Reševanje problemov

1. Predpogoji za reševanje problemov
2. Kateri problemi so sploh rešljivi?
3. Osnovni principi reševanja
4. Reševanje problemov z algoritmi
5. Primer razvoja algoritma

Osnovni principi reševanja problemov

1. Razumevanje problema
2. Abstrakcija problema
3. Izbira notacije
4. Razbitje na podprobleme
5. Podobnost med problemi
6. Izbira preiskovalne strategije

1. Razumevanje in 2. abstrakcija problema

Razumevanje:

1) Razjasnitev zahtev
   - Poišči implicitne podatke in jih izrazi eksplicitno
   - Identificiraj nepomembne podatke
2) Nedvoumna definicija problema
3) Opis **dejstev, pravil in ciljev**

Abstrakcija:

1) Identificiraj pomembne **objekte** in jih poimenuj
2) Določi **relacije** med objekti in možne **operacije** na njih
3) S tem je definiran **problemski prostor**

PRIMER

Teta Jožefina je sedaj trikrat toliko stara, kot je bil nečak Rožle pred desetimi leti. Rožle je sedaj pol toliko star, kot bo teta Jožefina čez pet let.

Koliko je teta Jožefina starejša od Rožleta?

Nepomembno:
- imena oseb
- sorodstvene vezi

Implicitno:
starost merimo v letih

Dejstva: opisujeta stavka

Pravila:
čez N let sem N let starejši

Cilji:
starost obeh oseb

Pomembni objekti:
starost Jožefine in Rožleta

Poimenujemo: J in R

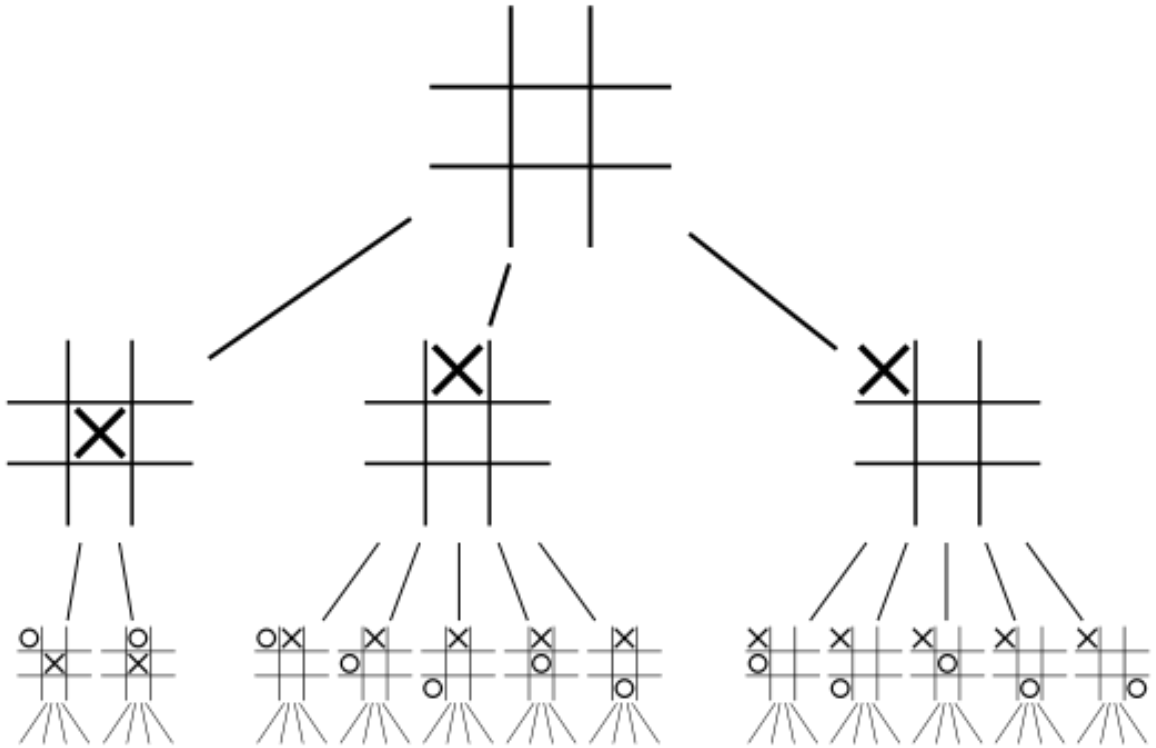
Operacije:
+, - , * , /, …

Relacije:
J = 3(R - 10)
2R = J + 5

Problemski prostor

1. Začetno stanje
2. Vsi možni operatorji za prehode med stanji
3. Ciljno stanje

Rešitev problema je pot od začetnega do ciljnega stanja.
Problemski prostor lahko ponazorimo kot graf (ali drevo).

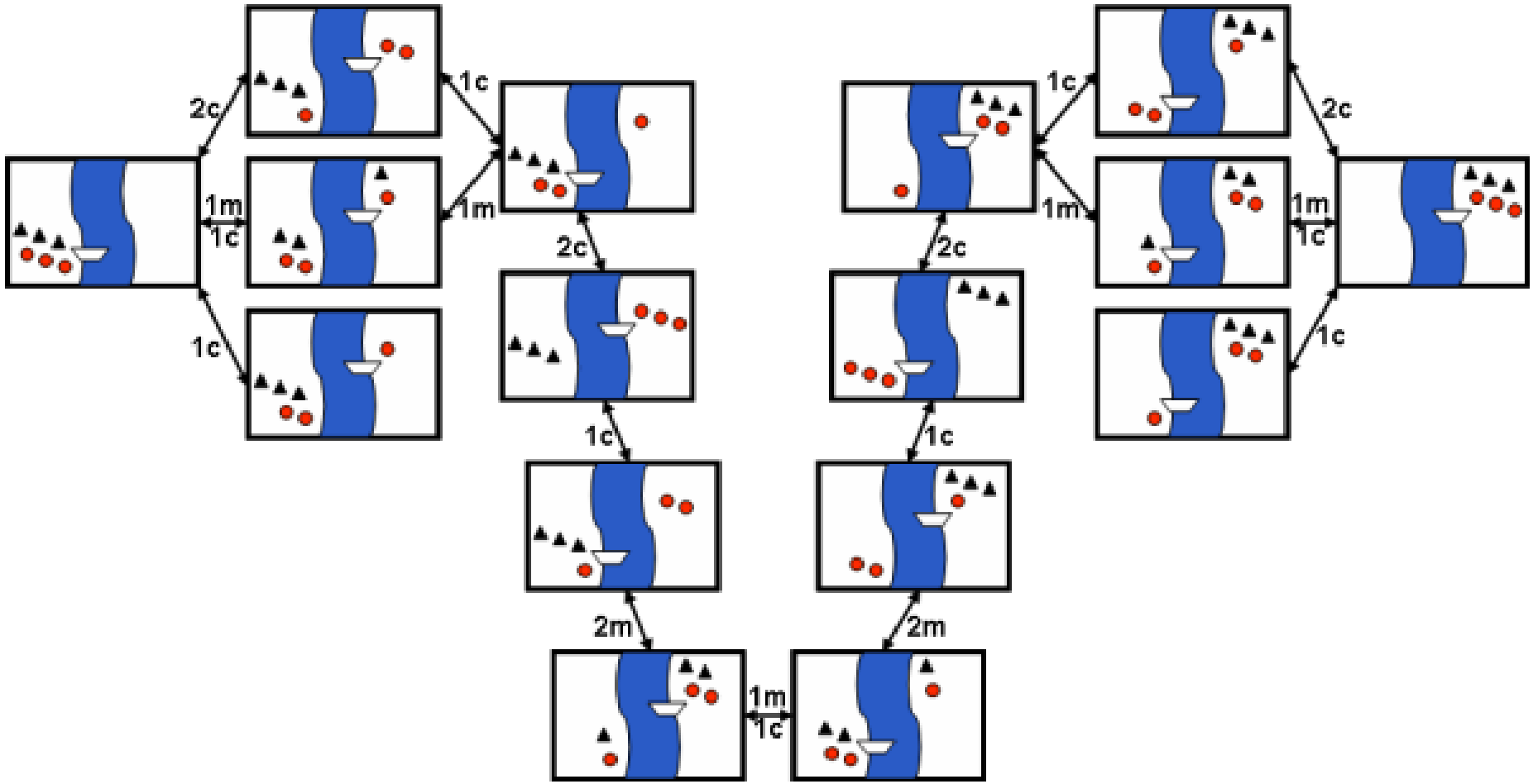


Primer: 3 kanibali in 3 misionarji

Za rešitev problema si lahko narišemo prostor stanj:

1) Začetno stanje

2) Vsi možni legalni prehodi (brez požrtije ☺)



Izbira notacije

Notacija je simbolična predstavitev, ki modelira skupne značilnosti razreda objektov ali situacije in možne operacije nad simboli.

Simboli so večinoma SLIKOVNI in BESEDNI

- Kaj je bolje? Neodločeno (npr. prometni znaki).

Besedni simboli so okrajšave za koncepte, ki so v pomoč spominu in reševanju problemov. Z njimi predstavimo:

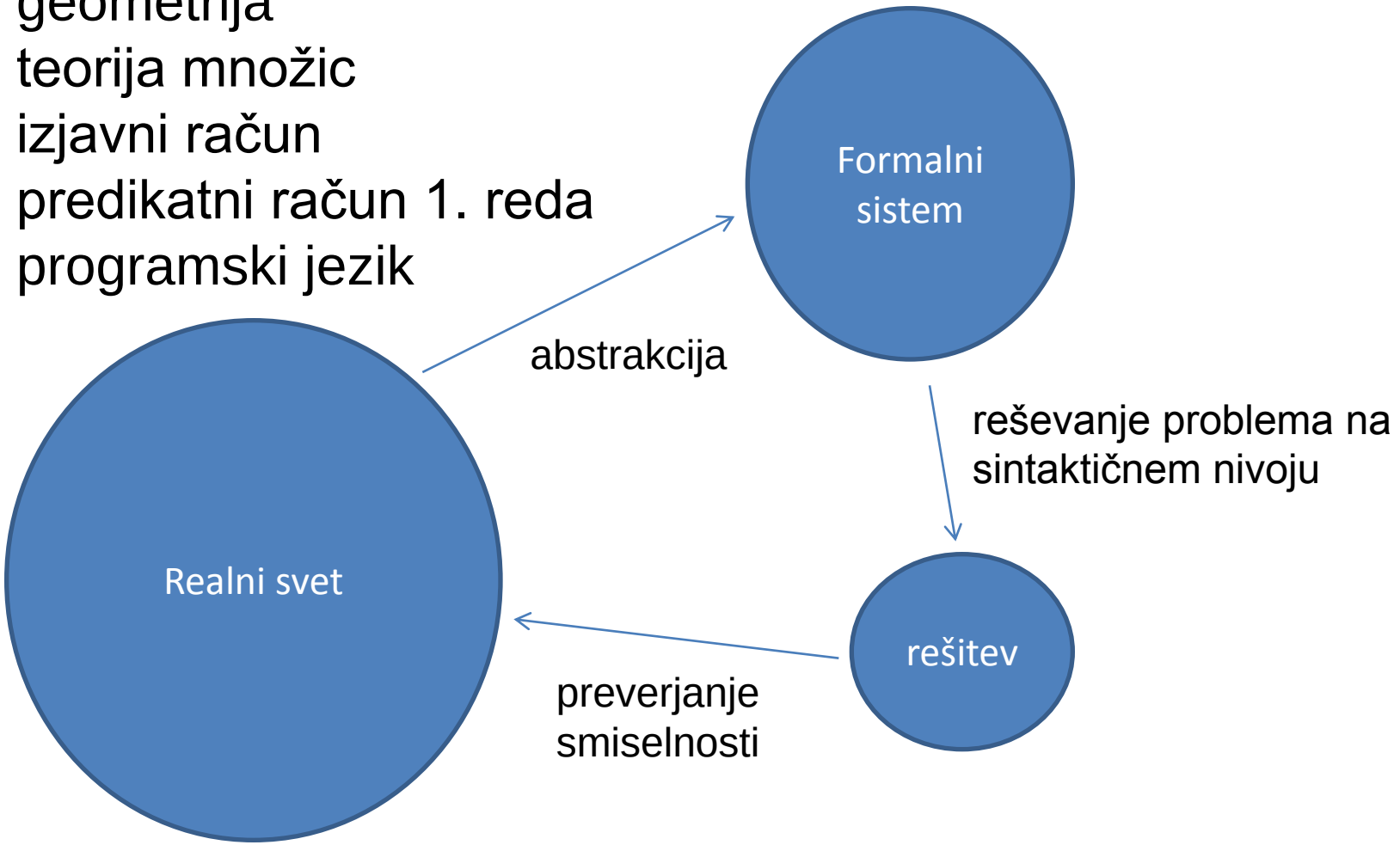
- objekte (imena: I, J, TEMP) in veličine (števila: 35, 17.3)
- operacije (+, -, *, /) in relacije (=, >, <)

Večji del programiranja je spreminjanje notacije, ki jo razume človek, v notacijo, ki jo "razume" računalnik.

Izbira notacije

Formalni sistemi:

- algebra
- geometrija
- teorija množic
- izjavni račun
- predikatni račun 1. reda
- programski jezik



PRIMER

Tone vrže žogo navpično s hitrostjo 8m/s. Do stropa je 3m. Kako dolgo bo žoga letela do stropa?

$s = v\,t + \frac{1}{2}\,g\,t^2$

$3 = 8\,t - 5\,t^2$

$g = -10\ m/s^2$

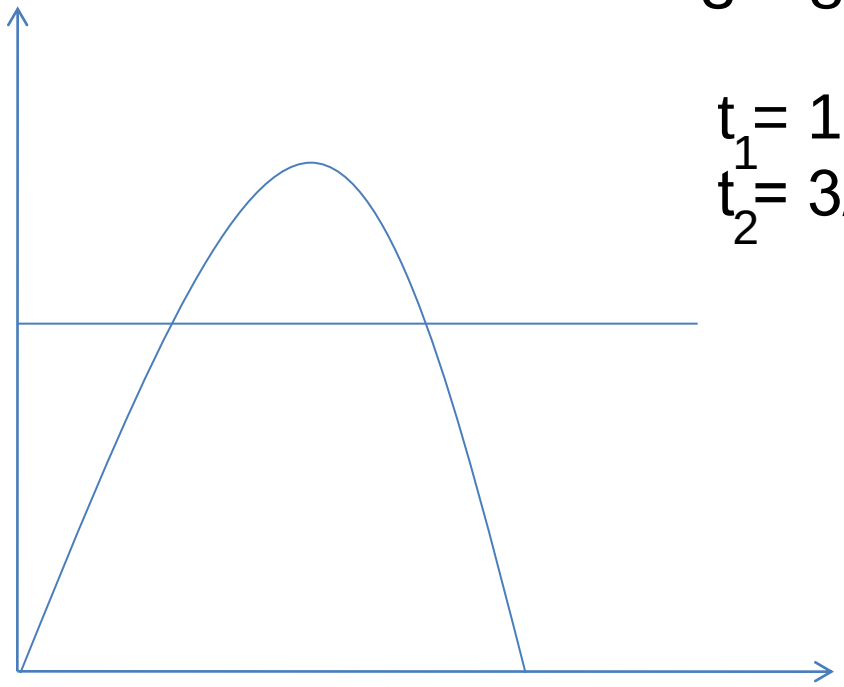
$v = 8\ m/s$

$s = 3\ m$

$t_1 = 1\ s$

$t_2 = 3/5\ s$

kateri čas je pravi?

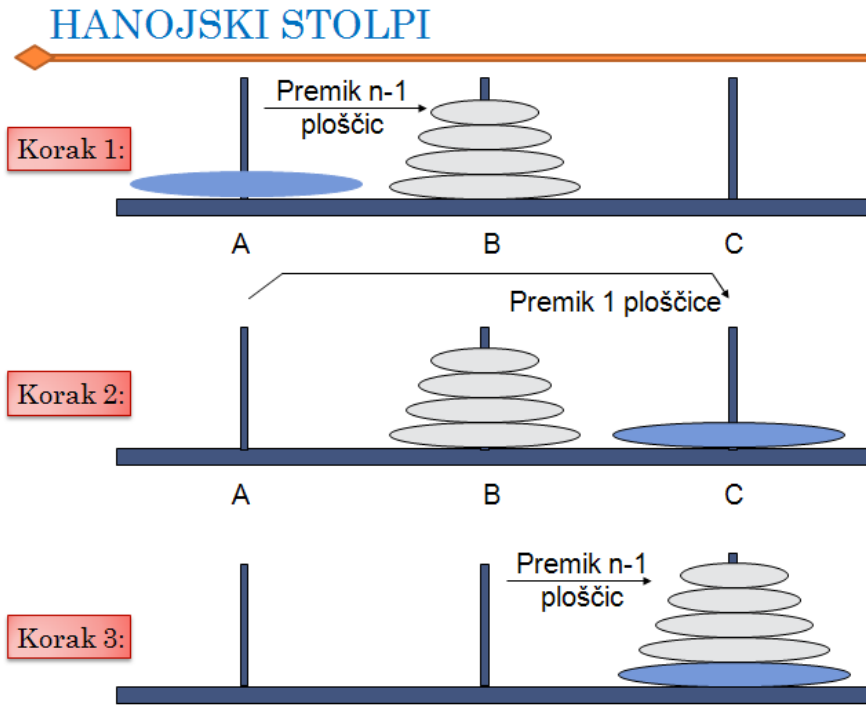


Razbitje problema na podprobleme

Težko brez napak razmišljamo o več stvareh naenkrat.

Razgrajujemo lahko:

- od zgoraj navzdol (top-down)



- od spodaj navzgor (bottom-up)

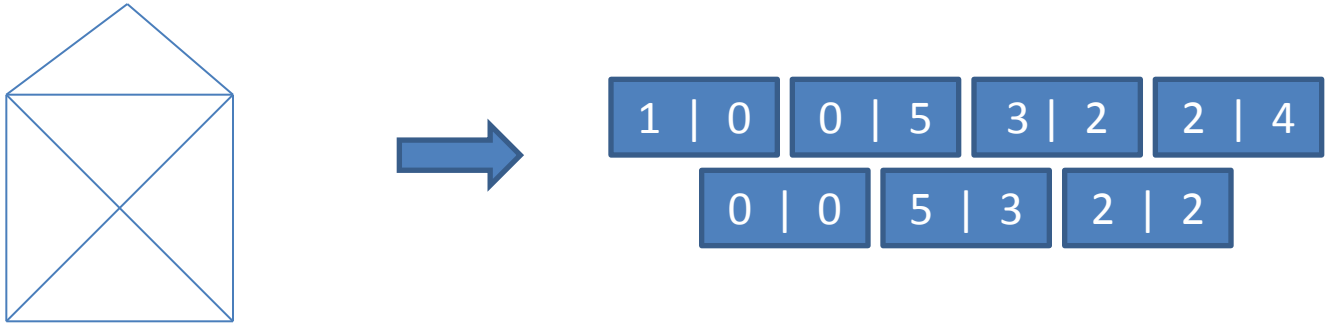
  DINAMIČNO PROGRAMIRANJE: primer Fibonacci

n	1,	2,	3,	4,	5,	6,	7,	8,	9, 10
F	1,	1,	2,	3,	5,	8,	13,	21,	34, 55

Podobnost med problemi

Problem transformiramo:

- ga poenostavimo
- posplošimo
- iščemo posebne primere (npr. ekstremne vrednosti)
- reformuliramo v izomorfni problem



- uporabimo matematično indukcijo
  1. Najdemo rešitev za začetno vrednost (n = 0 ali 1)
  2. Pokažemo, da iz rešitve za n-1 sledi rešitev za n

Matematična indukcija: IZZIV 1

Trditev: Vsi elementi v seznamu so identični.

Dokaz:

1) za n = 1 je trivialno res.

2) Predpostavimo, da je res za n-1. Vzemimo |S| =n:

S = $a_1, a_2, a_3, \ldots\ldots, a_i, \ldots\ldots\ldots, a_{n-1}, a_n$

Tvorimo S1 in S2 dolžine n-1:

S1 = $a_1, a_2, a_3, \ldots\ldots, a_i, \ldots\ldots\ldots, a_{n-1}$

S2 = $a_2, a_3, \ldots\ldots, a_i, \ldots\ldots\ldots, a_{n-1}, a_n$

Po predpostavki so vsi elementi v S1 identični.

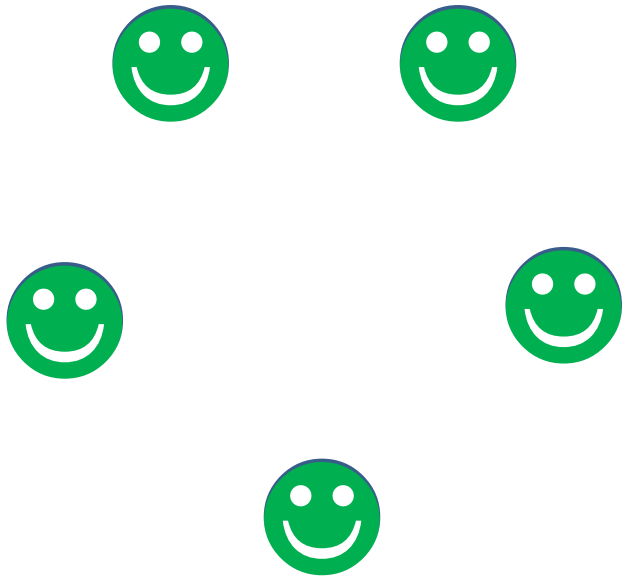
Po predpostavki so vsi elementi v S2 identični.

Iz tega sledi, da so tudi v S vsi elementi identični. ■

Kje je napaka v dokazu?

Matematična indukcija: IZZIV 2

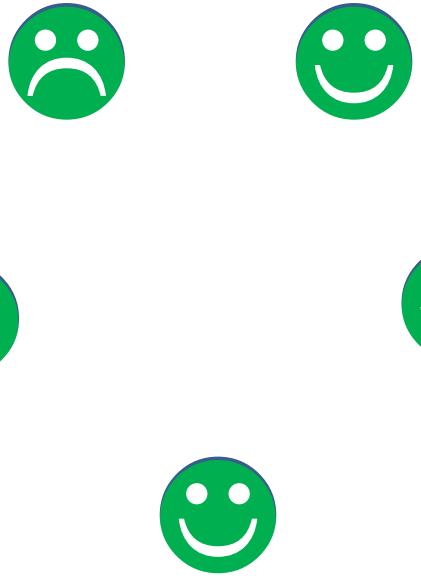
1. Pet profesorjev po prežurirani noči spi.
2. (Zlobni) študent jim vsem pobarva obraze z zeleno barvo.
3. Ko se profesorji zbudijo, vidijo obraze drugih (svojega ne).
4. Vsi se začnejo smejati drugim.



Matematična indukcija: IZZIV 2

1. Pet profesorjev po prežurirani noči spi.
2. (Zlobni) študent jim vsem pobarva obraze z zeleno barvo.
3. Ko se profesorji zbudijo, vidijo obraze drugih (svojega ne).
4. Vsi se začnejo smejati drugim.
5. Najpametnejši med njimi se čez čas neha smejati,
   ker ugotovi, da je tudi sam pobarvan.

Kako je to ugotovil?

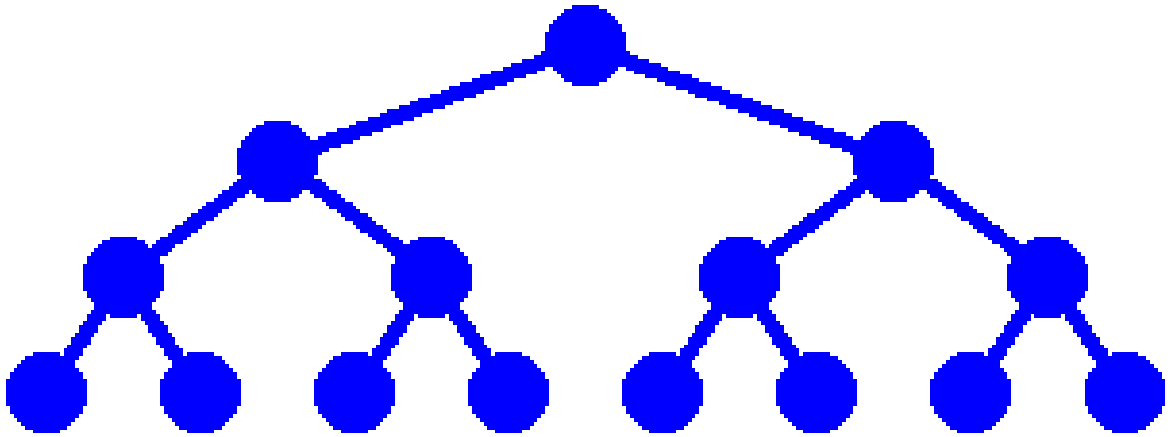


Izbira preiskovalne strategije

Problemski prostori so ponavadi zelo veliki.

Preiskovalno strategijo izberemo glede na velikost prostora in glede na poznavanje njegovih lastnosti.

1. Iskanje optimalne rešitve
2. Iskanje približne (suboptimalne) rešitve

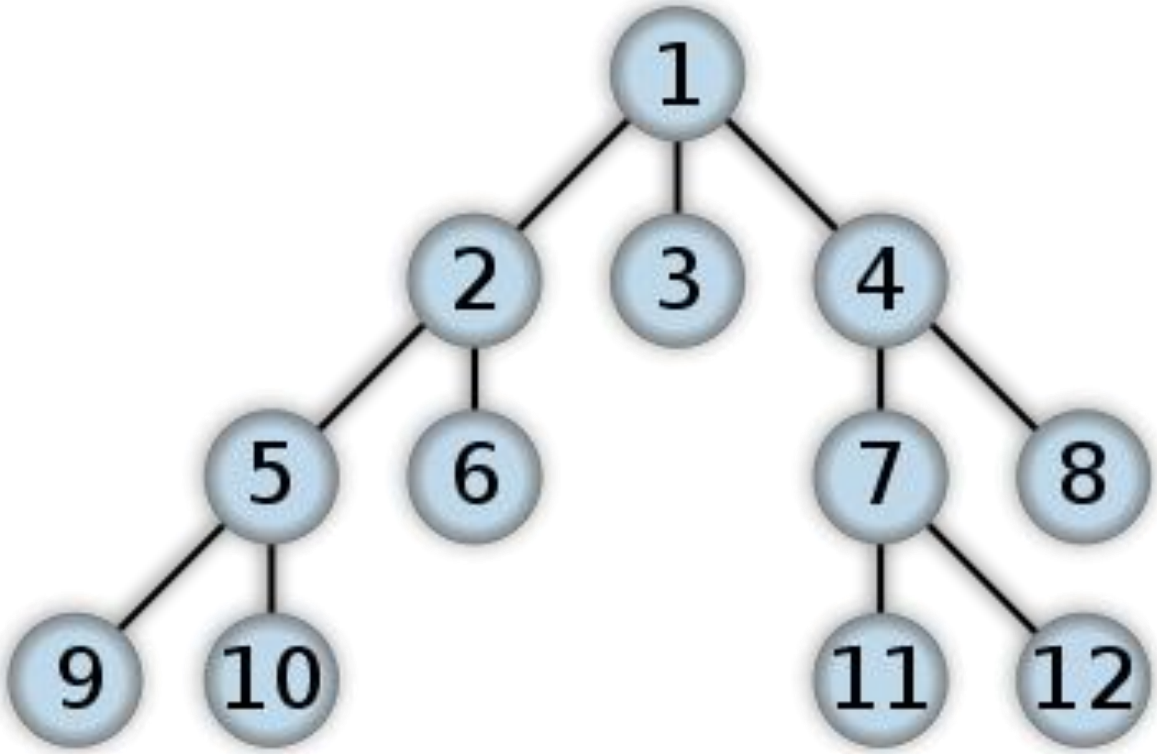


Iskanje optimalne rešitve

Osnovne preiskovalne strategije

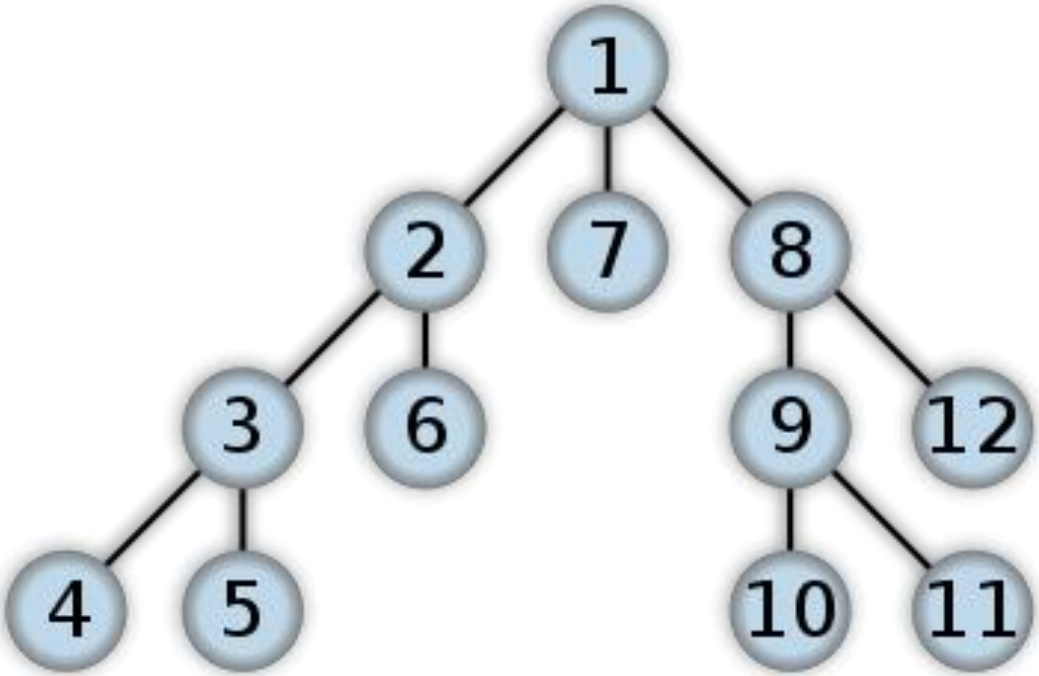
1. Izčrpno iskanje
   a. Iskanje v širino (breadth-first search)
   b. Iskanje v globino (depth-first search)
   c. Iterativno poglabljanje (iterative deepening)
2. Omejeno izčrpno iskanje
   (razveji in omeji – branch and bound)
3. Metoda najprej najboljši (best-first search)
4. Dinamično programiranje

Iskanje v širino



+ Garantirano najde najkrajšo pot do rešitve.
- Z globino prostorska zahtevnost narašča eksponentno!

Iskanje v globino



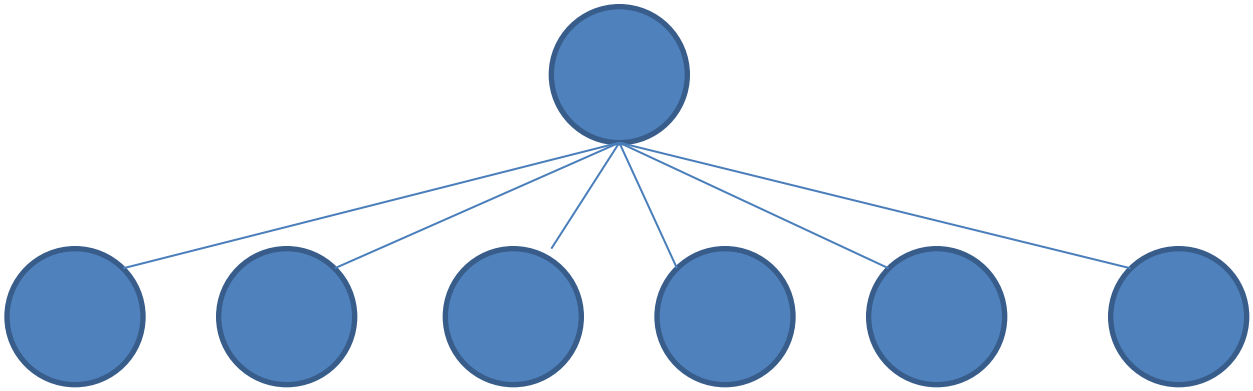
+ Z globino prostorska zahtevnost raste linearno.
- Lahko zgreši rešitev, ki je blizu začetnemu stanju
- Se lahko zacikla (izgubi v neskončni veji).

Iterativno poglabljanje

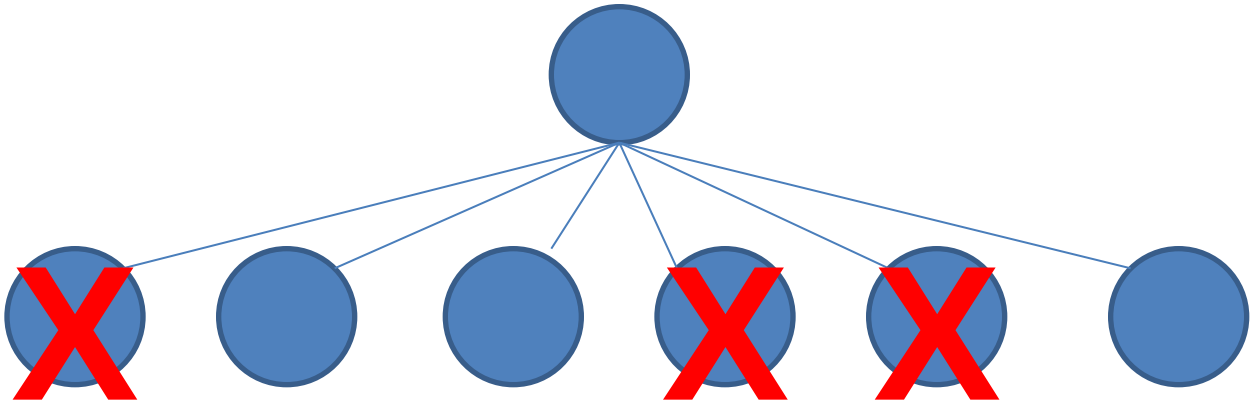
```
max = začetna globina;
Ponavljaj:
        išči v globino z globino, omejeno  na max;
        max = max + 1;
```

+ Garantirano najde najkrajšo pot do rešitve
  (ni težav s ciklanjem)
+ Z globino prostorska zahtevnost raste linearno.
- V vsaki iteraciji preiščemo cel prostor, ki smo ga že preiskali → eksponentna rast časa!

Omejeno izčrpno: razveji in omeji

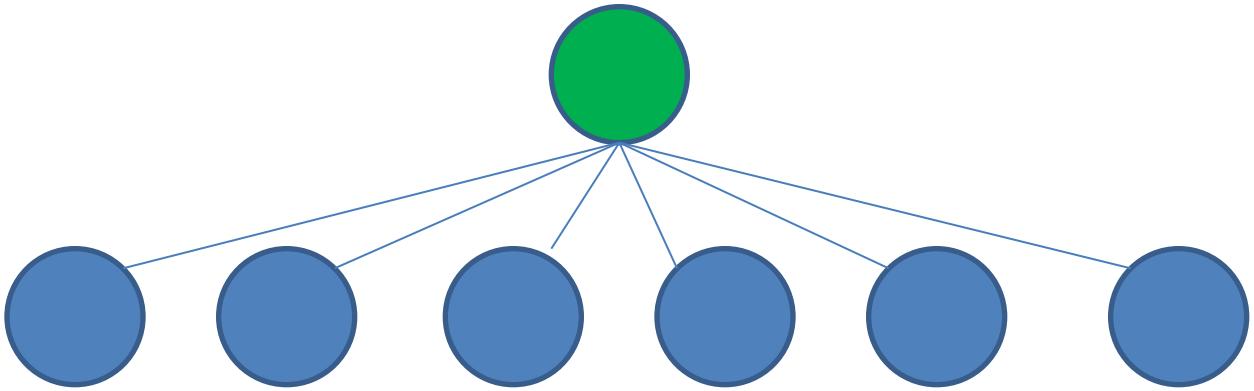


Omejeno izčrpno: razveji in omeji

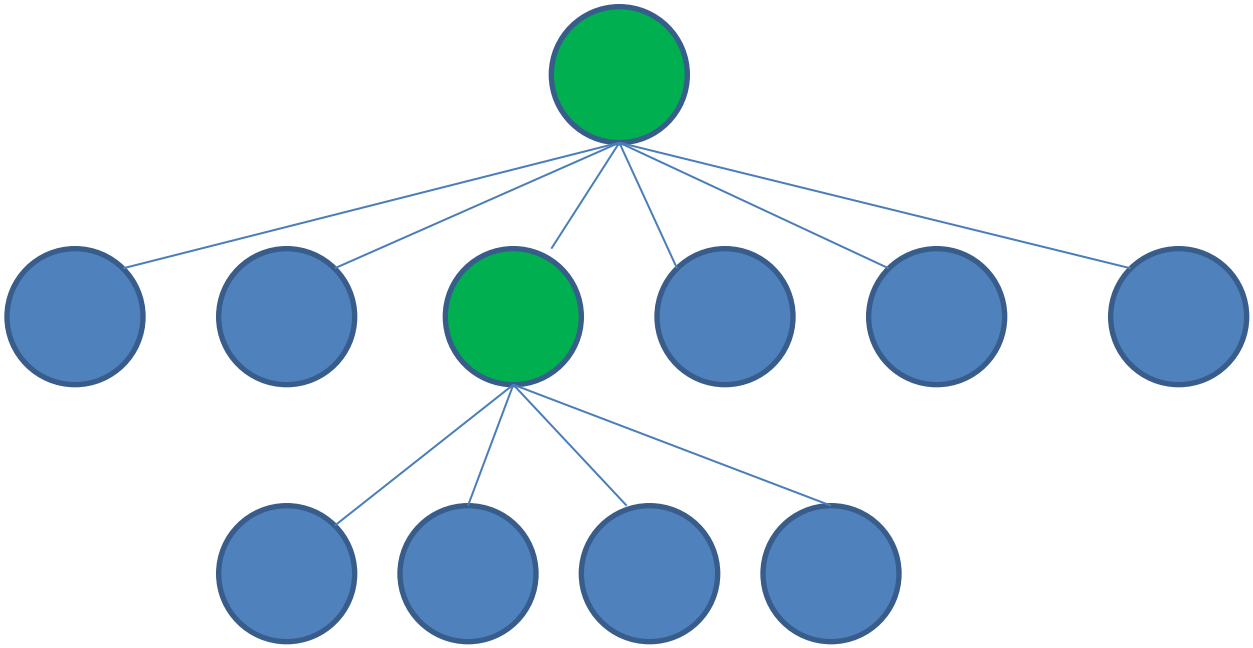


Učinkovitost omejevanja je odvisna od našega poznavanja problemskega prostora.

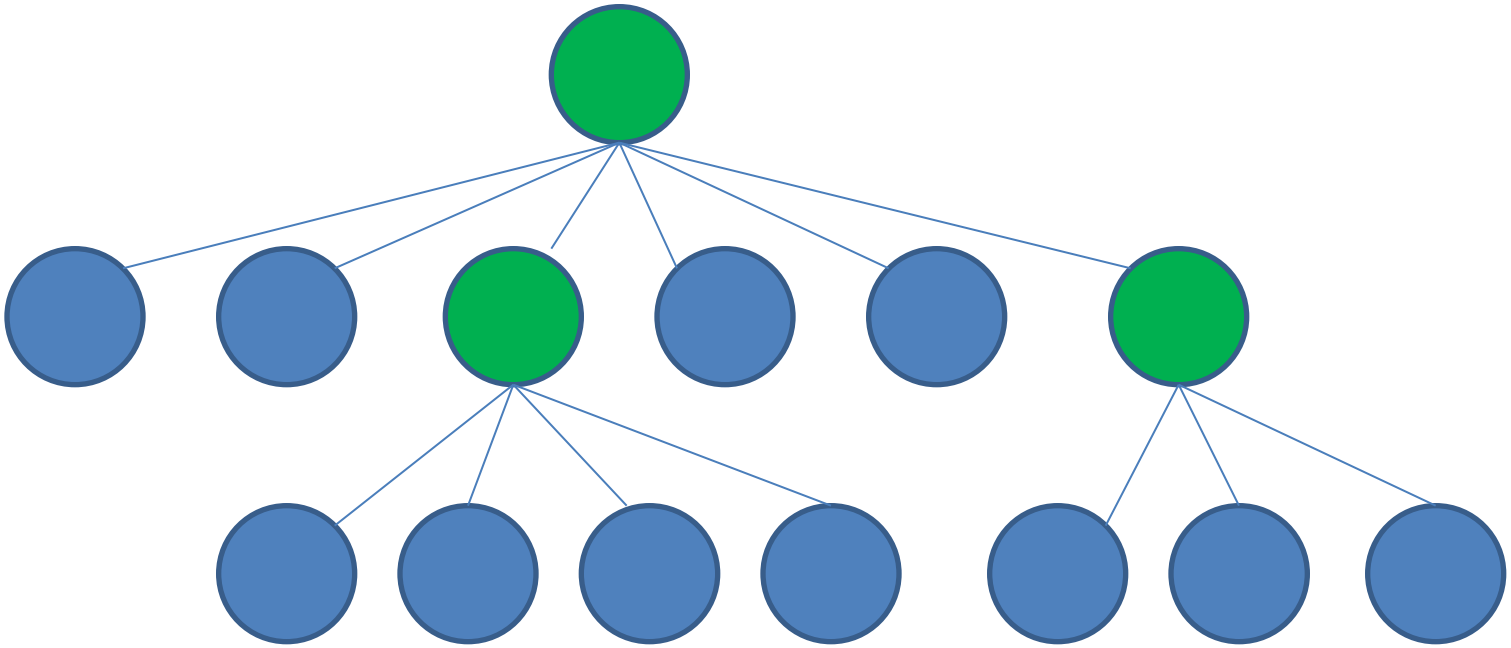
Najprej najboljši



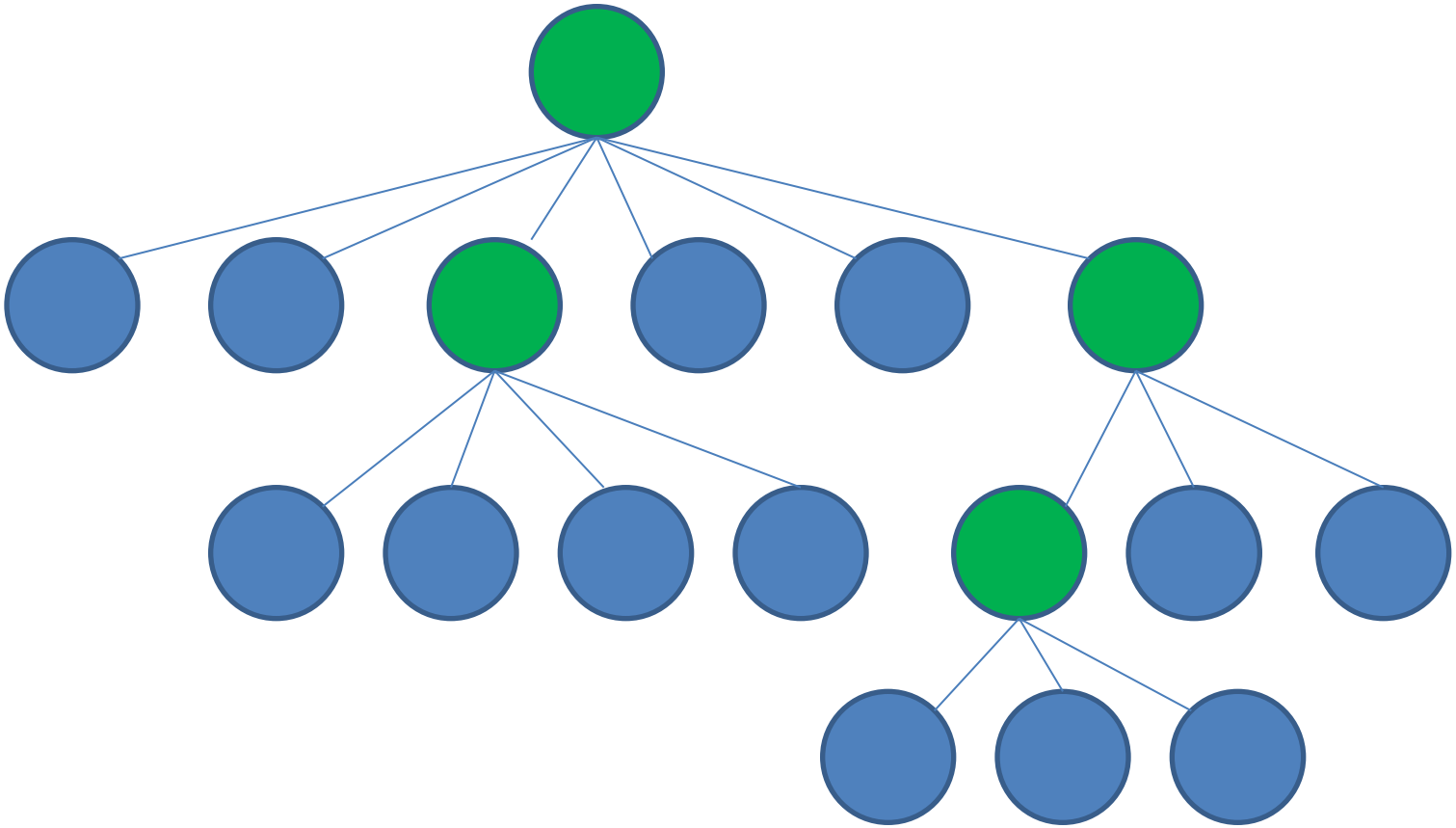
Najprej najboljši



Najprej najboljši

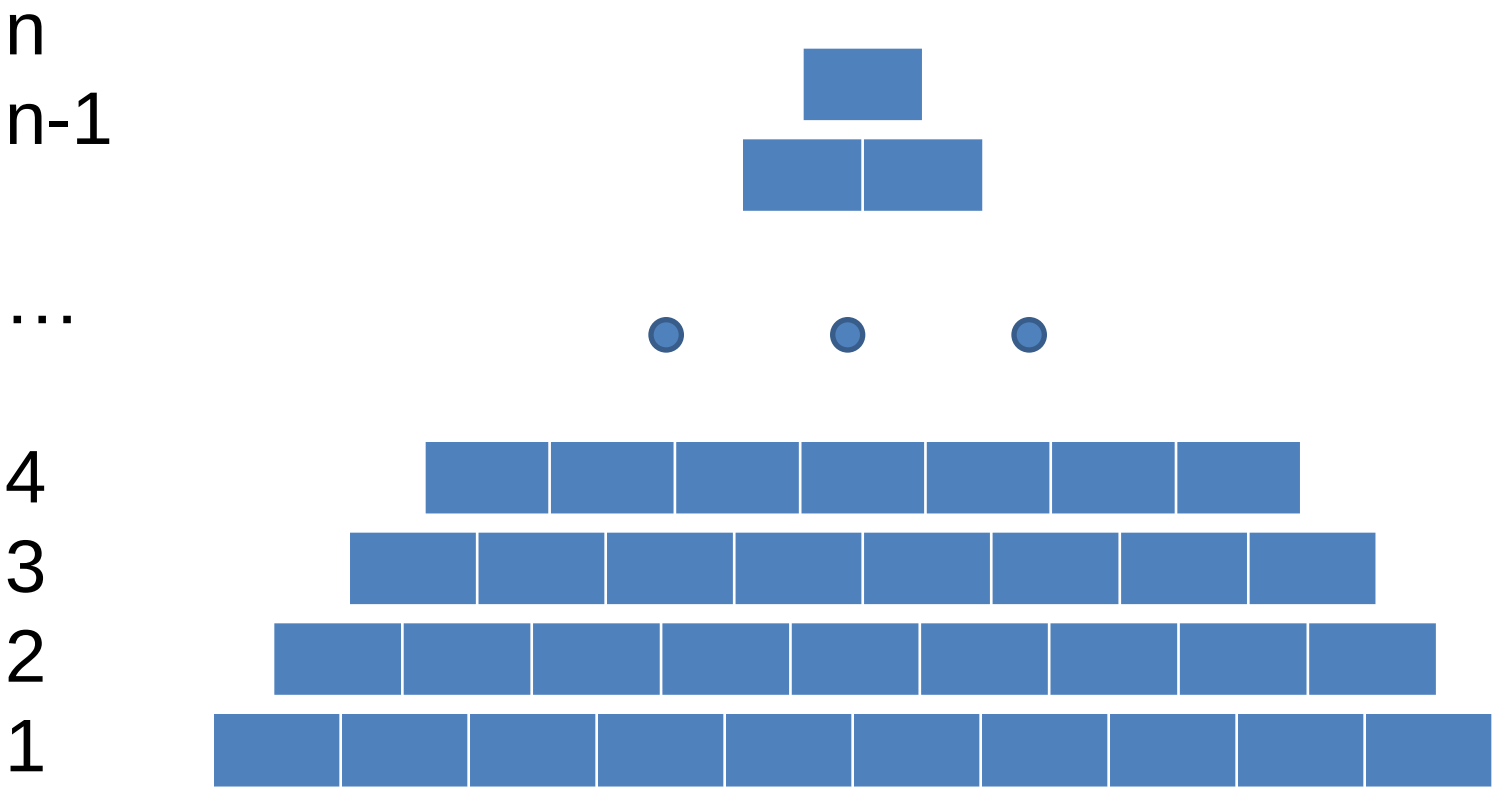


Najprej najboljši



+ V kombinaciji z “razveji in omeji” ena boljših strategij.
- Prostorska zahtevnost raste eksponentno!

Dinamično programiranje



Dinamično programiranje: IZZIV

Binomski koeficienti:

- reši rekurzivno (eksponentna časovna zahtevnost)
- reši z dinamičnim programiranjem (kvadratična časovna zahtevnost)

$$\binom{n}{n} = 1$$

$$\binom{n}{0} = 1$$

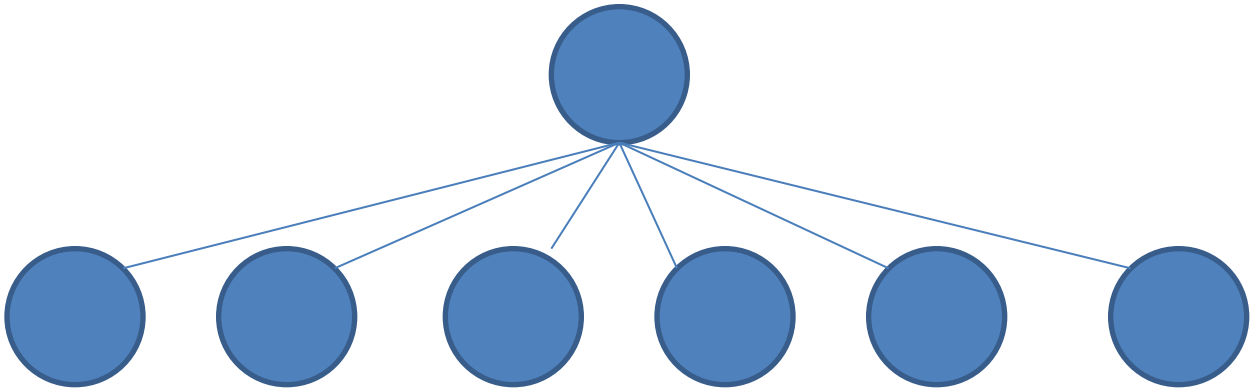
$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$$

Iskanje približne (suboptimalne) rešitve

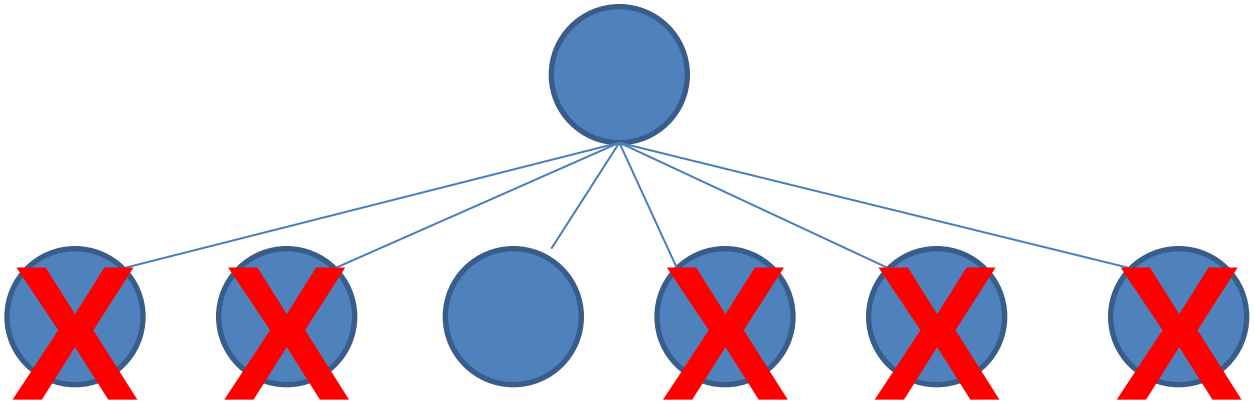
Osnovne preiskovalne strategije

1. Požrešno iskanje
   (greedy search, best-only search)
2. Iskanje v snopu (beam search)
3. Lokalna optimizacija
4. Gradientno iskanje
5. Stohastični algoritmi:
   a. Simulirano ohlajanje
   b. Genetski algoritmi

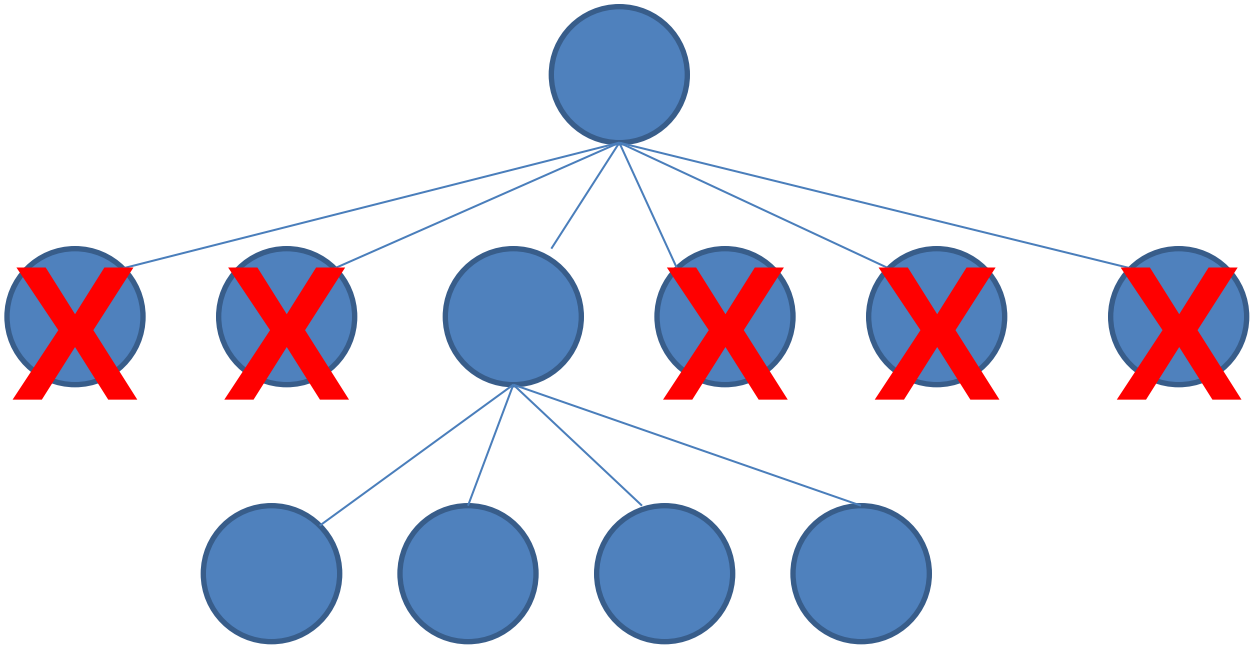
Požrešno iskanje



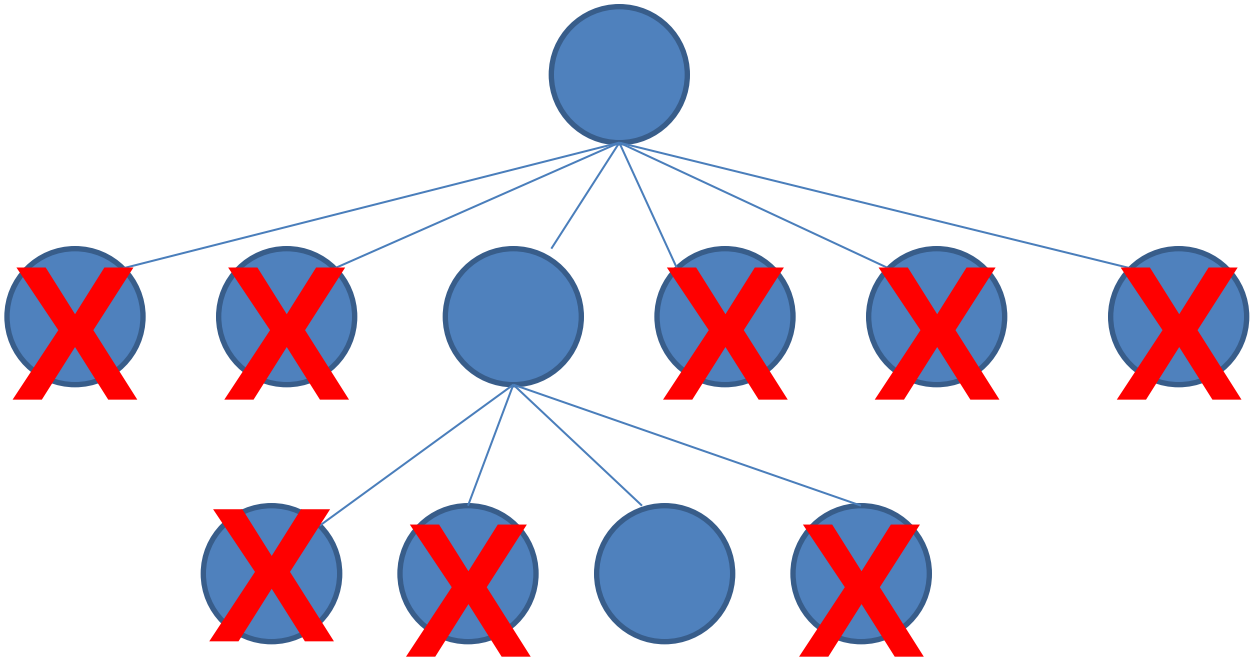
Požrešno iskanje



Požrešno iskanje

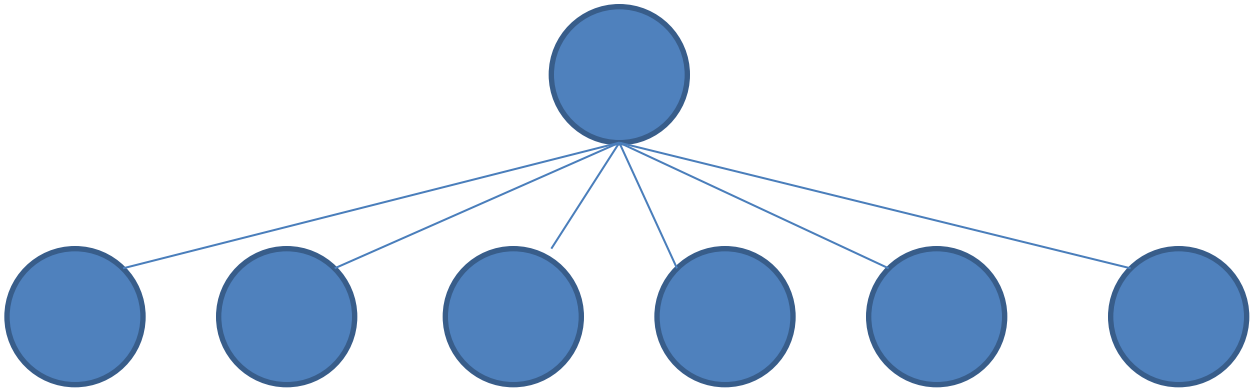


Požrešno iskanje

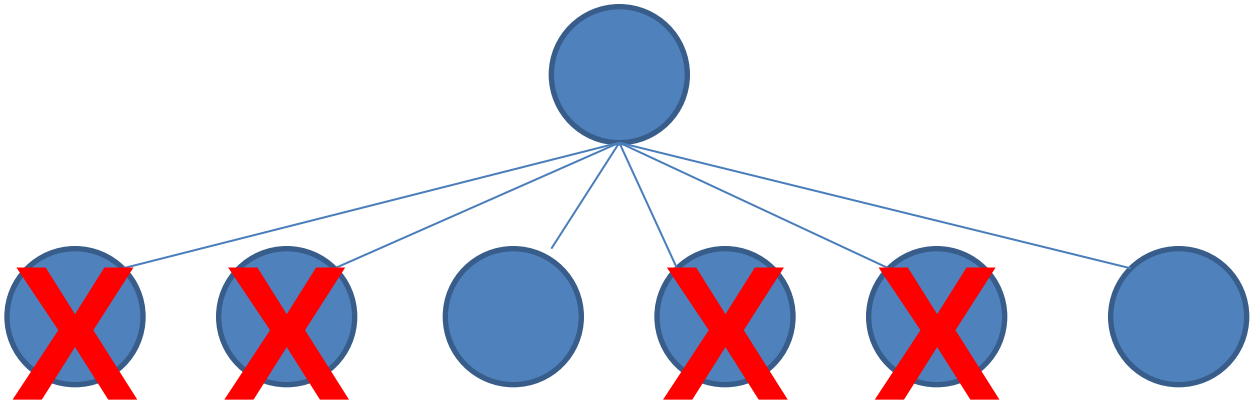


+ Izredno hitro napredujemo.
- Algoritem je “kratkoviden”, zato je rešitev samo približna.
+ V mnogih praktičnih problemih je rešitev zadovoljiva.

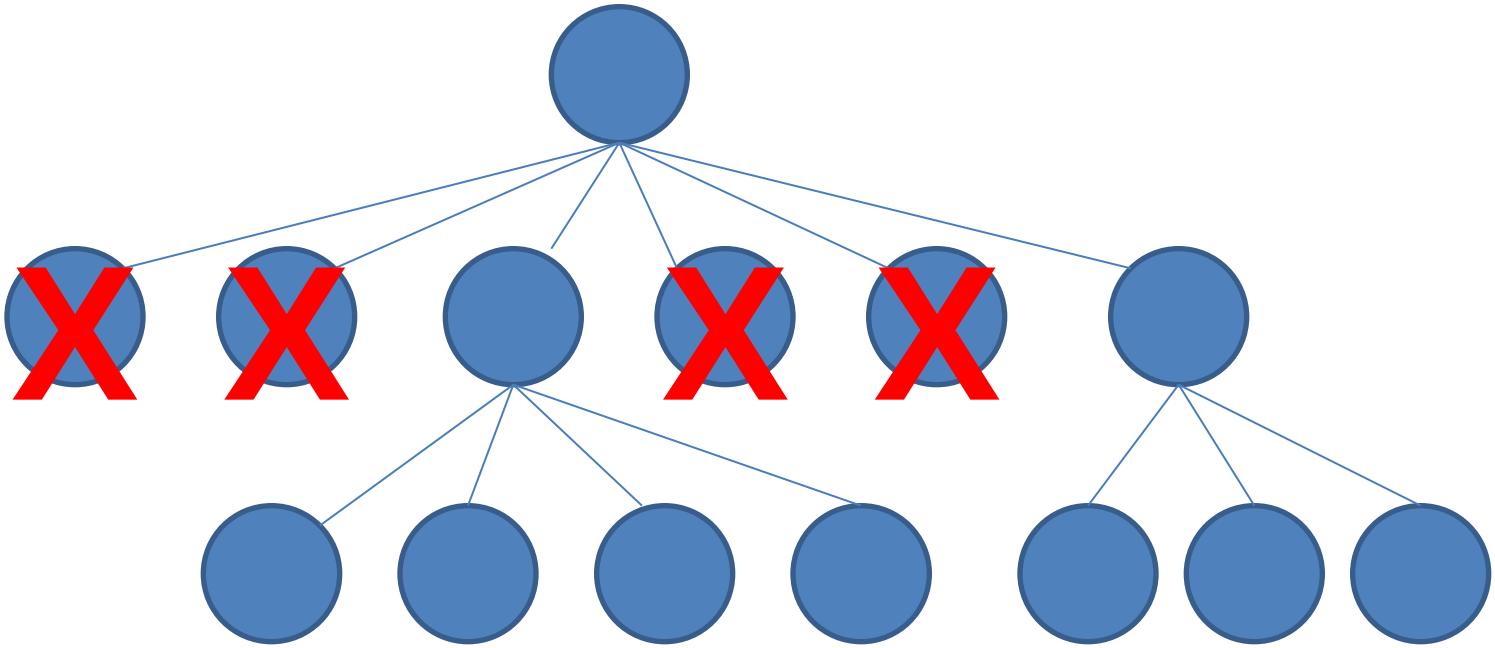
Iskanje v snopu



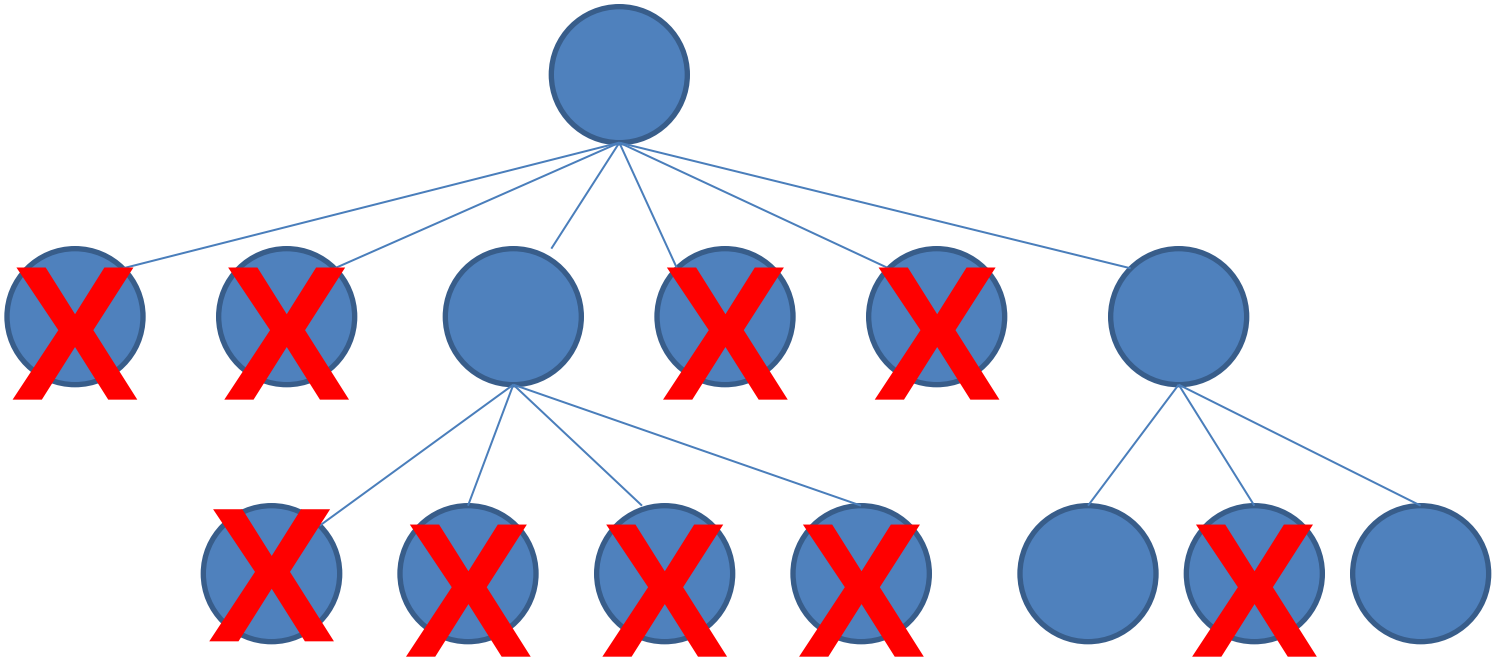
Iskanje v snopu



Iskanje v snopu



Iskanje v snopu



Posplošitev požrešnega iskanja:

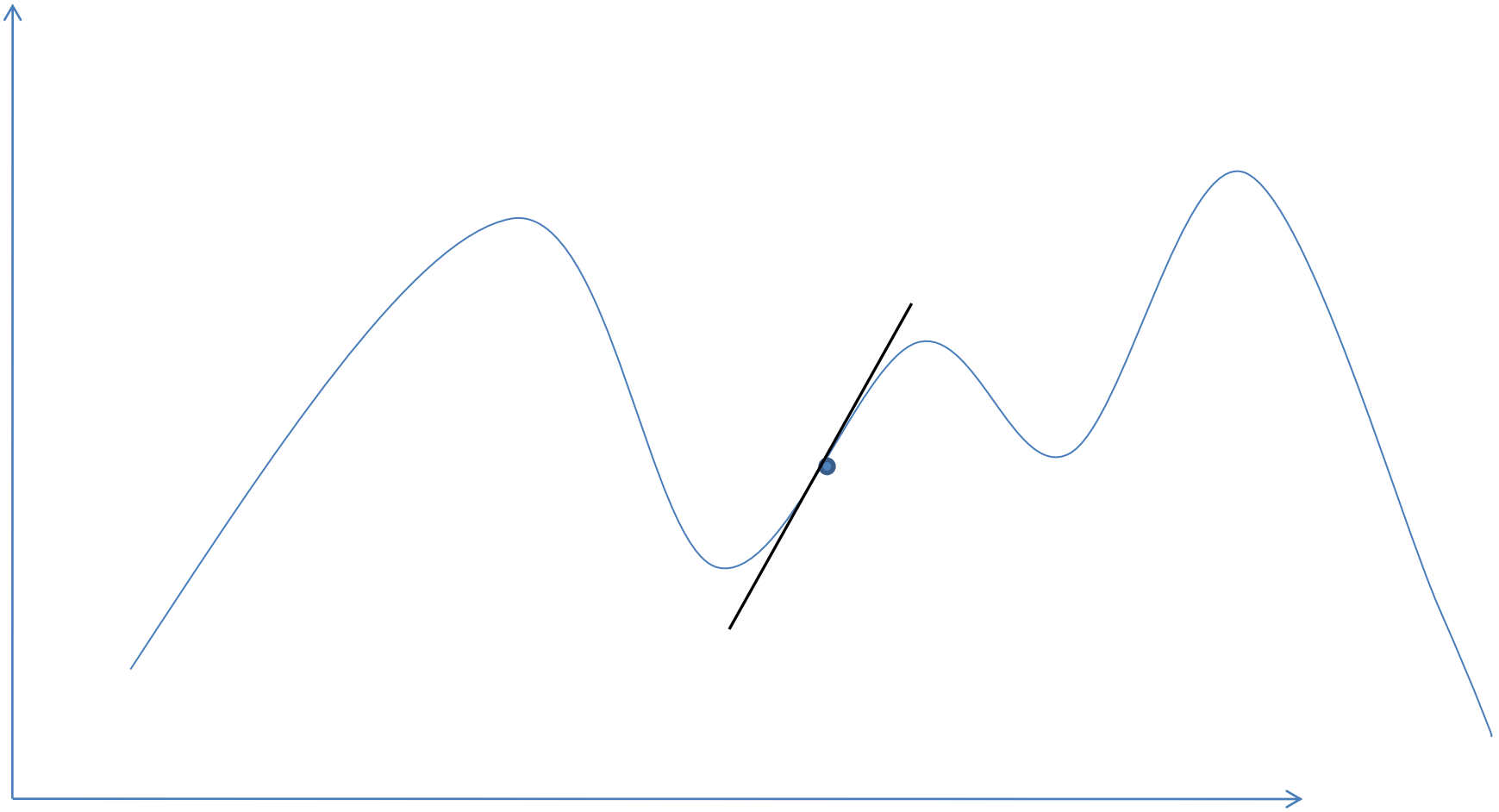
velikost snopa k = 1 → požrešno iskanje

Lokalna optimizacija

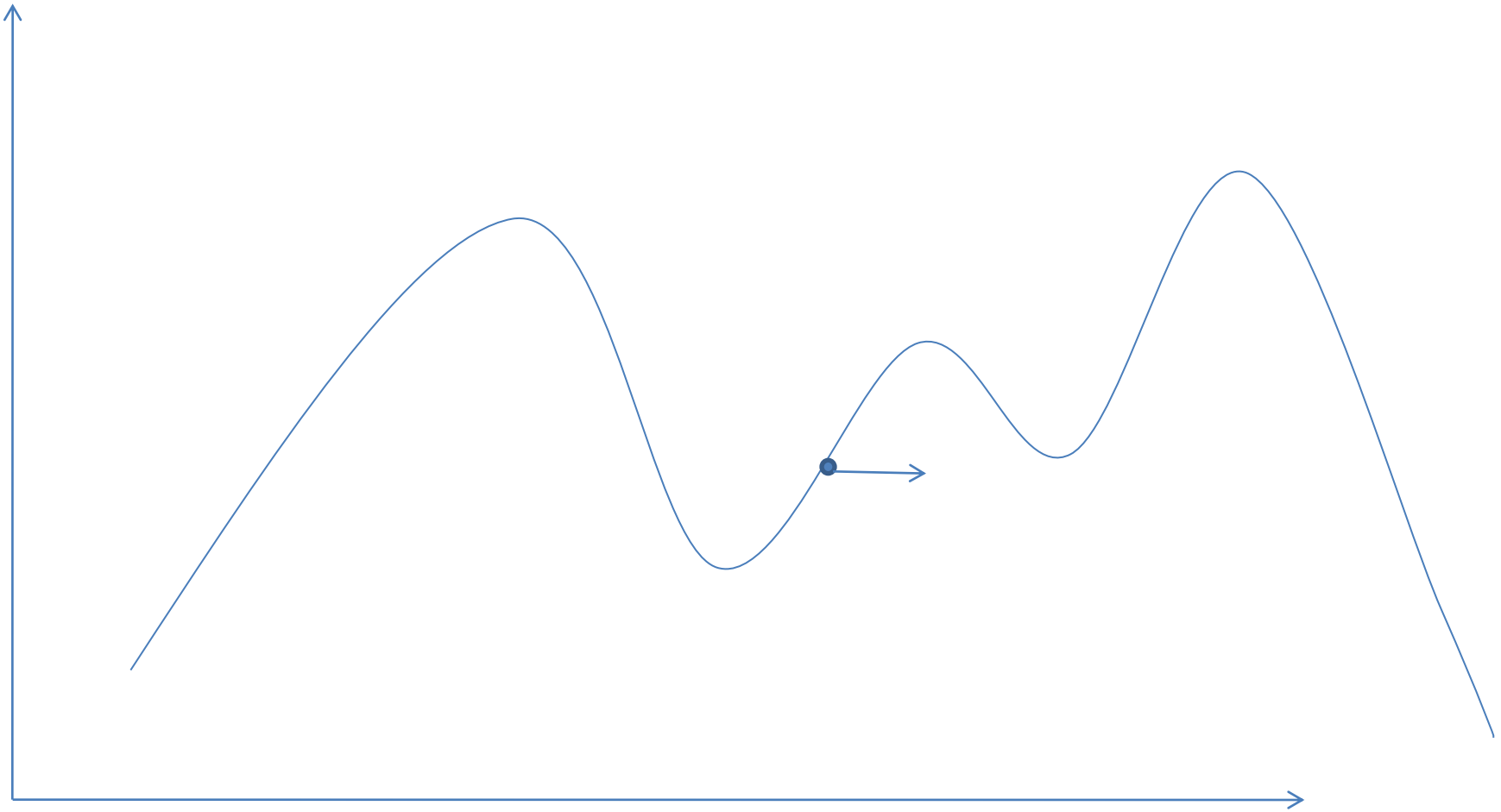
```
ponavljaj
    naključno izberi stanje (naključna rešitev);
    ponavljaj
        naredi najboljšo od možnih sprememb;
    dokler so še izboljšave;
dokler ni potekel dodeljen čas;
```

Je posplošitev požrešnega algoritma.
Večkrat ponoviš požrešno iskanje iz naključnega začetnega stanja.
+ Ena boljših strategij za preiskovanje velikih prostorov.

Gradientno iskanje

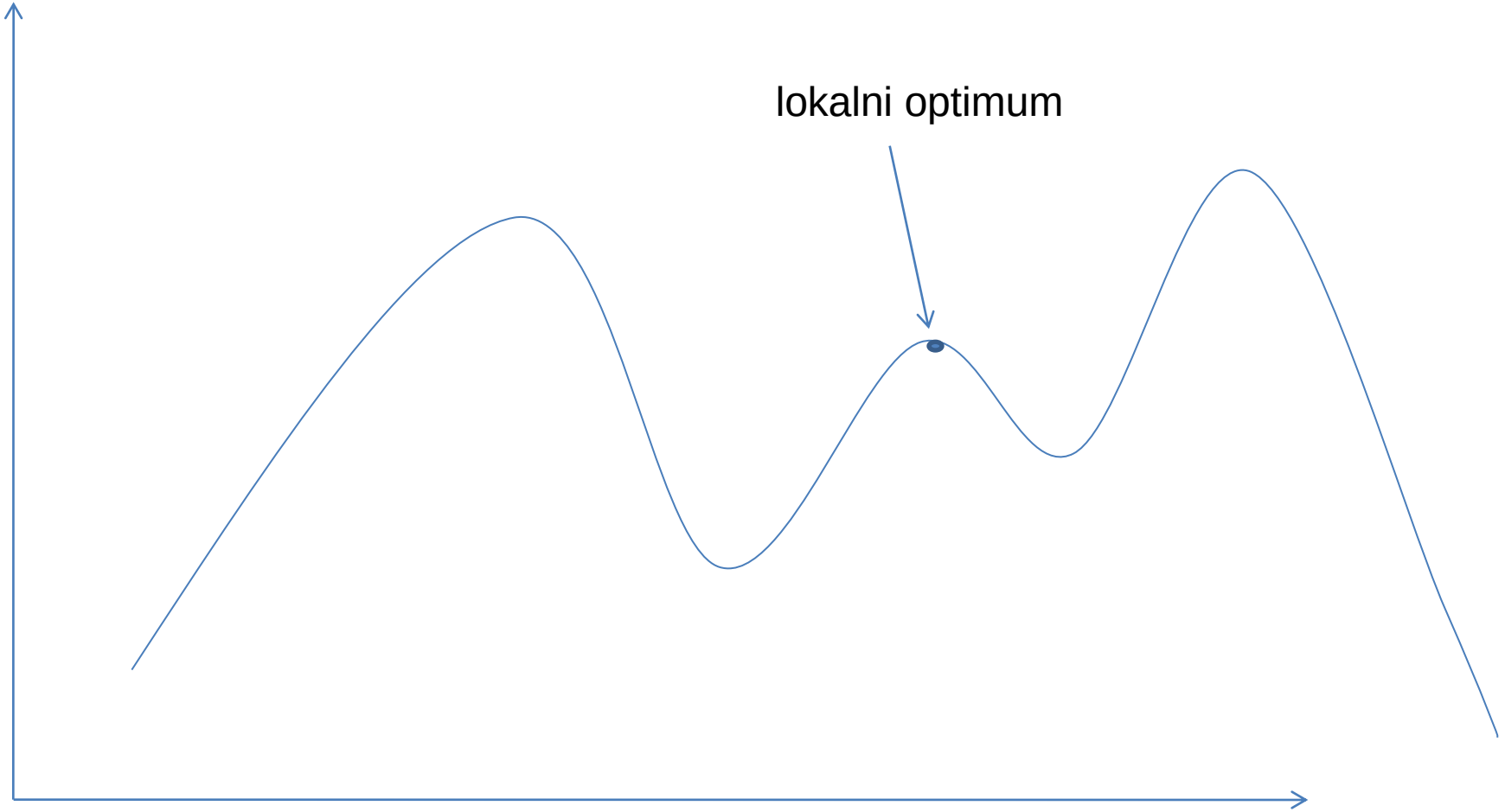


Gradientno iskanje



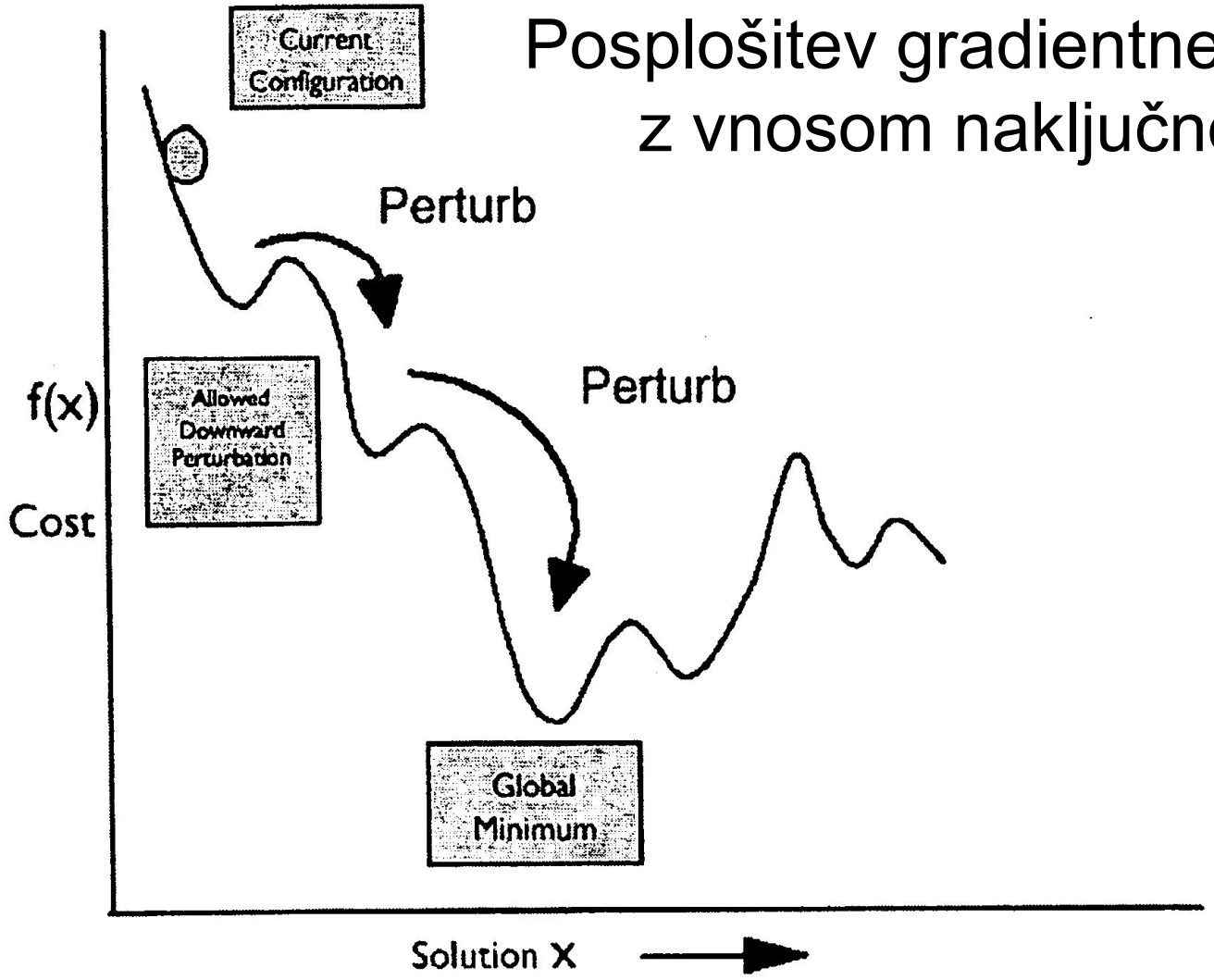
Gradientno iskanje

Je posplošitev požrešnega iskanja na zvezne prostore.
Spet kratkovidnost!



Simulirano ohlajanje

Posplošitev gradientnega iskanja z vnosom naključnosti.



Genetski algoritmi

Se zgledujejo po evoluciji: parjenje in boj za preživetje.
Dobri preživijo, slabi odmrejo. Najboljši imajo največ naslednikov.

Bistvo: izbira kodiranja.