,c]

f([a,b]*)* = [0,a,b]

S, T – rozhodovací problémy
A_S, A_T – algoritmy řešení
F – algoritmus pro výpočet *f(x)*



Je-li algoritmus *F* pro výpočet funkce *f* časově polynomiální, problém *S* je polynomiálně převeditelný na problém *T*, $S \leq_P T$.

Polynomiální převoditelnost problémů nám umožňuje dokázat, že problémy patří do P.

Tvrzení 1
Jsou-li S a T problémy takové, že $S \leq_P T$, potom $T \in$ P implikuje $S \in$ P.

Důkaz. Konstrukce časově polynomiálního algoritmu pro problém S pomocí polynomiálního algoritmu F a polynomiálního algoritmu A_T, ($T \in$ P), je na předcházejícím obrázku.

NP-úplnost

NP - úplné problémy jsou nejtěžší problémy v NP.

Problém T je **NP-úplný**, když

1. $T \in$NP a
2. $S \leq_P T$ pro každý problém $S \in$NP.

Jestliže problém splňuje vlastnost 2, ale ne nevyhnutně vlastnost 1, je NP-těžký.

NP-úplnost je klíčová pro P = NP.

Tvrzení 2
Je-li T NP-úplný a $T \in$ P, potom P = NP.

Důkaz. Pro každý problém $S \in$ NP, podle druhé vlastnosti NP-úplných problémů, máme $S \leq_P T$. Protože $T \in$ P, z tvrzení 1 plyne, že každý $S \in$ P a tedy P = NP .

Tvrzení 2 ekvivalentně:

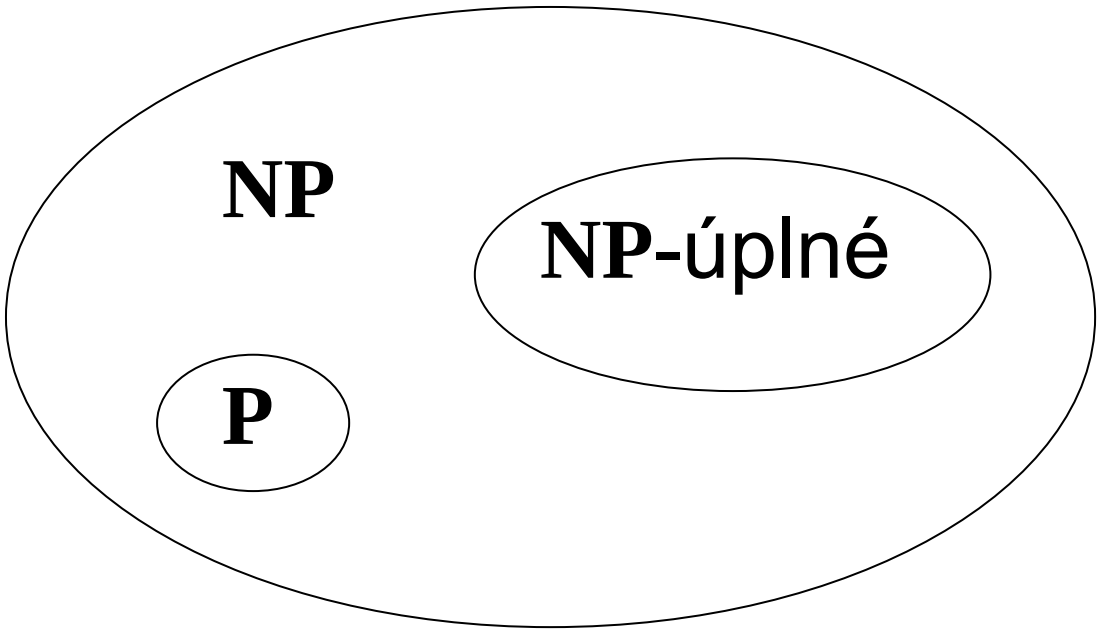
Existuje-li v NP problém, který není řešitelný časově polynomiálně, potom žádný NP-úplný problém není řešitelný v čase polynomiálně.

Důkaz. Tvrzení 2 můžeme verbálně přeformulovat takto:

Existuje-li NP-úplný problém, který je časově polynomiálně řešitelný, potom každý NP problém je časově polynomiálně řešitelný.

Použitím tautologie $(A \Rightarrow B) \Rightarrow (\neg B \Rightarrow \neg A)$ na přeformulované tvrzení 2 dostáváme přímo jeho ekvivalentní vyjádření.

- řešení stovek NP-úplných problémů je věnováno veliké úsilí
- pro žádný z nich nebyla dokázána existence časově polynomiálního algoritmu
- převládá názor, že $P \cap$ NP-úplné $= \varnothing$ a $P \neq NP$

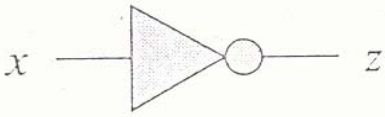


Existuje NP- úplný problém ?

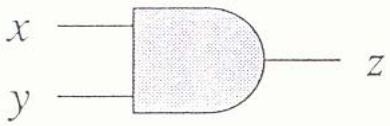
Splnitelnost logického obvodu

(podle Cormen et al: Introduction to Algorithms)

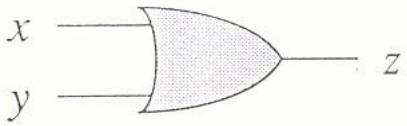
- kombinační logický obvod sestává z logických hradel, která realizují jednoduché logické funkce
- základní hradla realizují negaci, logický součin a logický součet



x	$\neg x$
0	1
1	0



x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1



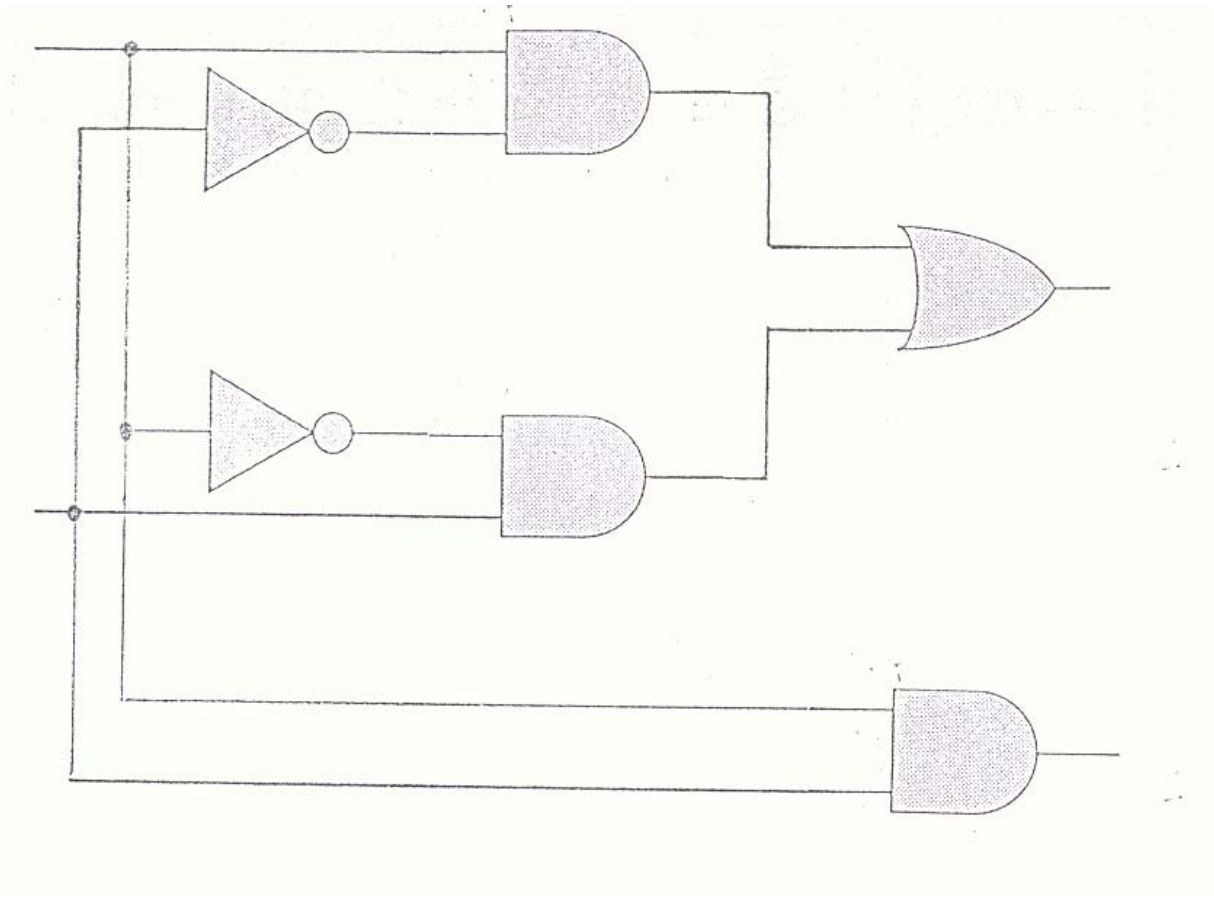
x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Logický kombinační obvod - propojení výstupů logických hradel na vstupy jiných.
Vstupy obvodu - vstupy hradel, kterým není připojen výstup žádného hradla.
Výstupy obvodu - výstupy hradel, které nejsou připojeny na žádný vstup.

Logické kombinační obvody jsou základem počítačového hardwaru vykonávajícího instrukce.

Příklad

Součet dvou jednobitových hodnot realizuje obvod:

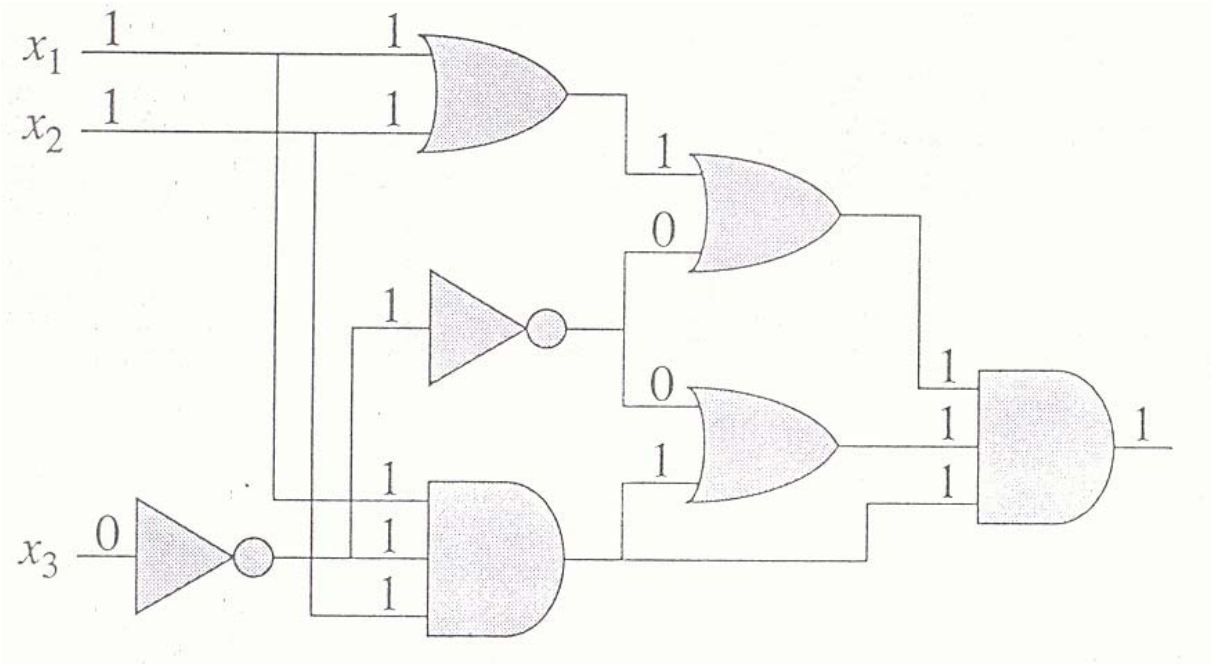


Vstupy: dva operandy součtu
Horní výstup: součet
Spodní výstup: přenos

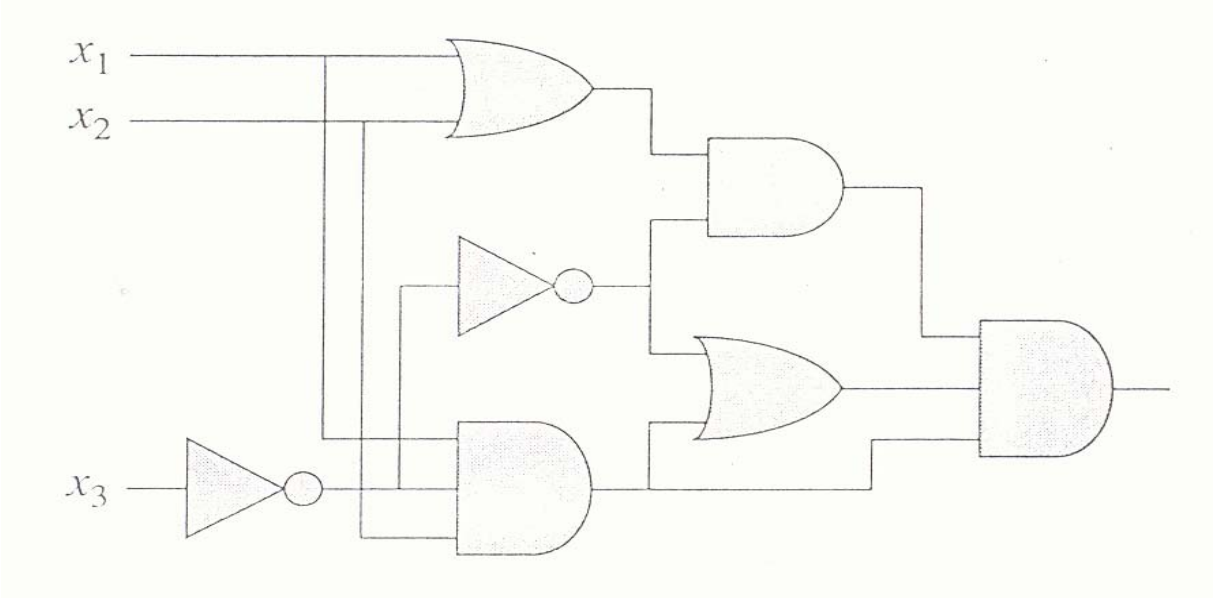
Pravdivostní přiřazení je množina logických vstupních hodnot.

Obvod s jedním výstupem je splnitelný, má-li pravdivostní přiřazení, pro které je jeho výstup 1.

Příklad splnitelného obvodu



Příklad nesplnitelného obvodu



Problém splnitelnosti obvodu

Je zadaný logický kombinační obvod, složený z hradel pro negaci, logický součin a logický součet, splnitelný?

Velikost problému je dána počtem hradel a jejich propojení. Podobně jako v případě grafu je jeho zakódování polynomiální.

Dokážeme, že problém splnitelnosti obvodu splňuje obě podmínky pro NP-úplný problém.

Tvrzení 3
Problém splnitelnosti obvodu patří do třídy NP.

Důkaz:
Problém splnitelnosti obvodu patří do třídy NP existuje-li časově polynomiální ověřovací algoritmus *A*, který pro instanci problému - logický kombinační obvod *C* a certifikát – pravdivostní přiřazení, ověří jeho splnitelnost.

Ověřovací algoritmus *A*:
Pro jednotlivé vstupní hodnoty pravdivostního přiřazení, vypočte výstup všech hradel.
Je-li výstup obvodu 1, vstupní hodnoty jsou splňující pravdivostní přiřazení a algoritmus *A* vrátí 1 jinak *A* vrátí 0.

Problém splnitelnosti obvodu patří do třídy NP.

Tvrzení 4
Problém splnitelnosti je NP- těžký.

Máme dokázat: Každý (rozhodovací) problém $S \in NP$ je polynomiálně převoditelný na problém splnitelnosti obvodu.

T - splnitelnosti obvodu
$S \in NP$

Úkol: Opsat časově polynomiální algoritmus *F,* který každou instanci *x* problému *S* zobrazí na obvod *C = f(x)* tak, aby řešení problému *S* pro *x* bylo stejné jako řešení problému splnitelnosti obvodu pro *C*.

Konstrukce obvodu *C*:
$S \in NP$, existuje pro něj časově polynomiální ověřovací algoritmus *A(x,y)*, kde *x* je instance problému *S* a *y* certifikát.

Řešení instance *x* rozhodovacího problému *S* je *A(x,y).*

HW počítače – logický obvod
vstupy – hodnoty v registrech a paměti počítače (konfigurace)
instrukce – změna konfigurace

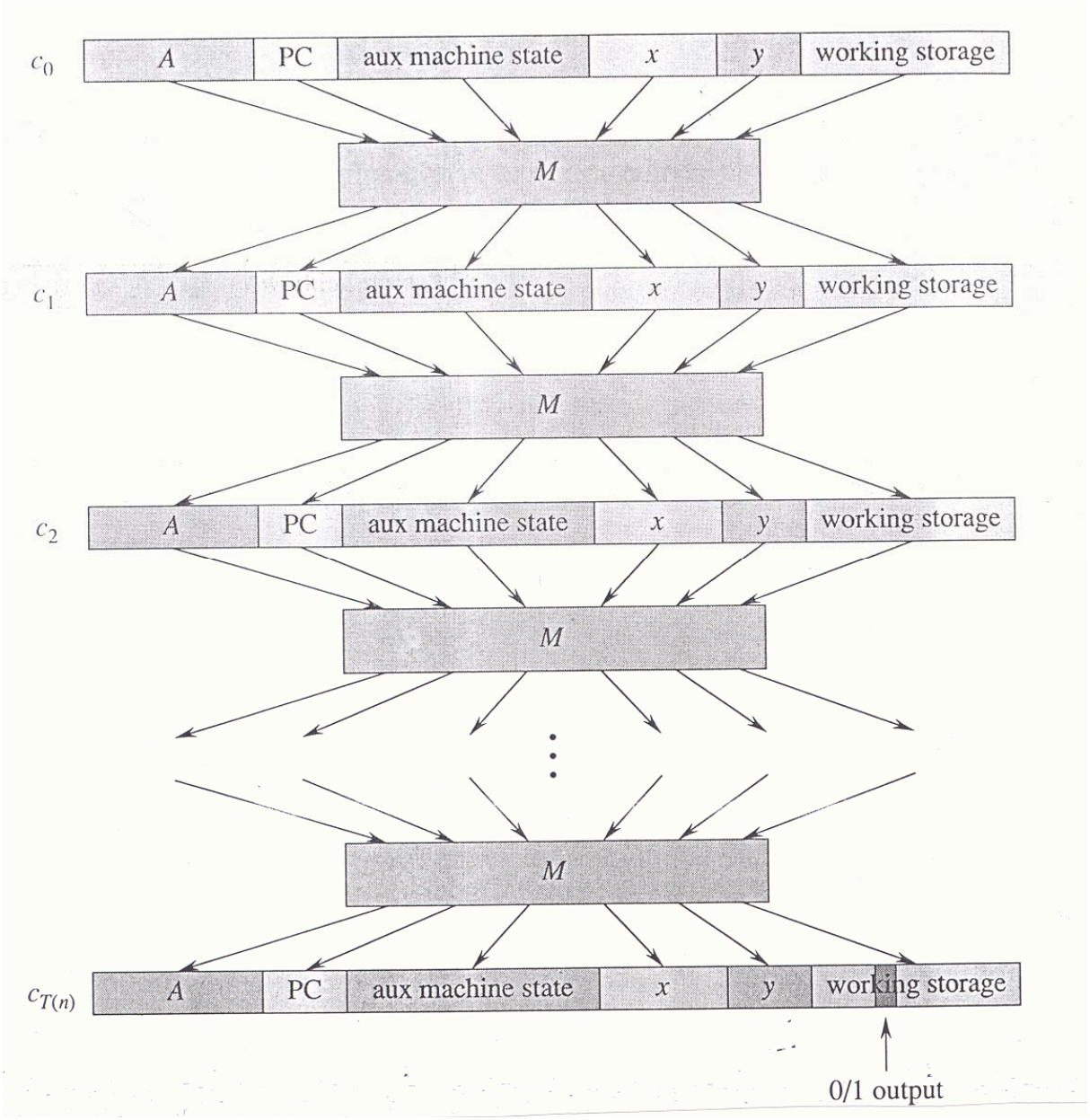
M - logický obvod (HW), který implementuje zobrazení jedné konfigurace na následující

A potřebuje v nejhorším případě vykonat $T(n) = O(n^k)$ instrukcí

F pro *x* vytvoří obvod *C* jako sekvenci *T(n)* logických obvodů *M*

počáteční konfigurace c_0 obsahuje program pro A, instanci problému *x* a certifikát *y*.

poslední instance $c_{T(n)}$ někde v pracovní paměti obsahuje výstup algoritmu *A* – 0 nebo 1.



Algoritmus *F* je časově polynomiální.

Obvod *C* počítá *C(y)* = *A(x,y)* pro libovolný $S \in$ NP.

Nechť existuje certifikát *y*, pro který *A(x,y)* =1. Jestli bity *y* budou vstupy *C*, potom *C(y)* = *A(x,y)* = 1 a *C* je splnitelný.

Je-li *C* splnitelný, potom existuje vstup *y*, pro který *C(y)* = 1 a tedy i *A(x, y)* = 1.

Důkaz NP- úplnosti

Tvrzení 5
Je-li problém T takový, že pro některý NP-úplný problém S je $S \leq_P T$, potom problém T je NP-těžký. Je-li k tomu $T \in$ NP, potom T je NP-úplný.

Důkaz. Pro všechny $S' \in$ NP je $S' \leq_P S$. Z předpokladu $S \leq_P T$ tedy je
i $S' \leq_P T$ a T je NP-těžký.

Převedením známého NP-úplného problému S na T, jsme implicitně převedli každý NP-problém na T.

Metoda důkazu, že problém T je NP-úplný.

1. Dokažte, že $T \in$ NP.

2. Vyberte známý NP-úplný problém S.

3. Opište algoritmus pro výpočet funkce f zobrazující, každou instanci x problému S na instanci $f(x)$ problému T.

4. Dokažte, že řešení problému S pro instanci x a řešení problému T pro $f(x)$ jsou stejné.

5. Dokažte, že algoritmus počítající f je časově polynomiální.

Příklad

Dokažme, že problém obchodního cestujícího (TSP) je NP-úplný, je-li známo, že problém existence hamiltonovské kružnice (HK) je NP-úplný.

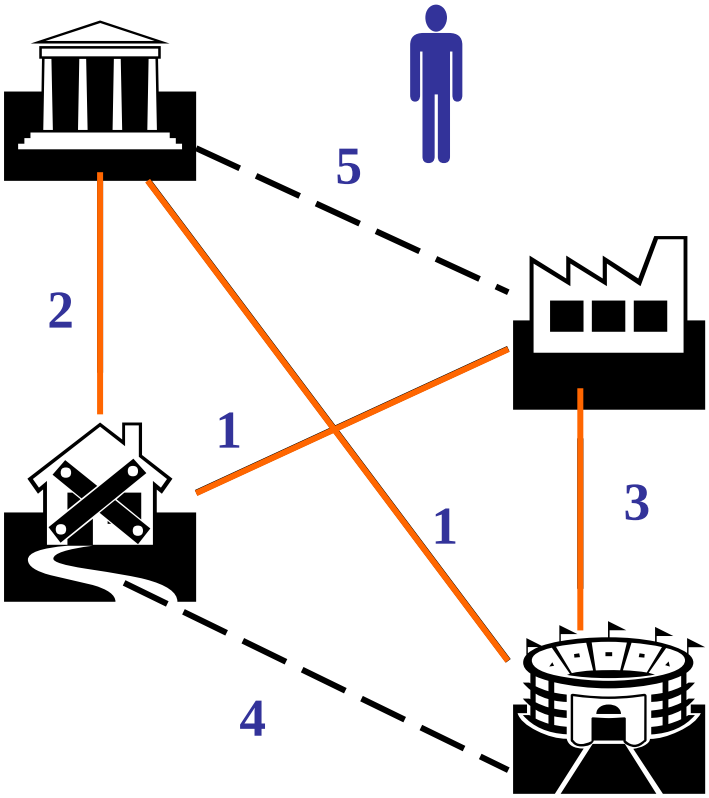
Problém obchodního cestujícího v jeho rozhodovací verzi má následující tvar.

-$G = (V, H)$ je úplný graf

-náklady na cestu z vrcholu i do vrcholu j jsou dány funkcí $c: V \times V \rightarrow R$

-existuje v G cesta obchodního cestujícího, který navštíví každý vrchol právě jednou, skončí ve výchozím vrcholu a náklady jsou nejvíce $k \in R$?

Instance problému obchodního cestujícího je tedy trojice (G, c, k).



Důkaz:

1. Je-li zadána instance problému a certifikát, verifikační algoritmus ověří, že certifikát obsahuje každý vrchol právě jednou, sečte náklady a ověří jsou-li nejvíce *k.*

TSP patří do NP.

2. HK $\leq_P$ TSP

3. *G = (V, H)* je instance HK. Instanci TSP sestrojíme následovně.

Vytvoříme úplný graf
G´ = (V, H´), H´={(i,j):i,j∈H a *i≠j}* a definujeme nákladovou funkci

c(i,j) = 0 je-li *(i,j)* $\in H$
= 1 je-li *(i,j)* $\notin H$

Instance TSP je (*G´, c*, 0)

4. Graf *G* má hamiltonovskou kružnici právě tehdy má-li graf *G´*cestu s náklady nejvíce 0.

Nechť *G* má hamiltonovskou kružnici *h*.

Každá hrana v *h* je prvkem *H*, má tedy v *G´* náklady 0 a *h* je cesta v *G´*s náklady 0.

Opačně, nechť graf *G´* má cestu *h´* s náklady nejvíce 0. Potom každá hrana v *h´* má náklady 0 a *h´* obsahuje pouze hrany z *H*, tedy *h´* je hamiltonovskou kružnicí v grafu G.

5. Instanci TSP - (*G´, c*, 0) vytvoříme v polynomiálním čase.

Zjistíme-li, že řešený problém je NP-úplný, je hledání časově polynomiálního algoritmu pravděpodobně mrháním času.

Je možno sáhnout k jiným přístupům:

aproximační
pravděpodobnostní
řešení speciálních případů
heuristické

Jedním z NP-úplných problémů je tzv. 15 rébus.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Nalezení nejkratší posloupnosti tahů, pomocí kterých ze zadané permutace (pokud je to možné), získáme uvedené uspořádání je NP-úplný problém.

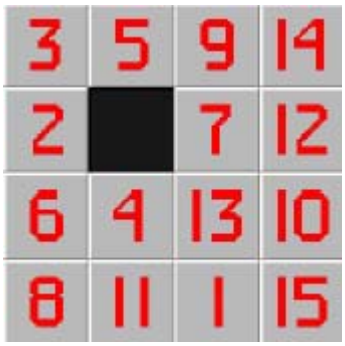
Heuristické řešení se snaží z možných tahů vybrat nejslibnější. Může tak udělat ohodnocením „vzdálenosti“ nové konfigurace vzniklé tahem od cílové.

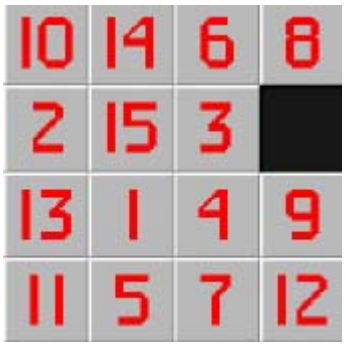
Označme souřadnice každého políčka i = 1, 2, …, 15 v cílové konfiguraci cx_i a cy_i. Například $cx_7 = 3$ a $cy_7 = 2$. Pokud v nějaké konfiguraci k má i souřadnice kx_i a ky_i, potom její vzdálenost od pozice v cílovém stavu můžeme vyjádřit vztahem

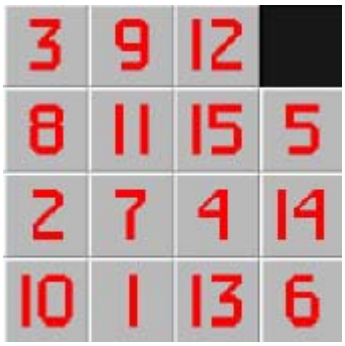
$$(cx_i - kx_i)^2 + (cy_i - ky_i)^2$$

Vzdálenost konfigurace od cílové potom můžeme vyjádřit jejich součtem pro všechna i. Cílového stavu dosáhneme bude-li tento součet nulový.

Heuristický algoritmus potom principiálně pracuje tak, že pro možné tahy na prázdné políčko (2 až 4 možnosti) vybere tu novou konfiguraci, pro kterou je uvedený součet nejmenší, atd.







Základní postavení a varianty se 4, 3 a 2 možnostmi dalšího postupu