Workbook:

Recursive Best First Search

Albert Sanchis
Jorge Civera

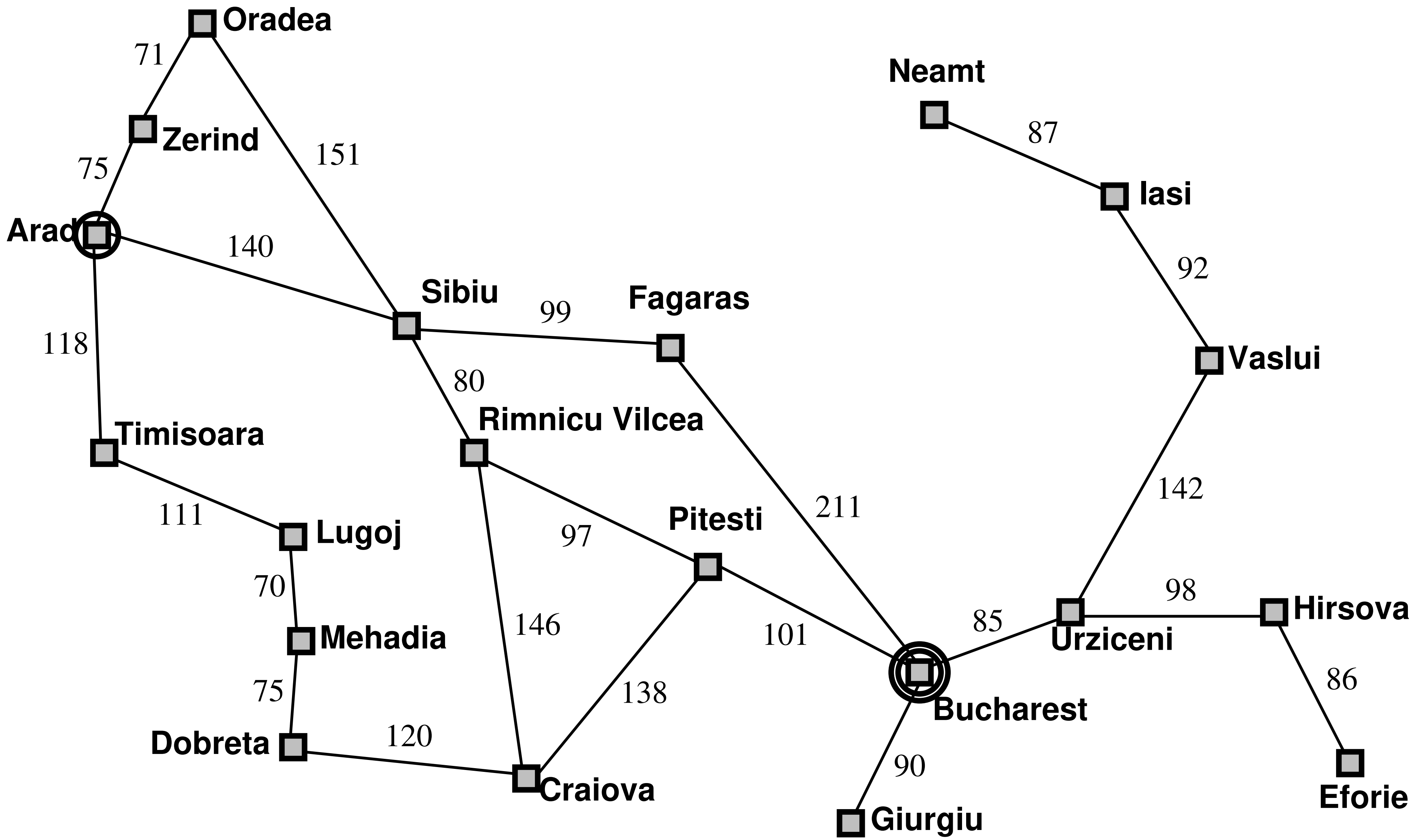
Departament de Sistemes
Informàtics i Computació

Learning objectives

- To describe the *Recursive Best First Search* (RBFS) algorithm.
- To draw the tree of RBFS search.
- To apply RBFS search to a well-known problem.
- To analyze the quality of RBFS search.

Problem: Shortest path between two points

Shortest path from Arad to Bucarest [1]:



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

Problem: Shortest path between two points

Straight-line distances to Bucharest:

	Bucharest		Bucharest
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

1 The RBFS algorithm (main) [2]

RBFS(G, s', f**)** // G weighed graph, s' start, ***evaluation function*** f

$P = InitStack(s')$ // Init ***Path*** with source node

$b = \infty$ // Init bound

$F_{s'} = f_{s'}$ // Stored value is initialised to f value

$(F_r, r) =$ **BT**$(G, P, F_{s'}, f, b)$ // Return goal state and its stored value

if $r \neq$ NULL: **return** P // If solution, return ***Path*** to goal

The RBFS algorithm (backtracking) [2]

```
BT(G, P, F_s, f, b)                          // G graph, Path P, stored value F_s', f, bound b
  s = Top(P)                                 // Path: extract node from stack
  if Goal(s): return (f_s, s)                // Solution found!
  O = InitQueue()                            // Open: priority queue for child nodes
  for all (s, n) ∈ Adjacents(G, s) and n ∉ P: // Generating child n not in the Path
    if f_s < F_s : F_n = max(f_n, F_s)       // If s visited, child may inherit stored value
    else: F_n = f_n                          // Otherwise, stored value is f value
    Push(O, n, F_n)                          // Sorting children by stored value in priority queue
  if EmptyQueue(O): return (∞, NULL)         // No children, bound = ∞
  while True:
    (n, F_n) = Top(O)                        // Best child according to stored value F
    if F_n > b: return (F_n, NULL)           // Exceeding bound, backtracking
    (n', F_n') = Top2(O)                     // 2nd best F or if it does not exist, then F_n' = ∞
    Push(P, n)                               // Add child to the Path being explored
    (F_n, r) = BT(G, P, F_n, f, min(b, F_n')) // Recursive call with possible new bound
    if r ≠ NULL: return (F_n, r)             // If sol. found, out of recursion without update
    Update(O, n, F_n)                        // Update node n in O
  Pop(P)                                     // Discard last child from Path
```

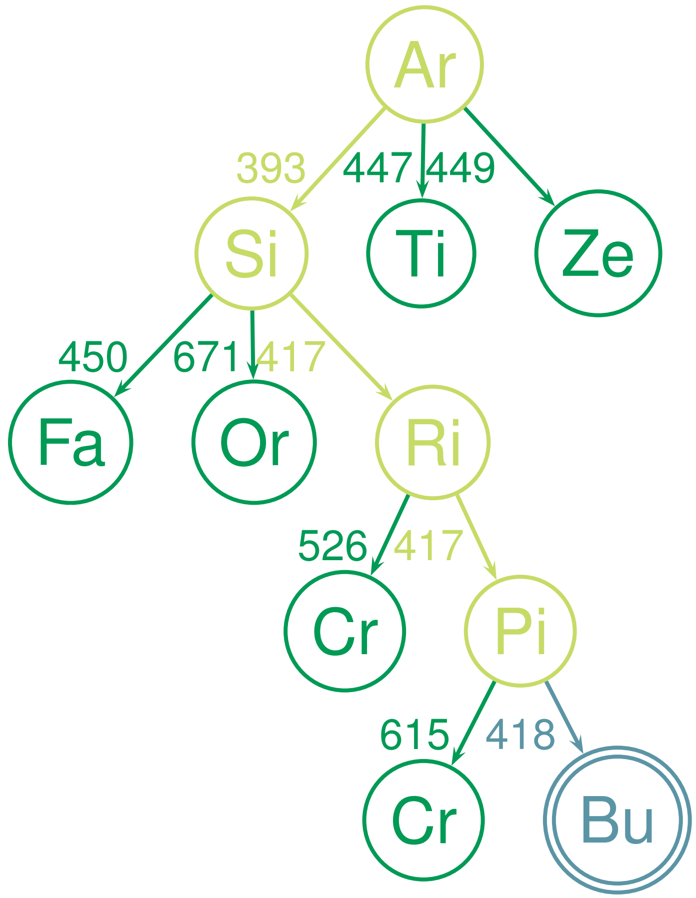
▸ ***Question 1***: Draw the search tree as a result of applying the RBFS algorithm to the problem of finding the shortest path from Arad to Bucarest.

b = ∞

b = 447

b = 447

b = 447



- ***Question 2***: Does the RBFS algorithm find a solution? ***Yes***
- ***Question 3***: If the answer is “Yes”:
  - What is the solution found? ***The solution path is: Arad, Sibiu, Rimnicu, Pitesti, Bucharest***
  - What is the cost of this solution? ***418***
  - Is this the solution of minimum cost? ***Yes***

References

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

[2] Richard E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41–78, 1993.