Chceme:

- Rýchle algoritmy
- Správne algoritmy
- Algoritmy riešiace ťažké problémy

NP-ťažké problémy

Trieda problémov P: problémy riešiteľné v **deterministickom** polynomiálnom čase

Trieda problémov NP: problémy riešiteľné v **nedeterministickom** polynomiálnom čase

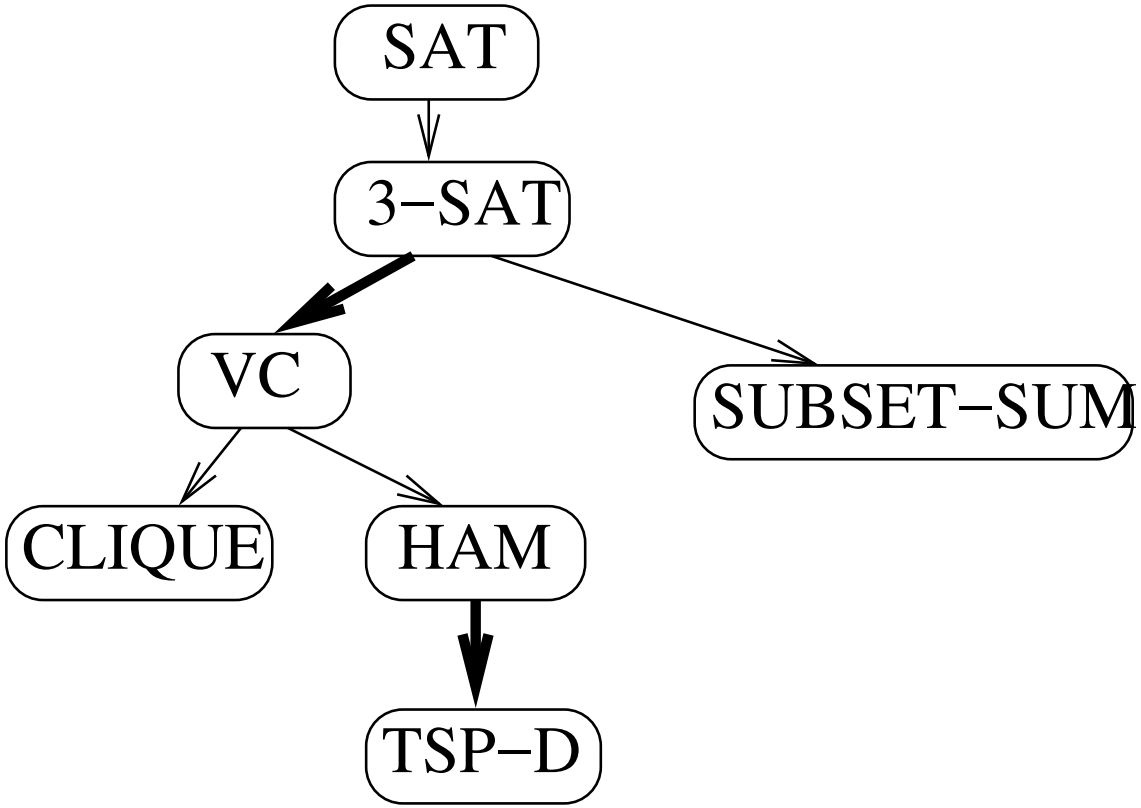
Príklad: nedeterministický algoritmus riešiaci TSP-D

```
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;
```

Ako dokázať, že problém Q je NP-ťažký?

Q je NP-ťažký, akk pre všetky $R \in \mathrm{NP}$ platí $R \leq_P Q$
ak navyše $Q \in \mathrm{NP}$, potom Q je NP-úplný

1. Vyberme si problém N o ktorom **už vieme**, že je NP-úplný
2. Ukážeme $N \leq_P Q$:
   - Navrhneme polynomiálny algoritmus, ktorý prerobí vstup x pre problém N na vstup $f(x)$ pre problém Q.
   - Dokážeme: Ak je x **pozitívny** vstup pre N, potom
     $f(x)$ je **pozitívny** vstup pre Q
   - Dokážeme: Ak je x **negatívny** vstup pre N, potom
     $f(x)$ je **negatívny** vstup pre Q
     —ALEBO—
     Ak $f(x)$ je **pozitívny** vstup pre Q, potom
     x je **pozitívny** vstup pre N
3. Keďže N je NP-úplný, Q musí byť **NP-ťažký**.


SAT
3−SAT
VC
SUBSET−SUM
CLIQUE
HAM
TSP−D

Dynamické programovanie

1. **Určíme podproblém.**
   - aké sú rozmery matice, ktorú budeme vypĺňať?
   - aký je presný význam každého políčka matice?
   - kde v matici nájdeme riešenie pôvodnej úlohy?
2. **Vyriešime podproblém za pomoci iných podproblémov.**
   Ako vypočítame jedno políčko matice z iných políčiek matice?
3. **Bázové podproblémy.** Ktoré políčka nemožno vypočítať pomocou vzťahov z predchádzajúceho kroku? Aké hodnoty by mali obsahovať?
4. **Vyberieme poradie vypĺňania.** V akom poradí musíme maticu vypĺňať tak, aby sme v každom kroku mali vypočítané všetky políčka, ktoré potrebujeme na výpočet daného políčka?

Exaktné metódy riešenia ťažkých problémov

- Inteligentné prehľadávanie (napr. A^* algoritmus)
- Pseudopolynomiálne algoritmy
- Celočíselné lineárne programovanie
- Parametrická zložitosť

Aproximačné algoritmy

Algoritmus A je $\boxed{k\text{-aproximačný}}$, ak pre každý vstup x
platí $A(x) \leq kOPT(x)$ (maximalizačné problémy),
resp. $A(x) \geq kOPT(x)$ (minimalizačné problémy).

Algoritmus $A(x, \varepsilon)$ je $\boxed{\text{polynomiálny aproximačná schéma (PTAS)}}$, ak jeho časová zložitosť je **polynomiálna vzhľadom k** $|x|$
a pre každé $\varepsilon > 0$
je $(1 + \varepsilon)$-aproximačný (maximalizačné problémy),
resp. $(1 - \varepsilon)$-aproximačný (minimalizačné problémy).

Algoritmus $A(x, \varepsilon)$ je
$\boxed{\text{plne polynomiálna aproximačná schéma (FPTAS)}}$, ak jeho časová zložitosť je navyše **polynomiálna vzhľadom na** $1/\varepsilon$.

Kladivá: Dôsledná analýza greedy algoritmov

(pre minimalizačné problémy)

- Problém: chceme porovnať výsledok greedy algoritmu s optimálnym riešením, ktorého hodnotu ale nevieme.
- **Dolný odhad D(x)** na hodnotu optimálneho riešenia na základe známych parametrov.
- **Horný odhad H(x)** na hodnotu greedy riešenia.
- Odhad aproximačného faktoru k:

$$k = \frac{\mathsf{GREEDY}(x)}{\mathsf{OPT}(x)} \leq \frac{H(x)}{D(x)}$$

(pre maximalizačné problémy obdobne)

Kladivá: Rozdeľ problém na podstatné a nepodstatné časti

- **Podstatné časti:**
  - Prispievajú **veľkou časťou** do hodnoty riešenia.
  - Majú špeciálne vlastnosti (napr. majú relatívne veľkú hodnotu, je ich pomerne málo, a pod.)
  - Na základe špeciálnych vlastností ich možno **efektívne riešiť optimálnym spôsobom** $\Rightarrow$ riešenie (*)
- **Nepodstatné časti:**
  - Priespievajú len **malou časťou** do hodnoty riešenia.
  - Vďaka tomu **nezáleží na tom**, ako presne ich doplníme ku riešeniu (*), pretože hodnotu príliš nezmenia.

Kladivá: Rozdeľ problém na ľahšie riešiteľné podproblémy

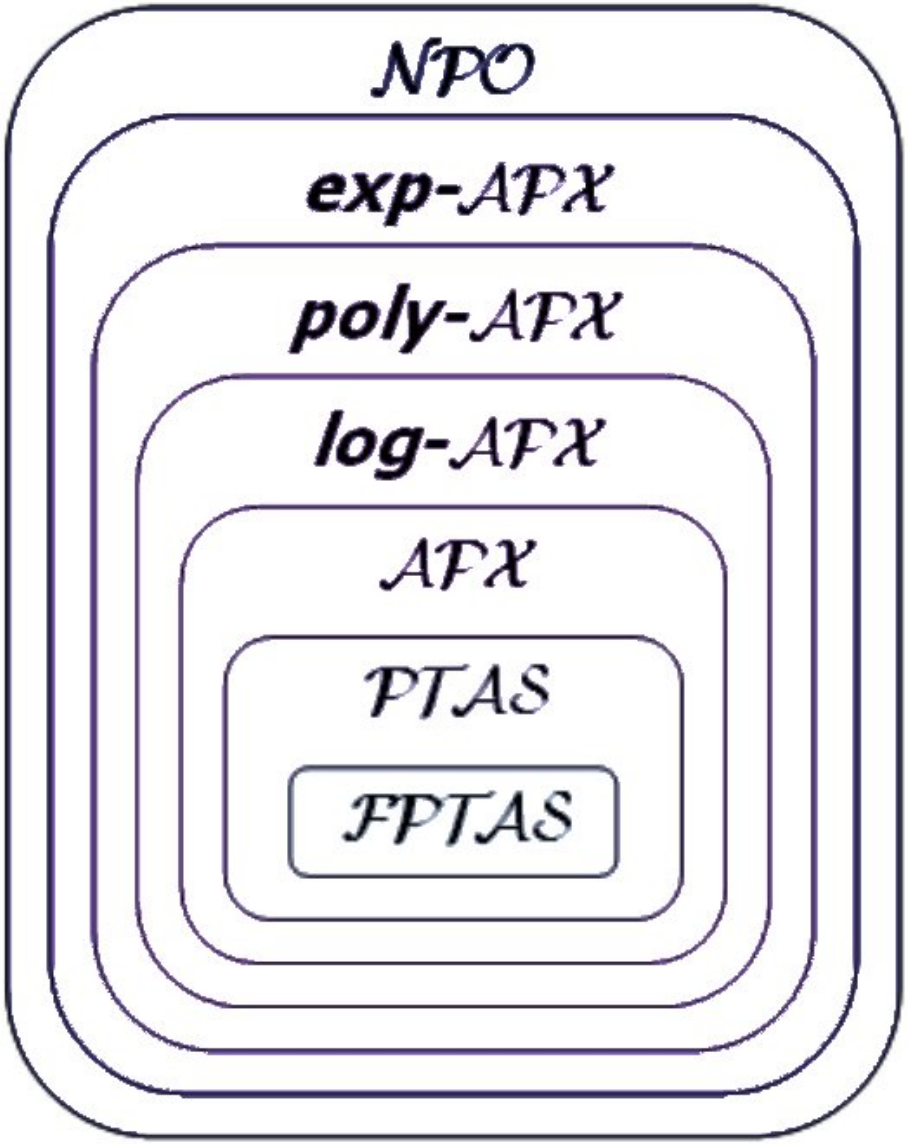
- **Ľahko riešiteľné podproblémy:**
  - Majú špeciálne vlastnosti (napr. malý rozsah, špeciálne usporiadanie prvkov a pod.)
  - Vďaka tomu ich možno **efektívne riešiť optimálnym spôsobom**
  - Každý podproblém $\Rightarrow$ samostatné čiastkové riešenie lokálne optimálne
- **Kombinácia čiastkových riešení:**
  - napr. priamočiaro spojíme čiastkové riešenia dohromady
  - Spojené riešenie nemusí byť optimálne (prekrývajúce sa prvky a pod.)
  - Ukážeme, že sme tým neurobili **príliš veľkú chybu.**

Kladivá: Zaokrúhli veľké čísla

- Predpoklad: pseudopolynomiálny algoritmus
- "Zmenšíme" čísla, ktoré vystupujú ako parametre v pseudopolynomiálnom algoritme, zaokrúhlime kde treba.
- Riešenie nie je exaktné (kôli zaokrúhľovaniu).
- Ukážeme, že zaokrúhľovaním sme neurobili veľkú chybu.

Kladivá: Relaxácia ILP

- Zapíšeme problém ako inštanciu celočíselného lineárneho programovania (ILP).
- **Relaxácia:** Nebudeme rešpektovať požiadavku na celočíselnosť riešenie, t.j. podmienky typu $x \in \{0, 1\}$ nahradíme podmienkou $0 \leq x \leq 1$.
- Riešením výsledného lineárneho programu dostaneme pseudoriešenie, ktorého hodnota **nie je horšia** ako optimálne riešenie.
- **Zaokrúhlenie:** Premenné, ktoré nie sú celé čísla, zaokrúhlime.
- Ukážeme, že sme tým **príliš nezmenili hodnotu riešenia.**


NPO
exp-APX
poly-APX
log-APX
APX
PTAS
FPTAS

Pravdepodobnostné algoritmy

Algoritmy, ktoré využívajú **náhodné čísla.**

Las Vegas algoritmy.

- Vždy dajú správnu odpoveď.
- Náhodné čísla ovplyvňujú čas $\Rightarrow$ **očakávaná časová zložitosť**

Monte Carlo algoritmy.

- Bežia vždy rýchlo.
- Občas dajú nesprávnu odpoveď $\Rightarrow$ **pravdepodobnosť chyby** p
  - Jednostranné chyby
    (napr. "áno" je vždy dobre, "nie" môže byť chybné)
  - Obojstranné chyby

Dôležité: Rýchlosť/chybovosť algoritmu **nezávisí od vstupu**,
ale len od výberu náhodných čísel! (t.j. neexistuje "zlý" vstup)

Kladivá (LV): Znáhodnenie deterministického algoritmu

- Namiesto deterministického kroku, ktorý vyžaduje aby sme urobili vyvážený výber, použijeme náhodný výber.
- Trik pri analýze očakávaného času:
  - Všetky prípady rozdelíme na **dobré** a **zlé**.
  - **Dobré prípady** nastávajú s vysokou pravdepodobnosťou a vieme urobiť dobrý horný odhad $h(x)$ na ich čas.
  - **Zlé prípady** nenastávajú často a aj keby sme použili najhorší možný horný odhad $H(x)$ času, tak to nevadí, lebo pravdepodobnosť je malá.

$$E[T(x,r)] = \sum_r \Pr(r).T(x,r) = \sum_{r \text{ je dobrý}} \Pr(r).T(x,r) + \sum_{r \text{ je zlý}} \Pr(r).T(x,r)$$

$$\leq \Pr(r \text{ je dobrý}).h(x) + \Pr(r \text{ je zlý}).H(x)$$

Kladivá (LV): Kernelizácia problému

- Predpoklad: Niektoré inštancie problému (napr. malé inštancie, riedke grafy, malé váhy a pod.) vieme riešiť efektívnejšie.
- Za pomoci **náhodného výberu** stransformujeme našu inštanciu na **menšiu/riedšiu/...** inštanciu, pričom nová inštancia **s vysokou pravdepodobnosťou** spĺňa podmienky, keď vieme inštancie riešiť efektívne.

Kladivá (LV+MC): Náhodné pochôdzky

- Problém zredukujeme na náhodnú pochôdzku.
- Na odhady potrebných pravdepodobností využijeme matematickú teóriu náhodných pochôdzok:
  - Ako dlho v priemere bude náhodná pochôdzka trvať?
  - Aká je distribúcia dĺžok náhodnej pochôdky?
  - Ako ďaleko sa v priemere počas náhodnej pochôdky dostaneme?
  - ...

Kladivá (LV+MC): Markovova nerovnosť (a ďalšie nerovnosti)

Nech X je náhodná premenná,
pričom $X \geq 0$ a $E[X] = \mu$.
Potom $\Pr(X \geq c\mu) \leq 1/c$

Príklad: Ak máme náhodnú pochôdzku, ktorá trvá v priemere k krokov, tak ak urobíme $2k$ krokov, tak s pravdepodobnosťou viac ako $1/2$ pochôdzku ukončíme!

Kladivá (LV+MC): Metóda svedkov

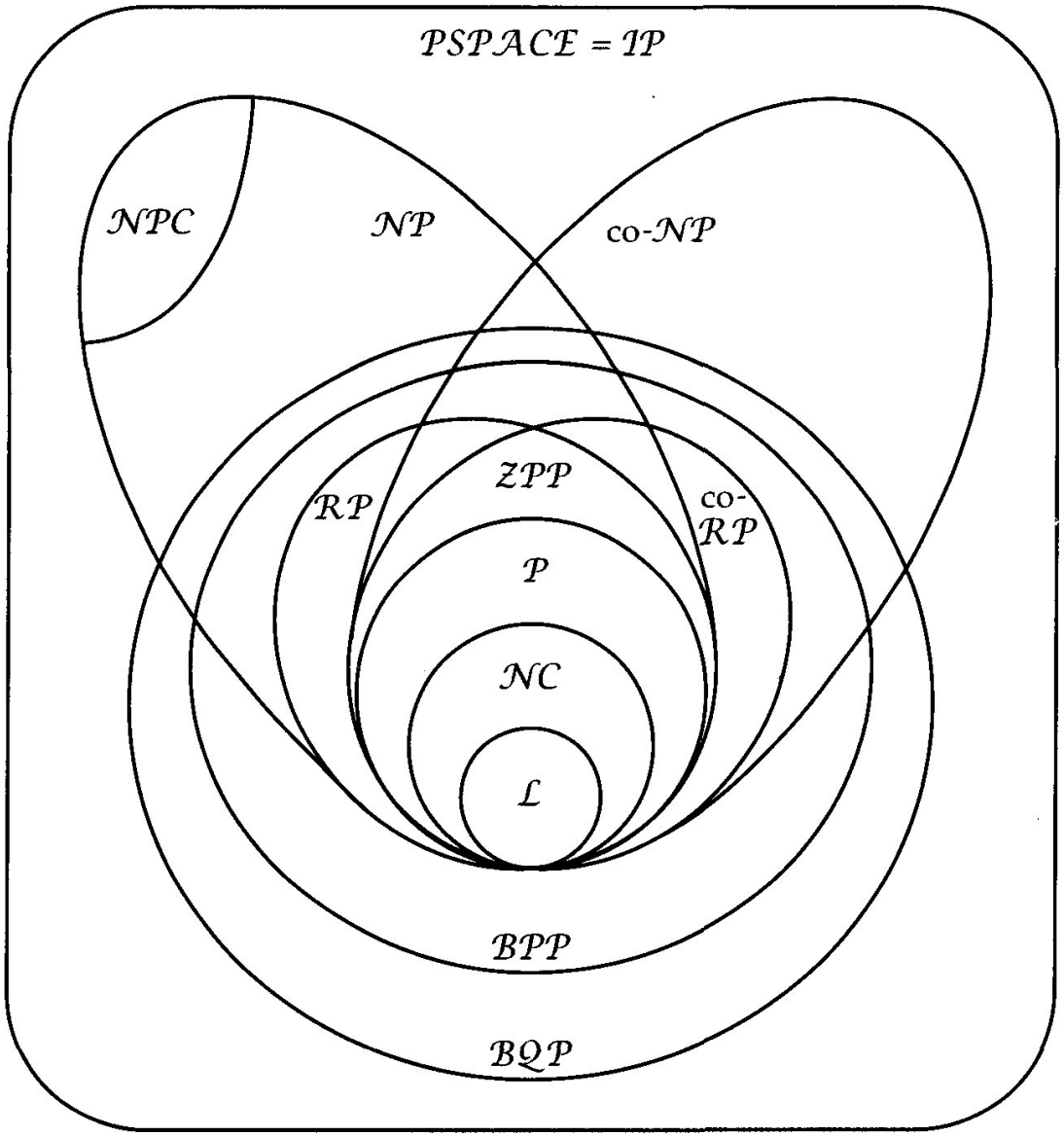
- Ak by sme dostali ku problému nejakú ďalšiu informáciu (napr. čiastočnú informáciu o poradí prvkov, Fermatov svedok pre zložené číslo), vedeli by sme problém riešiť veľmi efektívne.
- Takúto informáciu nazývame **svedok**.
- Použijeme **náhodne vygenerovaného na svedka**!
- Ukážeme, že **zlého svedka**, t.j. takého, ktorý vedie k dlhému času (LV) alebo k chybnému riešeniu (MC) vygenerujeme **s malou pravdepodobnosťou**.

Kladivá (MC): Zlepšovanie úspešnosti opakovaním

- Predpokladajme, že máme MC algoritmus, ktorý urobí **chybu s pravdepodobnosťou** p.
- Ak takýto algoritmus spustíme k, krát, pravdepodobnosť že urobí chybu vo všetkých behoch je p^k
- Jednostranný MC algoritmus s pravdepodobnosťou chyby $1/2$, pravdepodobnosť správnej odpovede pri 4 opakovaniach je $\approx 94\%$.
- Jednostranný MC algoritmus s pravdepodobnosťou chyby 90% 20 opakovaní: pravdepodobnosť **správnej odpovede** $\approx 88\%$

Kladivá (MC): Metóda odtlačkov (fingerprinting)

- Namiesto porovnávania (veľkých) objektov bit po bite porovnávame len ich **odtlačky.**
- Odtlačky sú obvykle krátke a jednoducho porovnateľné.
- Výpočet odtlačku závisí od vygenerovaných náhodných čísel.
- Analýza: aká časť odtlačkov môže viesť ku chybnej zhode?


PSPACE = IP
NPC
NP
co-NP
RP
ZPP
co-
RP
P
NC
L
BPP
BQP