Rješavanje problema pretraživanjem prostora stanja

predavanja iz Inteligentnih sustava

Zavod za elektroniku, mikroelektroniku, računalne i inteligentne sustave

Fakultet elektrotehnike i računarstva

Sveučilište u Zagrebu

Pretraživanje prostora stanja

- Općeniti koncept primjenljiv kod raznih razreda problema:
  - igre s jednim igračem (npr, konjićev skok)
  - igre s dva igrača (npr, šah)
  - optimizacijski problemi (putujući trgovac)
  - prospoj u VLSI sklopovima, autonomna navigacija . . .

Pretraživanje prostora stanja

- Općeniti koncept primjenljiv kod raznih razreda problema:
  - igre s jednim igračem (npr, konjićev skok)
  - igre s dva igrača (npr, šah)
  - optimizacijski problemi (putujući trgovac)
  - prospoj u VLSI sklopovima, autonomna navigacija . . .
- jedno od klasičnih poglavlja umjetne inteligencije
  - u početku se vjerovalo da se svi problemi mogu rješavati pretraživanjem: danas su ograničenja bolje poznata

Pretraživanje prostora stanja

- Općeniti koncept primjenljiv kod raznih razreda problema:
  - igre s jednim igračem (npr, konjićev skok)
  - igre s dva igrača (npr, šah)
  - optimizacijski problemi (putujući trgovac)
  - prospoj u VLSI sklopovima, autonomna navigacija . . .
- jedno od klasičnih poglavlja umjetne inteligencije
  - u početku se vjerovalo da se svi problemi mogu rješavati pretraživanjem: danas su ograničenja bolje poznata
- koncept je srodan pronalaženju najkraćeg puta u grafu; ovdje se međutim razmatra širi razred problema:
  - graf je **implicitno** zadan operatorima
  - graf je često praktično (ili stvarno) **beskonačan**

Sadržaj

- Rješavanje problema pretraživanjem prostora stanja
  - pretraživanje u kontekstu inteligentnog agenta
  - formalni opis koncepta pretraživanja
  - primjeri (didaktički, ozbiljniji, stvarni)

Sadržaj

- Rješavanje problema pretraživanjem prostora stanja
  - pretraživanje u kontekstu inteligentnog agenta
  - formalni opis koncepta pretraživanja
  - primjeri (didaktički, ozbiljniji, stvarni)
- Slijepo pretraživanje prostora stanja
  - opći postupak i strategije pretraživanja
  - neusmjerene strategije pretraživanja

Sadržaj

- Rješavanje problema pretraživanjem prostora stanja
  - pretraživanje u kontekstu inteligentnog agenta
  - formalni opis koncepta pretraživanja
  - primjeri (didaktički, ozbiljniji, stvarni)
- Slijepo pretraživanje prostora stanja
  - opći postupak i strategije pretraživanja
  - neusmjerene strategije pretraživanja
- Usmjereno pretraživanje
  - temeljni postupci (best-first, A*)
  - formuliranje heurističkih funkcija
  - pretraživanje s ograničenom memorijom (IDA*, SMA*)
  - postupci iterativnog poboljšanja (simulirano kaljenje)
- Problemi zadovoljenja ograničenja (CSP)

Širi kontekst pretraživanja

- Inteligentni sustav kao rješavač problema:
  - ustanovljivanje prioritetnog cilja
  - formalni opis problema (diskretna stanja, prijelazi, cilj)
  - **pronalaženje rješenja** (slijeda akcija koji vodi do cilja)

Širi kontekst pretraživanja

- Inteligentni sustav kao rješavač problema:
  - ustanovljivanje prioritetnog cilja
  - formalni opis problema (diskretna stanja, prijelazi, cilj)
  - **pronalaženje rješenja** (slijeda akcija koji vodi do cilja)
- ciklus "formuliraj – traži – djeluj":

```
# SA ... slijed akcija
while true:
  opažanje = promatrajSvijet()
  stanje = obnoviStanje(stanje, opažanje)
  if SA.prazan():
    cilj = ustanoviCilj(stanje)
    problem = formulirajProblem(stanje, cilj)
    SA = problem.traži()
  izvedi(SA.prvi(),stanje)
  SA.izbaciPrvog()
```

Formalni opis problema

- Ključni pojmovi (definiraju prostor stanja):
  - **model svijeta** – formalni prikaz svih relevantnih **stanja**
  - **model prijelaza** – dozvoljene akcije (potezi, operatori)
  - **model cilja** – ispitni predikat
  - **početno stanje** – mora biti poznato (dostupnost)

Formalni opis problema

- Ključni pojmovi (definiraju prostor stanja):
  - **model svijeta** – formalni prikaz svih relevantnih **stanja**
  - **model prijelaza** – dozvoljene akcije (potezi, operatori)
  - **model cilja** – ispitni predikat
  - **početno stanje** – mora biti poznato (dostupnost)
- pretraživanje je posebno prikladno za probleme koji su:
  - diskretni ili se mogu diskretizirati
  - dostupni (točno znamo gdje smo)
  - deterministični (ishodi akcija su neupitni)

Formalni opis problema

- Ključni pojmovi (definiraju prostor stanja):
  - **model svijeta** – formalni prikaz svih relevantnih **stanja**
  - **model prijelaza** – dozvoljene akcije (potezi, operatori)
  - **model cilja** – ispitni predikat
  - **početno stanje** – mora biti poznato (dostupnost)
- pretraživanje je posebno prikladno za probleme koji su:
  - diskretni ili se mogu diskretizirati
  - dostupni (točno znamo gdje smo)
  - deterministični (ishodi akcija su neupitni)
- koncept je srodan pronalaženju najkraćeg puta u grafu; ovdje se međutim razmatra širi razred problema:
  - graf je **implicitno** zadan operatorima
  - graf je često praktično (ili stvarno) **beskonačan**

Primjer: planiranje putovanja

- Situacija:
  - na turističkom smo obilasku Rumunjske, trenutno u Aradu
  - naš avion polijeće sutra iz Bukurešta

Primjer: planiranje putovanja

- Situacija:
  - na turističkom smo obilasku Rumunjske, trenutno u Aradu
  - naš avion polijeće sutra iz Bukurešta
- Cilj: doći u Bukurešt što kraćim putem

Primjer: planiranje putovanja

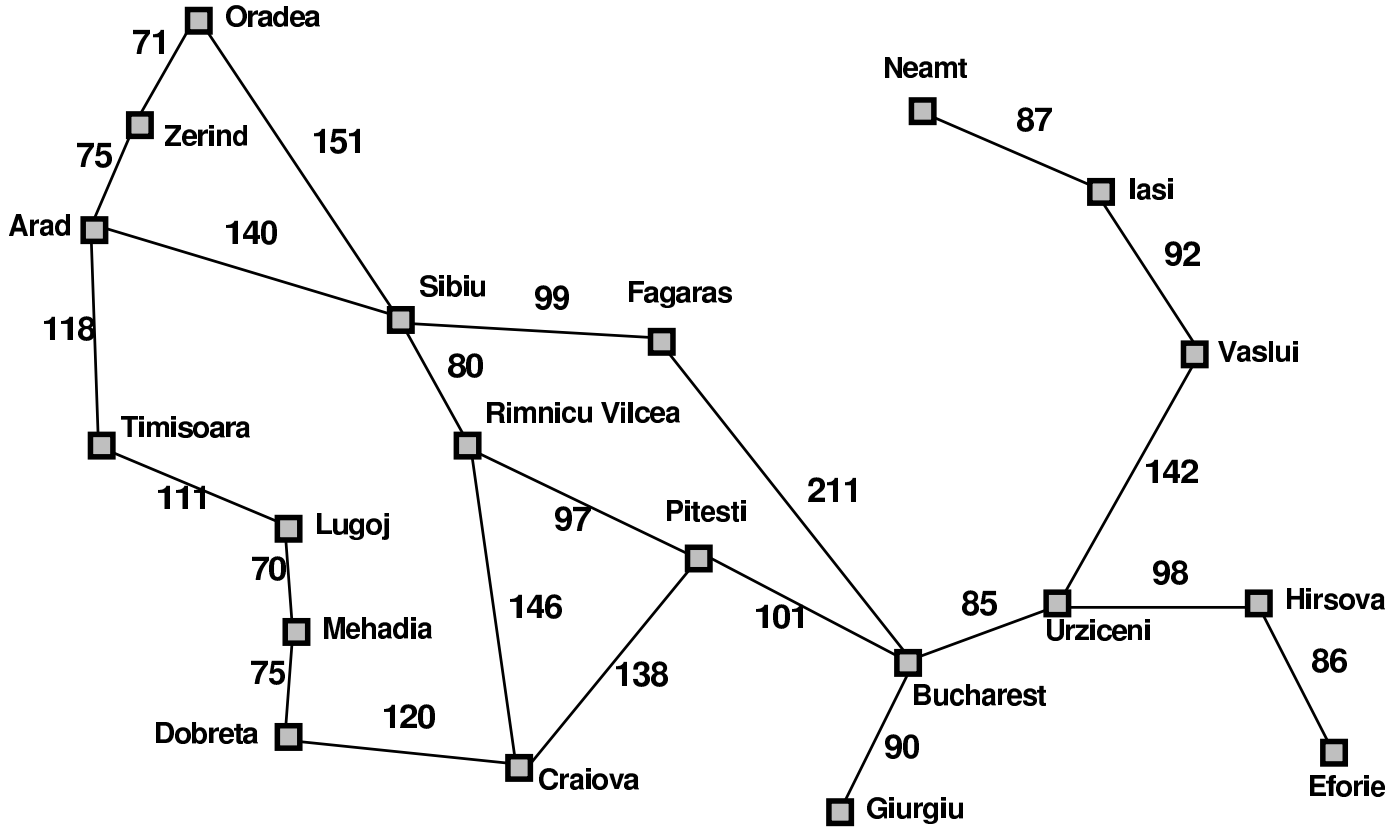
- Situacija:
  - na turističkom smo obilasku Rumunjske, trenutno u Aradu
  - naš avion polijeće sutra iz Bukurešta
- Cilj: doći u Bukurešt što kraćim putem
- Problem:
  - stanja svijeta: različiti gradovi (gdje smo trenutno?)
  - prijelazi: vožnja od grada do grada (što radimo?)
  - početno stanje: Arad
  - cilj: Bukurešt

Primjer: planiranje putovanja

- Situacija:
  - na turističkom smo obilasku Rumunjske, trenutno u Aradu
  - naš avion polijeće sutra iz Bukurešta
- Cilj: doći u Bukurešt što kraćim putem
- Problem:
  - stanja svijeta: različiti gradovi (gdje smo trenutno?)
  - prijelazi: vožnja od grada do grada (što radimo?)
  - početno stanje: Arad
  - cilj: Bukurešt
- Rješenje: niz gradova

Primjer: putovanje po Rumunjskoj

Romania with step costs in km



Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Formuliranje problema

- komponente formalnog opisa problema:

Formuliranje problema

- komponente formalnog opisa problema:
  - **početno stanje** s_0
  - **prijelazi** f definiraju:
    - sljedbenike tekućeg stanja $s_{ij} : s_i \longrightarrow s_{ij}, \forall j$
    - cijenu prijelaza $c(i, j)$
  - **prostor stanja** S: definiran sa s_0 i f
    (f inducira **graf** prostora stanja, koji je često ogroman)

Formuliranje problema

- komponente formalnog opisa problema:
  - **početno stanje** s_0
  - **prijelazi** f definiraju:
    - sljedbenike tekućeg stanja $s_{ij} : s_i \longrightarrow s_{ij}, \forall j$
    - cijenu prijelaza $c(i, j)$
  - **prostor stanja** S: definiran sa s_0 i f
    (f inducira **graf** prostora stanja, koji je često ogroman)
  - **ispitni predikat**: $p : S \longrightarrow \{\top, \bot\}$
    (ponekad se svodi na jednostavnu usporedbu!)

Kako uspoređivati probleme i postupke

- kriteriji za usporedbu formulacija problema:
  - $|S|$: broj članova prostora stanja
  - b: faktor grananja stabla pretraživanja
  - d: dubina optimalnog rješenja (s najmanjom cijenom puta)
  - m: najveća dubina stabla pretraživanja (može biti ∞)

Kako uspoređivati probleme i postupke

- kriteriji za usporedbu formulacija problema:
  - $|S|$: broj članova prostora stanja
  - b: faktor grananja stabla pretraživanja
  - d: dubina optimalnog rješenja (s najmanjom cijenom puta)
  - m: najveća dubina stabla pretraživanja (može biti ∞)
- kriteriji za vrednovanje postupaka:
  - da li postupak pronalazi rješenje? (potpunost)
  - da li je pronađeno rješenje najbolje? (dosežljivost)
  - cijena pretraživanja? (prostorna i vremenska složenost)

Oblikovanje prostora stanja

- stvaran svijet je **složen** do apsurda: vrlo važno je odlučiti što uvrstiti u formalni opis stanja i operatora, a što ne

Oblikovanje prostora stanja

- stvaran svijet je **složen** do apsurda: vrlo važno je odlučiti što uvrstiti u formalni opis stanja i operatora, a što ne
- **apstrakcija**: ignoriranje irelevantnih detalja iz stvarnog svijeta

Oblikovanje prostora stanja

- stvaran svijet je **složen** do apsurda: vrlo važno je odlučiti što uvrstiti u formalni opis stanja i operatora, a što ne
- **apstrakcija**: ignoriranje irelevantnih detalja iz stvarnog svijeta
- primjer: "Arad $\rightarrow$ Zerind" predstavlja sve moguće putove i zaobilaznice od Arada do Zerinda, uključujući i stanke za odmor
  $\Rightarrow$ zanemaruju se svi događaji koji su konceptualno ispod odabrane razine apstrakcije

Oblikovanje prostora stanja

- stvaran svijet je **složen** do apsurda: vrlo važno je odlučiti što uvrstiti u formalni opis stanja i operatora, a što ne
- **apstrakcija**: ignoriranje irelevantnih detalja iz stvarnog svijeta
- primjer: "Arad $\rightarrow$ Zerind" predstavlja sve moguće putove i zaobilaznice od Arada do Zerinda, uključujući i stanke za odmor
  $\Rightarrow$ zanemaruju se svi događaji koji su konceptualno ispod odabrane razine apstrakcije
- zanemaruju se i svi ostali detalji koji nisu relevantni u kontekstu prioritetnog cilja
  $\Rightarrow$ "nebitno": o kojem se automobilu radi, kakvo je društvo u automobilu, da li u autu svira radio, kakav krajolik se može vidjeti kroz prozor, da li ima policajaca u blizini, da li je vozač gladan ili je zaboravio natočiti gorivo...

Didaktički primjeri

- idealni slučajevi: koncizni i egzaktni; koriste se za ilustraciju ili vježbu različitih metodologija pretraživanja
- **Hanojski tornjevi** (n diskova)
  - stanje: n-torka brojeva $x_i \in \{0, 1, 2\}$
    npr, od najvećeg prema najmanjem: $(2, 1, 1, 1, 0)$

Didaktički primjeri

- idealni slučajevi: koncizni i egzaktni; koriste se za ilustraciju ili vježbu različitih metodologija pretraživanja
- **Hanojski tornjevi** (n diskova)
  - stanje: n-torka brojeva $x_i \in \{0, 1, 2\}$
    npr, od najvećeg prema najmanjem: $(2, 1, 1, 1, 0)$
  - $|S| = 3^n$

Didaktički primjeri

- idealni slučajevi: koncizni i egzaktni; koriste se za ilustraciju ili vježbu različitih metodologija pretraživanja
- **Hanojski tornjevi** (n diskova)
  - stanje: n-torka brojeva $x_i \in \{0, 1, 2\}$
    npr, od najvećeg prema najmanjem: $(2, 1, 1, 1, 0)$
  - $|S| = 3^n$
  - operator: $i \rightarrow j$, $i, j \in \{0, 1, 2\}$, $i \neq j$
    iz nekih stanja, neki potezi nisu dozvoljeni, npr: $0 \rightarrow 2$

Didaktički primjeri

- idealni slučajevi: koncizni i egzaktni; koriste se za ilustraciju ili vježbu različitih metodologija pretraživanja
- **Hanojski tornjevi** (n diskova)
  - stanje: n-torka brojeva $x_i \in \{0, 1, 2\}$
    npr, od najvećeg prema najmanjem: $(2, 1, 1, 1, 0)$
  - $|S| = 3^n$
  - operator: $i \rightarrow j$, $i, j \in \{0, 1, 2\}$, $i \neq j$
    iz nekih stanja, neki potezi nisu dozvoljeni, npr: $0 \rightarrow 2$
  - ispitni predikat: usporedba s jedinstvenim rješenjem

Didaktički primjeri

- idealni slučajevi: koncizni i egzaktni; koriste se za ilustraciju ili vježbu različitih metodologija pretraživanja
- **Hanojski tornjevi** (n diskova)
  - stanje: n-torka brojeva $x_i \in \{0, 1, 2\}$
    npr, od najvećeg prema najmanjem: $(2, 1, 1, 1, 0)$
  - $|S| = 3^n$
  - operator: $i \rightarrow j$, $i, j \in \{0, 1, 2\}$, $i \neq j$
    iz nekih stanja, neki potezi nisu dozvoljeni, npr: $0 \rightarrow 2$
  - ispitni predikat: usporedba s jedinstvenim rješenjem
  - cijena prijelaza: 1, za svaki potez
  - $b = 3$, $d = ?$

Didaktički primjeri

- **slagalica** $n \times n$
  - stanje: n^2-torka brojeva $x_i \in \{0, 1, \ldots, n^2 - 1\}$
    npr: $(1, 3, 0, 4, 8, 5, 6, 2, 7)$

Didaktički primjeri

- **slagalica** $n \times n$
  - stanje: n^2-torka brojeva $x_i \in \{0, 1, \ldots, n^2 - 1\}$
    npr: $(1, 3, 0, 4, 8, 5, 6, 2, 7)$
  - $|S| = (n^2)!$

Didaktički primjeri

- **slagalica** $n \times n$
  - stanje: n^2-torka brojeva $x_i \in \{0, 1, \ldots, n^2 - 1\}$
    npr: $(1, 3, 0, 4, 8, 5, 6, 2, 7)$
  - $|S| = (n^2)!$
  - operatori: potez lijevo, desno, gore ili dolje
    (nisu uvijek svi mogući)

Didaktički primjeri

- **slagalica** $n \times n$
  - stanje: n^2-torka brojeva $x_i \in \{0, 1, \ldots, n^2 - 1\}$
    npr: $(1, 3, 0, 4, 8, 5, 6, 2, 7)$
  - $|S| = (n^2)!$
  - operatori: potez lijevo, desno, gore ili dolje
    (nisu uvijek svi mogući)
  - ispitni predikat: usporedba s jedinstvenim rješenjem

Didaktički primjeri

- **slagalica** $n \times n$
  - stanje: n^2-torka brojeva $x_i \in \{0, 1, \ldots, n^2 - 1\}$
    npr: $(1, 3, 0, 4, 8, 5, 6, 2, 7)$
  - $|S| = (n^2)!$
  - operatori: potez lijevo, desno, gore ili dolje
    (nisu uvijek svi mogući)
  - ispitni predikat: usporedba s jedinstvenim rješenjem
  - cijena prijelaza: 1
  - $b \in [2, 4]$, $d = ?$

Didaktički primjeri

- **n kraljica na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n$)
    $s{=}(x_i)$, $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 4×4: (0:2, 2:1)

Didaktički primjeri

- **n kraljica na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n$)
    $s{=}(x_i)$, $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 4×4: (0:2, 2:1)
  - $|S| = \binom{n^2}{n}$

Didaktički primjeri

- **n kraljica na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n$)
    $s{=}(x_i)$, $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 4×4: (0:2, 2:1)
  - $|S| = \binom{n^2}{n}$
  - operator: postavljanje nove kraljice na ploču
    $x \in \{0, 1, \ldots, n-1\}^2$: $\forall i\ x \neq x_i$, neNapada(x,x_i)

Didaktički primjeri

- **n kraljica na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n$)
    s=(x_i), $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 4×4: (0:2, 2:1)
  - $|S| = \binom{n^2}{n}$
  - operator: postavljanje nove kraljice na ploču
    $x \in \{0, 1, \ldots, n-1\}^2$: $\forall i \; x \neq x_i$, neNapada(x,x_i)
  - ispitni predikat: vrijedi li: $k = n$?

Didaktički primjeri

- **n kraljica na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n$)
    s=(x_i), $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 4×4: (0:2, 2:1)
  - $|S| = \binom{n^2}{n}$
  - operator: postavljanje nove kraljice na ploču
    $x \in \{0, 1, \ldots, n-1\}^2$: $\forall i\ x \neq x_i$, neNapada(x,x_i)
  - ispitni predikat: vrijedi li: $k = n$?
  - cijena prijelaza: nije bitna!
    (nije nas briga kojim putem smo došli do rješenja)
    (analogija: razmještanje komponenata po čipu)
  - $b \in [1, n^2]$, $d = n$

Didaktički primjeri

- **konjićev skok na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n^2$)
    $s{=}(x_i)$, $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 8×8: (0:0, 1:2, 2:4)

Didaktički primjeri

- **konjićev skok na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n^2$)
    $s{=}(x_i)$, $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 8×8: (0:0, 1:2, 2:4)
  - $|S| \approx 8^{(n^2)}$ (najgori slučaj)

Didaktički primjeri

- **konjićev skok na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n^2$)
    $s{=}(x_i)$, $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 8×8: (0:0, 1:2, 2:4)
  - $|S| \approx 8^{(n^2)}$ (najgori slučaj)
  - operator: skok na do sada neposjećeno polje ploče

Didaktički primjeri

- **konjićev skok na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n^2$)
    $s{=}(x_i)$, $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 8×8: (0:0, 1:2, 2:4)
  - $|S| \approx 8^{(n^2)}$ (najgori slučaj)
  - operator: skok na do sada neposjećeno polje ploče
  - ispitni predikat: jesmo li posjetili sva polja?
    ekvivalentno: $k = n^2$?

Didaktički primjeri

- **konjićev skok na ploči** $n \times n$
  - stanje: k-torka parova (u ciljnom stanju, $k = n^2$)
    $s{=}(x_i)$, $x_i \in \{0, 1, \ldots, n-1\}^2$
    npr, za 8×8: (0:0, 1:2, 2:4)
  - $|S| \approx 8^{(n^2)}$ (najgori slučaj)
  - operator: skok na do sada neposjećeno polje ploče
  - ispitni predikat: jesmo li posjetili sva polja?
    ekvivalentno: $k = n^2$?
  - cijena prijelaza: nije bitna!
  - $b \in [1, 8]$, $d = n^2$

Ozbiljniji problem

- **sokoban** (1)
  - stanje: položaji “dijamanata”+ položaj skladištara

Ozbiljniji problem

- **sokoban** (1)
  - stanje: položaji "dijamanata"+ položaj skladištara
  - $|S| = \binom{n}{k+1}$

    n – broj slobodnih mjesta

    k – broj dijamanata

Ozbiljniji problem

- **sokoban** (1)
  - stanje: položaji "dijamanata"+ položaj skladištara
  - $|S| = \binom{n}{k+1}$
    n – broj slobodnih mjesta
    k – broj dijamanata
  - operatori: potez lijevo, desno, gore ili dolje
    (nisu uvijek svi mogući)

Ozbiljniji problem

- **sokoban** (1)
  - stanje: položaji "dijamanata"+ položaj skladištara
  - $|S| = \binom{n}{k+1}$
    n – broj slobodnih mjesta
    k – broj dijamanata
  - operatori: potez lijevo, desno, gore ili dolje
    (nisu uvijek svi mogući)
  - ispitni predikat:
    da li je svaki dijamant na ciljnom mjestu?

Ozbiljniji problem

- **sokoban** (1)
  - stanje: položaji "dijamanata"+ položaj skladištara
  - $|S| = \binom{n}{k+1}$
    n – broj slobodnih mjesta
    k – broj dijamanata
  - operatori: potez lijevo, desno, gore ili dolje
    (nisu uvijek svi mogući)
  - ispitni predikat:
    da li je svaki dijamant na ciljnom mjestu?
  - cijena prijelaza: uvijek 1, ili 1 samo za pomak dijamanta

Ozbiljniji problem

- **sokoban** (2)
  - stanje: položaji “dijamanata”+ okvirni položaj skladištara

Ozbiljniji problem

- **sokoban** (2)
  - stanje: položaji "dijamanata"+ okvirni položaj skladištara
  - $|S| = \binom{n}{k+1}$ (u najgorem slučaju)
    n – broj slobodnih mjesta
    k – broj dijamanata

Ozbiljniji problem

- **sokoban** (2)
  - stanje: položaji "dijamanata"+ okvirni položaj skladištara
  - $|S| = \binom{n}{k+1}$ (u najgorem slučaju)
    n – broj slobodnih mjesta
    k – broj dijamanata
  - operatori: pomak bilo kojeg od dijamanata lijevo, desno, gore ili dolje

Ozbiljniji problem

- **sokoban** (2)
  - stanje: položaji "dijamanata"+ okvirni položaj skladištara
  - $|S| = \binom{n}{k+1}$ (u najgorem slučaju)
    n – broj slobodnih mjesta
    k – broj dijamanata
  - operatori: pomak bilo kojeg od dijamanata lijevo, desno, gore ili dolje
  - ispitni predikat:
    da li je svaki dijamant na ciljnom mjestu?

Ozbiljniji problem

- **sokoban** (2)
  - stanje: položaji "dijamanata"+ okvirni položaj skladištara
  - $|S| = \binom{n}{k+1}$ (u najgorem slučaju)
    n – broj slobodnih mjesta
    k – broj dijamanata
  - operatori: pomak bilo kojeg od dijamanata lijevo, desno, gore ili dolje
  - ispitni predikat:
    da li je svaki dijamant na ciljnom mjestu?
  - cijena prijelaza: 1
  - $b \approx k * 4, d \approx 100$

    Andreas Junghanns and Jonathan Schaeffer. Sokoban: Enhancing General Single-Agent Search Methods Using Domain Knowledge, Artificial Intelligence, vol. 129, no. 1-2, pp. 219-251, 2001.

Stvarni problemi

- za razliku od didaktičkih problema, prvenstveno nas interesira rješenje

Stvarni problemi

- za razliku od didaktičkih problema, prvenstveno nas interesira rješenje
- problem: obično ima više različitih modela, nije uvijek jasno koji je bolji

Stvarni problemi

- za razliku od didaktičkih problema, prvenstveno nas interesira rješenje
- problem: obično ima više različitih modela, nije uvijek jasno koji je bolji
- primjeri:
  - traženje optimalnog puta (transport robe)
  - rješenja problema putujućeg trgovca (transport robe, upravljanje strojevima)

Stvarni problemi

- za razliku od didaktičkih problema, prvenstveno nas interesira rješenje
- problem: obično ima više različitih modela, nije uvijek jasno koji je bolji
- primjeri:
  - traženje optimalnog puta (transport robe)
  - rješenja problema putujućeg trgovca (transport robe, upravljanje strojevima)
  - izvedba prospoja u VLSI sklopovima
  - navigacija samostalnih robota

Stvarni problemi

- za razliku od didaktičkih problema, prvenstveno nas interesira rješenje
- problem: obično ima više različitih modela, nije uvijek jasno koji je bolji
- primjeri:
  - traženje optimalnog puta (transport robe)
  - rješenja problema putujućeg trgovca (transport robe, upravljanje strojevima)
  - izvedba prospoja u VLSI sklopovima
  - navigacija samostalnih robota
  - predviđanje strukture proteina (farmaceutska industrija, biotehnologija)
  - pretraživanje interneta

Opći postupak pretraživanja

- temeljna ideja: (simulirano) istraživanje grafa iterativnim širenjem sljedbenika posjećenih stanja

Opći postupak pretraživanja

- temeljna ideja: (simulirano) istraživanje grafa iterativnim širenjem sljedbenika posjećenih stanja
- **stablo pretraživanja**: podskup ukupnog grafa prostora stanja koji je otkriven do tekućeg trenutka

Opći postupak pretraživanja

- temeljna ideja: (simulirano) istraživanje grafa iterativnim širenjem sljedbenika posjećenih stanja
- **stablo pretraživanja**: podskup ukupnog grafa prostora stanja koji je otkriven do tekućeg trenutka
- **proširivanje stanja**: otvaranje (pronalaženje) svih sljedbenika tekućeg stanja u skladu s operatorima

Opći postupak pretraživanja

- temeljna ideja: (simulirano) istraživanje grafa iterativnim širenjem sljedbenika posjećenih stanja
- **stablo pretraživanja**: podskup ukupnog grafa prostora stanja koji je otkriven do tekućeg trenutka
- **proširivanje stanja**: otvaranje (pronalaženje) svih sljedbenika tekućeg stanja u skladu s operatorima
- **otvorena stanja**: skup svih posjećenih stanja koja još nisu proširena

Opći postupak pretraživanja

- temeljna ideja: (simulirano) istraživanje grafa iterativnim širenjem sljedbenika posjećenih stanja
- **stablo pretraživanja**: podskup ukupnog grafa prostora stanja koji je otkriven do tekućeg trenutka
- **proširivanje stanja**: otvaranje (pronalaženje) svih sljedbenika tekućeg stanja u skladu s operatorima
- **otvorena stanja**: skup svih posjećenih stanja koja još nisu proširena
- **zatvorena stanja**: skup svih već proširenih stanja

Primjer pretraživanja

- proširivanje stanja tijekom pretraživanja na primjeru planiranja puta do Bukurešta

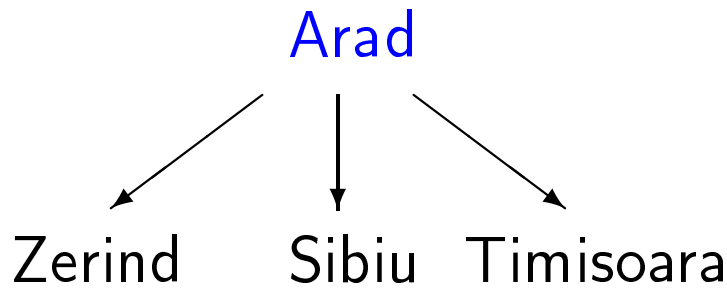
Primjer pretraživanja

- proširivanje stanja tijekom pretraživanja na primjeru planiranja puta do Bukurešta

Arad

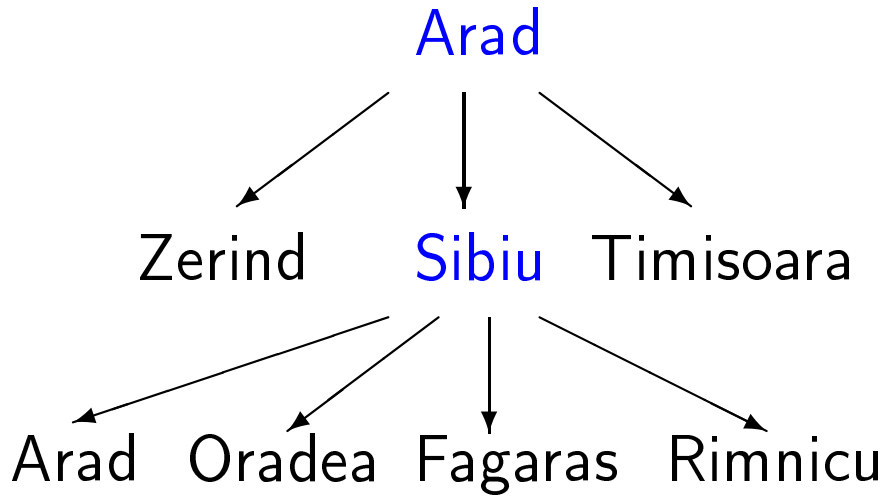
Primjer pretraživanja

- proširivanje stanja tijekom pretraživanja na primjeru planiranja puta do Bukurešta



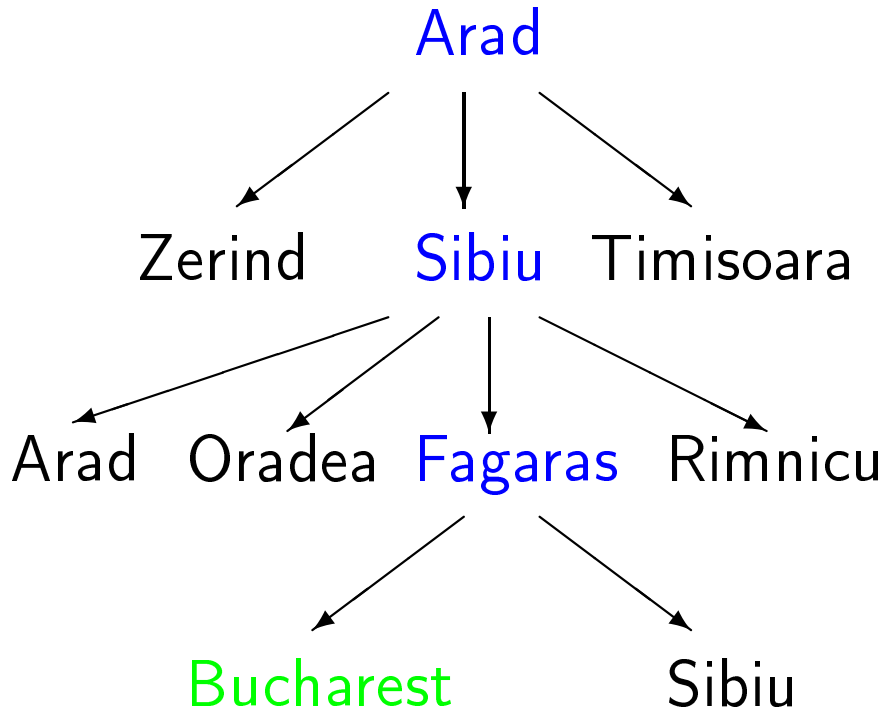
Primjer pretraživanja

- proširivanje stanja tijekom pretraživanja na primjeru planiranja puta do Bukurešta



Primjer pretraživanja

- proširivanje stanja tijekom pretraživanja na primjeru planiranja puta do Bukurešta



Opći postupak pretraživanja

- za probleme kod kojih u grafu prostora stanja nema ciklusa, nije potrebno održavati listu zatvorenih stanja:

Opći postupak pretraživanja

- za probleme kod kojih u grafu prostora stanja nema ciklusa, nije potrebno održavati listu zatvorenih stanja:

```
# funkcija vraća rješenje ili neuspjeh
# O ... prioritetni red otvorenih čvorova
# S ... strategija otvaranja novih čvorova
def općePretraživanje(problem, S):
  O.dodaj(problem.početnoStanje())
  while !O.prazan():
    s = O.izbaciPrvog()
    if (problem.ispitajCilj(s)):
      return rješenje

    for s_i in problem.proširi(s):
      O.dodaj(s_i, S.prioritet(s))

  return neuspjeh
```

Opći postupak pretraživanja

- za probleme sa ciklusima (npr, ako su operatori reverzibilni), lista zatvorenih stanja može biti nužno zlo:

Opći postupak pretraživanja

- za probleme sa ciklusima (npr, ako su operatori reverzibilni), lista zatvorenih stanja može biti nužno zlo:

```
# O ... prioritetni red otvorenih čvorova
# C ... hash tablica zatvorenih čvorova
# S ... strategija otvaranja novih čvorova
def općePretraživanje(problem, S):
  O.dodaj(problem.početnoStanje())
  while !O.prazan():
    s = O.izbaciPrvog()
    if problem.ispitajCilj(s):
      return rješenje

    C.dodaj(s)
    for s_i in problem.proširi(s):
      if s_i ∉ C:
        O.dodaj(s_i, S.prioritet(s))

  return neuspjeh
```

Strategije pretraživanja

- **strategija** pretraživanja definira redosljed proširivanja otvorenih čvorova

Strategije pretraživanja

- **strategija** pretraživanja definira redosljed proširivanja otvorenih čvorova
- primjerena strategija je kritična za performansu postupka pretraživanja!

Strategije pretraživanja

- **strategija** pretraživanja definira redosljed proširivanja otvorenih čvorova
- primjerena strategija je kritična za performansu postupka pretraživanja!
- kriteriji za vrednovanje strategija pretraživanja:
  - potpunost,
  - vremenska složenost,
  - prostorna složenost,
  - dosežljivost

Strategije pretraživanja

- **strategija** pretraživanja definira redosljed proširivanja otvorenih čvorova
- primjerena strategija je kritična za performansu postupka pretraživanja!
- kriteriji za vrednovanje strategija pretraživanja:
  - potpunost,
  - vremenska složenost,
  - prostorna složenost,
  - dosežljivost
- parametri za ocjenu vremenske i prostorne složenosti:
  - b: faktor grananja (prosječni ili najveći)
  - d: dubina optimalnog rješenja
  - m: najveća dubina stabla pretraživanja (može biti ∞)

Neusmjerene (slijepe) strategije

- pretraživanje u širinu

Neusmjerene (slijepe) strategije

- pretraživanje u širinu
- pretraživanje s jednolikom cijenom

Neusmjerene (slijepe) strategije

- pretraživanje u širinu
- pretraživanje s jednolikom cijenom
- pretraživanje u dubinu

Neusmjerene (slijepe) strategije

- pretraživanje u širinu
- pretraživanje s jednolikom cijenom
- pretraživanje u dubinu
- ograničeno pretraživanje u dubinu

Neusmjerene (slijepe) strategije

- pretraživanje u širinu
- pretraživanje s jednolikom cijenom
- pretraživanje u dubinu
- ograničeno pretraživanje u dubinu
- iterativno pretraživanje u dubinu

Neusmjerene (slijepe) strategije

- pretraživanje u širinu
- pretraživanje s jednolikom cijenom
- pretraživanje u dubinu
- ograničeno pretraživanje u dubinu
- iterativno pretraživanje u dubinu
- bidirekcionalno pretraživanje

Pretraživanje u širinu

- implementacija: `S.prioritet(x):=dubina(x)`
  (monotono raste za svaki novi poziv, neovisno o x)
  (nova stanja se umeću na **kraj** liste O)

Pretraživanje u širinu

- implementacija: `S.prioritet(x):=dubina(x)`
  (monotono raste za svaki novi poziv, neovisno o x)
  (nova stanja se umeću na **kraj** liste O)
- u svakom koraku, proširuje se najplići otvoreni čvor

Pretraživanje u širinu

- implementacija: `S.prioritet(x):=dubina(x)`
  (monotono raste za svaki novi poziv, neovisno o x)
  (nova stanja se umeću na **kraj** liste O)
- u svakom koraku, proširuje se najplići otvoreni čvor
- - potpunost: da (ako je b konačan)
  - vremenska složenost: $\sum_{i=0}^{d} b^i$=O(b^d)
  - prostorna složenost: O(b^d)
    (čuva se svaki čvor na posljednjoj dubini)
  - dosežljivost: da, ali samo ako je cijena prijelaza konstantna

Pretraživanje u širinu

- implementacija: `S.prioritet(x):=dubina(x)`
  (monotono raste za svaki novi poziv, neovisno o x)
  (nova stanja se umeću na **kraj** liste O)
- u svakom koraku, proširuje se najplići otvoreni čvor
- - potpunost: da (ako je b konačan)
  - vremenska složenost: $\sum_{i=0}^{d} b^i$=O(b^d)
  - prostorna složenost: O(b^d)
    (čuva se svaki čvor na posljednjoj dubini)
  - dosežljivost: da, ali samo ako je cijena prijelaza konstantna
- glavni problem je prostor!
  $b = 4$, $d = 16$, 100byte/stanje $\Rightarrow$ 429 Gb!!!

Primjer pretraživanja u širinu

- proširivanje stanja pri pretraživanju prostora u širinu za primjer planiranja puta do Bukurešta

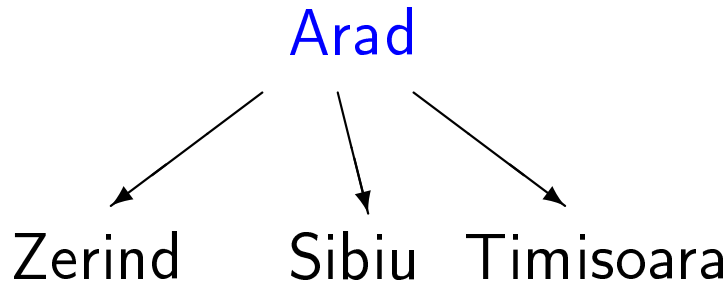
Primjer pretraživanja u širinu

- proširivanje stanja pri pretraživanju prostora u širinu za primjer planiranja puta do Bukurešta

Arad

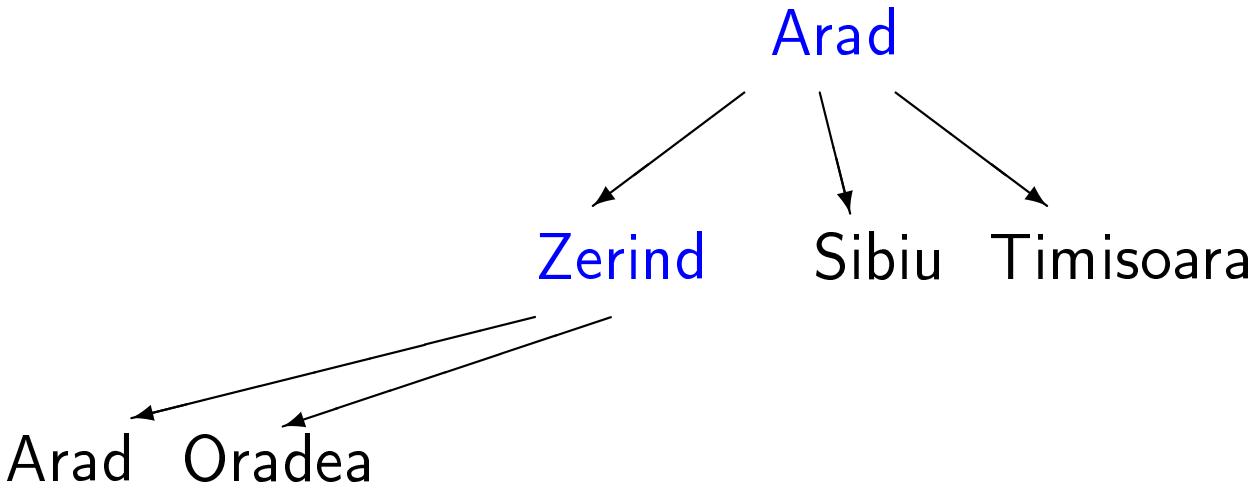
Primjer pretraživanja u širinu

- proširivanje stanja pri pretraživanju prostora u širinu za primjer planiranja puta do Bukurešta



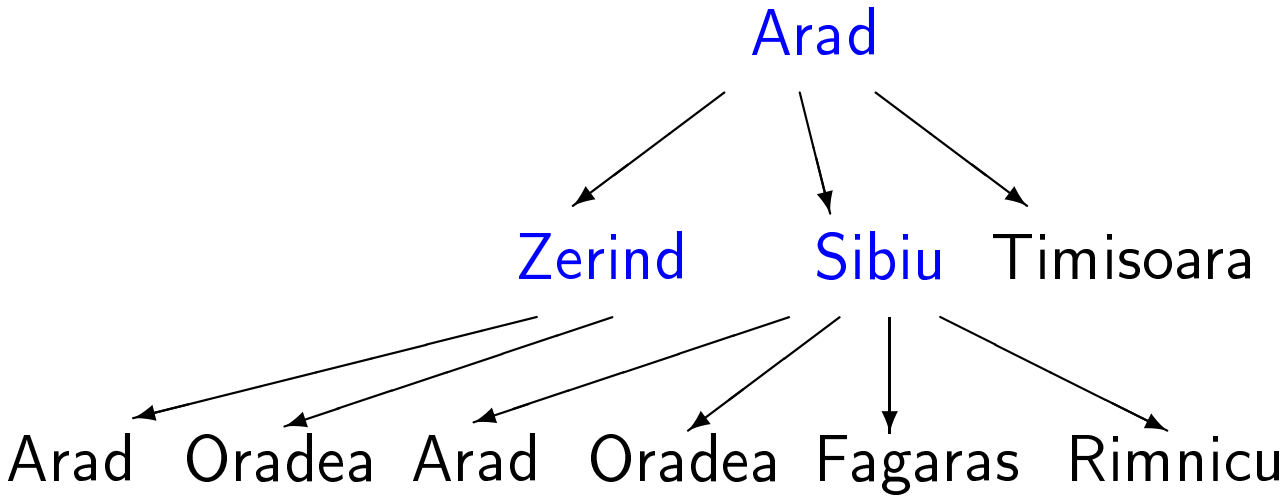
Primjer pretraživanja u širinu

- proširivanje stanja pri pretraživanju prostora u širinu za primjer planiranja puta do Bukurešta



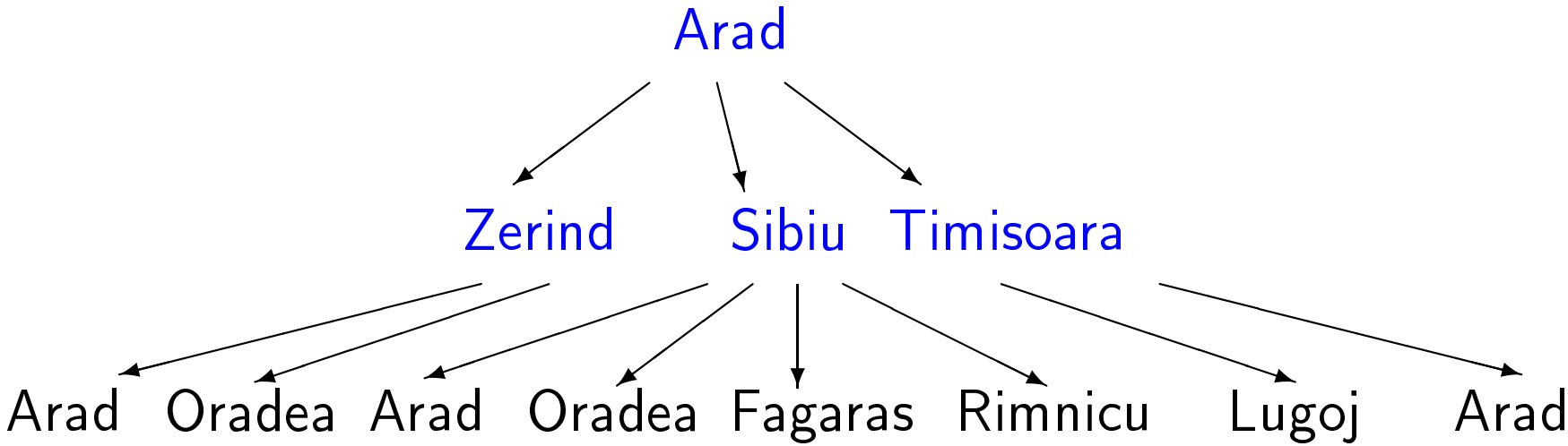
Primjer pretraživanja u širinu

- proširivanje stanja pri pretraživanju prostora u širinu za primjer planiranja puta do Bukurešta



Primjer pretraživanja u širinu

- proširivanje stanja pri pretraživanju prostora u širinu za primjer planiranja puta do Bukurešta



- primijetiti da korištenje liste zatvorenih čvorova nije obavezno, iako bi znatno suzilo stablo pretraživanja
- preferiraju se putovi sa manjim brojem prijelaza što u našem slučaju neće dovesti do optimalnog rješenja

Pretraživanje s jednolikom cijenom

- ideja: postaviti prioritet na ukupnu udaljenost od početnog čvora $\Rightarrow$ u svakom koraku, proširuje se najbliži otvoreni čvor

Pretraživanje s jednolikom cijenom

- ideja: postaviti prioritet na ukupnu udaljenost od početnog čvora $\Rightarrow$ u svakom koraku, proširuje se najbliži otvoreni čvor
- `S.prioritet(x):=c(prev(x),x)+p(prev(x))`
  - `p(prev(x))` $\equiv$ poznati prioritet prethodnog čvora
  - `c(prev(x),x)` $\equiv$ cijena tekućeg prijelaza

Pretraživanje s jednolikom cijenom

- ideja: postaviti prioritet na ukupnu udaljenost od početnog čvora $\Rightarrow$ u svakom koraku, proširuje se najbliži otvoreni čvor
- `S.prioritet(x):=c(prev(x),x)+p(prev(x))`
  - `p(prev(x))` $\equiv$ poznati prioritet prethodnog čvora
  - `c(prev(x),x)` $\equiv$ cijena tekućeg prijelaza
- broj stanja koja su bliža početnom od traženog: n_b=O(b^d)
  - potpunost: da (ako $n_b \neq \infty$)
  - vremenska složenost: n_b
  - prostorna složenost: n_b (najgori slučaj)
  - dosežljivost: da, ako cijena(x,y)$\geq 0, \forall x, y \in S$

Primjer pretraživanja s jednolikom cijenom

- proširivanje stanja pri pretraživanju prostora po principu “prvo najbliži”, za primjer planiranja puta do Bukurešta

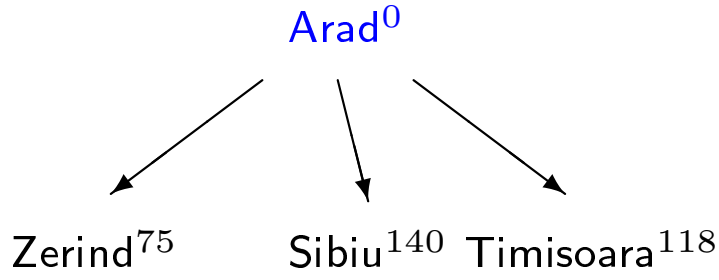
Primjer pretraživanja s jednolikom cijenom

- proširivanje stanja pri pretraživanju prostora po principu “prvo najbliži”, za primjer planiranja puta do Bukurešta

Arad

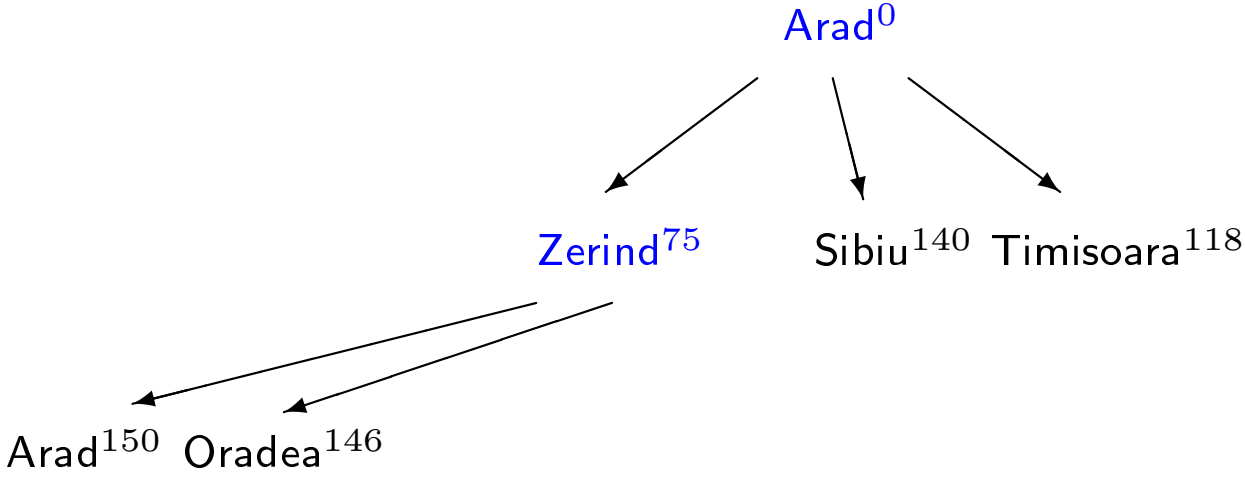
Primjer pretraživanja s jednolikom cijenom

- proširivanje stanja pri pretraživanju prostora po principu “prvo najbliži”, za primjer planiranja puta do Bukurešta



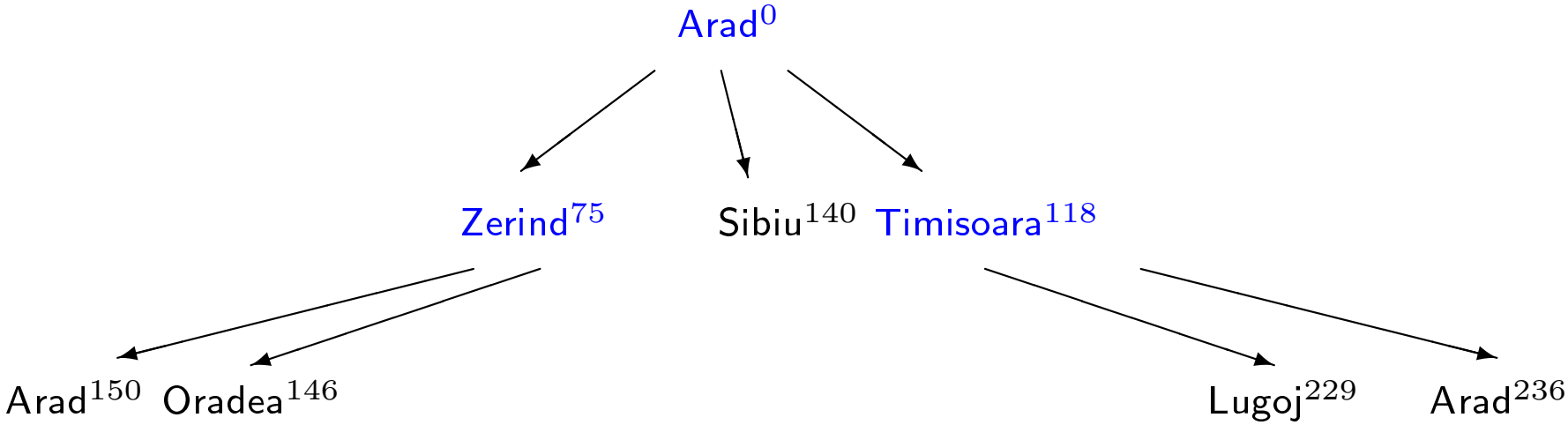
Primjer pretraživanja s jednolikom cijenom

- proširivanje stanja pri pretraživanju prostora po principu “prvo najbliži”, za primjer planiranja puta do Bukurešta



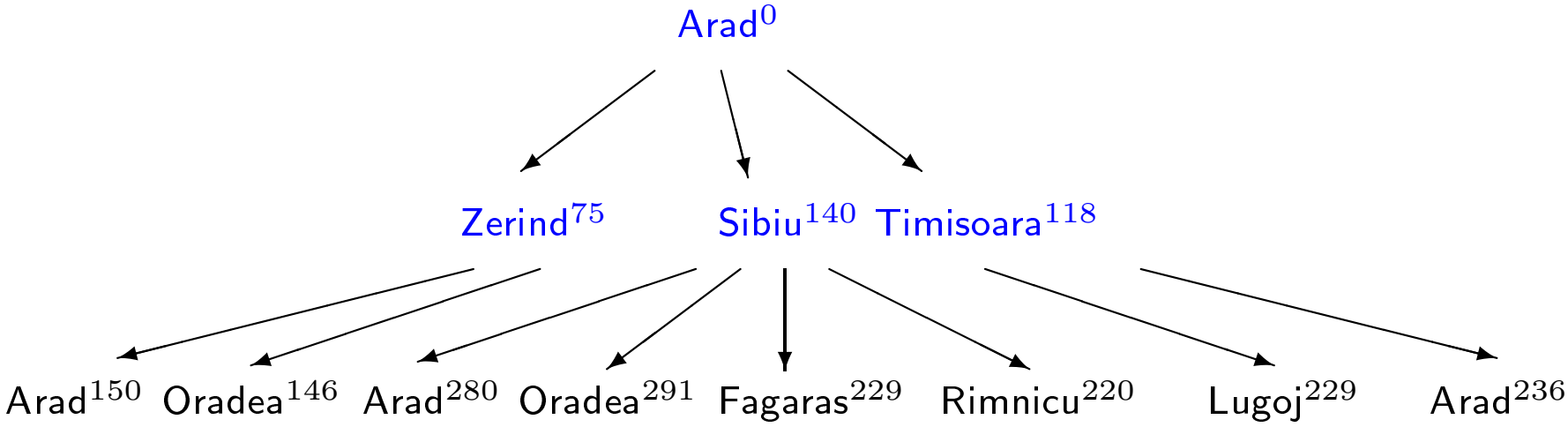
Primjer pretraživanja s jednolikom cijenom

- proširivanje stanja pri pretraživanju prostora po principu "prvo najbliži", za primjer planiranja puta do Bukurešta



Primjer pretraživanja s jednolikom cijenom

- proširivanje stanja pri pretraživanju prostora po principu "prvo najbliži", za primjer planiranja puta do Bukurešta



- primijetiti da korištenje liste zatvorenih čvorova ni ovdje nije obavezno (iako bi se suzilo stablo pretraživanja)
- preferiraju se bliži putovi što dovodi do optimalnog rješenja

Pretraživanje u dubinu

- implementacija: `S.prioritet(x):=-dubina(x)`
  (monotono pada za svaki novi poziv, neovisno o x)
  (nova stanja se umeću na **početak** liste O)

Pretraživanje u dubinu

- implementacija: `S.prioritet(x):=-dubina(x)`
  (monotono pada za svaki novi poziv, neovisno o x)
  (nova stanja se umeću na **početak** liste O)
- u svakom koraku, proširuje se najdublji otvoreni čvor:
  - potpunost: ne (u konačnim prostorima, da)
  - vremenska složenost: O(b^m) (m – najdublja dubina stabla)
    $\Rightarrow$grozno ukoliko je $m \gg d$!!

Pretraživanje u dubinu

- implementacija: `S.prioritet(x):=-dubina(x)`
  (monotono pada za svaki novi poziv, neovisno o x)
  (nova stanja se umeću na **početak** liste O)
- u svakom koraku, proširuje se najdublji otvoreni čvor:
  - potpunost: ne (u konačnim prostorima, da)
  - vremenska složenost: O(b^m) (m – najdublja dubina stabla)
    $\Rightarrow$grozno ukoliko je $m \gg d$!!
  - linearna prostorna složenost!!
    - prostorno najjeftiniji oblik pretraživanja
    - čuvaju se neproširene čvorovi na svim dubinama: O($b \cdot m$)
    - nažalost, sve to vrijedi ukoliko nam ne treba lista zatvorenih čvorova: inače, O(b^m)!

Pretraživanje u dubinu

- implementacija: `S.prioritet(x):=-dubina(x)`
  (monotono pada za svaki novi poziv, neovisno o x)
  (nova stanja se umeću na **početak** liste O)
- u svakom koraku, proširuje se najdublji otvoreni čvor:
  - potpunost: ne (u konačnim prostorima, da)
  - vremenska složenost: O(b^m) (m – najdublja dubina stabla)
    $\Rightarrow$grozno ukoliko je $m \gg d$!!
  - linearna prostorna složenost!!
    - prostorno najjeftiniji oblik pretraživanja
    - čuvaju se neproširene čvorovi na svim dubinama: O($b \cdot m$)
    - nažalost, sve to vrijedi ukoliko nam ne treba lista zatvorenih čvorova: inače, O(b^m)!
  - dosežljivost: ne

Pretraživanje u dubinu – svojstva

- dobitna kombinacija ukoliko su rješenja gusta: velika vjerojatnost da ubodemo neko “iz prve”

Pretraživanje u dubinu – svojstva

- dobitna kombinacija ukoliko su rješenja gusta: velika vjerojatnost da ubodemo neko "iz prve"
- inače, pristup je često nepraktičan zbog prevelike vremenske složenosti O(b^m) (Arad – Bukurešt via Zagreb)

Pretraživanje u dubinu – svojstva

- dobitna kombinacija ukoliko su rješenja gusta: velika vjerojatnost da ubodemo neko "iz prve"
- inače, pristup je često nepraktičan zbog prevelike vremenske složenosti O(b^m) (Arad – Bukurešt via Zagreb)
- na primjeru traženja izlaska iz labirinta:
  - pretraživanje u dubinu $\equiv$ pravilo desne ruke
  - pretraživanje u širinu $\equiv$???

Pretraživanje u dubinu – svojstva

- dobitna kombinacija ukoliko su rješenja gusta: velika vjerojatnost da ubodemo neko "iz prve"
- inače, pristup je često nepraktičan zbog prevelike vremenske složenosti O(b^m) (Arad – Bukurešt via Zagreb)
- na primjeru traženja izlaska iz labirinta:
  - pretraživanje u dubinu $\equiv$ pravilo desne ruke
  - pretraživanje u širinu $\equiv$???
- nedostatci:
  - postupak može ne biti potpun ukoliko neke grane prostora stanja nisu konačne
  - problemi sa ciklusima (rješivi) i jako povezanim prostorima stanja (teško rješivi) (višestruko posjećivanje velikih podstabala!)
  - kako odabrati sljedećeg iz skupa najdubljih čvorova?

Pretraživanje u dubinu – Prolog

- Prolog koristi pretraživanje u dubinu kao glavni mehanizam zaključivanja (stablo izvođenja!)

Pretraživanje u dubinu – Prolog

- Prolog koristi pretraživanje u dubinu kao glavni mehanizam zaključivanja (stablo izvođenja!)
- efikasnost pretraživanja osigurava se pažljivom konstrukcijom prostora stanja

Pretraživanje u dubinu – Prolog

- Prolog koristi pretraživanje u dubinu kao glavni mehanizam zaključivanja (stablo izvođenja!)
- efikasnost pretraživanja osigurava se pažljivom konstrukcijom prostora stanja
- zadatak programera: osigurati da stablo izvođenja generirano programom ne sadrži ni cikluse ni beskonačne grane

Primjer neispravnog programa

```
% procedura stvara listu od N atoma a
% zašto je procedura neispravna?
napravi(0,[]).
napravi(N,[a|R]):-
  N1 is N-1,
  napravi(N1,R).

% da li X sadrži [a,a,a]?
provjeri(X):-
  napravi(3,Aaa),
  povezi(Xp, _, X),
  povezi(_,Aaa, Xp).

% beskonačna petlja:
?-provjeri([a,a,s,a]).
```

Ograničeno pretraživanje u dubinu

- implementacija: postupak pretraživanja na dubini većoj od d_p ne generira sljedbenike

Ograničeno pretraživanje u dubinu

- implementacija: postupak pretraživanja na dubini većoj od d_p ne generira sljedbenike
- npr, d_p se može postaviti na broj članova prostora stanja $|S|$, ili na poznatu dubinu rješenja d

Ograničeno pretraživanje u dubinu

- implementacija: postupak pretraživanja na dubini većoj od d_p ne generira sljedbenike
- npr, d_p se može postaviti na broj članova prostora stanja $|S|$, ili na poznatu dubinu rješenja d
- ako je rješenje na dubini $d_p + 1$, nećemo ga naći

Iterativno pretraživanje u dubinu

- implementacija: OPD za $d_p = 1, 2, 3, 4, \ldots$

Iterativno pretraživanje u dubinu

- implementacija: OPD za $d_p = 1, 2, 3, 4, \ldots$
- u velikom broju slučajeva najbolja slijepa strategija:
  - potpunost: da
  - vremenska složenost: $O(b^d)$

$$b^d > \sum_{i=0}^{d-1} b^i; \quad \text{npr} : 3^5 = 243 > \sum_{i=0}^{4} 3^i = 121$$

  - prostorna složenost: $O(b \cdot d)$ – linearno!!
  - dosežljivost: da, ako je cijena prijelaza konstantna (postoji varijanta koja je uvijek dosežljiva)

Iterativno pretraživanje u dubinu

- implementacija: OPD za $d_p = 1, 2, 3, 4, \ldots$
- u velikom broju slučajeva najbolja slijepa strategija:
  - potpunost: da
  - vremenska složenost: $O(b^d)$

$$b^d > \sum_{i=0}^{d-1} b^i; \quad \text{npr} : 3^5 = 243 > \sum_{i=0}^{4} 3^i = 121$$

  - prostorna složenost: $O(b \cdot d)$ – linearno!!
  - dosežljivost: da, ako je cijena prijelaza konstantna (postoji varijanta koja je uvijek dosežljiva)
- osjetljivost na jako povezane prostore stanja
- metoda izbora kad je $|S|$ velik, a d nepoznata!

Bidirekcionalno pretraživanje

- ideja: istovremeno pretraživati od početka prema cilju i od cilja prema početku

Bidirekcionalno pretraživanje

- ideja: istovremeno pretraživati od početka prema cilju i od cilja prema početku
- ukoliko je primjenjiva, najbolja slijepa strategija
  - potpunost: da
  - vremenska složenost: $O(b^{d/2})$
  - prostorna složenost: $O(b^{d/2})$
    (barem jedan smjer mora biti u širinu)
  - dosežljivost: da

Bidirekcionalno pretraživanje

- ideja: istovremeno pretraživati od početka prema cilju i od cilja prema početku
- ukoliko je primjenjiva, najbolja slijepa strategija
  - potpunost: da
  - vremenska složenost: $O(b^{d/2})$
  - prostorna složenost: $O(b^{d/2})$
    (barem jedan smjer mora biti u širinu)
  - dosežljivost: da
- broj otvorenih čvorova za konkretan slučaj $b = 10, d = 6$:
  - pretraživanje u širinu: preko 10^6
  - bidirekcionalno pretraživanje: oko 2000

Bidirekcionalno pretraživanje – problemi

- primjenjivo za probleme:
  - u kojima operatori imaju inverze

Bidirekcionalno pretraživanje – problemi

- primjenjivo za probleme:
  - u kojima operatori imaju inverze
  - u kojima ima mali broj unaprijed poznatih ciljnih stanja (ne vrijedi npr, za šah)

Bidirekcionalno pretraživanje – problemi

- primjenjivo za probleme:
  - u kojima operatori imaju inverze
  - u kojima ima mali broj unaprijed poznatih ciljnih stanja (ne vrijedi npr, za šah)
- potrebno je naći efikasan način usporedbe novih stanja sa svim posjećenim stanjima drugog smjera pretraživanja

Bidirekcionalno pretraživanje – problemi

- primjenjivo za probleme:
  - u kojima operatori imaju inverze
  - u kojima ima mali broj unaprijed poznatih ciljnih stanja (ne vrijedi npr, za šah)
- potrebno je naći efikasan način usporedbe novih stanja sa svim posjećenim stanjima drugog smjera pretraživanja
- potrebno odlučiti koje će se pretraživanje koristiti u kojoj polovici pretraživanja
  - širina – širina
  - širina – dubina
  - širina – iterativno produbljivanje

Usmjereno pretraživanje

- Metode najboljeg prvog
  (pohlepno pretraživanje, A*)
- Formuliranje heurističkih funkcija
- Pretraživanje s ograničenom memorijom
  (IDA*, SMA*)
- Postupci iterativnog poboljšanja
  (uspon na vrh, simulirano kaljenje)

Metoda najboljeg prvog

```
# funkcija vraća rješenje ili neuspjeh
# O ... prioritetni red otvorenih čvorova
# S ... strategija otvaranja novih čvorova
def općePretraživanje(problem, strategija):
  O.dodaj(problem.početnoStanje())
  while !O.prazan():
    s = O.izbaciPrvog()
    if (problem.ispitajCilj(s)):
      return rješenje
    for s_i in problem.proširi(s):
      O.dodaj(s_i, S.prioritet(s))
  return neuspjeh;
```

- opća strategija pretraživanja dodjeljuje prioritet svakom otvorenom stanju s (u literaturi, često $f(n)$)

Metoda najboljeg prvog

```
# funkcija vraća rješenje ili neuspjeh
# O ... prioritetni red otvorenih čvorova
# S ... strategija otvaranja novih čvorova
def općePretraživanje(problem, strategija):
  O.dodaj(problem.početnoStanje())
  while !O.prazan():
    s = O.izbaciPrvog()
    if (problem.ispitajCilj(s)):
      return rješenje
    for s_i in problem.proširi(s):
      O.dodaj(s_i, S.prioritet(s))
  return neuspjeh;
```

- opća strategija pretraživanja dodjeljuje prioritet svakom otvorenom stanju s (u literaturi, često $f(n)$)
- metoda najboljeg prvog: prioritet odgovara poželjnosti stanja!
- specijalni slučajevi: pohlepno pretraživanje, A*

Pohlepno pretraživanje

- poželjnost stanja određuje se isključivo na temelju procijenjene udaljenosti od tekućeg čvora n do cilja $f(n) = h(n)$

Pohlepno pretraživanje

- poželjnost stanja određuje se isključivo na temelju procijenjene udaljenosti od tekućeg čvora n do cilja $f(n) = h(n)$
- $h(n)$: heuristička funkcija, ili, kraće heuristika (cf. Heureka!):
  - pridjev se obično odnosi na tehnike koje poboljšavaju prosječnu, ali ne i performansu u najgorem slučaju
  - heuristička funkcija sadrži znanje o konkretnom problemu
  - mora biti jednostavna jer se primjenjuje na svaki čvor!

Pohlepno pretraživanje

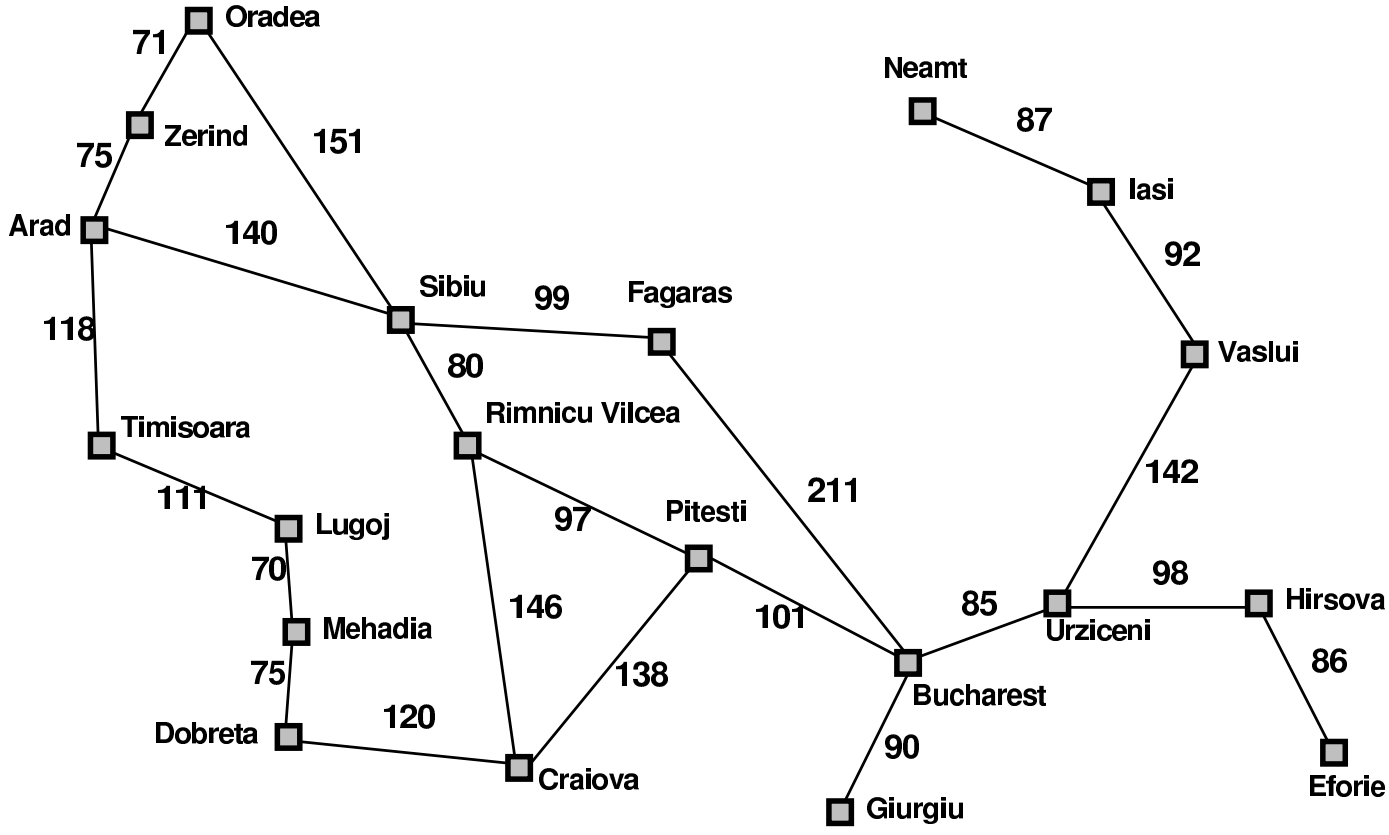
- poželjnost stanja određuje se isključivo na temelju procijenjene udaljenosti od tekućeg čvora n do cilja $f(n) = h(n)$
- $h(n)$: heuristička funkcija, ili, kraće heuristika (cf. Heureka!):
  - pridjev se obično odnosi na tehnike koje poboljšavaju prosječnu, ali ne i performansu u najgorem slučaju
  - heuristička funkcija sadrži znanje o konkretnom problemu
  - mora biti jednostavna jer se primjenjuje na svaki čvor!
- npr, za primjer putovanja po Rumunjskoj, uvodimo $h(n)$=pravocrtna udaljenost do Bukurešta

Pohlepno pretraživanje

- poželjnost stanja određuje se isključivo na temelju procijenjene udaljenosti od tekućeg čvora n do cilja $f(n) = h(n)$
- $h(n)$: heuristička funkcija, ili, kraće heuristika (cf. Heureka!):
  - pridjev se obično odnosi na tehnike koje poboljšavaju prosječnu, ali ne i performansu u najgorem slučaju
  - heuristička funkcija sadrži znanje o konkretnom problemu
  - mora biti jednostavna jer se primjenjuje na svaki čvor!
- npr, za primjer putovanja po Rumunjskoj, uvodimo $h(n)$=pravocrtna udaljenost do Bukurešta
- pohlepno pretraživanje odabire uvijek onaj čvor koji se čini najbliži cilju!

Primjer: putovanje po Rumunjskoj

Romania with step costs in km



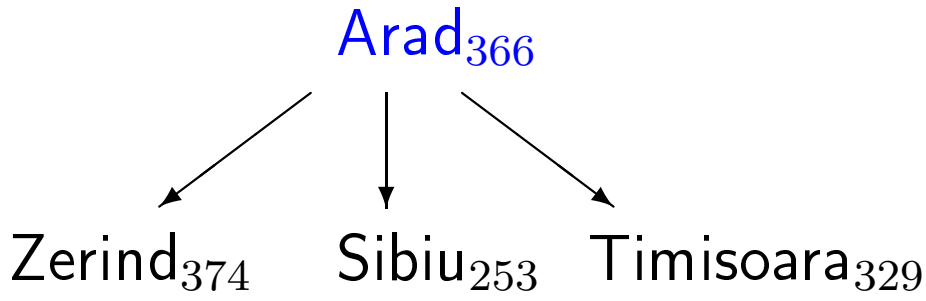
Straight–line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Pohlepno pretraživanje — primjer

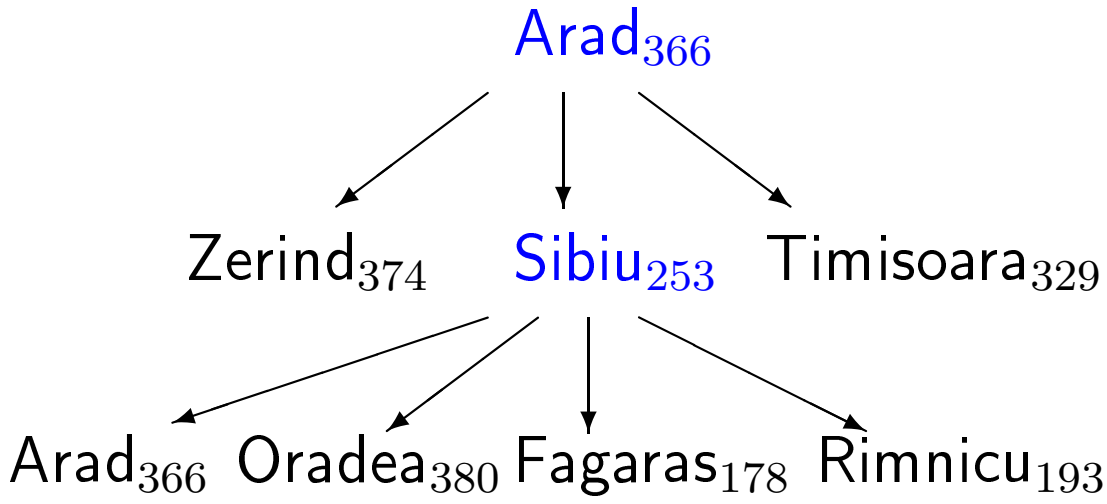
Pohlepno pretraživanje — primjer

Arad_{366}

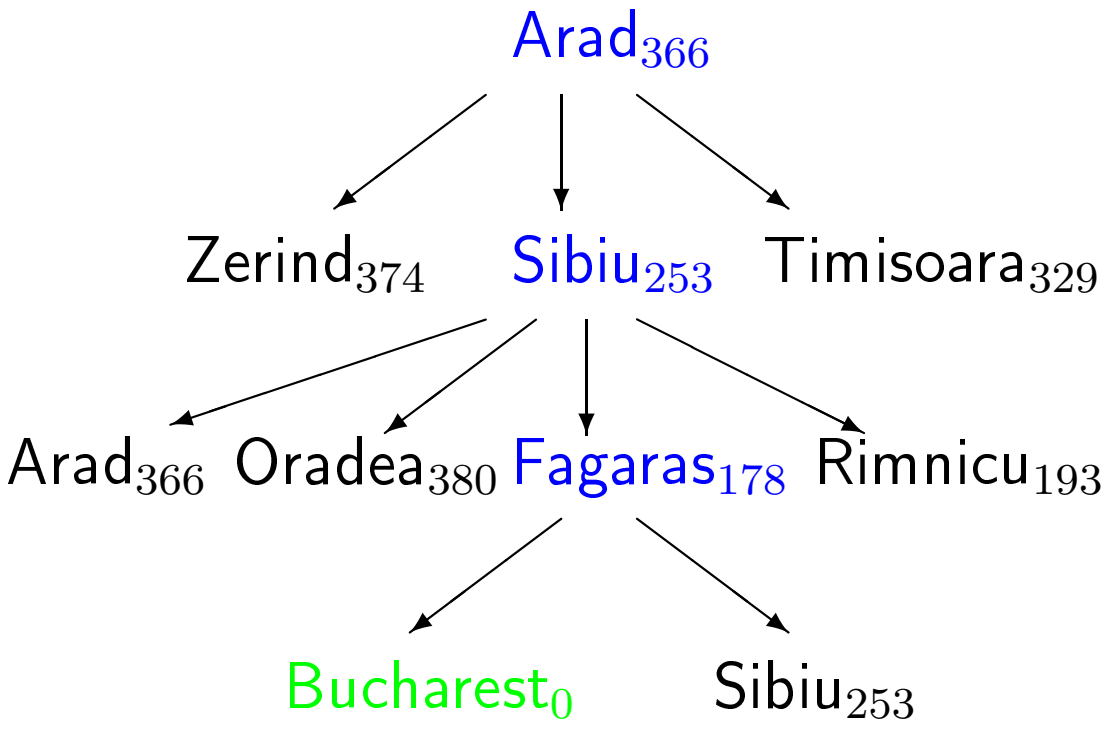
Pohlepno pretraživanje — primjer



Pohlepno pretraživanje — primjer



Pohlepno pretraživanje — primjer



- u ovom slučaju, nađeno je rješenje koje nije optimalno, uz minimalnu cijenu pretraživanja
- u općem slučaju, pretraživanje nije linearno (putevi zaobilaze planinska područja): Iasi $\longrightarrow$ Fagaras?
- ciklusi kritični, potrebna je lista zatvorenih čvorova

Pohlepno pretraživanje — svojstva

- strategija je pohlepna jer pokušava doći čim bliže rješenju u tekućem koraku, bez dugoročnijih razmatranja

Pohlepno pretraživanje — svojstva

- strategija je pohlepna jer pokušava doći čim bliže rješenju u tekućem koraku, bez dugoročnijih razmatranja
- performansa uvelike ovisi o heurističkoj funkciji, ali obično nema garancije za najgori slučaj:

Pohlepno pretraživanje — svojstva

- strategija je pohlepna jer pokušava doći čim bliže rješenju u tekućem koraku, bez dugoročnijih razmatranja
- performansa uvelike ovisi o heurističkoj funkciji, ali obično nema garancije za najgori slučaj:
  - potpunost: da (u konačnim prostorima)
  - vremenska složenost: O(b^m)
    (dosta ovisi o h)
  - prostorna složenost: O(b^m)
    (svi čvorovi se pamte, dosta ovisi o h)
  - dosežljivost: ne!
- nema ograničenja na svojstva heurističke funkcije

Pretraživanje A*

- za razliku od pohlepnog pretraživanja, A* izbjegava nastavljanje skupih puteva: $f(n) = g(n) + h(n)$:

Pretraživanje A*

- za razliku od pohlepnog pretraživanja, A* izbjegava nastavljanje skupih puteva: $f(n) = g(n) + h(n)$:
  - $g(n)$ — poznata ukupna udaljenost od s_0 do n (koristi se u pretraživanju s jednolikom cijenom)
  - $h(n)$ — procijenjena udaljenost od n do cilja (heuristička funkcija, kao u pohlepnom postupku)

Pretraživanje A*

- za razliku od pohlepnog pretraživanja, A* izbjegava nastavljanje skupih puteva: $f(n) = g(n) + h(n)$:
  - $g(n)$ — poznata ukupna udaljenost od s_0 do n (koristi se u pretraživanju s jednolikom cijenom)
  - $h(n)$ — procijenjena udaljenost od n do cilja (heuristička funkcija, kao u pohlepnom postupku)
- zahtijeva se svojstvo dosežljivosti za $h(n)$: $\forall n : h(n) \leq h^*(n)$, $h^*(n)$ je prava udaljenost od n do cilja $\Rightarrow h(n)$ optimistična, nikad ne precjenjuje

Pretraživanje A*

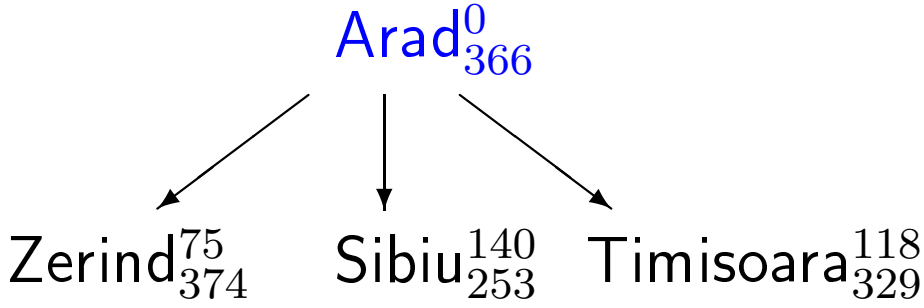
- za razliku od pohlepnog pretraživanja, A* izbjegava nastavljanje skupih puteva: $f(n) = g(n) + h(n)$:
  - $g(n)$ — poznata ukupna udaljenost od s_0 do n (koristi se u pretraživanju s jednolikom cijenom)
  - $h(n)$ — procijenjena udaljenost od n do cilja (heuristička funkcija, kao u pohlepnom postupku)
- zahtijeva se svojstvo dosežljivosti za $h(n)$: $\forall n : h(n) \leq h^*(n)$, $h^*(n)$ je prava udaljenost od n do cilja $\Rightarrow h(n)$ optimistična, nikad ne precjenjuje
- za naš primjer: pravocrtna udaljenost je dosežljiva

Pretraživanje A* — primjer

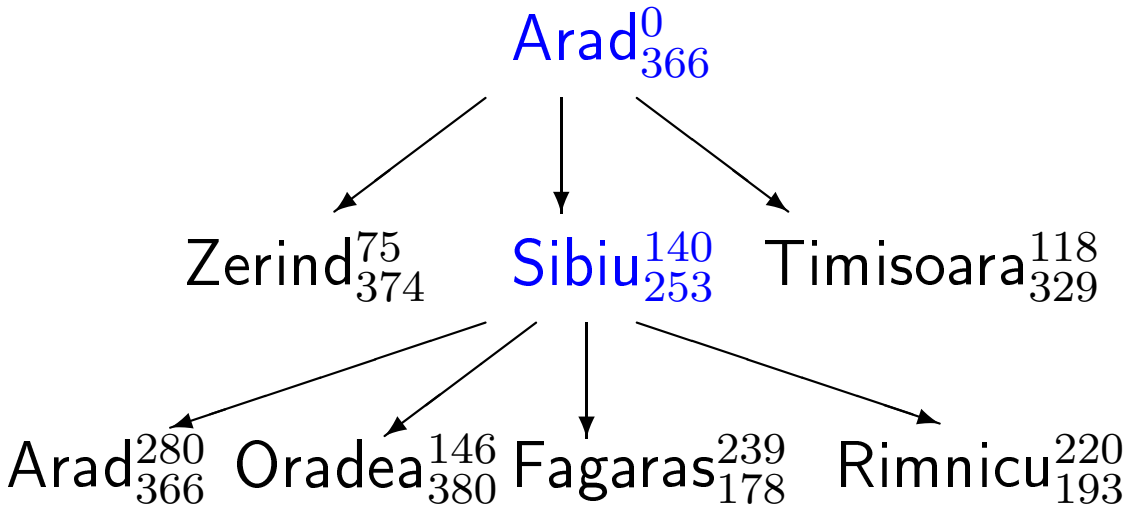
Pretraživanje A* — primjer

Arad_{366}

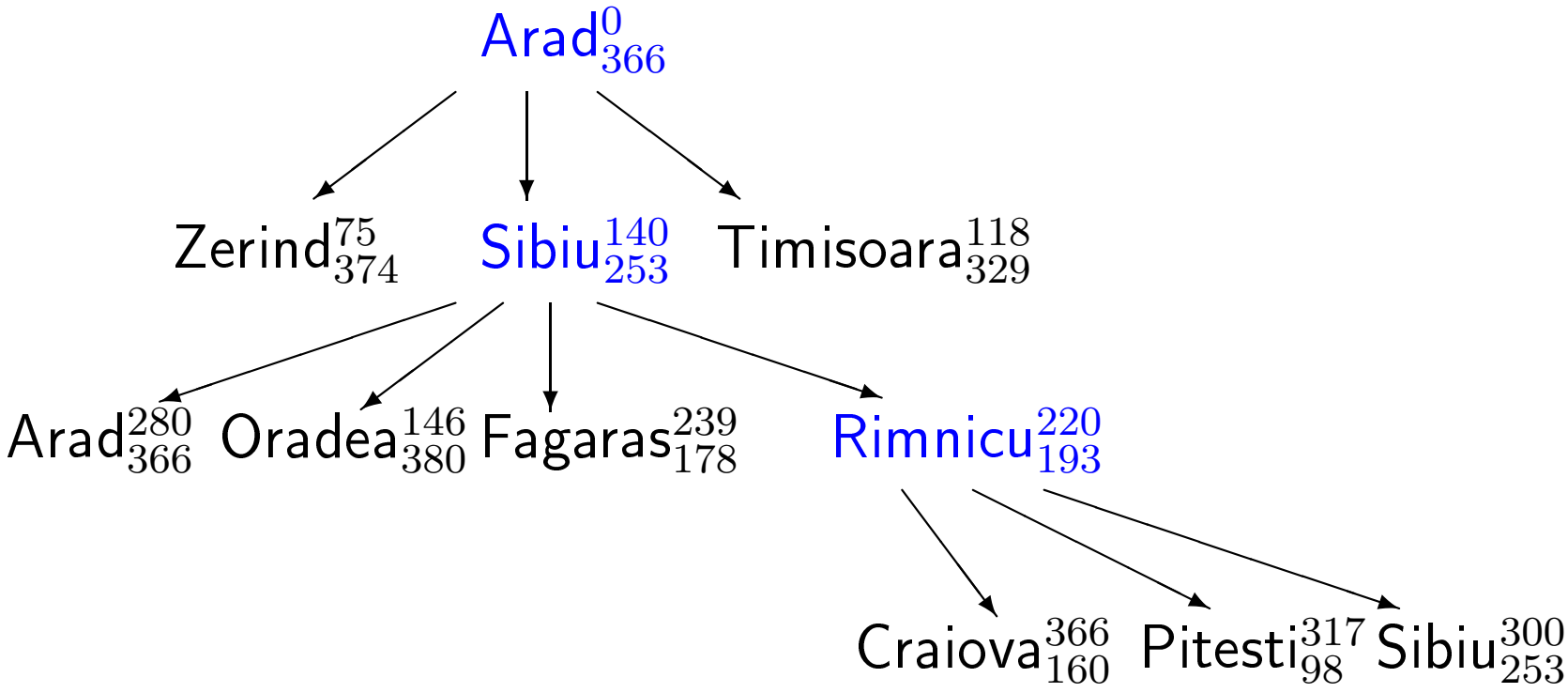
Pretraživanje A* — primjer



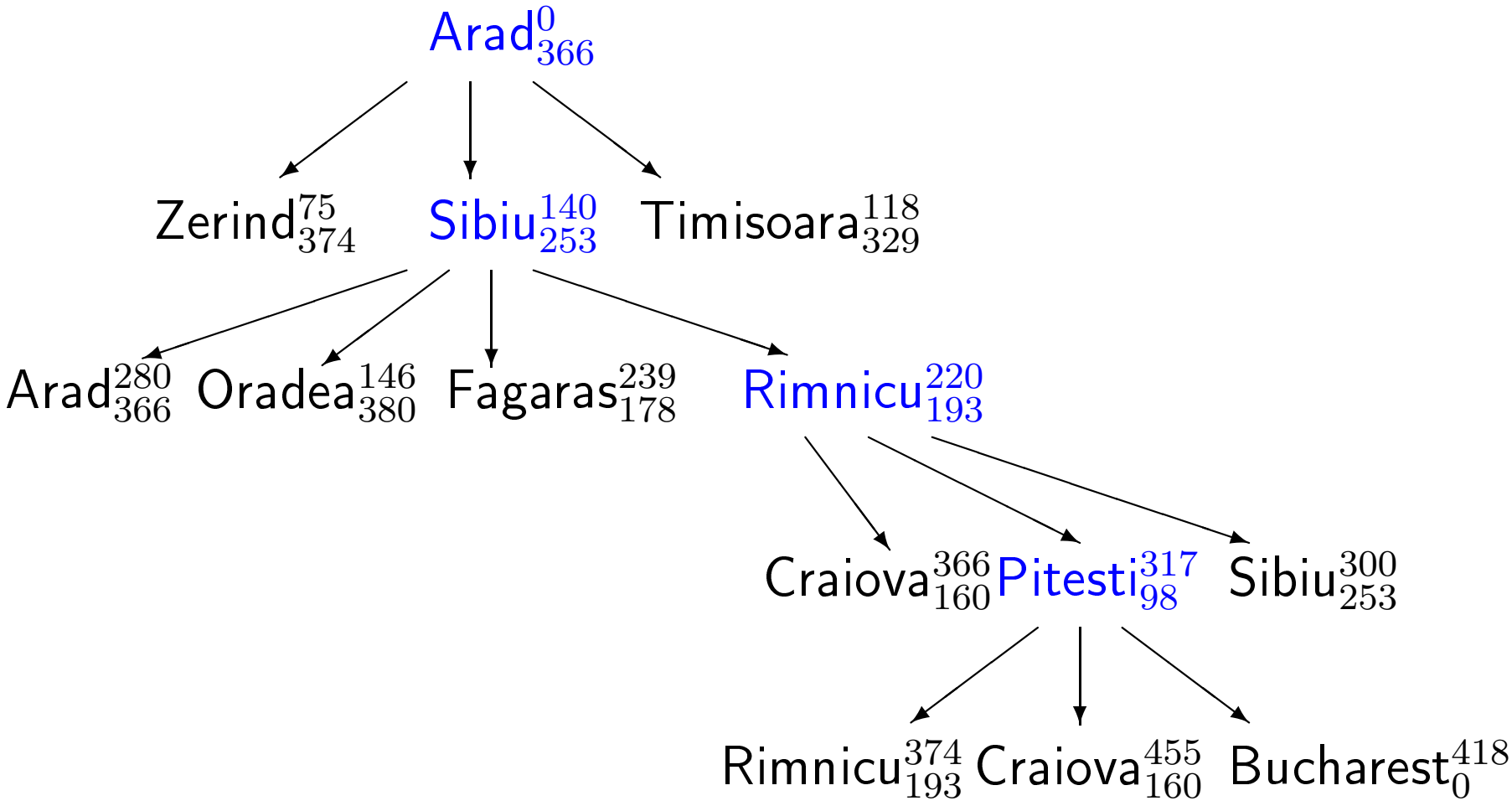
Pretraživanje A* — primjer



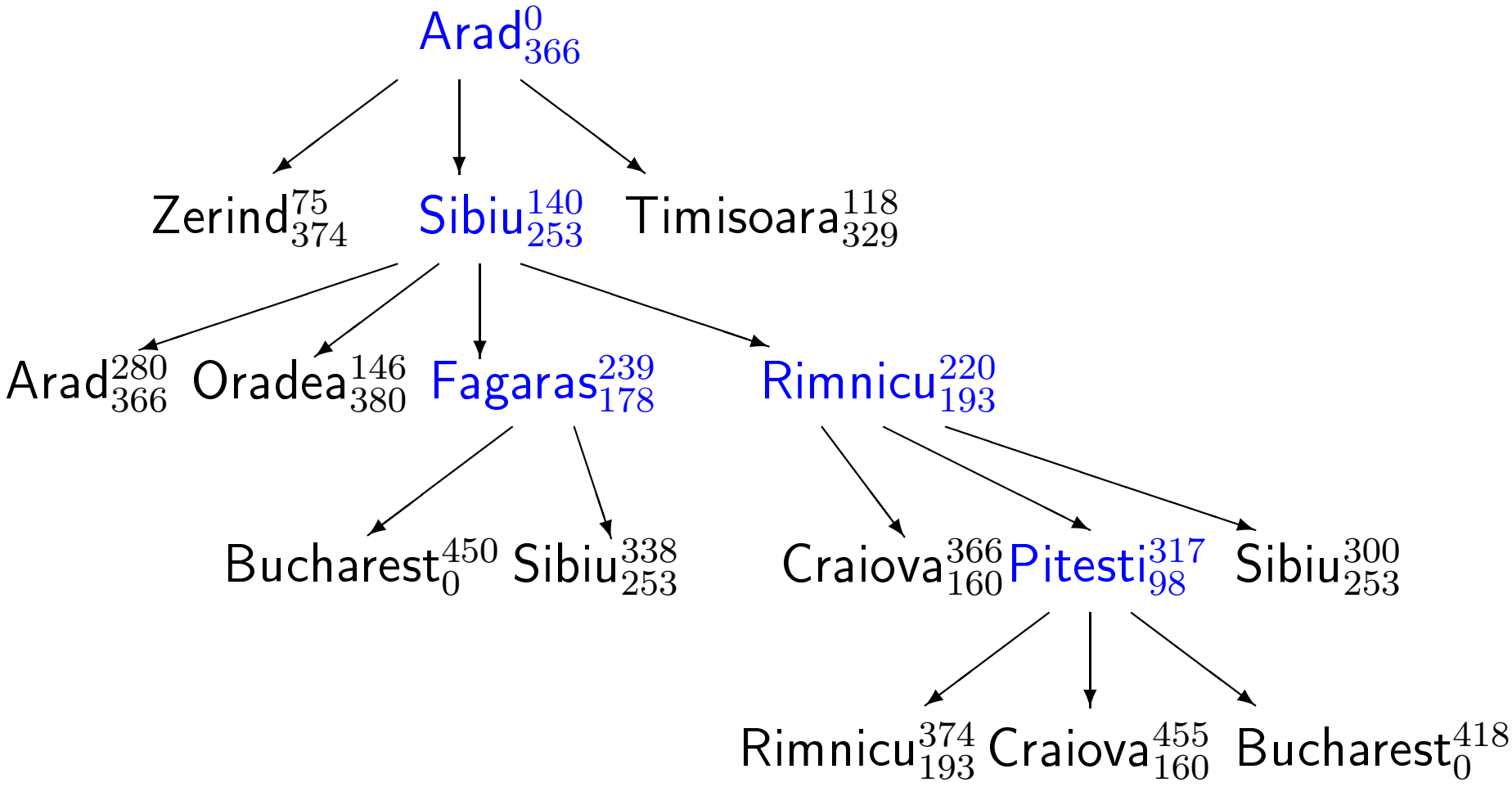
Pretraživanje A* — primjer



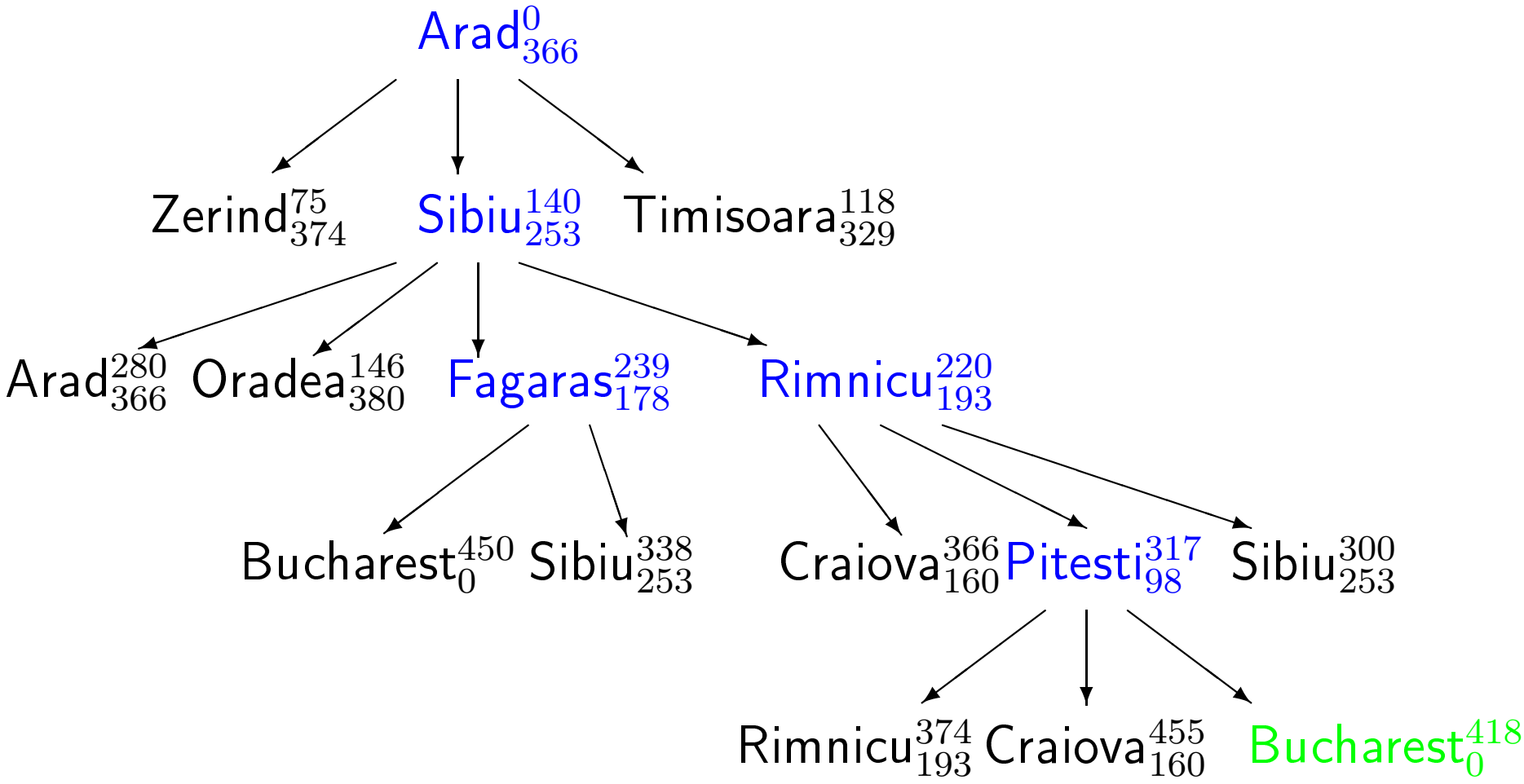
Pretraživanje A* — primjer



Pretraživanje A* — primjer



Pretraživanje A* — primjer



- pronađeno je optimalno rješenje uz razumnu cijenu pretraživanja, ciklusi smetaju ali nisu kritični

Optimalnost A*

- pretpostavke:
  - suboptimalni ciljni čvor s_{g2} nalazi se u O
  - optimalni ciljni čvor s_g nije otvoren
  - otvoren je čvor s_n, koji vodi prema s_g

Optimalnost A*

- pretpostavke:
  - suboptimalni ciljni čvor s_{g2} nalazi se u O
  - optimalni ciljni čvor s_g nije otvoren
  - otvoren je čvor s_n, koji vodi prema s_g
- npr: s_{g2}=Bucharest$_0^{450}$, s_g=Bucharest$_0^{418}$, s_n=Pitesti$_{102}^{317}$

Optimalnost A*

- pretpostavke:
  - suboptimalni ciljni čvor s_{g2} nalazi se u O
  - optimalni ciljni čvor s_g nije otvoren
  - otvoren je čvor s_n, koji vodi prema s_g
- npr: s_{g2}=Bucharest$_0^{450}$, s_g=Bucharest$_0^{418}$, s_n=Pitesti$_{102}^{317}$
- vrijedi:
  - $f(s_{g2})$=$g(s_{g2})$, $f(s_g)$=$g(s_g)$ (jer $h(s_{g2})$=$h(s_g)$=0!)
  - $g(s_{g2})$>$g(s_g)$ (jer s_{g2} nije optimalno)
  - $f(s_g) \geq f(s_n)$ (jer je h dosežljiva!)
  - $\Rightarrow f(s_{g2}) > f(s_n)$!

Optimalnost A*

- pretpostavke:
  - suboptimalni ciljni čvor s_{g2} nalazi se u O
  - optimalni ciljni čvor s_g nije otvoren
  - otvoren je čvor s_n, koji vodi prema s_g
- npr: s_{g2}=Bucharest$_0^{450}$, s_g=Bucharest$_0^{418}$, s_n=Pitesti$_{102}^{317}$
- vrijedi:
  - $f(s_{g2})$=$g(s_{g2})$, $f(s_g)$=$g(s_g)$ (jer $h(s_{g2})$=$h(s_g)$=0!)
  - $g(s_{g2})$>$g(s_g)$ (jer s_{g2} nije optimalno)
  - $f(s_g) \geq f(s_n)$ (jer je h dosežljiva!)
  - $\Rightarrow f(s_{g2}) > f(s_n)$!
- A* neće nikad proširiti s_{g2}, QED

Optimalnost A*, intuicija

- Kako h potcjenjuje pravu udaljenost, u načelu očekujemo porast f uzduž svakog puta stabla pretraživanja

Optimalnost A*, intuicija

- Kako h potcjenjuje pravu udaljenost, u načelu očekujemo porast f uzduž svakog puta stabla pretraživanja
- za neke legalne (dosežljive) heuristike, to može ne biti zadovoljeno

Optimalnost A*, intuicija

- Kako h potcjenjuje pravu udaljenost, u načelu očekujemo porast f uzduž svakog puta stabla pretraživanja
- za neke legalne (dosežljive) heuristike, to može ne biti zadovoljeno
- npr, $n \rightarrow n', c(n, n') = 1$:
  - n: $g(n)$=3, $h(n)$=4 $\Rightarrow$ $f(n)$=7
  - n': $g(n')$=4, $h(n')$=2 $\Rightarrow$ $f(n')$=6

Optimalnost A*, intuicija

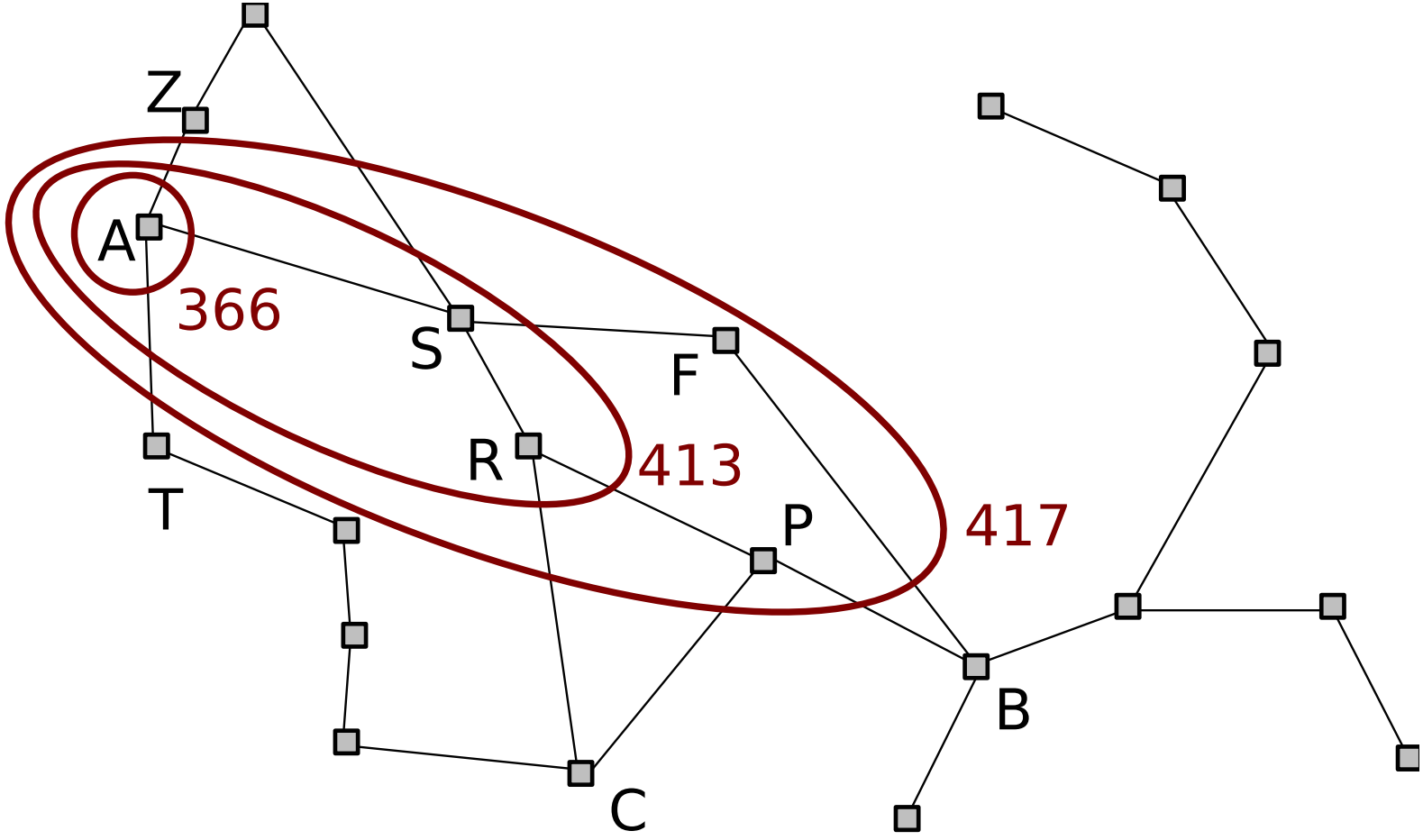
- Kako h potcjenjuje pravu udaljenost, u načelu očekujemo porast f uzduž svakog puta stabla pretraživanja
- za neke legalne (dosežljive) heuristike, to može ne biti zadovoljeno
- npr, $n \rightarrow n', c(n, n') = 1$:
  - n: $g(n)$=3, $h(n)$=4 $\Rightarrow$ $f(n)$=7
  - n': $g(n')$=4, $h(n')$=2 $\Rightarrow$ $f(n')$=6
- međutim, tzv. pathmax postupkom nemonotonu heuristiku možemo "popraviti":
  $f(n') =$max$(f(n), g(n') + h(n'))$

Optimalnost A*, intuicija

- Kako h potcjenjuje pravu udaljenost, u načelu očekujemo porast f uzduž svakog puta stabla pretraživanja
- za neke legalne (dosežljive) heuristike, to može ne biti zadovoljeno
- npr, $n \rightarrow n', c(n, n') = 1$:
  - n: $g(n)$=3, $h(n)$=4 $\Rightarrow$ $f(n)$=7
  - n': $g(n')$=4, $h(n')$=2 $\Rightarrow$ $f(n')$=6
- međutim, tzv. pathmax postupkom nemonotonu heuristiku možemo "popraviti":
  $f(n') =$max$(f(n), g(n') + h(n'))$
- $\Rightarrow$ vrijednost heurističke funkcije f monotono raste duž svakog puta u stablu pretraživanja, za svaku dosežljivu (optimističnu) heuristiku h!

Optimalnost A*, intuicija

A* širi čvorove prema rastućim vrijednostima f:
niti jedan čvor sa $f(n) > f^*(s_0)$ neće biti proširen!



A* — svojstva

- slična strategiji jednolike cijene, uz usmjeravanje fronte širenja pretraživanja u skladu sa zadanom heurističkom funkcijom

A* — svojstva

- slična strategiji jednolike cijene, uz usmjeravanje fronte širenja pretraživanja u skladu sa zadanom heurističkom funkcijom
- pokazuje se da je A* optimalno efikasan, tj, ima najmanju cijenu pretraživanja, za svaku legalnu heuristiku

A* — svojstva

- slična strategiji jednolike cijene, uz usmjeravanje fronte širenja pretraživanja u skladu sa zadanom heurističkom funkcijom
- pokazuje se da je A* optimalno efikasan, tj, ima najmanju cijenu pretraživanja, za svaku legalnu heuristiku
- unatoč dobrim svojstvima, složenost algoritma ostaje eksponencijalna osim za ograničen broj slučaja za koje vrijedi $|h(n) - h^*(n)| \leq O(log(h^*(n)))$:
  - potpunost: da (ako je b konačan)
  - vremenska složenost: eksponencijalna u $|h(n) - h^*(n)| \cdot d$
  - prostorna složenost: O(b^d) usko grlo!
    (pamti se cijela fronta, oblik ovisi o h)
  - dosežljivost: da!
- dozvoljene samo dosežljive (optimistične) heurističke funkcije

Vrednovanje heurističkih funkcija

- ključni parametar: broj proširenih čvorova N
- efektivni faktor grananja b^*:
  - svojstvo stabla pretraživanja određenog strategijom
  - vrijedi: $N = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$
  - omogućava vrednovanje u pojedinačnim slučajevima
  - da li možemo dati općenitu ocjenu?

Vrednovanje heurističkih funkcija

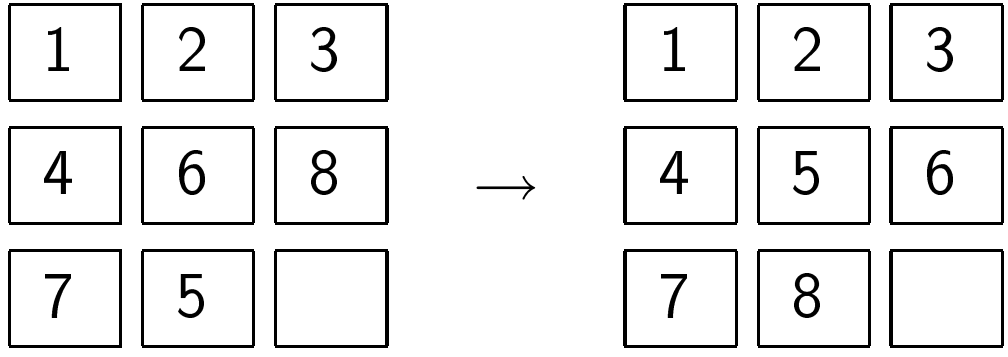
- ključni parametar: broj proširenih čvorova N
- efektivni faktor grananja b^*:
  - svojstvo stabla pretraživanja određenog strategijom
  - vrijedi: $N = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$
  - omogućava vrednovanje u pojedinačnim slučajevima
  - da li možemo dati općenitu ocjenu?
- za svaku dosežljivu heuristiku h, te za svaki čvor n vrijedi:
  - što je $h(n)$ veći, to je $h(n)$ preciznija!
  - preciznost u smislu $h^*(n) - h(n)$
  - preciznija heuristika implicira usmjerenije pretraživanje, manji ukupni broj čvorova N, veću performansu
- Ako za dvije heuristike h_1 i h_2 vrijedi:
  $\forall n : h_2(n) \geq h_1(n)$
  onda je h_2 bolja heuristika i kažemo da h_2 dominira h_1

Vrednovanje heurističkih funkcija

- ključni parametar: broj proširenih čvorova N
- efektivni faktor grananja b^*:
  - svojstvo stabla pretraživanja određenog strategijom
  - vrijedi: $N = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$
  - omogućava vrednovanje u pojedinačnim slučajevima
  - da li možemo dati općenitu ocjenu?
- za svaku dosežljivu heuristiku h, te za svaki čvor n vrijedi:
  - što je $h(n)$ veći, to je $h(n)$ preciznija!
  - preciznost u smislu $h^*(n) - h(n)$
  - preciznija heuristika implicira usmjerenije pretraživanje, manji ukupni broj čvorova N, veću performansu
- Ako za dvije heuristike h_1 i h_2 vrijedi:
  $\forall n : h_2(n) \geq h_1(n)$
  onda je h_2 bolja heuristika i kažemo da h_2 dominira h_1

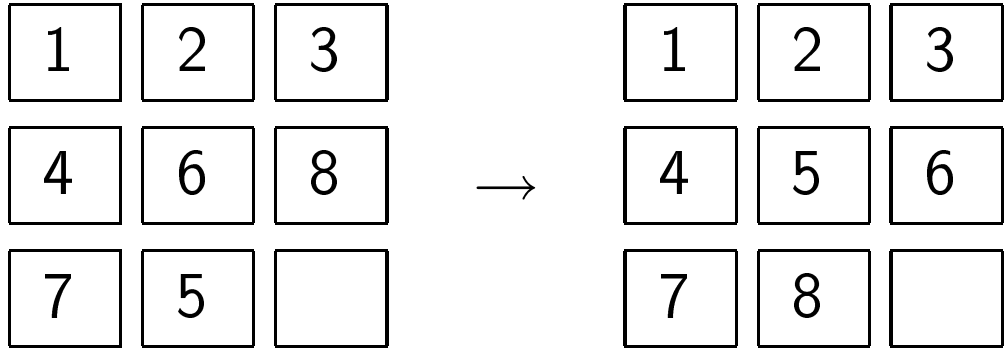
Vrednovanje heuristika – primjer

- razmotrimo problem slagalice 3×3:



Vrednovanje heuristika – primjer

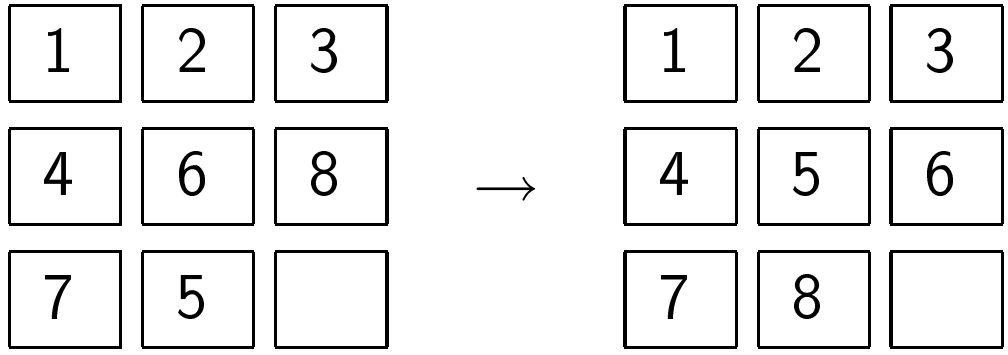
- razmotrimo problem slagalice 3×3:



- jednostavni primjeri heurističkih funkcija:
  - h1(n) ... broj pločica koje nisu na svom mjestu
  - h2(n) ... zbroj L1 udaljenosti pločica do svog mjesta

Vrednovanje heuristika – primjer

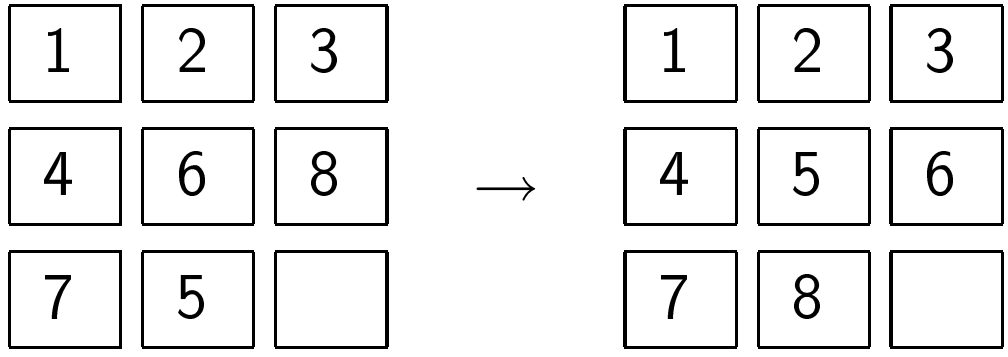
- razmotrimo problem slagalice 3×3:



- jednostavni primjeri heurističkih funkcija:
  - h1(n) ... broj pločica koje nisu na svom mjestu =3
  - h2(n) ... zbroj L1 udaljenosti pločica do svog mjesta =4
  - očito, h_2 dominira h_1!

Vrednovanje heuristika – primjer

- razmotrimo problem slagalice 3×3:



- jednostavni primjeri heurističkih funkcija:
  - h1(n) ... broj pločica koje nisu na svom mjestu =3
  - h2(n) ... zbroj L1 udaljenosti pločica do svog mjesta =4
  - očito, h_2 dominira h_1!

- empirijski rezultati (100 primjera/dubina)

	IPD	A*$_{h1}$	A*$_{h2}$
N(d=14)	$3 \cdot 10^6$	539	113
N(d=22)	–	$4 \cdot 10^4$	1500
b^*	2.8	1.5	1.3

Pronalaženje heurističkih funkcija

- dosežljiva heuristika može se generirati kao **točna** cijena problema s opuštenim ograničenjima. Za slagalicu, npr:
  - dozvolimo da se pločice mogu pomicati proizvoljno
  - dozvolimo pomicanje na sva susjedna mjesta

Pronalaženje heurističkih funkcija

- dosežljiva heuristika može se generirati kao **točna** cijena problema s opuštenim ograničenjima. Za slagalicu, npr:
  - dozvolimo da se pločice mogu pomicati proizvoljno $(=h_1)$
  - dozvolimo pomicanje na sva susjedna mjesta $(=h_2)$

Pronalaženje heurističkih funkcija

- dosežljiva heuristika može se generirati kao **točna** cijena problema s opuštenim ograničenjima. Za slagalicu, npr:
  - dozvolimo da se pločice mogu pomicati proizvoljno $(=h_1)$
  - dozvolimo pomicanje na sva susjedna mjesta $(=h_2)$
  - sokoban: udaljenost dijamanta do najbližeg cilja, ako uklonimo sve ostale dijamante iz igre (moguće tabelirati!)

Pronalaženje heurističkih funkcija

- dosežljiva heuristika može se generirati kao **točna** cijena problema s opuštenim ograničenjima. Za slagalicu, npr:
  - dozvolimo da se pločice mogu pomicati proizvoljno $(=h_1)$
  - dozvolimo pomicanje na sva susjedna mjesta $(=h_2)$
  - sokoban: udaljenost dijamanta do najbližeg cilja, ako uklonimo sve ostale dijamante iz igre (moguće tabelirati!)
- ukoliko imamo više heuristika od kojih niti jedna ne dominira rezultate je moguće kombinirati:
  $h(n) = \max_i h_i(n)$, $h(n)$ dominira $h_i(n), \forall i$!

Pronalaženje heurističkih funkcija

- dosežljiva heuristika može se generirati kao **točna** cijena problema s opuštenim ograničenjima. Za slagalicu, npr:
  - dozvolimo da se pločice mogu pomicati proizvoljno $(=h_1)$
  - dozvolimo pomicanje na sva susjedna mjesta $(=h_2)$
  - sokoban: udaljenost dijamanta do najbližeg cilja, ako uklonimo sve ostale dijamante iz igre (moguće tabelirati!)
- ukoliko imamo više heuristika od kojih niti jedna ne dominira rezultate je moguće kombinirati:
  $h(n) = \max_i h_i(n)$, $h(n)$ dominira $h_i(n), \forall i$!
- pronaći ključne značajke problema, h odrediti kao npr. linearnu kombinaciju, uz podešavanje parametara strojnim učenjem

Pretraživanje s ograničenom memorijom

- unatoč poboljšanjima na području strategija i heuristika, neki problemi su inherentno teški

Pretraživanje s ograničenom memorijom

- unatoč poboljšanjima na području strategija i heuristika, neki problemi su inherentno teški
- u velikom broju slučajeva, usko grlo je *raspoloživa memorija*

Pretraživanje s ograničenom memorijom

- unatoč poboljšanjima na području strategija i heuristika, neki problemi su inherentno teški
- u velikom broju slučajeva, usko grlo je *raspoloživa memorija*
- razvijene metode koje opuštaju memorijske zahtjeve A*, na račun povećane vremenske složenosti

Pretraživanje s ograničenom memorijom

- unatoč poboljšanjima na području strategija i heuristika, neki problemi su inherentno teški
- u velikom broju slučajeva, usko grlo je *raspoloživa memorija*
- razvijene metode koje opuštaju memorijske zahtjeve A*, na račun povećane vremenske složenosti
- IDA* – heurističko proširenje iterativnog pretraživanja u dubinu

Pretraživanje s ograničenom memorijom

- unatoč poboljšanjima na području strategija i heuristika, neki problemi su inherentno teški
- u velikom broju slučajeva, usko grlo je *raspoloživa memorija*
- razvijene metode koje opuštaju memorijske zahtjeve A*, na račun povećane vremenske složenosti
- IDA* – heurističko proširenje iterativnog pretraživanja u dubinu
- SMA* – A* s ograničenom memorijom

IDA* — glavna ideja

- kao i kod od IPD, u svakoj iteraciji se provodi efikasno pretraživanje u dubinu

IDA* — glavna ideja

- kao i kod od IPD, u svakoj iteraciji se provodi efikasno pretraživanje u dubinu
- umjesto dubine, međutim, ograničava se vrijednost funkcije $f(n) = g(n) + h(n)$

IDA* — glavna ideja

- kao i kod od IPD, u svakoj iteraciji se provodi efikasno pretraživanje u dubinu
- umjesto dubine, međutim, ograničava se vrijednost funkcije $f(n) = g(n) + h(n)$
- proširuju se svi čvorovi unutar tekuće f-konture, ograničenje u sljedećem koraku se postavlja na minimalnu veću vrijednost f, za sve otvorene čvorove

IDA* — glavna ideja

- kao i kod od IPD, u svakoj iteraciji se provodi efikasno pretraživanje u dubinu
- umjesto dubine, međutim, ograničava se vrijednost funkcije $f(n) = g(n) + h(n)$
- proširuju se svi čvorovi unutar tekuće f-konture, ograničenje u sljedećem koraku se postavlja na minimalnu veću vrijednost f, za sve otvorene čvorove
- lista zatvorenih čvorova se ne koristi (naravno)

IDA* — nedostatci

- performansa algoritma kritično ovisi o kvaliteti heuristike:
  - dobra heuristika $f(s_0) \approx f(\mathbf{cilj})$: u svakoj iteraciji proširujemo više čvorova
  - loša heuristika $f(s_0) \ll f(\mathbf{cilj})$: u najgorem slučaju, u svakoj iteraciji proširujemo točno po jedan čvor $\Rightarrow N$ iteracija, ukupno N^2 čvorova (grozno)

IDA* — nedostatci

- performansa algoritma kritično ovisi o kvaliteti heuristike:
  - dobra heuristika $f(s_0) \approx f(\mathbf{cilj})$: u svakoj iteraciji proširujemo više čvorova
  - loša heuristika $f(s_0) \ll f(\mathbf{cilj})$: u najgorem slučaju, u svakoj iteraciji proširujemo točno po jedan čvor $\Rightarrow N$ iteracija, ukupno N^2 čvorova (grozno)
- moguće rješenje je ϵ-dosežljivi postupak:
  - f granicu se uvećava za ne manje od ϵ
  - smanjuje se cijena pretraživanja, po cijenu dosežljivosti
  - cijena rješenja je najviše za ϵ veća od optimalne

IDA* — nedostatci

- performansa algoritma kritično ovisi o kvaliteti heuristike:
  - dobra heuristika $f(s_0) \approx f(\mathbf{cilj})$: u svakoj iteraciji proširujemo više čvorova
  - loša heuristika $f(s_0) \ll f(\mathbf{cilj})$: u najgorem slučaju, u svakoj iteraciji proširujemo točno po jedan čvor $\Rightarrow N$ iteracija, ukupno N^2 čvorova (grozno)
- moguće rješenje je ϵ-dosežljivi postupak:
  - f granicu se uvećava za ne manje od ϵ
  - smanjuje se cijena pretraživanja, po cijenu dosežljivosti
  - cijena rješenja je najviše za ϵ veća od optimalne
- osjetljivost na jako povezane prostore stanja

IDA* — svojstva

- potpunost: da (ako je b konačan)
- vremenska složenost u odnosu na A*:
  - ◇ više otvorenih čvorova (ako je h dobra, ne puno više)
  - ◇ manja složenost rukovanja čvorovima (obična rekurzija vs. prioritetni red)
- prostorna složenost: $O(b \cdot d)$
  (posljedica pretraživanja u dubinu, jako dobro)
- dosežljivost: da (ili ϵ-dosežljivost)

- dozvoljene su samo dosežljive (optimistične) heurističke funkcije

SMA* — glavna ideja

- kritika IDA*: premalo informacije se prenosi između iteracija, samo tekuća granica funkcije f

SMA* — glavna ideja

- kritika IDA*: premalo informacije se prenosi između iteracija, samo tekuća granica funkcije f
- SMA* konstantno koristi svu raspoloživu memoriju: radi se o A* sa ograničenjem na broj čvorova u redu O

SMA* — glavna ideja

- kritika IDA*: premalo informacije se prenosi između iteracija, samo tekuća granica funkcije f
- SMA* konstantno koristi svu raspoloživu memoriju: radi se o A* sa ograničenjem na broj čvorova u redu O
- prilikom unošenja n+1-og čvora u O, iz O se briše najplići čvor s najvećom vrijednošću f: ta vrijednost se naknadno upisuje u posebno polje prethodnika izbrisanog čvora

SMA* — glavna ideja

- kritika IDA*: premalo informacije se prenosi između iteracija, samo tekuća granica funkcije f
- SMA* konstantno koristi svu raspoloživu memoriju: radi se o A* sa ograničenjem na broj čvorova u redu O
- prilikom unošenja n+1-og čvora u O, iz O se briše najplići čvor s najvećom vrijednošću f: ta vrijednost se naknadno upisuje u posebno polje prethodnika izbrisanog čvora
- ukoliko prioritet nekog izbrisanog čvora spremljen u njegovom prethodniku postane najbolji među svim otvorenim čvorovima, izbrisani čvor se mora rekonstruirati (dvostruki posao)

SMA* — svojstva

- potpunost: da, ako dubina najplićeg rješenja nije veća od raspoložive memorije
- vremenska složenost u odnosu na A*: znatno veća
- prostorna složenost: konstantna (algoritam će iskoristiti svu raspoloživu memoriju)
- dosežljivost: da, ako dubina najplićeg optimalnog rješenja nije veća od raspoložive memorije

SMA* — svojstva

- potpunost: da, ako dubina najplićeg rješenja nije veća od raspoložive memorije
- vremenska složenost u odnosu na A*: znatno veća
- prostorna složenost: konstantna (algoritam će iskoristiti svu raspoloživu memoriju)
- dosežljivost: da, ako dubina najplićeg optimalnog rješenja nije veća od raspoložive memorije

- ako ima dovoljno memorije za cijelo stablo, postupak je ekvivalentan A*

SMA* — svojstva

- potpunost: da, ako dubina najplićeg rješenja nije veća od raspoložive memorije
- vremenska složenost u odnosu na A*: znatno veća
- prostorna složenost: konstantna (algoritam će iskoristiti svu raspoloživu memoriju)
- dosežljivost: da, ako dubina najplićeg optimalnog rješenja nije veća od raspoložive memorije

- ako ima dovoljno memorije za cijelo stablo, postupak je ekvivalentan A*
- definitivno bolji od IDA* za jako povezane prostore stanja i realne heuristike

SMA* — svojstva

- potpunost: da, ako dubina najplićeg rješenja nije veća od raspoložive memorije
- vremenska složenost u odnosu na A*: znatno veća
- prostorna složenost: konstantna (algoritam će iskoristiti svu raspoloživu memoriju)
- dosežljivost: da, ako dubina najplićeg optimalnog rješenja nije veća od raspoložive memorije
- ako ima dovoljno memorije za cijelo stablo, postupak je ekvivalentan A*
- definitivno bolji od IDA* za jako povezane prostore stanja i realne heuristike
- usko grlo: rekonstrukcija izbrisanih čvorova, što je posebno izraženo kod lošijih heuristika

Postupci iterativnog poboljšanja

- Neki problemi imaju svojstvo da **put** do rješenja nije relevantan (npr, 8 kraljica, VLSI prospoj, putujući trgovac)

Postupci iterativnog poboljšanja

- Neki problemi imaju svojstvo da **put** do rješenja nije relevantan (npr, 8 kraljica, VLSI prospoj, putujući trgovac)
- takvi problemi daju se izraziti tako da prostor pretraživanja odgovara skupu svih "kompletnih konfiguracija" (za razliku od parcijalnih rješenja)

Postupci iterativnog poboljšanja

- Neki problemi imaju svojstvo da **put** do rješenja nije relevantan (npr, 8 kraljica, VLSI prospoj, putujući trgovac)
- takvi problemi daju se izraziti tako da prostor pretraživanja odgovara skupu svih "kompletnih konfiguracija" (za razliku od parcijalnih rješenja)
- tražimo onu konfiguraciju koja je optimalna (prospoj), ili zadovoljava sva ograničenja (diskretni optimizacijski problem!)

Postupci iterativnog poboljšanja

- Neki problemi imaju svojstvo da **put** do rješenja nije relevantan (npr, 8 kraljica, VLSI prospoj, putujući trgovac)
- takvi problemi daju se izraziti tako da prostor pretraživanja odgovara skupu svih "kompletnih konfiguracija" (za razliku od parcijalnih rješenja)
- tražimo onu konfiguraciju koja je optimalna (prospoj), ili zadovoljava sva ograničenja (diskretni optimizacijski problem!)
- prednost: u memoriji možemo držati samo tekuće stanje prostorna složenost: $O(1)$!
- npr, za problem n kraljica:
  - raspodijeliti kraljice po stupcima uz nasumični izbor redaka
  - iterativno se pomicati "u smjeru" rješenja
  - $3232 \rightarrow 3231 \rightarrow 3031 \rightarrow 2031$

Pretraživanje usponom na vrh

```
# funkcija vraća rješenje kao
# lokalni maksimum funkcije value
def usponNaVrh(problem):
  s := problem.početnoStanje()
  while true:
    s_next := max(problem.proširi(s))

    # završavamo ako je vrijednost u najboljem sljedbeniku
    # manje dobra nego u tekućem čvoru
    if (value(s_next)<value(s)):
      return s

    s := s_next
```

Pretraživanje usponom na vrh

```
# funkcija vraća rješenje kao
# lokalni maksimum funkcije value
def usponNaVrh(problem):
  s := problem.početnoStanje()
  while true:
    s_next := max(problem.proširi(s))

    # završavamo ako je vrijednost u najboljem sljedbeniku
    # manje dobra nego u tekućem čvoru
    if (value(s_next)<value(s)):
      return s

    s := s_next
```

- kao uspon na Everest kroz gustu maglu, s amnezijom!

Pretraživanje usponom na vrh

```
# funkcija vraća rješenje kao
# lokalni maksimum funkcije value
def usponNaVrh(problem):
  s := problem.početnoStanje()
  while true:
    s_next := max(problem.proširi(s))

    # završavamo ako je vrijednost u najboljem sljedbeniku
    # manje dobra nego u tekućem čvoru
    if (value(s_next)<value(s)):
      return s

    s := s_next
```

- kao uspon na Everest kroz gustu maglu, s amnezijom!
- vrste problema: brežuljak, hrbat, zaravan

Simulirano kaljenje

```
# plan ... f: vrijeme -> temperatura
def kaljenje(problem,plan):
  s := problem.početnoStanje()
  for t in 0..∞:
    T := plan(t)
    if (T==0):
      return s
    s₁ := slučajni sljedbenik od s
    ΔE := value(s₁) - value (s)
    if ΔE>0:
      s := s₁
    else:
      s := s₁, uz p = e^(ΔE/T)
```

Simulirano kaljenje

```
# plan ... f: vrijeme -> temperatura
def kaljenje(problem,plan):
  s := problem.početnoStanje()
  for t in 0..∞:
    T := plan(t)
    if (T==0):
      return s
    s_1 := slučajni sljedbenik od s
    ΔE := value(s_1) - value (s)
    if ΔE>0:
      s := s_1
    else:
      s := s_1, uz p = e^(ΔE/T)
```

- stohastičko proširenje uspona na vrh (postupak ide i “nizbrdo”)

Simulirano kaljenje

```
# plan ... f: vrijeme -> temperatura
def kaljenje(problem,plan):
  s := problem.početnoStanje()
  for t in 0..∞:
    T := plan(t)
    if (T==0):
      return s
    s_1 := slučajni sljedbenik od s
    ΔE := value(s_1) - value (s)
    if ΔE>0:
      s := s_1
    else:
      s := s_1, uz p = e^(ΔE/T)
```

- stohastičko proširenje uspona na vrh (postupak ide i "nizbrdo")
- odabir plana hlađenja jako značajan za rezultate postupka

Simulirano kaljenje

```
# plan ... f: vrijeme -> temperatura
def kaljenje(problem,plan):
  s := problem.početnoStanje()
  for t in 0..∞:
    T := plan(t)
    if (T==0):
      return s
    s_1 := slučajni sljedbenik od s
    ΔE := value(s_1) - value (s)
    if ΔE>0:
      s := s_1
    else:
      s := s_1, uz p = e^(ΔE/T)
```

- stohastičko proširenje uspona na vrh (postupak ide i "nizbrdo")
- odabir plana hlađenja jako značajan za rezultate postupka
- `value` ... ukupna energija atomâ, `T` ... temperatura

Problem zadovoljenja ograničenja

- definicija i primjeri
- izravno rješenje pretraživanjem (k1, k2)
- poboljšanje efikasnosti napuštanjem bezperspektivnih parcijalnih rješenja (k3)
- dinamička prilagodba redosljeda odabira
- postupci iterativnog poboljšanja

Problem zadovoljenja ograničenja

- standardni problem pretraživanja:
  - stanje je neprozirni objekt (opaque)
  - operacije nad stanjem:
    - `proširi()`
    - `ispitajCilj()` ili `hfun()`
    - `usporedba()`

Problem zadovoljenja ograničenja

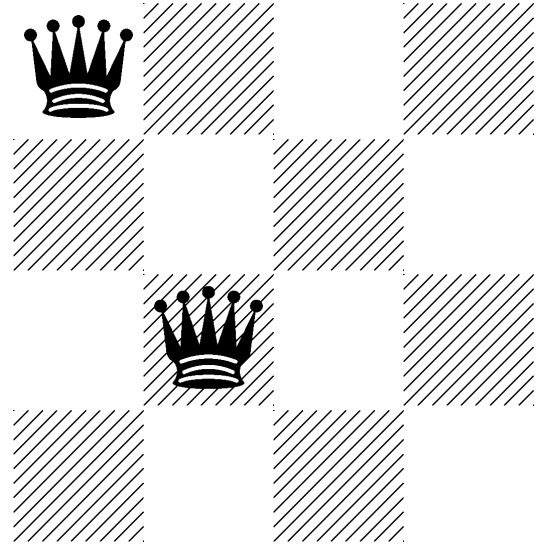
- standardni problem pretraživanja:
  - stanje je neprozirni objekt (opaque)
  - operacije nad stanjem:
    - `proširi()`
    - `ispitajCilj()` ili `hfun()`
    - `usporedba()`
- CSP:
  - **stanje**: n-torka varijabli $(V_i), V_i \in D_i$
  - **cilj**: zadovoljiti skup ograničenja koja su definirana nad podskupovima varijabli
  - put do rješenja nije bitan!

Problem zadovoljenja ograničenja

- standardni problem pretraživanja:
  - stanje je neprozirni objekt (opaque)
  - operacije nad stanjem:
    - `proširi()`
    - `ispitajCilj()` ili `hfun()`
    - `usporedba()`
- CSP:
  - **stanje**: n-torka varijabli $(V_i), V_i \in D_i$
  - **cilj**: zadovoljiti skup ograničenja koja su definirana nad podskupovima varijabli
  - put do rješenja nije bitan!
- CSP kao **specifičnija** formalizacija pretraživanja u pravilu rezultira efikasnijim postupcima

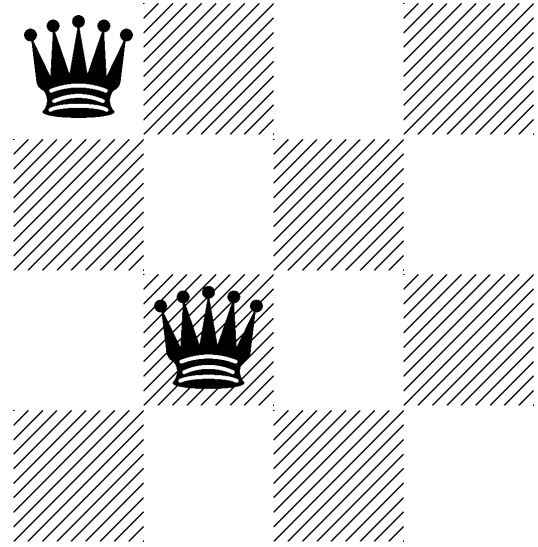
Primjer: problem n kraljica, $n = 4$

- formulacija: jedna kraljica po stupcu
  pitanje: u koji redak ide svaka od njih?



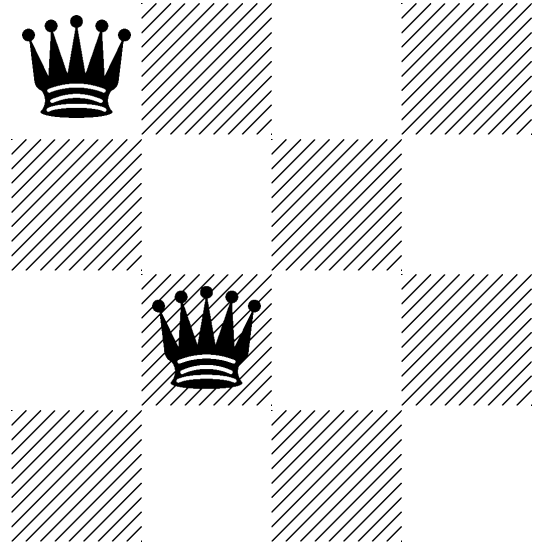
Primjer: problem n kraljica, $n = 4$

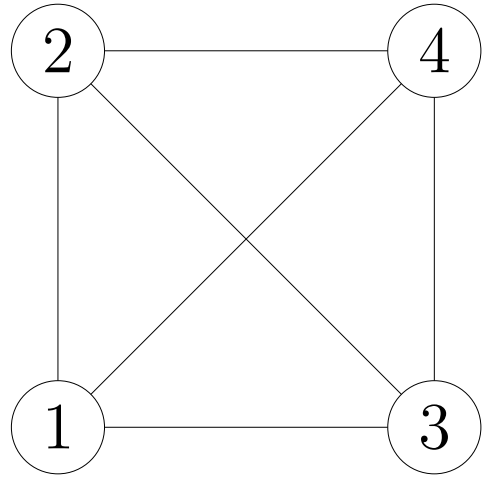
- formulacija: jedna kraljica po stupcu
  pitanje: u koji redak ide svaka od njih?
- **varijable**: $\{V_1, V_2, V_3, V_4\}$
- **domene**: $D_i = \{1, 2, 3, 4\}, \forall i$



Primjer: problem n kraljica, $n = 4$

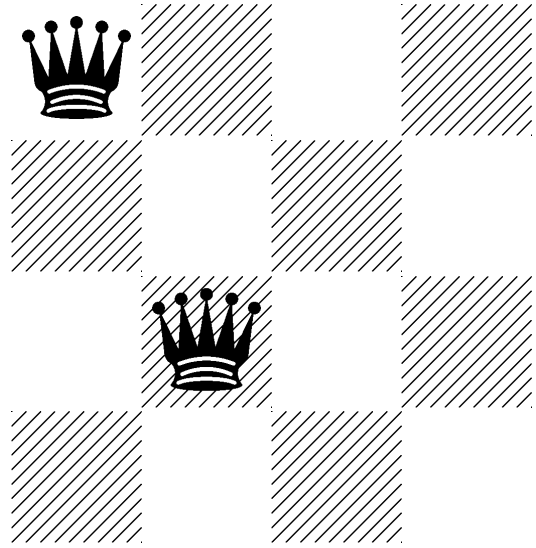
- formulacija: jedna kraljica po stupcu
  pitanje: u koji redak ide svaka od njih?
- **varijable**: $\{V_1, V_2, V_3, V_4\}$
- **domene**: $D_i = \{1, 2, 3, 4\}, \forall i$
- **ograničenja** (binarna):
  - $V_i \neq V_j$ ($\neg$ isti red)
  - $|V_i - V_j| \neq |i - j|$ ($\neg$ dijagonale)
  - binarni problem $\longrightarrow$ **graf**!

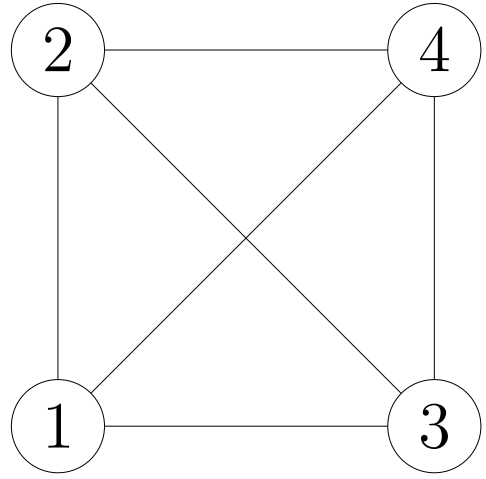




Primjer: problem n kraljica, $n = 4$

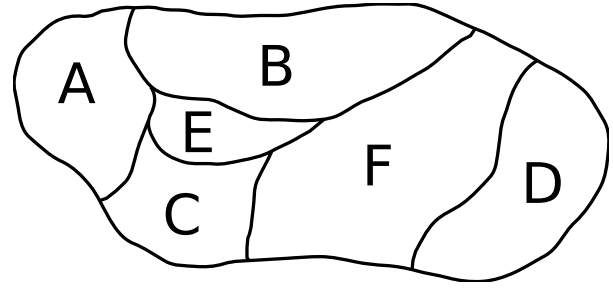
- formulacija: jedna kraljica po stupcu
  pitanje: u koji redak ide svaka od njih?
- **varijable**: $\{V_1, V_2, V_3, V_4\}$
- **domene**: $D_i = \{1, 2, 3, 4\}, \forall i$
- **ograničenja** (binarna):
  - $V_i \neq V_j$ ($\neg$ isti red)
  - $|V_i - V_j| \neq |i - j|$ ($\neg$ dijagonale)
  - binarni problem $\longrightarrow$ **graf**!
- svako ograničenje se prevodi u skup prihvatljivih vrijednosti varijabli
- npr, za (V_1, V_2):
  $(1,3)(1,4)(2,4)(3,1)(4,1)(4,2)$





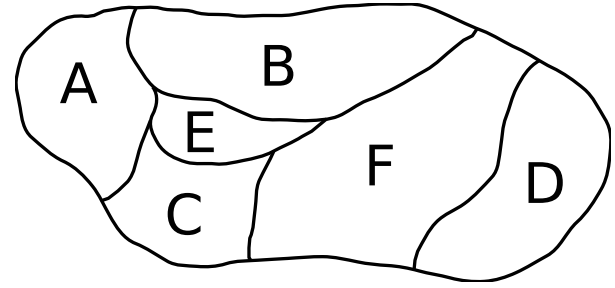
Primjer: bojanje mape

- potrebno obojati političku kartu tako da susjedne zemlje nemaju istu boju

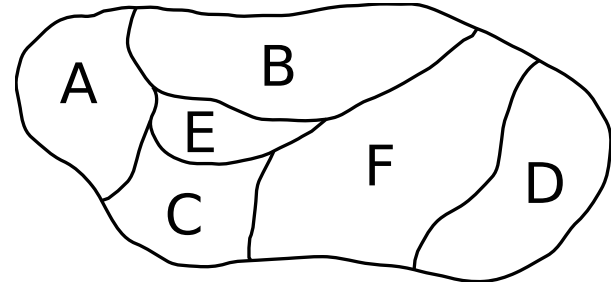


Primjer: bojanje mape

- potrebno obojati političku kartu tako da susjedne zemlje nemaju istu boju
- **varijable**: $\{A, B, C, D, E, F\}$
- **domene**:
  $D_i = \{\text{Crvena}, \text{Zelena}, \text{Plava}\}, \forall i$



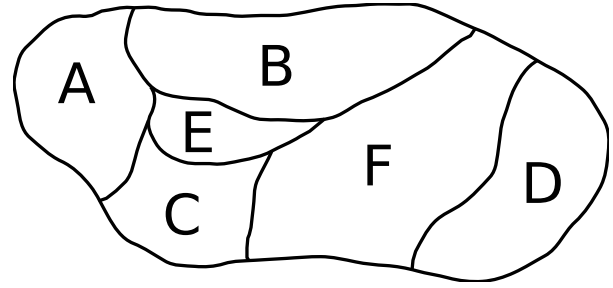
Primjer: bojanje mape

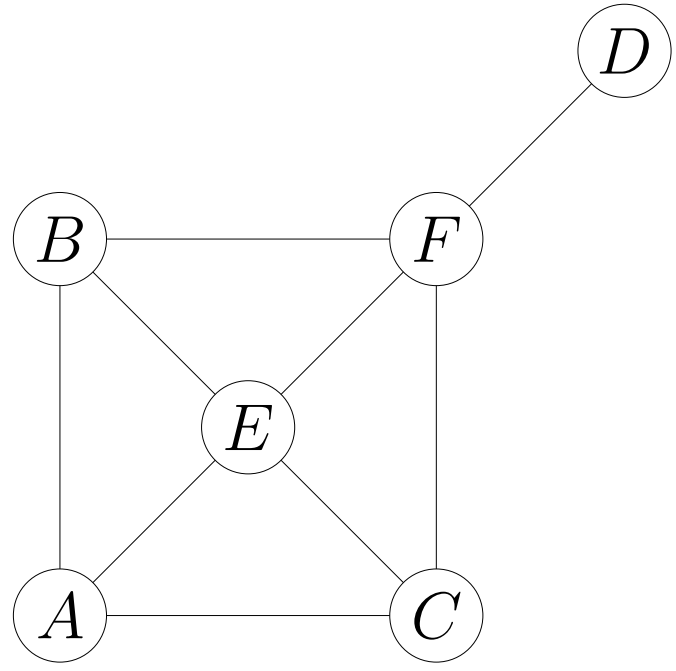


- potrebno obojati političku kartu tako da susjedne zemlje nemaju istu boju
- **varijable**: $\{A, B, C, D, E, F\}$
- **domene**:
  $D_i = \{\text{Crvena}, \text{Zelena}, \text{Plava}\}, \forall i$
- **ograničenja** (binarna):
  - $A \neq B$, $A \neq C$, ...

Primjer: bojanje mape

- potrebno obojati političku kartu tako da susjedne zemlje nemaju istu boju
- **varijable**: $\{A, B, C, D, E, F\}$
- **domene**: $D_i = \{\text{Crvena}, \text{Zelena}, \text{Plava}\}, \forall i$
- **ograničenja** (binarna):
  - $A \neq B$, $A \neq C$, ...
- odnose u binarnom problemu zorno opisuje graf ograničenja





Primjer: kriptoaritmetika

$$\begin{array}{cccccc} & & M & O & R & E \\ + & & B & U & R & A \\ \hline & S & U & N & C & E \end{array}$$

Primjer: kriptoaritmetika

$$\begin{array}{cccccc} & & M & O & R & E \\ + & & B & U & R & A \\ \hline & S & U & N & C & E \end{array} \quad \rightarrow \quad \begin{array}{cccccc} & & 4 & 7 & 3 & 5 \\ + & & 8 & 2 & 3 & 0 \\ \hline & 1 & 2 & 9 & 6 & 5 \end{array}$$

- **varijable**: $\{S, M, B, U, O, N, R, C, E, A\}$
- **domene**: $D_i = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}, \forall i$

Primjer: kriptoaritmetika

$$\begin{array}{cccccc} & & M & O & R & E \\ + & & B & U & R & A \\ \hline & S & U & N & C & E \end{array} \quad \rightarrow \quad \begin{array}{cccccc} & & 4 & 7 & 3 & 5 \\ + & & 8 & 2 & 3 & 0 \\ \hline & 1 & 2 & 9 & 6 & 5 \end{array}$$

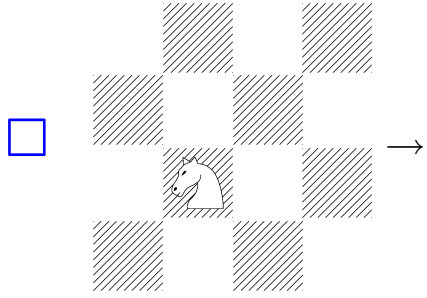
- **varijable**: $\{S, M, B, U, O, N, R, C, E, A\}$
- **domene**: $D_i = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}, \forall i$
- **ograničenja**
  - $M \neq 0, B \neq 0, S \neq 0$
  - $E = (A + E)\%10$, $(A + E > 9) \vee (C = (R + R)\%10)$, ...
  - $S \neq M$, $S \neq B$, ...

Primjer: kriptoaritmetika

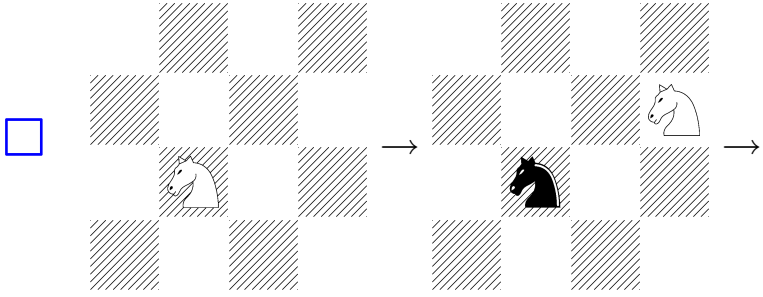
$$\begin{array}{cccccc} & & M & O & R & E \\ + & & B & U & R & A \\ \hline & S & U & N & C & E \end{array} \quad \rightarrow \quad \begin{array}{cccccc} & & 4 & 7 & 3 & 5 \\ + & & 8 & 2 & 3 & 0 \\ \hline & 1 & 2 & 9 & 6 & 5 \end{array}$$

- **varijable**: $\{S, M, B, U, O, N, R, C, E, A\}$
- **domene**: $D_i = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 0\}, \forall i$
- **ograničenja**
  - $M \neq 0, B \neq 0, S \neq 0$
  - $E = (A + E)\%10$, $(A + E > 9) \vee (C = (R + R)\%10)$, ...
  - $S \neq M$, $S \neq B$, ...

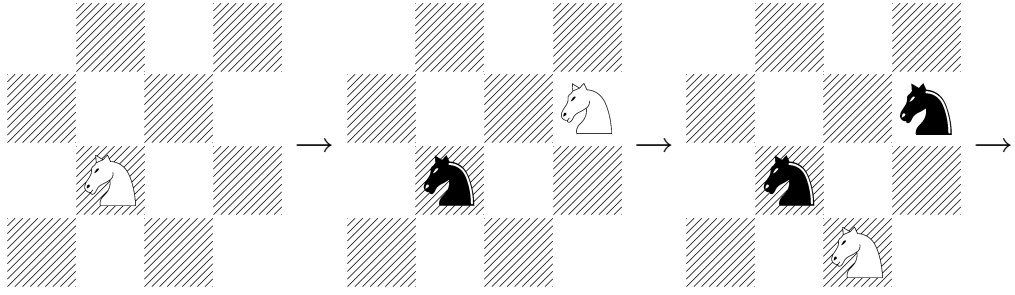
Primjer: konjićev skok



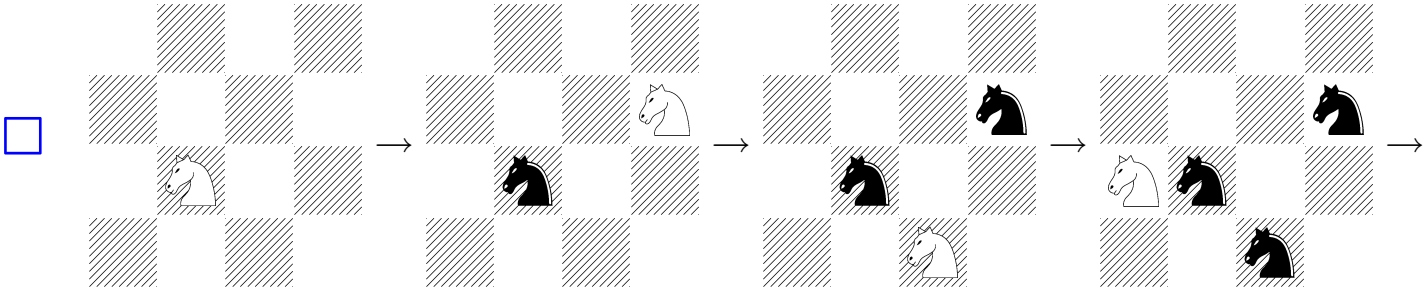
Primjer: konjićev skok



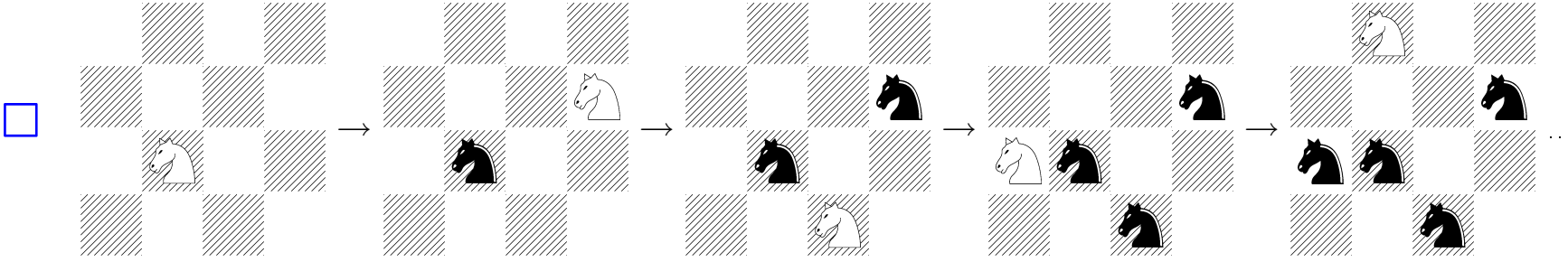
Primjer: konjićev skok



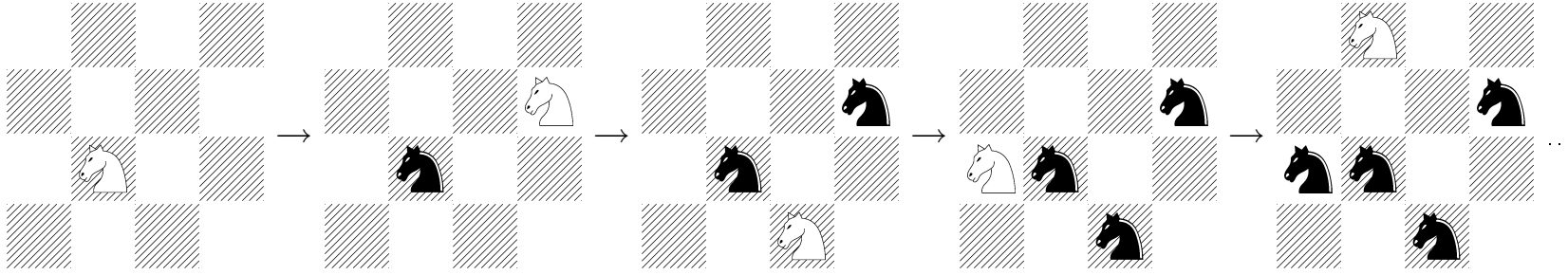
Primjer: konjićev skok



Primjer: konjićev skok

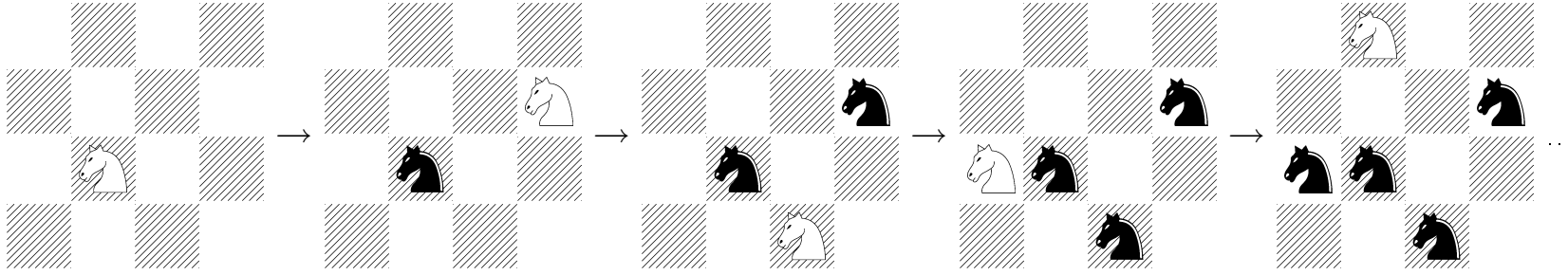


Primjer: konjićev skok

- 

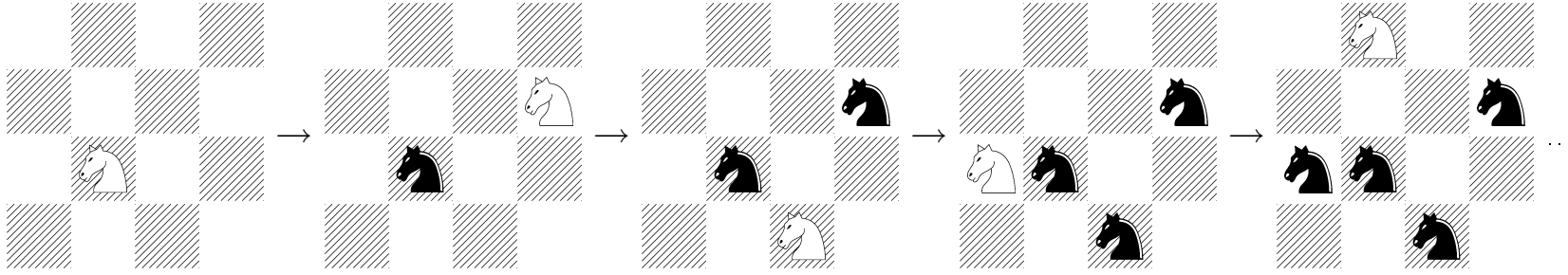
- **varijable**: $\{p_i\}, i = 1, 2, \ldots, n^2$
- **domene**: $D_i = \{1, 2, \ldots, n^2\}, \forall i$

Primjer: konjićev skok

- 

- **varijable**: $\{p_i\}, i = 1, 2, \ldots, n^2$
- **domene**: $D_i = \{1, 2, \ldots, n^2\}, \forall i$
- **ograničenja**
  - $p_i \longrightarrow p_{i+1}$: potez skakača
  - $p_i \neq p_j$

Primjer: konjićev skok

- 

- **varijable**: $\{p_i\}, i = 1, 2, \ldots, n^2$
- **domene**: $D_i = \{1, 2, \ldots, n^2\}, \forall i$
- **ograničenja**
  - $p_i \longrightarrow p_{i+1}$: potez skakača
  - $p_i \neq p_j$
- Rješenje: nema (za $n = 4$)

Stvarni primjeri problema s ograničenjima

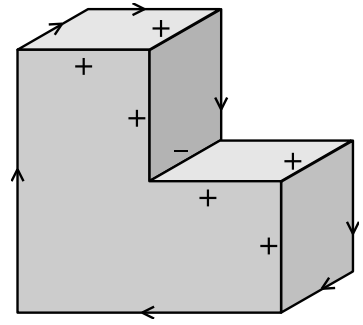
- vremensko planiranje:
  satnica: kad će se predavanja održati i gdje?
  usklađivanje resursa: proizvodnja, transport, ...

Stvarni primjeri problema s ograničenjima

- vremensko planiranje:
  satnica: kad će se predavanja održati i gdje?
  usklađivanje resursa: proizvodnja, transport, ...
- dodjeljivanje:
  koji ponuđač će obaviti koji posao?

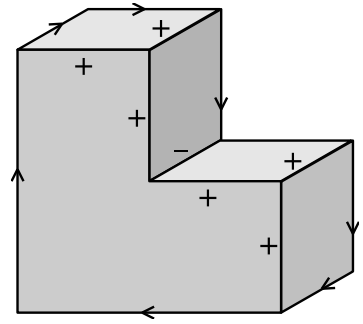
Stvarni primjeri problema s ograničenjima

- vremensko planiranje:
  satnica: kad će se predavanja održati i gdje?
  usklađivanje resursa: proizvodnja, transport, ...
- dodjeljivanje:
  koji ponuđač će obaviti koji posao?
- razumijevanje pravocrtnih crteža:



Stvarni primjeri problema s ograničenjima

- vremensko planiranje:
  satnica: kad će se predavanja održati i gdje?
  usklađivanje resursa: proizvodnja, transport, ...
- dodjeljivanje:
  koji ponuđač će obaviti koji posao?
- razumijevanje pravocrtnih crteža:



- pronalaženje vrijednosti logičkih varijabli koje zadovoljavaju logičku formulu $f = (\neg A \vee B \vee \neg C) \wedge (A \vee \neg B \vee C)$,,,

Zadovoljenje ograničenja pretraživanjem

- pristup: prikazati problem na najjednostavniji i neefikasan način, postupno poboljšavati
- **stanja**: k-torke do sada dodijeljenih varijabli, $k < n$

Zadovoljenje ograničenja pretraživanjem

- pristup: prikazati problem na najjednostavniji i neefikasan način, postupno poboljšavati
- **stanja**: k-torke do sada dodijeljenih varijabli, $k < n$
- **početno stanje**: sve varijable slobodne

Zadovoljenje ograničenja pretraživanjem

- pristup: prikazati problem na najjednostavniji i neefikasan način, postupno poboljšavati
- **stanja**: k-torke do sada dodijeljenih varijabli, $k < n$
- **početno stanje**: sve varijable slobodne
- **operator**: dodjela vrijednosti sljedećoj varijabli

Zadovoljenje ograničenja pretraživanjem

- pristup: prikazati problem na najjednostavniji i neefikasan način, postupno poboljšavati
- **stanja**: k-torke do sada dodijeljenih varijabli, $k < n$
- **početno stanje**: sve varijable slobodne
- **operator**: dodjela vrijednosti sljedećoj varijabli
- **cilj**:
  - sve varijable dodijeljene ($k = n$),
  - svi zahtjevi zadovoljeni
  - zahtjevi mogu biti prikazani eksplicitno (tablica) ili implicitno (funkcija)

Zadovoljenje ograničenja pretraživanjem

- pristup: prikazati problem na najjednostavniji i neefikasan način, postupno poboljšavati
- **stanja**: k-torke do sada dodijeljenih varijabli, $k < n$
- **početno stanje**: sve varijable slobodne
- **operator**: dodjela vrijednosti sljedećoj varijabli
- **cilj**:
  - sve varijable dodijeljene ($k = n$),
  - svi zahtjevi zadovoljeni
  - zahtjevi mogu biti prikazani eksplicitno (tablica) ili implicitno (funkcija)
- procedura je ista za sve probleme ograničenja!

Zadovoljenje ograničenja pretraživanjem

- pristup: prikazati problem na najjednostavniji i neefikasan način, postupno poboljšavati
- **stanja**: k-torke do sada dodijeljenih varijabli, $k < n$
- **početno stanje**: sve varijable slobodne
- **operator**: dodjela vrijednosti sljedećoj varijabli
- **cilj**:
  - sve varijable dodijeljene ($k = n$),
  - svi zahtjevi zadovoljeni
  - zahtjevi mogu biti prikazani eksplicitno (tablica) ili implicitno (funkcija)
- procedura je ista za sve probleme ograničenja!
- mnogi problemi su nažalost dokazano NP teški
  $\Rightarrow$ eksponencijano teški najgori slučajevi

Zadovoljenje ograničenja pretraživanjem

- svojstva, uz n – broj varijabli:
  - **maksimalna dubina stabla**: $m = n$
    dubina rješenja: $d = n$
    odlično! $\Rightarrow$ pretražujemo u dubinu!
  - ne razmatraju se dodjeljivanja koja krše ograničenja u tekućem parcijalnom rješenju! (k1$\rightarrow$k2)
  - faktor grananja (najgori slučaj) $b = |D_i|$

Zadovoljenje ograničenja pretraživanjem

- svojstva, uz n – broj varijabli:
  - **maksimalna dubina stabla**: $m = n$
    dubina rješenja: $d = n$
    odlično! $\Rightarrow$ pretražujemo u dubinu!
  - ne razmatraju se dodjeljivanja koja krše ograničenja u tekućem parcijalnom rješenju! (k1$\rightarrow$k2)
  - faktor grananja (najgori slučaj) $b = |D_i|$
- što se najgoreg slučaja tiče, uglavnom ne možemo bolje

Zadovoljenje ograničenja pretraživanjem

- svojstva, uz n – broj varijabli:
  - **maksimalna dubina stabla**: $m = n$
    dubina rješenja: $d = n$
    odlično! $\Rightarrow$ pretražujemo u dubinu!
  - ne razmatraju se dodjeljivanja koja krše ograničenja u tekućem parcijalnom rješenju! (k1$\rightarrow$k2)
  - faktor grananja (najgori slučaj) $b = |D_i|$
- što se najgoreg slučaja tiče, uglavnom ne možemo bolje
- postoje međutim strategije koje mogu dramatično ubrzati ono što nam najčešće treba, pronalazak prvog rješenja

Konzistencija parcijalnog rješenja

- ideja: odbaciti dodjeljivanja koja će onemogućiti pronalaženje rješenja u budućnosti! (k3)

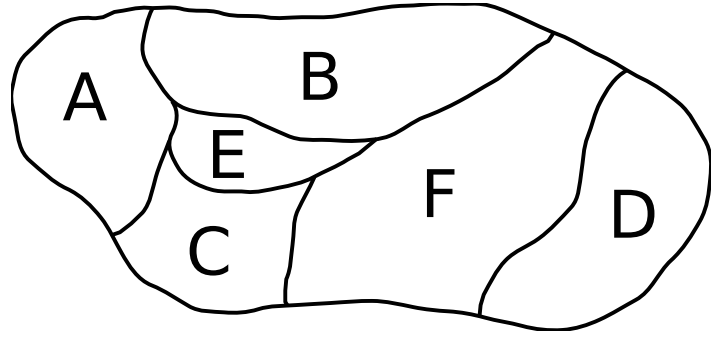
Konzistencija parcijalnog rješenja

- ideja: odbaciti dodjeljivanja koja će onemogućiti pronalaženje rješenja u budućnosti! (k3)
- za slučaj bojanja grafa: evidencija primjenljivih boja $\forall$ čvor! $\exists$ "izolirani čvor" $\Rightarrow$ rezati granu stabla pretraživanja

Konzistencija parcijalnog rješenja

- ideja: odbaciti dodjeljivanja koja će onemogućiti pronalaženje rješenja u budućnosti! (k3)
- za slučaj bojanja grafa: evidencija primjenljivih boja $\forall$ čvor! $\exists$ "izolirani čvor" $\Rightarrow$ rezati granu stabla pretraživanja

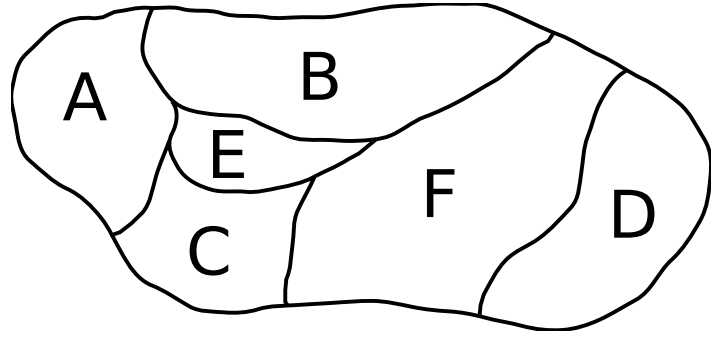
	crvena	zelena	plava
A	$\surd$		
B	$\times$		
C	$\times$		
D			
E	$\times$		
F			



Konzistencija parcijalnog rješenja

- ideja: odbaciti dodjeljivanja koja će onemogućiti pronalaženje rješenja u budućnosti! (k3)
- za slučaj bojanja grafa: evidencija primjenljivih boja $\forall$ čvor! $\exists$ "izolirani čvor" $\Rightarrow$ rezati granu stabla pretraživanja

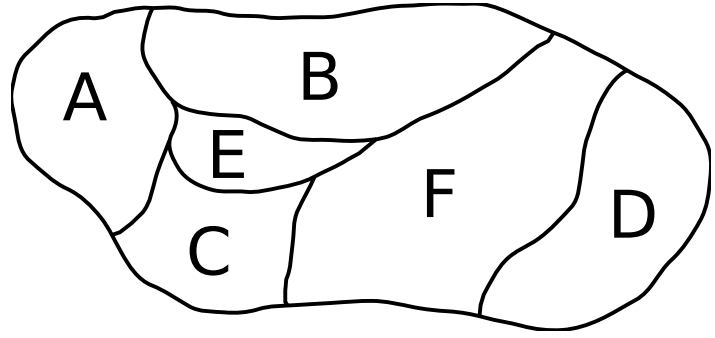
	crvena	zelena	plava
A	$\surd$	$\times$	
B	$\times$	$\times$	
C	$\times$	$\times$	
D			
E	$\times$	$\surd$	
F		$\times$	



Konzistencija parcijalnog rješenja

- ideja: odbaciti dodjeljivanja koja će onemogućiti pronalaženje rješenja u budućnosti! (k3)
- za slučaj bojanja grafa: evidencija primjenljivih boja $\forall$ čvor! $\exists$ "izolirani čvor" $\Rightarrow$ rezati granu stabla pretraživanja

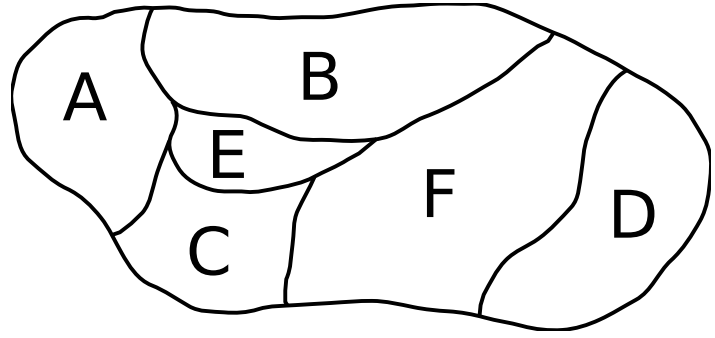
	crvena	zelena	plava
A	$\surd$	$\times$	
B	$\times$	$\times$	$\times$
C	$\times$	$\times$	$\times$
D			$\times$
E	$\times$	$\surd$	$\times$
F		$\times$	$\surd$



Konzistencija parcijalnog rješenja

- ideja: odbaciti dodjeljivanja koja će onemogućiti pronalaženje rješenja u budućnosti! (k3)
- za slučaj bojanja grafa: evidencija primjenljivih boja $\forall$ čvor!
  $\exists$ "izolirani čvor" $\Rightarrow$ rezati granu stabla pretraživanja

	crvena	zelena	plava
A	$\surd$	$\times$	
B	$\times$	$\times$	$\times$
C	$\times$	$\times$	$\times$
D			$\times$
E	$\times$	$\surd$	$\times$
F		$\times$	$\surd$



- dramatično ubrzanje prvog rješenja, unatoč većem poslu;
  npr, za 20 kraljica: t(k2)=$15{,}0$ s; t(k3)=$3{,}5$ s

Dinamička prilagodba redosljeda odabira

- do sada: čvrsti redoslijed odabira varijabli i dodjele vrijednosti

Dinamička prilagodba redosljeda odabira

- do sada: čvrsti redoslijed odabira varijabli i dodjele vrijednosti
- **obično**, prvo rješenje se brže nalazi dinamičkim odabirom

Dinamička prilagodba redosljeda odabira

- do sada: čvrsti redoslijed odabira varijabli i dodjele vrijednosti
- **obično**, prvo rješenje se brže nalazi dinamičkim odabirom
- heuristike:
  - metoda najograničenije ili najograničavajuće **varijable**: nakon provjere konzistencije, odabire se varijabla
    - s najmanjim brojem legalnih vrijednosti

Dinamička prilagodba redosljeda odabira

- do sada: čvrsti redoslijed odabira varijabli i dodjele vrijednosti
- **obično**, prvo rješenje se brže nalazi dinamičkim odabirom
- heuristike:
  - metoda najograničenije ili najograničavajuće **varijable**: nakon provjere konzistencije, odabire se varijabla
    - s najmanjim brojem legalnih vrijednosti
    - koja najviše ograničava ostale varijable

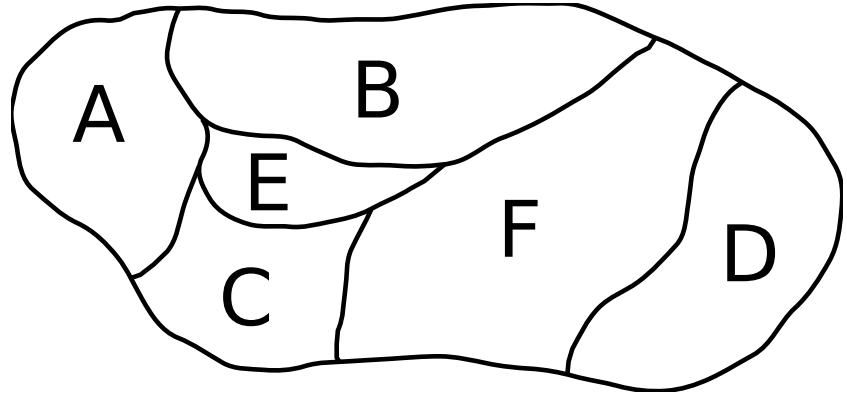
Dinamička prilagodba redosljeda odabira

- do sada: čvrsti redoslijed odabira varijabli i dodjele vrijednosti
- **obično**, prvo rješenje se brže nalazi dinamičkim odabirom
- heuristike:
  - metoda najograničenije ili najograničavajuće **varijable**: nakon provjere konzistencije, odabire se varijabla
    - s najmanjim brojem legalnih vrijednosti
    - koja najviše ograničava ostale varijable
  - metoda najmanje ograničavajuće **vrijednosti**: to je ona koja najmanje ograničava ostale varijable

Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

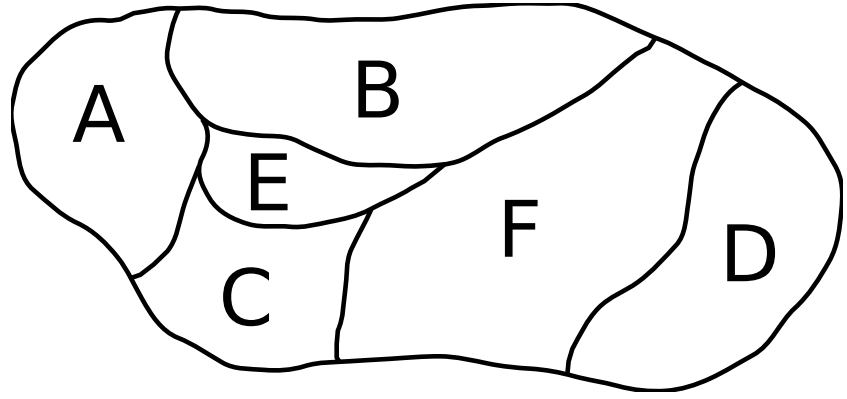
	crvena	zelena	plava
A	×		
B	×		
C	×		
D			
E	√		
F	×		



Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

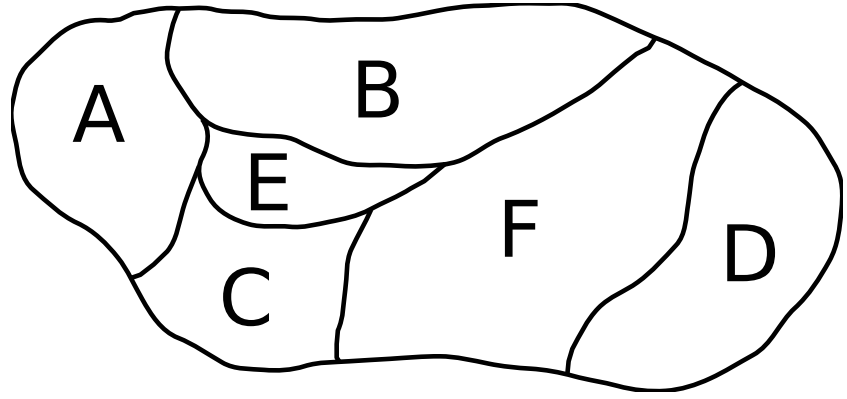
	crvena	zelena	plava
A	×		
B	×	×	
C	×	×	
D		×	
E	√	×	
F	×	√	



Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

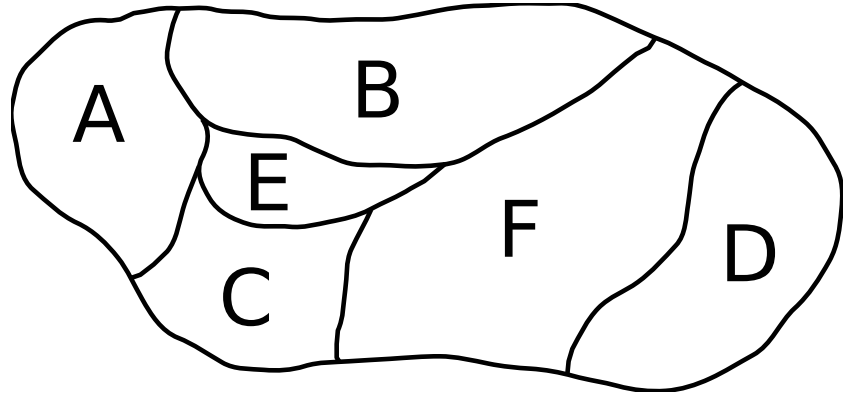
	crvena	zelena	plava
A	×		×
B	×	×	
C	×	×	√
D		×	
E	√	×	×
F	×	√	×



Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

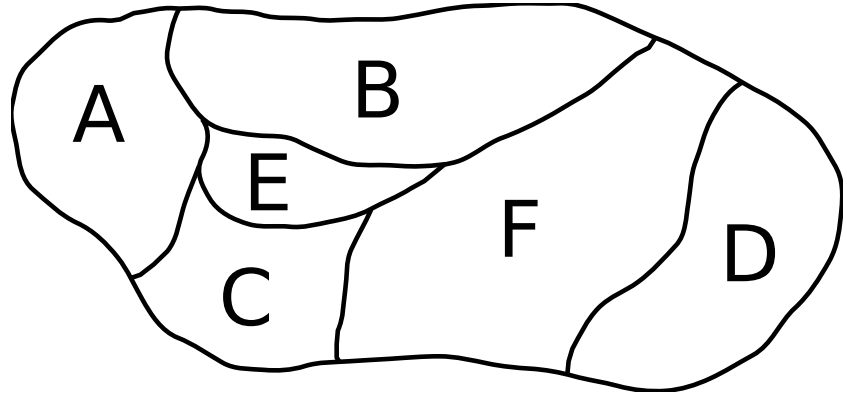
	crvena	zelena	plava
A	×		×
B	×	×	√
C	×	×	√
D		×	
E	√	×	×
F	×	√	×



Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

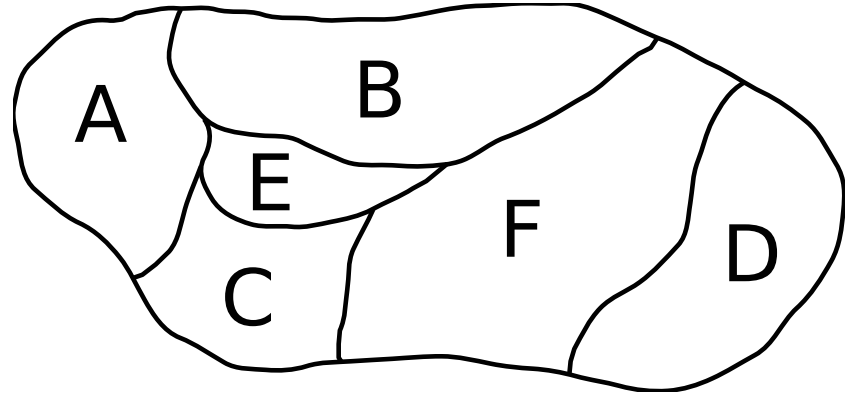
	crvena	zelena	plava
A	×	√	×
B	×	×	√
C	×	×	√
D		×	
E	√	×	×
F	×	√	×



Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

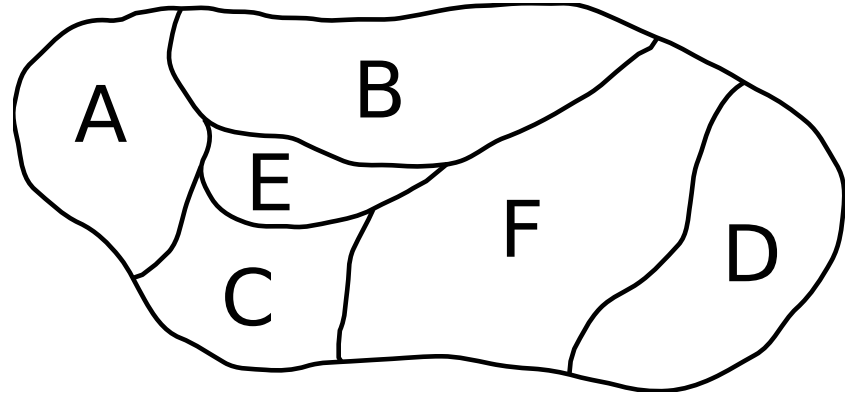
	crvena	zelena	plava
A	×	√	×
B	×	×	√
C	×	×	√
D		×	√
E	√	×	×
F	×	√	×



Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

	crvena	zelena	plava
A	×	√	×
B	×	×	√
C	×	×	√
D		×	√
E	√	×	×
F	×	√	×

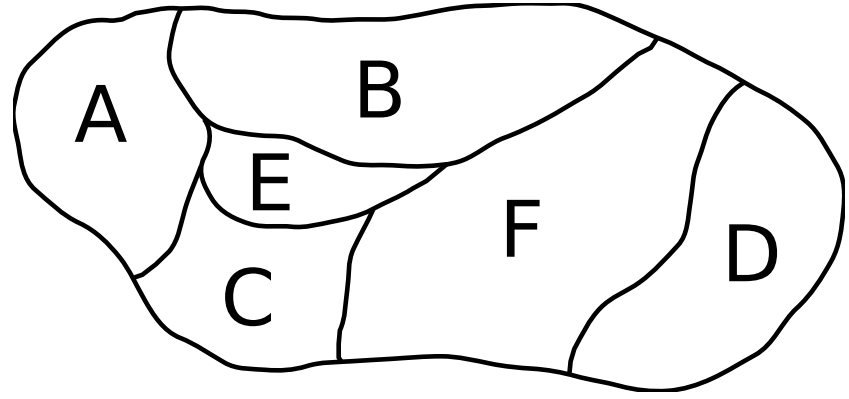


- metoda najmanje ograničavajuće vrijednosti:
  A=crvena, B=zelena, C=?

Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

	crvena	zelena	plava
A	×	√	×
B	×	×	√
C	×	×	√
D		×	√
E	√	×	×
F	×	√	×

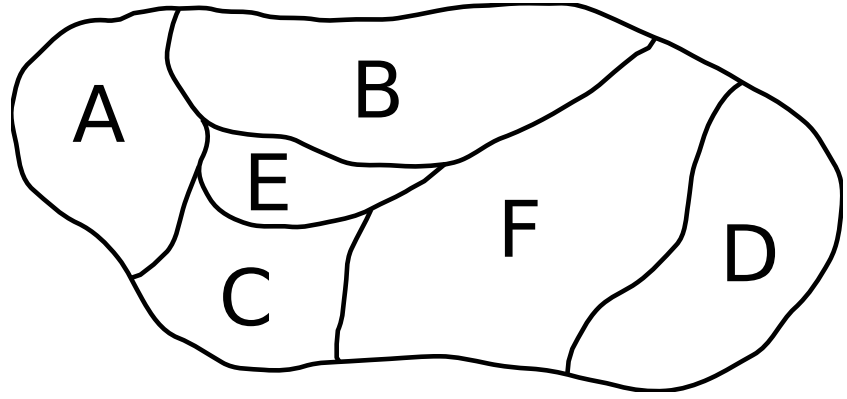


- metoda najmanje ograničavajuće vrijednosti:
  A=crvena, B=zelena, C= zelena

Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

	crvena	zelena	plava
A	×	√	×
B	×	×	√
C	×	×	√
D		×	√
E	√	×	×
F	×	√	×

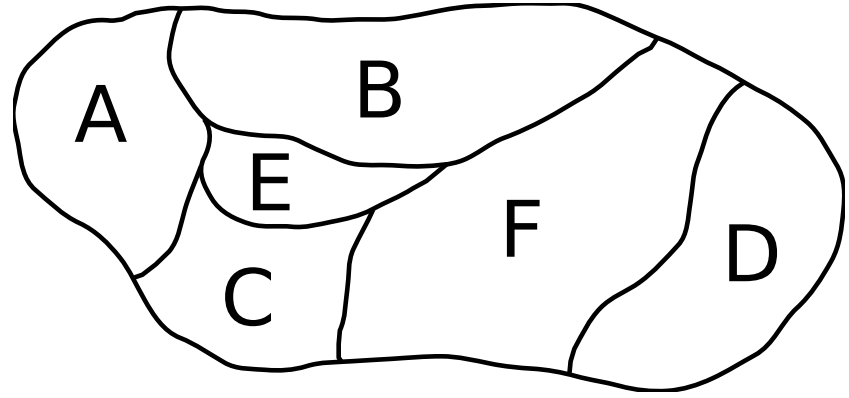


- metoda najmanje ograničavajuće vrijednosti:
  A=crvena, B=zelena, C= zelena
- metoda najograničenije varijable:
  A=crvena, B=zelena, ?

Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

	crvena	zelena	plava
A	×	√	×
B	×	×	√
C	×	×	√
D		×	√
E	√	×	×
F	×	√	×

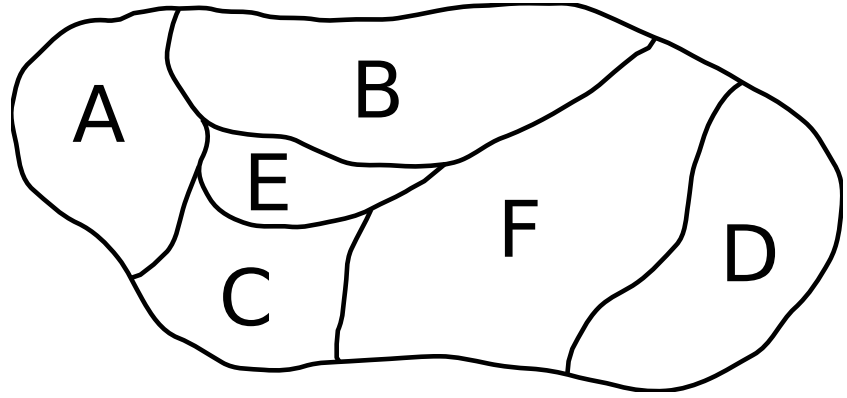


- metoda najmanje ograničavajuće vrijednosti:
  A=crvena, B=zelena, C= zelena
- metoda najograničenije varijable:
  A=crvena, B=zelena, E=plava

Dinamička prilagodba redosljeda: primjer

- metoda najograničavajuće varijable:

	crvena	zelena	plava
A	×	√	×
B	×	×	√
C	×	×	√
D		×	√
E	√	×	×
F	×	√	×



- metoda najmanje ograničavajuće vrijednosti:
  A=crvena, B=zelena, C= zelena
- metoda najograničenije varijable:
  A=crvena, B=zelena, E=plava , C=zelena, F=crvena,
  D=plava/zelena

Postupci inkrementalnog poboljšanja

- primjena metoda uspona na vrh ili simuliranog kaljenja

Postupci inkrementalnog poboljšanja

- primjena metoda uspona na vrh ili simuliranog kaljenja
- polazi se od stanja u kojem sve varijable imaju vrijednosti, uz povrijeđena ograničenja

Postupci inkrementalnog poboljšanja

- primjena metoda uspona na vrh ili simuliranog kaljenja
- polazi se od stanja u kojem sve varijable imaju vrijednosti, uz povrijeđena ograničenja
- početno stanje dobiveno slučajnim odabirom ili nekom brzom pohlepnom metodom

Postupci inkrementalnog poboljšanja

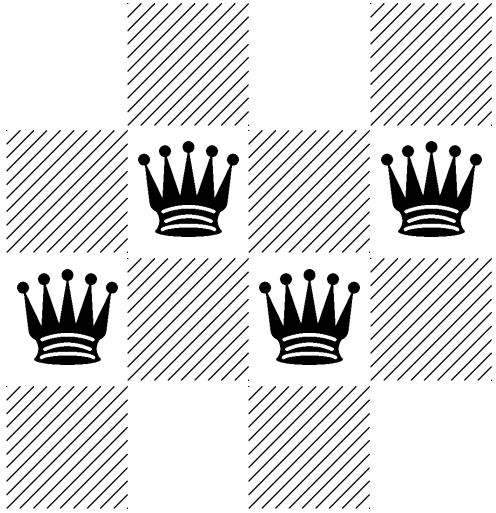
- primjena metoda uspona na vrh ili simuliranog kaljenja
- polazi se od stanja u kojem sve varijable imaju vrijednosti, uz povrijeđena ograničenja
- početno stanje dobiveno slučajnim odabirom ili nekom brzom pohlepnom metodom
- heuristika najmanjeg konflikta:
  - odaberi varijablu slučajno ili kao onu koja krši najviše ograničenja
  - dodijeli joj vrijednost koja krši najmanje ograničenja
  - ekvivalentno usponu na vrh, uz h(n)=broj konflikta

Heuristika najmanjeg konflikta – primjer

- problem 4 kraljice: (q_1, q_2, q_3, q_4), $q_1 \in \{1, 2, 3, 4\}$

Heuristika najmanjeg konflikta – primjer

- problem 4 kraljice: (q_1, q_2, q_3, q_4), $q_1 \in \{1, 2, 3, 4\}$

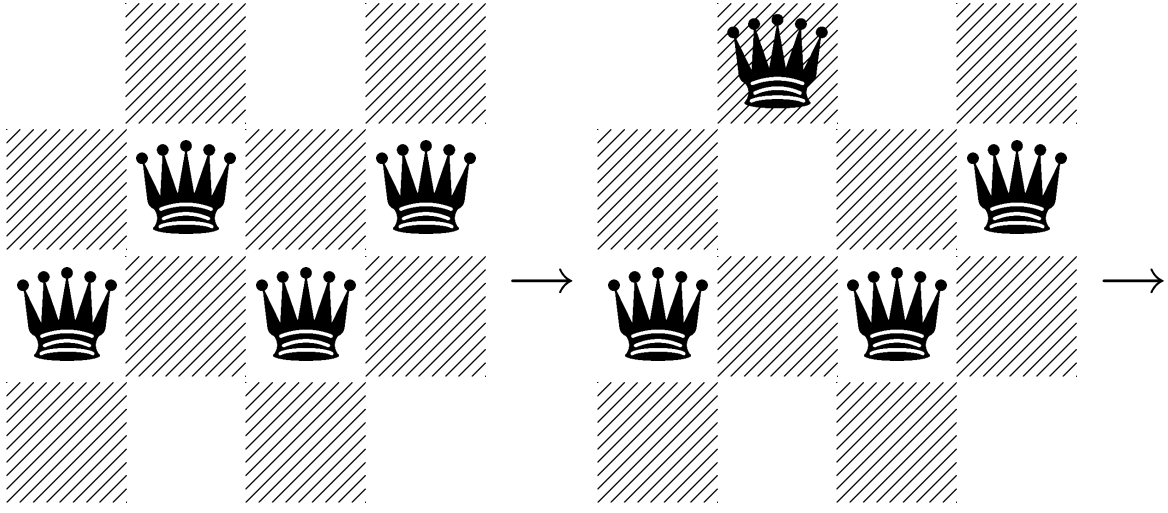


$h = 5$

$\longrightarrow$

Heuristika najmanjeg konflikta – primjer

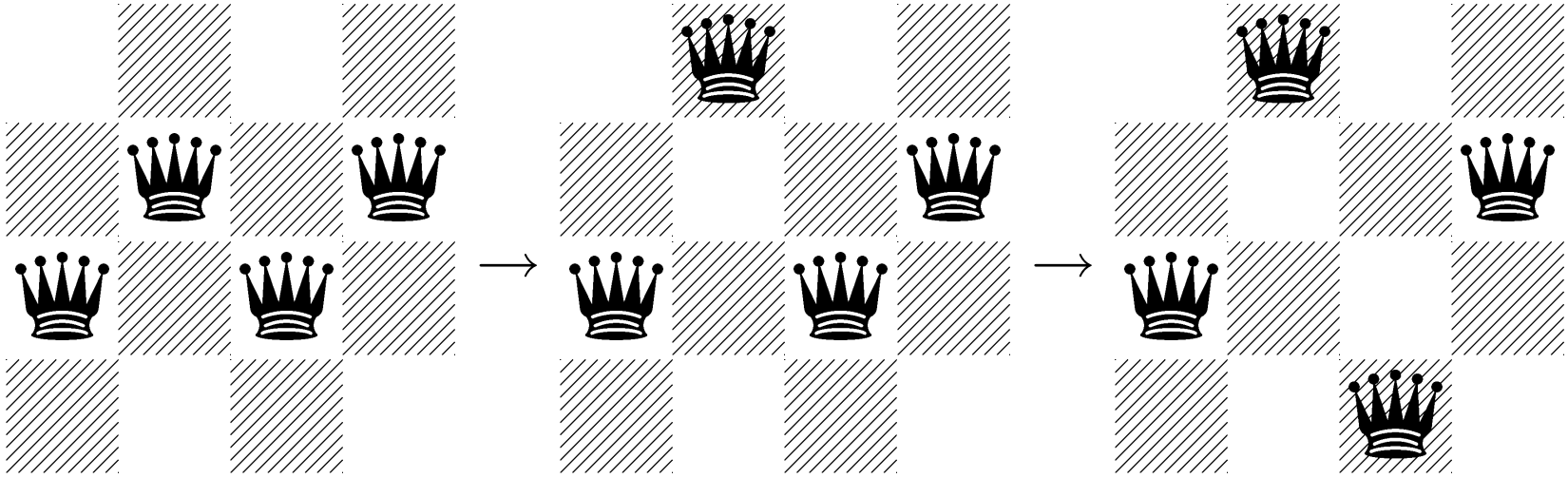
- problem 4 kraljice: (q_1, q_2, q_3, q_4), $q_1 \in \{1, 2, 3, 4\}$



$h = 5$ $h = 2$

Heuristika najmanjeg konflikta – primjer

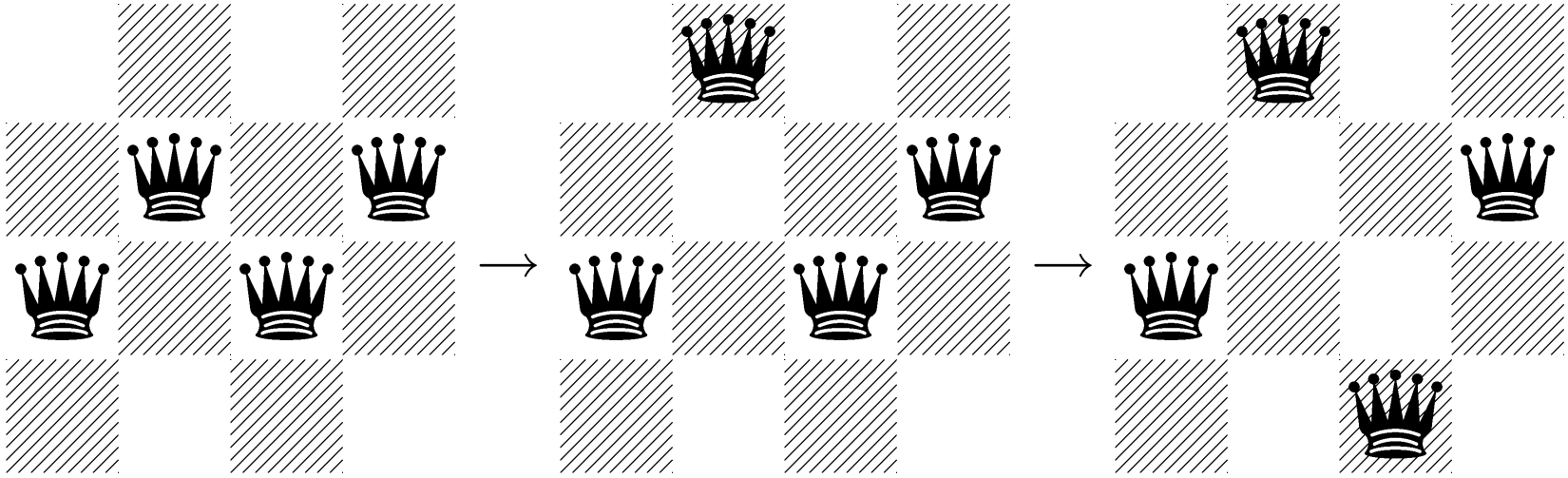
- problem 4 kraljice: (q_1, q_2, q_3, q_4), $q_1 \in \{1, 2, 3, 4\}$



$h = 5$ $h = 2$ $h = 0$

Heuristika najmanjeg konflikta – primjer

- problem 4 kraljice: (q_1, q_2, q_3, q_4), $q_1 \in \{1, 2, 3, 4\}$

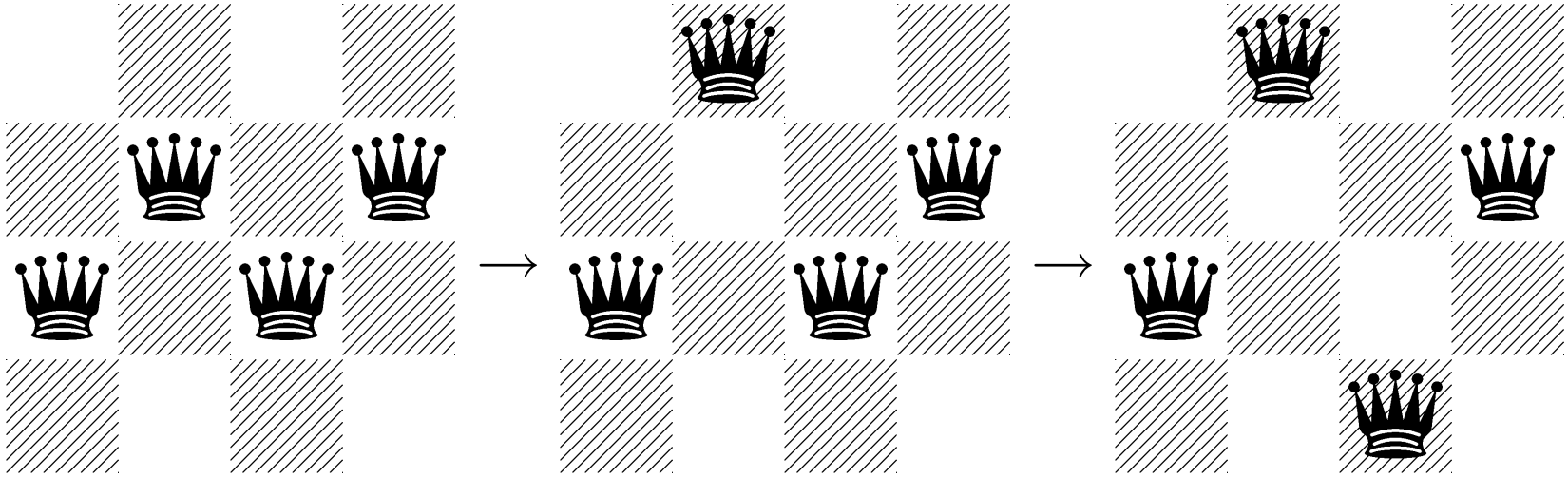


$h = 5$ $h = 2$ $h = 0$

- ovakva metoda rješava probleme do $n = 10^7$

Heuristika najmanjeg konflikta – primjer

- problem 4 kraljice: (q_1, q_2, q_3, q_4), $q_1 \in \{1, 2, 3, 4\}$

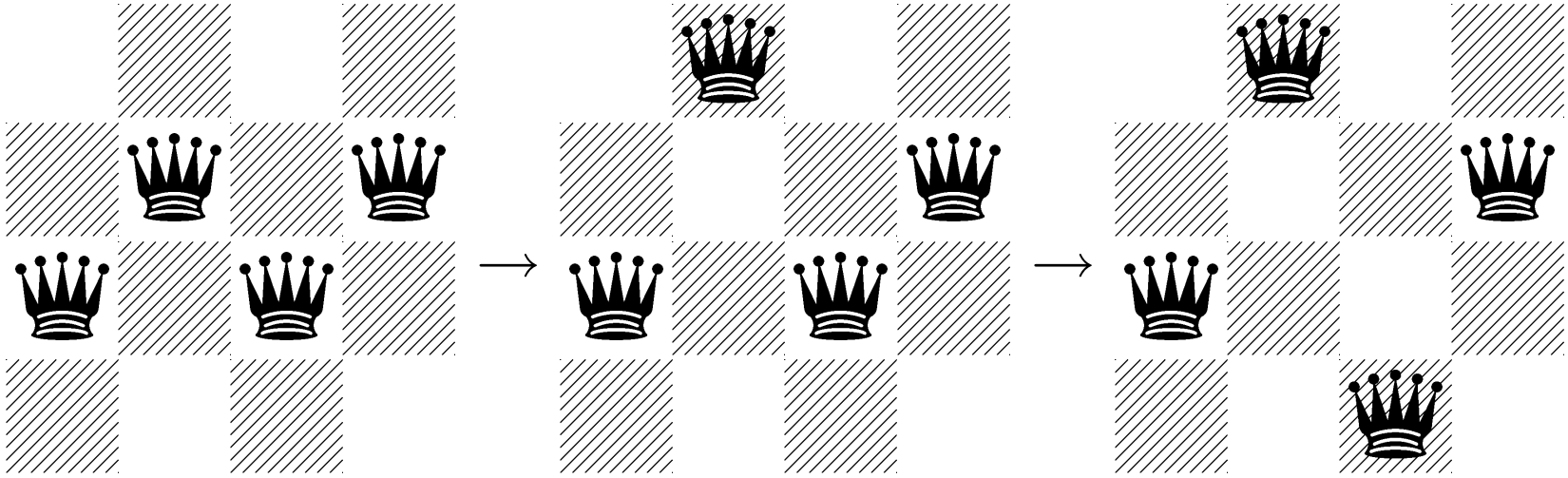


$h = 5$ $h = 2$ $h = 0$

- ovakva metoda rješava probleme do $n = 10^7$
- deterministički pristupi za nekoliko redova veličine slabiji!

Heuristika najmanjeg konflikta – primjer

- problem 4 kraljice: (q_1, q_2, q_3, q_4), $q_1 \in \{1, 2, 3, 4\}$



$h = 5$ $h = 2$ $h = 0$

- ovakva metoda rješava probleme do $n = 10^7$
- deterministički pristupi za nekoliko redova veličine slabiji!

Zadovoljenje ograničenja: sažetak

- CSP – posebna vrsta problema:
  - **stanja** definirana **vrijednostima varijabli**
  - **početno stanje**: sve varijable slobodne
  - **cilj** definiran postavljenim **ograničenjima** na varijable
  - obično nas ne interesira put do cilja

Zadovoljenje ograničenja: sažetak

- CSP – posebna vrsta problema:
  - **stanja** definirana **vrijednostima varijabli**
  - **početno stanje**: sve varijable slobodne
  - **cilj** definiran postavljenim **ograničenjima** na varijable
  - obično nas ne interesira put do cilja
- osnovna metoda (k2): pretraživanje u **dubinu** s:
  - čvrstim poretkom razmatranja varijabli
  - razmatranjem samo **legalnih** parcijalnih rješenja

Zadovoljenje ograničenja: sažetak

- CSP – posebna vrsta problema:
  - **stanja** definirana **vrijednostima varijabli**
  - **početno stanje**: sve varijable slobodne
  - **cilj** definiran postavljenim **ograničenjima** na varijable
  - obično nas ne interesira put do cilja
- osnovna metoda (k2): pretraživanje u **dubinu** s:
  - čvrstim poretkom razmatranja varijabli
  - razmatranjem samo **legalnih** parcijalnih rješenja
- **provjera konzistencije ograničenja** ne dozvoljava dodjele koje garantiraju kasniji nesupjeh (k3)

Zadovoljenje ograničenja: sažetak

- CSP – posebna vrsta problema:
  - **stanja** definirana **vrijednostima varijabli**
  - **početno stanje**: sve varijable slobodne
  - **cilj** definiran postavljenim **ograničenjima** na varijable
  - obično nas ne interesira put do cilja
- osnovna metoda (k2): pretraživanje u **dubinu** s:
  - čvrstim poretkom razmatranja varijabli
  - razmatranjem samo **legalnih** parcijalnih rješenja
- **provjera konzistencije ograničenja** ne dozvoljava dodjele koje garantiraju kasniji nesupjeh (k3)
- heuristike **poretka varijabli** i **odabira vrijednosti** mogu dramatično poboljšati efikasnost

Zadovoljenje ograničenja: sažetak

- CSP – posebna vrsta problema:
  - **stanja** definirana **vrijednostima varijabli**
  - **početno stanje**: sve varijable slobodne
  - **cilj** definiran postavljenim **ograničenjima** na varijable
  - obično nas ne interesira put do cilja
- osnovna metoda (k2): pretraživanje u **dubinu** s:
  - čvrstim poretkom razmatranja varijabli
  - razmatranjem samo **legalnih** parcijalnih rješenja
- **provjera konzistencije ograničenja** ne dozvoljava dodjele koje garantiraju kasniji nesupjeh (k3)
- heuristike **poretka varijabli** i **odabira vrijednosti** mogu dramatično poboljšati efikasnost
- iterativni pristup uz **najmanji konflikt** često najefikasniji