# Metode de cautare informata

Catalin Stoean

catalin.stoean@inf.ucv.ro

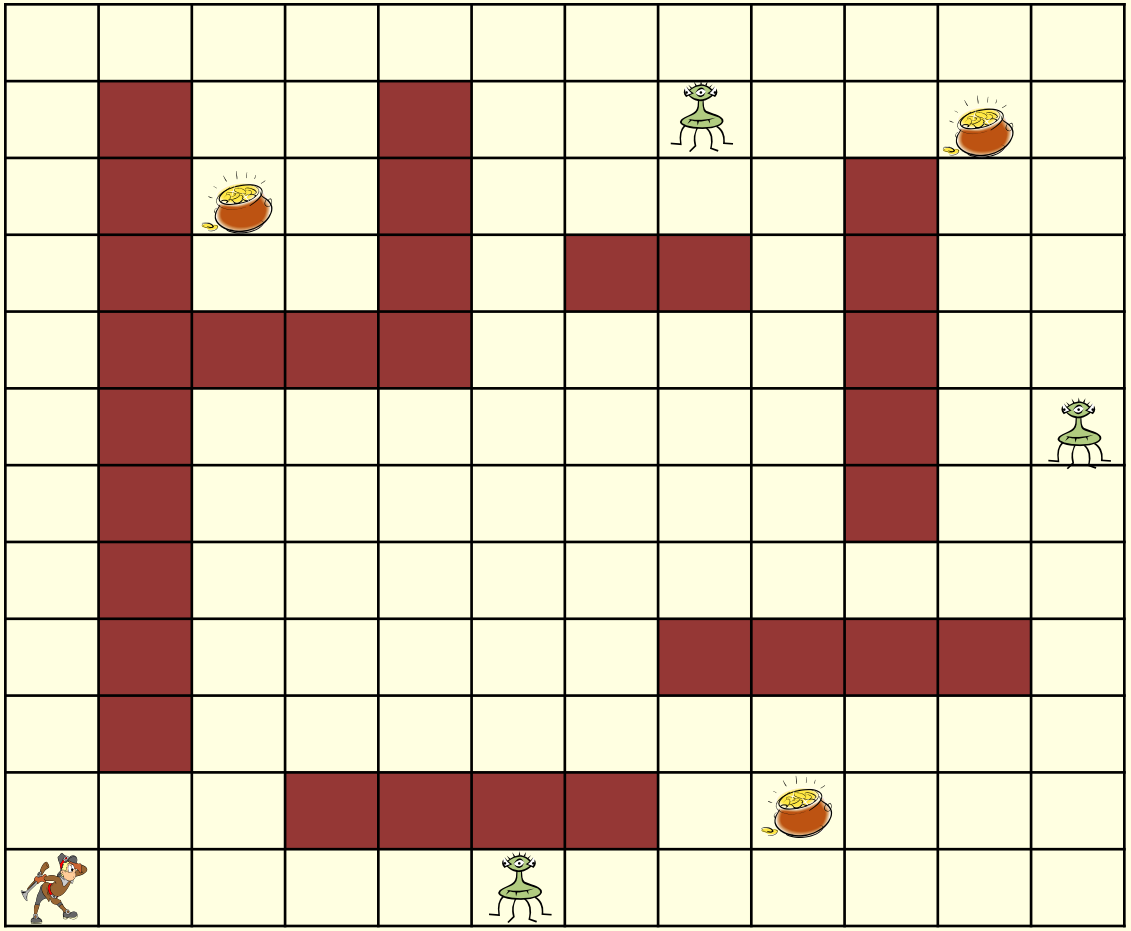
http://inf.ucv.ro/~cstoean

# Cursul anterior

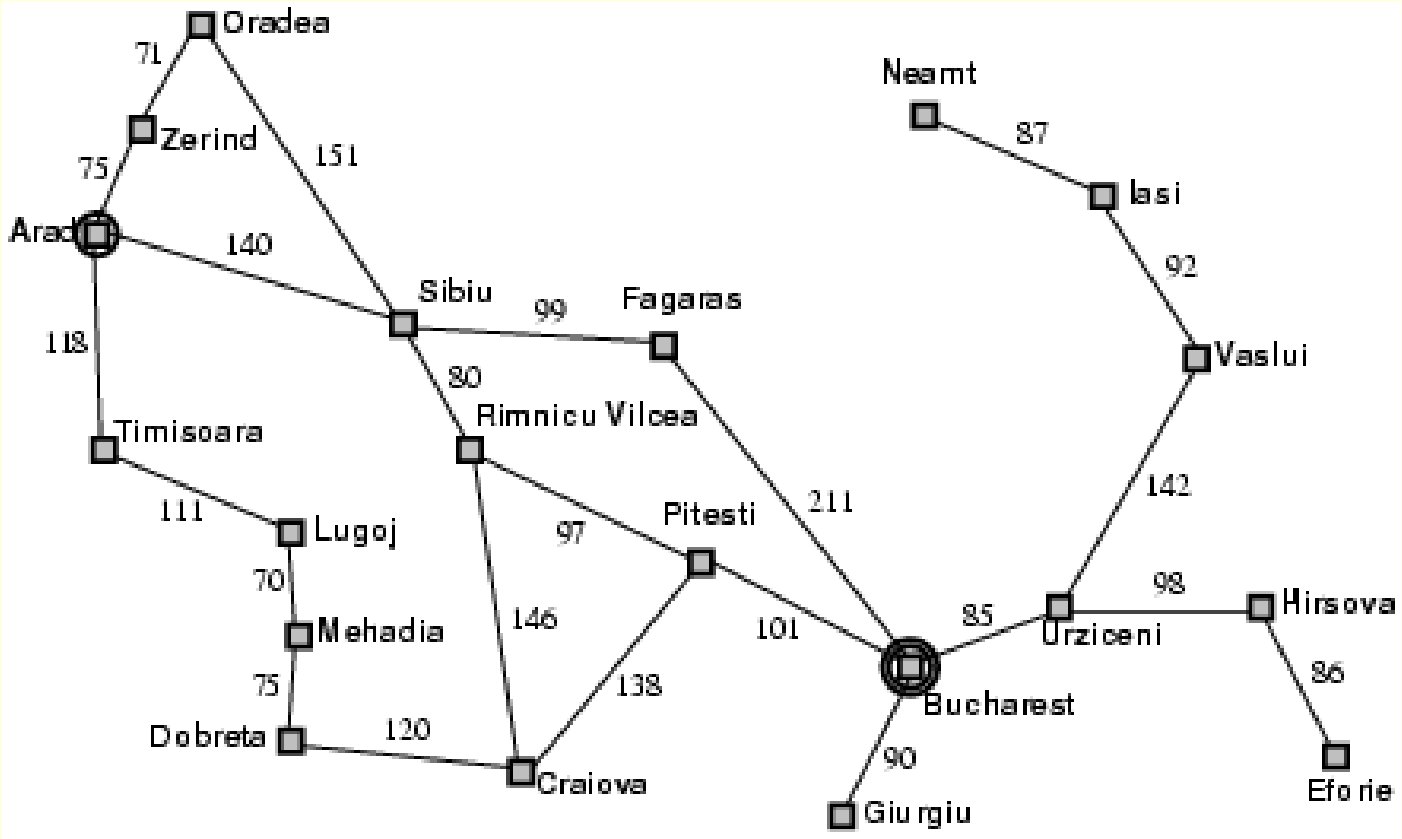
- Metode de **cautare neinformata** (blind search)
  - Solutiile la probleme sunt gasite prin generarea sistematica de stari noi si verificarea daca s-a ajuns la starea tinta (solutia problemei).
  - O strategie de cautare este data de **ordinea de expandare a nodurilor**.
  - In cele mai multe cazuri, acestea sunt ineficiente…
- In strategiile de **cautare informata**, se utilizeaza cunostinte specifice problemei pentru a gasi solutiile in mod eficient.

# Cursul anterior

Apropos…
tema?



# Cautarea informata



# Cautarea informata

- Folosind un algoritm general de cautare, cunostintele aditionale despre problema pot fi adaugate la functia care stabileste ce noduri vor fi expandate.
- Ideea este de a se utiliza o **functie de evaluare** $f(n)$ pentru fiecare nod $n$.
  - Ajuta la luarea de decizii in expandarea nodurilor.
  - Se ordoneaza nodurile astfel incat cel cu cea mai buna evaluare este expandat primul (strategia de cautare **intai cel mai bun**).

# Strategia *intai cel mai bun*

Functia de evaluare

***functia*** cautare_intai_best(*problema, eval*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))
*f_coada* = o functie care ordoneaza nodurile dupa *eval*

*Cat timp* solutie negasita si noduri ≠ ∅ *executa*
nod = scoate_din_fata(noduri)
*Daca* testare_tinta[problema] se aplica la stare(nod) *atunci*
***intoarce*** nod
*Altfel*
noduri = adauga(noduri, expandare(nod, *f_coada*))
*Sfarsit cat timp*

# Strategia *intai cel mai bun*

- Pentru a directiona cautarea, masura de evaluare trebuie sa incorporeze o masura a costului drumului de la starea curenta pana la starea tinta.

- Doua abordari:
  - Cautarea Greedy
  - Cautarea A*

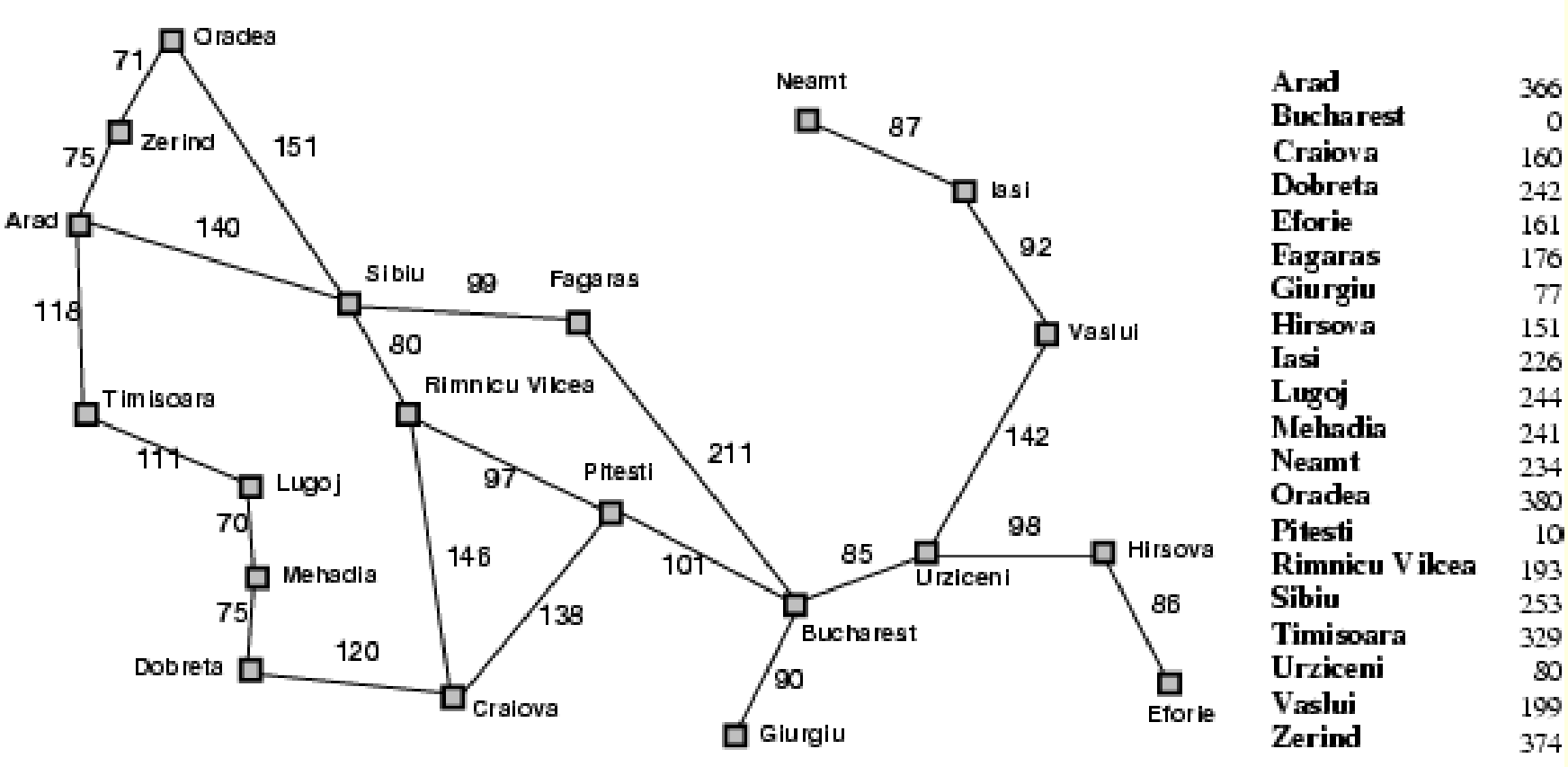
# Cautarea Greedy

- Se bazeaza pe faptul ca trebuie minimizat costul de ajungere la nodul tinta.
- Concluzie: nodul care reprezinta starea care este cea mai aproape de starea tinta este intotdeauna expandat primul.
- La cele mai multe probleme, costul de a ajunge de la o stare la starea finala nu poate determinat exact, doar estimat.
- O functie care estimeaza astfel de costuri se numeste **functie euristica** si este notata de obicei cu *h*.

# Cautarea Greedy

- ***h*(*n*)** = costul estimat pentru cel mai scurt drum de la nodul *n* pana la starea tinta.
  - Daca *n* este chiar nodul tinta, atunci *h*(*n*) = 0.
- O cautare *intai cel mai bun* care utilizeaza asemenea functie euristica se numeste **cautare greedy**.

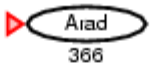
# Avem distantele pana la Bucuresti

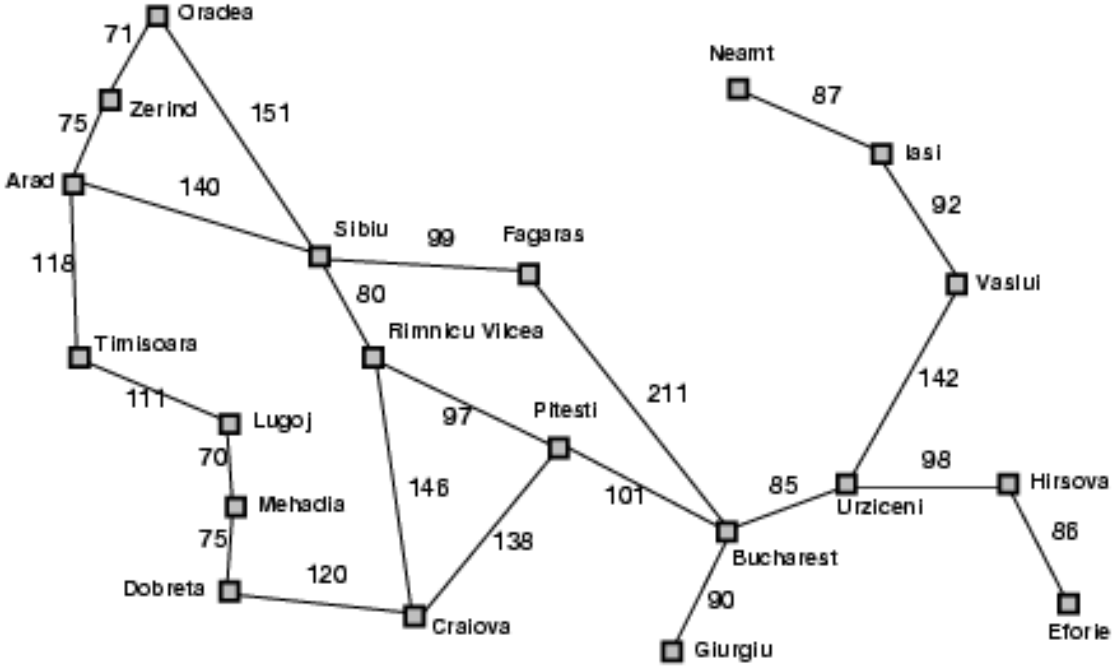



| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

- $h(n)$ = distanta in linie dreapta de la orasul $n$ pana la Bucuresti.

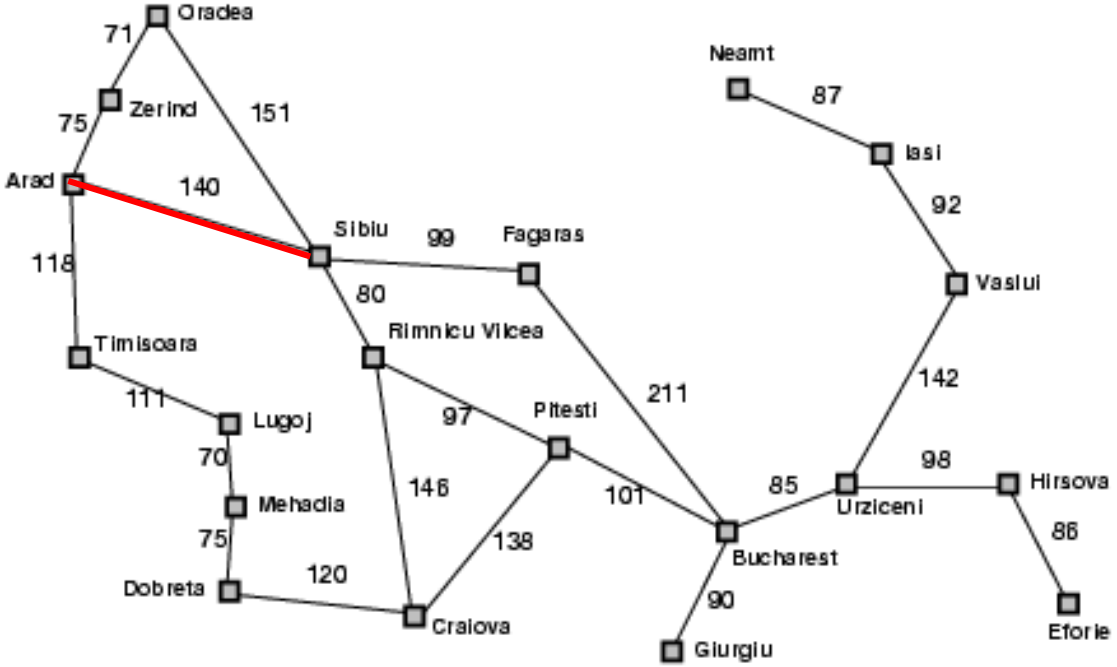
# Cautarea Greedy



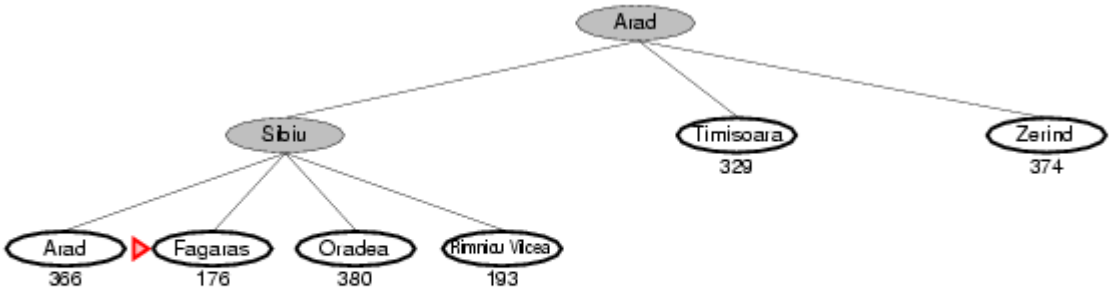

| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

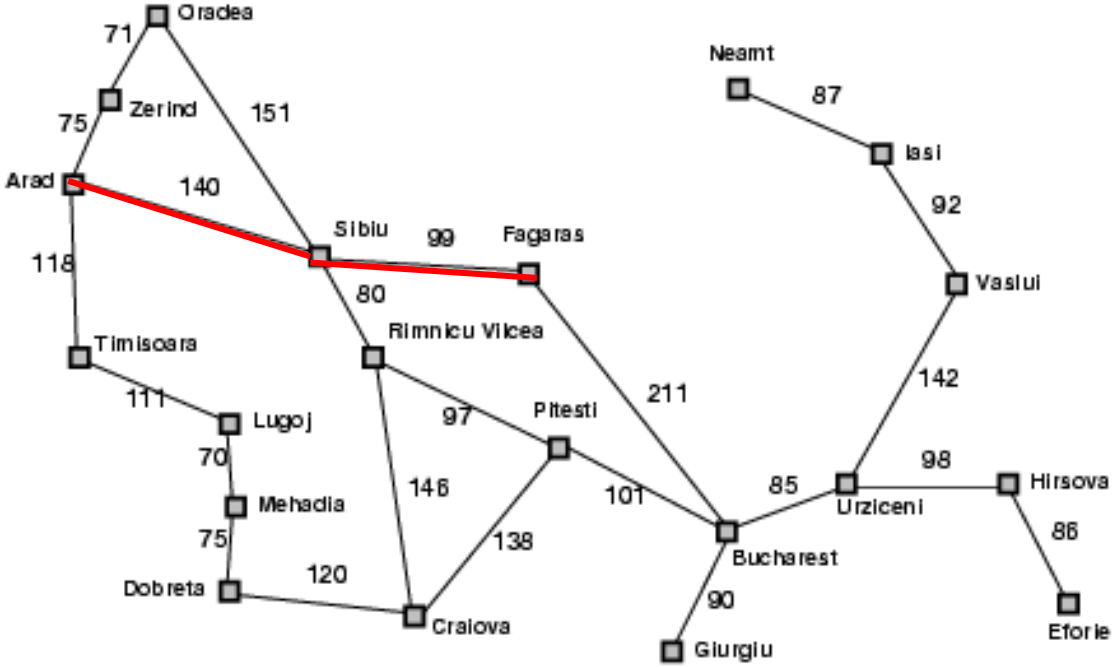
# Cautarea Greedy


| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

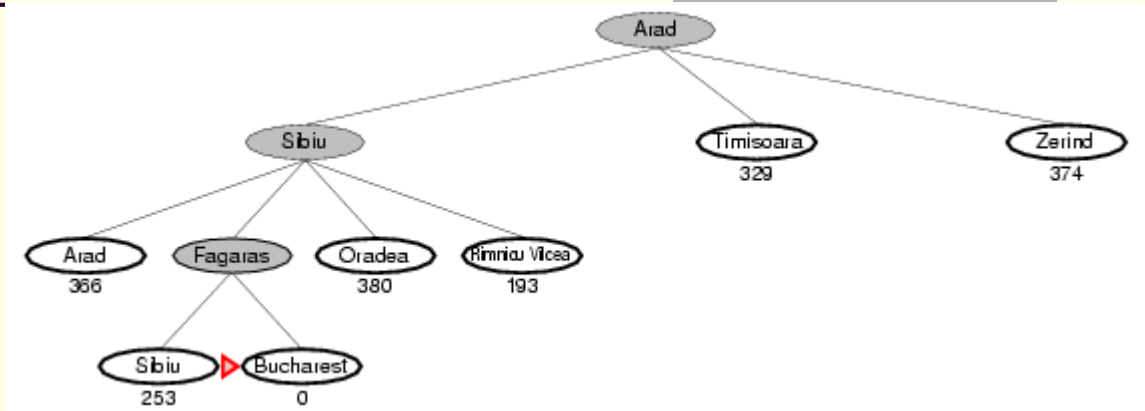
# Cautarea Greedy

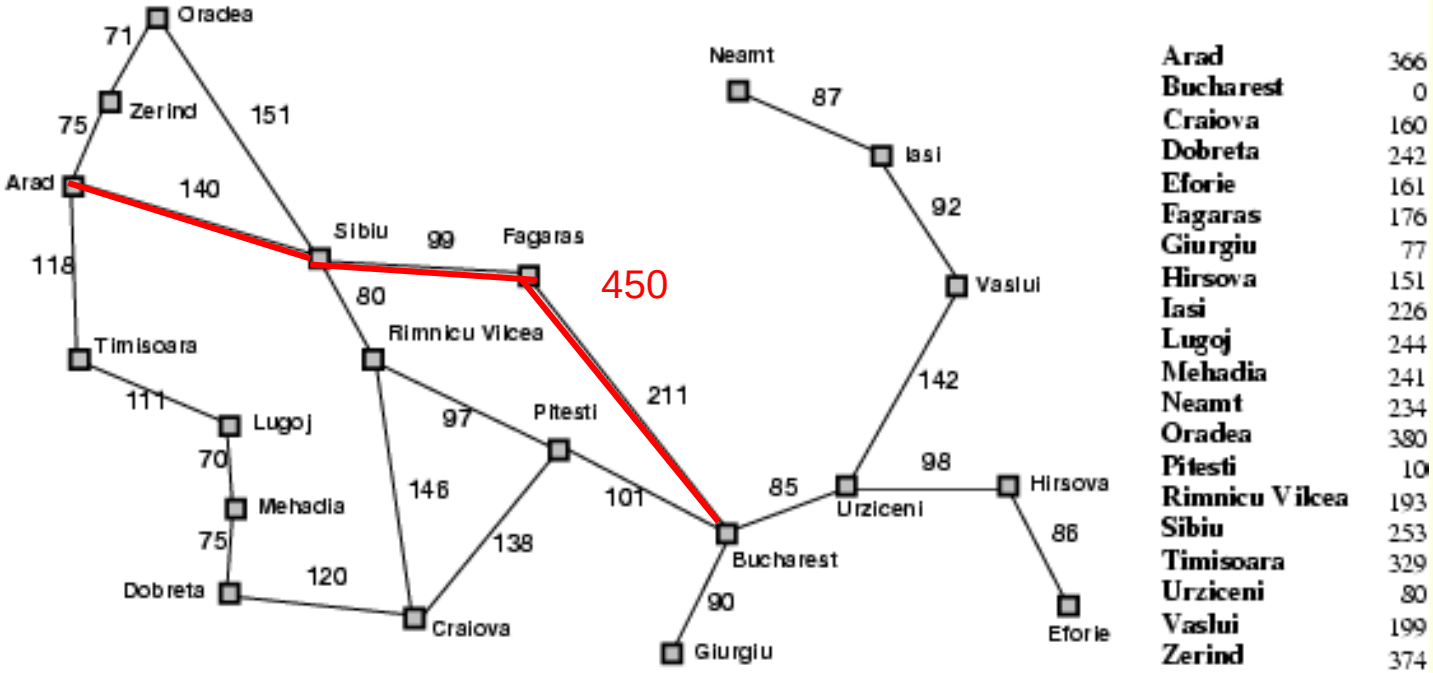



| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

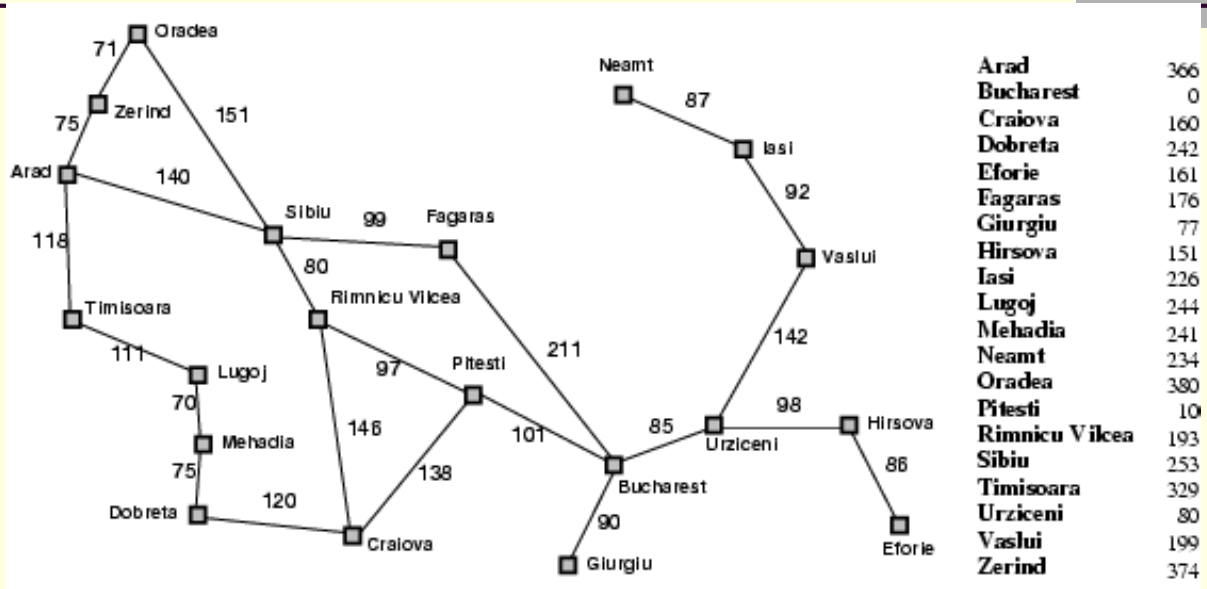
# Cautarea Greedy




| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

# Greedy



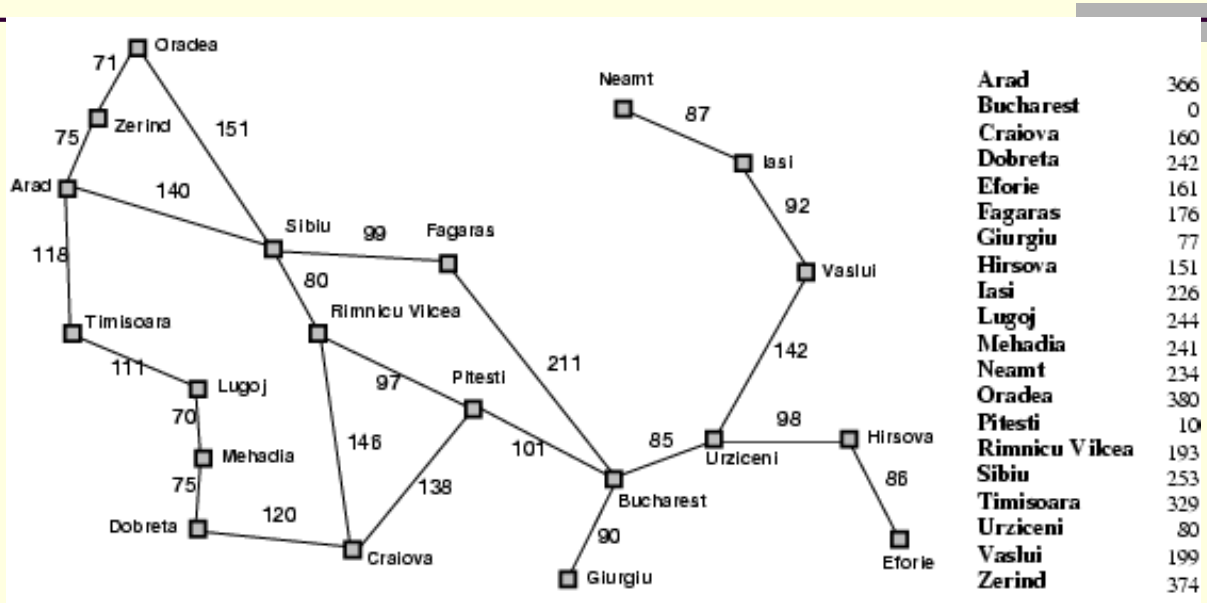

366 Arad

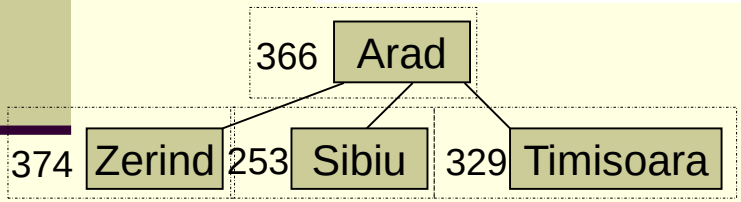
Faţa **noduri** Sfarsit

| | |
|---|---|
| Arad | |
| 366 | |

Parcurgerea: Arad,

# Greedy



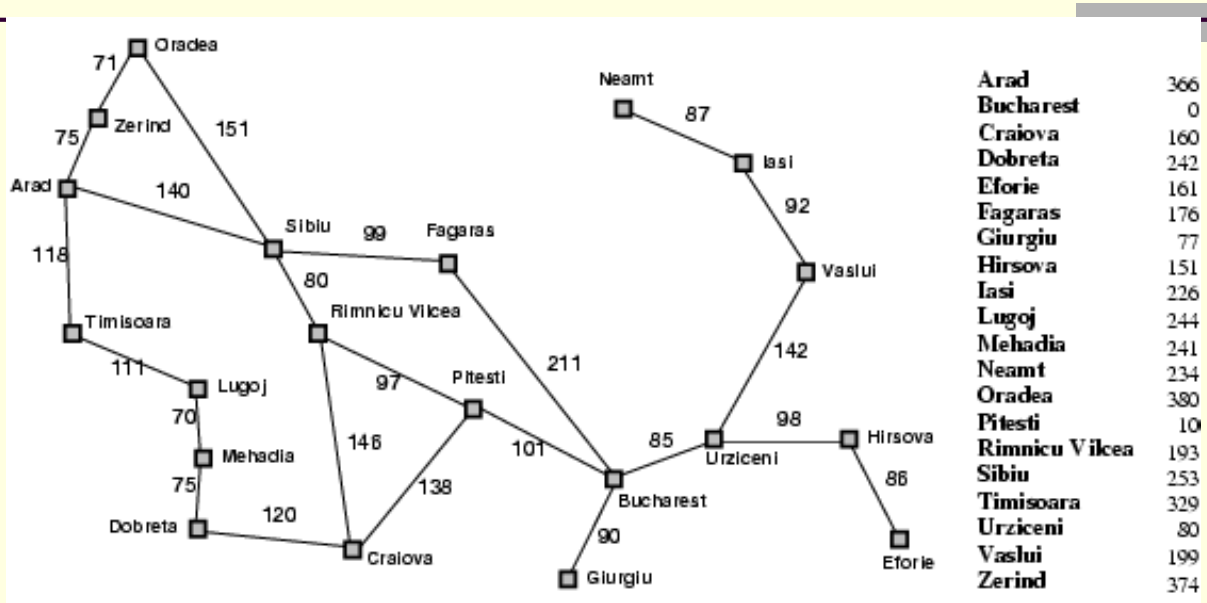




Se adauga in noduri, din faţă, incepand cu nodul cu evaluarea cea mai buna (orasul aflat la cei mai putini km, in acest caz).

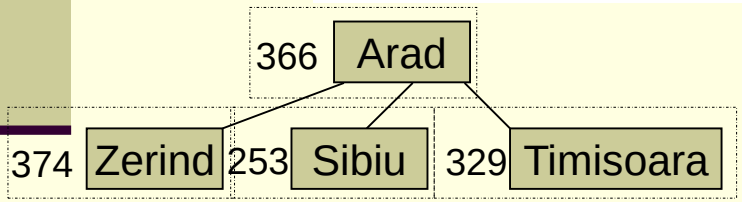
| Faţa | **noduri** | Sfarsit |
|---|---|---|
| Arad | | |
| 366 | | |

Parcurgerea: Arad,

# Greedy




| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

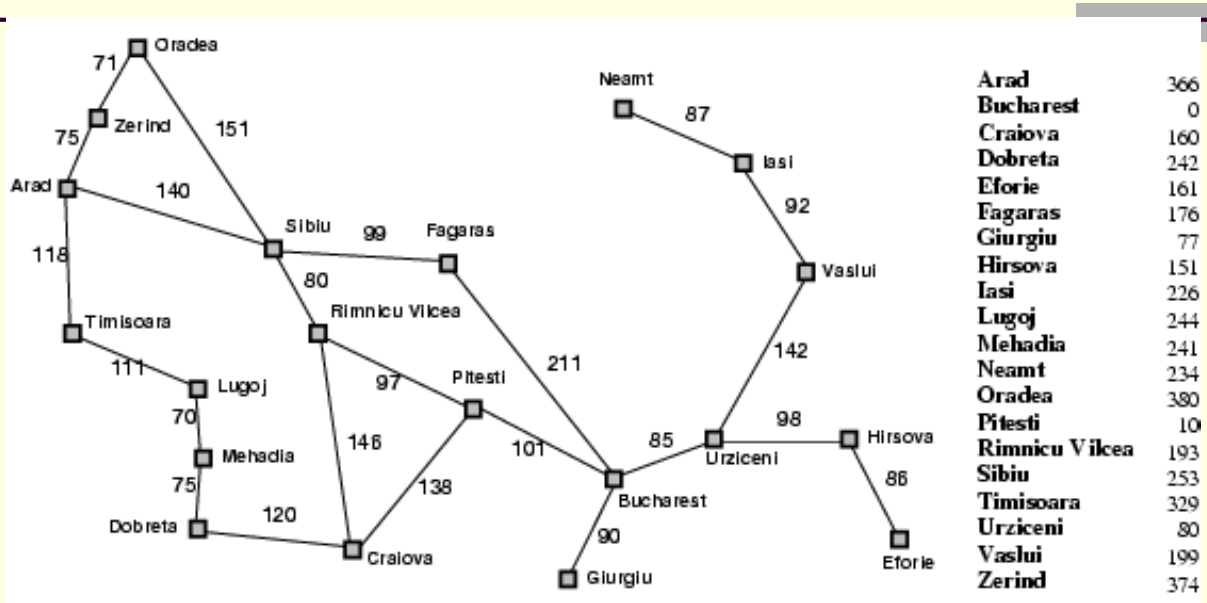


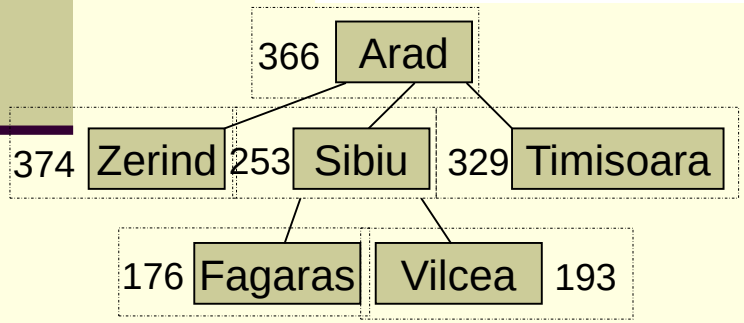

Faţa **noduri** Sfarsit

| | | | |
|---|---|---|---|
| Sibiu | Timisoara | Zerind | |
| 253 | 329 | 374 | |

Parcurgerea: Arad,

# Greedy



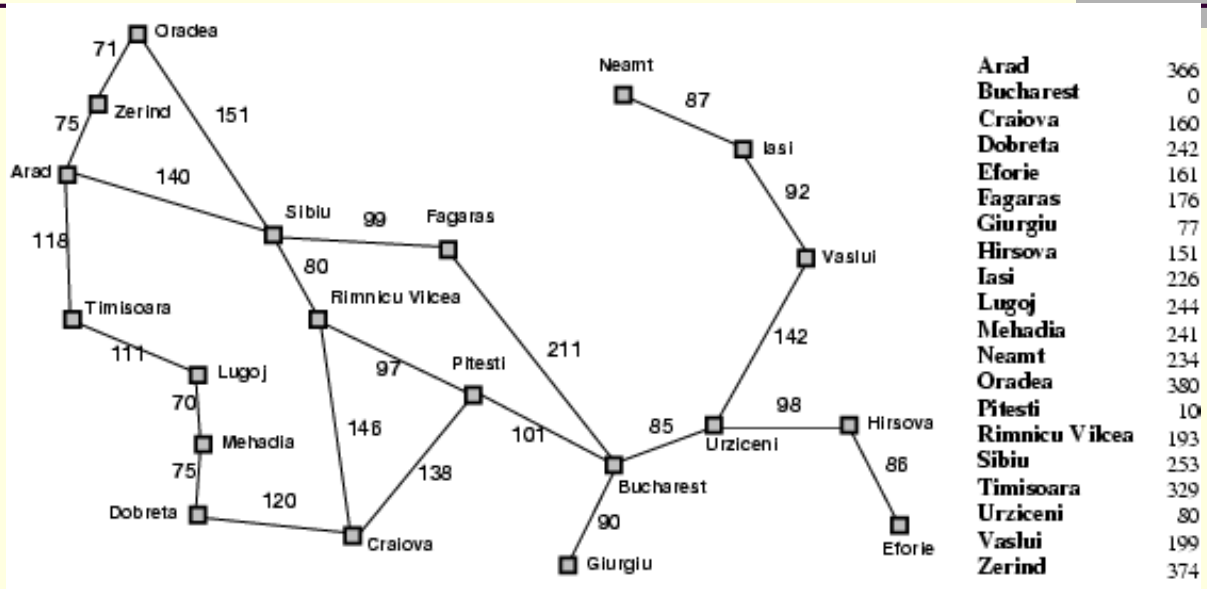



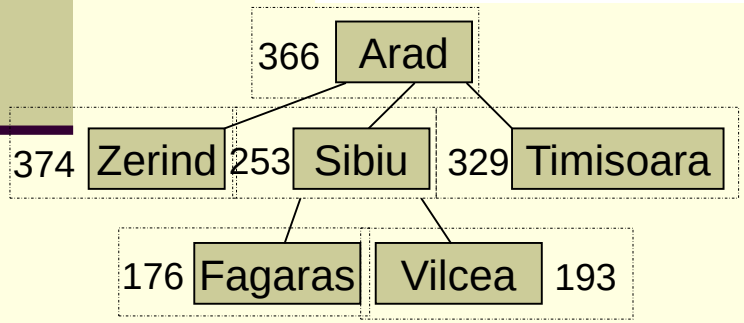

Faţa **noduri** Sfarsit

| | | | |
|---|---|---|---|
| Sibiu | Timisoara | Zerind | |
| 253 | 329 | 374 | |

Parcurgerea: Arad, Sibiu

# Greedy



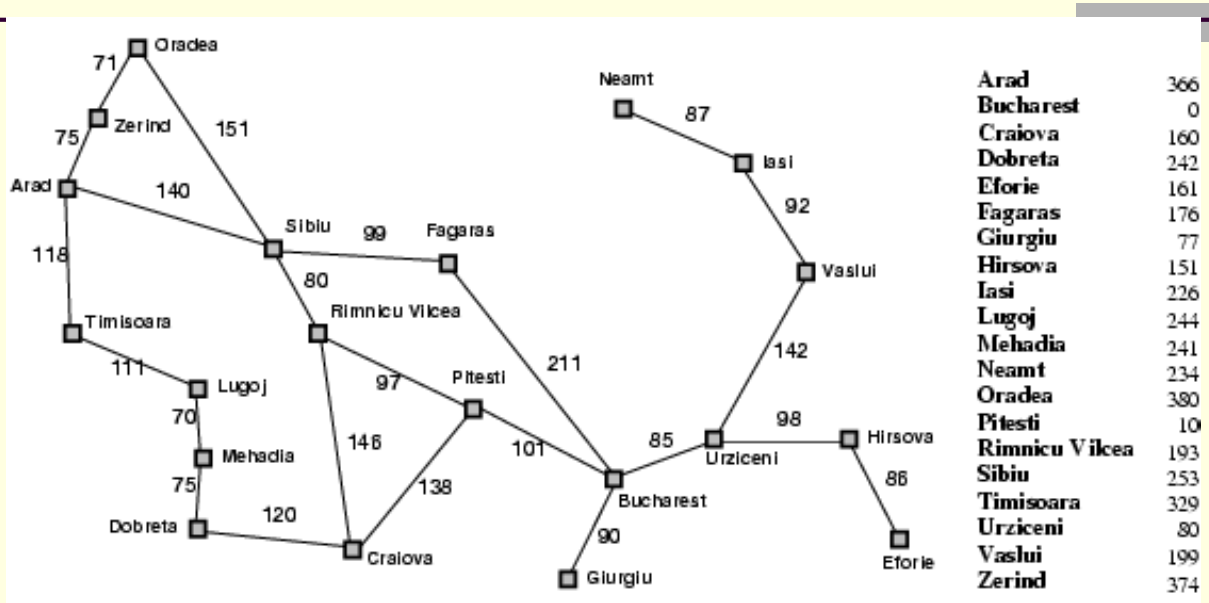



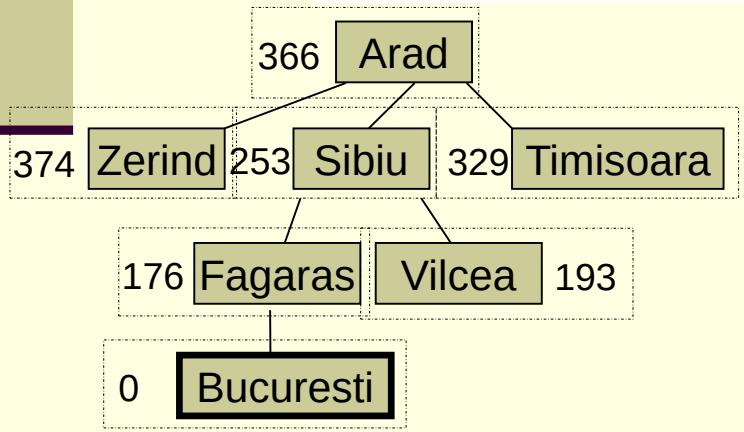

Faţa **noduri** Sfarsit

| Fagaras | Vilcea | Timisoara | Zerind | |
|---|---|---|---|---|
| 176 | 193 | 329 | 374 | |

Parcurgerea: Arad, Sibiu

# Greedy



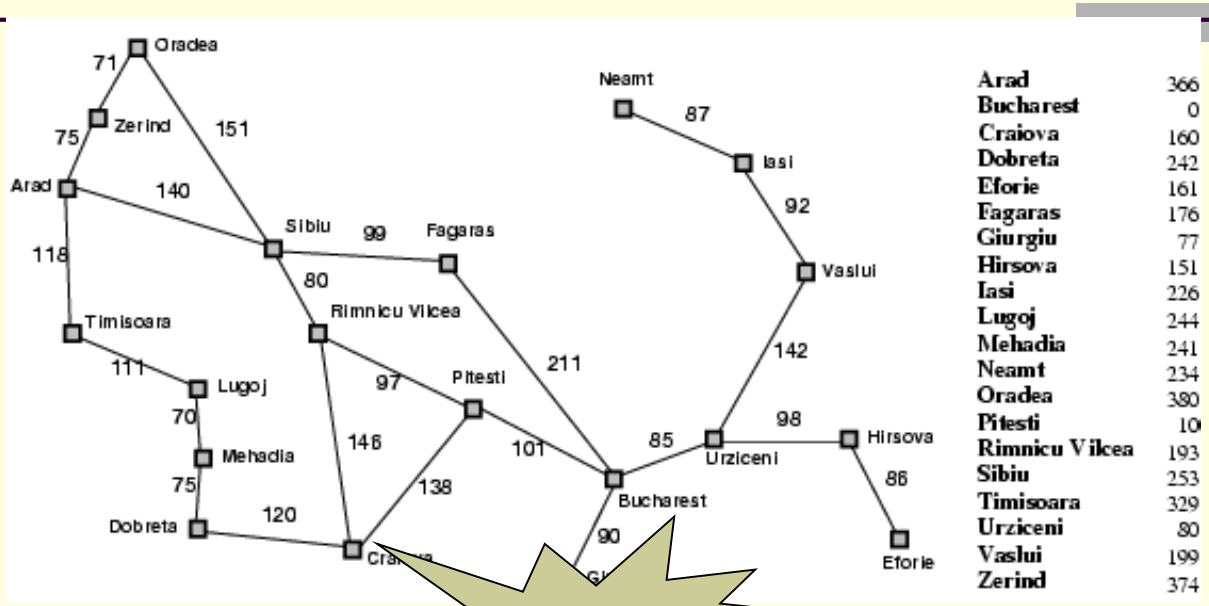




Faţa **noduri** Sfarsit

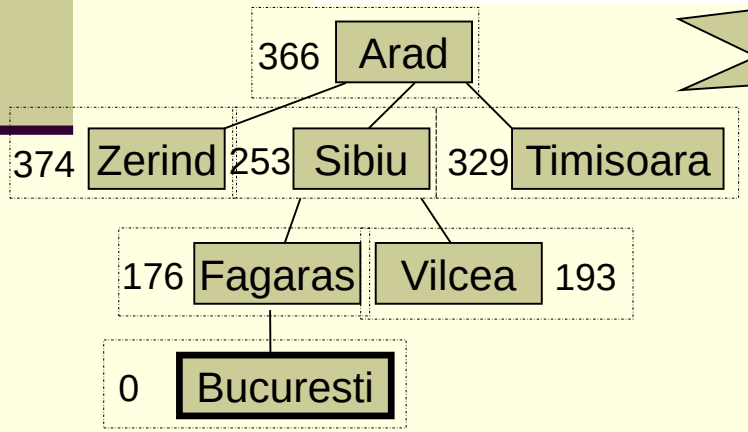
| Fagaras | Vilcea | Timisoara | Zerind | |
|---|---|---|---|---|
| 176 | 193 | 329 | 374 | |

Parcurgerea: Arad, Sibiu, Fagaras

# Greedy




**Nod tinta!**



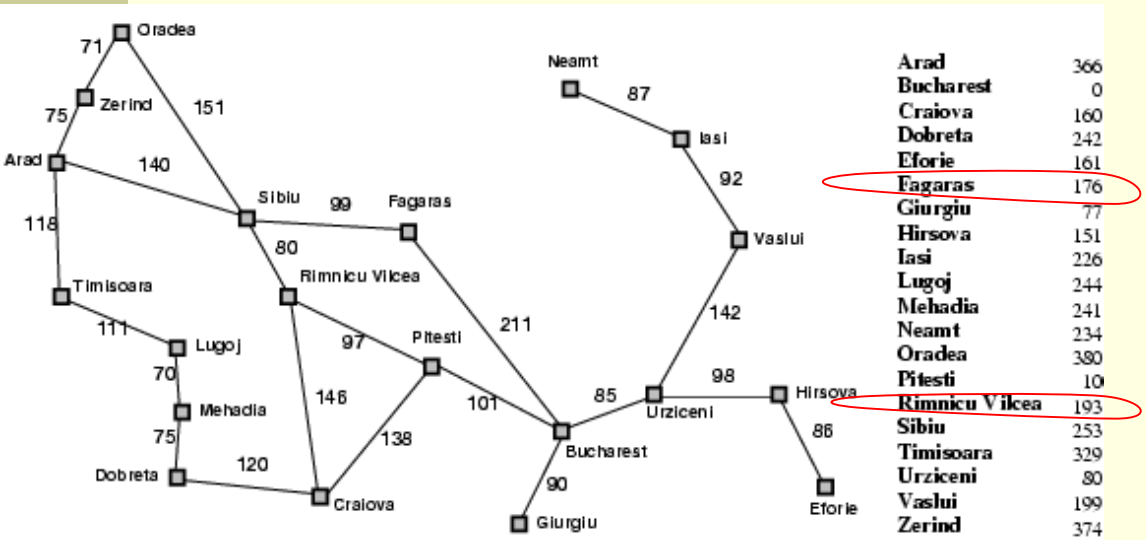

Faţă **noduri** Sfarsit

| Bucuresti | Vilcea | Timisoara | Zerind | |
|---|---|---|---|---|
| 0 | 193 | 329 | 374 | |

Parcurgerea: Arad, Sibiu, Fagaras, Bucuresti.

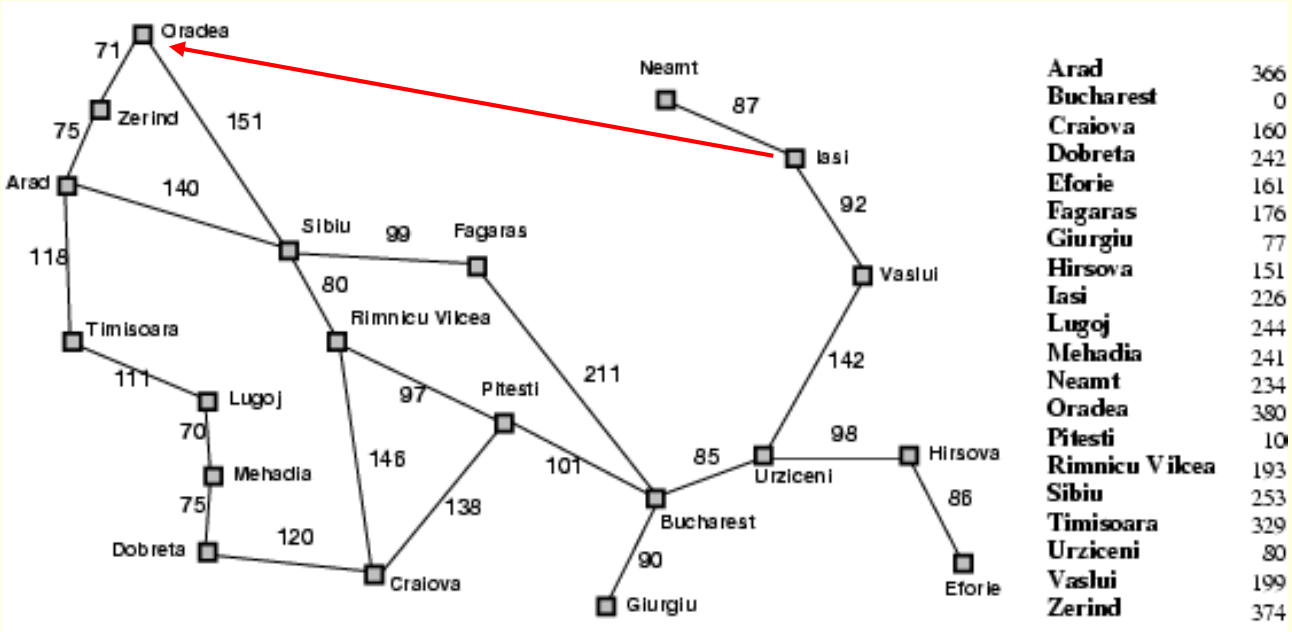
# Cautarea Greedy

- Totusi, solutia nu este optimala.
  - Via Sibiu -> Fagaras -> Bucuresti este cu 32 km mai mult decat Sibiu -> Rimnicu Vilcea -> Pitesti -> Bucuresti.
  - De ce nu a fost aleasa calea mai convenabila?...
    - Pentru ca Fagaras este mai aproape de Bucuresti in linie dreapta decat Rimnicu Vilcea!



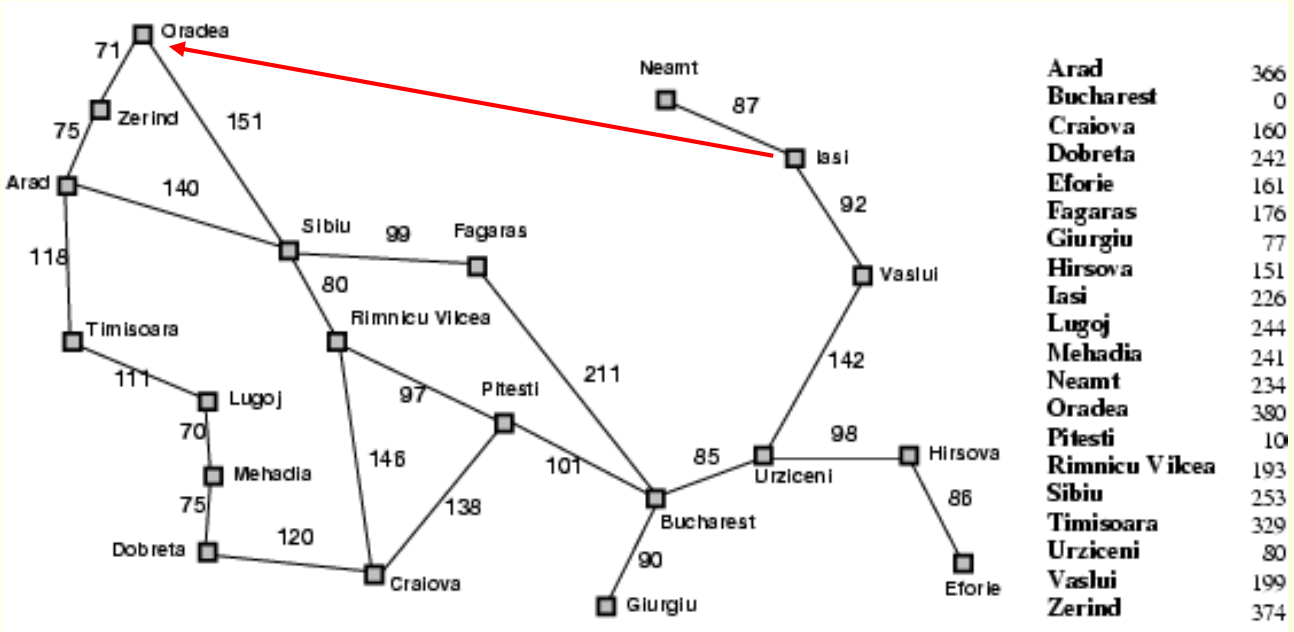

Cautarea Greedy in general gaseste solutia rapid insa nu gaseste intotdeauna solutia optimala.

# Cautarea Greedy – dezavantaj…




- Cautarea Greedy poate avea starturi gresite.
- De exemplu, daca vrem sa ajungem din Iasi in Oradea, functia euristica va expanda intai nodul Neamt, insa acesta este un nod care se inchide => Iasi ->Neamt -> Iasi-> Neamt…

# Cautarea Greedy – dezavantaj…




- Solutia ar fi sa se deplaseze in Vaslui, un nod care este mai departat de nodul tinta - conform cu euristica – si sa continue apoi cu Urziceni, Bucuresti etc.
- Daca nu formulam atent problema pentru a nu face pasi repetitivi, cautarea poate oscila la infinit intre Neamt si Iasi.

# Cautarea Greedy

- Este similara cu cautarea in adancime
  - Ca si in acel caz, merge pe un singur drum pana la capat si, daca nu gaseste nodul tinta, se intoarce pentru a alege un alt drum.
- Ca si in cazul cautarii in adancime, algoritmul de cautare Greedy nu este **optimal** si este **incomplet** pentru ca o poate apuca pe un drum infinit si astfel sa nu se mai intoarca pentru a alege alte posibilitati.
- Complexitatea temporala si spatiala: $O(b^m)$, unde
  - $b$ este numarul de noduri in care se expandeaza fiecare nod
  - $m$ este adancimea maxima a spatiului de cautare.

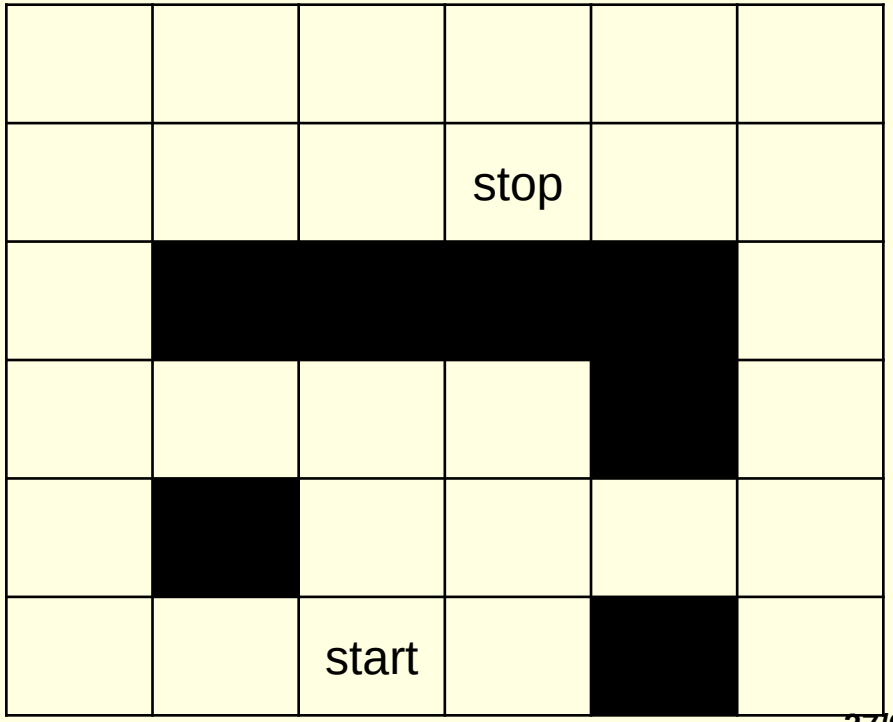
# Exercitiu

- Consideram distantele directe dintre 9 puncte date ca in tabelul de mai jos. Cazul in care intr-o casuta din tabel nu este trecuta nicio valoare semnifica faptul ca nu exista un drum direct intre cele doua puncte (de exemplu, intre A si C). Sa se aplice cautarea bidirectionala pentru gasirea drumului de la J la A, astfel incat din J sa se porneasca cu cautarea cu cost uniform, iar din A cu cautarea in latime: reprezentati arborii si scrieti ruta obtinuta in final.

| | A | B | C | D | E | F | G | H | J |
|---|---|---|---|---|---|---|---|---|---|
| A | 0 | 10 | - | - | 7 | - | - | - | - |
| B | 10 | 0 | 2 | - | - | 5 | - | - | - |
| C | - | 2 | 0 | 10 | - | 4 | 11 | - | - |
| D | - | - | 10 | 0 | 12 | 7 | - | - | - |
| E | 7 | - | - | 12 | 0 | 4 | - | - | - |
| F | - | 5 | 4 | 7 | 4 | 0 | - | 8 | - |
| G | - | - | 11 | - | - | - | 0 | - | 10 |
| H | - | - | - | - | - | 8 | - | 0 | 15 |
| J | - | - | - | - | - | - | 10 | 15 | 0 |

# Exercitiu

- Consideram problema gasirii unui rute in figura de mai jos de la start la stop.
- Agentul se muta un patrat la fiecare pas vertical sau orizontal.
- Nu se poate deplasa in patratele hasurate.
- 1. Etichetati cu litere in ordine alfabetica patratele daca se utilizeaza o cautare Greedy.
- Pentru aproximare, folositi distanta Manhattan – numarul de patrate vizitate in directia x + patratele in directia y pana la tinta.




Nu se tine cont de patratele hasurate pentru distanta Manhattan

.

# Cautarea A*

- **Cautarea Greedy** minimizeaza costul estimat catre tinta ***h*(*n*)**, ceea ce face sa scada considerabil costul cautarii.
  - Nu este insa nici optimal, nici complet!
- **Cautarea cu cost uniform** minimizeaza costul drumului pana la momentul curent folosind ***g*(*n*)**.
  - Este optimal si complet insa poate fi ineficient!

- Prin combinarea celor doua strategii, vom obtine o functie care tine cont si de costul de pana acum in drumul considerat, si de costul estimat pana la tinta:

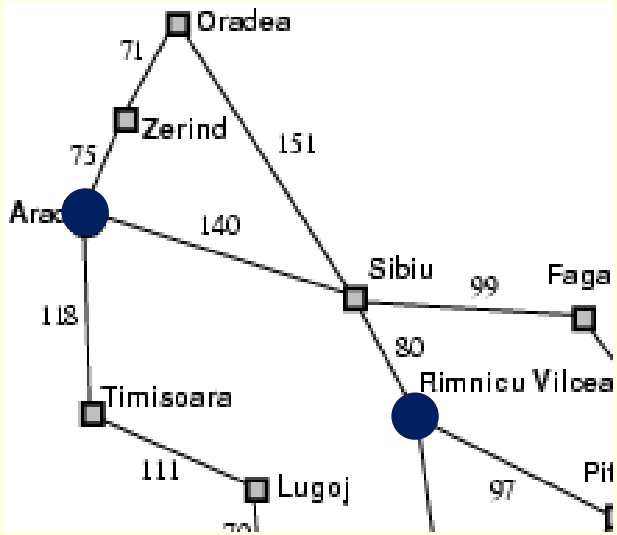
$$f(n) = g(n) + h(n)$$

Cum era…

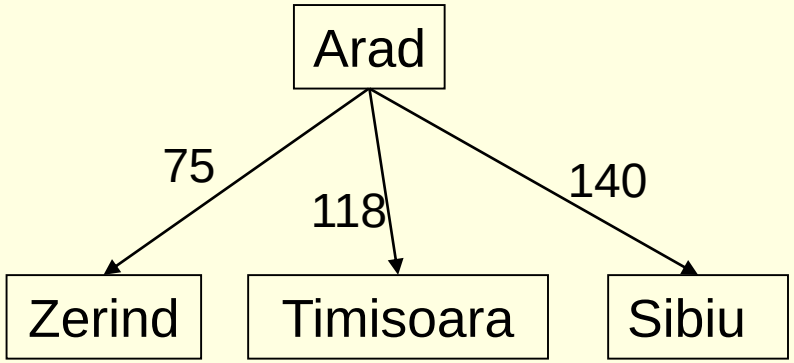
# Cautarea cu cost uniform

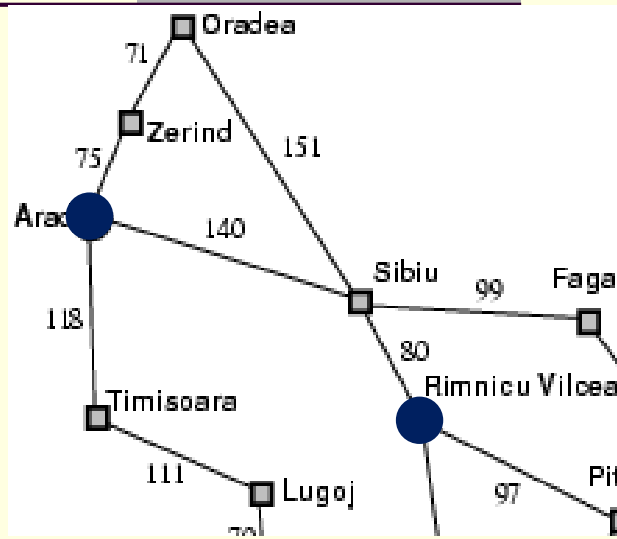
- Este echivalenta cu cautarea in latime daca toate costurile sunt egale.
- Extinde mereu nodul cu costul minim.
- Solutia cu cost minim va fi garantat gasita pentru ca daca exista o cale cu un cost mai mic aceasta este aleasa.

Vrem sa ajungem de la Arad la Rimnicu Vilcea.



# Cautarea cu cost uniform

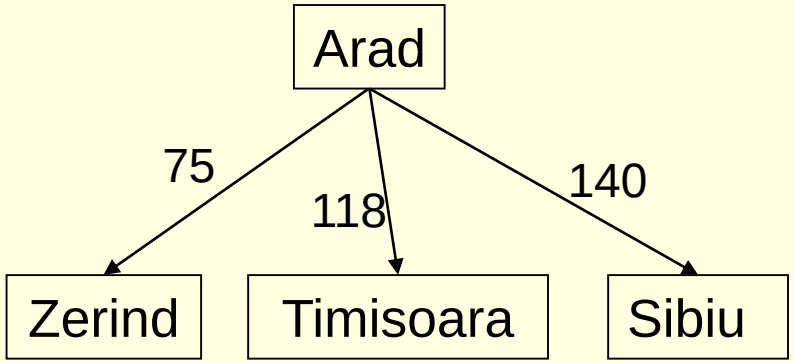


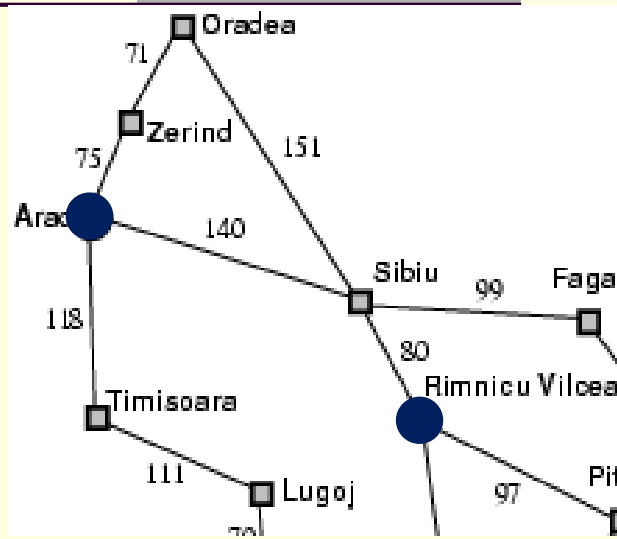




| Faţa | **noduri** | Sfarsit |
|---|---|---|
| Arad | | |
| 0 | | |

Parcurgerea: Arad,

# Cautarea cu cost uniform



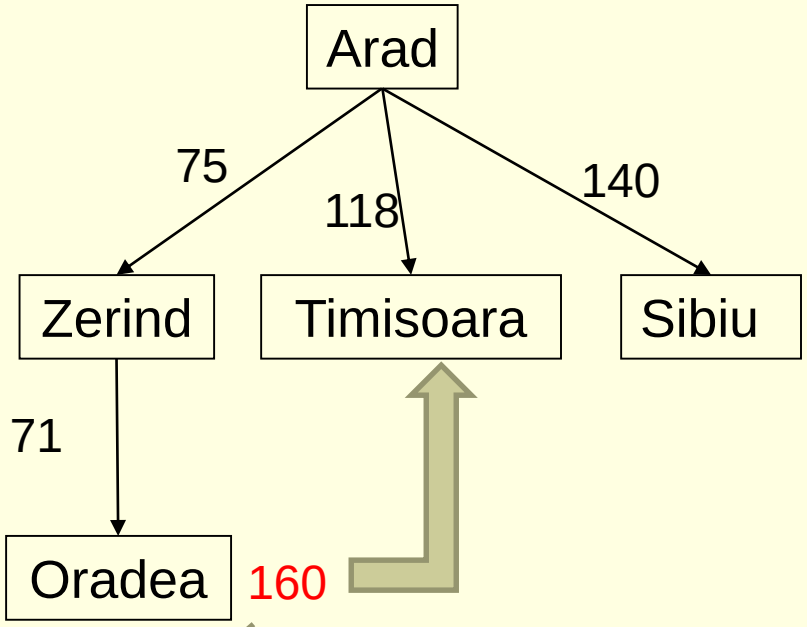



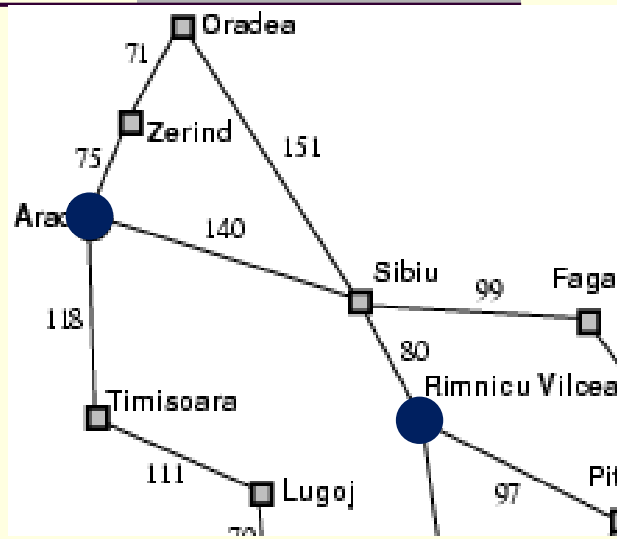

Faţa **noduri** Sfarsit

| | | | |
|---|---|---|---|
| Zerind | TM | SB | |
| 75 | 118 | 140 | |

Parcurgerea: Arad,

# Cautarea cu cost uniform



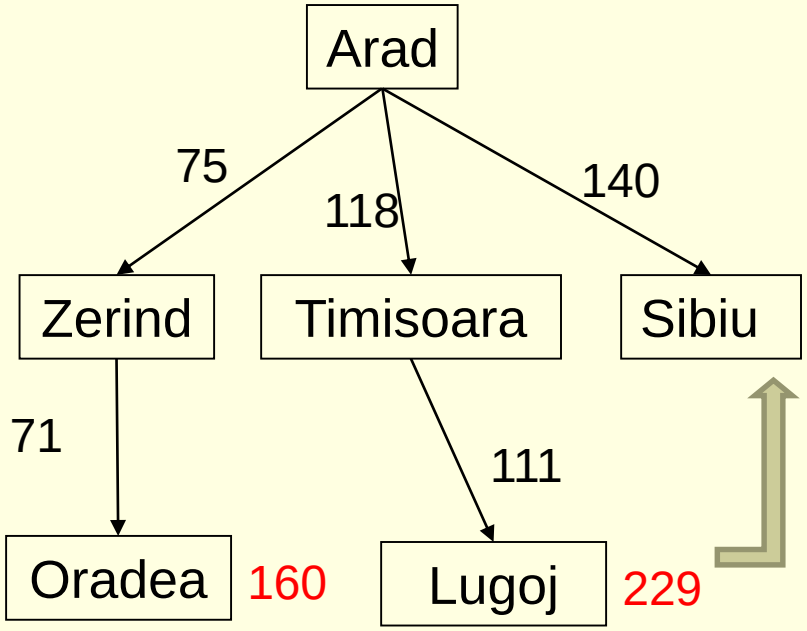



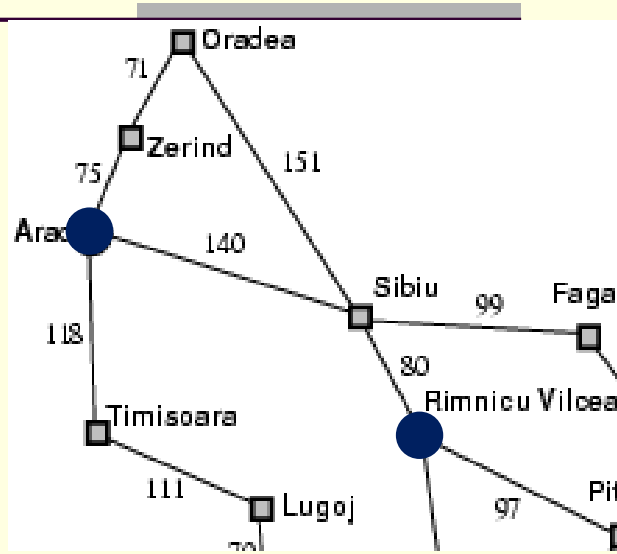

In total, de la Arad la Oradea.

| Faţa | **noduri** | | Sfarsit |
|---|---|---|---|
| TM | SB | Oradea | |
| 118 | 140 | 160 | |

Parcurgerea: Arad, Zerind

# Cautarea cu cost uniform



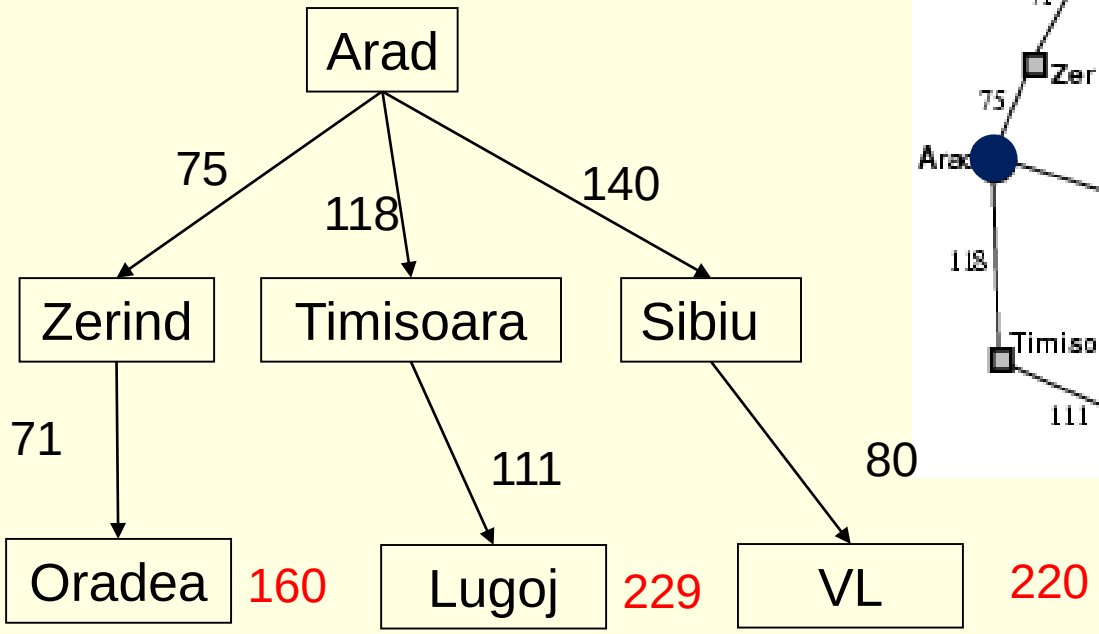



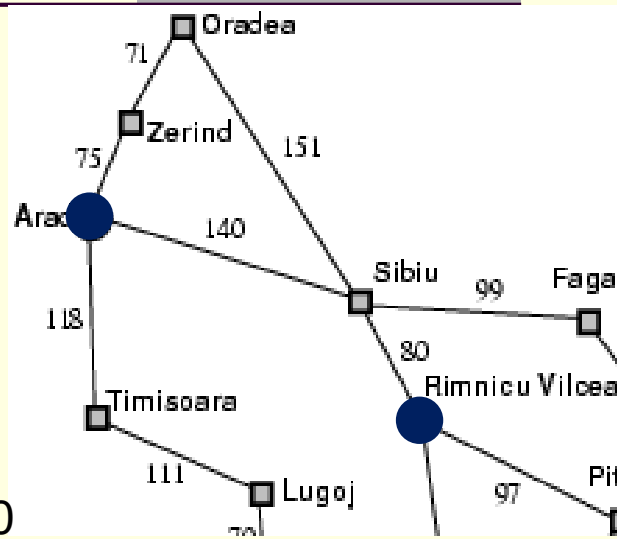

Faţa **noduri** Sfarsit

| SB | Oradea | Lugoj | |
|---|---|---|---|
| 140 | 160 | 229 | |

Parcurgerea: Arad, Zerind, TM

# Cautarea cu cost uniform



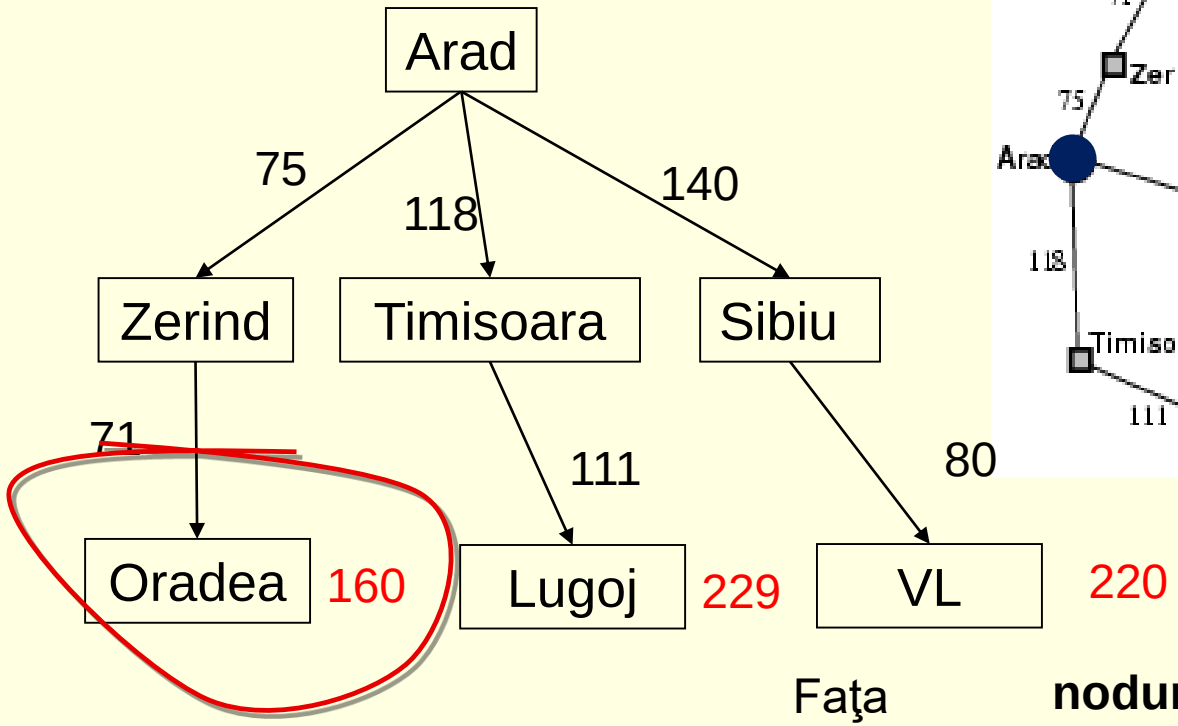



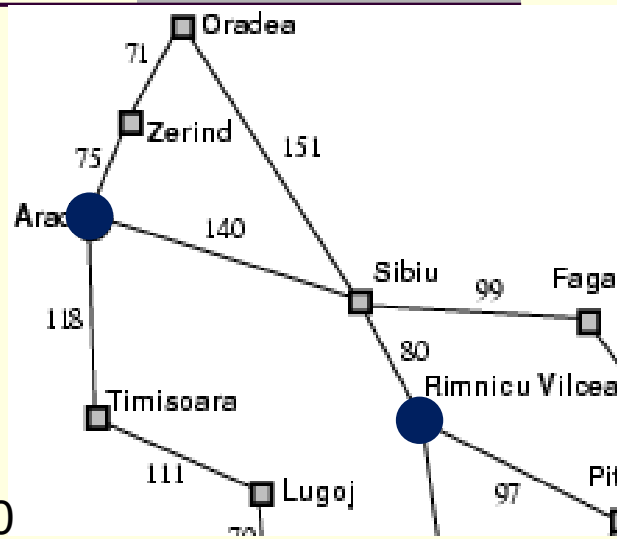

Faţa **noduri** Sfarsit

| Oradea | VL | Lugoj | |
|---|---|---|---|
| 160 | 220 | 229 | |

Parcurgerea: Arad, Zerind, TM, SB

# Cautarea cu cost uniform



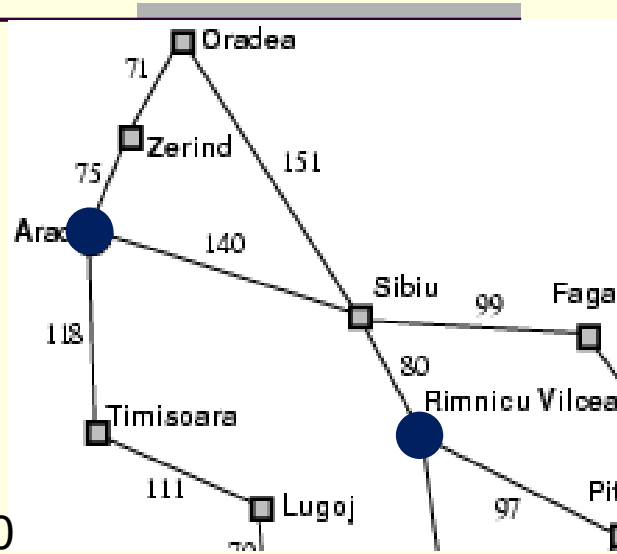

Faţa **noduri** Sfarsit

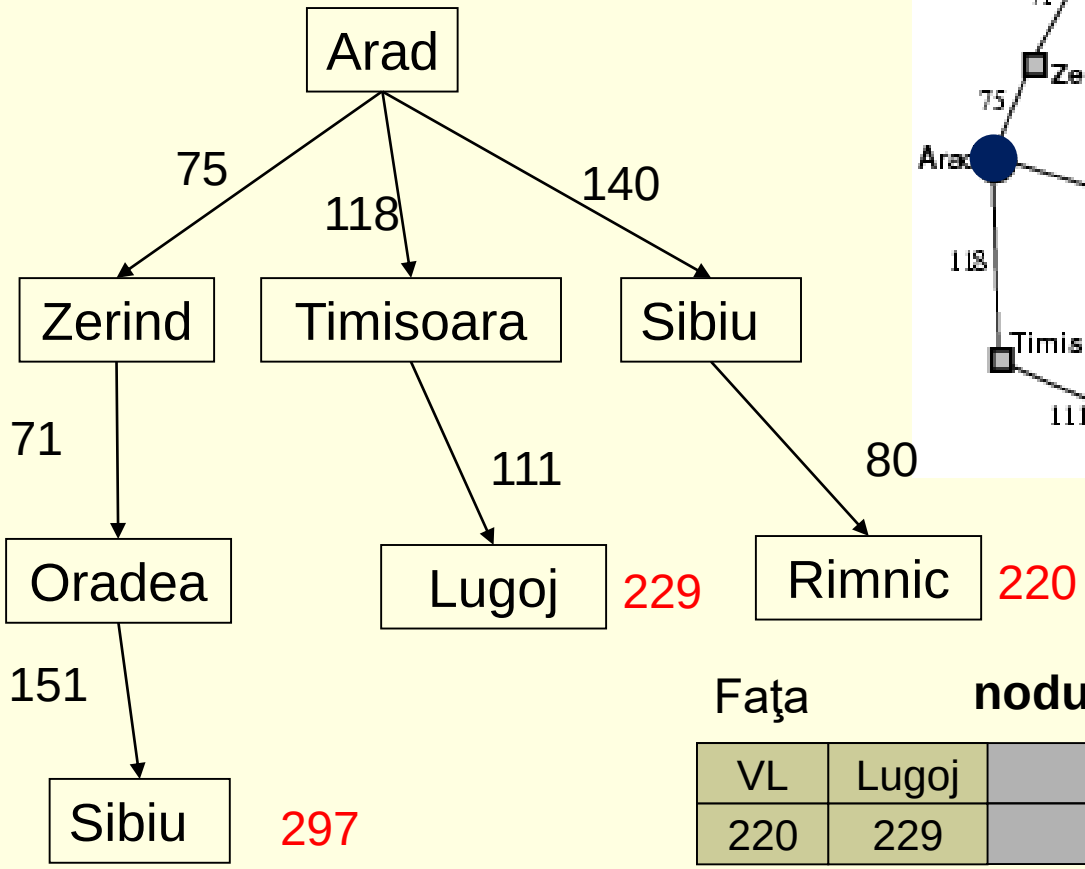
| Oradea | VL | Lugoj | |
|---|---|---|---|
| 160 | 220 | 229 | |

Parcurgerea: Arad, Zerind, TM, SB

Nu ne oprim pana cand nu este depasita valoarea 220 pe toate rutele posibile.

# Cautarea cu cost uniform




Nu adaugam SB pentru ca are o evaluare mai mare decat vechea valoare a orasului.

Faţa **noduri** Sfarsit

| | | |
|---|---|---|
| VL | Lugoj | |
| 220 | 229 | |

Parcurgerea: Arad, Zerind, TM, SB, Oradea, VL

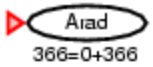
# Cautarea A*

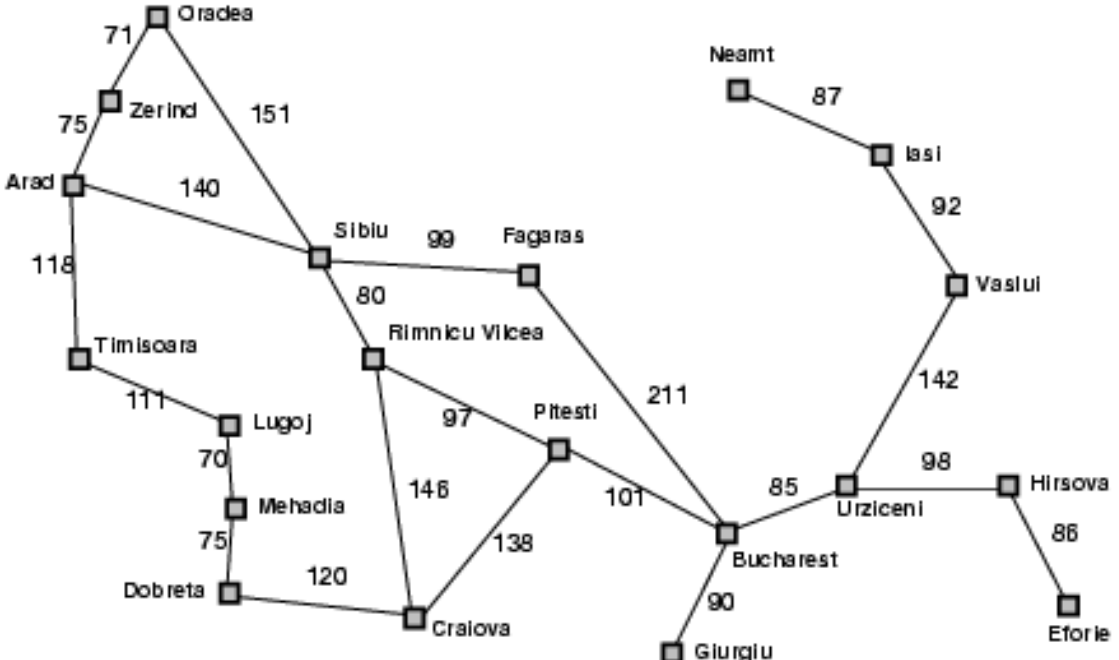
- Cum *g*(*n*) da costul drumului de la starea initiala pana la nodul *n*, iar *h*(*n*) estimeaza costul celui mai convenabil drum de la *n* pana la tinta…

*f*(*n*) = costul estimat al celei mai convenabile solutii prin *n*.

- Este demonstrabil faptul ca algoritmul care foloseste cautarea A* este **complet** si **optimal** cu conditia ca functia *h* **nu supraestimeaza** costul de ajungere la solutie.

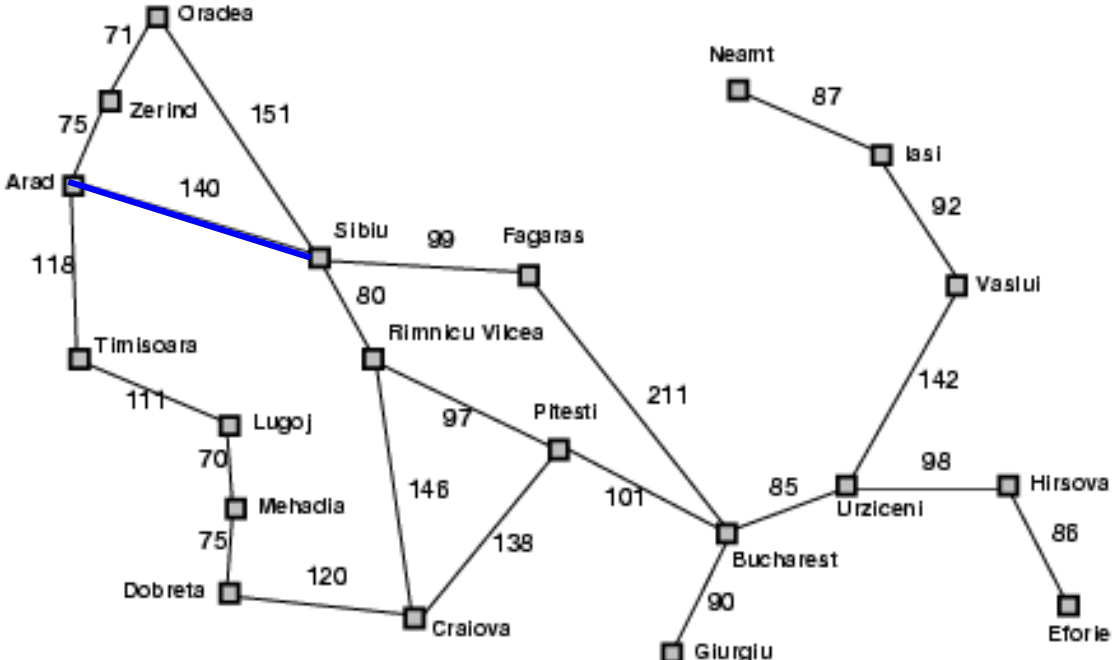
# Cautarea A*



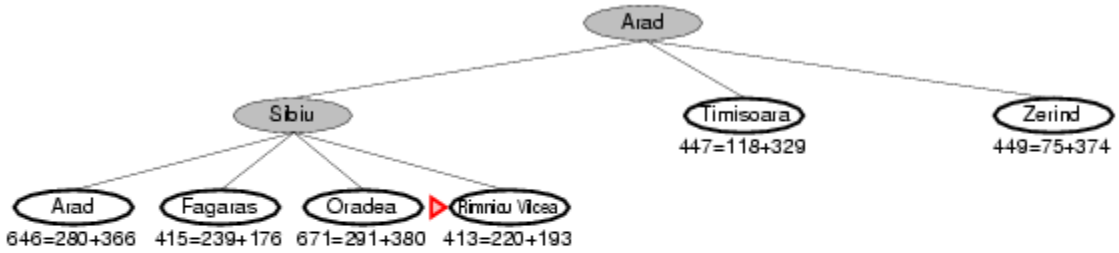

| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

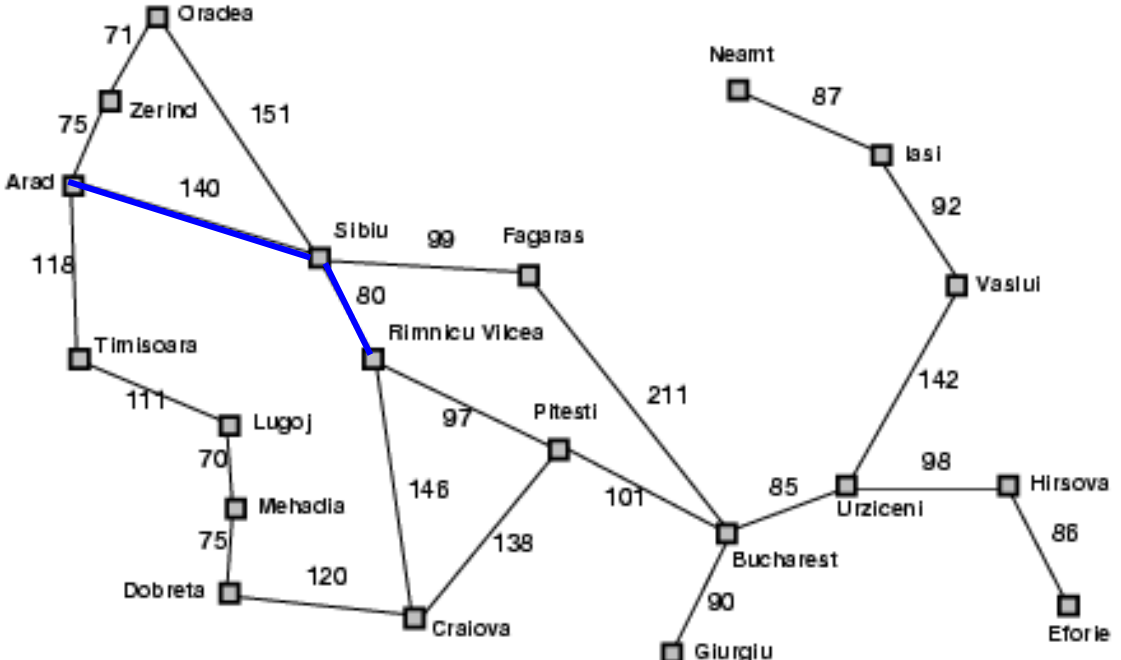
# Cautarea A*


| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | |

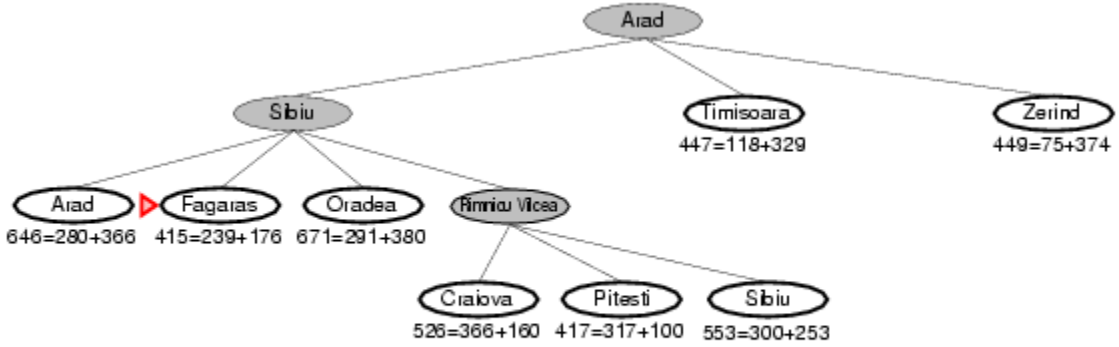
# Cautarea A*

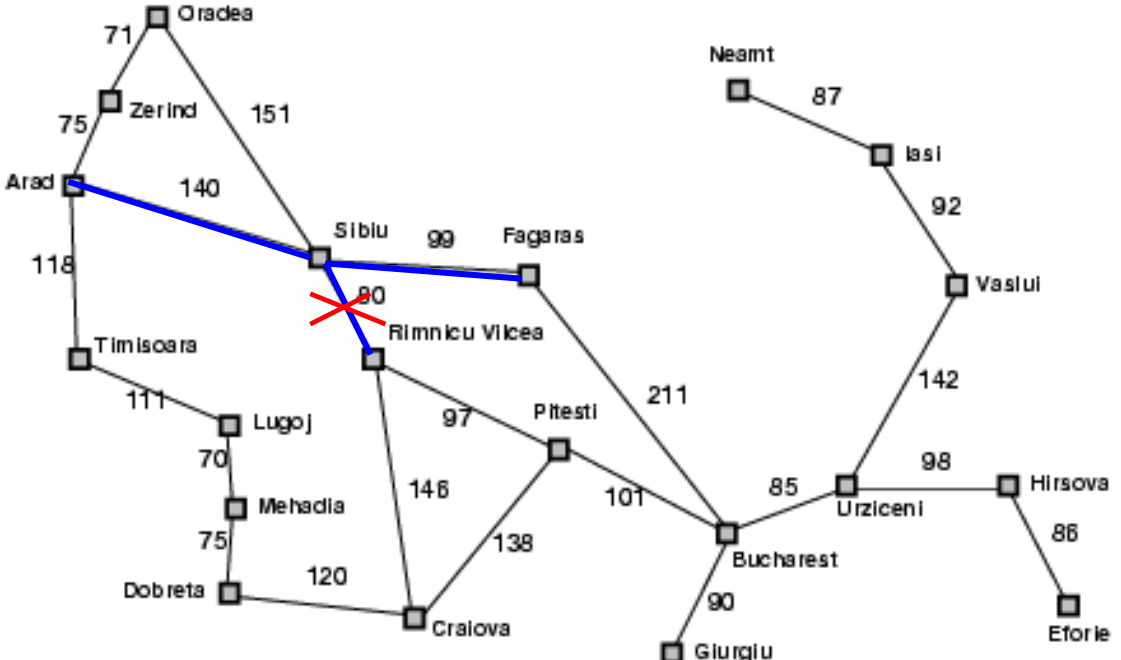


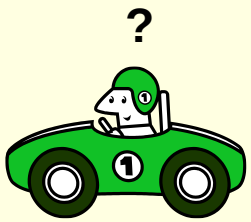

| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | |

# Cautarea A*

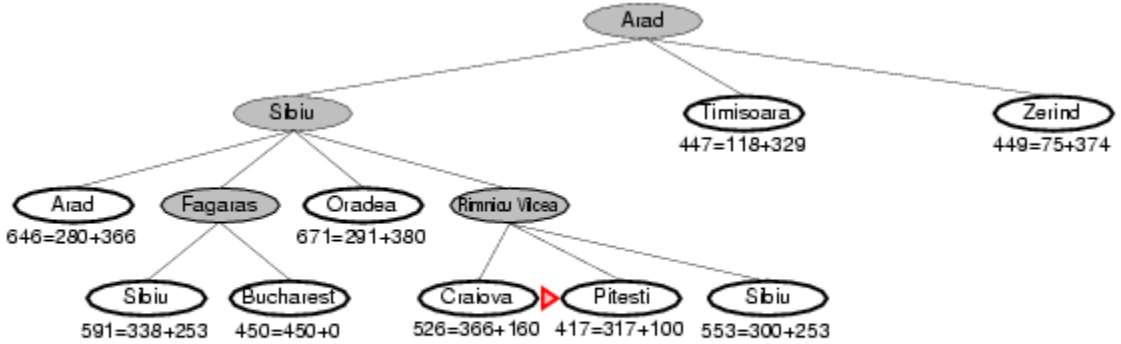


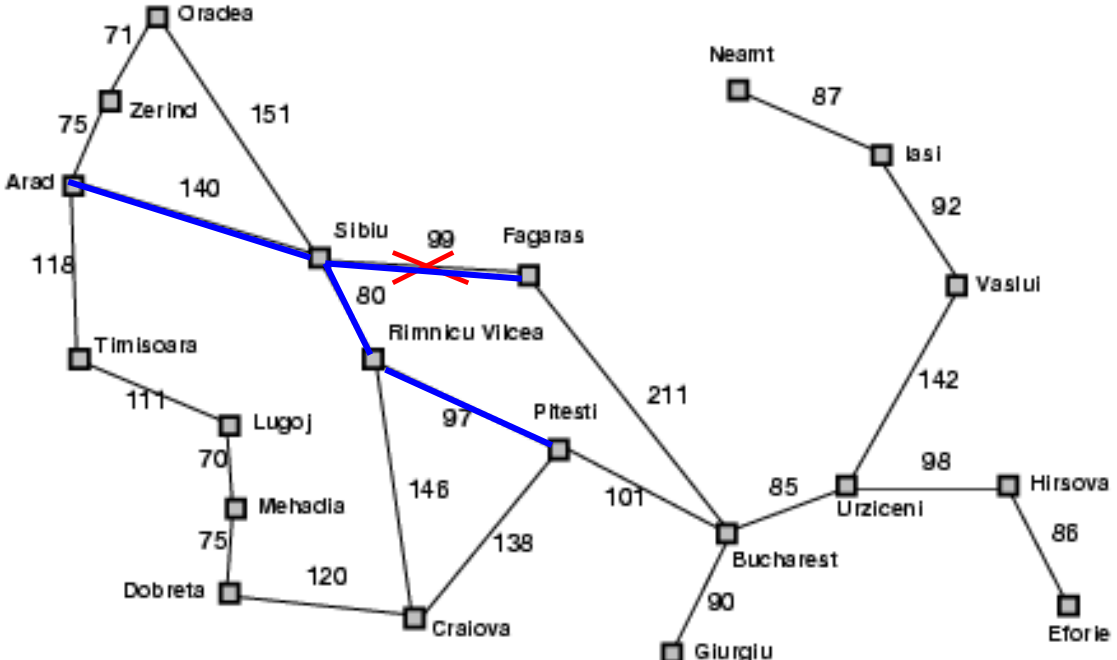




| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

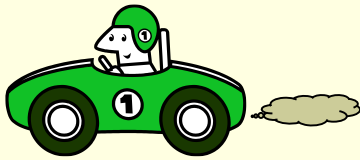


# Cautarea A*

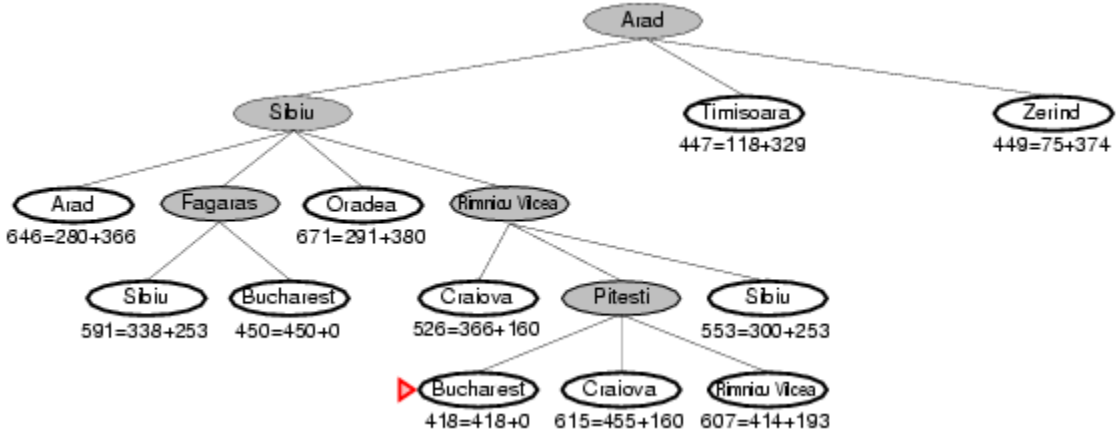


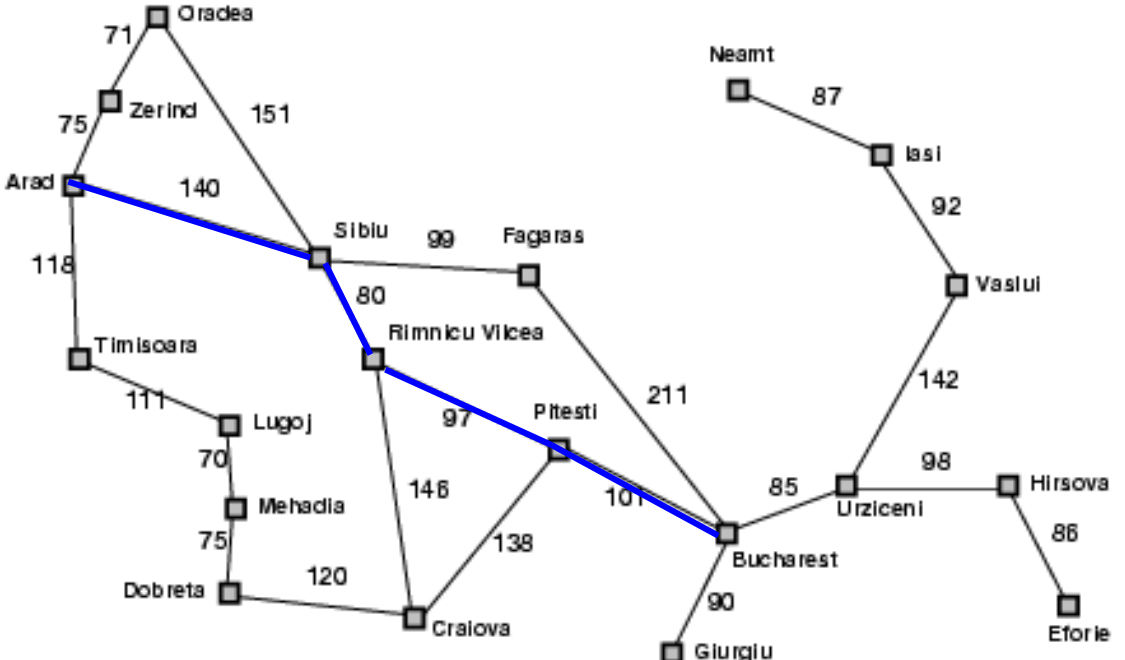




| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | |



# Cautarea A*



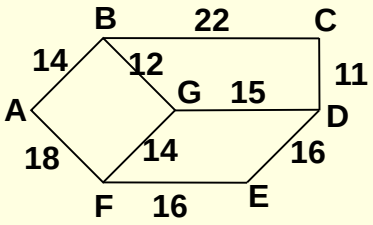

| | |
|---|---|
| Arad | 366 |
| Bucharest | 0 |
| Craiova | 160 |
| Dobreta | 242 |
| Eforie | 161 |
| Fagaras | 176 |
| Giurgiu | 77 |
| Hirsova | 151 |
| Iasi | 226 |
| Lugoj | 244 |
| Mehadia | 241 |
| Neamt | 234 |
| Oradea | 380 |
| Pitesti | 10 |
| Rimnicu Vilcea | 193 |
| Sibiu | 253 |
| Timisoara | 329 |
| Urziceni | 80 |
| Vaslui | 199 |
| Zerind | 374 |

# Cautarea A*

- Complexitatea spatiala: toate nodurile sunt retinute in memorie.
- Complexitatea temporala: $O(b^m)$.
- Este complet si optimal daca $h$ nu supraestimeaza costul pana la starea tinta.

# Exercitiu - Cautarea informata

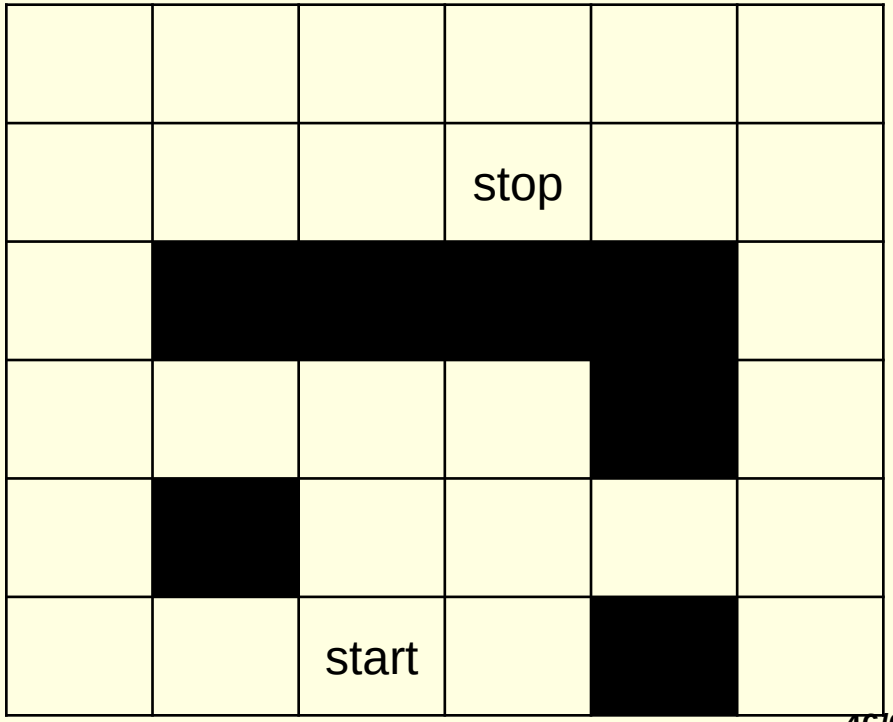
- Consideram asezarea a sapte orase ca in figura de mai jos. Scopul este ca, pornind din starea initiala D, sa ajungem in starea finala A. Distantele din figura sunt cele rutiere care se gasesc intre orase (deci nu in linie dreapta). Distantele in linie dreapta de la fiecare oras catre orasul A sunt urmatoarele:
  - D: 30
  - G: 14
  - C: 24
  - B: 11
  - E: 21
  - F: 9




- Se cere
- Sa se calculeze distanta de la D la A folosind cautarea Greedy. Justificati.
- Acelasi lucru folosind cautarea A*.

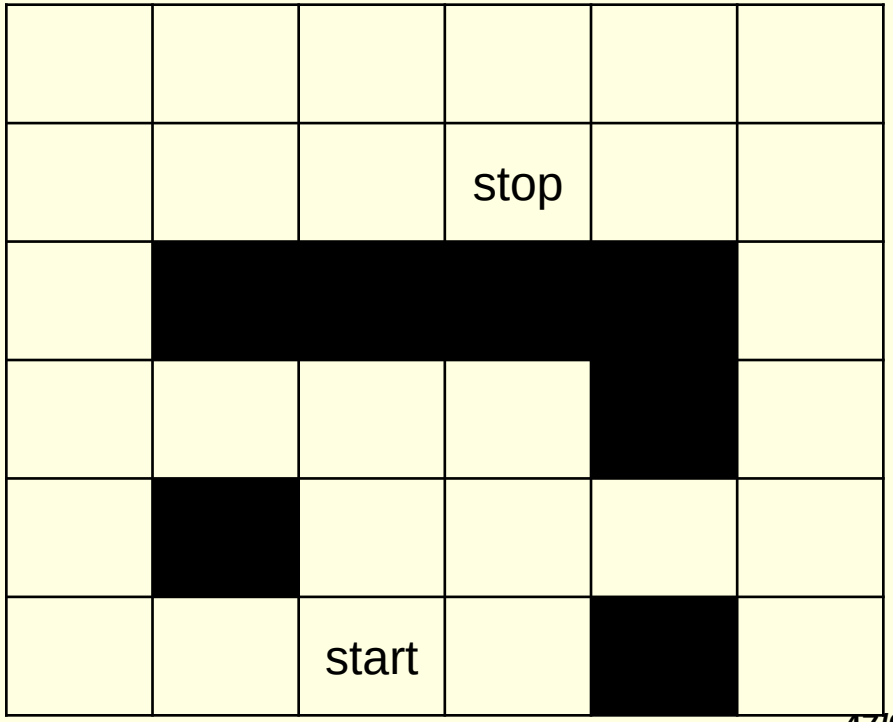
# Exercitiu

- Consideram problema gasirii unui rute in figura de mai jos de la start la stop.
- Agentul se muta un patrat la fiecare pas vertical sau orizontal.
- Nu se poate deplasa in patratele hasurate.
- 2. Etichetati cu litere in ordine alfabetica patratele daca se utilizeaza o cautare A*.
- Pentru aproximare, folositi distanta Manhattan.



# Exercitiu

- Consideram problema gasirii unui rute in figura de mai jos de la start la stop.
- Agentul se muta un patrat la fiecare pas vertical sau orizontal.
- Nu se poate deplasa in patratele hasurate.
- 3. Presupunem ca graful se extinde la infinit in toate directiile. start si stop raman la aceleasi locatii, la fel si patratele negre. Ce metoda nu mai gaseste nicio solutie?



# Functii euristice

- Pentru problema de gasire de rute o functie euristica potrivita este cea aleasa, distanta directa dintre 2 orase.
- Alegerea functiilor euristice depinde de la o problema la alta.
- De alegerea functiei euristice depind complexitatile temporala si spatiala.
- Alegerea potrivita a unei functii euristice se face dupa o buna studiere a spatiului starilor problemei.

# Functii euristice

**Stare curenta**

| | | |
|---|---|---|
| | 2 | 3 |
| **1** | **4** | 6 |
| **8** | **7** | **5** |

**Stare tinta**

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

- Cum factorul de expandare este aproximativ 3, intr-o cautare la o adancime de 20 de nivele, se ajunge sa fie parcurse aproximativ $3^{20}$ stari = $3.5 * 10^9$.
- Avem nevoie de o functie euristica cu proprietatea ca nu supraestimeaza numarul de pasi pana la starea tinta.

# Functii euristice

**Stare curenta**

| | | |
|---|---|---|
| | 2 | 3 |
| **1** | **4** | 6 |
| **8** | **7** | **5** |

**Stare tinta**

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

- $h_1$ = numarul de cifre care sunt gresit pozitionate.

$h_1$

| | | |
|---|---|---|
| | 2 | 3 |
| **1** | **4** | 6 |
| **8** | **7** | **5** |

= 5

# Functii euristice – distanta Manhattan

**Stare curenta**

| | | |
|---|---|---|
| | 2 | 3 |
| 1 | 4 | 6 |
| 8 | 7 | 5 |

**Stare tinta**

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | |

- $h_2$ = suma mutarilor necesare pentru ca toate cifrele sa ajunga la starea tinta ca si cum toate celelalte casute ar fi goale.
- Deplasarile se pot face numai pe orizontala si verticala.

$h_2$

| | | |
|---|---|---|
| | 2 | 3 |
| 1 | 4 | 6 |
| 8 | 7 | 5 |

= 0 + 0 + 1 + 1 + 0 + 1 + 1 + 2 = 6

Inteligenta Artificiala

# Tema

**Doua puncte la examenul final!**

- Folosind cautarea Greedy, rezolvati puzzle-ul cu 8 valori folosind pe rand euristicile $h_1$ si $h_2$ .

$h_1$

| | | |
|---|---|---|
| | 2 | 3 |
| 1 | 4 | 6 |
| 8 | 7 | 5 |

= 5

$h_2$

| | | |
|---|---|---|
| | 2 | 3 |
| 1 | 4 | 6 |
| 8 | 7 | 5 |

= 0 + 0 + 1 + 1 + 0 + 1 + 1 + 2 = 6

- $h_1$ = numarul de cifre care sunt gresit pozitionate.
- $h_2$ = suma mutarilor necesare pentru ca toate cifrele sa ajunga la starea tinta ca si cum toate celelalte casute ar fi goale.
  - Deplasarile se pot face numai pe orizontala si verticala.

# Efectul functiilor euristice asupra performantei algoritmului

- **Factorul efectiv de ramificare** *b** afecteaza in mod direct performanta algoritmului.
- Daca numarul total de noduri expandate de catre A* pentru o anumita problema este *N* si solutia se gaseste la o adancime *d* atunci b* este **factorul de ramificare** (in cate noduri se expandeaza fiecare nod) pe care l-ar avea un arbore uniform de adancime *d* care ar contine *N* noduri:

$$N = 1 + b^* + (b^*)^2 + \ldots + (b^*)^d$$

- O euristica este cu atat mai bine definita cu cat face ca numarul de noduri in care se expandeaza fiecare nod sa fie, in medie, cat mai mic.
- *b** trebuie sa fie cat mai aproape de 1!

# Dominare functii euristice

- Daca avem doua functii euristice adimisibile $h_1$ si $h_2$ cu proprietatea ca $h_2(n) \geq h_1(n)$ pentru orice nod $n$ atunci spunem ca $h_2$ **domina** $h_1$.
  - $h_2$ este functia euristica mai buna decat $h_1$ pentru cautare.
- Au fost generate aleator 100 de probleme, fiecare cu solutii aflate la adancimea 2, 4, …, 24 pentru a fi rezolvate cu A* utilizand pe rand, $h_1$ si $h_2$.
- In toate situatiile, $h_2$ este mai buna decat $h_1$.

# Dominare functii euristice

| | Costul cautarii | | Factorul efectiv de ramificare | |
|---|---|---|---|---|
| ***d*** | **A*($h_1$)** | **A*($h_2$)** | **A*($h_1$)** | **A*($h_2$)** |
| 2 | 6 | 6 | 1.79 | 1.79 |
| 4 | 13 | 12 | 1.48 | 1.45 |
| 8 | 39 | 25 | 1.33 | 1.24 |
| 12 | 227 | 73 | 1.42 | 1.24 |
| 16 | 1301 | 211 | 1.45 | 1.25 |
| 24 | 39 135 | 1 641 | 1.48 | 1.26 |

Folosind cautarea in adancime iterativa, pentru d = 12, costul cautarii este 364 404, iar factorul efectiv de ramificare 2.78.

# Dominare functii euristice

- Asadar, A* folosind $h_2$ expandeaza mai putine noduri in medie decat A* cu $h_1$.
- Concluzia: este intotdeauna mai bine sa se utilizeze functii euristice care intorc valori mai mari, doar sa nu supraestimeze valoarea efectiva.

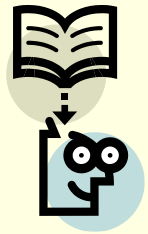
# Relaxarea problemelor

- O problema cu mai putine restrictii asupra actiunilor posibile duce la o **relaxare** a problemei initiale.
- Costul unei solutii optimale obtinut pentru o problema *relaxata* reprezinta o functie euristica admisibila pentru problema originala.
- Daca regulile de la puzzle-ul cu 8 valori sunt relaxate astfel incat o cifra se poate muta oriunde dintr-o singura deplasare, atunci $h_1(n)$ da solutia cea mai scurta.
- Daca regulile sunt relaxate astfel incat o cifra se poate muta la orice casuta **adiacenta** atunci $h_2(n)$ da solutia cea mai scurta.

# Functii euristice

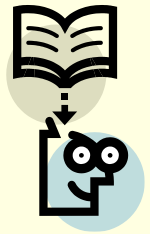
- De multe ori este nevoie sa tinem cont de anumite **caracteristici** ale starii curente care contribuie la evaluarea functiei euristice.
- De exemplu, ce caracteristici ar fi elocvente in cazul functiei euristice pentru jocul de sah? Chiar daca starea tinta este sah mat, se poate tine cont de…
  - Numarul de piese pe care il au fiecare din cei doi jucatori.
  - Numarul de piese atacate de catre piesele adversarului.
    - Tipul pieselor pe care le are fiecare din cei doi jucatori.
    - s.a.m.d.

# Recapitulare 1/2



- **Cautarea *intai cel mai bun*** este doar un arbore general de cautare in care nodurile de cost minim (in functie de o anumita masura) sunt expandate mai intai.
- Daca minimizam costul estimat pentru a ajunge la tinta, $h(n)$, avem **cautare greedy**.
  - Timpul de cautare este redus in comparatie cu algoritmii *neinformati*, insa nu este nici complet, nici optimal.

# Recapitulare 2/2



- Minimizarea lui $f(n) = g(n) + h(n)$ combina avantajele cautarii cu cost uniform cu cele ale cautarii greedy.
- Daca evitam starile repetate si nu supraestimam $h(n)$, avem **cautarea A***.
- A* este complet, optimal, insa complexitatile pentru timp si spatiu sunt inca mari.
- Complexitatea temporala a algoritmilor euristici depinde de calitatea functiei euristice folosite.