

NP úplnosť: osnova

- Rozhodovacie vs. optimalizačné problémy.
- Trieda problémov P.
- Nedeterministické výpočty a trieda problémov NP.
- Polynomiálne transformácie a NP-úplnosť.
- Cookova veta: Existuje NP-úplný problém.
- Ako ukázať, že problém je NP-úplný?
- Základné “portfólio” NP-úplných problémov.

Nedeterministický algoritmus pre riešenie TSP-D

```
function TSP-D
  visited[i]:=false for all vertices;
  last_visited:=1; visited[1]:=true;
  length:=0;
  repeat n-1 times
    choose next_visited between 1 and n;
    if visited[next_visited] then reject;
    //we cannot visit a single vertex twice
    visited[next_visited]:=true;
    length:=length+w(last_visited,next_visited);
    last_visited:=next_visited;
  length:=length+w(last_visited,1);
  if length<=B then accept;
  else reject;
```

Def: SAT (splniteľnosť) Uvažujme booleovské premenné $(u_1, \dots, u_m)$ a logickú formulu f .

Problém: Existuje priradenie hodnôt premenných také, aby f bola splnená?

Veta: (Cook) SAT je NP-úplný

Náčrt dôkazu: Potrebujeme dokázať:

- 1 SAT $\in$ NP –alebo–
Existuje nedeterministický polynomiálny algoritmus, ktorý rieši SAT.
- 2 SAT je NP-ťažký –alebo–
pre ľubovoľný problém $Q \in NP$: $Q \leq_p \text{SAT}$

1 SAT $\in$ NP

```
for i:=1 to m do
    choose u[i] between 0 and 1; // 0 means false,
                                // 1 means true

evaluate formula f with assignment
(u[1],u[2],...,u[m])

if f is satisfied then ACCEPT
    else REJECT
```

2 SAT je NP-ťažký

Uvažujme $Q \in \text{NP}$

$\implies$ existuje polynomiálny nedeterministický algoritmus, ktorý rieši Q

Ako taký algoritmus zapíšeme?

- Každý register má v sebe uložené číslo konštantnej veľkosti (registre označíme $R_1, R_2, \dots$)
- Program je nemiata sa postupnosť príkazov s konštantným počtom očíslovaných riadkov
- Na začiatku je vstup uložený v prvých n registroch (n je veľkosť vstupu)
- Program beží nanajvýš $p(n)$ krokov a pristupuje najviac ku $q(n)$ prvým registrom ($p(n)$ a $q(n)$ sú polynómy závisiace od n)

- Sada inštrukcií:
 - ACCEPT
 - REJECT
 - GOTO m
 - IF $R_\ell = 0$ THEN GOTO m
 - CHOOSE R_ℓ BETWEEN 0 AND 1
 - základné aritmetické operácie
(napr. $R_\ell := R_u + R_v$, $R_\ell := R_u * R_v$)
 - nejaký mechanizmus na adresáciu prvých $q(n)$ registrov
(details sú mierne komplikované, ale dá sa)

2 SAT je NP-ťažký: $Q \leq_p \text{SAT}$

ňbf Chceme:

- Daný je program A , ktorý rieši problém Q v polynomiálnom čase a inštancia $x = x_1, x_2, \dots, x_n$.
- Vyrobitíme veľkú logickú formulu f , ktorá “simuluje” program A na vstupe x ;
- A dosiahne ACCEPT $\iff f$ je splniteľná

Premenné formuly f :

- $Q[i, k]$ – v čase i program vykonáva riadok k
- $S[i, j, k]$ – v čase i má register R_j hodnotu k

Formula f bude konjunkcia (“AND”) niekoľkých menších formúl t.j. všetky tieto menšie formuly musia byť splnené, aby formula f bola splnená

1 “V každom čase i program vykonáva práve jeden riadok.”

$$\boxed{\neg(Q[i, k] \wedge Q[i, \ell])} \quad \text{pre všetky } i \text{ a } k \neq \ell$$

2 “V každom čase i každý register obsahuje práve jednu hodnotu.”

$$\boxed{\neg(S[i, j, k] \wedge S[i, j, \ell])} \quad \text{pre všetky } i, j \text{ a } k \neq \ell$$

3 V čase 0:

- Program vykonáva riadok 1: $\boxed{Q[0, 1]}$
- Prvých n registrov má hodnoty $x_1, \dots, x_n$:

$$\boxed{S[0, 1, x_1] \wedge S[0, 2, x_2] \wedge \dots \wedge S[0, n, x_n]}$$

- Ostatné registre majú hodnotu 0:

$$\boxed{S[0, n+1, 0] \wedge S[0, n+2, 0] \wedge \dots \wedge S[0, q(n), 0]}$$

4 “Po $p(n)$ krokoch program dosiahne riadok s inštrukciou ACCEPT”

$$\boxed{Q[p(n), k]} \quad k \text{ je riadok s inštrukciou “ACCEPT”}$$

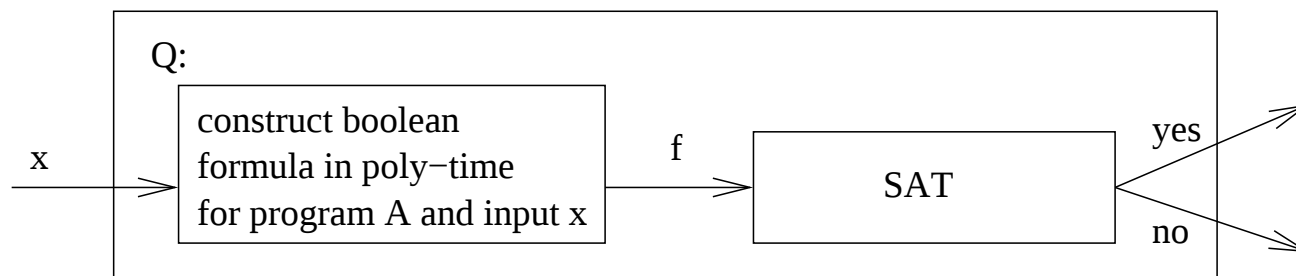
5 “Stav počítača sa mení v čase v súlade s programom.”

k -ty riadok	Formula
ACCEPT alebo REJECT	$Q[i, k] \Rightarrow Q[i + 1, k]$
GOTO ℓ	$Q[i, k] \Rightarrow Q[i + 1, \ell]$
IF $R_\ell = 0$ THEN GOTO m	$Q[i, k] \wedge S[i, \ell, 0] \Rightarrow Q[i + 1, m]$ $Q[i, k] \wedge \neg S[i, \ell, 0] \Rightarrow Q[i + 1, k + 1]$
CHOOSE R_ℓ	$Q[i, k] \Rightarrow Q[i + 1, k + 1] \wedge$ $(S[i + 1, \ell, 0] \vee S[i + 1, \ell, 1])$
atď. pre ďalšie inštrukcie	

SAT je NP-ťaždký: zhrnutie

Vyššie uvedeným postup skonštruujeme pre daný algoritmus A a vstup x formulu f :

- Postup možno zrealizovať v polynomiálnom čase v závislosti od n .
- Výsledná formula má polynomiálnu veľkosť v závislosti od n .
- f je splniteľná $\iff A$ akceptuje x



$\implies$ **Ukázali sme: $Q \leq_p \text{SAT}$ pre ľubovoľné $Q \in NP$**