Složenost računanja i klase problema ($NP\ problemi$)

Algoritmi

Jednostavni problemi

- Do sada smo posmatrali jednostavne probleme!
  - Pretrage, sortiranja, obilazak grafa, ...
- Za jednostavne probleme postoje efikasni algoritmi čija je vremenska složenost $O(P(n))$ ograničena nekim polinomom $P(n)$ od veličine ulaza n**.**
- Klasu efikasnih algoritama označavamo sa P jer imaju polinomsko vreme izvršavanja $(\sim n^c)$.
  - U ovu klasu spadaju i problemi čija je složenost npr. $O(n^{10})$, $O(n^{100})$,...
    To nisu brzi algoritmi, a i izmišljeni su!
  - U praktičnoj primeni je obično mali stepen vremenske složenosti P algoritama (npr. n^2)

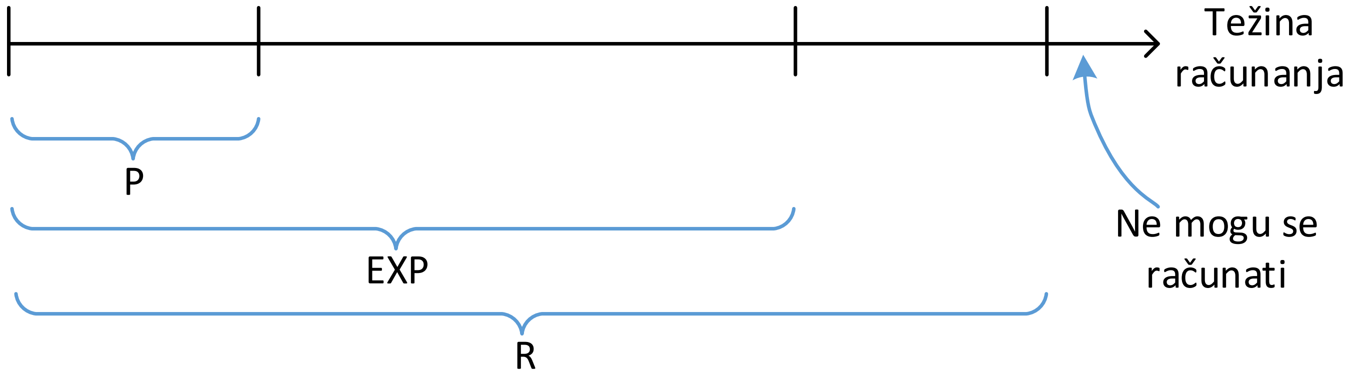
- Takođe, rešenje P problema se može jednostavno proveriti.

„Složeni“ problemi

- Nemaju svi problemi rešenja u obliku jednostavnih i efikasnih algoritama.
  - **Ne mogu se svi problemi rešiti u polinomskom vremenu**!
- Postoji dosta problema za čije rešenje se ne zna ni jedan algoritam polinomske složenosti.

Osnovne “klase” problema

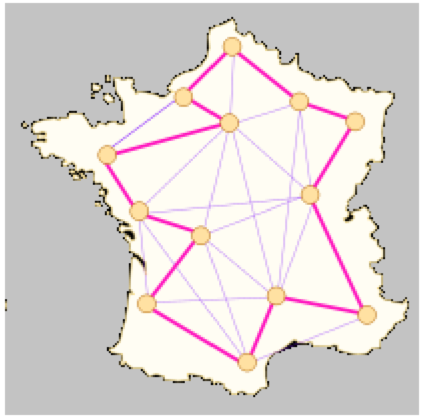
- P – problemi rešivi u **polinomskom** vremenu $(\sim n^c)$
- EXP – problemi rešivi u **eksponencijalnom** vremenu $(\sim 2^{n^c})$
- R – problemi rešivi u **konačnom** vremenu
- Nerešivi problemi



$$P \subset EXP \subset R$$
$$P \neq EXP \neq R$$

Primer složenog problema

- Problem: Preduzeće koje isporučuje pošiljke koristi vozila za prevoz. Svako vozilo u radnom danu treba da preveze n paketa na n lokacija i da se vrati u garažu.
  - To znači da vozilo treba da poseti $n + 1$ lokaciju.
  - Pretpostavimo da se za prevoz iz svake u svaku drugu lokaciju poznaju troškovi prevoza. (tabela sa $(n + 1) \cdot (n + 1)$ vrednosti)
  - Treba odrediti putanju (rutu) koja polazi i završava se u garaži a da su ukupni troškovi prevoza što je moguće manji!
  - Ovo je poznat problem „Trgovačkog putnika“



Primer složenog problema (2)

- Koliko je ovaj problem težak?

- Koliko ima mogućih različitih putanja?

  Broj putanja je $n \cdot (n-1) \cdot (n-2) \ldots \cdot 3 \cdot 2 \cdot 1 = n!$
- Realni podaci: vozilo u proseku prevozi 170 paketa od kojih se nekoliko isporuči na istu adresu. Neka se radi o svega 20 različitih adresa, što daje 20! = 2.432.902.008.176.640.000 putanja.
  - Ako program generiše/procesira 10^{12} (nerealno veliki broj) putanja svake sekunde treba mu oko mesec dana da obradi sve putanje!
  - Ako program generiše/procesira 10^{9} (i dalje veliki broj) putanja svake sekunde treba mu oko 1000 meseci!
  - Ta obrada se odnosi samo na jedno vozilo i jedan dan. Veliki isporučioci imaju desetine hiljada vozila i rade svaki dan u nedelji.

Primer složenog problema (3)

- Rešavanje problema analizom svih mogućih putanja nema smisla (za veći broj lokacija) i uzimanje one koja ima najmanje troškove – nije praktičan algoritam!
- Da li postoji bolji algoritam?
  - Da, postoji, ali nije poznat algoritam polinomske složenosti.
  - Ali, nije ni dokazano da takvo rešenje ne postoji.

- Šta onda da radimo!?
  - Biramo inženjerski pristup i tragamo za rešenjem koje možda nije najbolje, ali je „dovoljno dobro“.
  - Za ovu klasu problema postoje brojni aproksimativni algoritmi.
    - Npr. aproksimativni algoritam za problem Trgovačkog putnika radi u polinomskom vremenu a daje putanju koja je 50% duža od optimalne.

Još primera …

- Otkrivanje zatvorene putanje u usmerenog grafu čije je pojačanje negativno $\in P$
- Šah na tabli $n \times n \in EXP$ (ne pripada $\notin P$)
  - Ko će pobediti ako se zna stanje na tabli?
- Tetris $\in EXP$ ali se ne zna da li je $\in P$
  - Da li se data sekvenca datih oblika može „preživeti“?
- *Halting* problem $\notin R$
  - Da li se neki kompjuterski program završava (za dati ulaz) ili se beskonačno izvršava?
  - Ni jedan algoritam ovo ne rešava korektno u konačnom vremenu za proizvoljan dati program (+ulaz).
  - Ujedno je i **problem odlučivanja** čiji je izlaz {Da, Ne}

Problem odlučivanja

- Problem odlučivanja je funkcija $f : \mathbb{N} \rightarrow \{0,1\}$
  - Ulaz je binarni string ≈ celobrojan nenegativan broj $\in \mathbb{N}$
  - Izlaz je {Ne, Da} = {0,1}

- Primeri problema:
  - Da li je paran broj? – preslikava skup $\mathbb{N}$ na 01010101010101…
  - Da li je neparan broj? – preslikava skup $\mathbb{N}$ na 10101010101010…
  - Da li je prost broj? – preslikava skup $\mathbb{N}$ na 01101010001010…

- Svaki od ovih problema preslikava skup $\mathbb{N}$ na njegov partitivan skup (podskup elemenata iz $\mathbb{N}$) $\mathcal{P}(\mathbb{N})$. Primetiti da je broj partitivnih skupova $\mathcal{P}(\mathbb{N})$ veći od $\mathbb{N}$.

- Alternativno tumačenje: ako se svaki rezultat problema odlučivanja predstavi kao 0,xxx… gde je sa xxx... označen beskonačan niz sastavljen od 0 i 1 (npr. 0,01010101010101…), što je definicija realnog broja, tj. problema odlučivanja ima $|\mathbb{R}|$.

Problem odlučivanja

- Ako posmatramo sve moguće programe (program $\approx$ konačan binarni string $\in \mathbb{N}$) i sve moguće probleme odlučivanja ispada da je broj programa manji od broja problema!

  tj. $|\mathbb{N}| < |\mathcal{P}(\mathbb{N})|$ ili $|\mathbb{N}| < |\mathbb{R}|$

- **Znači: nemaju svi problemi rešenje (program koji rešava problem)!**
  - Ovo je teorijski interesantno, ali srećom, većina praktičnih problema ima rešenje

„Nalik“ slični problemi

- Traženje najkraćeg/najdužeg puta u grafu
  - Traženje najkraćeg puta u grafu $G(V, E)$ od datog čvora je algoritam složenosti $O(m + n \log_2 n)$
  - Traženje najdužeg puta u grafu između dva čvora je $NP - kompletan$ problem.

- Ojlerova putanja/Hamiltonov ciklus u grafu
  - Ojlerova putanja u grafu je kružna putanja koja prolazi kroz svaku granu grafa (tačno jednom) i pri tome dozvoljava višestruke posete istom čvoru. Grane Ojlerove putanje se mogu odrediti u $O(m)$ vremenu.
  - Hamiltonov ciklus je zatvorena putanja koja sadrži svaki čvor grafa. To je $NP - kompletan$ problem.

- ...

NP problemi

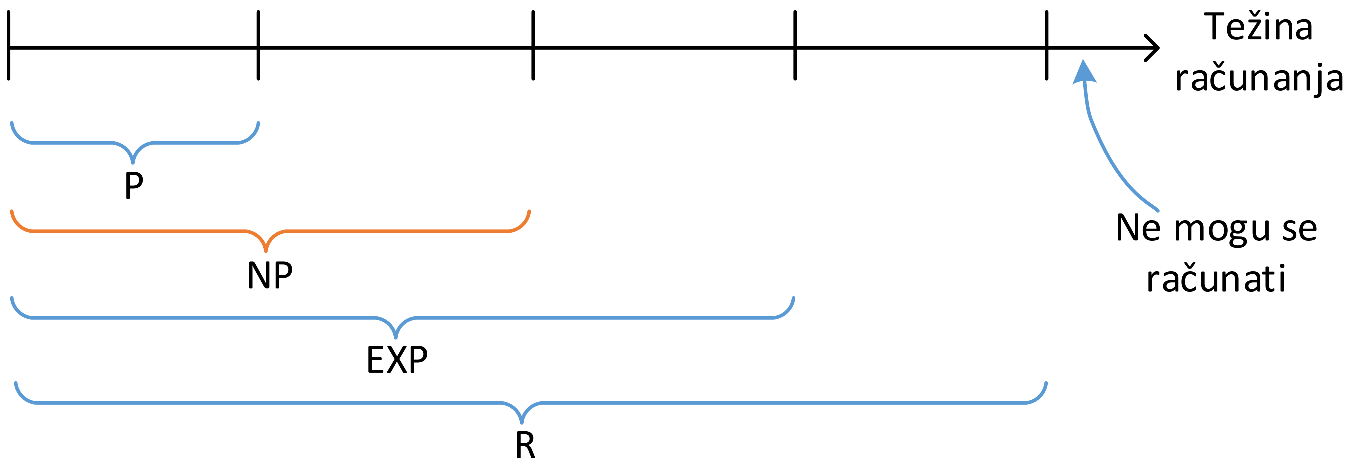
- Posebna klasa problema? (dilema!)
- NP problem je podskup **problema odlučivanja**.
- Definicija: Svaki NP problem je rešiv u polinomskom vremenu uz upotrebu **nedeterminističkog modela računanja** (nedeterminističke mašine)
  - gde svako odlučivanje ima grananje i ona se mogu jednovremeno sračunati.
- NP sadrži P probleme
- Nedeterministički model računanja se može sprovoditi na determinističkoj mašini (~ klasičan računar), ali sporo (rekurzivno, bez mogućnosti jednovremenog računanja)
  - Ili, deterministička mašina bi radila efikasno kao i nedeterministička kada bi kod svakog odlučivanja „srećno" pogodila granu (put) koji vodi ka rešenju (bez probanja svih opcija kod grananja).

NP problemi

- (kada se koristi deterministička mašina)
  Kod NP problema se možda ne zna način (algoritam) za „brzo“ dobijanje rešenja (rešenje se ne dobija u polinomskom vremenu), ali se nekako dobijeno **rešenje može proveriti u polinomskom vremenu**.
  - ovde se govori o konkretnom rešenju problema, a ne samo o „Da/Ne?“

- Primer: Problem sume podskupa: Dat je skup celih brojeva. Da li postoji neprazan podskup čija je suma nula?
  - Na skupu {−2, −3, 15, 14, 7, −10} postoji podskup {−2, −3, −10, 15} čija je suma 0.
  - Jednostavno se proverava – u linearnom vremenu (treba samo sabrati sve elemente podskupa).
  - Teško se pronalazi jer treba probati $2^n - 1$ kombinacija (što je eksponencijalno vreme).

NP problemi (2)



- Primer: Tetris $\in NP$ (za datu listu ulaza određuje da li će igrač „preživeti“)
  - Nedeterministički algoritam – odluka kod svakog poteza
  - Dokaz za „preživljavanje“ je jednostavan – treba implementirati ponašanje poteza i primeniti ih.

P problem je NP problem

- Svaki P problem je ujedno i NP problem jer ne samo da se za dato rešenje P problema može proveriti da je ono korektno, nego se za P vreme može i pronaći takvo rešenje.

Milenijumski problem: $P = NP$ ili $P \neq NP$?

- Da li za svaki problem za koji neki algoritam može brzo da proveri dato rešenje (u polinomskom vremenu) takođe postoji algoritam koji brzo pronalazi takvo rešenje (u polinomskom vremenu)?
- Drugačije rečeno: Da li su (ili nisu) svi NP problemi ujedno P problemi?

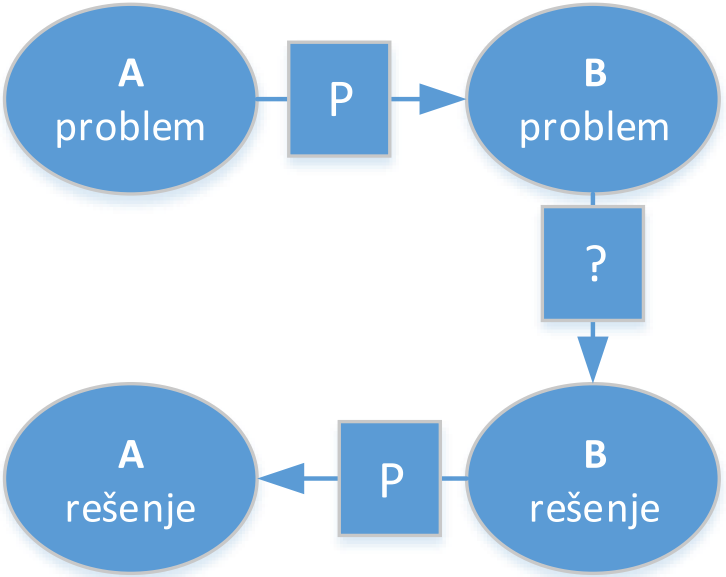
- Većina smatra da je $P \neq NP$ ali za to nema dokaz.
- U načelu traženje rešenja problema (ili dokazivanje da je rešenje korektno) je teže od njegove provere.

- Jedan od 7 matematičkih problema definisan 2000. godine od strane Clay Mathematics instituta. Korektno rešenje donosi nagradu od 1M$.

Svođenje ili redukcija problema

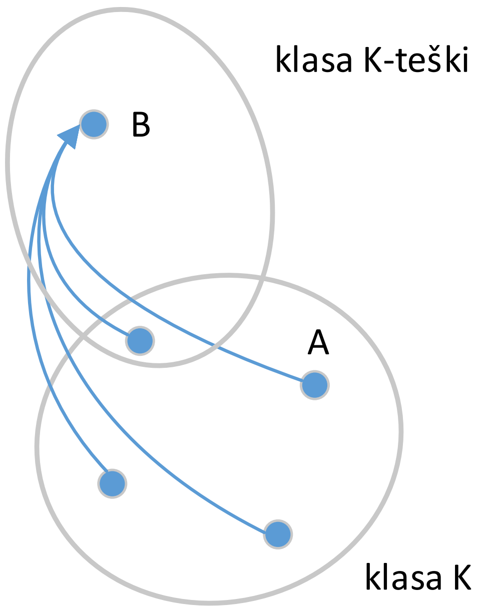
- Redukcija problema A je transformacija tog problema u drugi problem B čijim se rešavanjem dobija rešenje osnovnog problema A.
  - transformacija $\in P$
  - ako znam da rešim B onda znam da rešim A

- Ovo daje poredak problema po složenosti, tj. A je barem težak kao B



Svođenje ili redukcija problema (2)

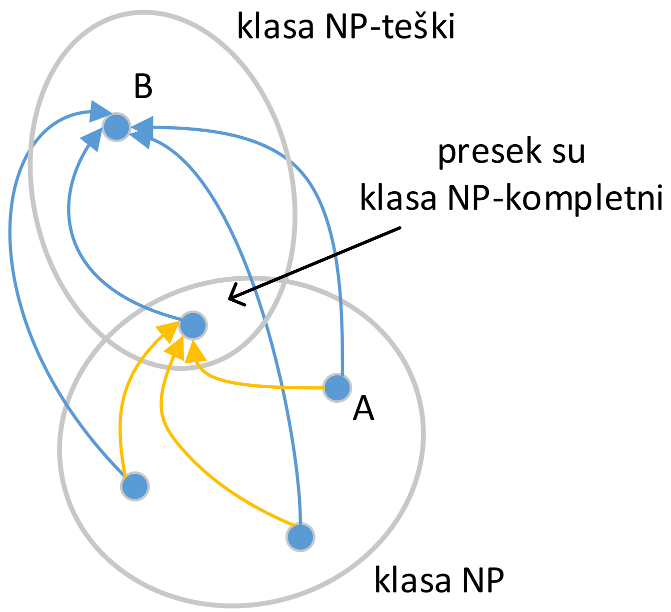
- Definicija: Ako se svaki problem iz klase K (A problemi) može redukovati na problem B onda B pripada klasi $K - teških$ problema ($K - hard$).



$NP - težak, NP - kompletan, \ldots$

- $NP - težak$ problem je ($NP - hard$) problem težak barem kao svaki NP problem.
  - ne mora biti NP problem jer se njihova rešenja ne moraju proveriti u polinomskom vremenu.

- $NP - kompletan$ problem je $NP \cap NP - hard$



$NP - težak, NP - kompletan$, … (2)

- Na sličan način se definišu $EXP - teški$ i $EXP - kompletni$ problemi (i za svaku drugu klasu, ali npr. za P klasu nema smisla)

- Primer Tetris:
  - Ako je $P \neq NP$ onda Tetris ∈ $NP - P$
  - Tetris je i $NP - težak$ problem, pa je time $NP - kompletan$ problem
- Primer Šah:
  - Ako je $NP \neq EXP$ onda Šah ∈ $EXP - NP$
  - Šah je i $EXP - težak$ problem, pa je time $EXP - kompletan$ problem

$NP - kompletni$ problemi

- $NP - kompletni$ problemi ($NP - complete$) su „najteži“ u NP klasi problema
- Problem je tipa $NP - kompletan$ ako zadovoljava 2 uslova:
  1. On je podklasa NP problema (rešenje se može proveriti u polinomskom vremenu)
  2. Ako za problem postoji algoritam koji se izvršava u polinomskom vremenu, onda postoji način da se svaki problem u NP konvertuje u taj problem na način da se svi oni izvršavaju u polinomskom vremenu.
     - Ili: Efikasan algoritam za $NP - kompletan$ problem postoji ako i samo ako za svaki $NP - kompletan$ problem postoji efikasan algoritam.
- Rasprostranjeno je verovanje da $NP - kompletni$ problemi nemaju efikasan algoritam, ali to nije dokazano, a ni obrnuto.

$NP - kompletni$ problemi

- Danas se za dosta problema zna da su klase $NP - kompletni$.
- Postoje u brojnim oblastima.
  - bulova logika, grafovi, aritmetika, dizajn mreža, skupovi i particionisanje, skladištenje i očitavanje podataka, algebra i teorija brojeva, teorija igara, slagalice, automati, optimizacija, biologija, hemija, fizika, ...
- $NP - kompletni$ problemi se rešavaju tehnikama koje daju približno rešenje – rešenja nisu optimalna (najbolja moguća).
  - Približno rešenje je bolje nego nikakvo, i
  - Približno rešenje je obično dovoljno dobro.

Primeri $NP - kompletnih$ problema

- Problem ranca (*Knapsack problem*)
  Problem kombinatorne optimizacije: Dat je skup elemenata gde su za svaki elemenat poznati masa i vrednost. Odrediti koje elemente treba odabrati tako da stanu u ranac poznate nosivosti, a da im je vrednost što veća (maksimalna).
- Hamiltonov put
  Problem teorije grafova: da li postoji putanja u grafu koja posećuje svaki čvor samo jednom.
- Trgovački putnik
  Problem kombinatorne optimizacije: Dat je spisak gradova i međusobnih rastojanja. Koja je najkraća putanja gde se svaki grad posećuje jednom i vraća se u polaznu tačku?
- Izomorfizam podgrafova
  Data su dva grafa. Da li je drugi sadržan u prvom (da li je njegov podgraf jednak drugom grafu)?
- Problem sume podskupa
  Dat je skup celih brojeva. Da li postoji neprazan podskup čija je suma nula?
- Klik problem (*Clique problem*)
  Teorija grafova: Traži se maksimalan podgraf u grafu gde su svi čvorovi međusobno povezani svaki sa svakim.
- Bojenje grafa
  Teorija grafova: Traži se minimalan broj boja kojima možemo obojiti čvorove grafa tako da susedni čvorovi budu različitih boja.
- Najduža zajednička podsekvenca od *n* stringova
- Minesweeper, Sudoku, većina *puzzle* igara
- …

Kako znamo da je problem $NP - kompletan$?

- Kada određujemo da je problem $NP - kompletan$ onda definišemo koliko je problem težak.
  - uz ogradu da se radi o problemima odlučivanja
- Pokušavamo da dokažemo da je za očekivati da ne postoji efikasan algoritam koji rešava problem.

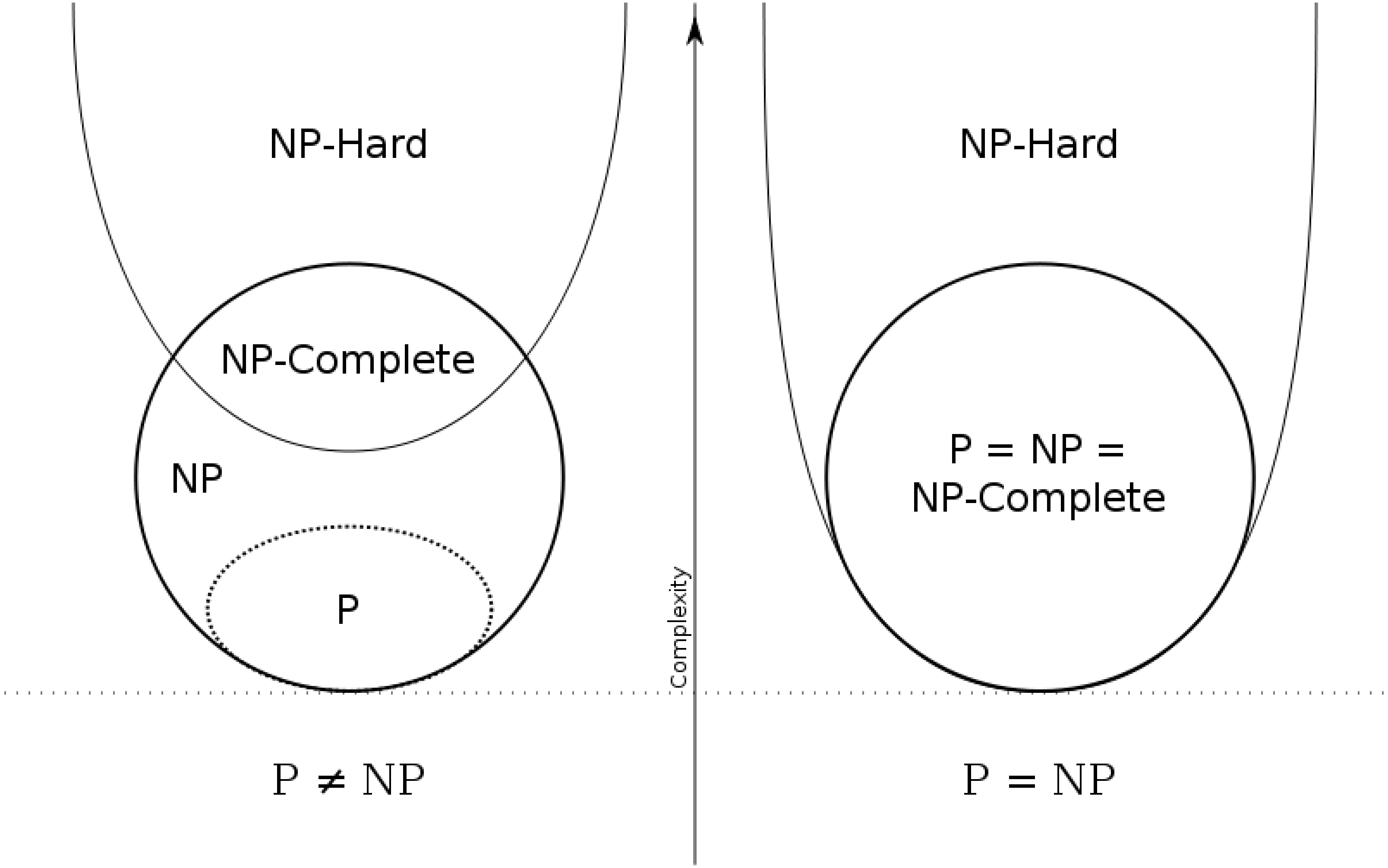
- Tri koncepta se koriste da se pokaže da je problem $NP - kompletan$:
  1. problem se svodi na problem odluke (tako da se rešenje problema svodi na odgovor „da“/“ne“).
  2. Problem se **redukuje** (u polinomskom vremenu) na drugi problem za koji se pretpostavlja da je klase NP i tako redom dok se transformacijama problem ne svede na „prvi $NP - kompletan$ problem“.
  3. „prvi $NP - kompletan$ problem“ je problem za koji je dokazano da je $NP - kompletan$.

$NP - kompletni$ problemi su bitni

- Ako se problem proglasi (pokaže) da je $NP - kompletan$, onda je to dovoljan dokaz da ga ne možemo (za sada) efikasno rešiti
- tada treba primeniti inženjerski pristup
  - ne tražiti (uzaludno) brz algoritam
  - usmeriti se razvoju algoritma za aproksimativno rešenje

- Mnogi često sretani i interesantni problemi naizgled se čine ne težim od problema sortiranja, graph pretrage, i sl., ali su $NP - kompletni$ problemi.

Odnos: P, NP, NP-kompletan, NP-težak



PARALELNI ALGORITMI

Primena paralelnih algoritama

- U poslednje vreme je jedan od glavnih pravaca razvoja računarstva.
- Primenjuju se u računaru sa procesorom sa više jezgara i/ili više procesora.
- Razvijaju se različite tehnike i modeli izračunavanja:
  u osnovi omogućavaju jednovremeno izvršavanje delova algoritma.

Sekvencijalni i paralelni algoritmi

- Do sada su posmatrani sekvencijalni algoritmi
- Osnovne mere složenosti sekvencijalnih algoritama su:
  - Vreme izvršavanje, i
  - Veličina upotrebljene memorije.
- Kod paralelnih algoritama su te iste mere veoma bitne, ali se vodi računa i o drugim resursima:
  - npr. broj angažovanih procesora

Problemi koje rešavaju paralelni algoritmi

- Nisu svi problemi pogodni za rešavanje paralelnim algoritmima!
  - Suštinski sekvencijalni problemi se ne mogu brže izvršavati ni kada imamo neograničen broj procesora.
- Većina algoritama se može makar delom „paralelizovati“, ali ne u svim delovima
  - Npr. problem koji je idealan za paralelizovanje će se $2\ \times$ brže izvršavati upotrebom 2 procesora u odnosu na upotrebu 1 procesora (to nije realno očekivati!)

Amdalov zakon (*Amdahl's law*)

- Daje teorijski maksimalno ubrzanje obrade (algoritma) kada se upotrebi više procesora.
- Vreme izvršavanja upotrebom više procesora je:

$$T(n) = T(1)\left(B + \frac{1}{n}(1 - B)\right)$$

gde je:

- n – broj procesora (niti izvršavanja),
- $B \in [0,1]$ - deo algoritma koji je sekvencijalan,
- $T(1)$ – vreme izvršavanja upotrebom 1 procesora.

- Tj. ubrzanje je

$$S(n) = \frac{T(1)}{T(n)} = \frac{1}{B + \frac{1}{n}(1 - B)}$$

Primer primene Amdalovog zakona

Neka je za izvršavanje algoritma na jednom procesoru potrebno 30 min. Taj algoritam ima deo od 3 min koji se ne može paralelizovati, dok se preostalih 27 min izvršavanja (90%) može paralelizovati.

- Maksimalno ubrzanje ne može biti bolje od 10 puta.

$$S(\propto) = \frac{1}{\left(0.1 + \frac{1}{\propto}0.9\right)} = 10$$

- Ubrzanje sa 4 procesora je 3.08 puta

$$S(4) = \frac{1}{\left(0.1 + \frac{1}{4}0.9\right)} = 3.08$$

Poteškoće implementacije paralelnih rešenja

- Paralalni algoritmi se oslanjaju na paralelno programiranja i/ili distribuirano programiranje.
- Pored vremena potrebnog za izvršavanje (korisnog dela) algoritma potrebno je dodatno vreme za:
  - Inicijalizaciju i/ili sinhronizaciju izvršavanja niti,
  - Sinhronizaciju pristupa deljenim promenljivim,
  - Komunikaciju između niti (kod distribuiranih rešenja).
- Nekada je trajanje „dodatnih aktivnosti“ značajno u odnosu na vreme izvršavanja sekvencijalnog algoritma te se opravdanost paralelizma dovodi u pitanje!
- Dodatno ...
  - Broj raspoloživih procesora je konačan,
  - Angažovanje dodatnih procesora je trošak.

Neki jednostavni paralelni problemi

- Suma elemenata niza (srednja vrednost niza brojeva)
- Traženje maksimuma (ili minimuma)
- Sortiranje – *merge sort*
- Elementarne matrične operacije
- Stablo razapinjanja (u grafu)
- Većina NP-problema se približno rešava „pametno osmišljenom“ grubom silom
- ...