Aplicatii la cautarea neinformata si informata

Catalin Stoean

catalin.stoean@inf.ucv.ro

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

Cuprins

Formularea problemelor

- O problema se defineste prin patru puncte:
  1. **Starea initiala** in care se afla agentul (de exemplu, Arad).
  2. **Actiuni** sau **functia succesor** S(x) – fiind data o stare x, S(x) intoarce multimile de stari in care se poate ajunge din x printr-o singura actiune (S(Arad) = {Zerind, Sibiu, Timisoara})
  3. **Testarea tintei problemei** – se verifica daca starea curenta a atins tinta problemei (x = Bucuresti, sah_mat(x))
  4. Functia de **cost al drumului** –
     - calculeaza un cost *g* pentru drumul curent (suma distantelor, numarul actiunilor executate etc).
     - c(x, y) – costul pasului, presupus sa fie ≥ 0
- O solutie este o secventa de actiuni care merg de la starea initiala la starea tinta

Algoritm general de cautare

functia cautare_generala(*problema, strategie*)
intoarce solutie sau esec
Initializeaza arborele de cautare folosind starea initiala a problemei.

Cat timp este posibil *executa*
Daca nu mai sunt posibilitati de incercat *atunci*
intoarce esec
Alege un nod posibil in concordanta cu strategia
Daca nodul contine o stare tinta *atunci*
intoarce solutia corespunzatoare
Altfel gaseste toate posibilitatile ce pornesc din acest nod si
adauga-le la arborele de cautare
Sfarsit cat timp

Algoritm general de cautare

functia cautare_generala(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

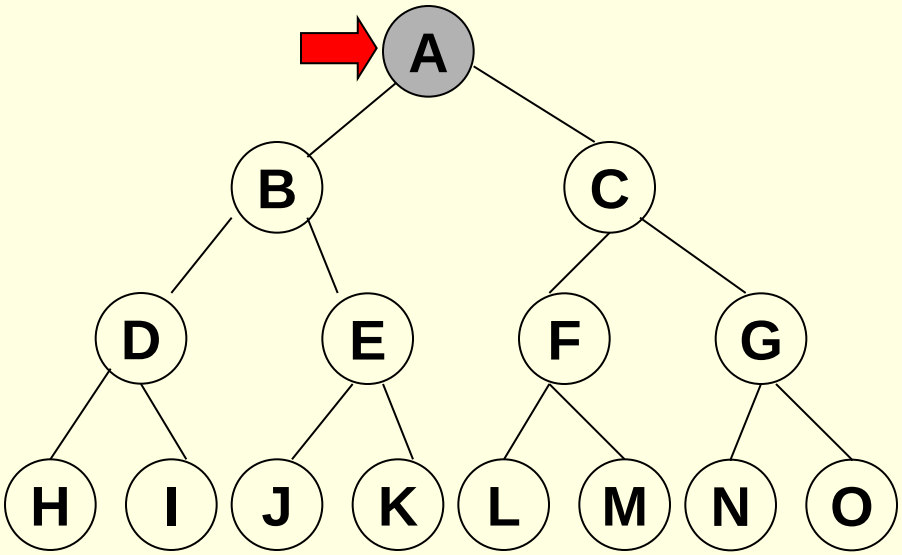
Cat timp este posibil *executa*
Daca noduri = vida *atunci*
    intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
    intoarce nod
Altfel
  noduri = adauga(noduri, expandare(nod, actiuni[problema]))
Sfarsit cat timp

Algoritm cautare in latime

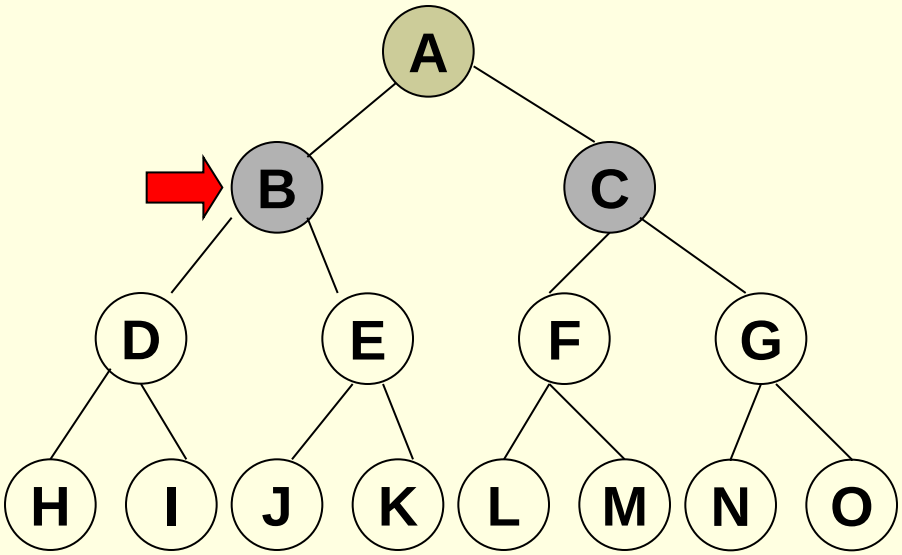
functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp

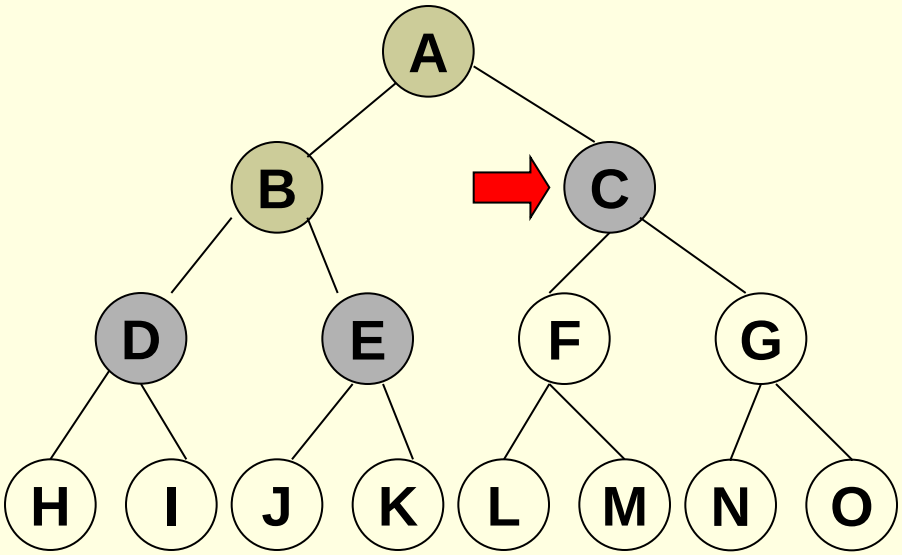
Cautarea in latime



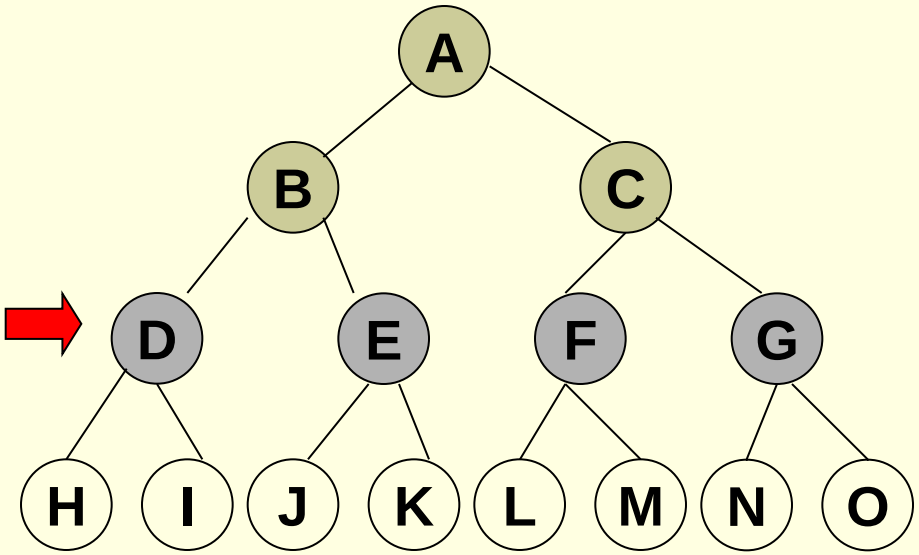
Cautarea in latime



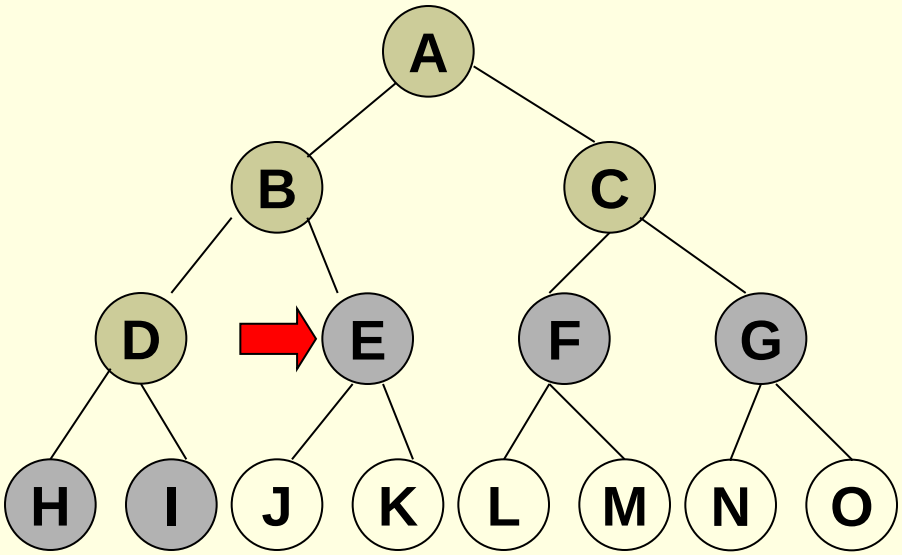
Cautarea in latime



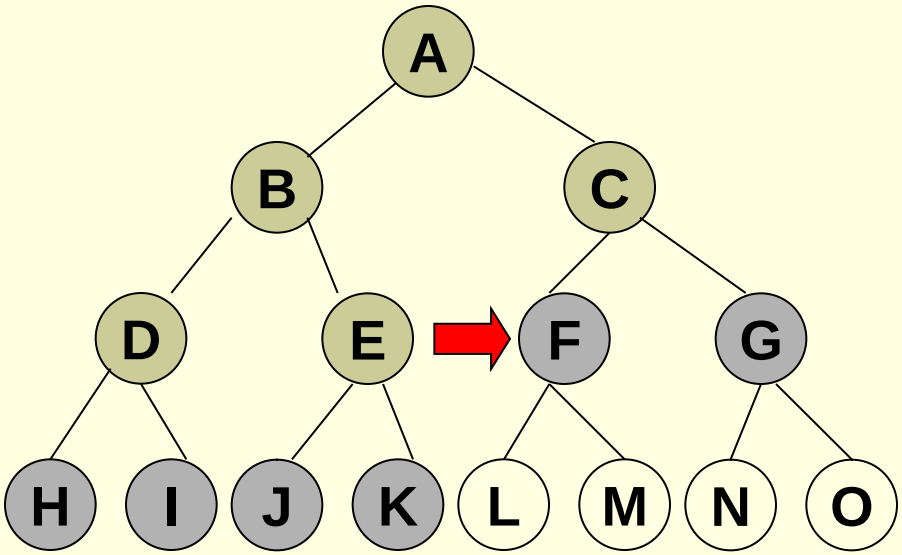
Cautarea in latime



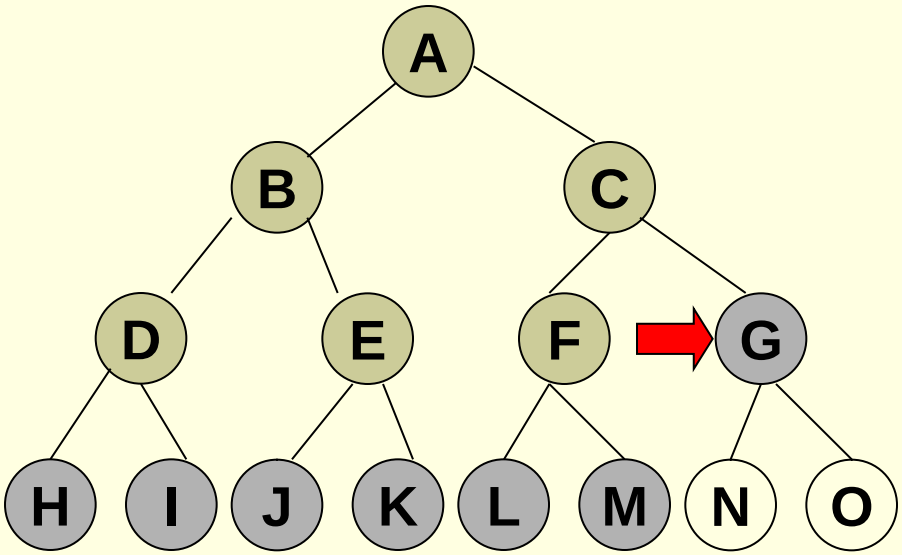
Cautarea in latime



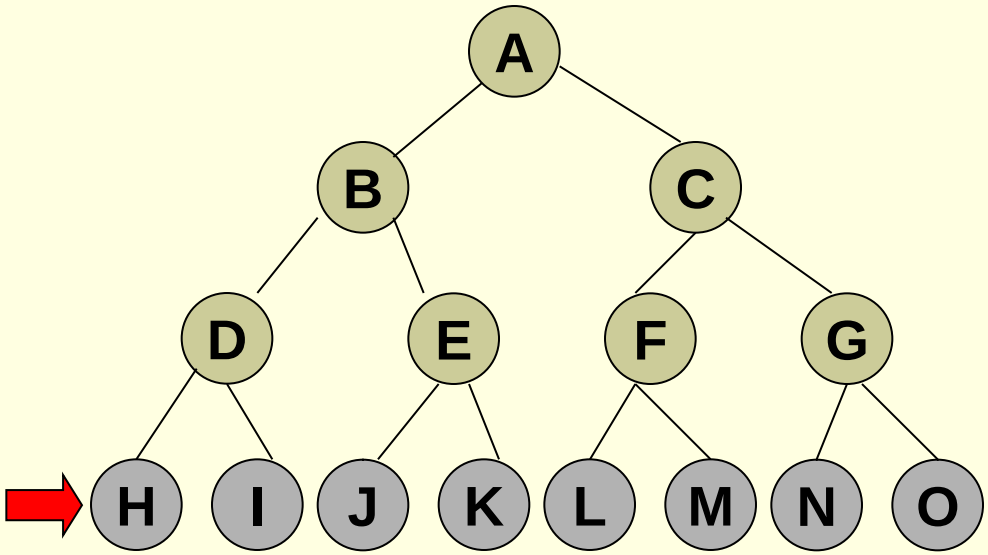
Cautarea in latime



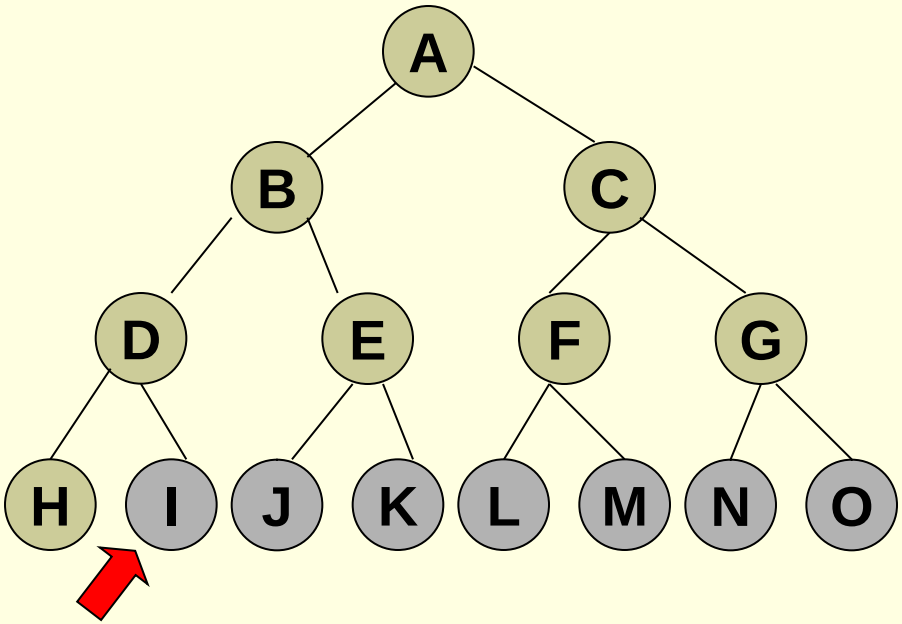
Cautarea in latime



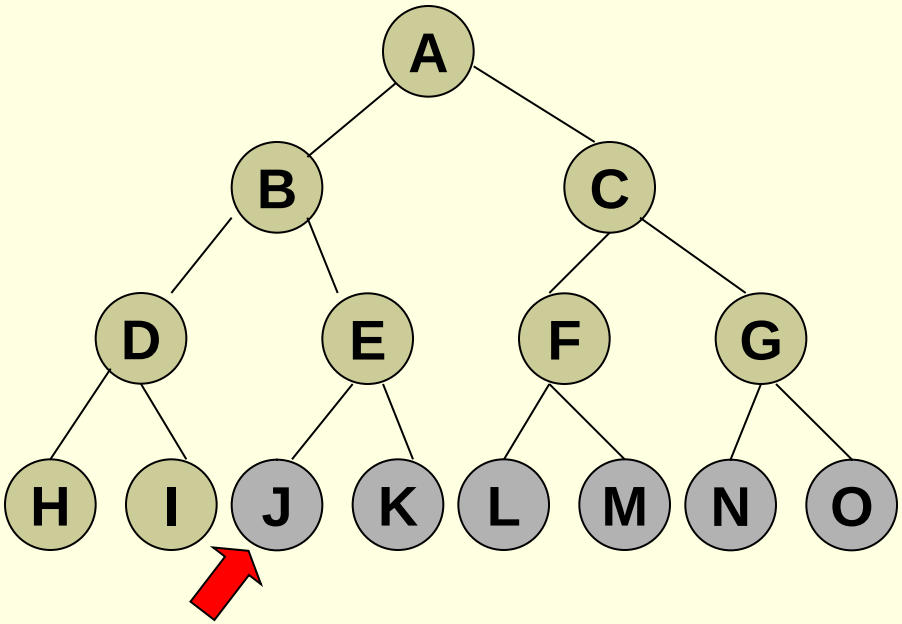
Cautarea in latime



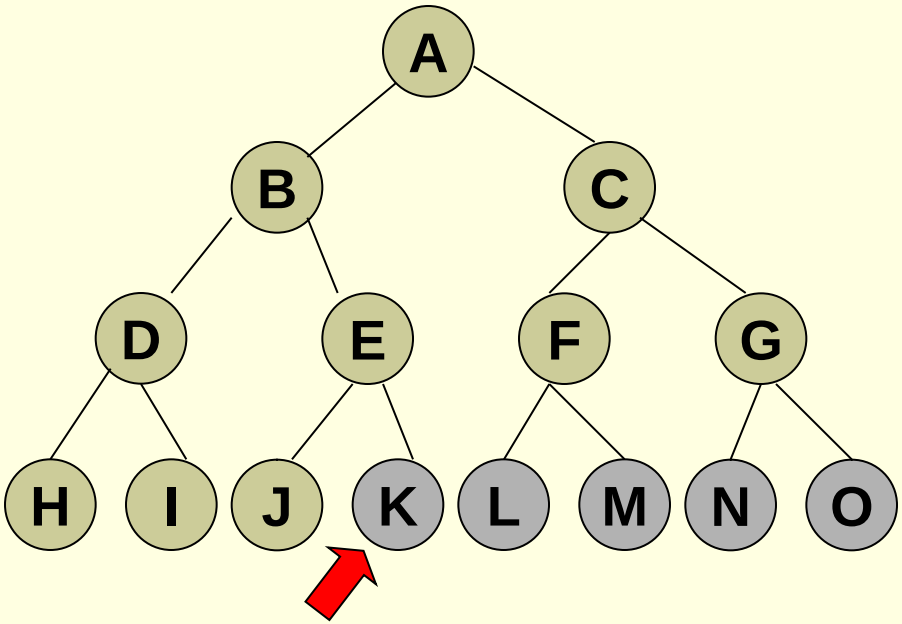
Cautarea in latime



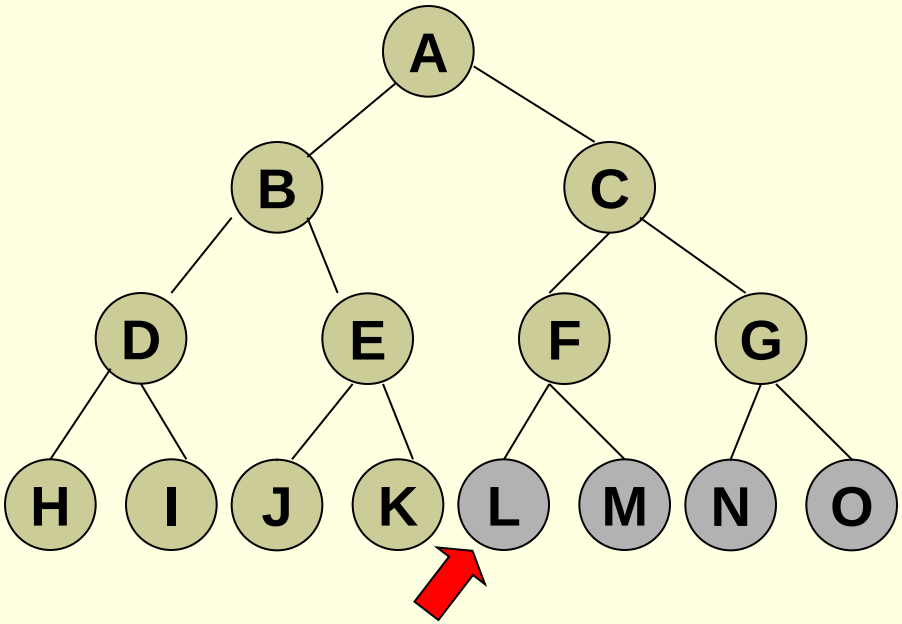
Cautarea in latime



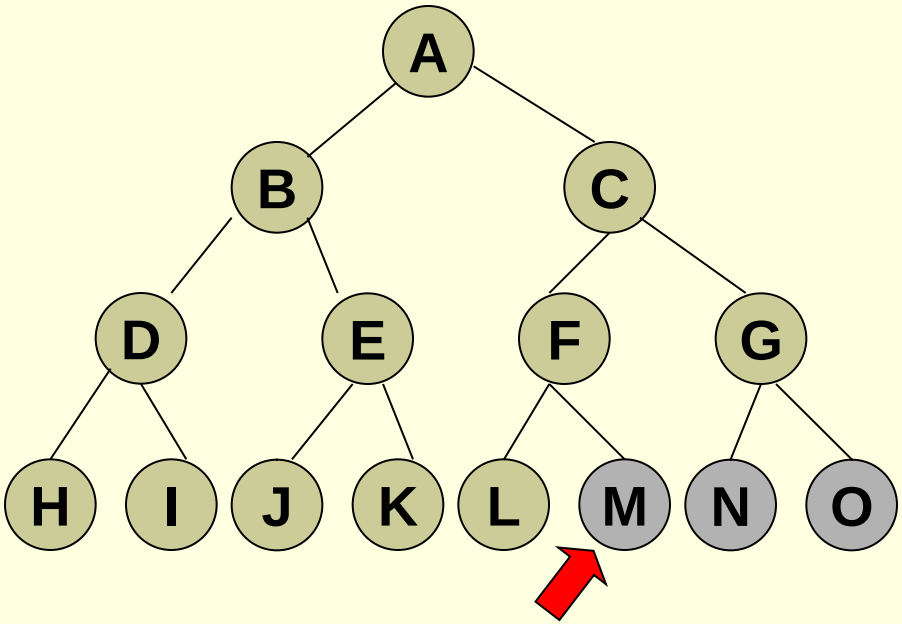
Cautarea in latime



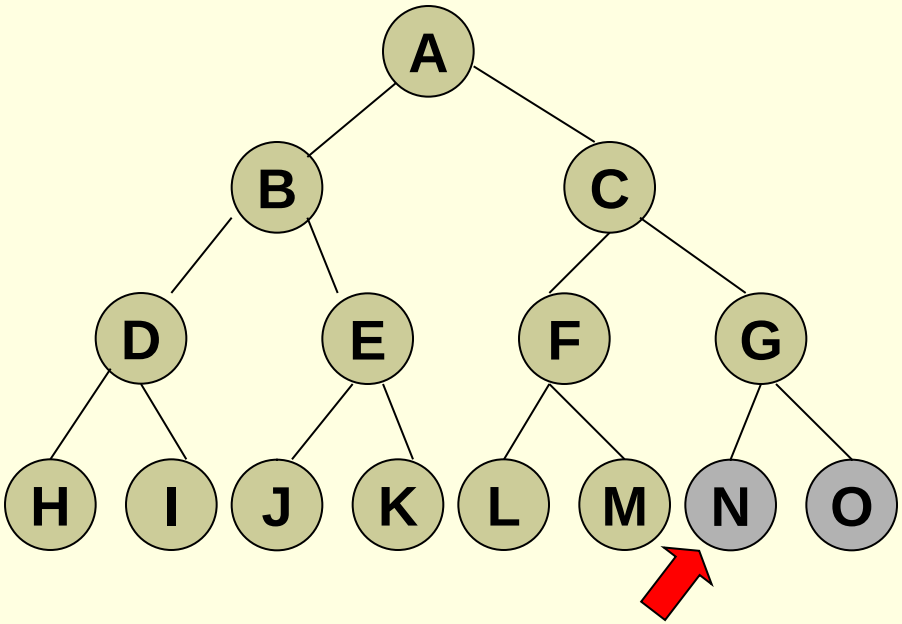
Cautarea in latime



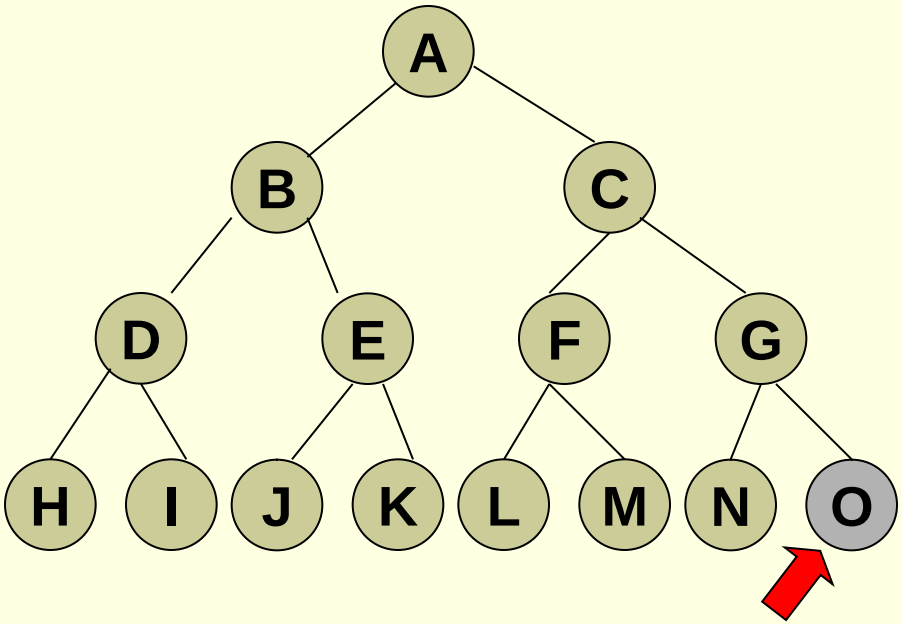
Cautarea in latime



Cautarea in latime



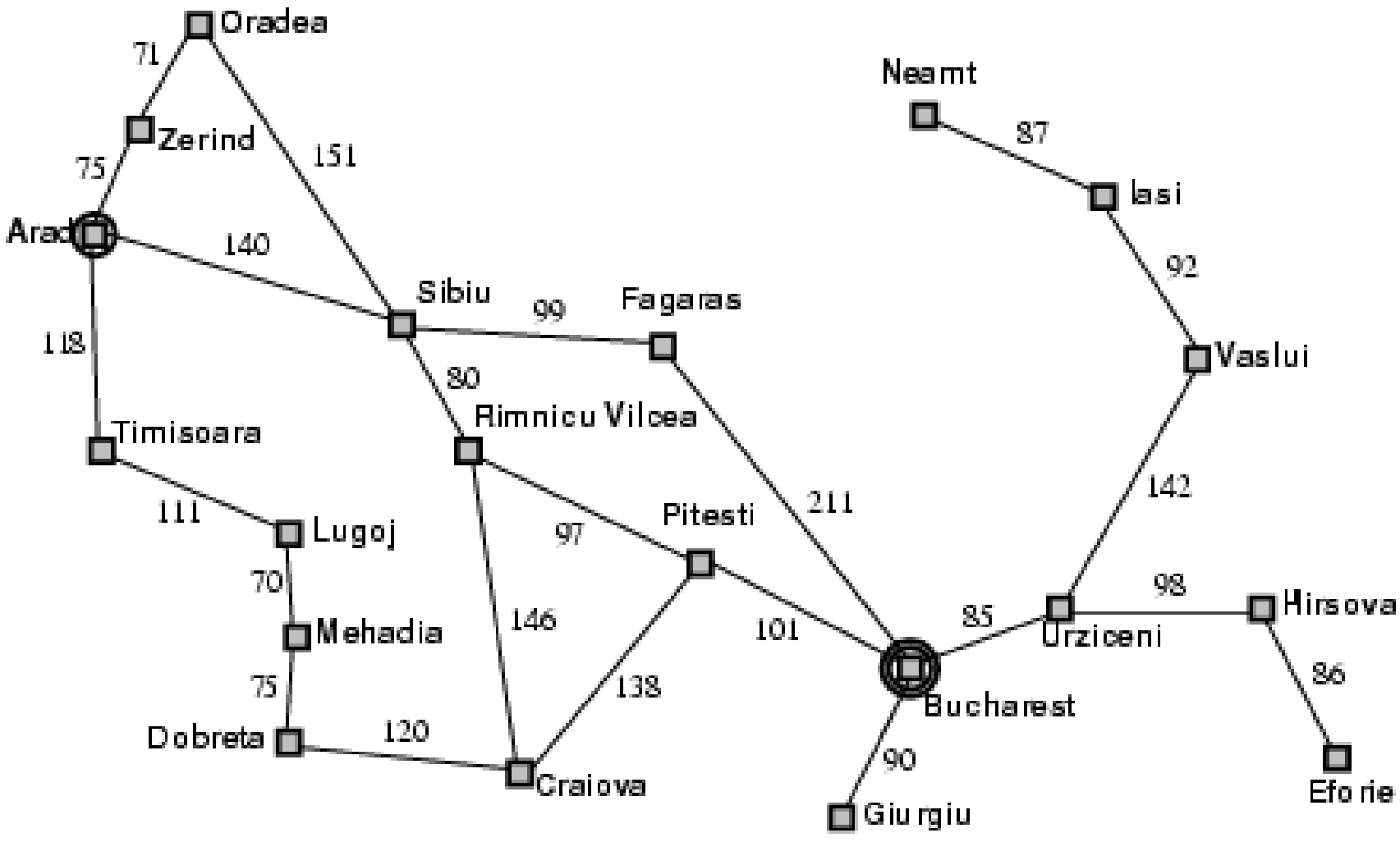
Cautarea in latime



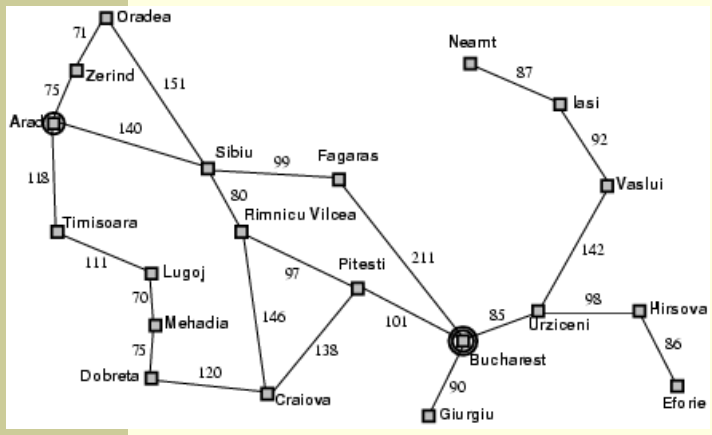
Problema de rutare: un agent *american*

- Vacanta in Romania – in Arad.
- In ziua urmatoare ii pleaca avionul din Bucuresti.
- **Formularea scopului:**
  - Ajungerea in Bucuresti
- **Formularea problemei:**
  - **Stari:** diverse orase
  - **Actiuni:** de a merge dintr-un oras in altul
- **Gasirea solutiei:**
  - O secventa de orase, de ex: Arad, Sibiu, Fagaras, Bucuresti.

Un agent *american*



Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp

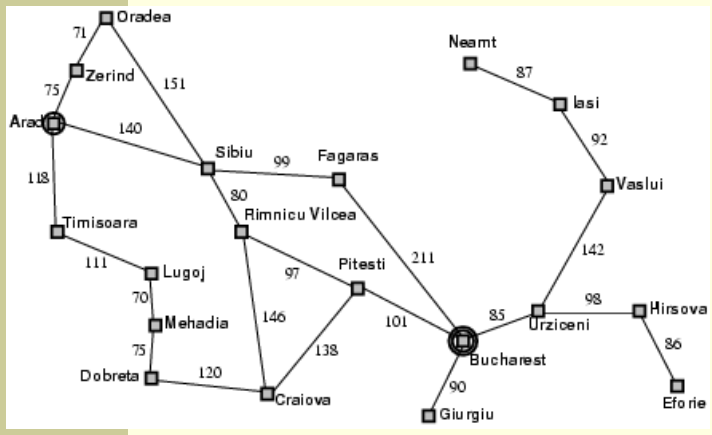
Arad

Faţa **noduri** Sfarsit

Arad

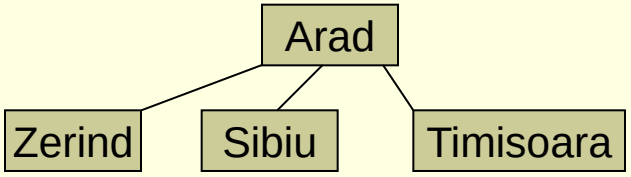
Parcurgerea: Arad,

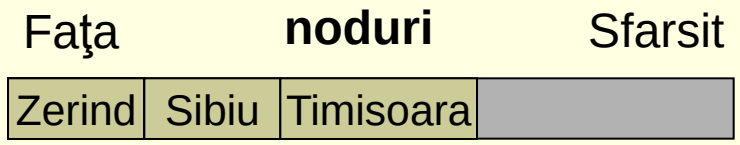
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

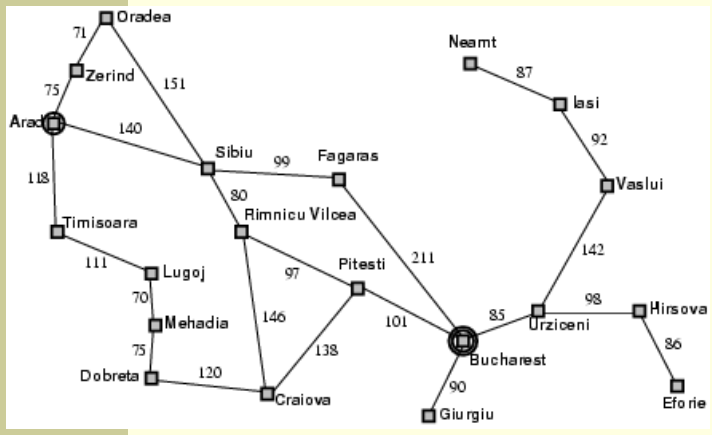
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp





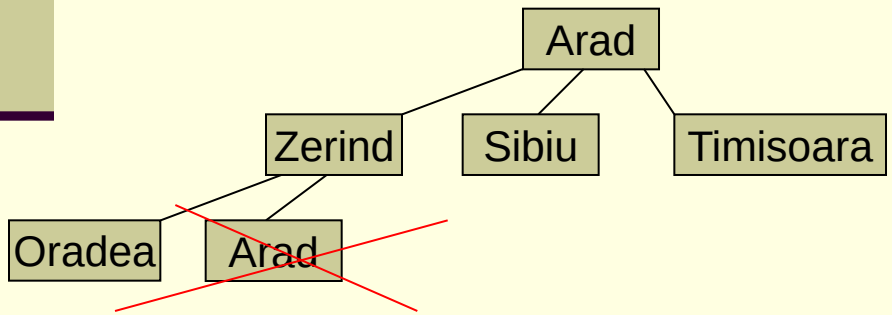
Parcurgerea: Arad, Zerind

Algoritm cautare in latime

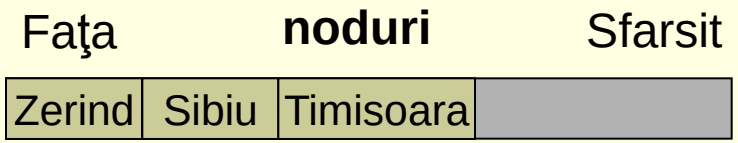


functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp

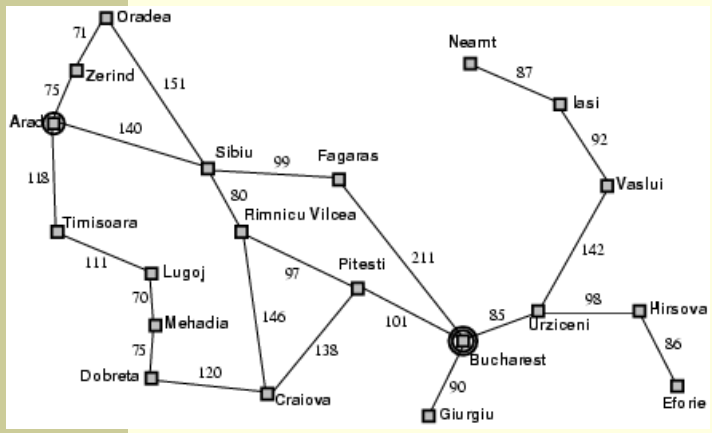


A mai aparut acest nod!



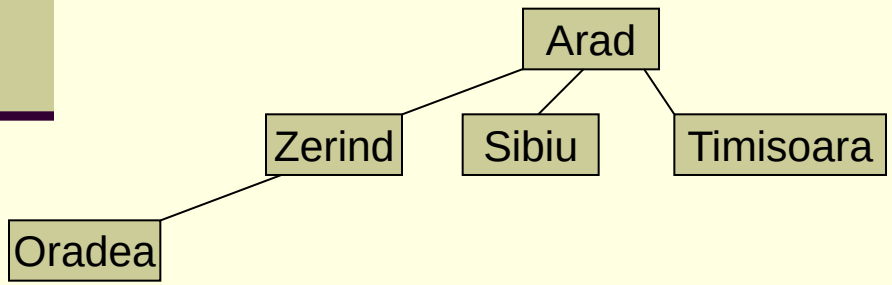
Parcurgerea: Arad, Zerind

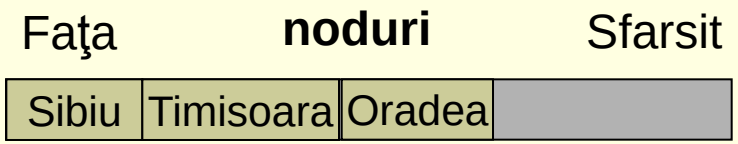
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

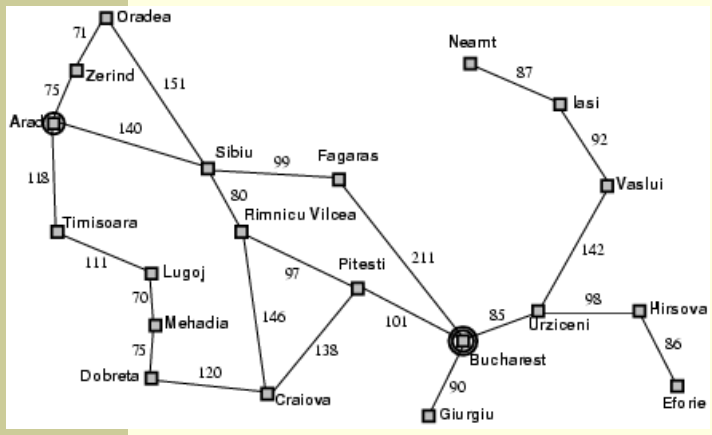
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp





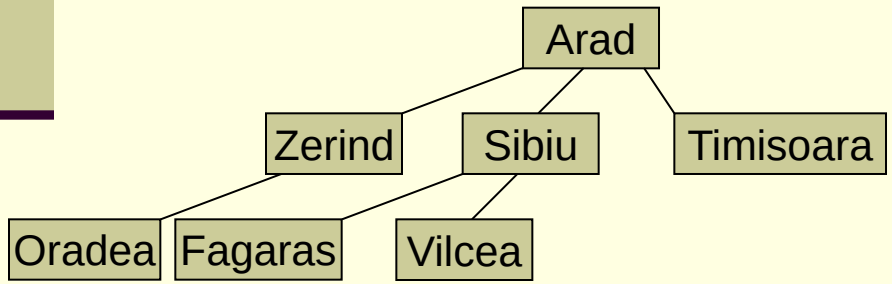
Parcurgerea: Arad, Zerind, Sibiu

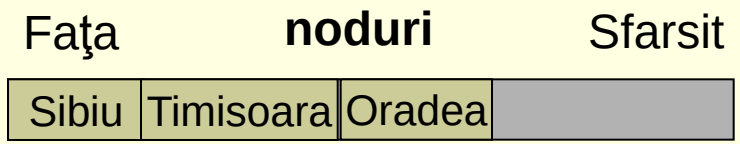
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

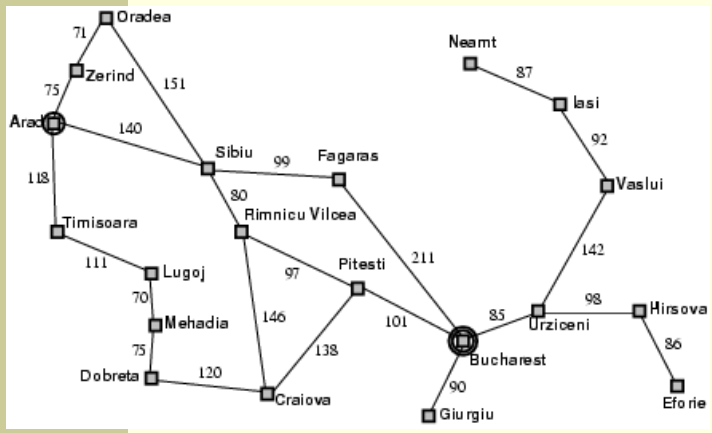
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp





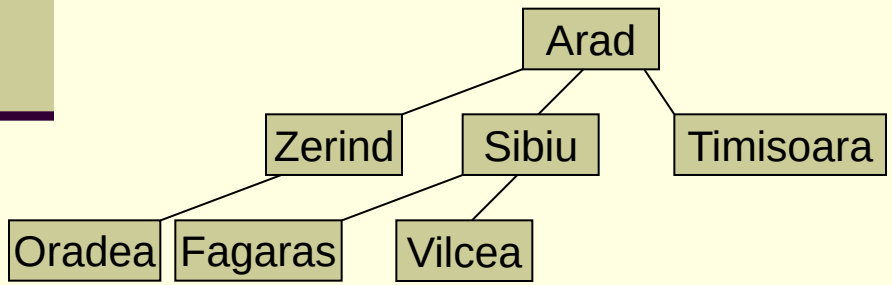
Parcurgerea: Arad, Zerind, Sibiu

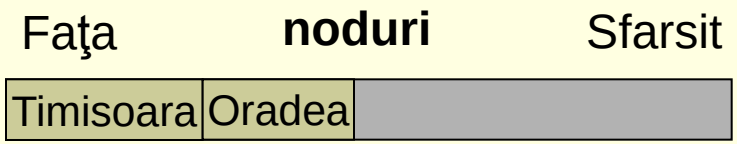
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

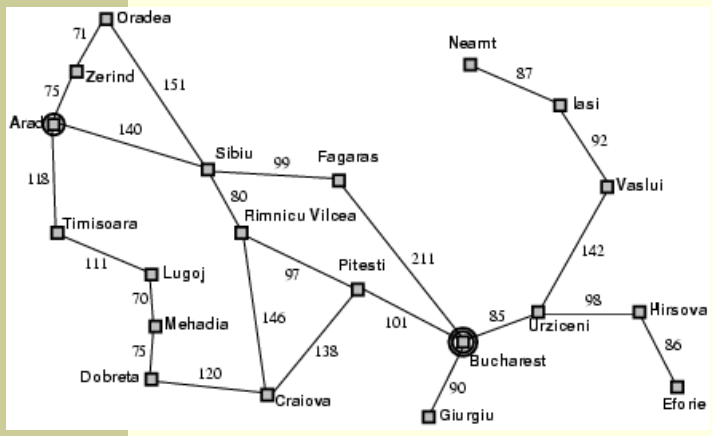
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp





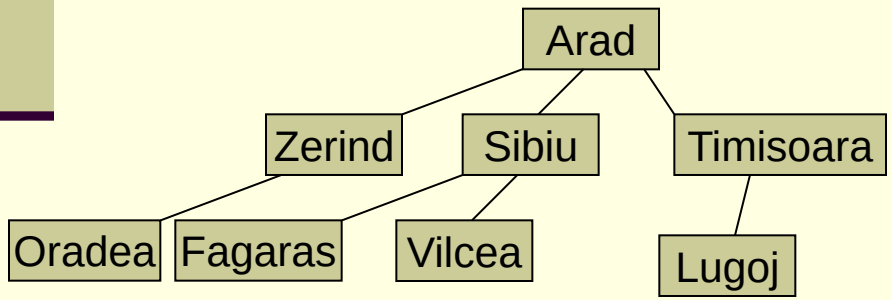
Parcurgerea: Arad, Zerind, Sibiu, Timisoara

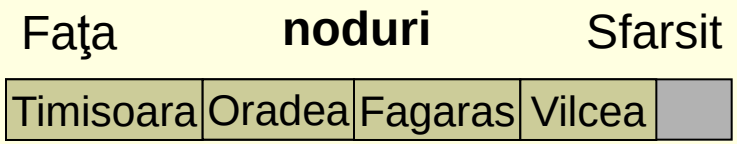
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

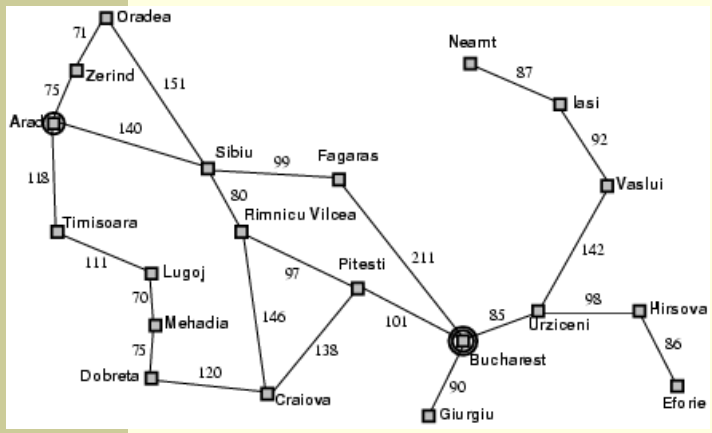
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp





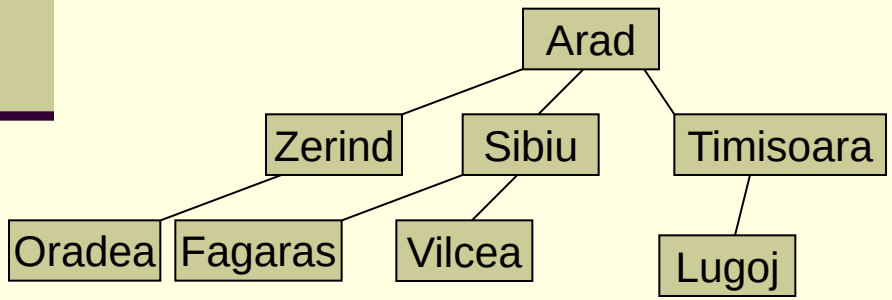
Parcurgerea: Arad, Zerind, Sibiu, Timisoara

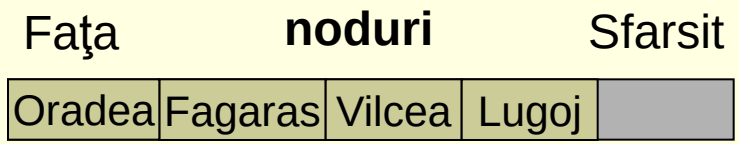
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

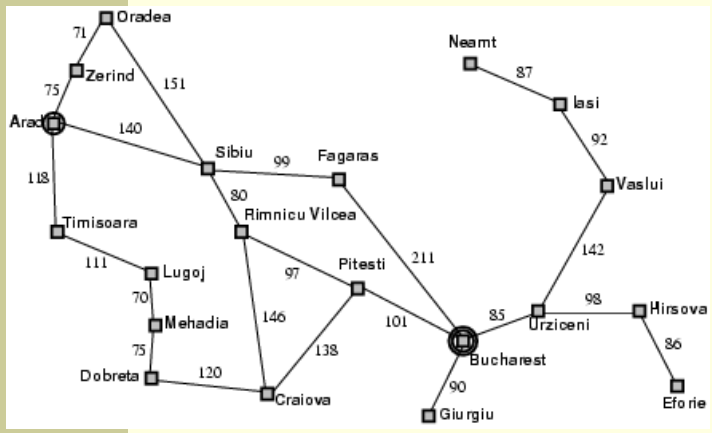
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp



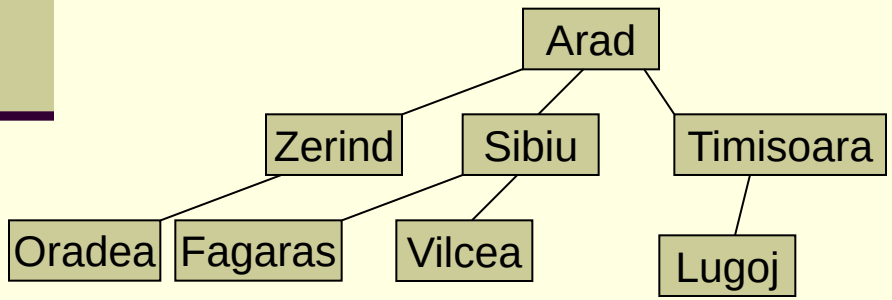


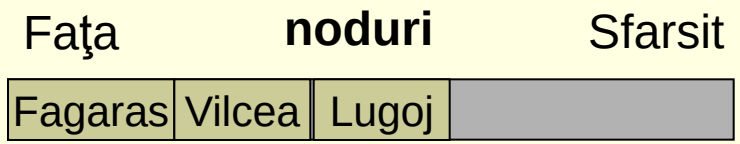
Parcurgerea: Arad, Zerind, Sibiu, Timisoara, Oradea

Algoritm cautare in latime



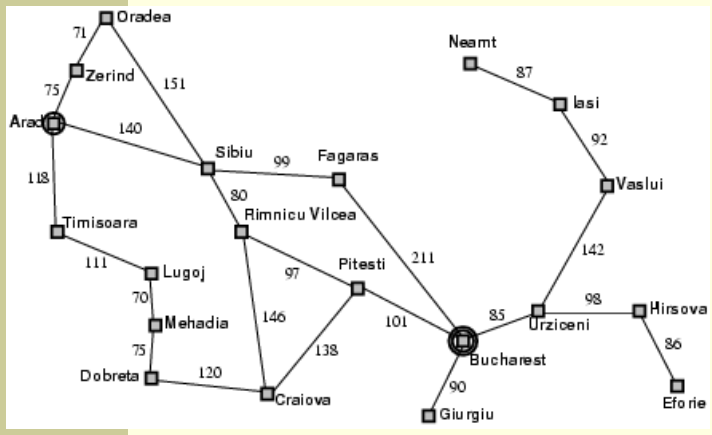

```
functia cautare_latime(problema) intoarce solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil executa
        Daca noduri = vida atunci
                intoarce esec
        nod = scoate_din_fata(noduri)
        Daca testare_tinta[problema] se aplica la stare(nod) atunci
                intoarce nod
        Altfel
          noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp
```





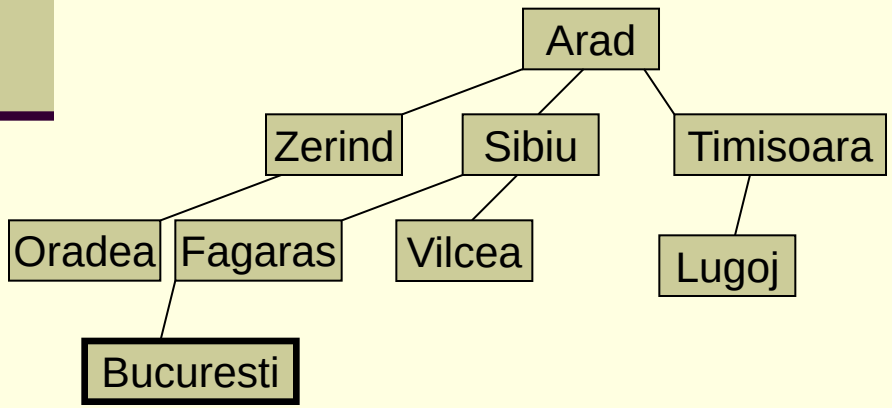
Parcurgerea: Arad, Zerind, Sibiu, Timisoara, Oradea, Fagaras

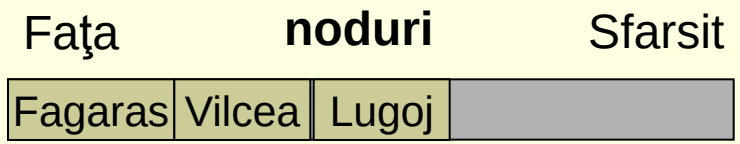
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

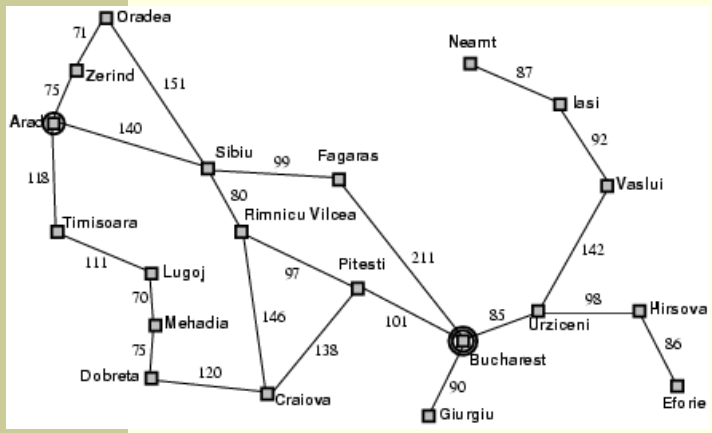
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp





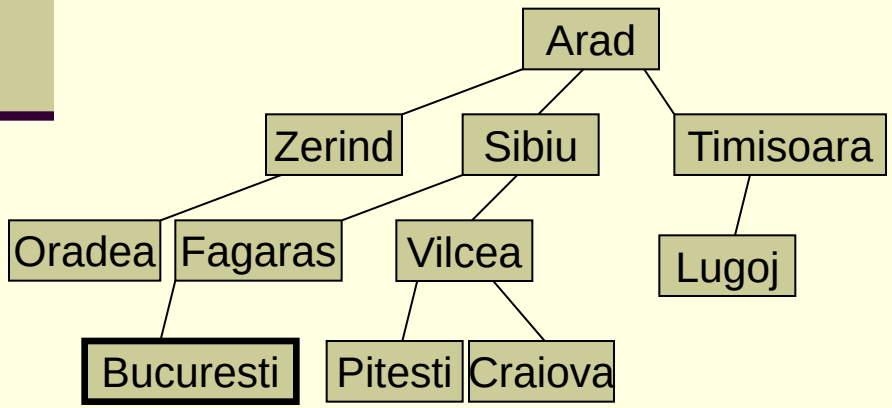
Parcurgerea: Arad, Zerind, Sibiu, Timisoara, Oradea, Fagaras

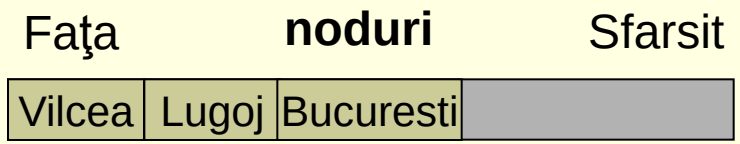
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

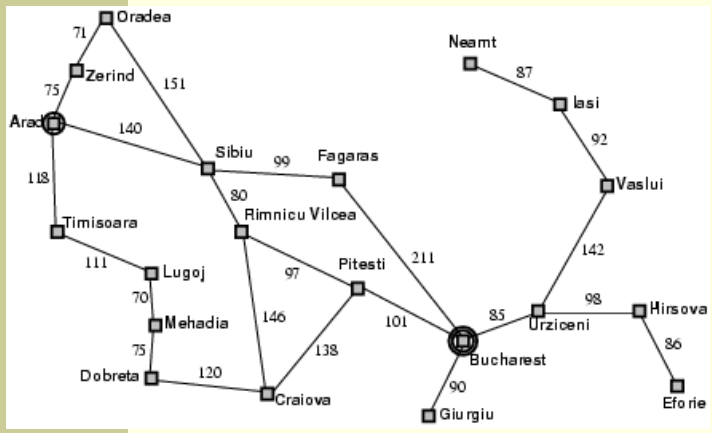
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp





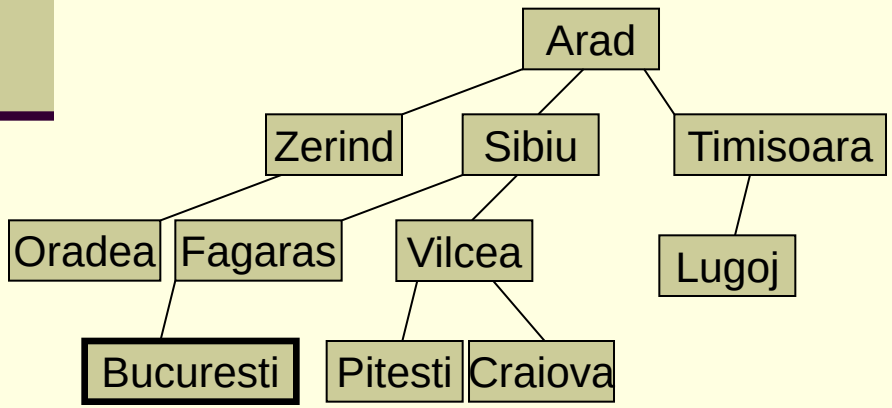
Parcurgerea: Arad, Zerind, Sibiu, Timisoara, Oradea, Fagaras, Vilcea

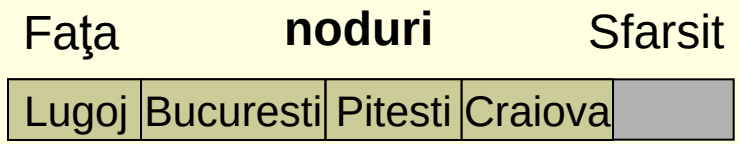
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

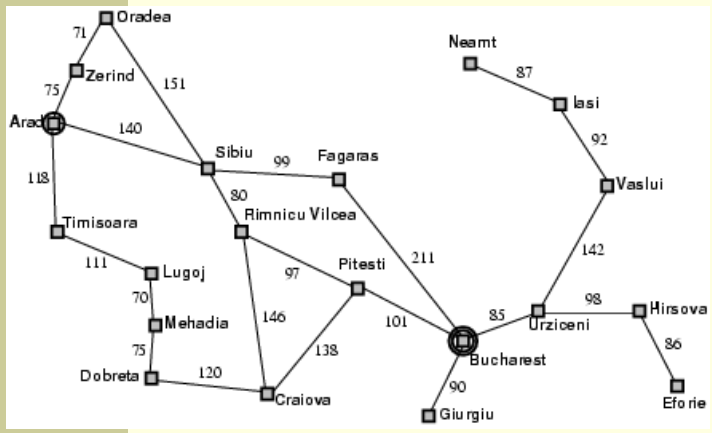
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp





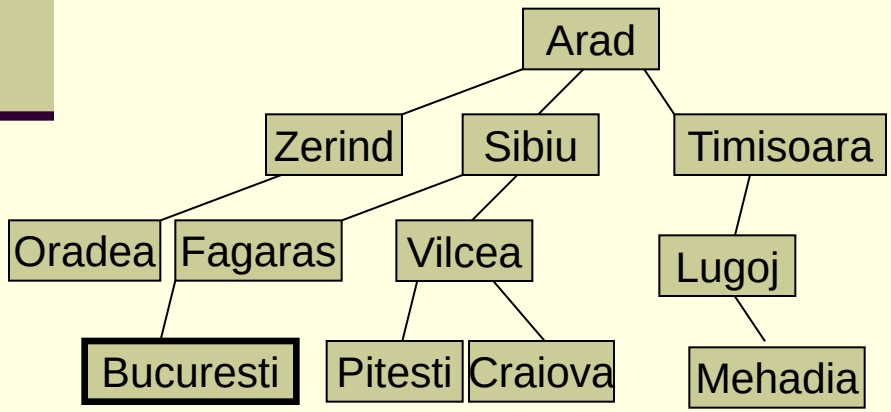
Parcurgerea: Arad, Zerind, Sibiu, Timisoara, Oradea, Fagaras, Vilcea, Lugoj

Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit cat timp

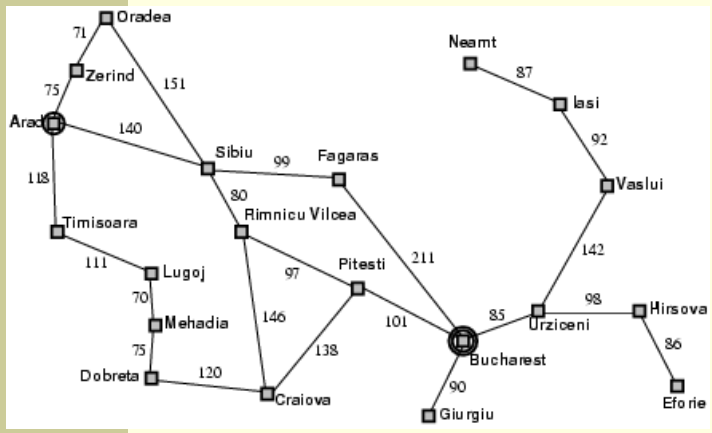


Faţa **noduri** Sfarsit

Lugoj	Bucuresti	Pitesti	Craiova	

Parcurgerea: Arad, Zerind, Sibiu, Timisoara, Oradea, Fagaras, Vilcea, Lugoj

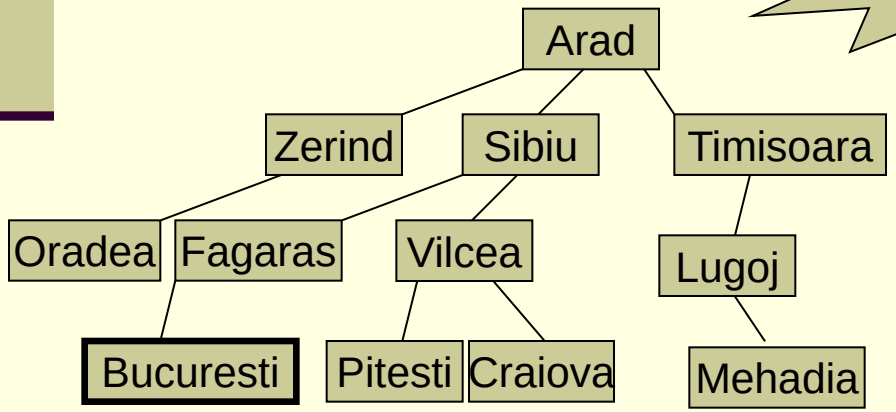
Algoritm cautare in latime



functia cautare_latime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_sfarsit))
Sfarsit ca

Nod tinta!

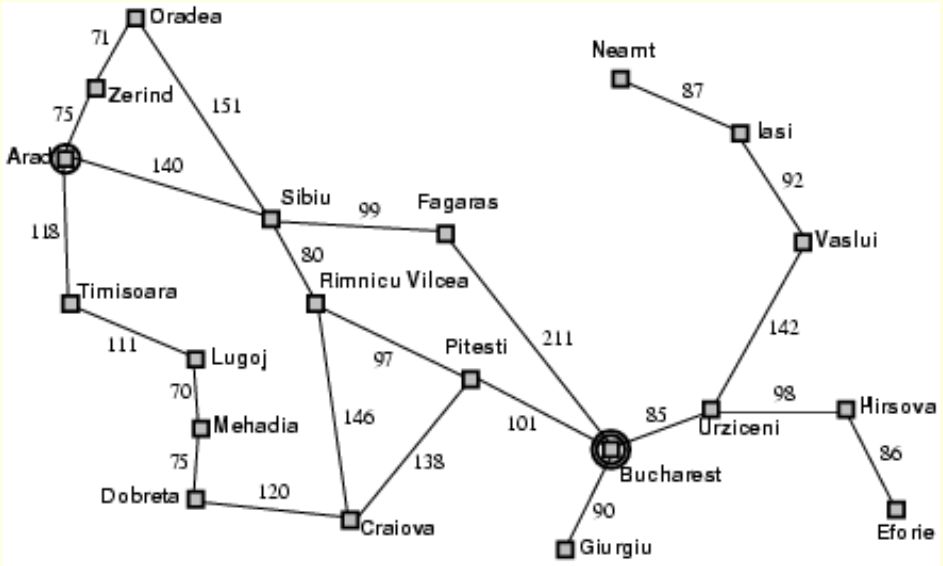


Faţa **noduri** Sfarsit

Bucuresti	Pitesti	Craiova	Mehadia	

Parcurgerea: Arad, Zerind, Sibiu, Timisoara, Oradea, Fagaras, Vilcea, Lugoj, **Bucuresti**!

Exercitiu



Faţa **noduri** Sfarsit

Gasiti o ruta de la Bucuresti la Sibiu folosind parcurgerea in latime. Desenati arborele, scrieti parcurgerea si continutul pentru noduri la fiecare pas.

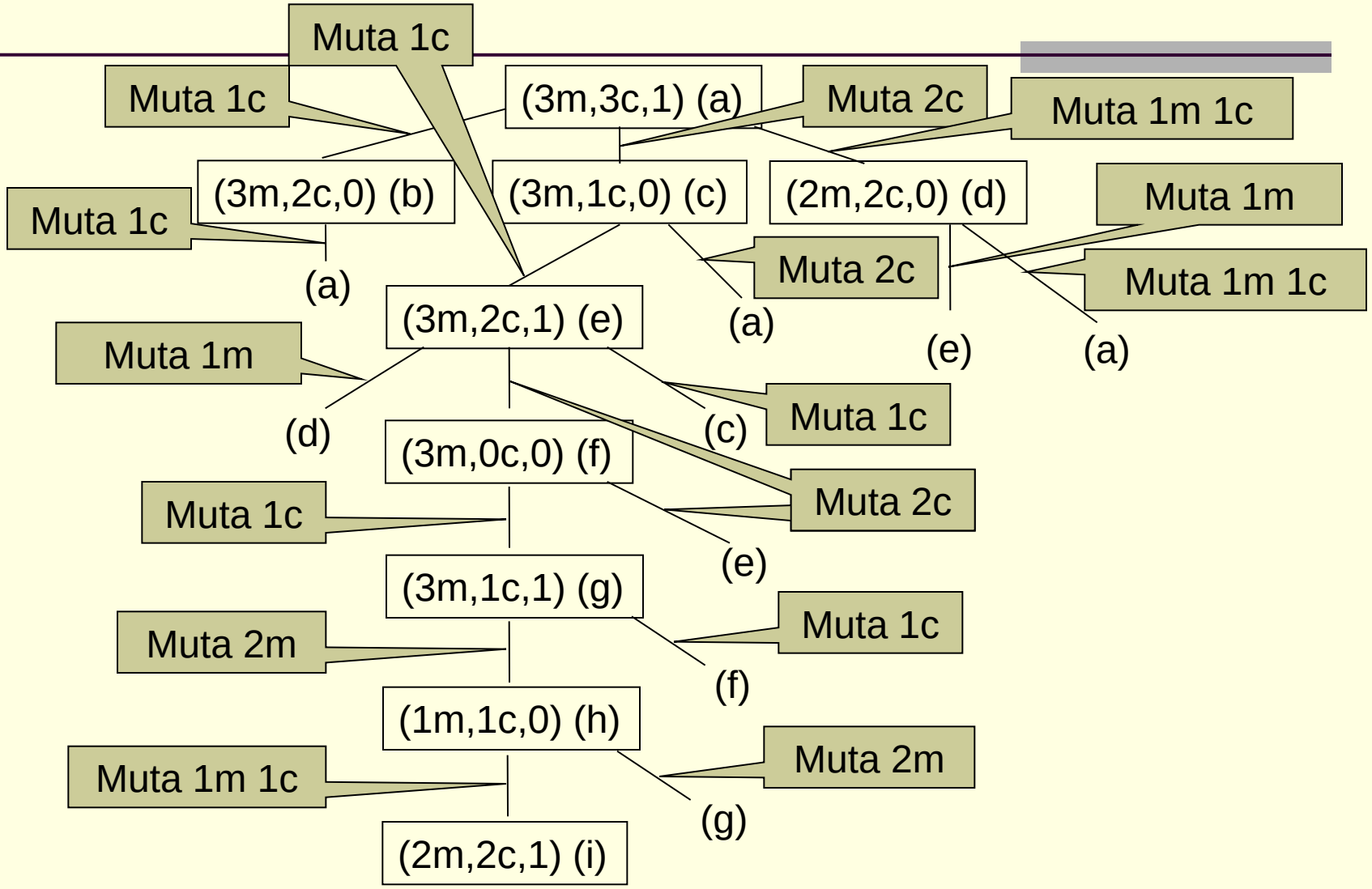
Parcurgerea: Bucuresti, …, Rm. Vilcea

Misionarii si canibalii

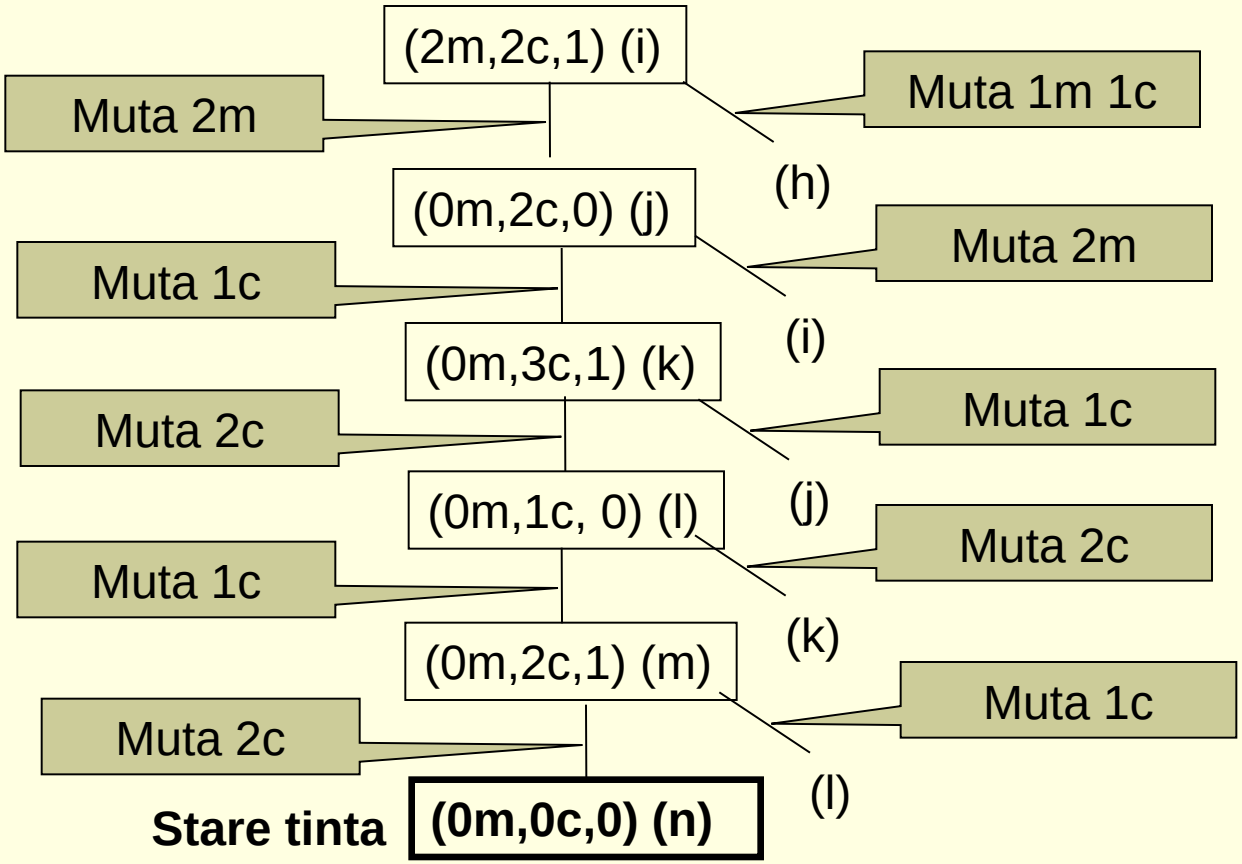
- Trei misionari si trei canibali se afla de o parte a raului. Ei au o barca ce poate duce cel mult doi oameni. Gasiti o posibilitate sa traverseze toti de cealalta parte a raului cu conditia sa nu existe mai multi canibali decat misionari intr-o parte.

- **Stari**: secvente ordonate de 3 numere reprezentand numarul de misionari, canibali si barci de partea in care se aflau initial (3, 3, 1).
- **Actiuni**: mutarea unui misionar sau canibal sau 2 canibali, 2 misionari sau un misionar si un canibal de pe o parte pe alta.
- **Testarea tintei**: starea (0, 0, 0).
- **Costul drumului**: numarul de traversari.

Misionarii si canibalii



Misionarii si canibalii



Puzzle cu 8 valori

	2	3
1	4	6
7	5	8

Starea initiala

1	2	3
4	5	6
7	8	

Starea tinta

- **Stari**: este descrisa locatia fiecarei cifre in una din cele 9 casute.
- **Actiuni**: casuta goala se misca la stanga, dreapta, sus sau jos.
- **Testarea tintei**: starea se gaseste in configuratia din dreapta.
- **Costul drumului**: fiecare pas are costul 1, deci costul drumului este dat de numarul de mutari.

Puzzle cu 15 valori

- Sam Loyd a introdus in 1878 15-puzzle-ul.
- A oferit un premiu de 1 000 $ din proprii bani pentru cel care rezolva problema in situatia in care doar 14 si 15 sunt inversate, restul fiind aliniate crescator.

Puzzle cu 15 valori

1	2	3	4
5	6	7	8
9	10	11	12
13	15	14	

?

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Nimeni nu a castigat premiul!

Puzzle cu 8 valori

- Cate stari posibile exista?
  - 9! = 362 880 stari
- Numai la **jumatate** din aceste stari se poate ajunge din orice stare data.

Posibilitatea de a atinge scopul final...

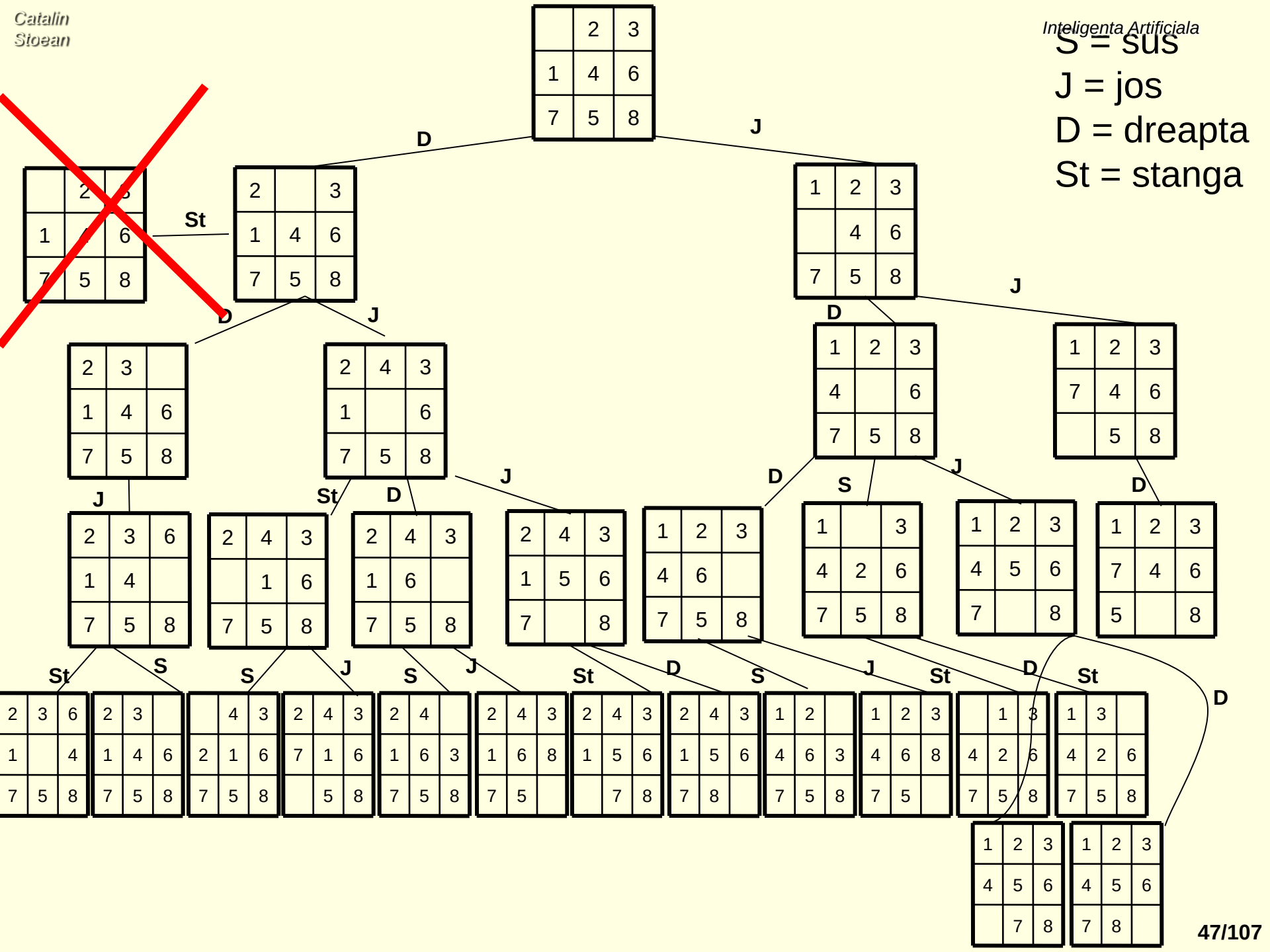
- O valoare *j* apare dupa o valoare *i* daca
  - *j* apare pe acelasi rand, dar in dreapta lui *i*
  - *j* se afla sub linia lui *i*
- Pentru $i = 1, 2, ..., 8$, fie n_i numarul de aparitii ale lui *j* (*j* < *i*) dupa *i*.
- $N = n_2 + n_3 + \ldots + n_8$

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

$n_2 = 1$
$n_3 = 1$
$n_4 = 0$
$n_5 = 0$
$n_6 = 1$
$n_7 = 1$
$n_8 = 0$

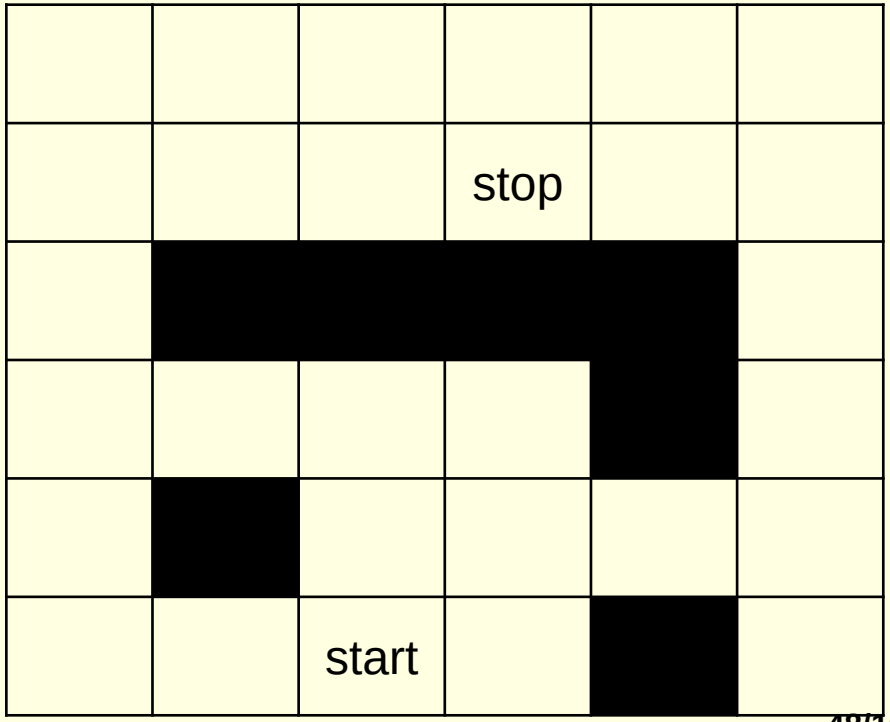
$N = 4$
Pentru *N* par, puzzle-ul cu 8 valori este rezolvabil!

S = sus
J = jos
D = dreapta
St = stanga



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.
- Etichetati cu litere in ordine alfabetica patratele daca se utilizeaza o cautare in adancime, iar ordinea operatiilor este: sus, stanga, dreapta si jos.



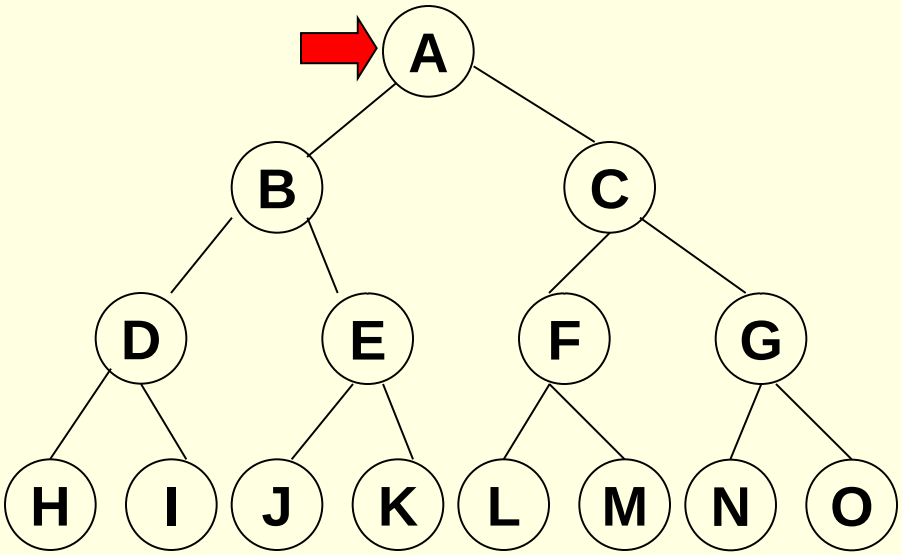
Exercitiu - Problema celor 4 dame

- **Stari**: orice aranjament de 0 pana la 4 dame care nu se ataca.
- **Actiuni**: adauga o dama pe coloana cea mai din stanga a.i. sa nu fie atacata de alta dama.
- **Testarea tintei**: 4 dame care nu se ataca pe tabla.
- **Costul drumului**: 0.

- Pornind de la o tabla de 4x4 goala si folosind datele problemei de mai sus, sa se construiasca printr-o cautare in latime arborele complet care duce la rezolvarea problemei. Numerotati nodurile in ordinea in care au fost vizitate.

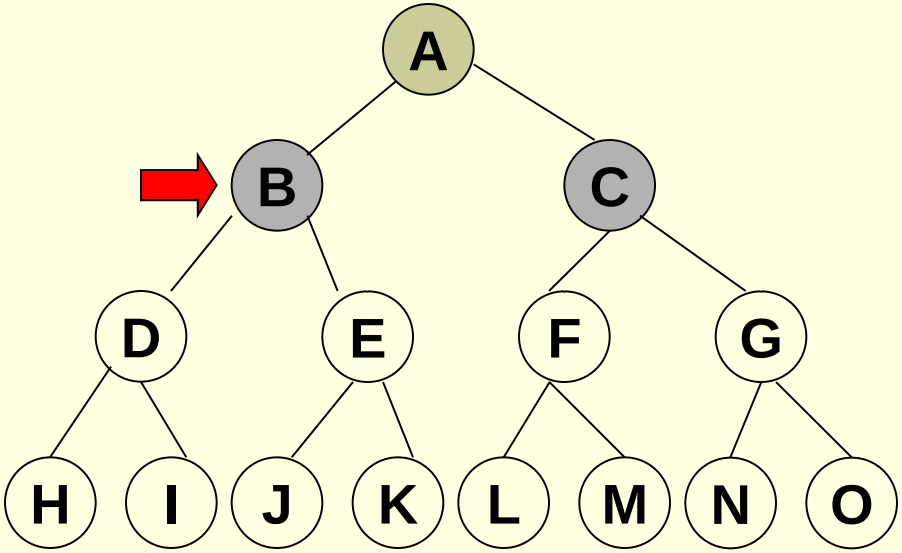
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



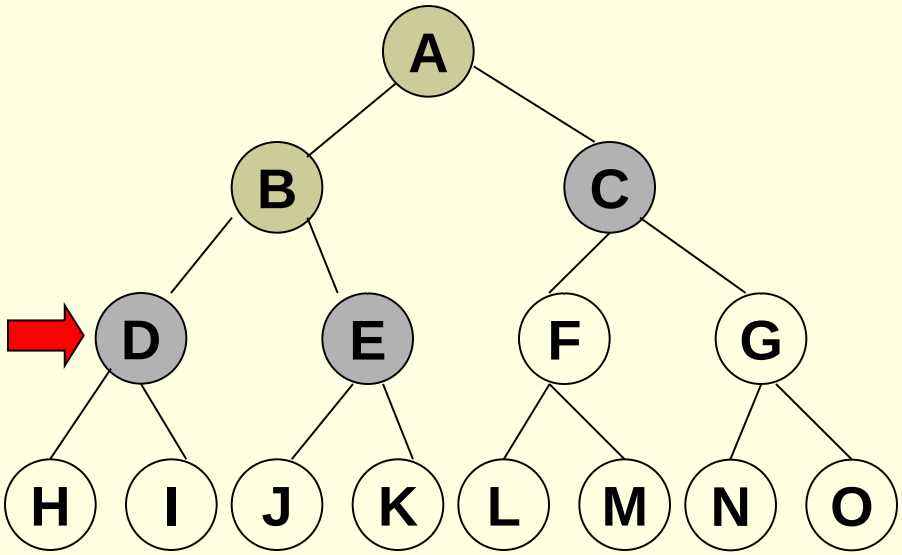
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



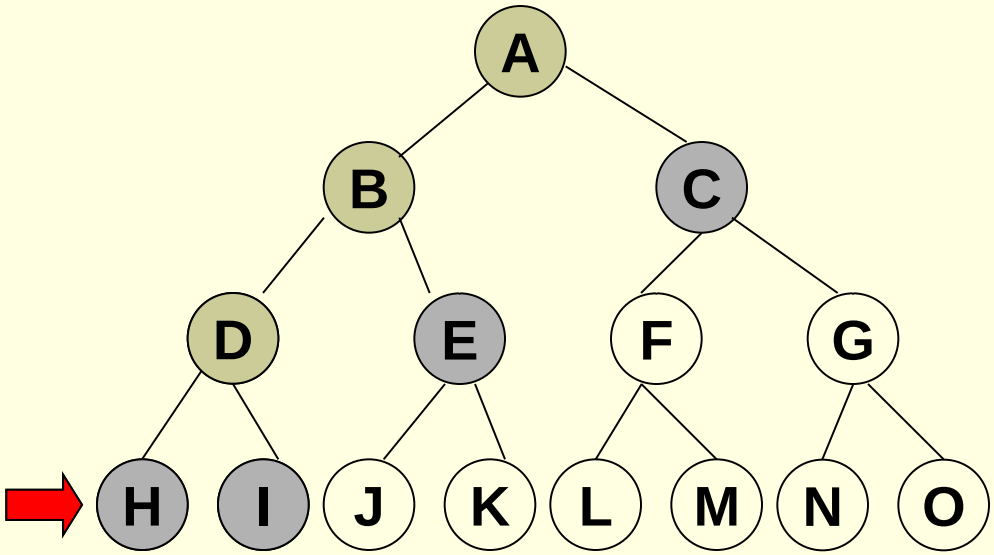
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



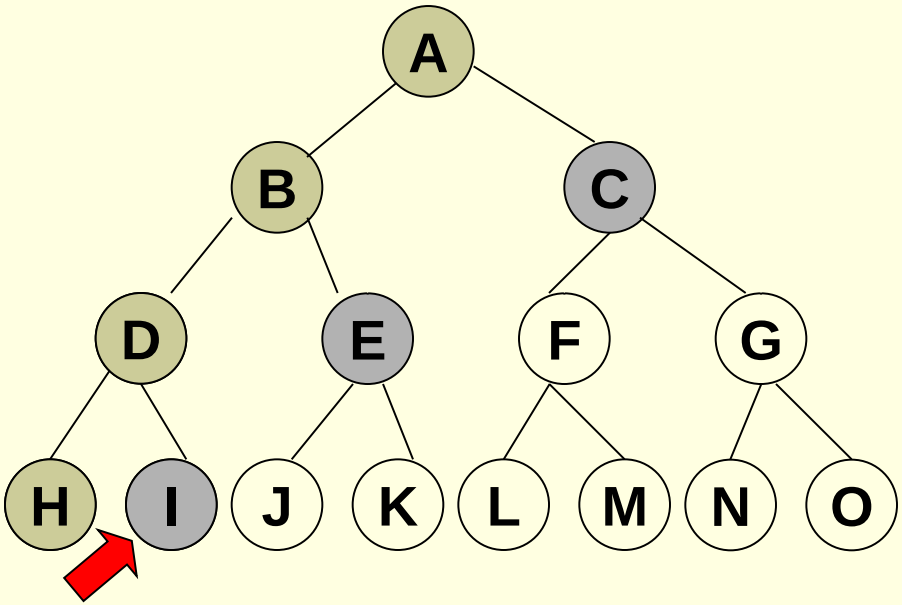
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



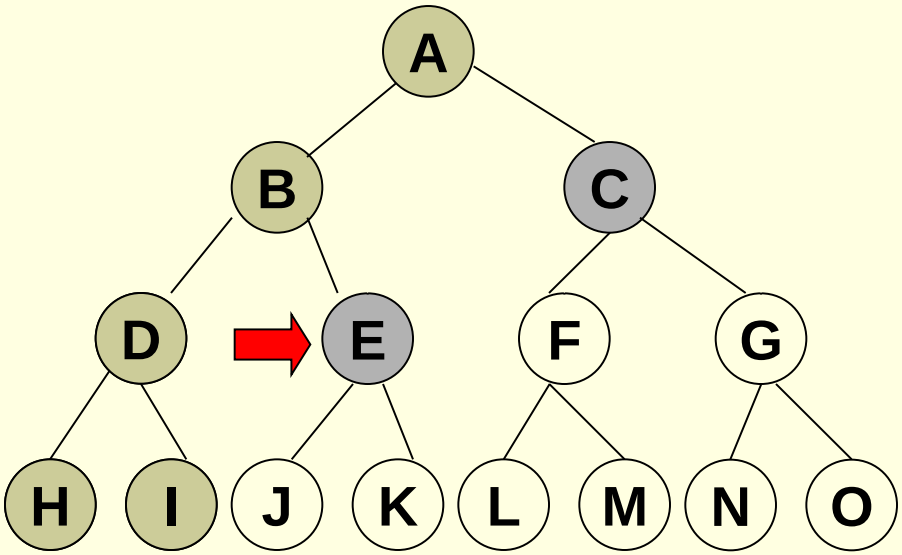
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



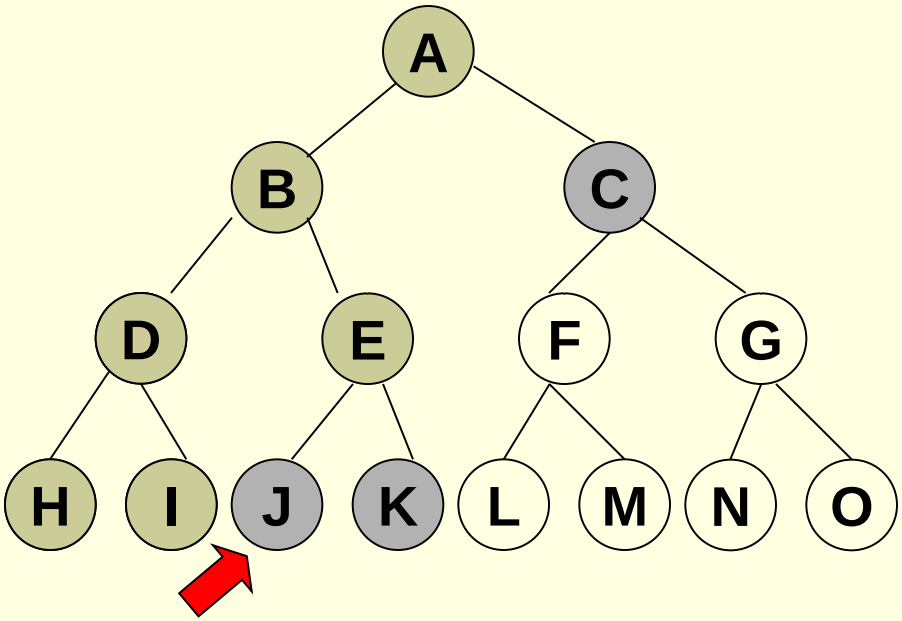
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



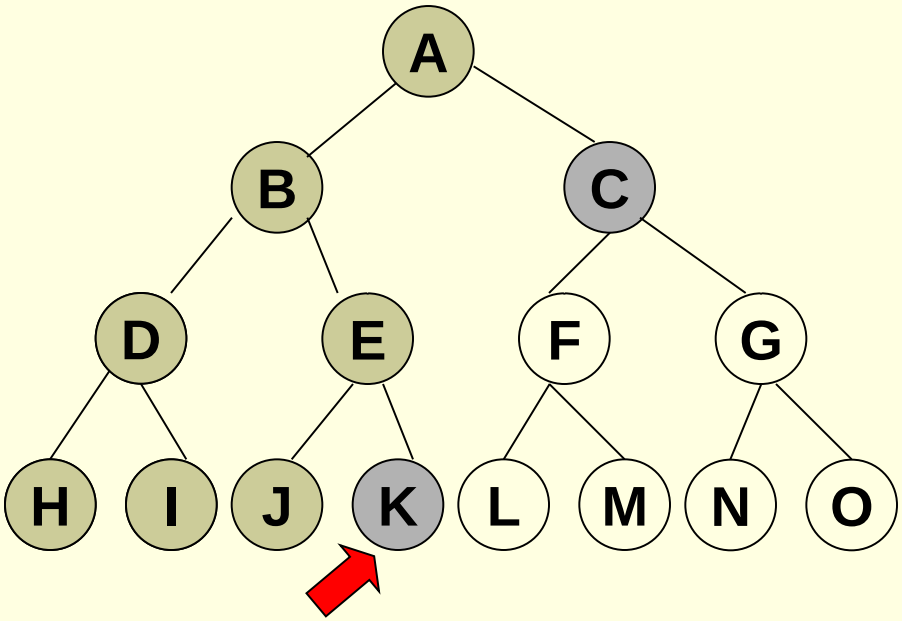
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



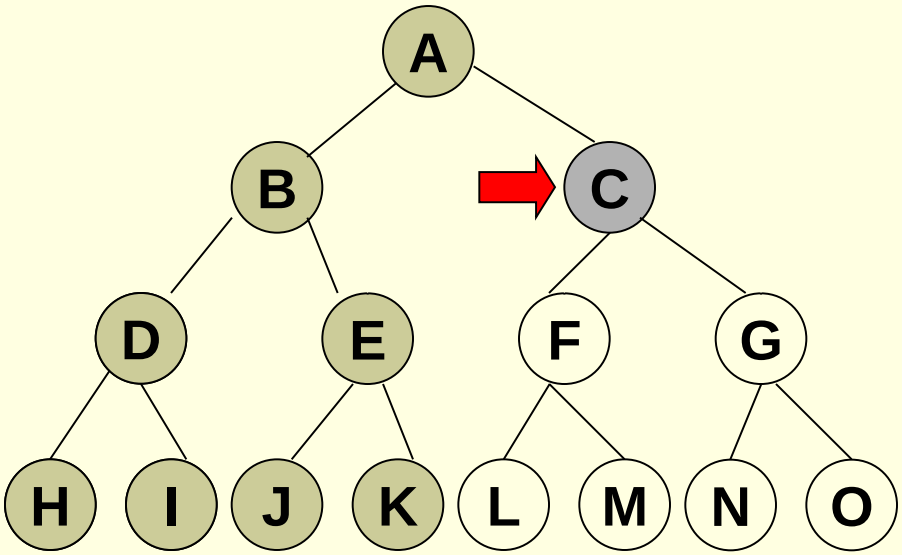
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



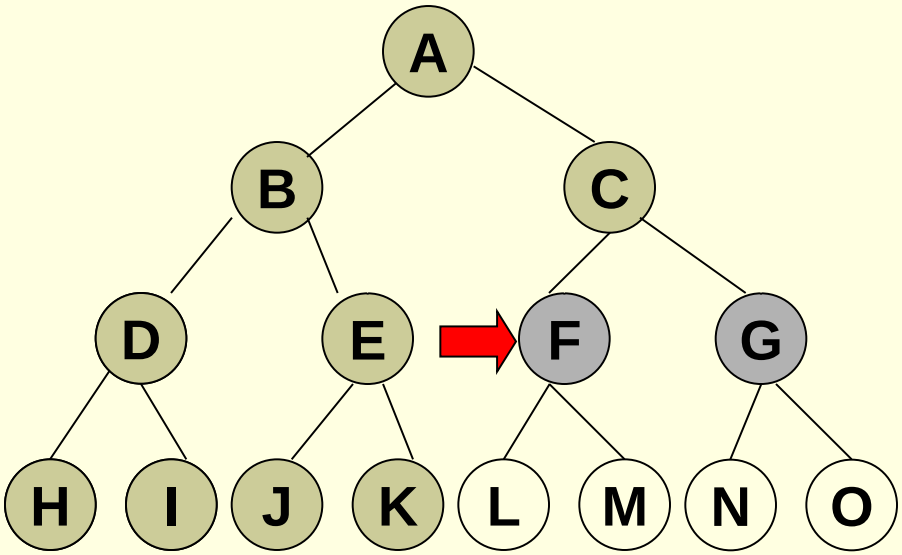
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



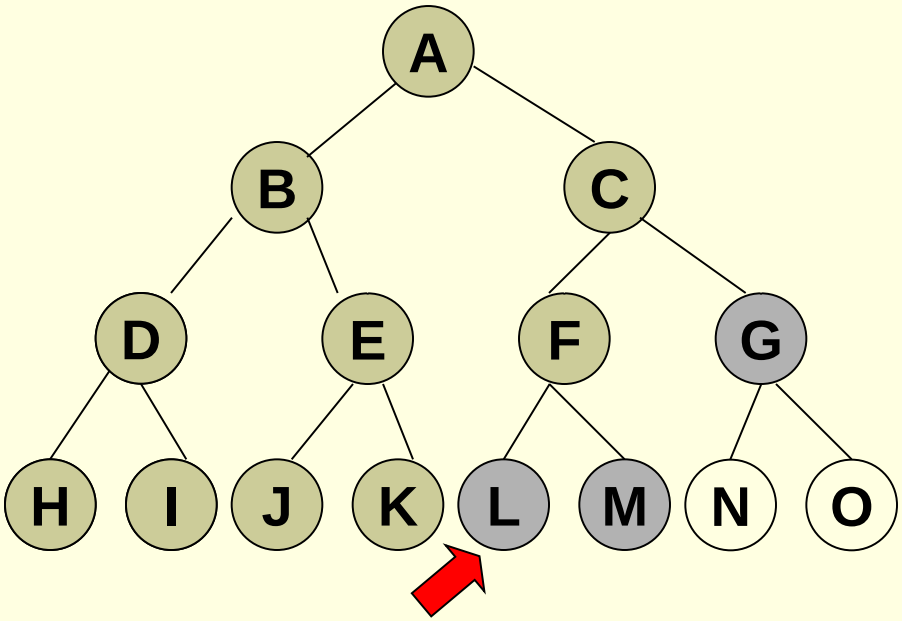
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



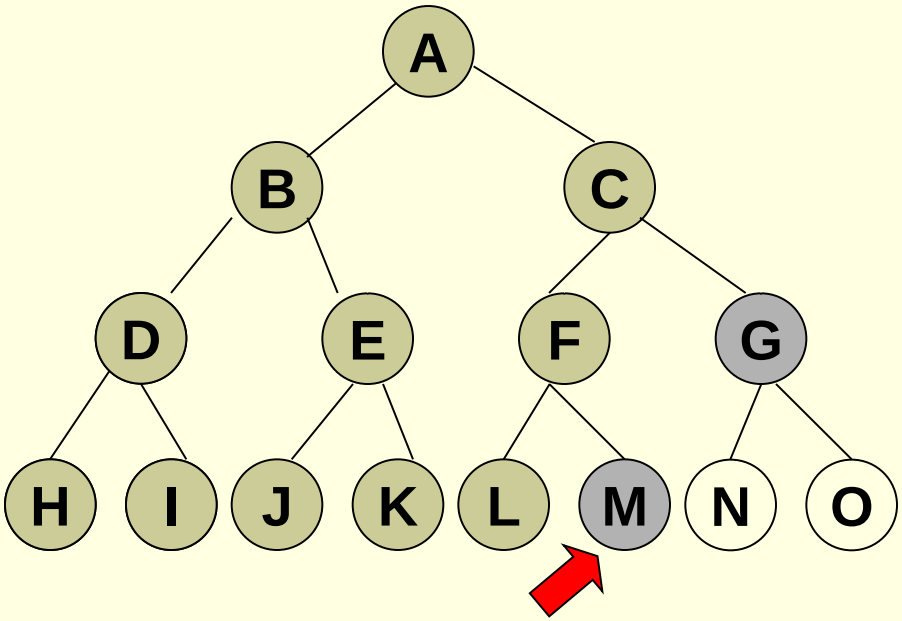
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



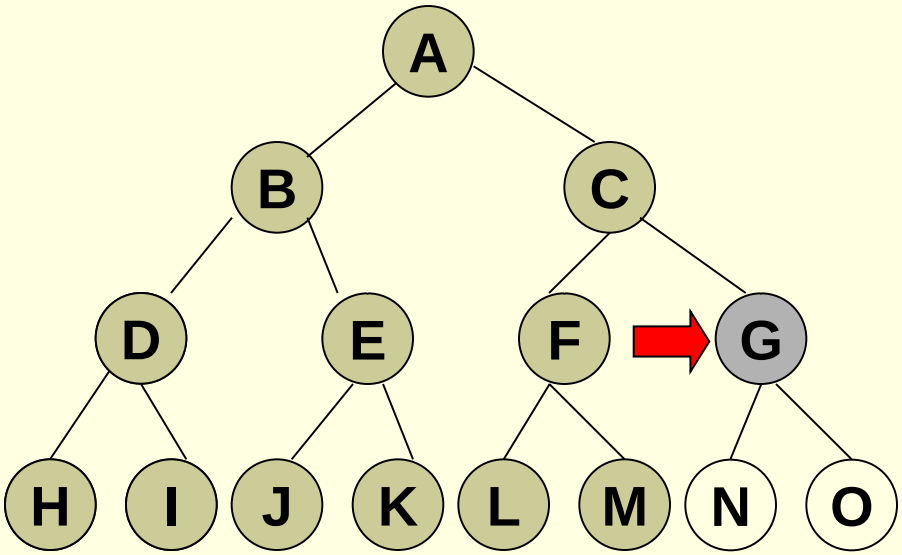
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



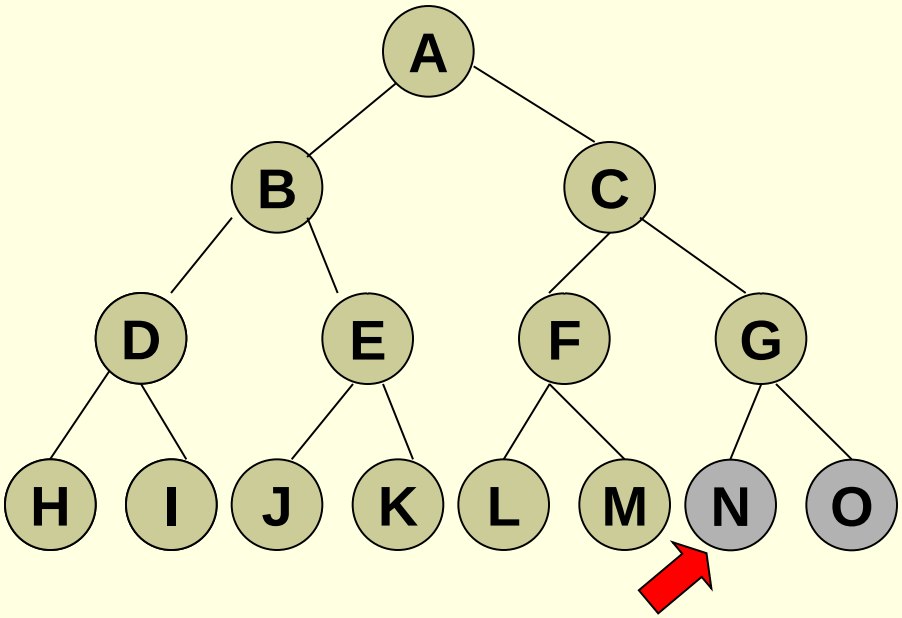
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



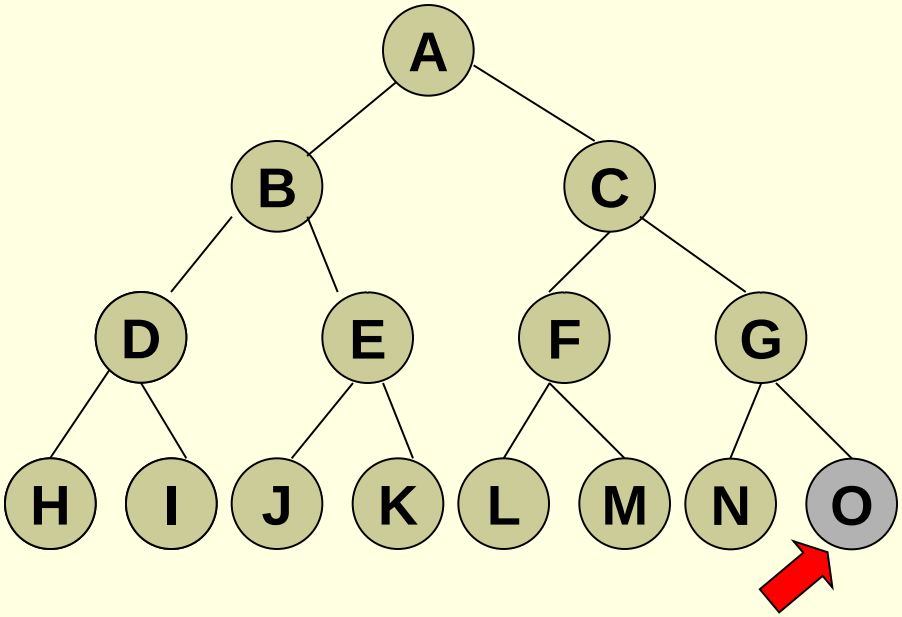
Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



Cautarea in adancime

- Se expandeaza nodul radacina, apoi se merge pe un drum pana se ajunge la cel mai adanc nivel al arborelui.
- Numai cand se ajunge la final (la nodurile frunza), cautarea se intoarce si expandeaza noduri de la nivele mai putin adanci.



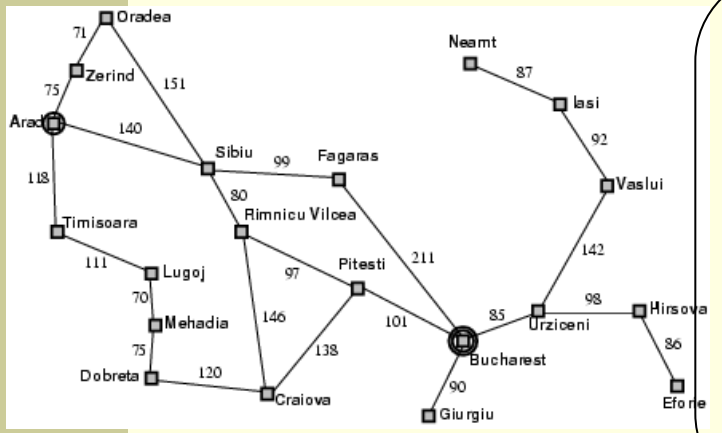
Parcurgerea in adancime: A, B, D, H, I, E, J, K, C, F, L, M, G, N, O.

Algoritm cautarea in adancime

functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
    intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
    intoarce nod
Altfel
  noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp

Algoritm cautare in adancime



Arad

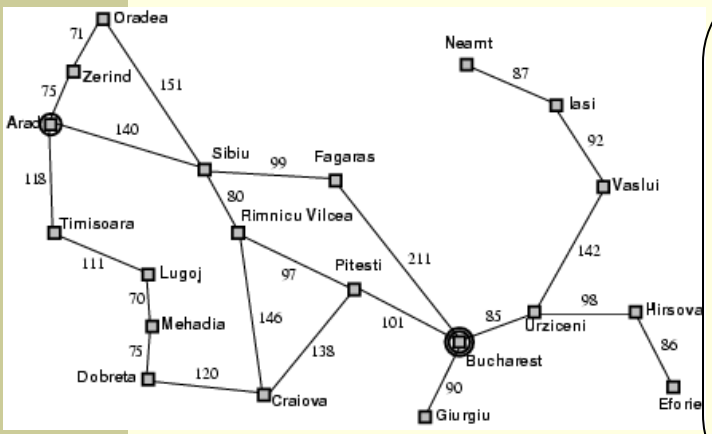
functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp

Faţa	**noduri**	Sfarsit
Arad		

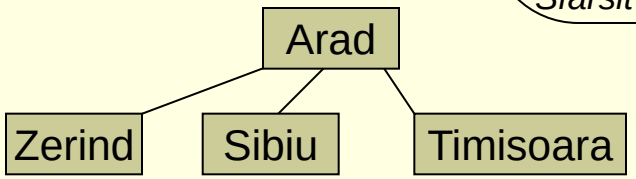
Parcurgerea: Arad,

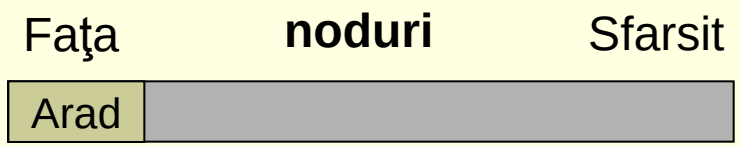
Algoritm cautare in adancime



functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

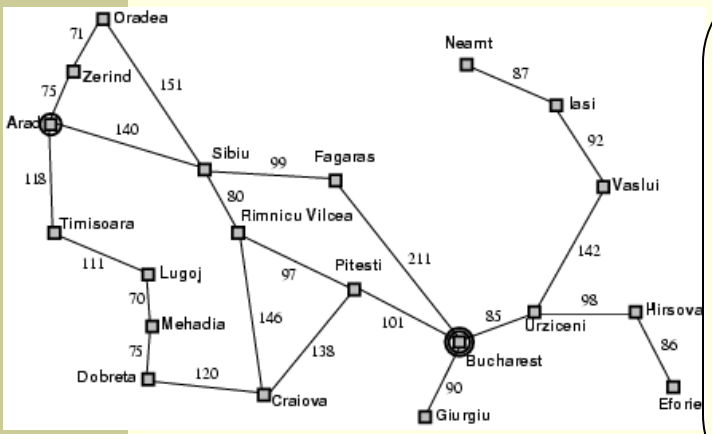
Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp





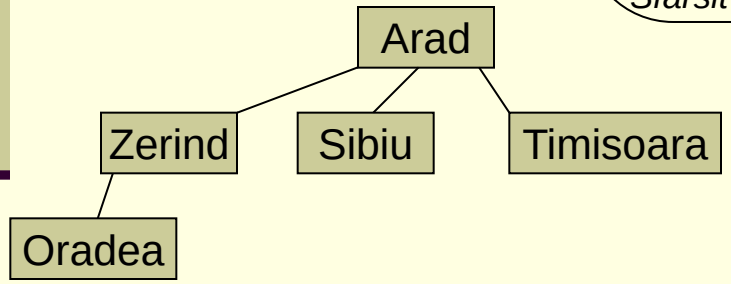
Parcurgerea: Arad,

Algoritm cautare in adancime



functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp

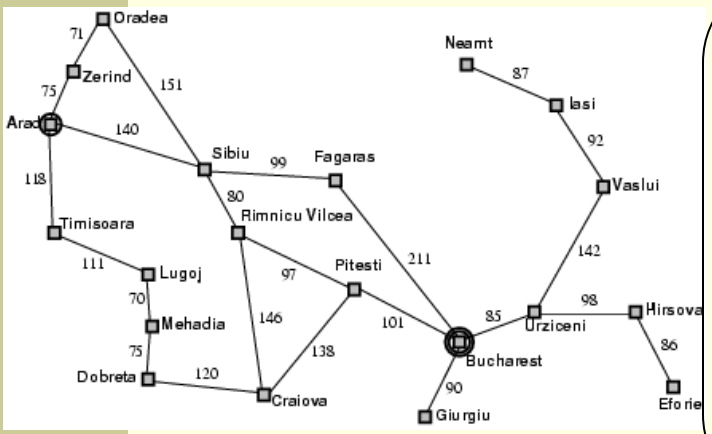


Faţa **noduri** Sfarsit

Zerind	Sibiu	Timisoara	

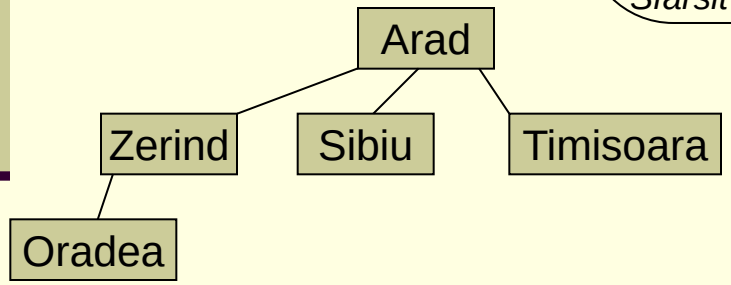
Parcurgerea: Arad, Zerind,

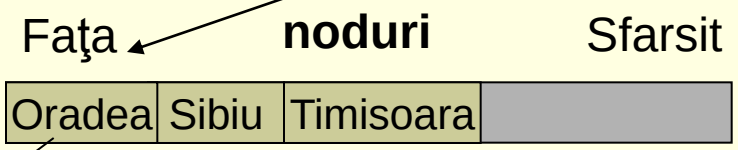
Algoritm cautare in adancime



functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
    Daca noduri = vida *atunci*
        intoarce esec
    nod = scoate_din_fata(noduri)
    Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
        intoarce nod
    Altfel
        noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp

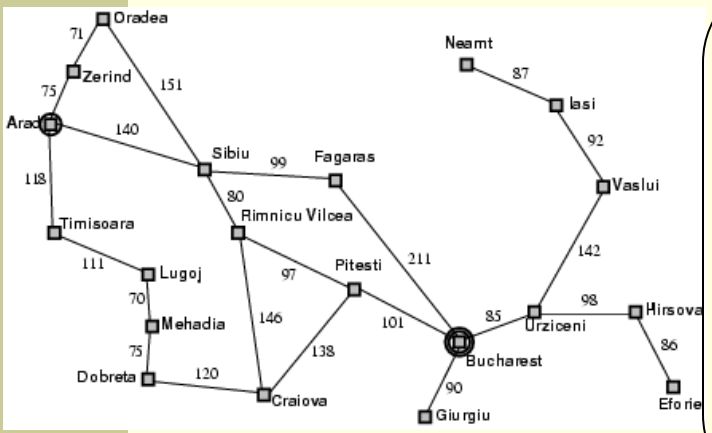




Parcurgerea: Arad, Zerind,

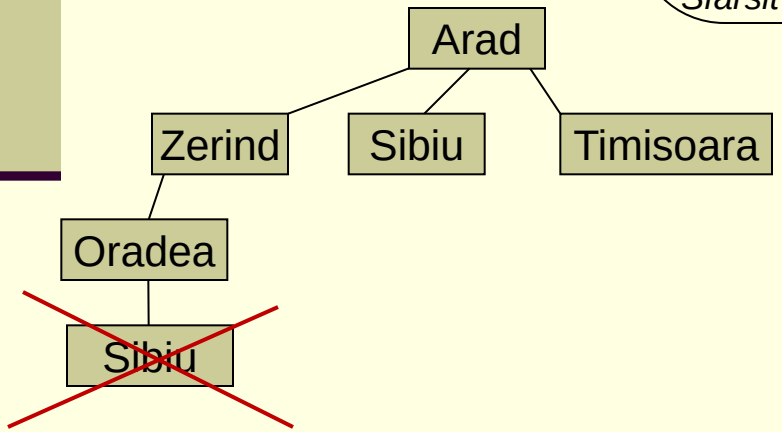
Adaugarea se face prin faţă!

Algoritm cautare in adancime



functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp

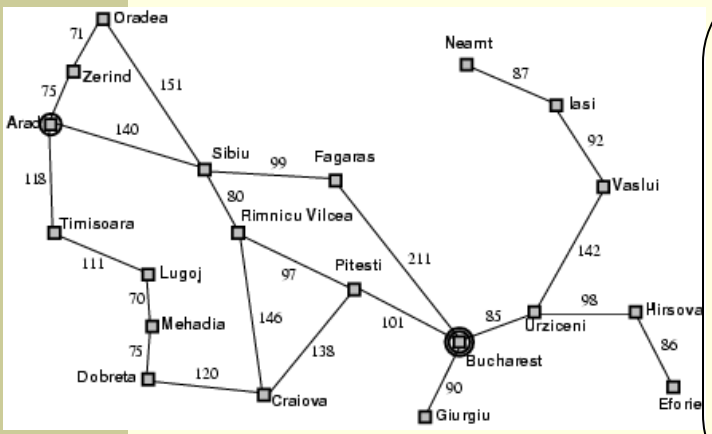


Faţa **noduri** Sfarsit

Oradea	Sibiu	Timisoara	

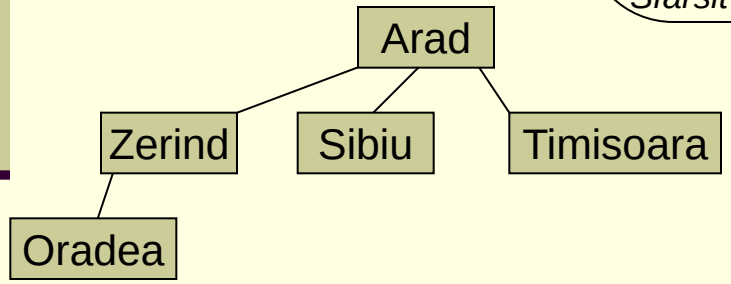
Parcurgerea: Arad, Zerind, Oradea

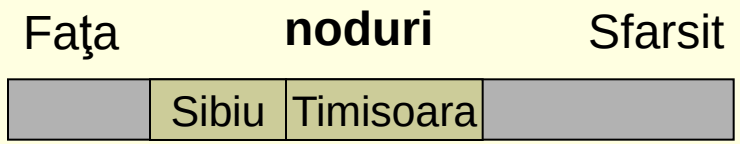
Algoritm cautare in adancime



functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

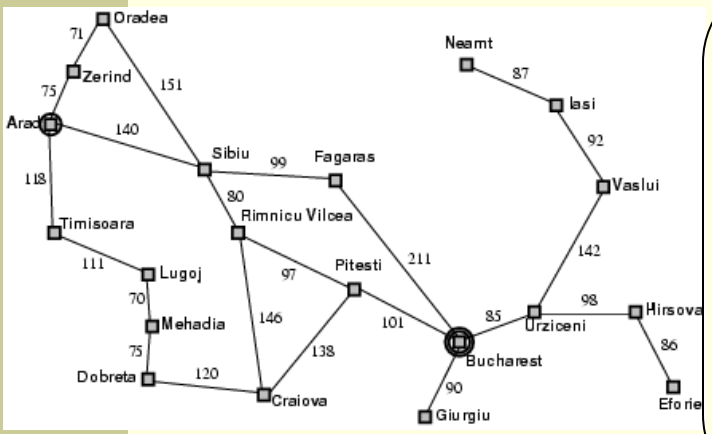
Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp





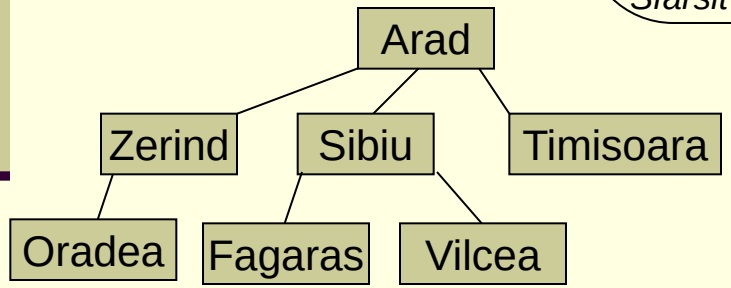
Parcurgerea: Arad, Zerind, Oradea, Sibiu

Algoritm cautare in adancime



functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp

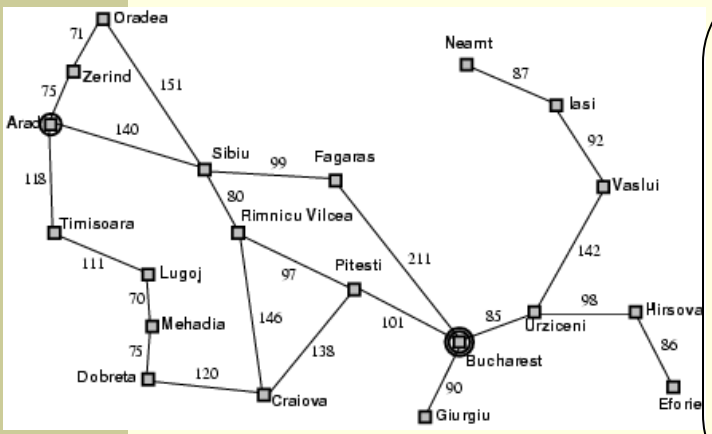


Faţa **noduri** Sfarsit

Fagaras	Vilcea	Timisoara	

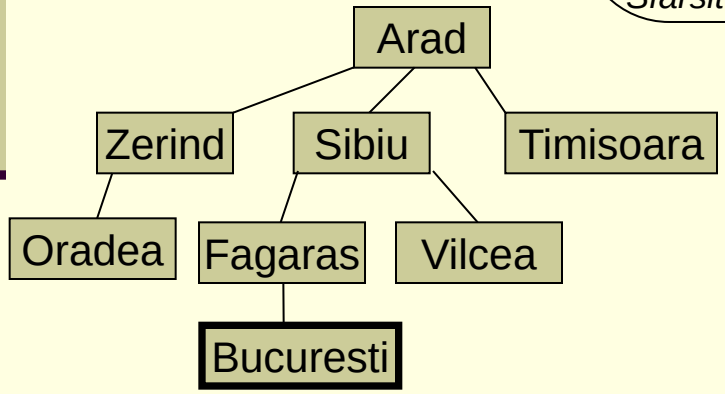
Parcurgerea: Arad, Zerind, Oradea, Sibiu

Algoritm cautare in adancime



functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat timp

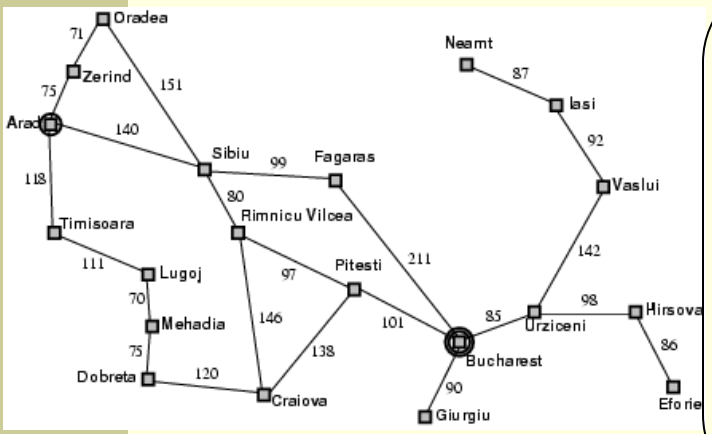


Faţa **noduri** Sfarsit

Fagaras	Vilcea	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu, Fagaras

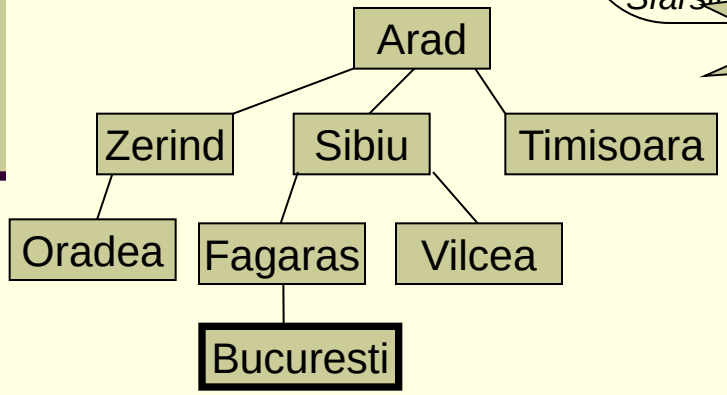
Algoritm cautare in adancime



functia cautare_adancime(*problema*) ***intoarce*** solutie sau esec
noduri = genereaza_coada(genereaza_nod(stare_initiala[problema]))

Cat timp este posibil *executa*
Daca noduri = vida *atunci*
intoarce esec
nod = scoate_din_fata(noduri)
Daca testare_tinta[problema] se aplica la stare(nod) *atunci*
intoarce nod
Altfel
noduri = adauga(noduri, expandare(nod, adauga_la_inceput))
Sfarsit cat

Nod tinta!

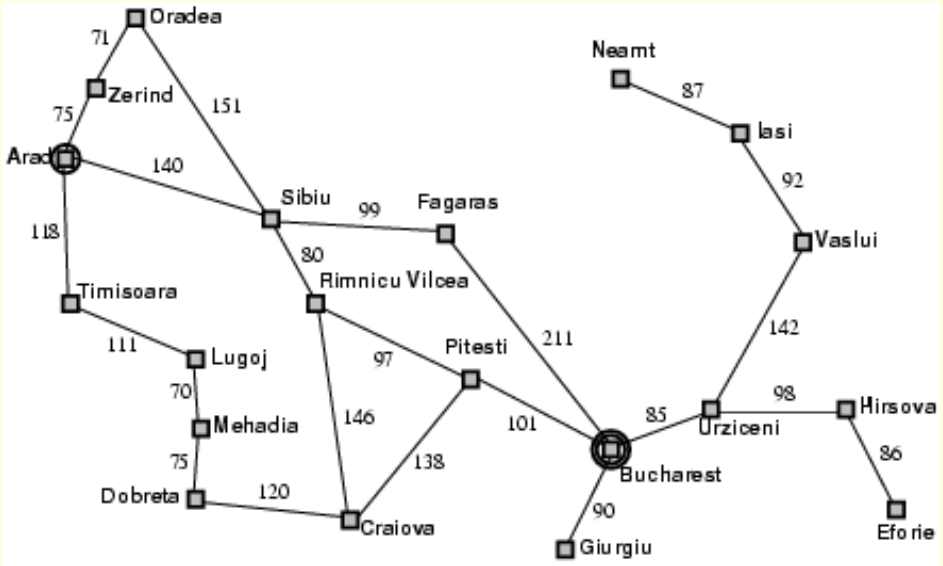


Faţa **noduri** Sfarsit

Bucuresti	Vilcea	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu, Fagaras, Bucuresti

Exercitiu



Faţa **noduri** Sfarsit

Gasiti o ruta de la Timisoara la Vaslui folosind parcurgerea in adancime. Desenati arborele, scrieti parcurgerea si continutul pentru noduri la fiecare pas.

Parcurgerea: Timisoara, …, Vaslui

Exercitiu - Problema celor 4 dame

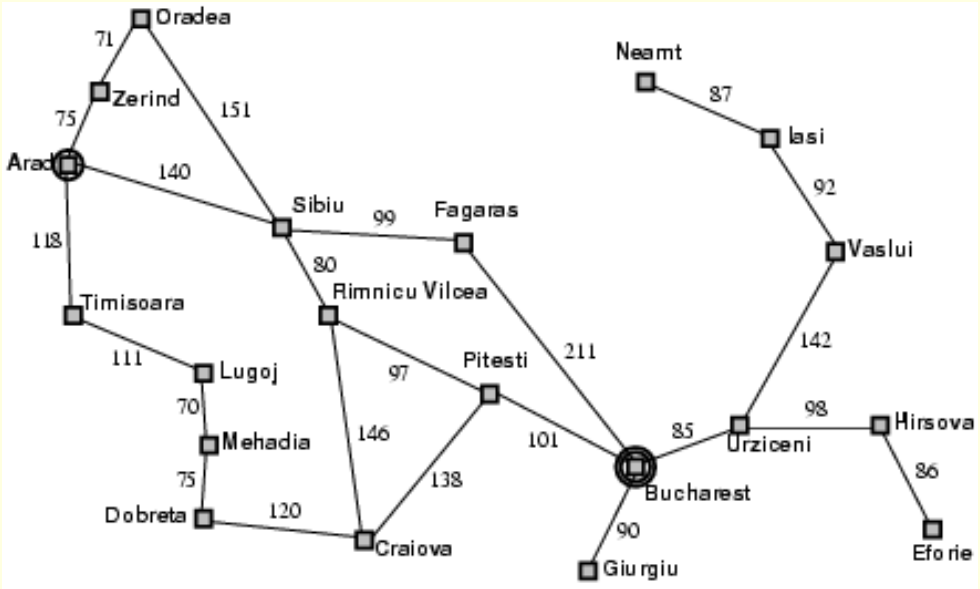
- **Stari**: orice aranjament de 0 pana la 4 dame care nu se ataca.
- **Actiuni**: adauga o dama pe coloana cea mai din stanga a.i. sa nu fie atacata de alta dama.
- **Testarea tintei**: 4 dame care nu se ataca pe tabla.
- **Costul drumului**: 0.

- Pornind de la o tabla de 4x4 goala si folosind datele problemei de mai sus, sa se construiasca printr-o cautare in adancime arborele complet care duce la rezolvarea problemei. Numerotati nodurile in ordinea in care au fost vizitate.

Cautarea limitata in adancime

- Impune o margine superioara pentru lungimea unui drum.
- Se poate utiliza la probleme unde stim la ce adancime maxima trebuie sa gasim solutia
  - Ex: avem 20 de orase, ne aflam in orasul A, solutia trebuie sa se gaseasca la maxim 19 pasi.
- Daca *l* este limita de adancime stabilita, atunci complexitatile:
  - Pentru timp: $O(b^l)$
  - Pentru spatiu: $O(bl)$.

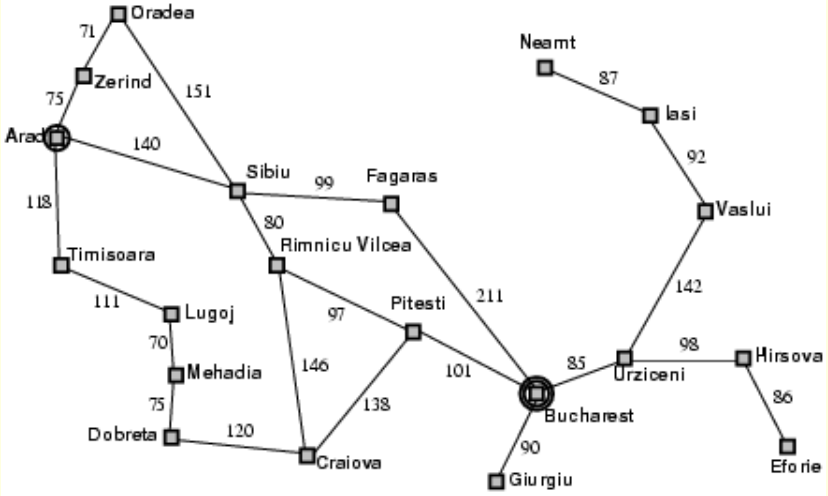
Cautarea limitata in adancime



Stabilim limita de adancime egala cu 3.

Sa se gaseasca o ruta de la Arad la Bucuresti folosind cautarea limitata in adancime.

Cautarea limitata in adancime



Arad

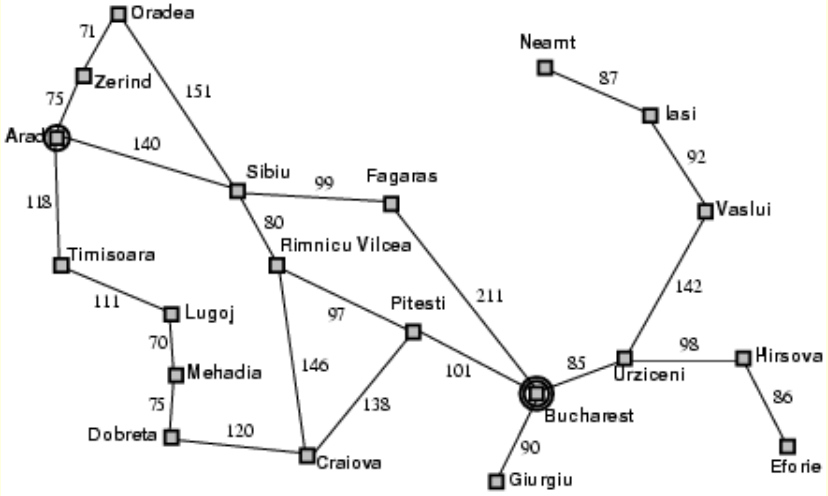
Faţa **noduri** Sfarsit

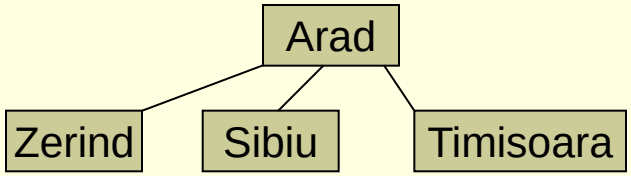
Arad

Parcurgerea: Arad,

Adancime: 0

Cautarea limitata in adancime





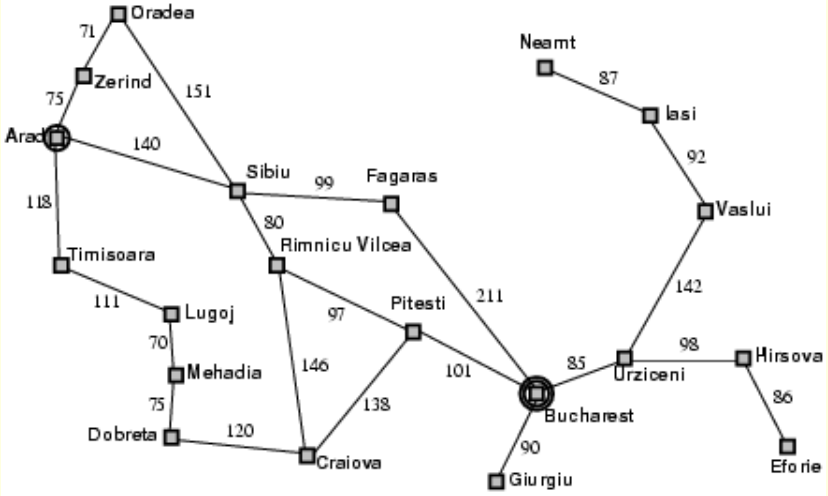
Faţa **noduri** Sfarsit

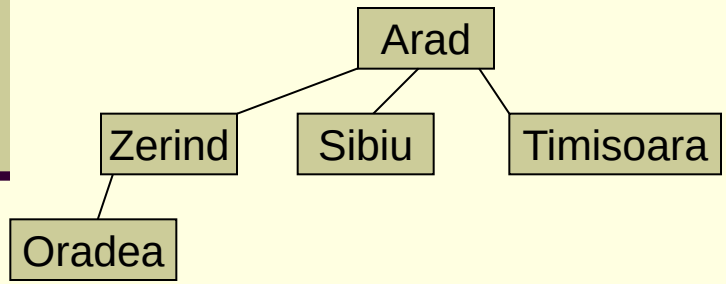
Arad

Parcurgerea: Arad,

Adancime: 1

Cautarea limitata in adancime





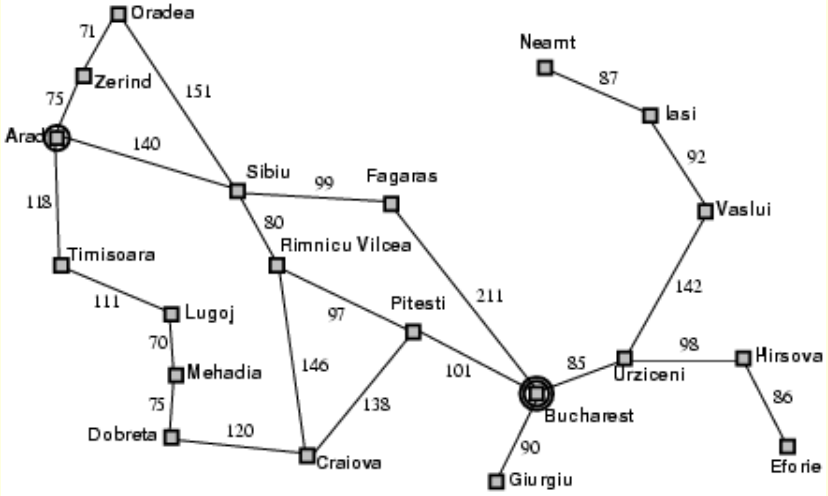
Faţa **noduri** Sfarsit

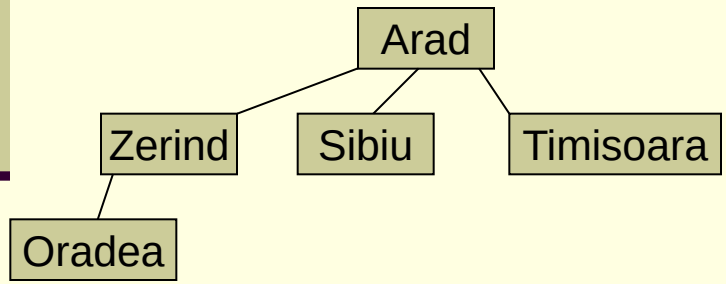
Zerind	Sibiu	Timisoara	

Parcurgerea: Arad, Zerind,

Adancime: 1

Cautarea limitata in adancime





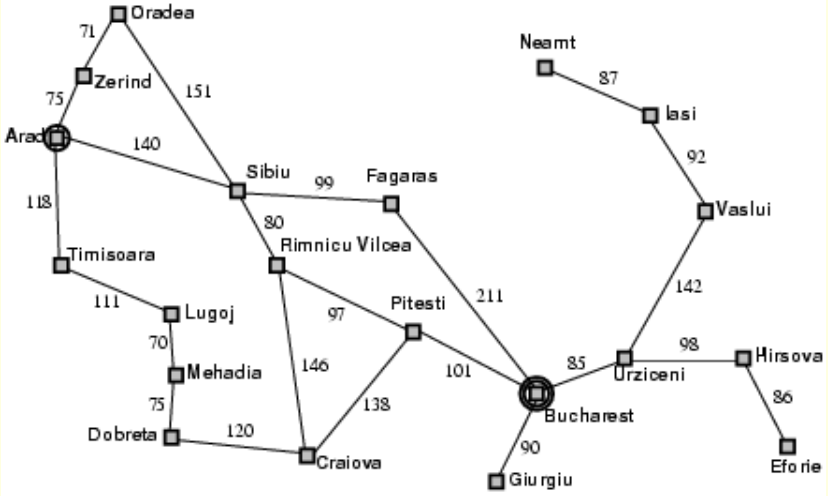
Faţa **noduri** Sfarsit

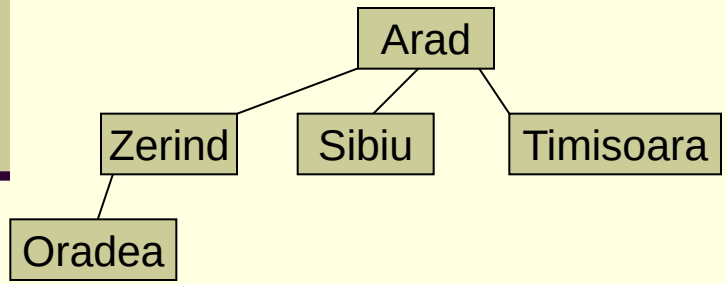
Oradea	Sibiu	Timisoara	

Parcurgerea: Arad, Zerind, Oradea

Adancime: 2

Cautarea limitata in adancime





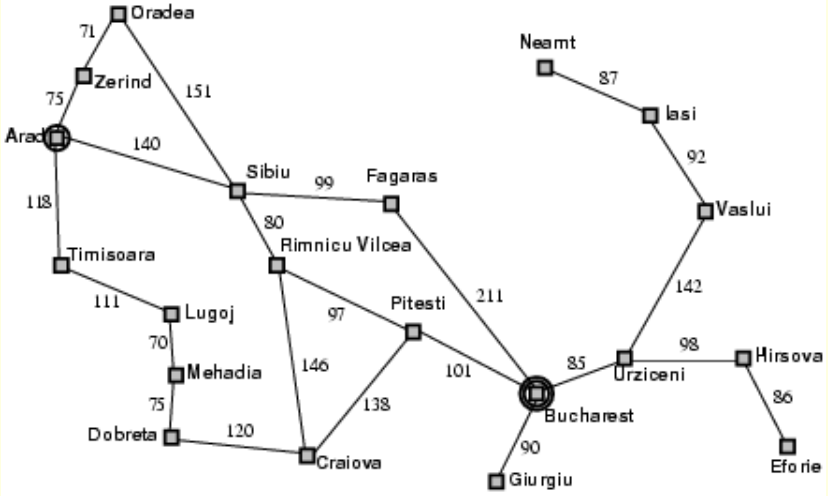
Faţa **noduri** Sfarsit

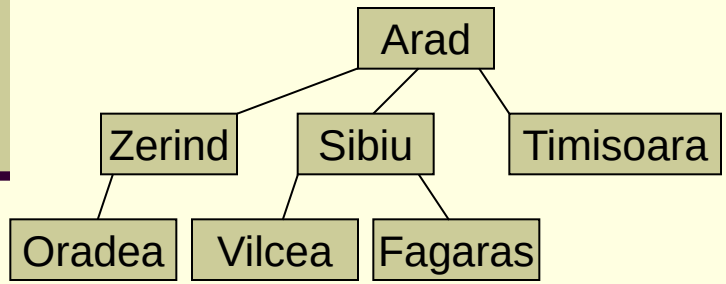
Sibiu	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu

Adancime: 1

Cautarea limitata in adancime





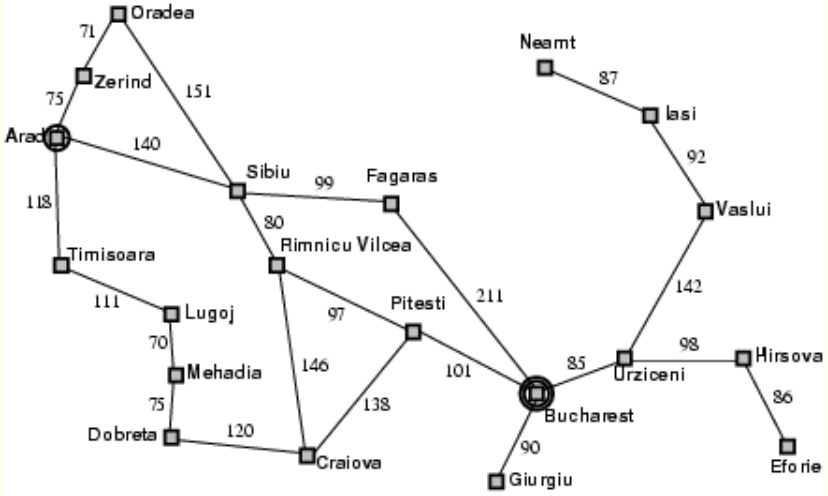
Faţa **noduri** Sfarsit

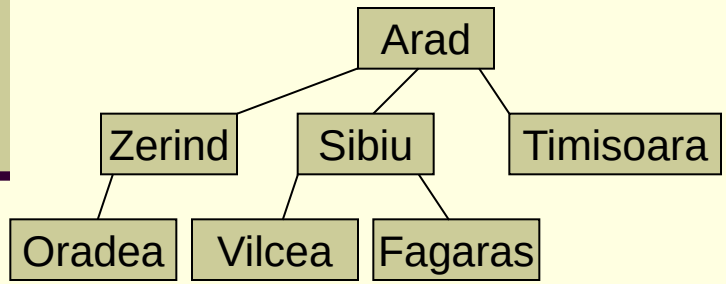
Sibiu	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu

Adancime: 1

Cautarea limitata in adancime





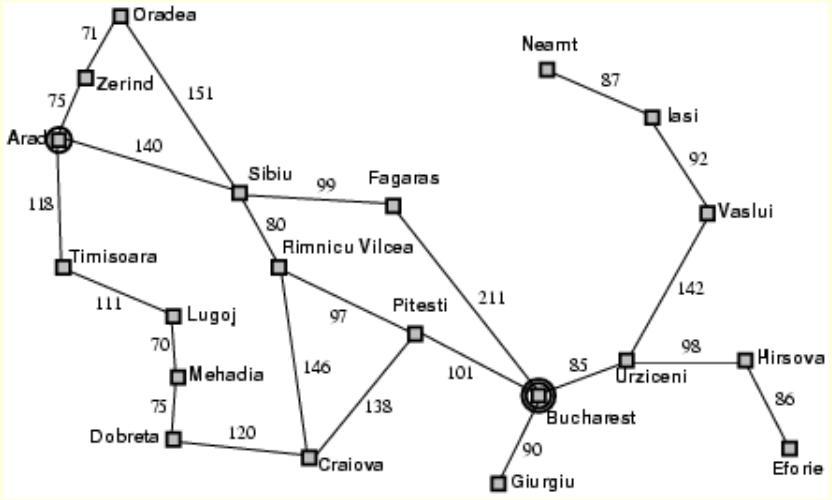
Faţa **noduri** Sfarsit

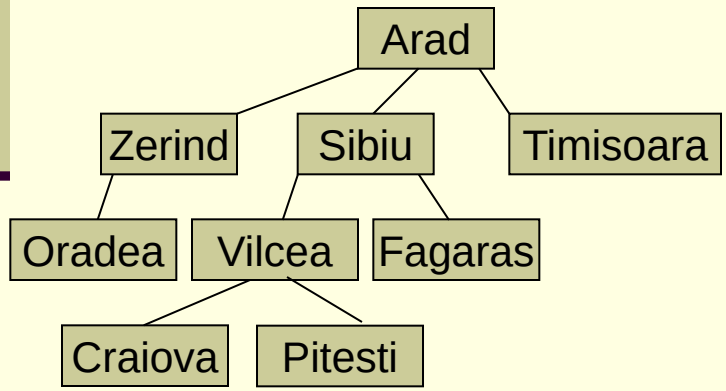
Vilcea	Fagaras	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu, Vilcea

Adancime: 2

Cautarea limitata in adancime





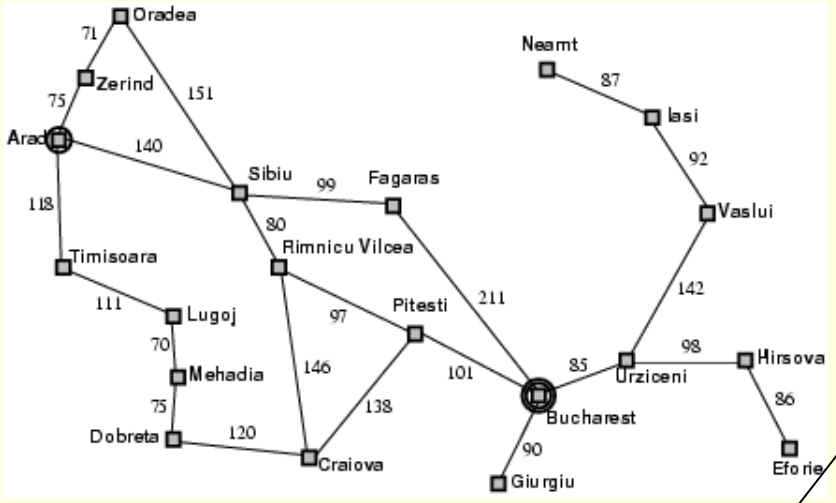
Faţa **noduri** Sfarsit

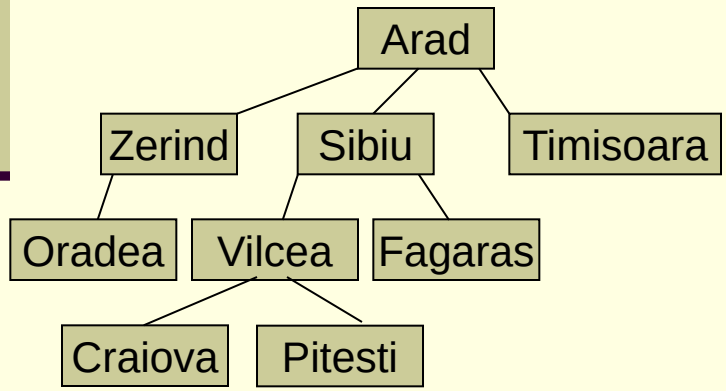
Vilcea	Fagaras	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu, Vilcea

Adancime: 3

Cautarea limitata in adancime





Nu este nodul tinta!!

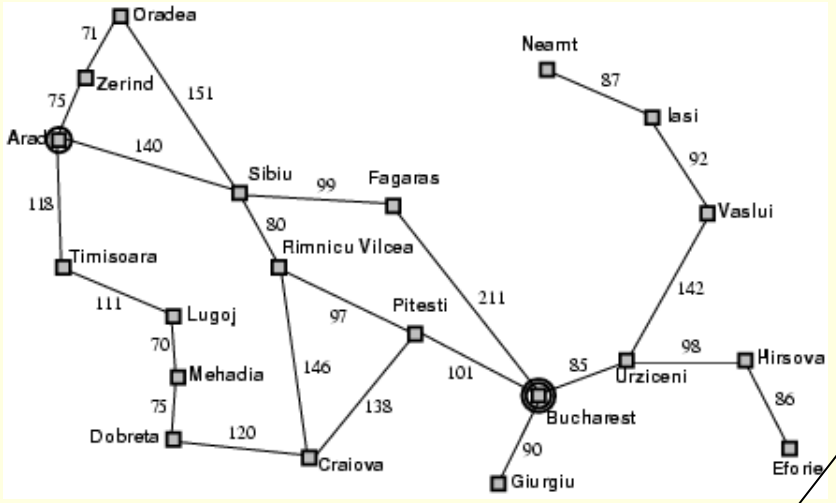
Faţa **noduri** Sfarsit

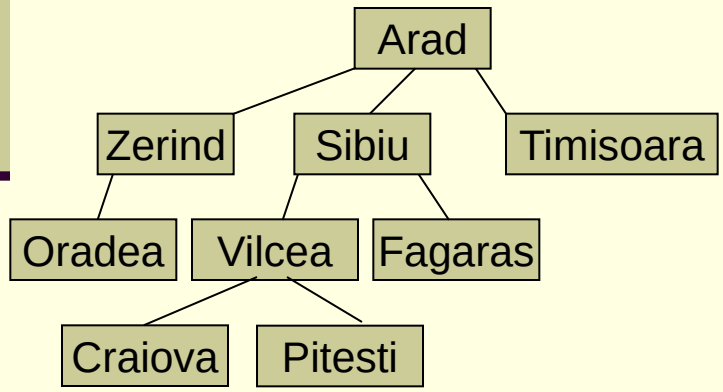
Craiova	Pitesti	Fagaras	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu, Vilcea, Craiova

Adancime: 3

Cautarea limitata in adancime





Nu este nodul tinta!!

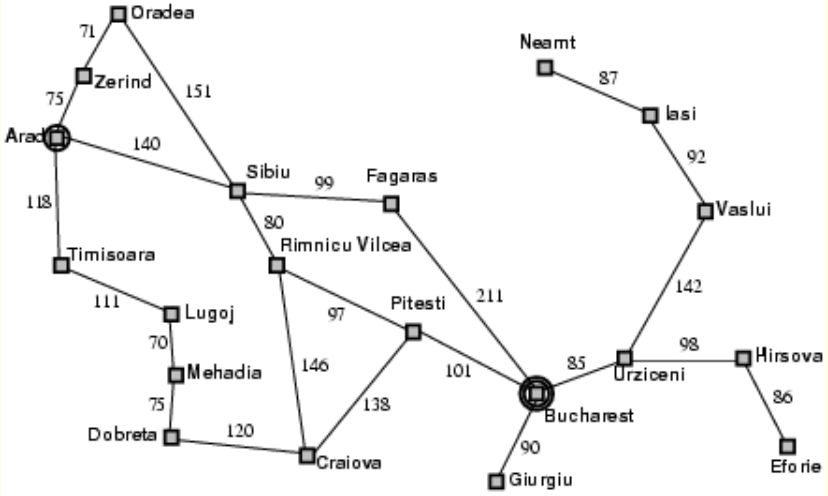
Faţa **noduri** Sfarsit

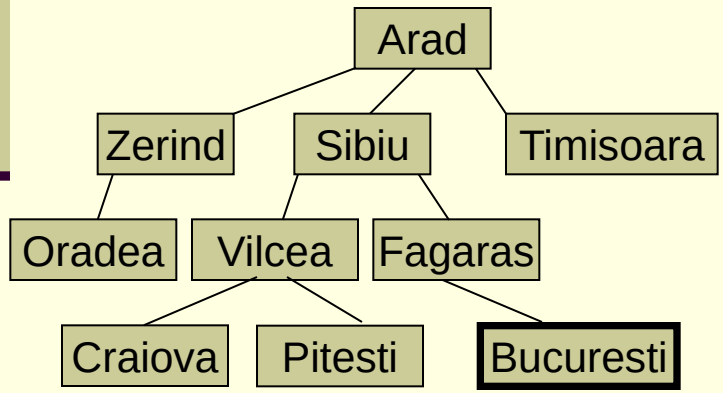
Pitesti	Fagaras	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu, Vilcea, Craiova, Pitesti

Adancime: 3

Cautarea limitata in adancime





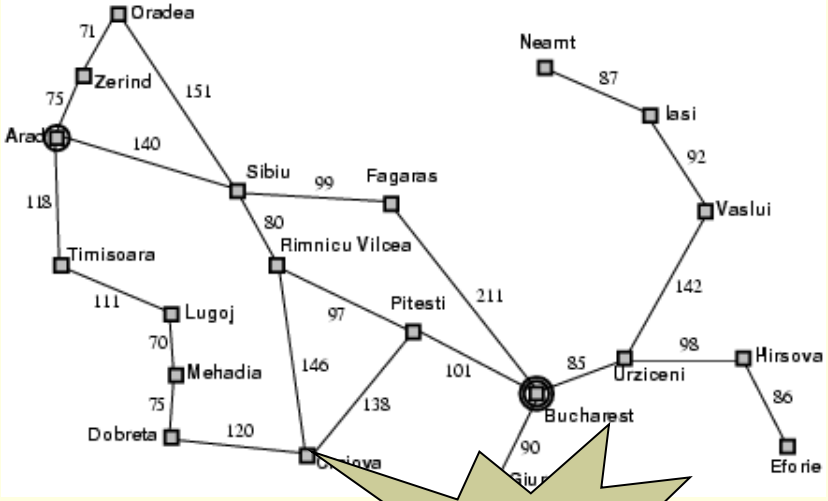
Faţa **noduri** Sfarsit

Fagaras	Timisoara	

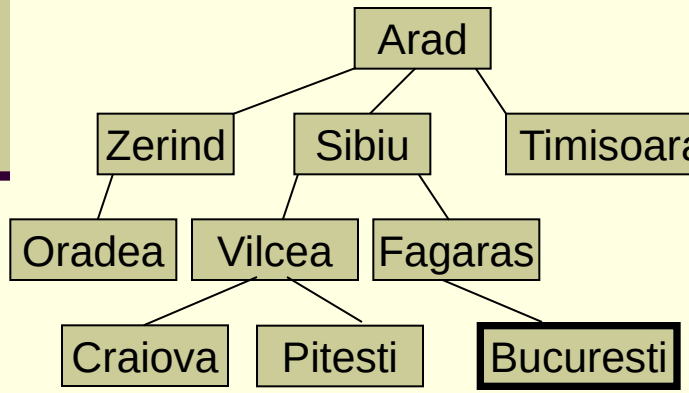
Parcurgerea: Arad, Zerind, Oradea, Sibiu, Vilcea, Craiova, Pitesti, Fagaras

Adancime: 2

Cautarea limitata in adancime



Nod tinta!



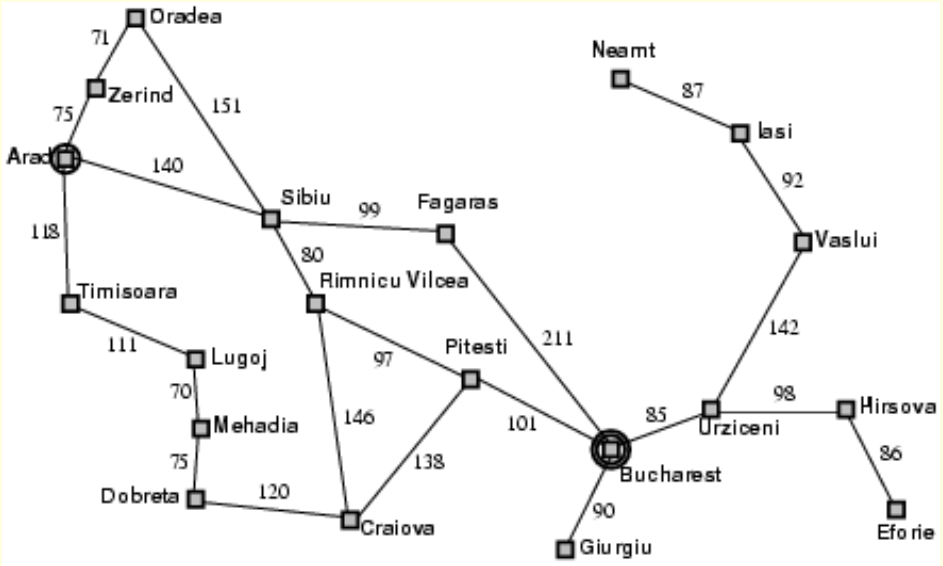
Faţa **noduri** Sfarsit

Bucuresti	Timisoara	

Parcurgerea: Arad, Zerind, Oradea, Sibiu, Vilcea, Craiova, Pitesti, Fagaras, Bucuresti.

Adancime: 3

Exercitiu



Faţa **noduri** Sfarsit

Gasiti o ruta de la Bucuresti la Rimnicu Vilcea folosind parcurgerea limitata in adancime cu limita 3.
Desenati arborele, scrieti parcurgerea si continutul pentru noduri la fiecare pas.

Parcurgerea: Bucuresti, …, Rm. Vilcea

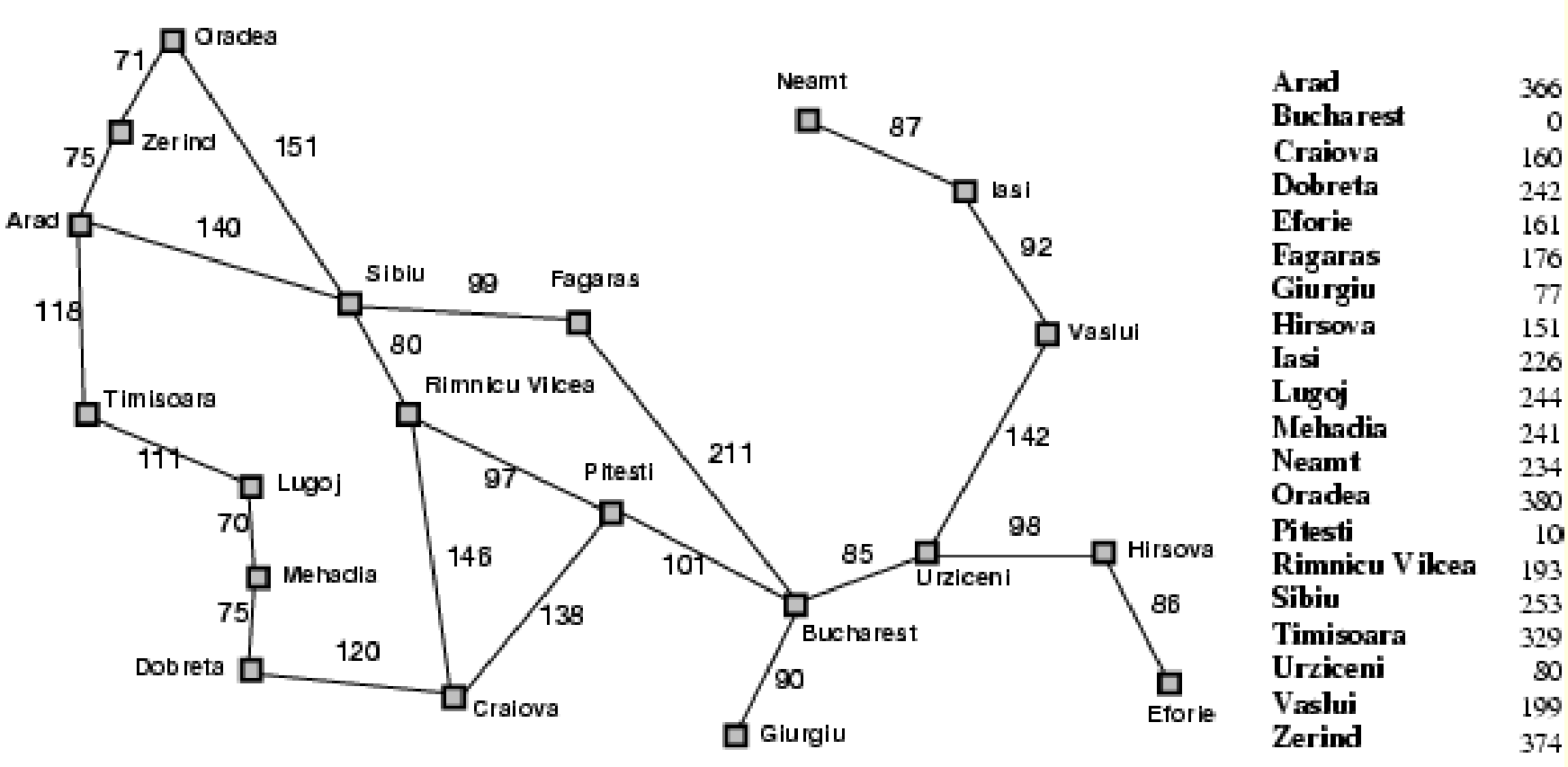
Cautarea Greedy

- 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

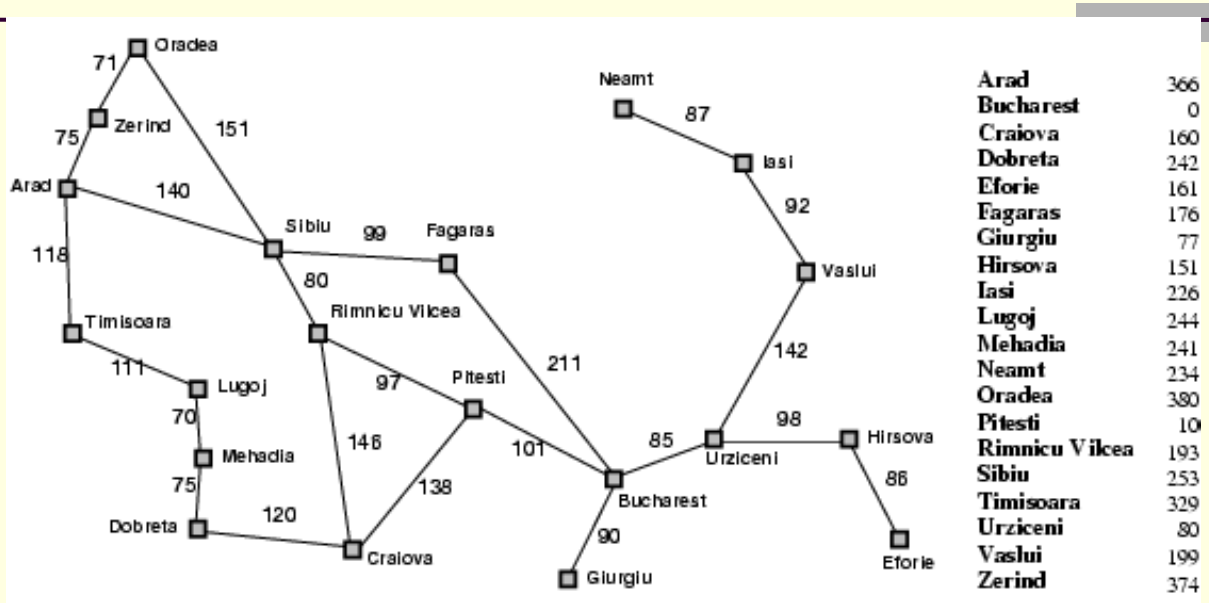
- ***h*(*n*)** = costul estimat pentru cel mai scurt drum de la nodul *n* pana la starea tinta.
  - Daca *n* este chiar nodul tinta, atunci *h*(*n*) = 0.
- O cautare *intai cel mai bun* care utilizeaza asemenea functie euristica se numeste **cautare greedy**.

Avem distantele pana la Bucuresti



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

Greedy



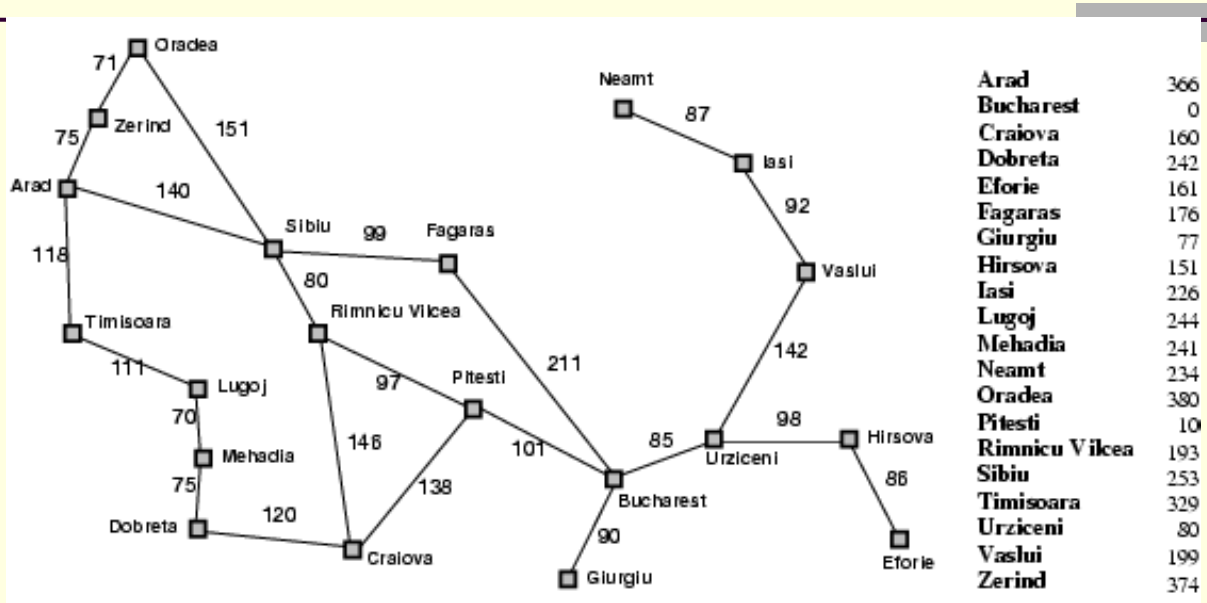
366 Arad

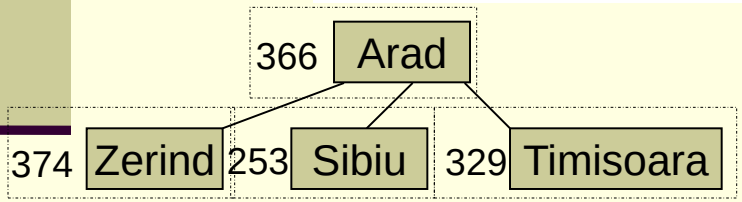
Faţa **noduri** Sfarsit

Arad

Parcurgerea: Arad,

Greedy





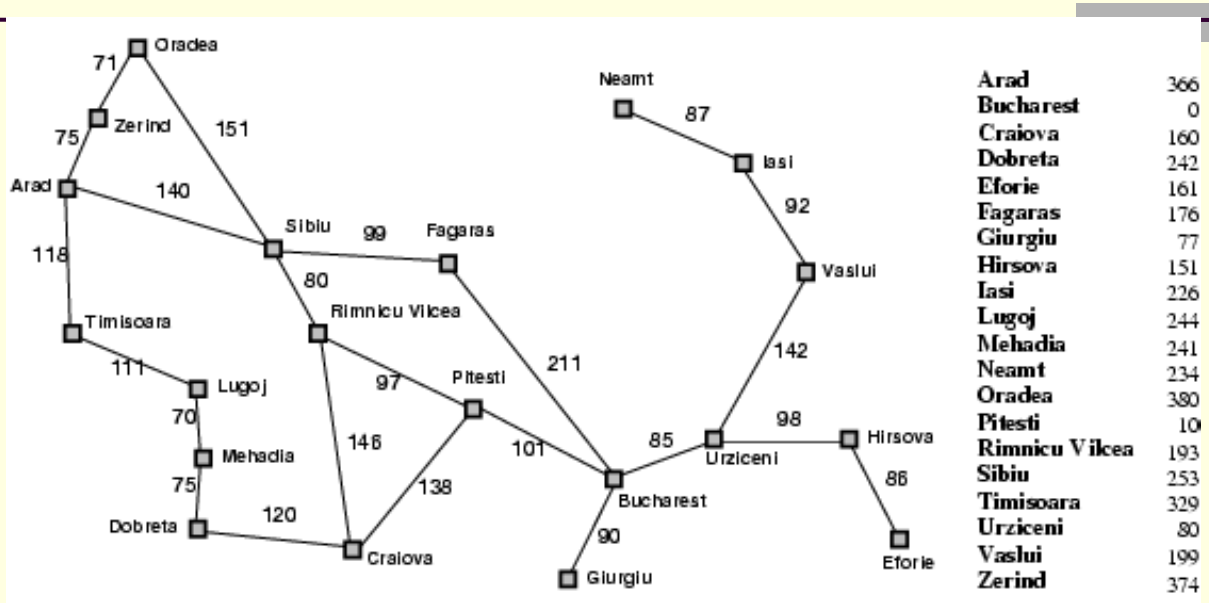
Faţa **noduri** Sfarsit

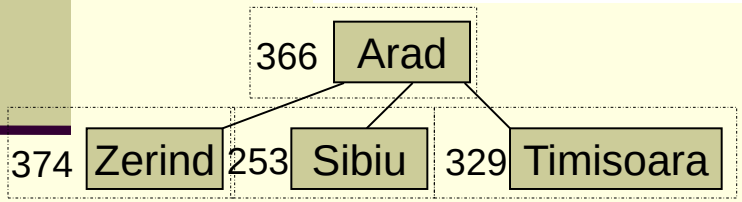
Arad

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

Parcurgerea: Arad,

Greedy



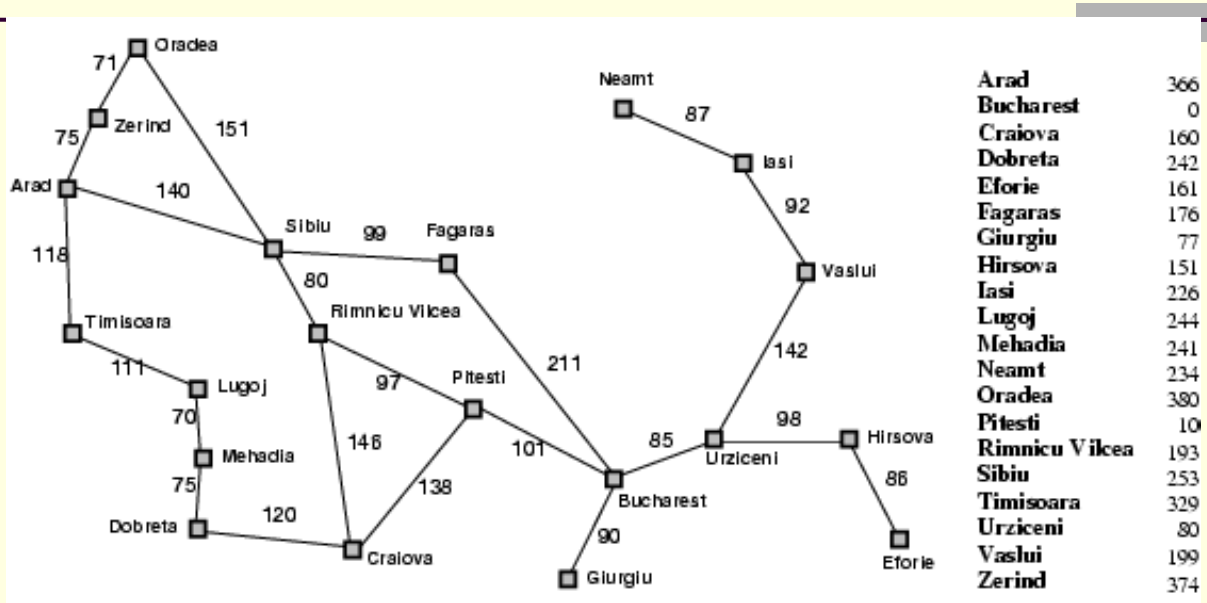


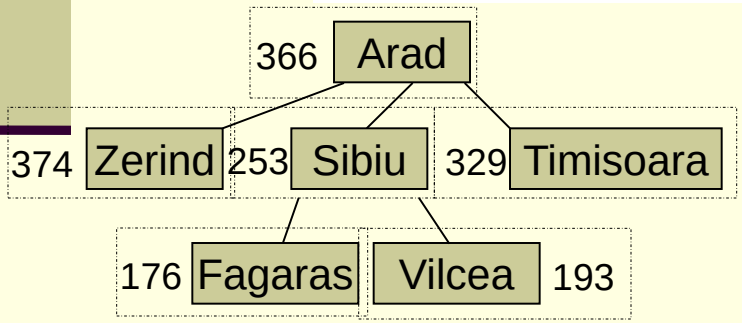
Faţa **noduri** Sfarsit

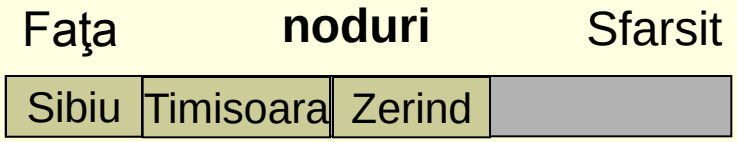
Sibiu	Timisoara	Zerind	

Parcurgerea: Arad,

Greedy

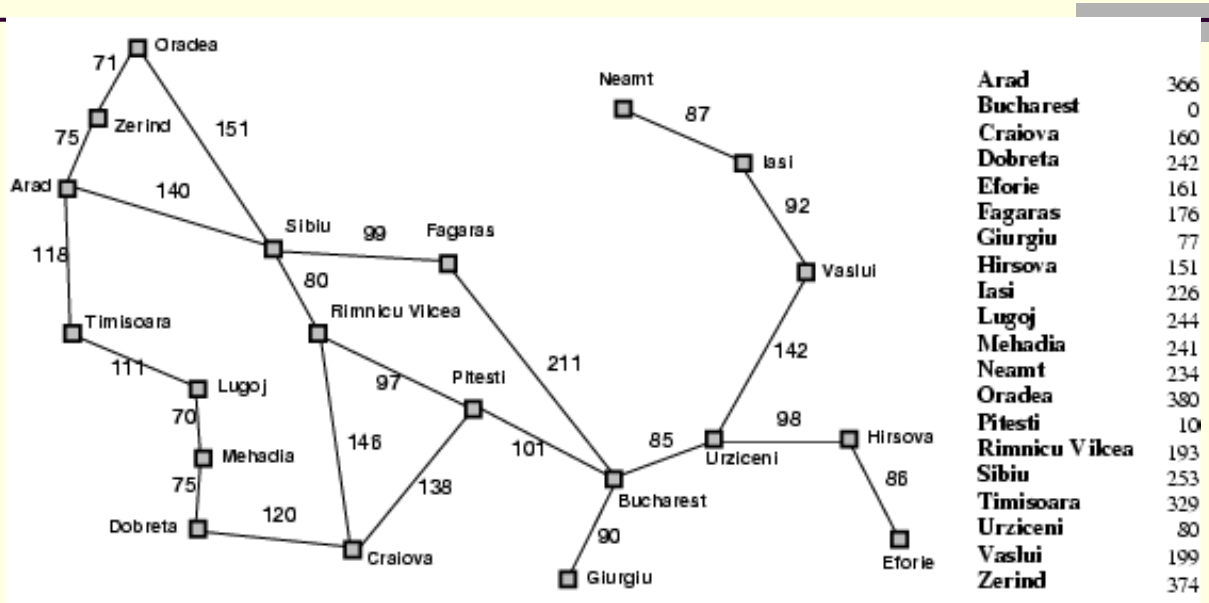


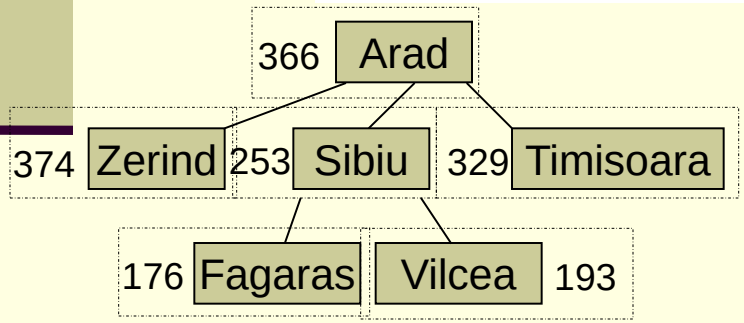




Parcurgerea: Arad, Sibiu

Greedy



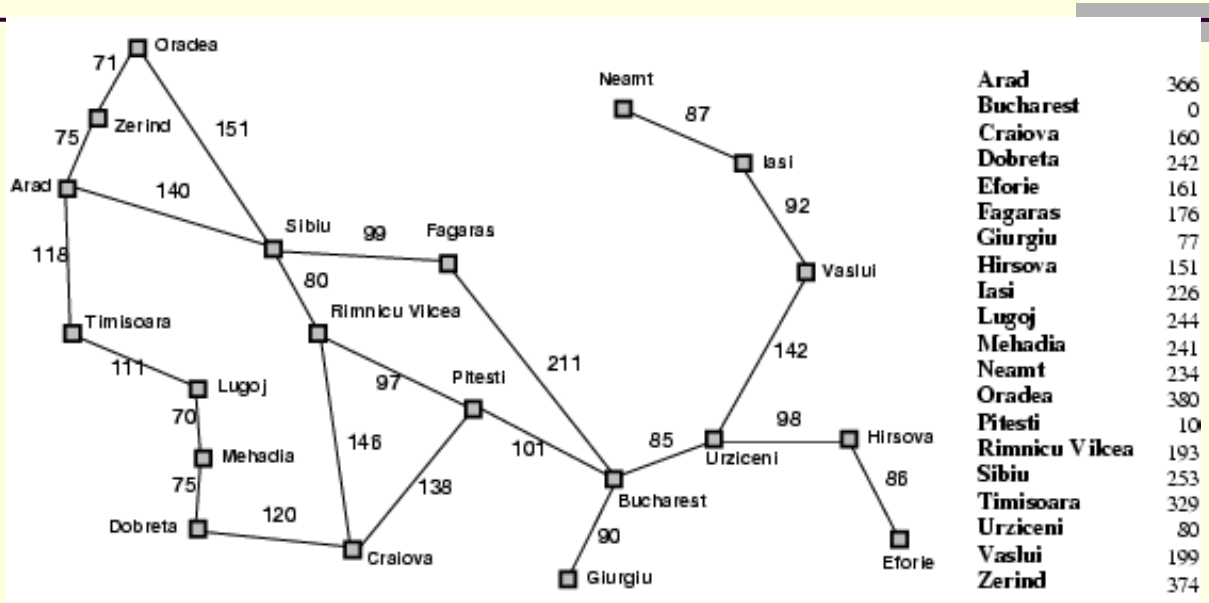


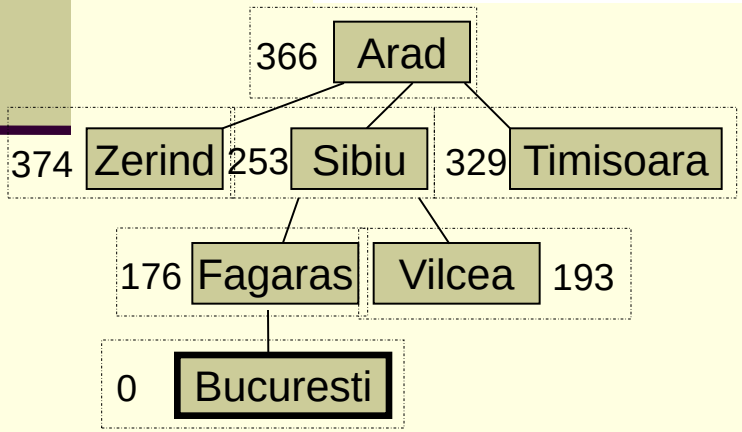
Faţa **noduri** Sfarsit

Fagaras	Vilcea	Timisoara	Zerind	

Parcurgerea: Arad, Sibiu

Greedy



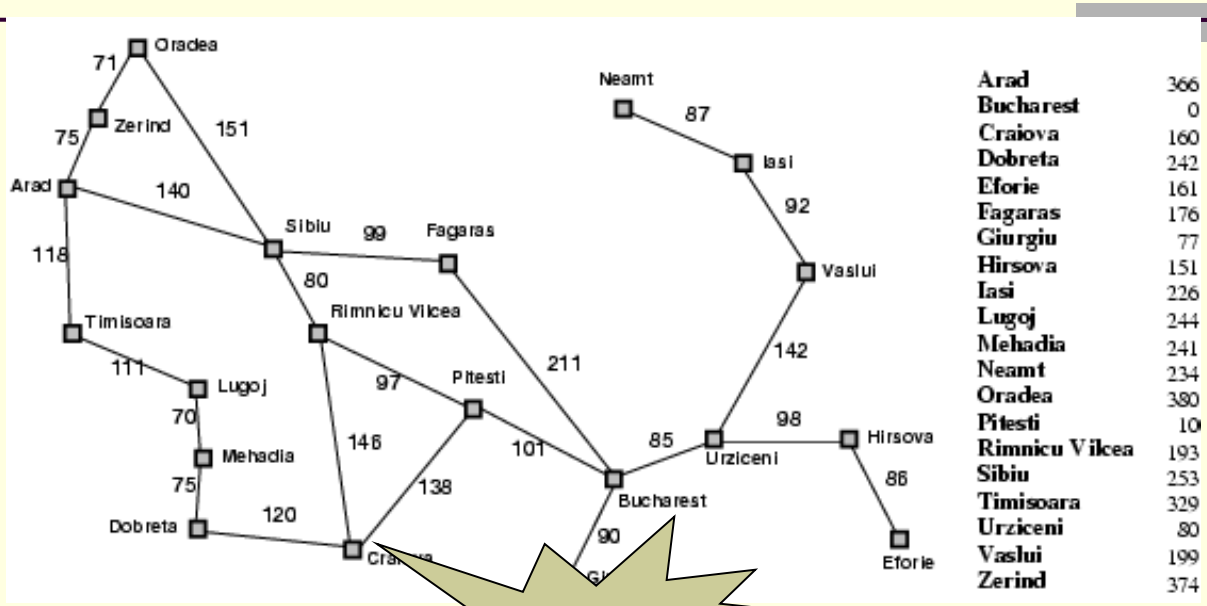


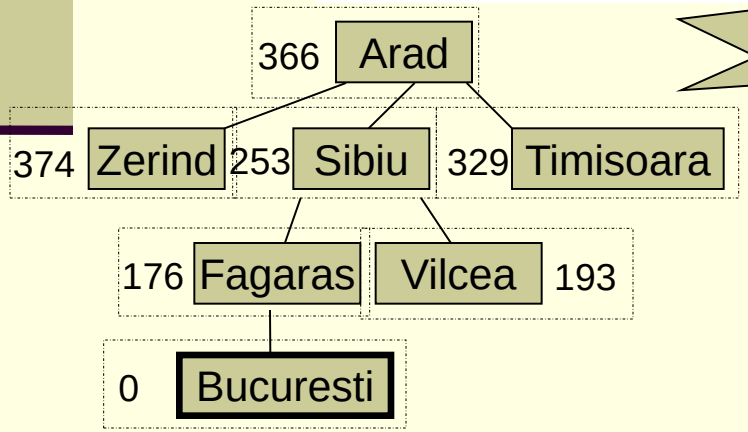
Faţa **noduri** Sfarsit

Fagaras	Vilcea	Timisoara	Zerind	

Parcurgerea: Arad, Sibiu, Fagaras

Greedy





Nod tinta!

Faţă **noduri** Sfarsit

Bucuresti	Vilcea	Timisoara	Zerind	

Parcurgerea: Arad, Sibiu, Fagaras, Bucuresti.

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

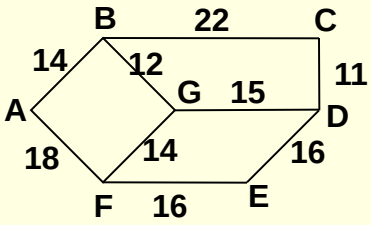
	2	3
1	**4**	6
8	**7**	**5**

= 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.

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*.

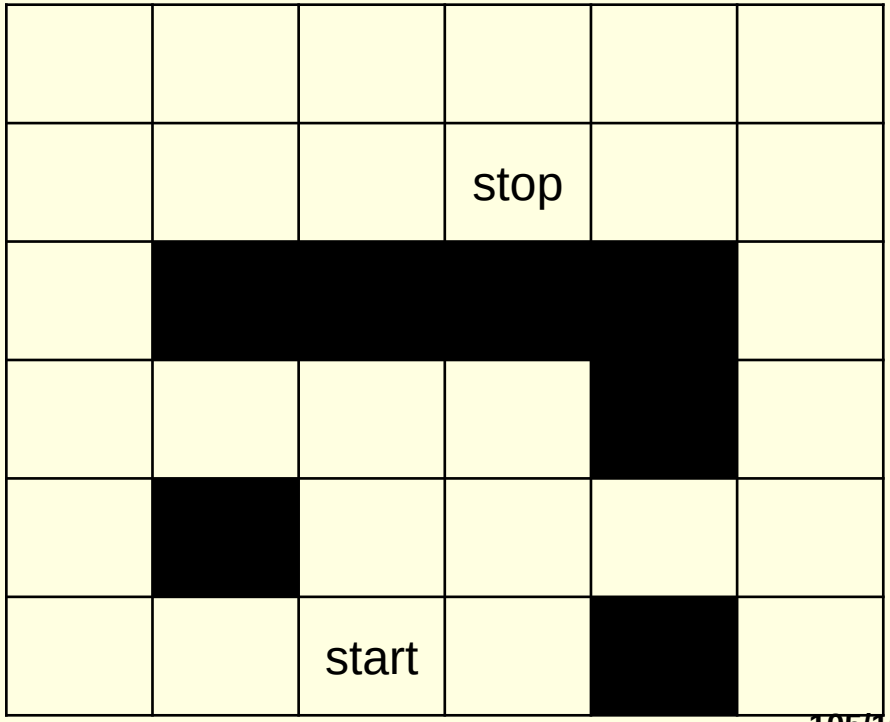
Exercitiu

- Consideram 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.

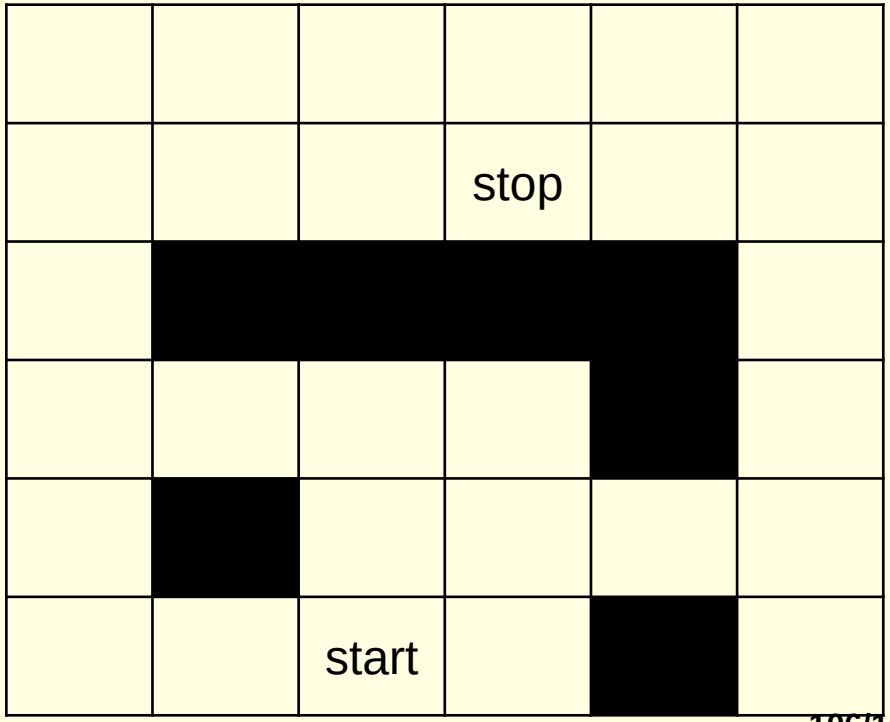


Nu se tine cont de patratele hasurate pentru distanta Manhattan

.

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?