UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Workbook:
Depth-first search (tree search)

Albert Sanchis
Jorge Civera

DSIC

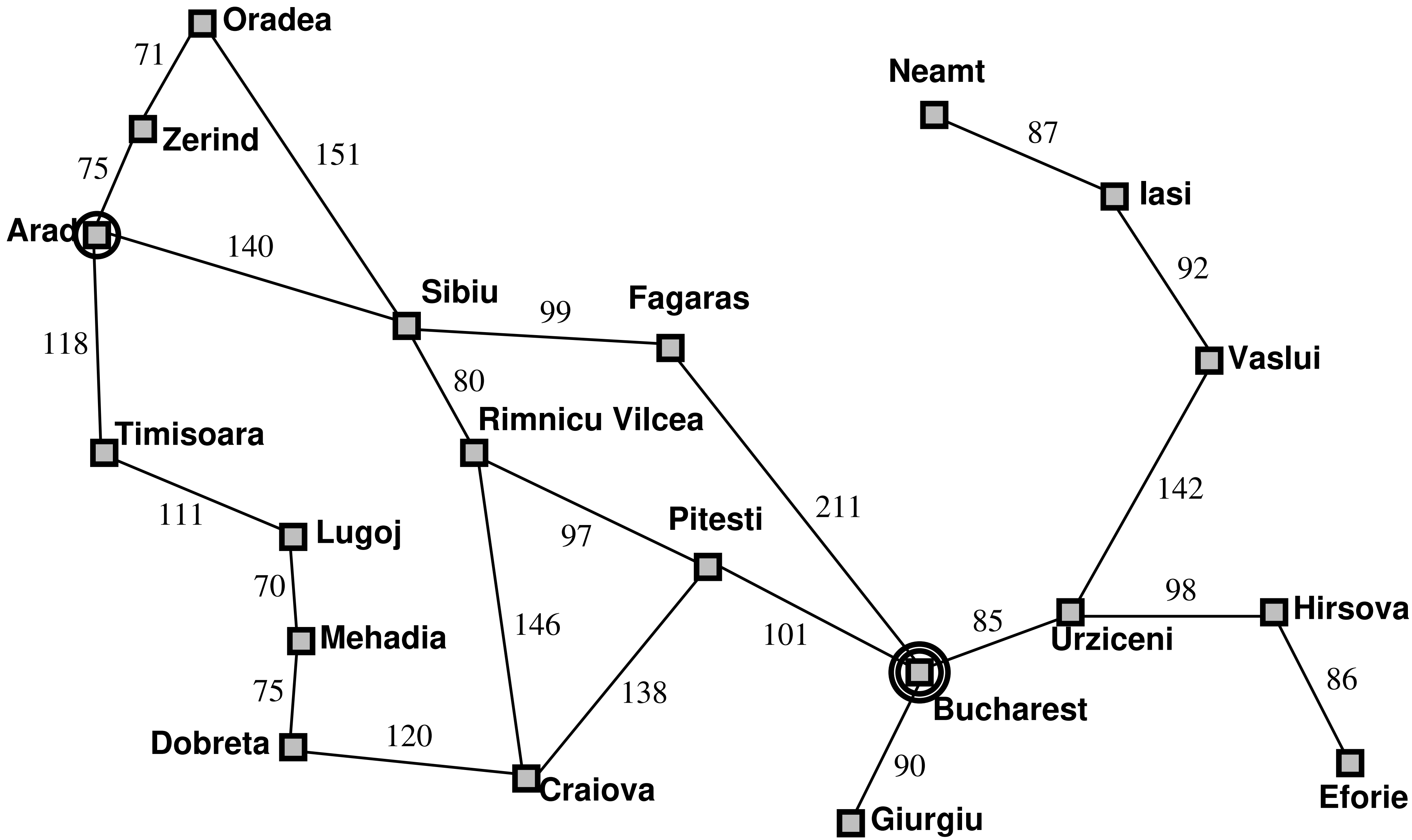
Departament de Sistemes Informàtics i Computació

Learning objectives

- To describe depth-first search (tree search).
- To draw the tree of depth-first search.
- To apply depth-first search (tree search) to a well-known problem.
- To analyze the quality of depth-first search (tree search).

Problem: Shortest path between two points

Shortest path from Arad to Bucarest [1]:



Actions(Arad) = {Move(Sibiu), Move(Timisoara), Move(Zerind)}.

1 Depth-first search [1, 2]

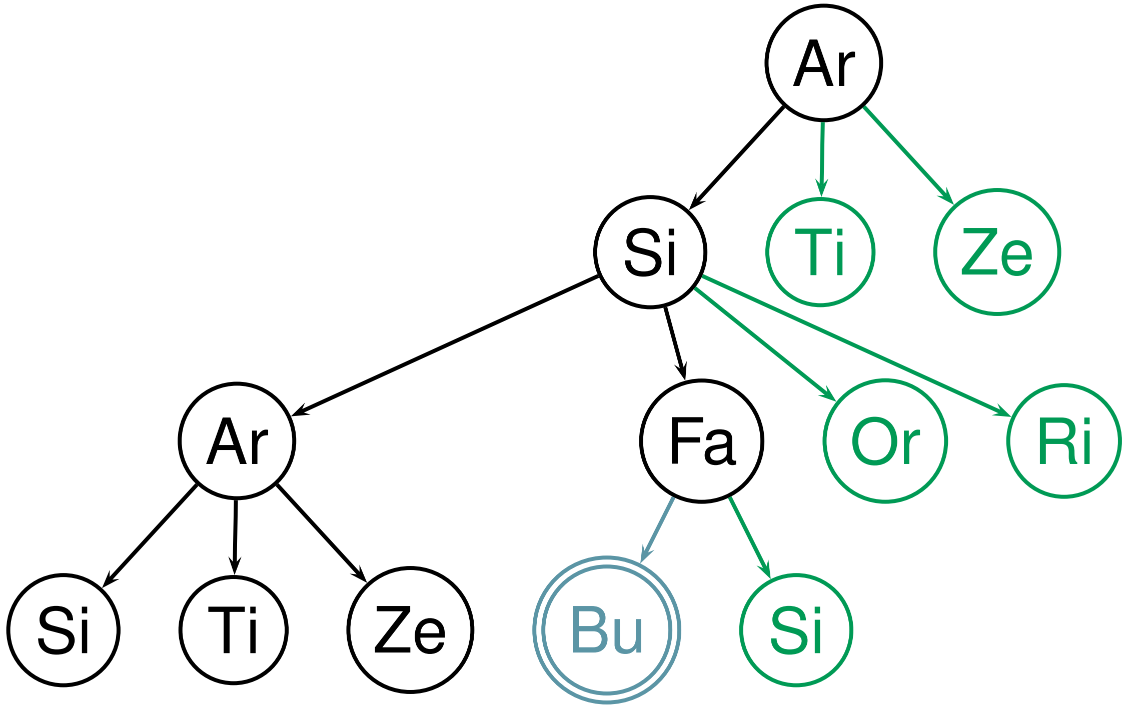
```
DFS(G, s', m)                    // Depth-first search with maximum depth of m
  O = InitStack(s')                              // Open: search frontier-stack
  while not EmptyStack(O):
    s = Pop(O)                         // selection LIFO (Last in, first out)
    if Goal(s) return s                                     // solution found!
    if Depth(s) < m:                             // maximum depth not reached
      forall (s, n) ∈ Adjacents(G, s):           // generation: n child of s
        Push(O, n)                                   // n added to the stack
  return NULL                                             // no solution found
```


UNIVERSITAT POLITÈCNICA DE VALÈNCIA

- ***Question 1***: Write a trace of the ***DFS*** algorithm (tree search) applied to the problem of finding the shortest path from Arad to Bucarest with maximum depth $m = 3$.

O	s
{Arad (d=0)}	–
{Sibiu (d=1), Timisoara (d=1), Zerind (d=1)}	Arad (d=0)
{Arad (d=2), Fagaras (d=2), Oradea (d=2), Rimnicu(d=2), Timisoara (d=1), Zerind (d=1)}	Sibiu (d=1)
{Sibiu (d=3), Timisoara (d=3), Zerind (d=3), Fagaras (d=2), Oradea (d=2), Rimnicu(d=2), Timisoara (d=1), Zerind (d=1)}	Arad (d=2)
{Timisoara (d=3), Zerind (d=3), Fagaras (d=2), Oradea (d=2), Rimnicu(d=2), Timisoara (d=1), Zerind (d=1)}	Sibiu (d=3)
{Zerind (d=3), Fagaras (d=2), Oradea (d=2), Rimnicu(d=2), Timisoara (d=1), Zerind (d=1)}	Timisoara (d=3)
{Fagaras (d=2), Oradea (d=2), Rimnicu(d=2), Timisoara (d=1), Zerind (d=1)}	Zerind (d=3)
{Bucharest (d=3), Sibiu (d=3), Oradea (d=2), Rimnicu(d=2), Timisoara (d=1), Zerind (d=1)}	Fagaras (d=2)
{Sibiu (d=3), Oradea (d=2), Rimnicu(d=2), Timisoara (d=1), Zerind (d=1)}	Bucharest (d=3)

- ***Question 2***: Draw the search tree as a result of applying the ***DFS*** algorithm (tree search) to the problem of finding the shortest path from Arad to Bucarest with maximum depth $m = 3$.



- ***Question 3***: Does the DFS algorithm (tree search) find a solution? ***Yes***

- ***Question 4***: If the answer is “Yes”:
  - What is the solution found? ***The solution path is: Arad, Sibiu, Fagaras, Bucharest***
  - What is the cost of this solution? ***450***
  - Is this the solution of minimum cost? ***No, because there is an alternative solution with lower cost of 418: Arad, Sibiu, Rimnicu, Pitesti, Bucharest***
  - What type of solution is found by the DFS algorithm (tree search)? ***Search for solutions exploring first the deepest paths up to the maximum depth***

- ***Question 5***: What happens if a maximum depth is not defined? ***The solution would not be found, since the algorithm goes into a loop between Arad and Sibiu***

References

[1] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson, third edition, 2010.

[2] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2018.