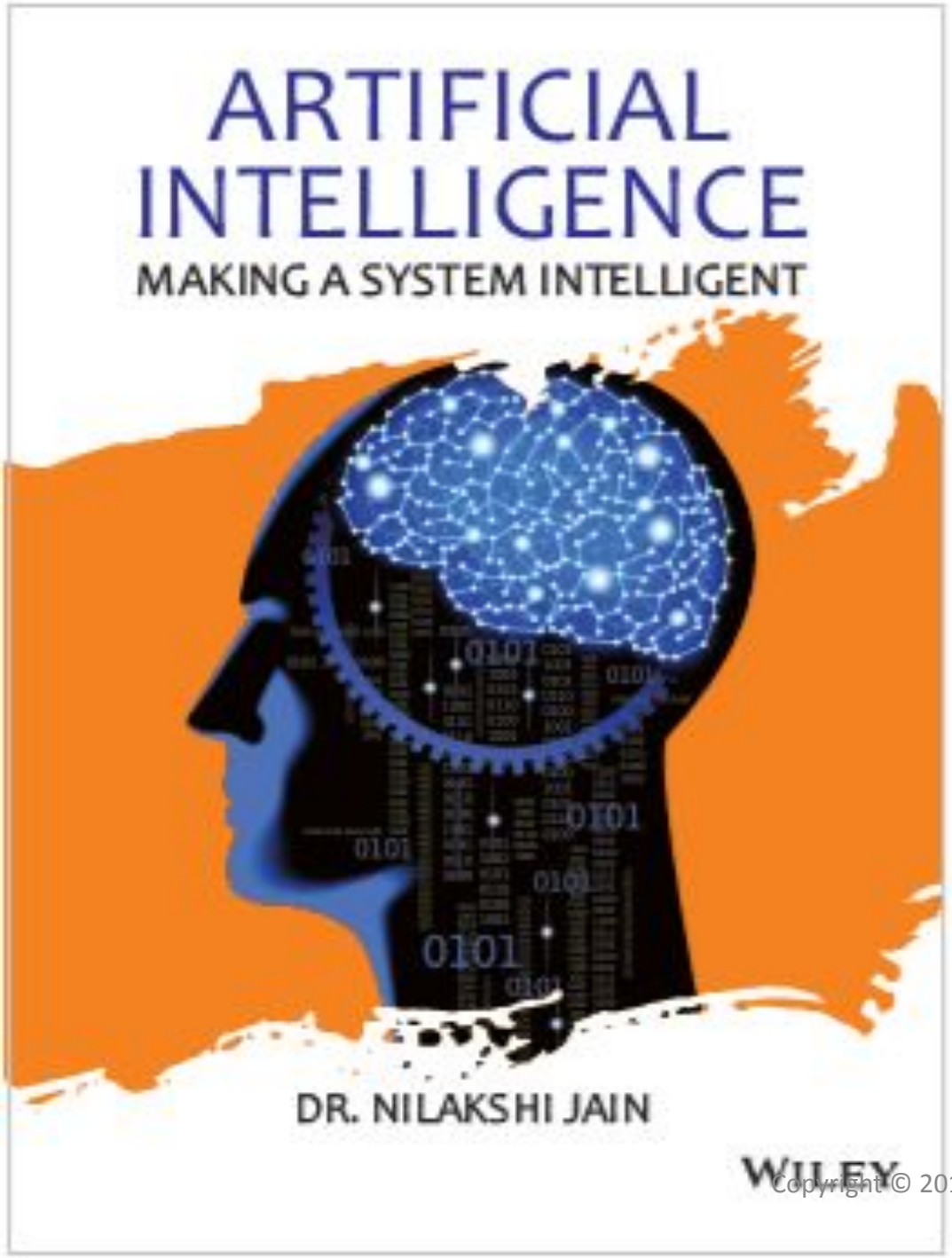


Chapter Four

Uninformed search strategies

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

4.1 Introduction

4.2 Breadth first search

4.3 Depth first search

4.4 Difference between BFS and DFS

4.5 Uniform cost search

4.6 Depth-limited search

4.7 Iterative-deeping DFs

4.8 Bidirectional Search

4.9 Comparison of uninformed search

Learning objectives.

After this chapter , students will be able to :

- Understand the concept of uninformed search
- Apply various techniques of uninformed search on various applications
- Identify, contrast and apply simple examples the major search techniques have been developed for problem-solving in artificial intelligence

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Introduction

- The uniformed search is also known as *Blind Search*. Therefore, we can also say that this type of search implies that these provide no extra information reading the states except information given by the definition of problem. Also, blind search might generate successor states distinguishing between a goal state and a non-goal state.
- Blind search or brute force is a uniformed exploration of the search space, which does not take into account either execution efficiency or planning efficiency. Blind search is also known as *uniform search*.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

- **Uninformed (blind) search strategies use only the information available in the problem definition:**
  - **Breadth-first search**
  - **Uniform-cost search**
  - **Depth-first search**
  - **Depth-limited search**
  - **Iterative deepening search**
  - **Bidirectional search**
- **Key issue: type of queue used for the fringe of the search tree**
- **(collection of tree nodes that have been generated but not yet expanded)**

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Blind Strategies(uniform search)

- Breadth-first
  - Bidirectional
- Depth-first
  - Depth-limited
  - Iterative deepening

} Arc cost = 1

- Uniform-Cost (variant of breadth-first)

} Arc cost = c(action) ≥ ε > 0

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Breadth first search

- A basic technique of traversing a graph is breadth-first search or BFS. It first finds the shortest path always; however, this might also lead to using of more memory. The state space is shown in the form of a tree in this type of search.
- One can obtain the solution by traversing through the tree. The nodes of the tree display the start value or starting state, various intermediate states and the final state.
- Each node in the search tree is developed breadth-wise at each level.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Breadth first search **ALGORITHM**

Begin

Initially, OPEN has the root node and CLOSE is EMPTY OPEN=[start]

CLOSE=[ ]

The loop is continued till OPEN list is not EMPTY While OPEN [] do

Begin

Remove the leftmost state from OPEN and let it be *X*

If *X* is GOAL then Return SUCCESS Else

Begin

Generate children of *x*

Substitute *x* on close.

Discard the children of *x* if already on OPEN or CLOSE.

Place the remaining children on the right side of the OPEN.

(For BFS, we place the children to the right side of OPEN End)

End

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Breadth first search

ADVANTAGES

- In this procedure, at any way it will find the GOAL.
- It does not follow a single unfruitful path for a long time.
- It finds the minimal solution in case of multiple paths.

DISADVANTAGES

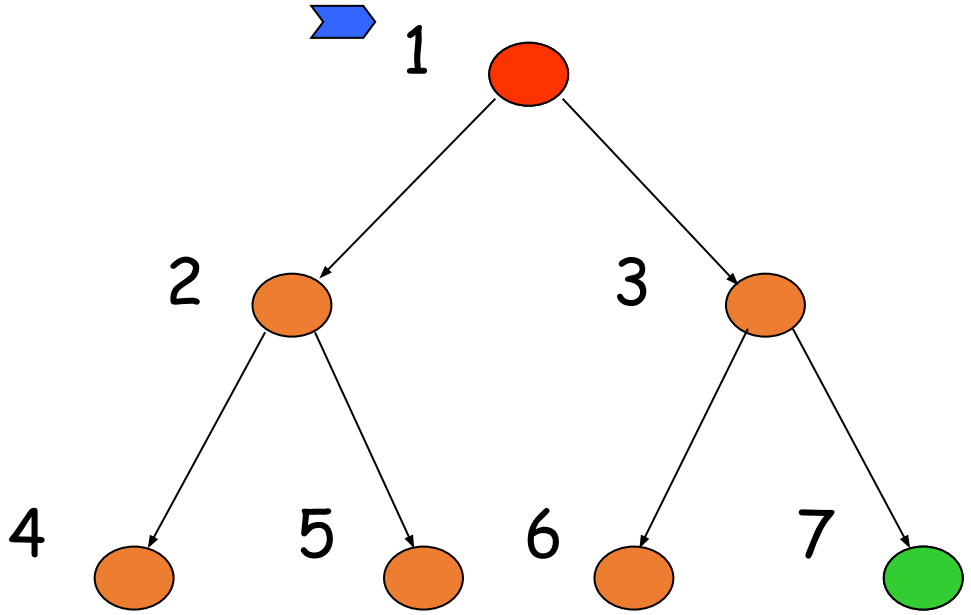
1. Large memory space is consumed by BFS.
2. Time complexity is more.
3. It also has long pathways, when all the paths to a destination are on approximately the same search depth.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Breadth-First Strategy

New nodes are inserted at the end of FRINGE

FRINGE = (1)

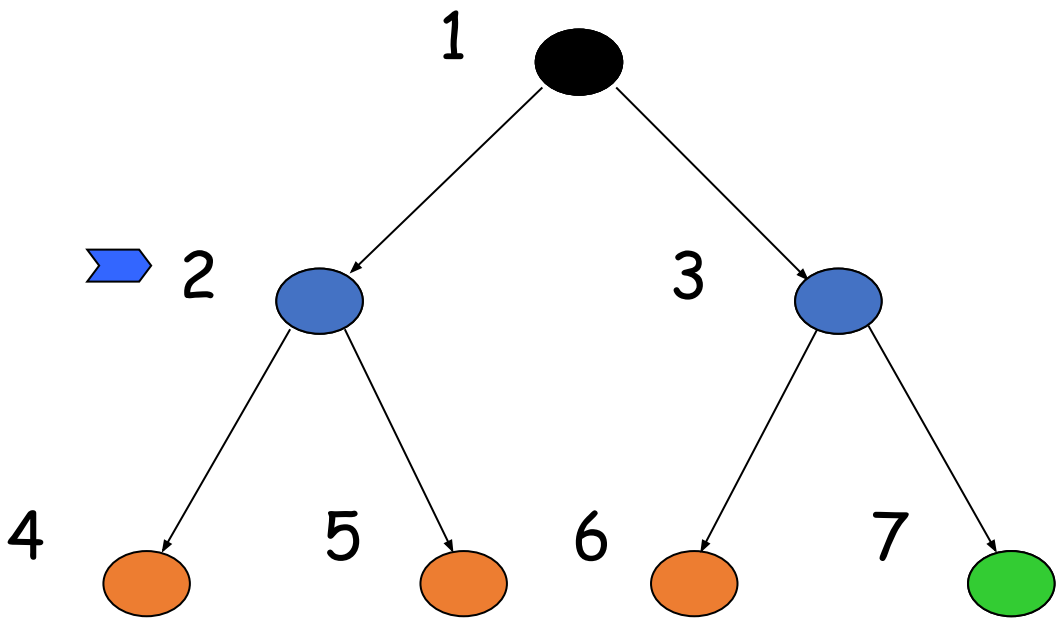


Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Breadth-First Strategy

New nodes are inserted at the end of FRINGE

FRINGE = (2, 3)

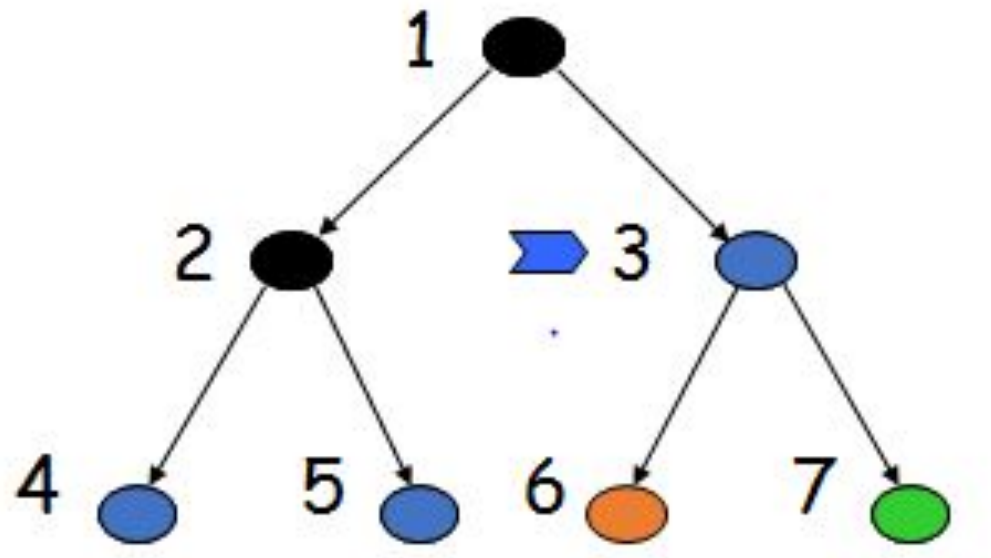


Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Breadth-First Strategy

New nodes are inserted at the end of FRINGE

FRINGE = (3, 4, 5)

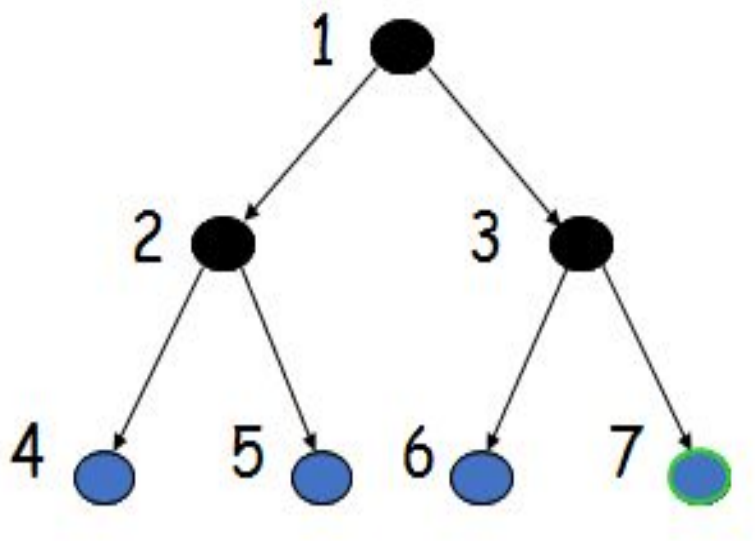


Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Breadth-First Strategy

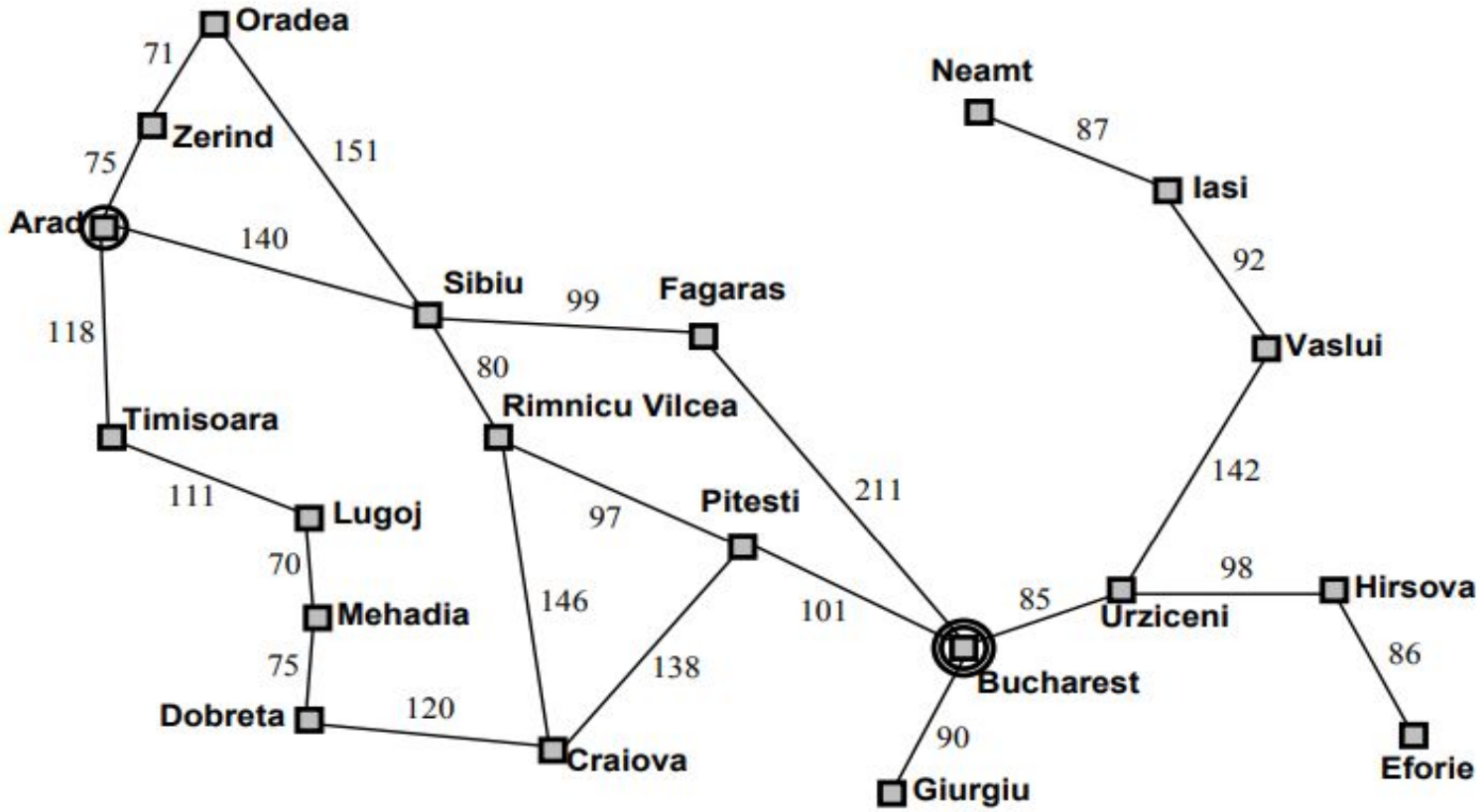
New nodes are inserted at the end of FRINGE

FRINGE = (4, 5, 6, 7)



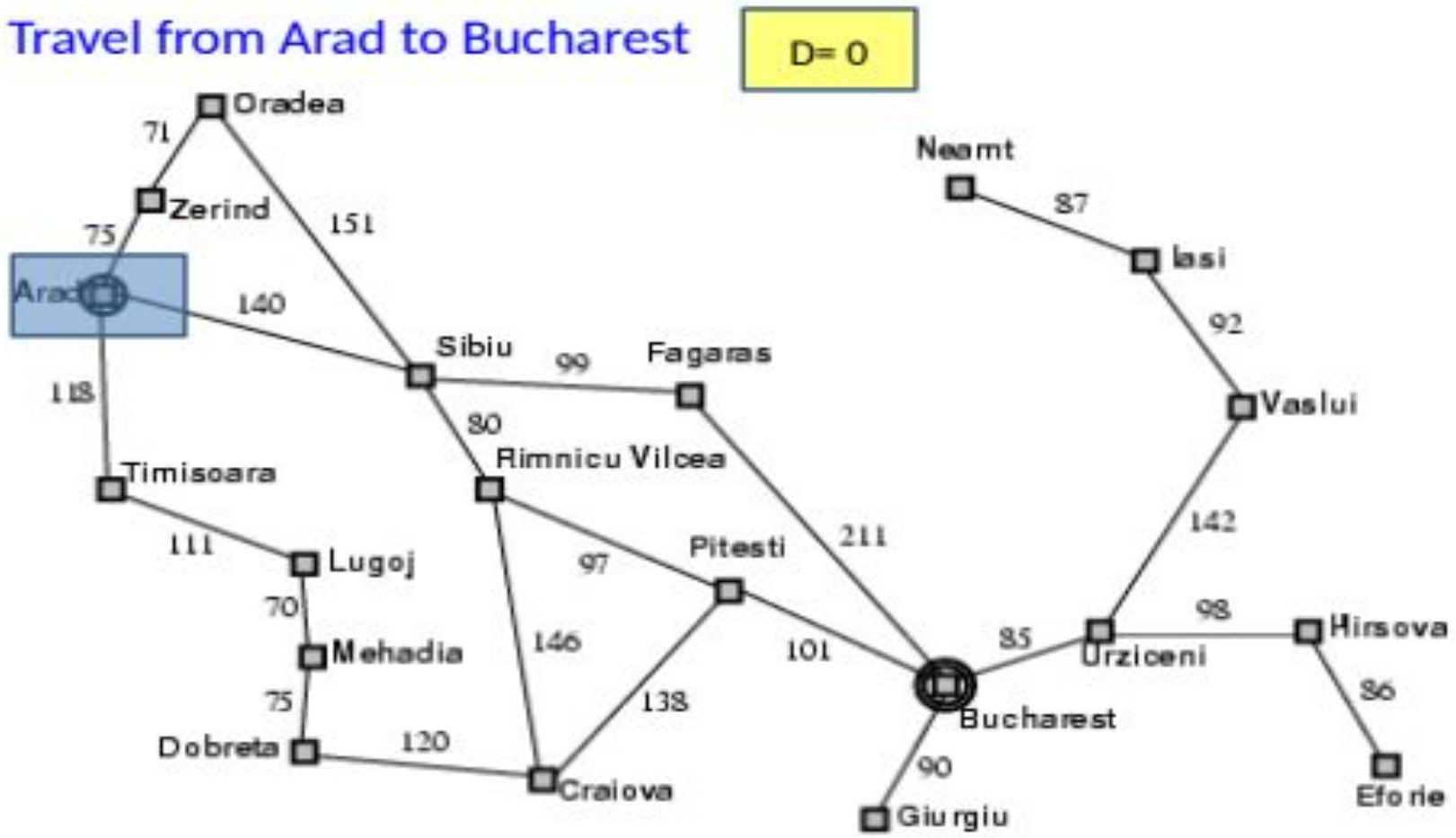
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Problem: To find a route from Arad to Bucharest (Romania Problem)



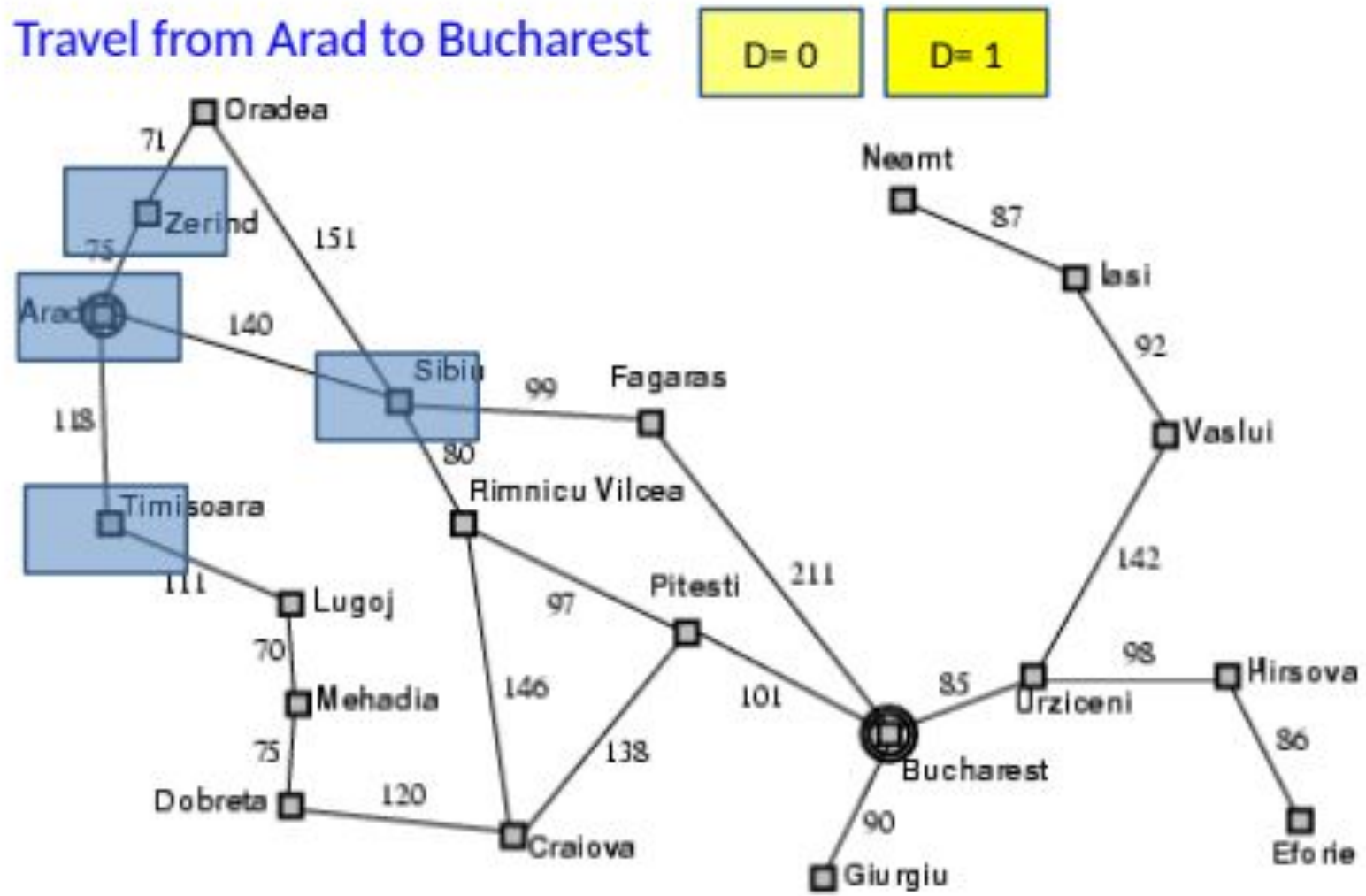
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania BFS



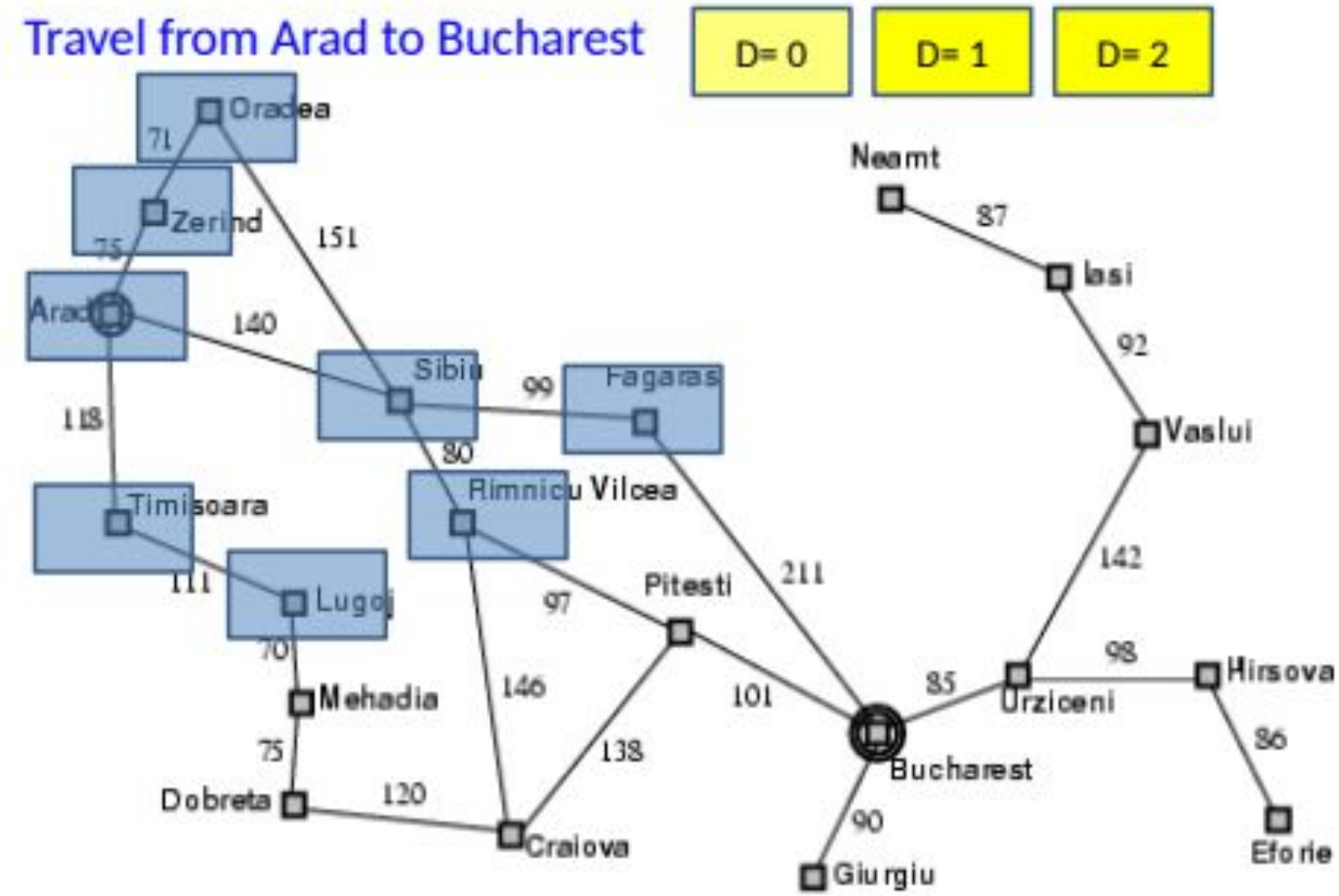
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania BFS



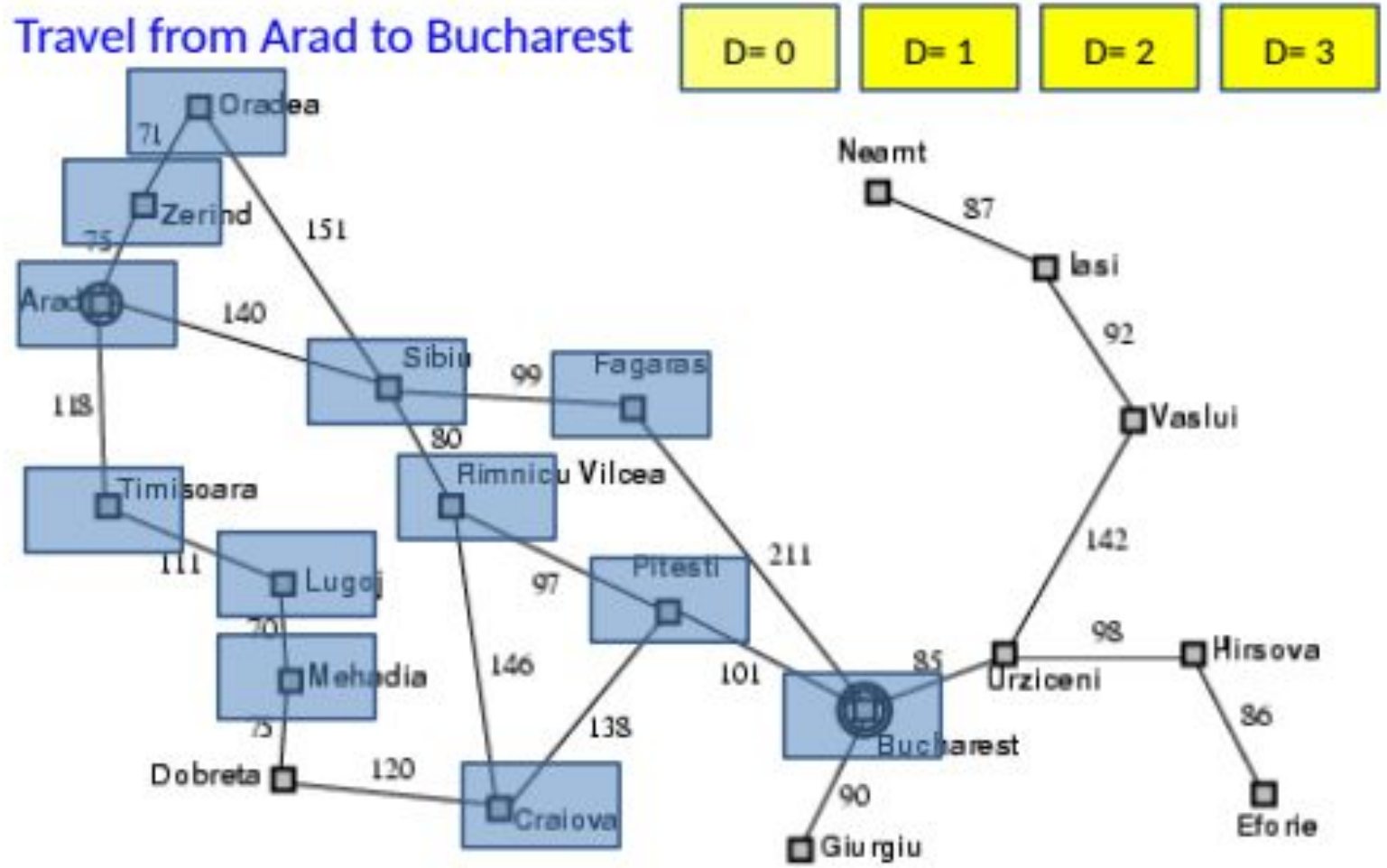
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania BFS



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania BFS



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Evaluation

- b: branching factor
- d: depth of shallowest goal node
- Breadth-first search is:
  - Complete
  - Optimal if step cost is 1
- Number of nodes generated:
  $1 + b + b^2 + \ldots + b^d = (b^{d+1}-1)/(b-1) = O(b^d)$
- Time and space complexity is $O(b^d)$

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Big O Notation

$g(n) = O(f(n))$ if there exist two positive constants a and N such that:

for all $n > N$: $g(n) \leq a \times f(n)$

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Time and Memory Requirements

d	# Nodes	Time	Memory
2	111	.01 msec	11 Kbytes
4	11,111	1 msec	1 Mbyte
6	~10^6	1 sec	100 Mb
8	~10^8	100 sec	10 Gbytes
10	~10^{10}	2.8 hours	1 Tbyte
12	~10^{12}	11.6 days	100 Tbytes
14	~10^{14}	3.2 years	10,000 Tbytes

Assumptions: b = 10; 1,000,000 nodes/sec; 100bytes/node

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Time and Memory Requirements

d	# Nodes	Time	Memory
2	111	.01 msec	11 Kbytes
4	11,111	1 msec	1 Mbyte
6	~10^6	1 sec	100 Mb
8	~10^8	100 sec	10 Gbytes
10	~10^{10}	2.8 hours	1 Tbyte
12	~10^{12}	11.6 days	100 Tbytes
14	~10^{14}	3.2 years	10,000 Tbytes

Assumptions: b = 10; 1,000,000 nodes/sec; 100bytes/node

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

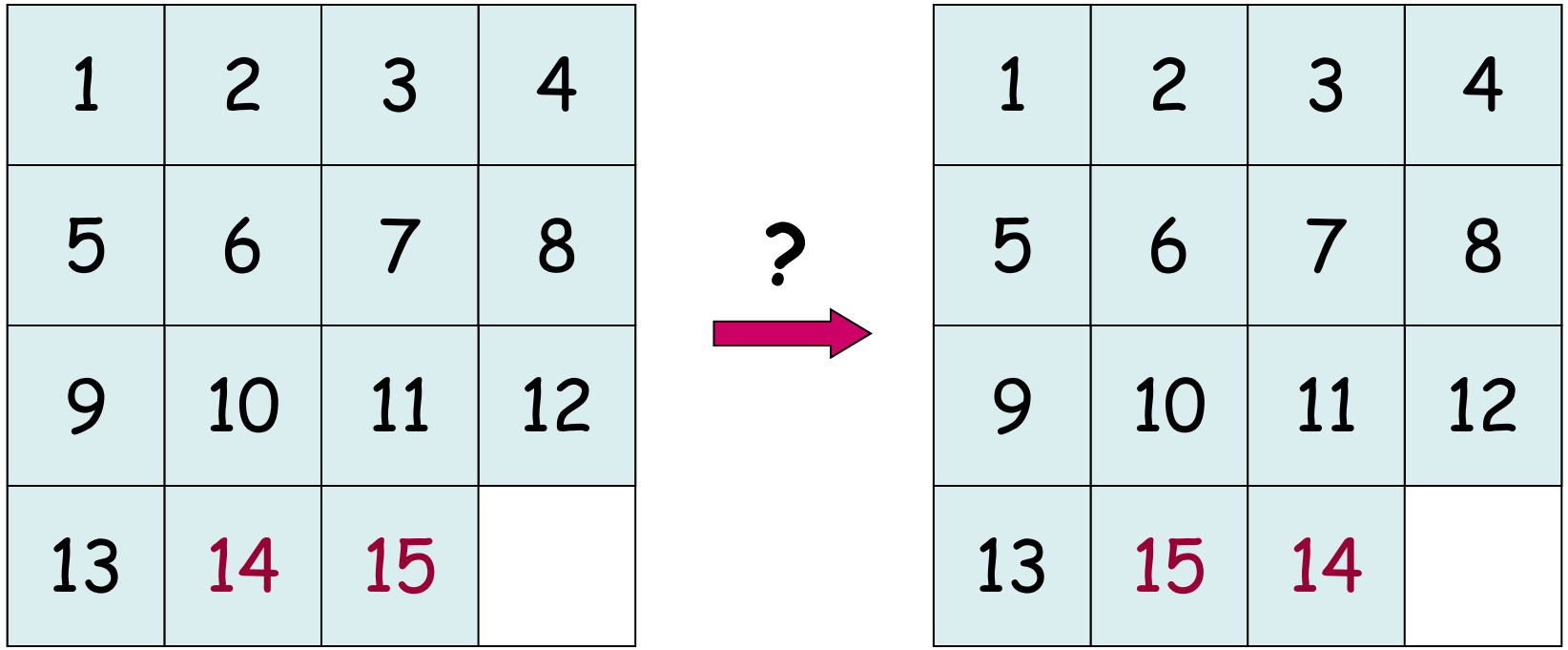
TIME COMPLEXITY

A hypothetical state space is considered where each state has 'b' successors. The root that generates 'b' is not the last node generating $bd + 1 - d$ nodes at level $d + 1$. Thus, the number of nodes generated will be $b + b2 + b3 + \ldots + bd + (bd + 1 - d)$. Hence, the time complexity = $O(bd + 1)$.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Remark

If a problem has no solution, breadth-first may run for ever (if the state space is infinite or states can be revisited arbitrary many times)



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

- In this, instead of probing the width, one branch of a tree can be explored until the solution is found. Also, it might be decided to terminate the search due to the reason of it either not meeting the dead end, or encountering some previous state, or might be because the process becomes longer than the set time limit. If any of the aforementioned situations is encountered, the process is terminated resulting in a backtracking. A fresh search will be performed on some other branch of the tree, repeating the process until the goal state is reached. This type of technique is known as DFS technique to left.
- DFS always expands one of the nodes at the deepest level of the tree. Once the search hits the dead end, that is, the non-goal node with no expansion, the backtracks technique starts at the shallower level and starts expanding from there on.
- For the state space with branching factor b and maximum depth m, storage required by DFS is only Nodes bm in contrast to $O(b4)$ of the BFS.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

ALGORITHM

Begin

Initially, OPEN has the root node and CLOSE is EMPTY

OPEN=[start] CLOSE=[]

The loop is continued till OPEN list is not EMPTY

While OPEN #[] Begin

The leftmost state is removed from OPEN and is called as X If X=GOAL then Return SUCCESS Else

Begin

Children of X is generated.
X is placed on close.
The children of X is discarded in case already on OPEN or CLOSE.
The remaining children are placed on the left side of OPEN.

End Return Fall End

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

ADVANTAGES

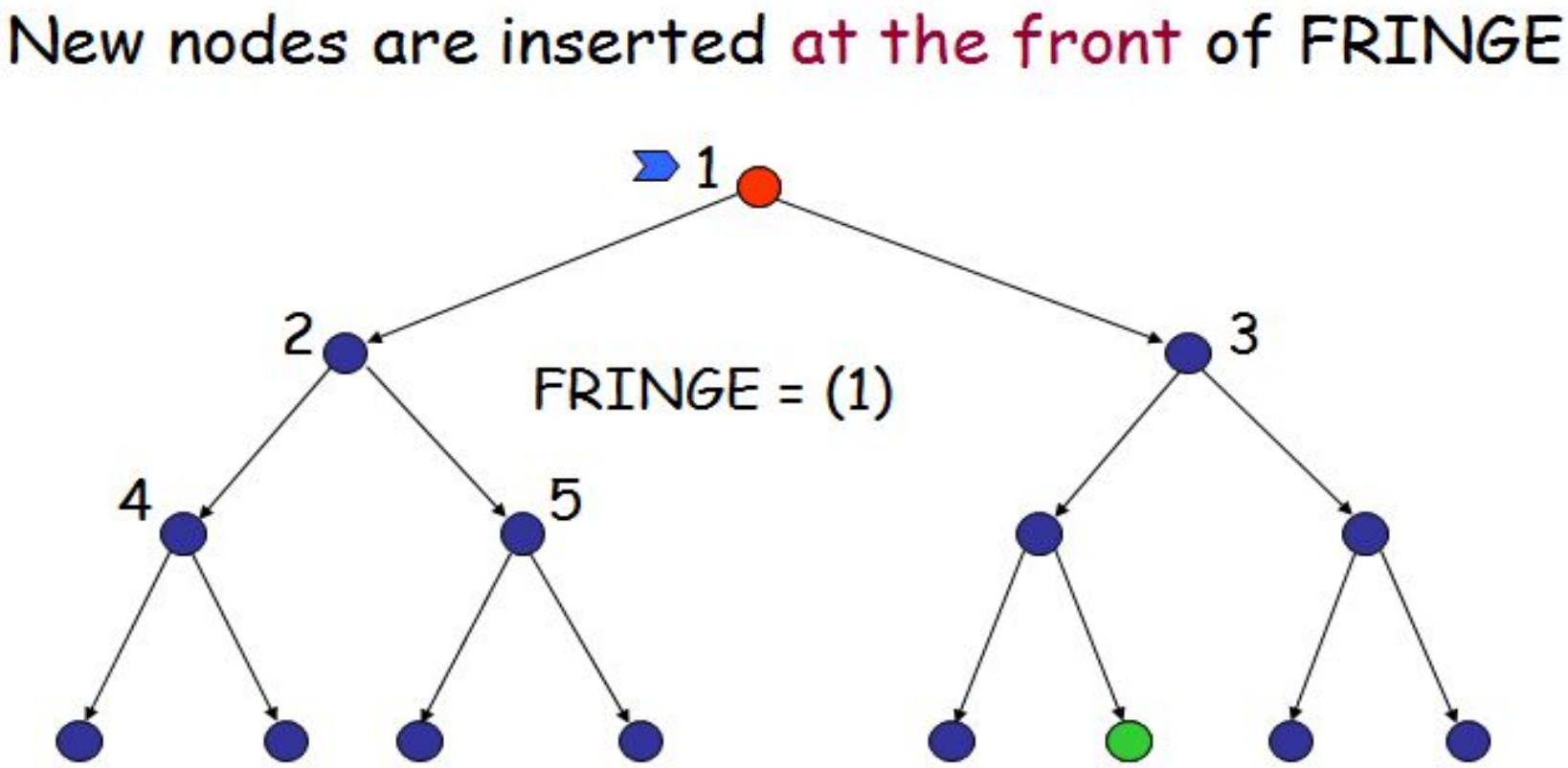
- DFS requires less memory as only the nodes to the current path are stored.
- In case DFS might also find a solution without examining much of the search space decreasing the amount of time associated considerably.

DISADVANTAGES

- DFS can be trapped by following a single unfaithful path, which might terminate in a state which has no successors.
- Also, no minimal solution is guaranteed by DFS.

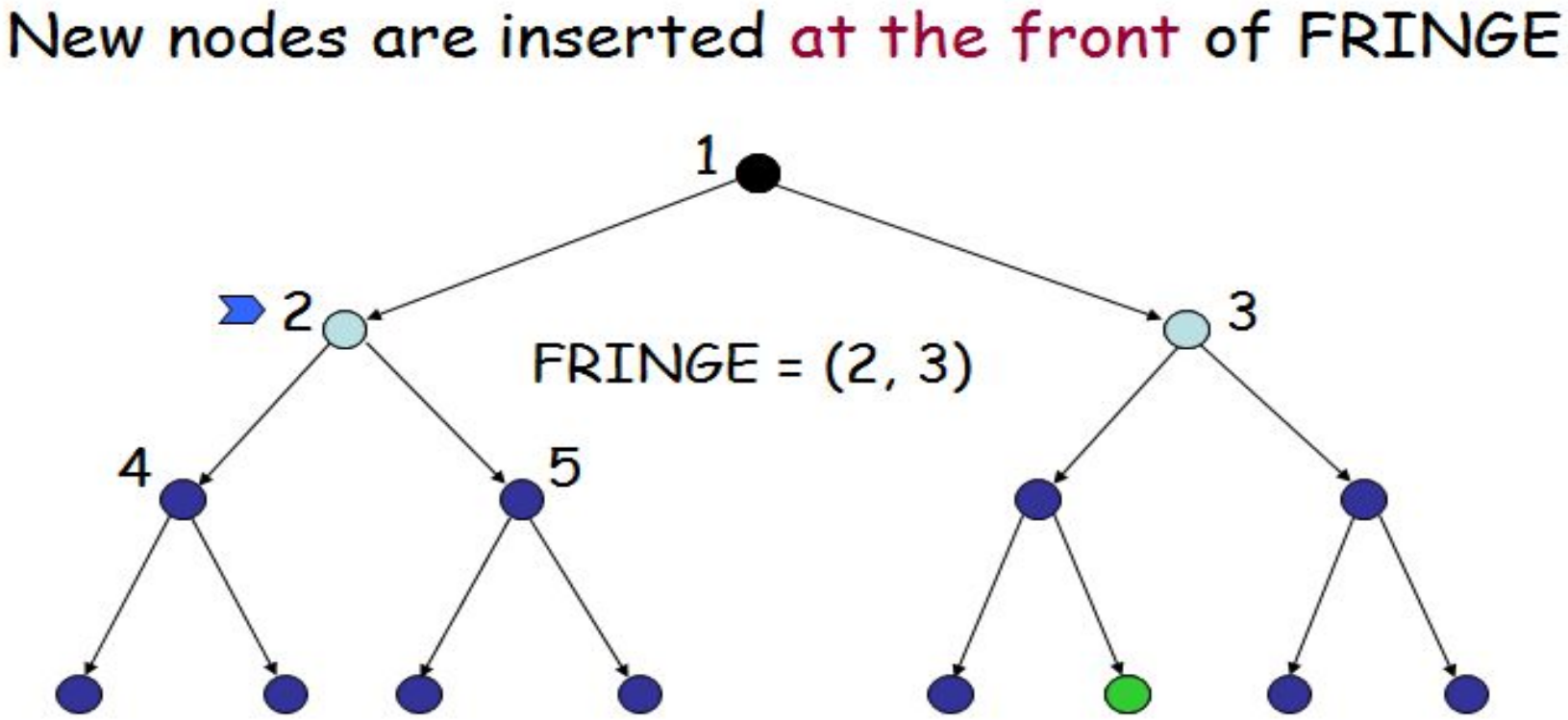
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search



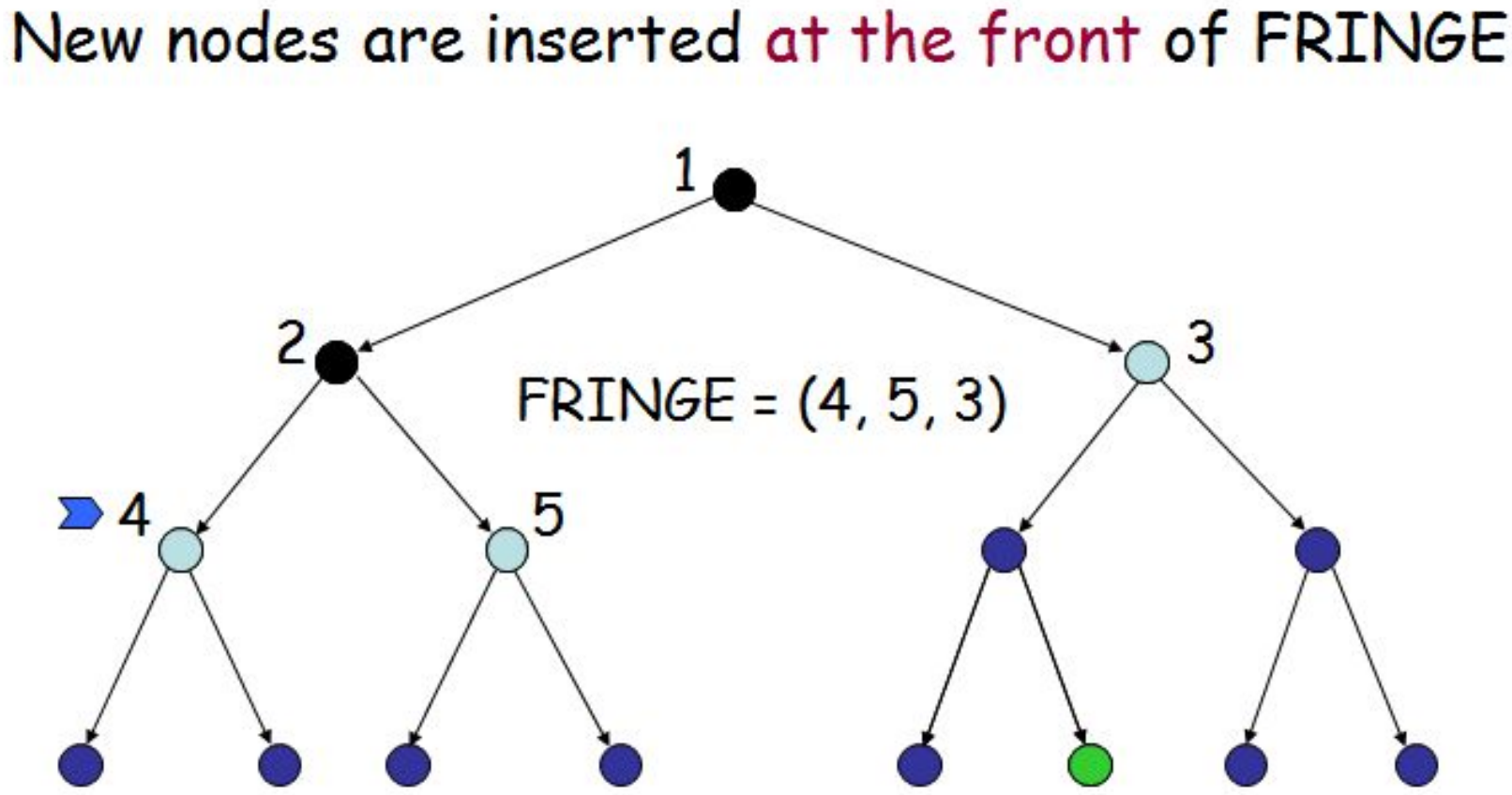
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search



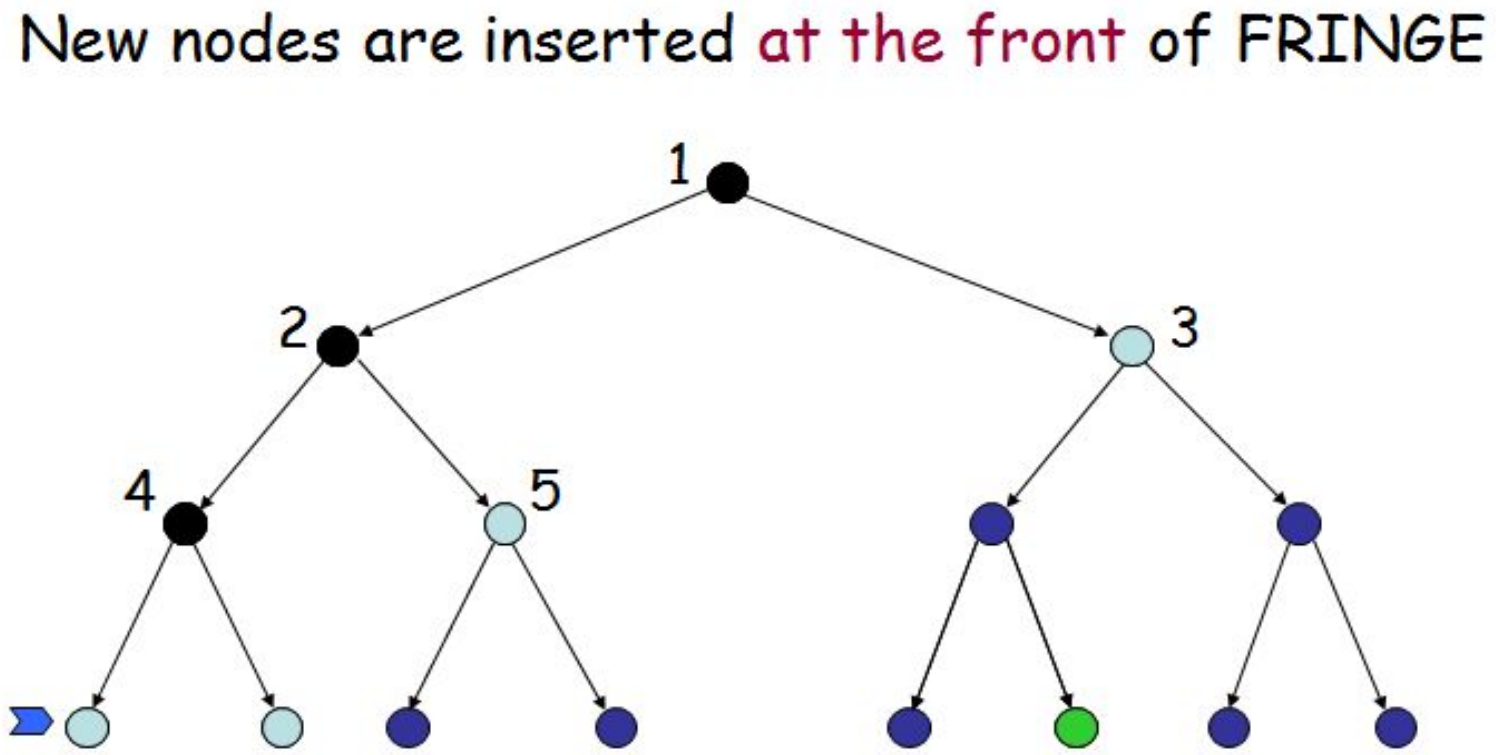
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

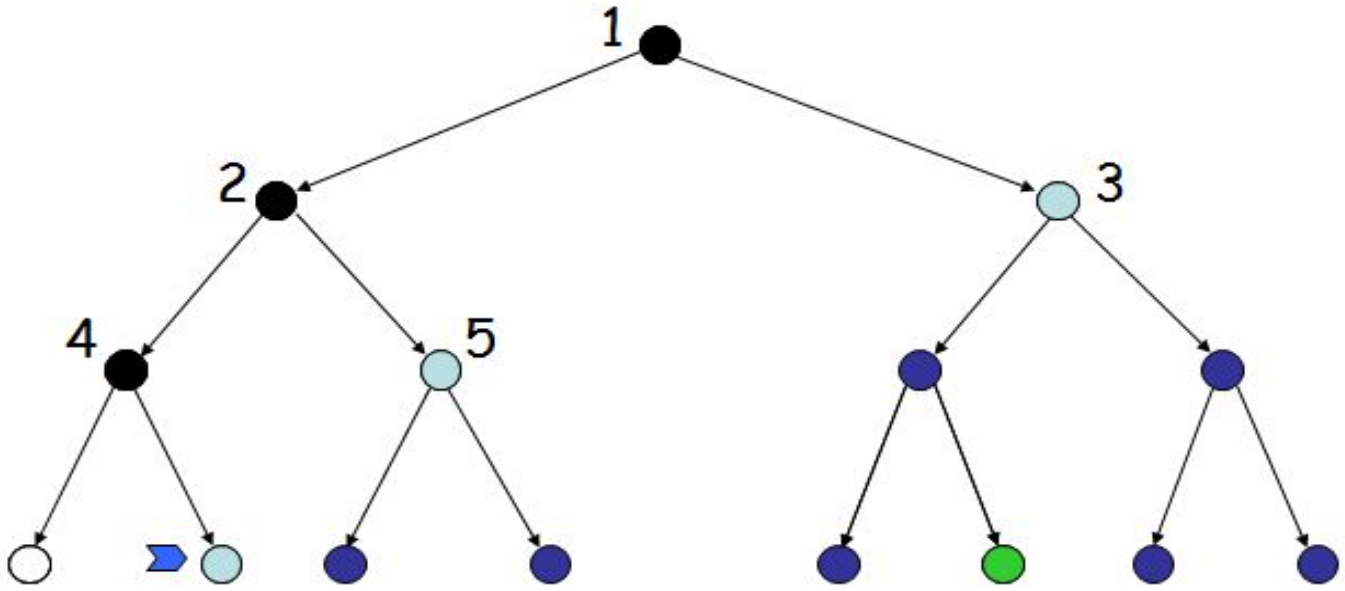
Depth first search



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

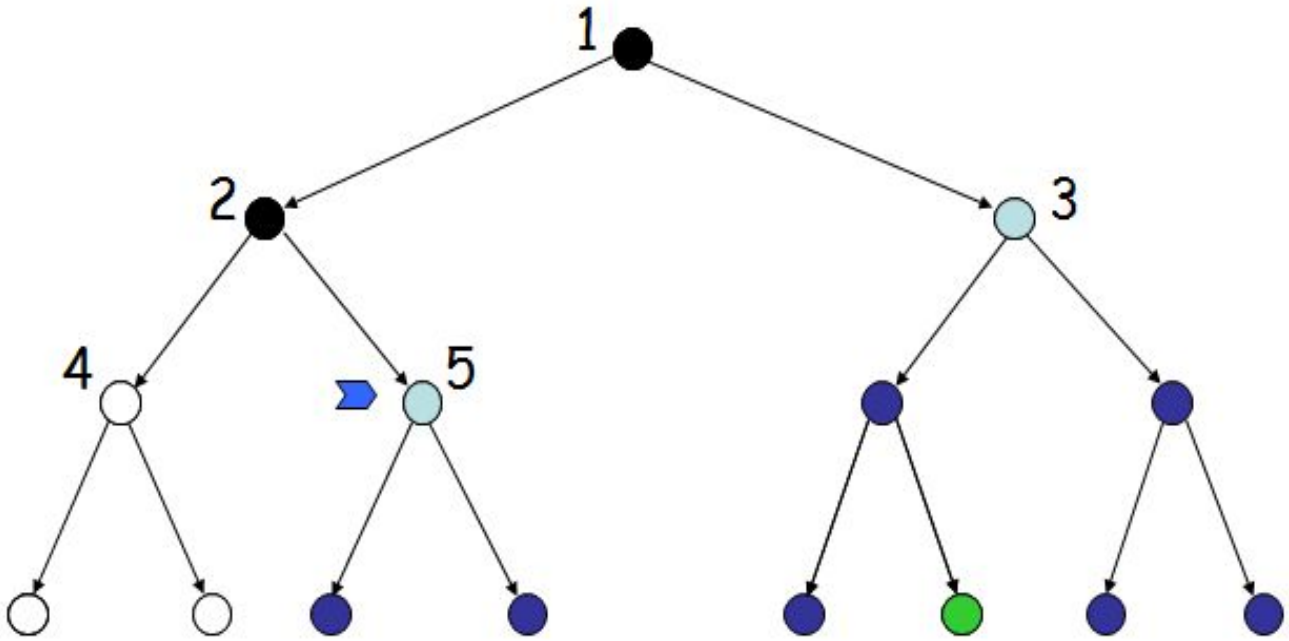
New nodes are inserted at the front of FRINGE



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

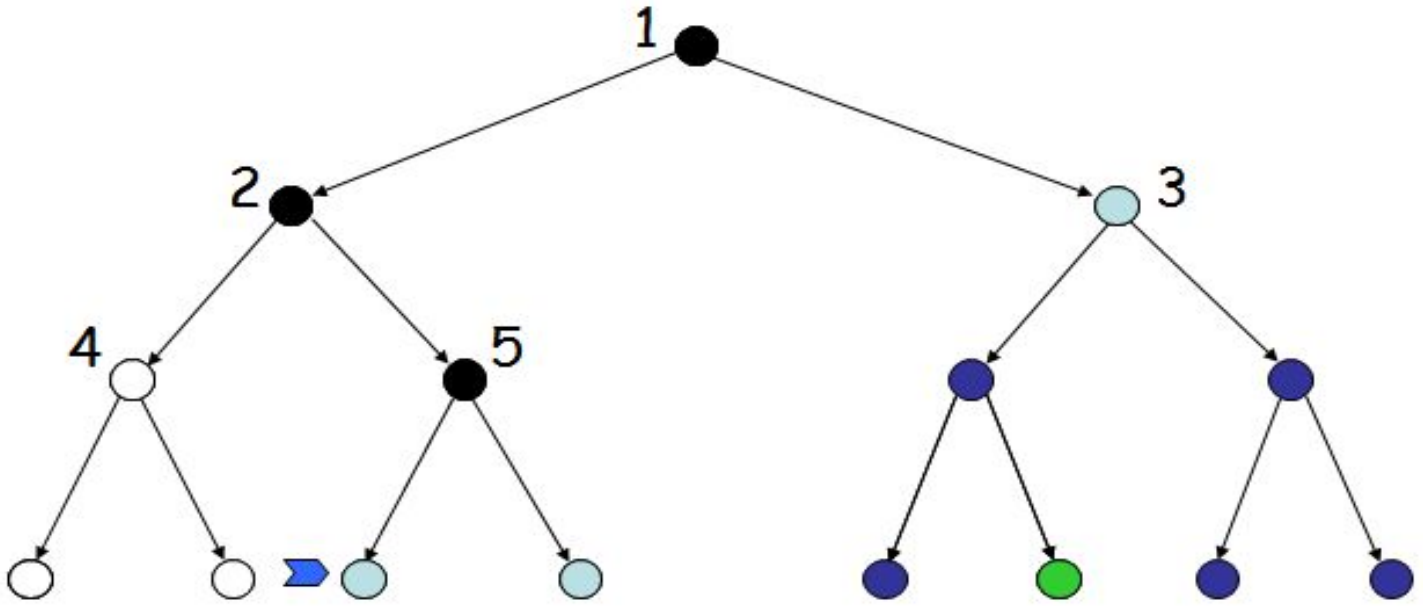
New nodes are inserted at the front of FRINGE



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

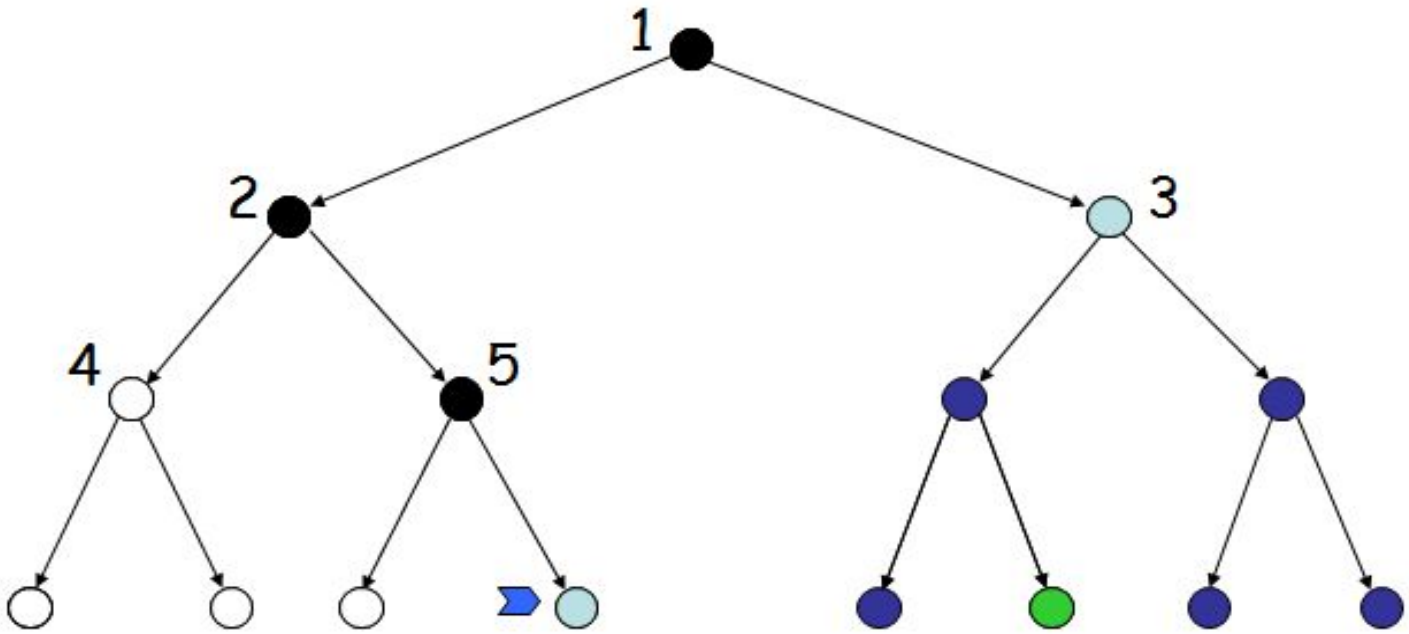
New nodes are inserted at the front of FRINGE



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

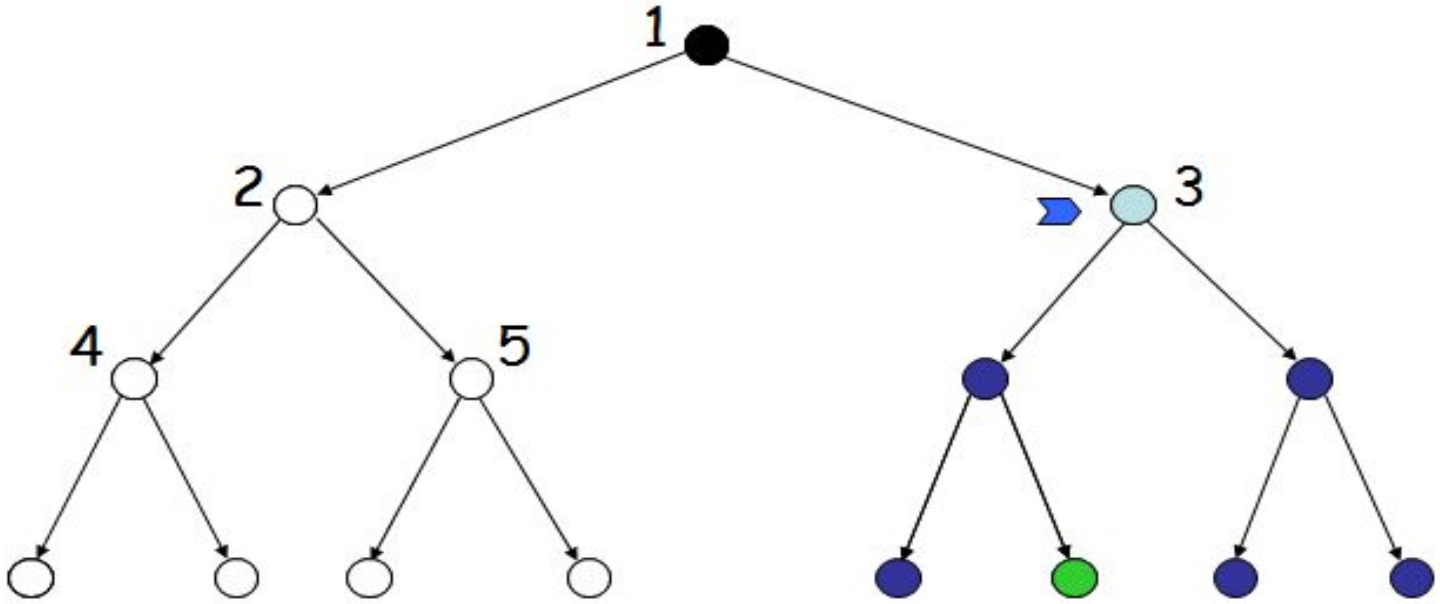
New nodes are inserted at the front of FRINGE



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

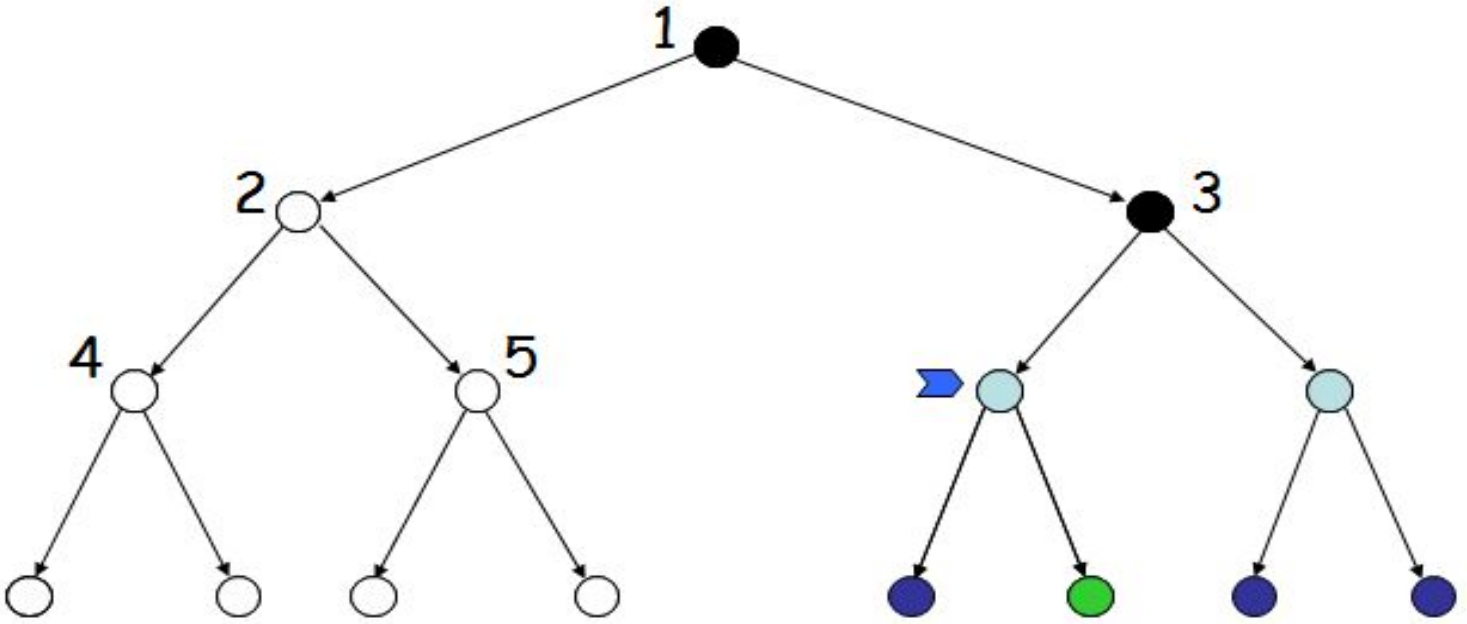
New nodes are inserted at the front of FRINGE



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth first search

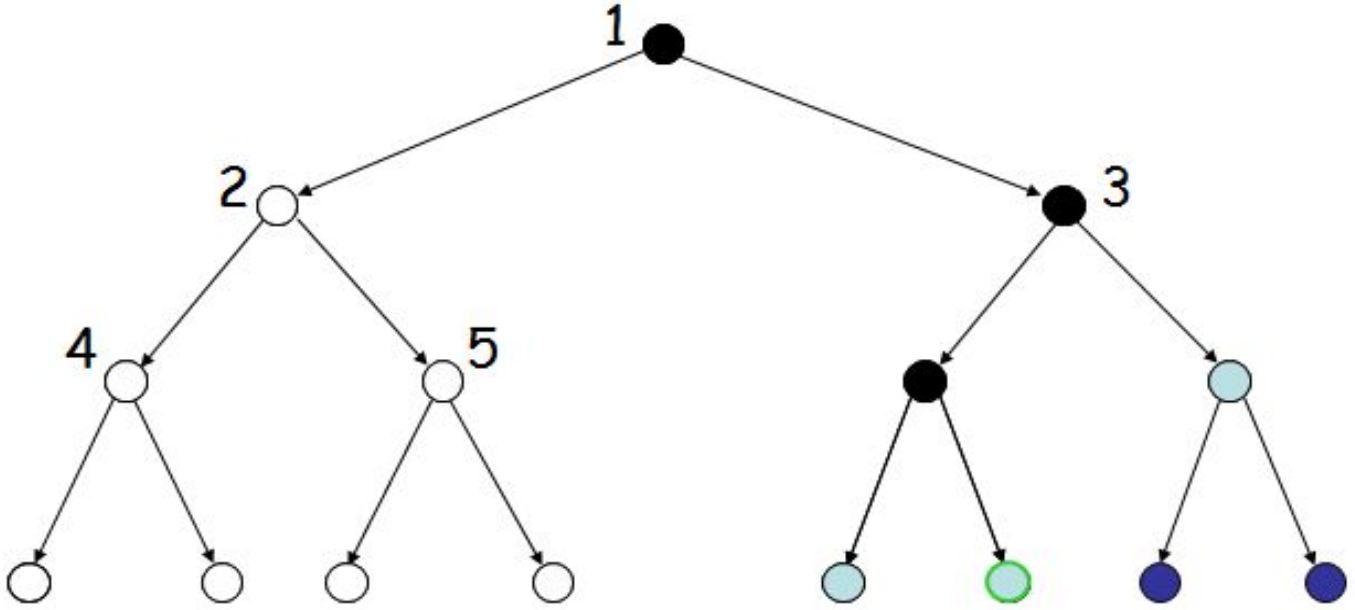
New nodes are inserted at the front of FRINGE



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

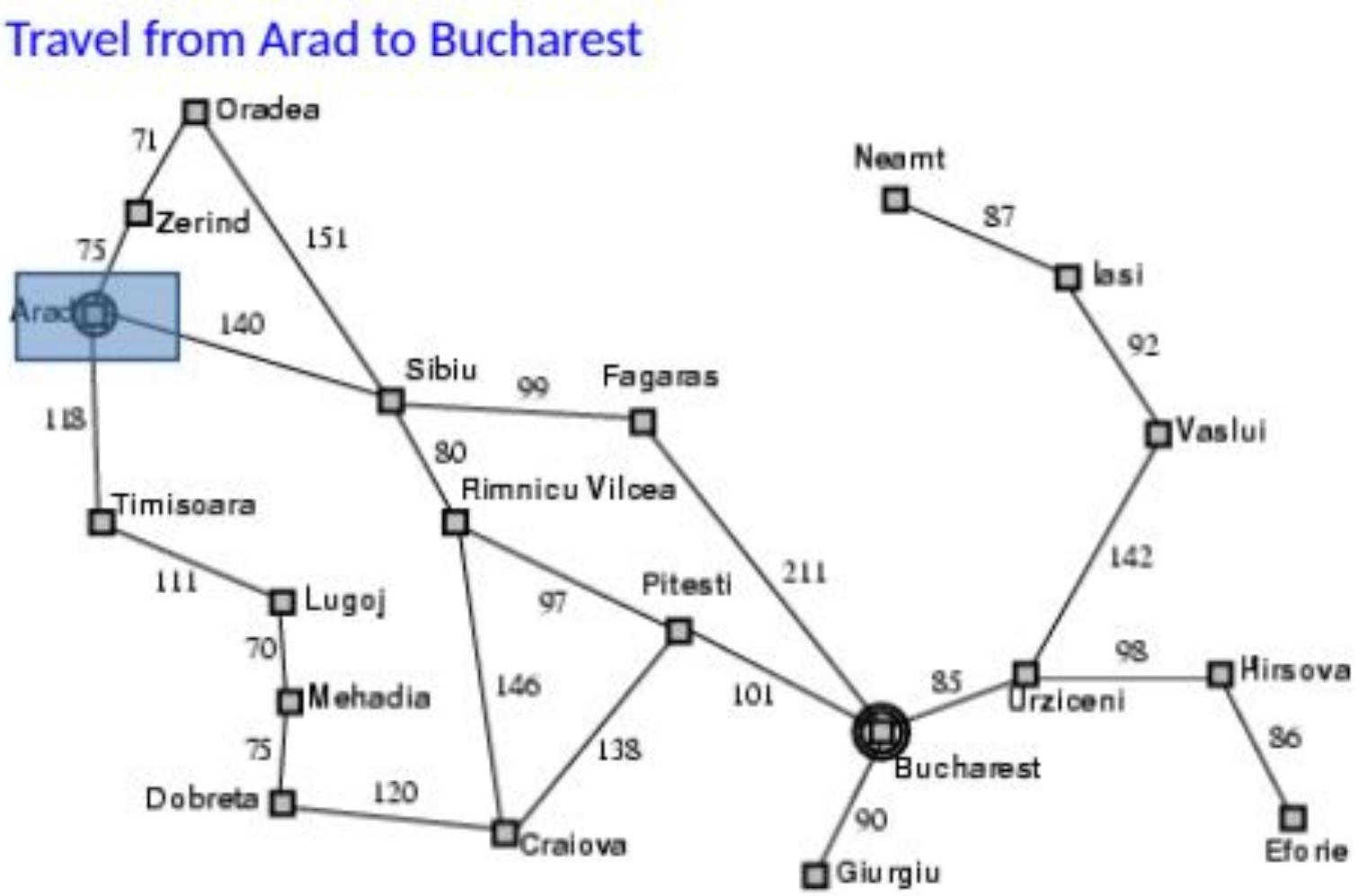
Depth first search

New nodes are inserted at the front of FRINGE



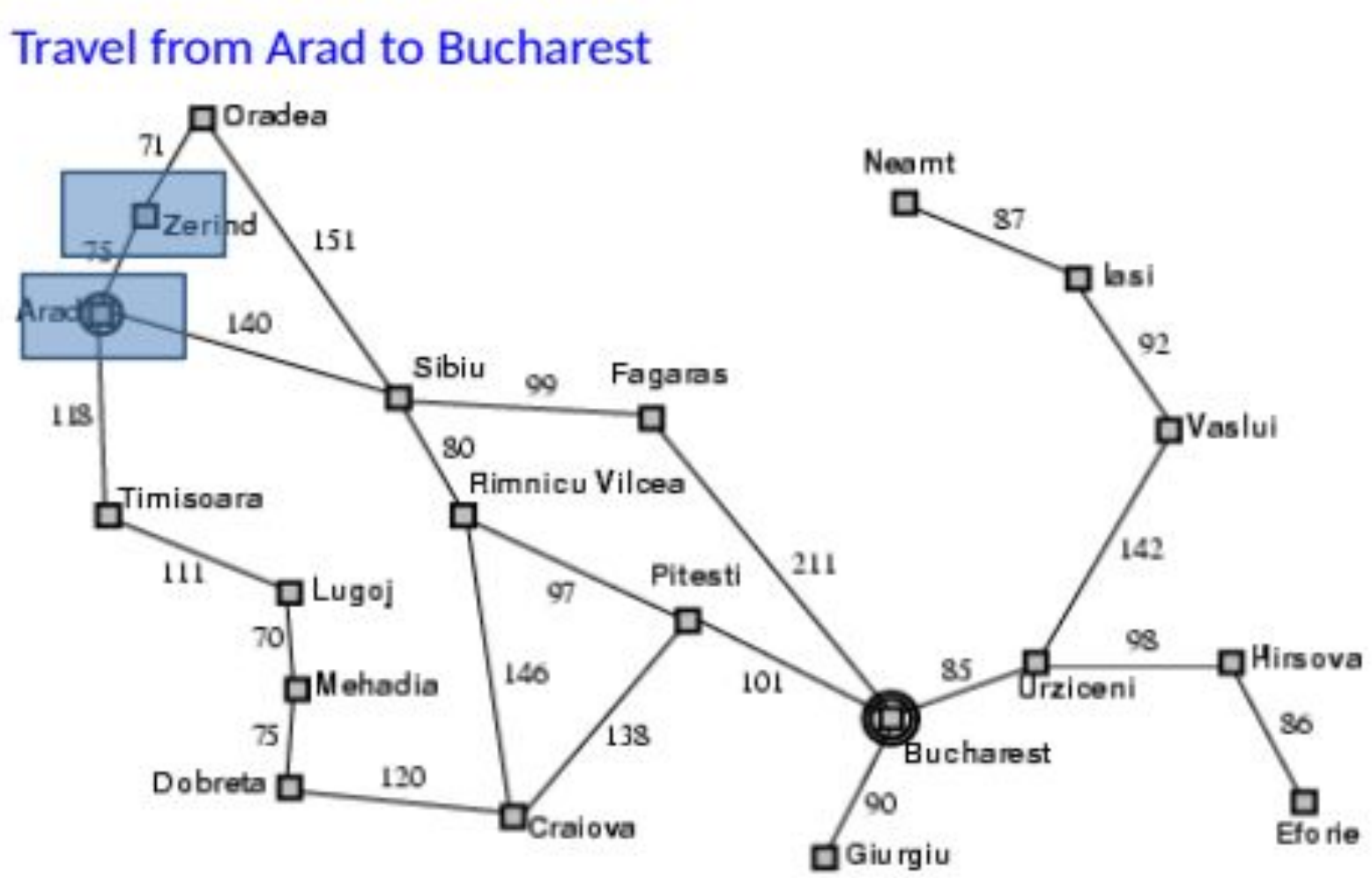
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania DFS



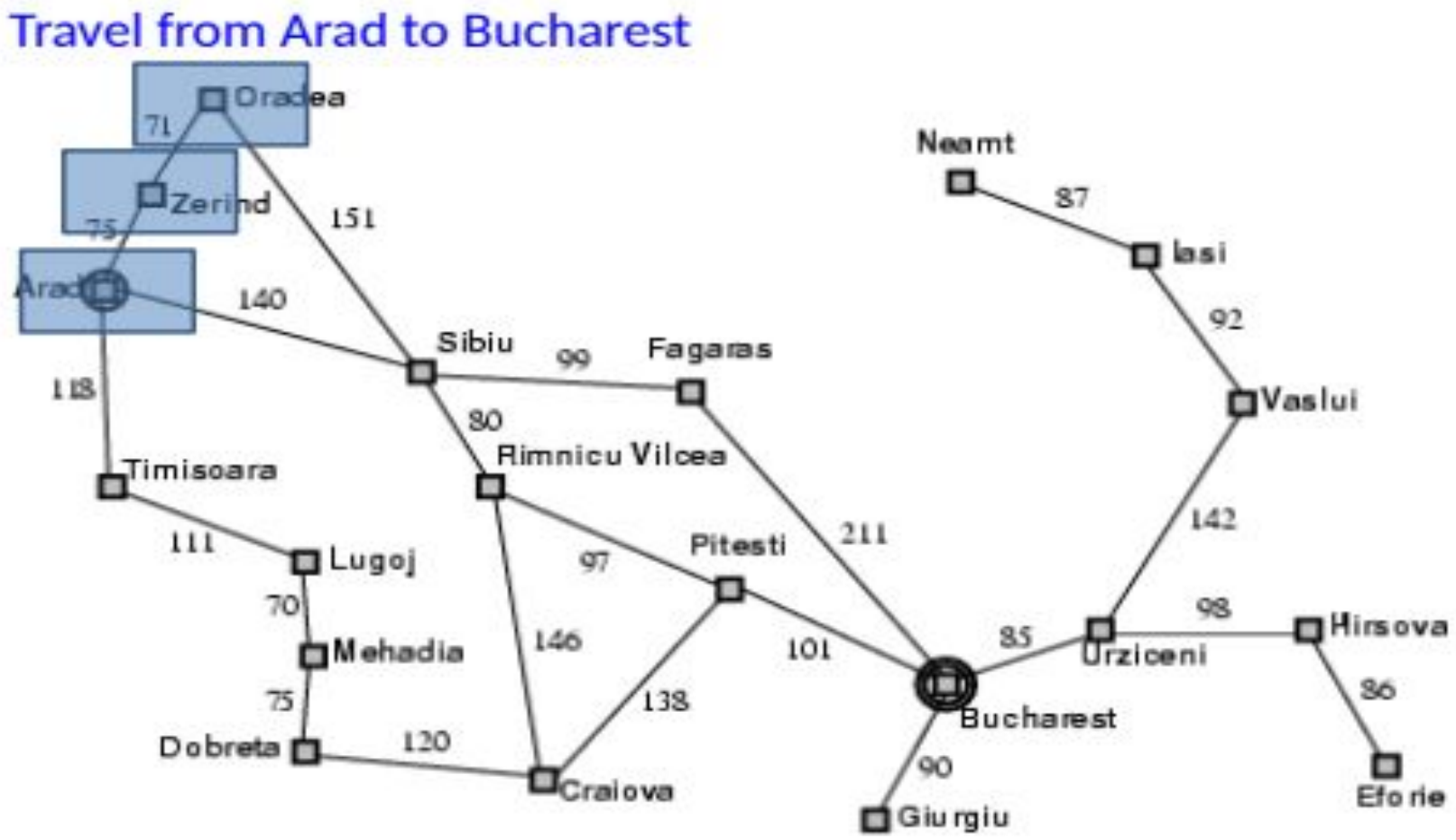
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania DFS



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania DFS

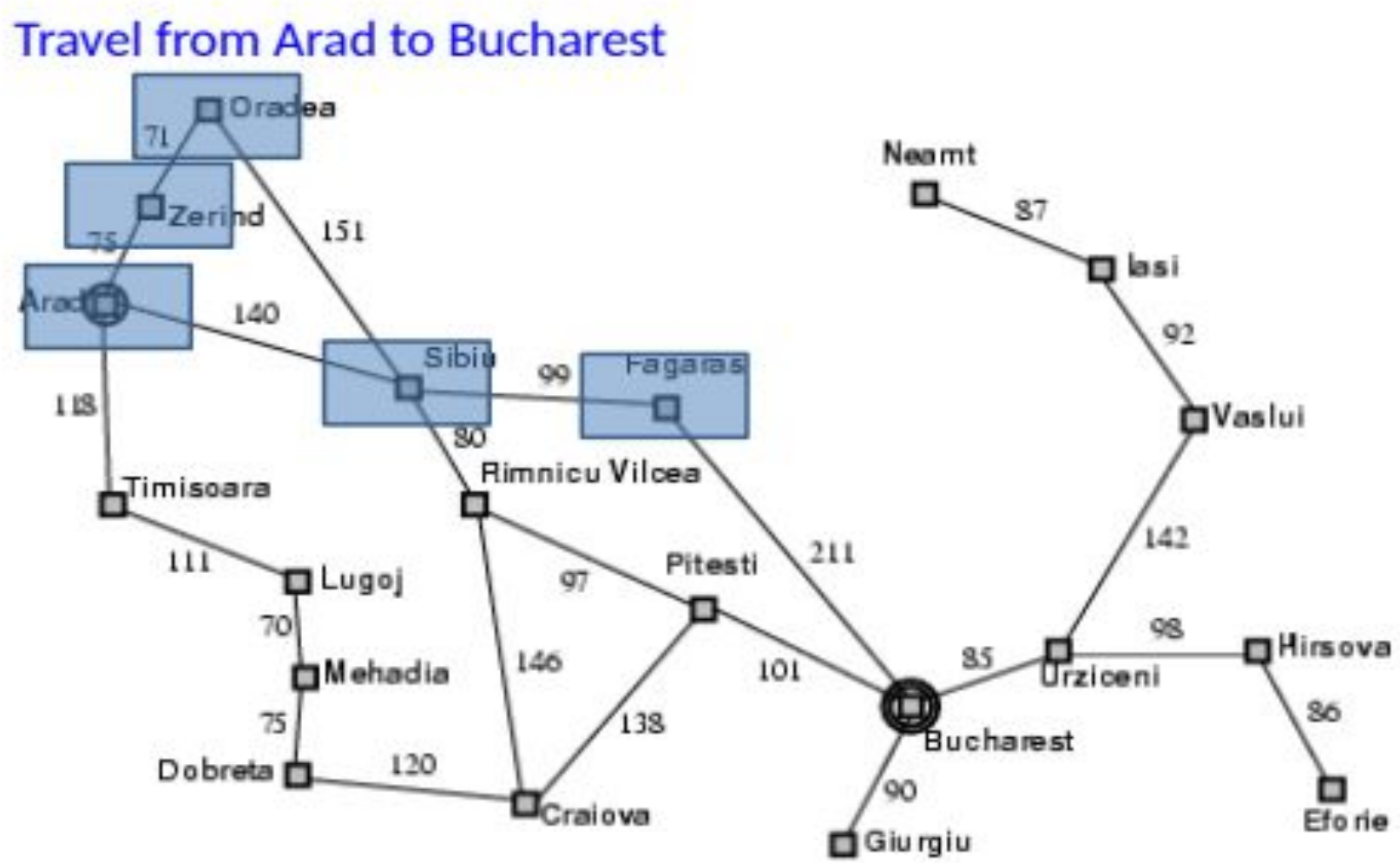


Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania DFS

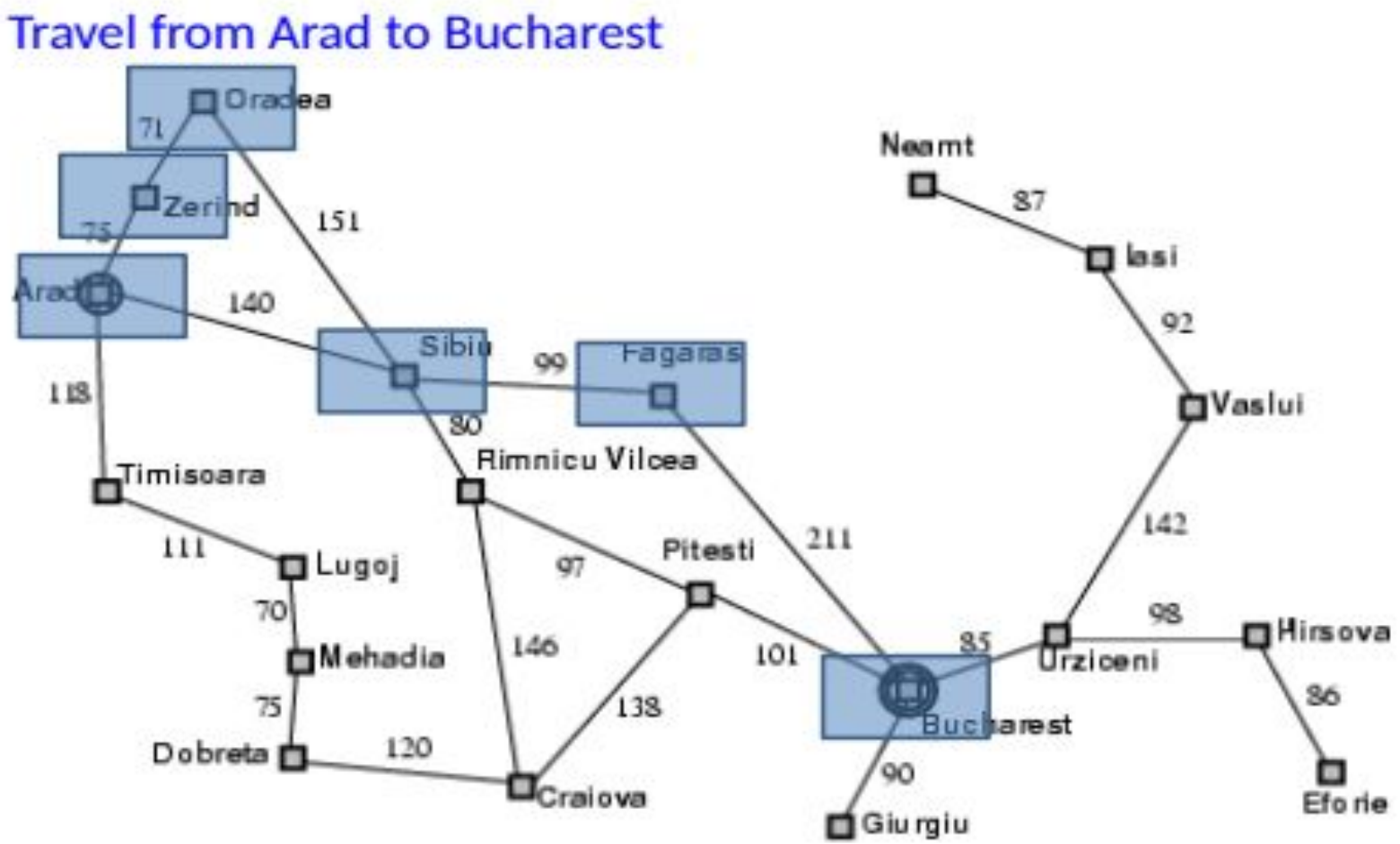
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania DFS



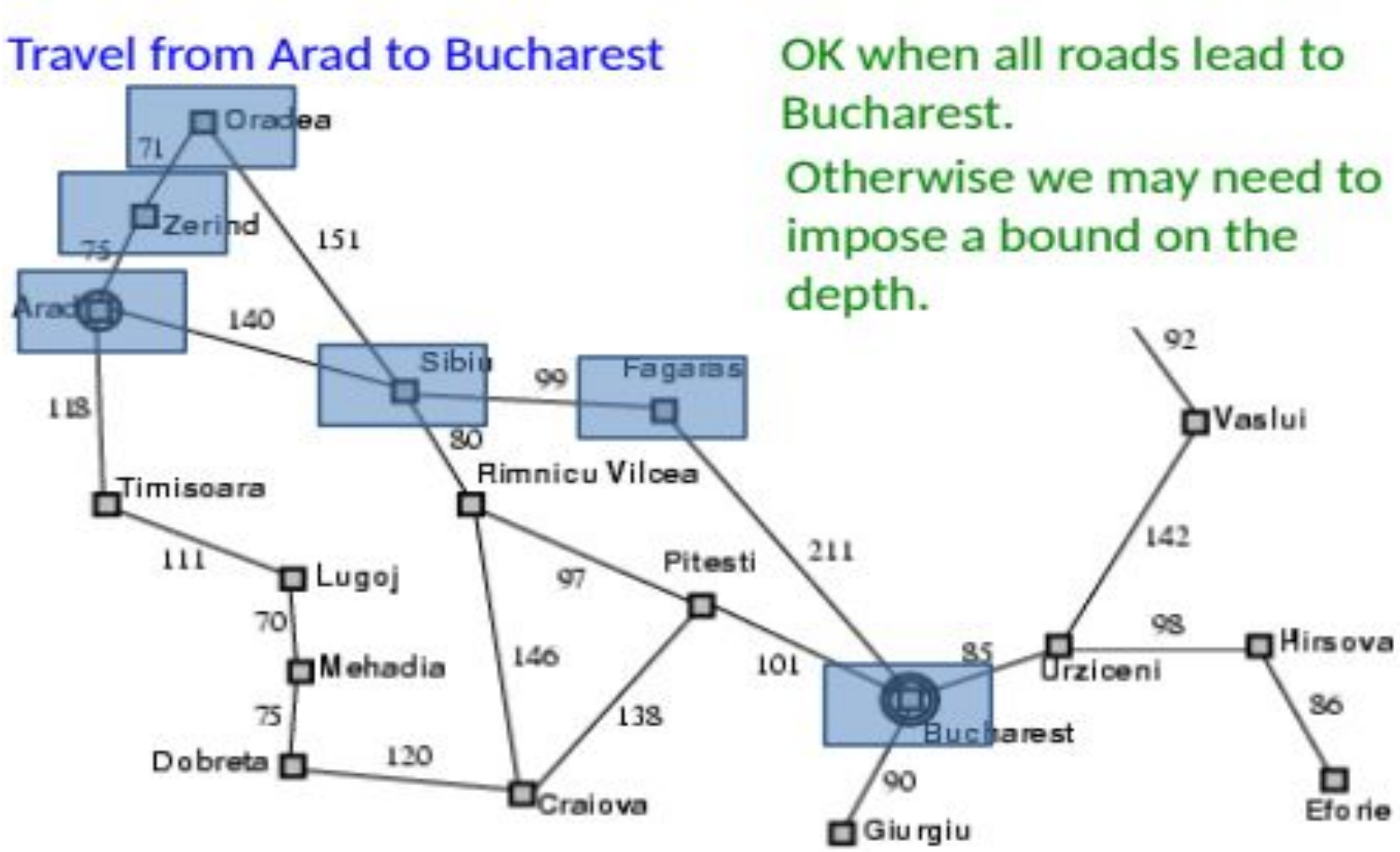
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania DFS



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania DFS



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Evaluation

- b: branching factor
- d: depth of shallowest goal node
- m: maximal depth of a leaf node
- Depth-first search is:
  - Complete only for finite search tree
  - Not optimal
- Number of nodes generated:
  $1 + b + b^2 + \ldots + b^m = O(b^m)$
- Time complexity is $O(b^m)$
- Space complexity is $O(bm)$ [or $O(m)$]

[Reminder: Breadth-first requires $O(b^d)$ time and space]

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Difference between BFS and DFS

BASIS FOR COMPARISON	BFS	DFS
Basic	Vertex-based algorithm	Edge-based algorithm
Data structure used to store the nodes	Queue	Stack
Memory consumption	Inefficient	Efficient
Structure of the constructed tree	Wide and short	Narrow and long
Traversing fashion	Oldest unvisited vertices are explored at first.	Vertices along the edge are explored in the beginning.
Optimality	Optimal for finding the shortest distance, not in cost.	Not optimal
Application	Examines bipartite graph, connected component and shortest path present in a graph.	Examines two-edge connected graph, strongly connected graph, acyclic graph and topological order.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Uniform cost search

- We factor in the cost of each step (e.g., distance form current state to the neighbors). Assumption: costs are non-negative.

  g(n) = cost so far to reach n

- Queue ordered by cost

- If all the steps are the same breadth-first search is optimal since it always expands the shallowest (least cost)!

- Uniform-cost search expand first the nodes with lowest cost (instead of depth).

Does it ring a bell?

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Uniform cost search

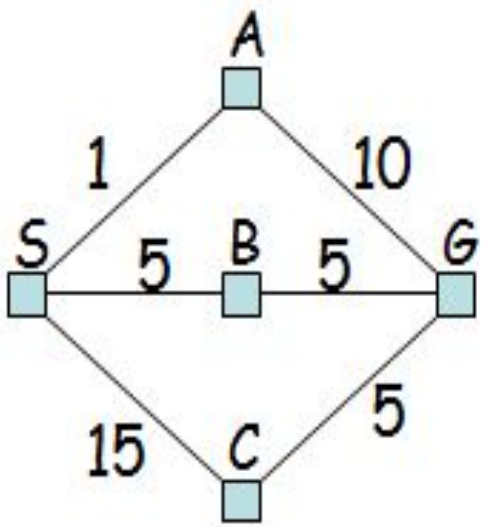
- Each arc has some cost $c \geq \varepsilon > 0$
- The cost of the path to each fringe node N is

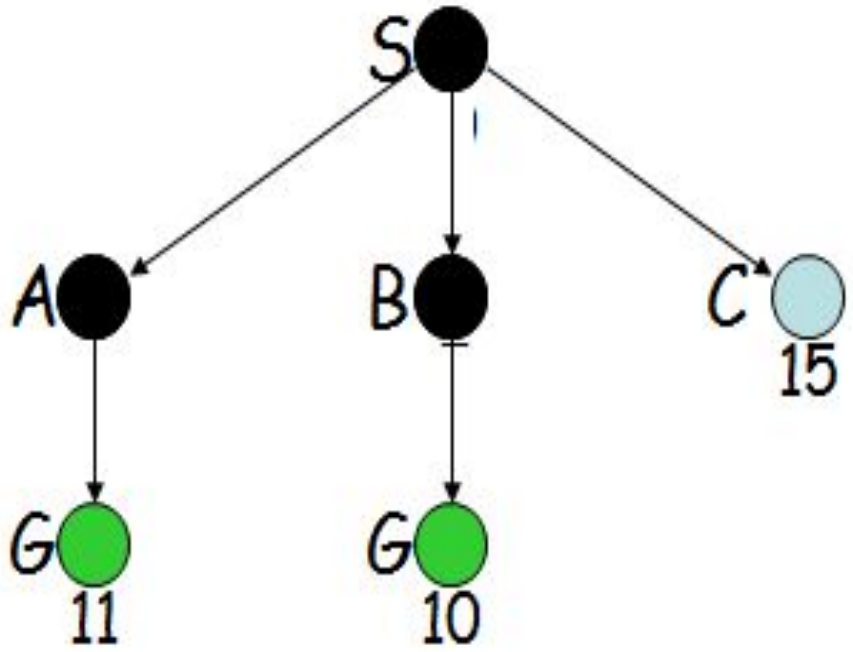
$$g(N) = \Sigma \text{ costs of arcs}$$

- The goal is to generate a solution path of minimal cost
- The queue FRINGE is sorted in increasing cost

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

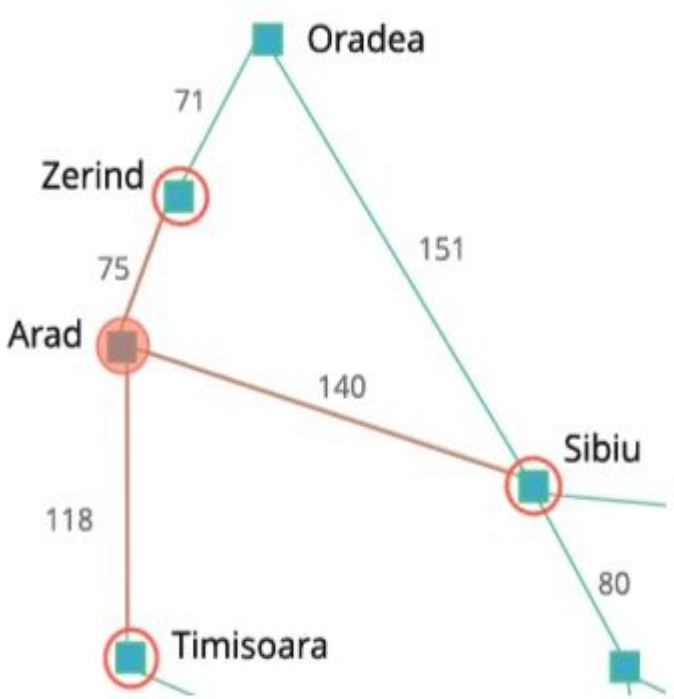
Uniform cost search (Example)

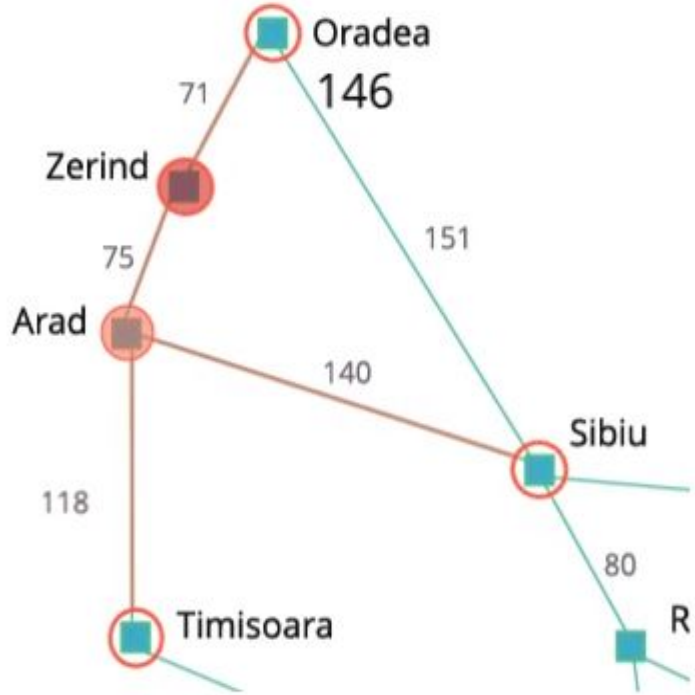




Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Uniform cost search , Also known as the Cheapest First Search, the **Uniform Cost Search** algorithm always chooses the frontier that has the cheapest path cost.





In our route-finding example, in the beginning, we have three frontiers: Zerind, Sibiu, and Timisoara. Zerind has the lowest path cost of 75, so Zerind will be chosen.

Then, we have three frontiers: Oradea, Sibiu, and Timisoara. Timisoara has the lowest path cost of 118, so we choose Timisoara.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

4.1 Introduction

4.2 Breadth first search

4.3 Depth first search

4.4 Difference between BFS and DFS

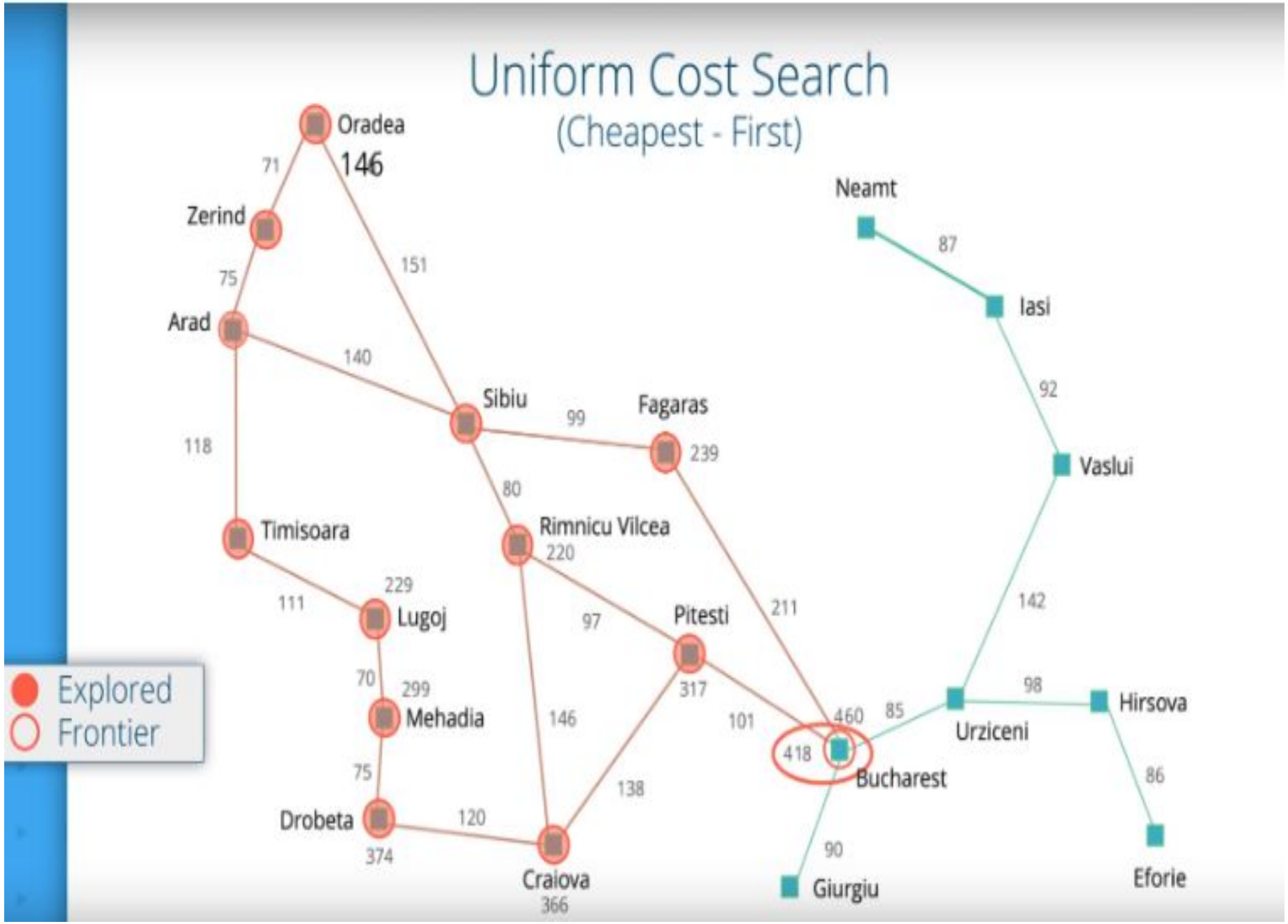
4.5 Uniform cost search

4.6 Depth-limited search

4.7 Iterative-deeping DFs

4.8 Bidirectional Search

4.9 Comparison of uninformed search



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

For Uniform Cost Search, it is essential to continue searching even when the goal state is reached. If you stop immediately after finding a path, then you cannot guarantee that it is the shortest path, since there might have been a frontier that could have reached the goal state faster.

For example, In our Arad to Bucharest example, the first path you find (Arad - Sibiu - Fagaras - Bucharest) with the uniform cost search has the cost of 460. However, there is another path (Arad - Sibiu - Rimnicu Vilcea - Pitesti - Bucharest) that as the cost of 418.

Therefore, you need to wait until all the other frontiers have a path cost higher than the path cost of your path to the goal state. This is ensured exactly when your path to the goal state is chosen from the list of frontiers, since by definition, uniform cost search chooses the frontier with least path cost.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

- **Expand least-cost (of path to) unexpanded node**
- **(e.g. useful for finding shortest path on map)**

Implementation:

- ***fringe* = queue *ordered by path cost*** g – cost of reaching a node

- Complete? Yes, if step cost ≥ ε **(>0)**
  Time? # of nodes with $g \leq$ cost of optimal solution (C^*),
  $O(b^{(1+\lfloor C^*/\varepsilon \rfloor)})$
- Space? # of nodes with $g \leq$ cost of optimal solution,
- $O(b^{(1+\lfloor C^*/\varepsilon \rfloor)})$
- Optimal? Yes – nodes expanded in increasing order of $g(n)$
- *Note: Some subtleties (e.g. checking for goal state).*

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth-limited search

- Depth-first with depth cutoff k (depth below which nodes are not expanded)

- Three possible outcomes:
  - Solution
  - Failure (no solution)
  - Cutoff (no solution within cutoff)

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth-limited search

- The problem related to the unbounded tree can be alleviated by providing the DFS with the predetermined depth limit of l.
- The nodes at depth l are considered to have no successors. This approach is known as *depth-limited search*.
- This search solves the infinite path problem, however, addition source of incompleteness is introduced if we choose l less than d, i.e., shallowest goal is beyond the depth limit (this is not likely in case of d being unknown).
- This search will also remain non-optimal if l selected is great than d.
- It has the time complexity of $O(bl)$ and its space complexity is $O(bl)$. DFS can be seen as the special case of depth-limited search with $l = \acute{E}$.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth-limited search

Pseudo-code

```
function DEPTH-LIMITED-SEARCH(problem, limit) returns a solution, or failure/cut-off
return RECURSIVE-DLS (MAKE NODE(INITIAL-STATE[problem]), problem, limit)
function RECURSIVE-DLS) (node, problem, limit) then returns a solution, of failure/cut-off cut-
        off_occured?←false
```

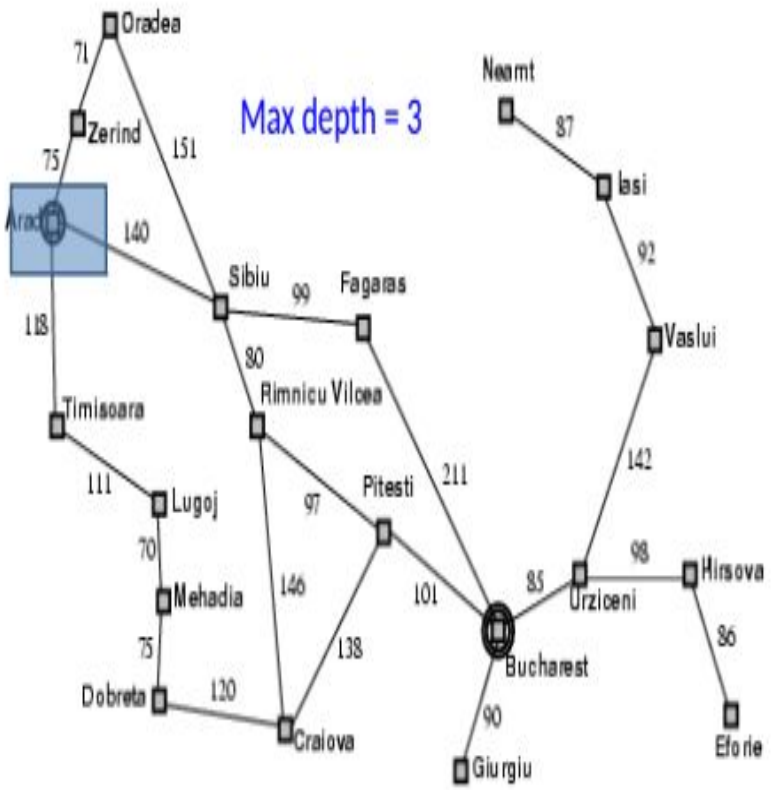
```
if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
else if DEPT[node]=limit then return cut-off
else for each successor in EXPAND(node, problem) do
result ← RECURSIVE-DLS(successor, problem, limit)
if result = cut-off then cut-off_occurred? ← true
else if result ≠ failure then return result
if cut-off_occured? then return cut-off else return failure
```

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road,
Daryaganj, New Delhi-110002

Depth Limited Search

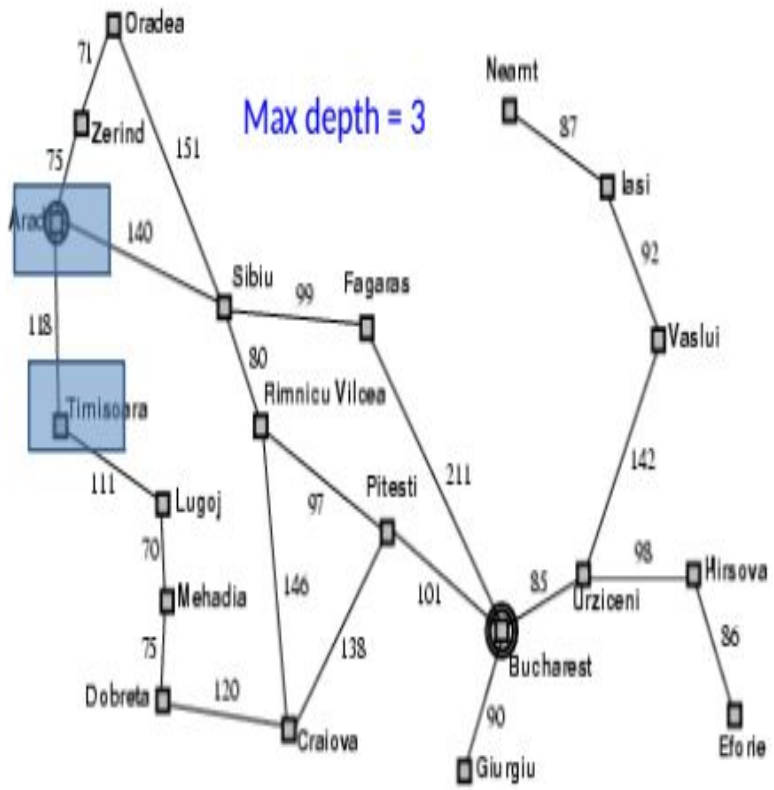
Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.



Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.

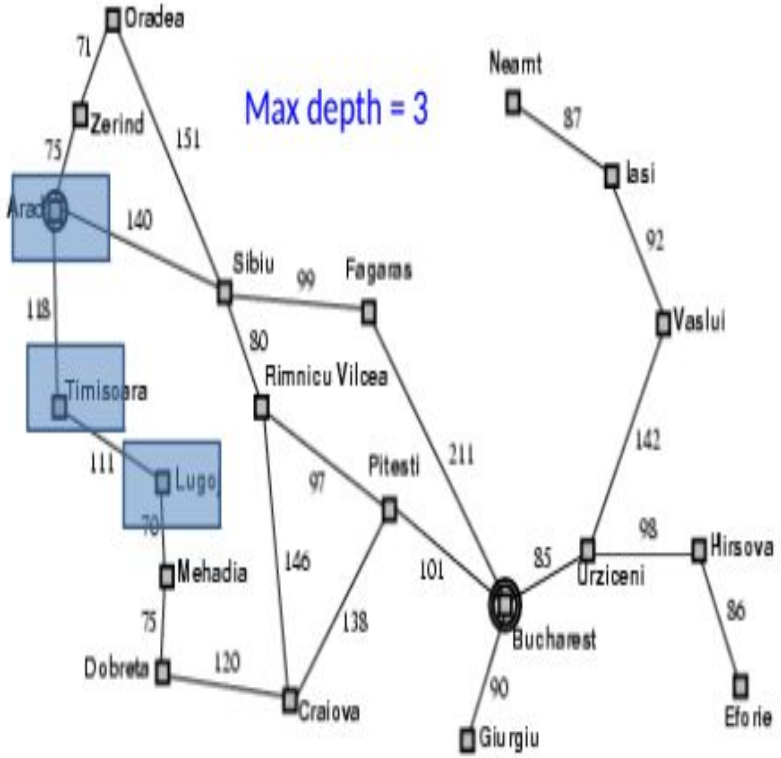


Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth Limited Search

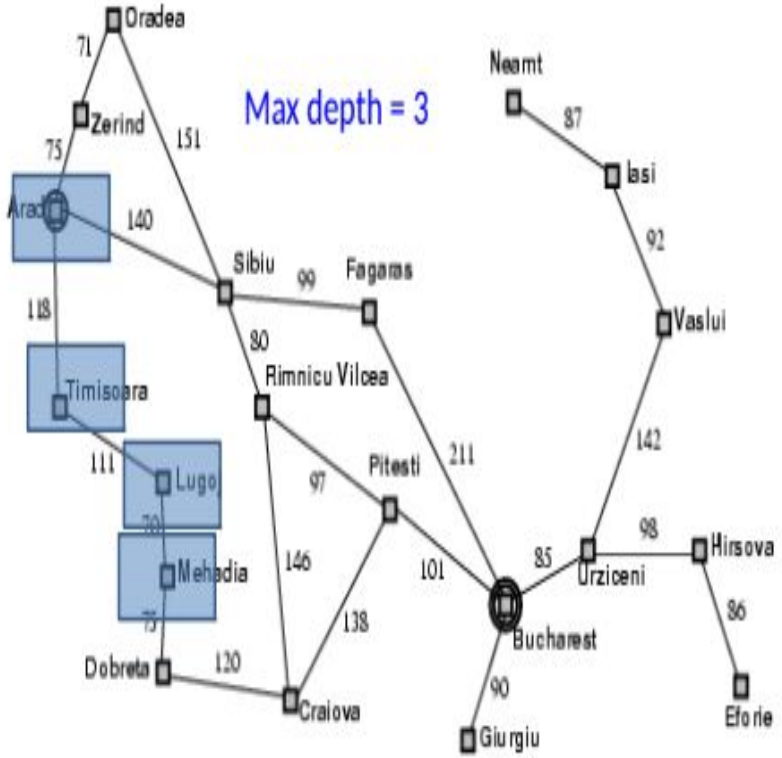
Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.



Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.

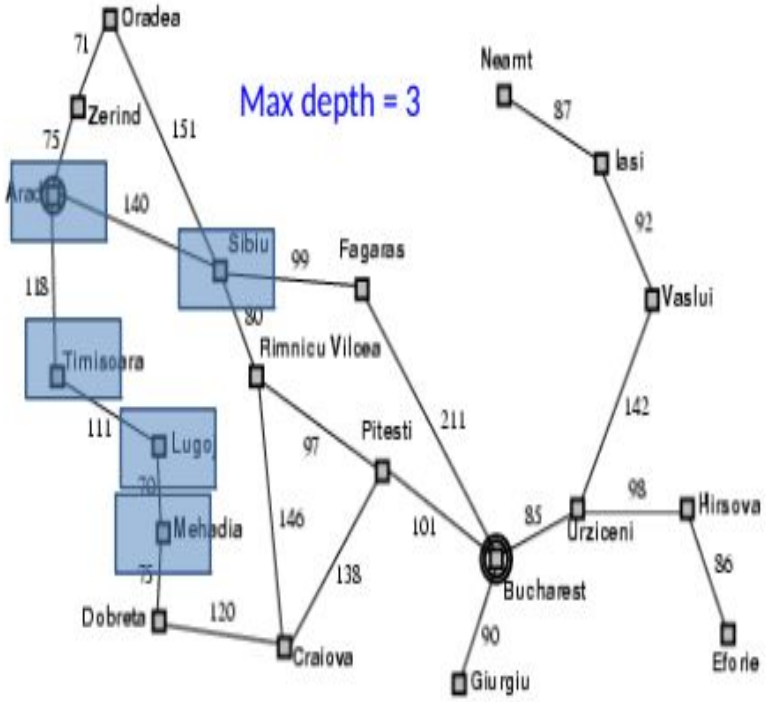


Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth Limited Search

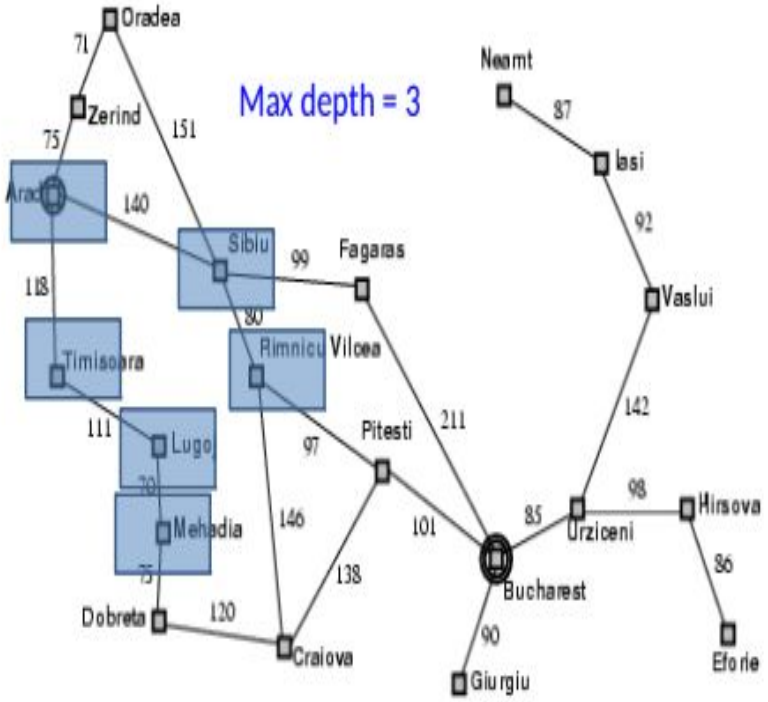
Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.



Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.

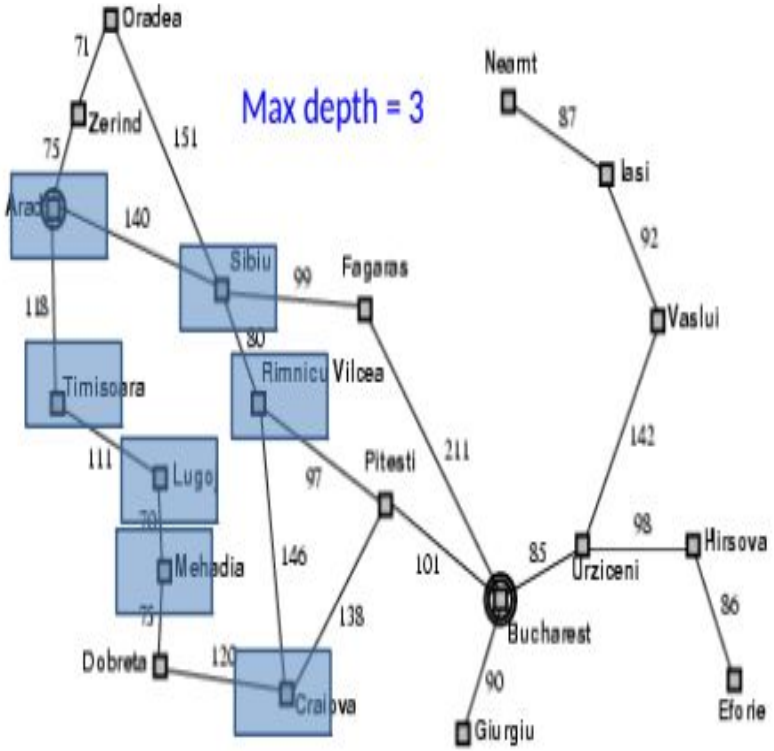


Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth Limited Search

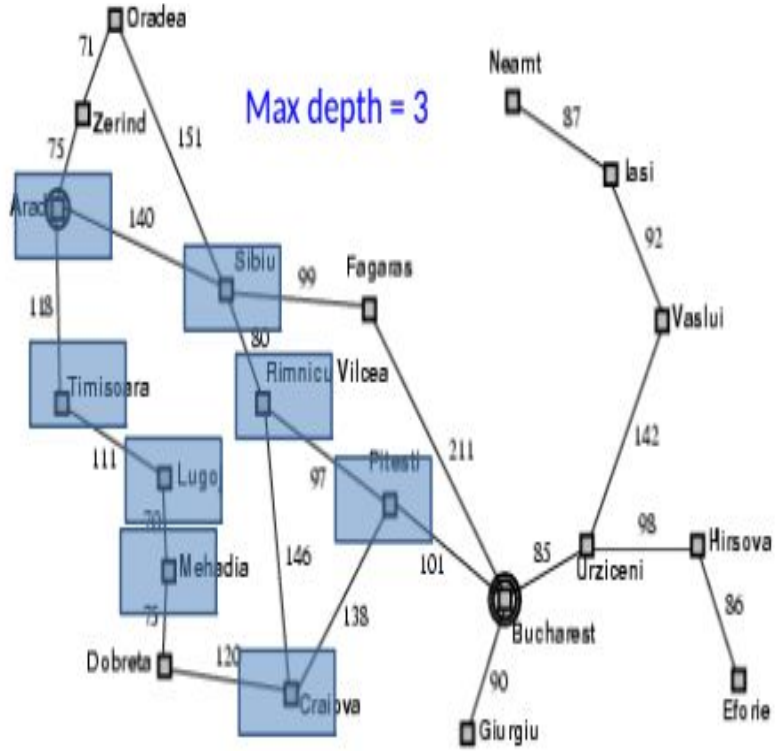
Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.



Example: Romania Problem

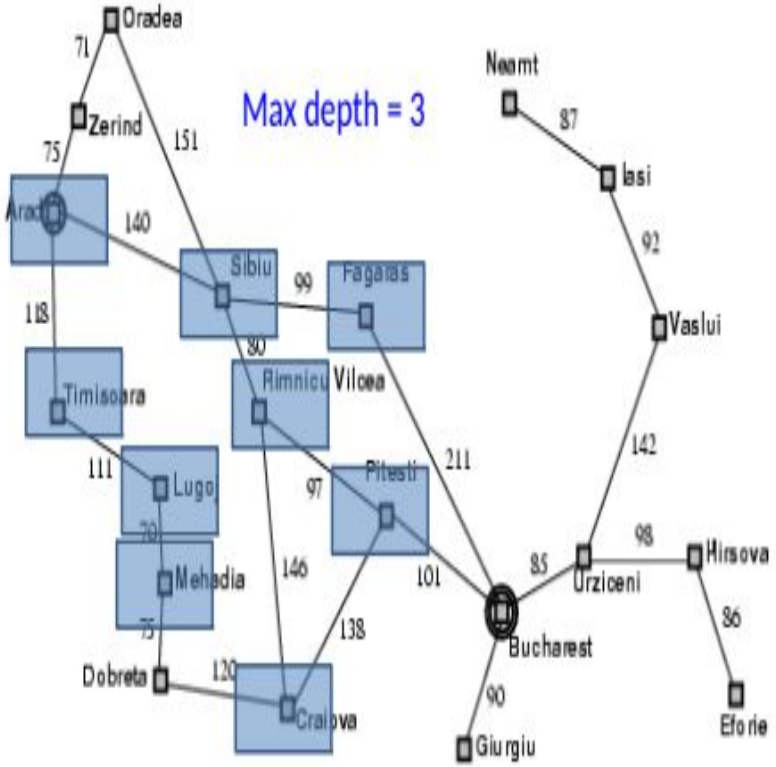
Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

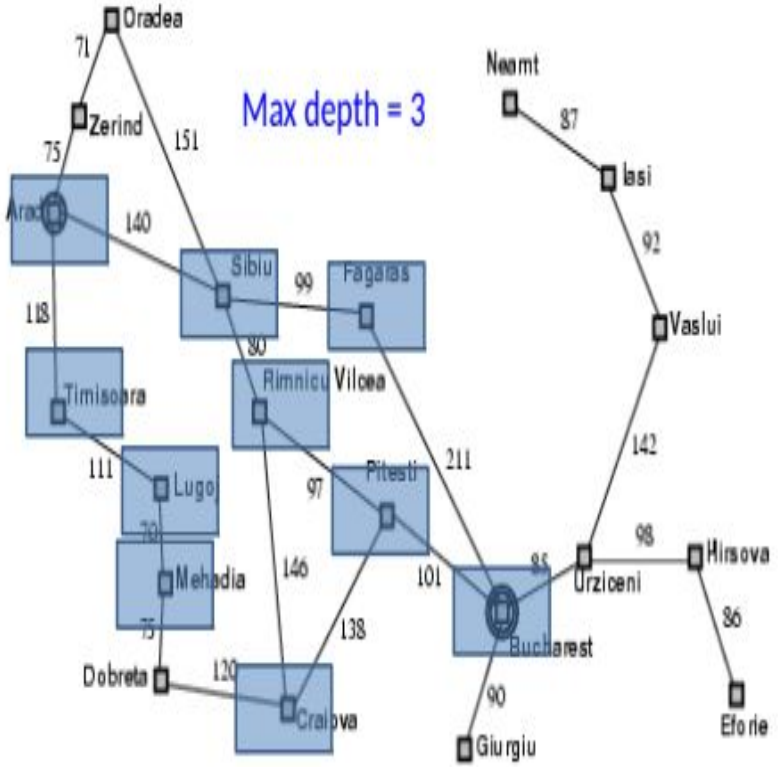
Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.



Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.

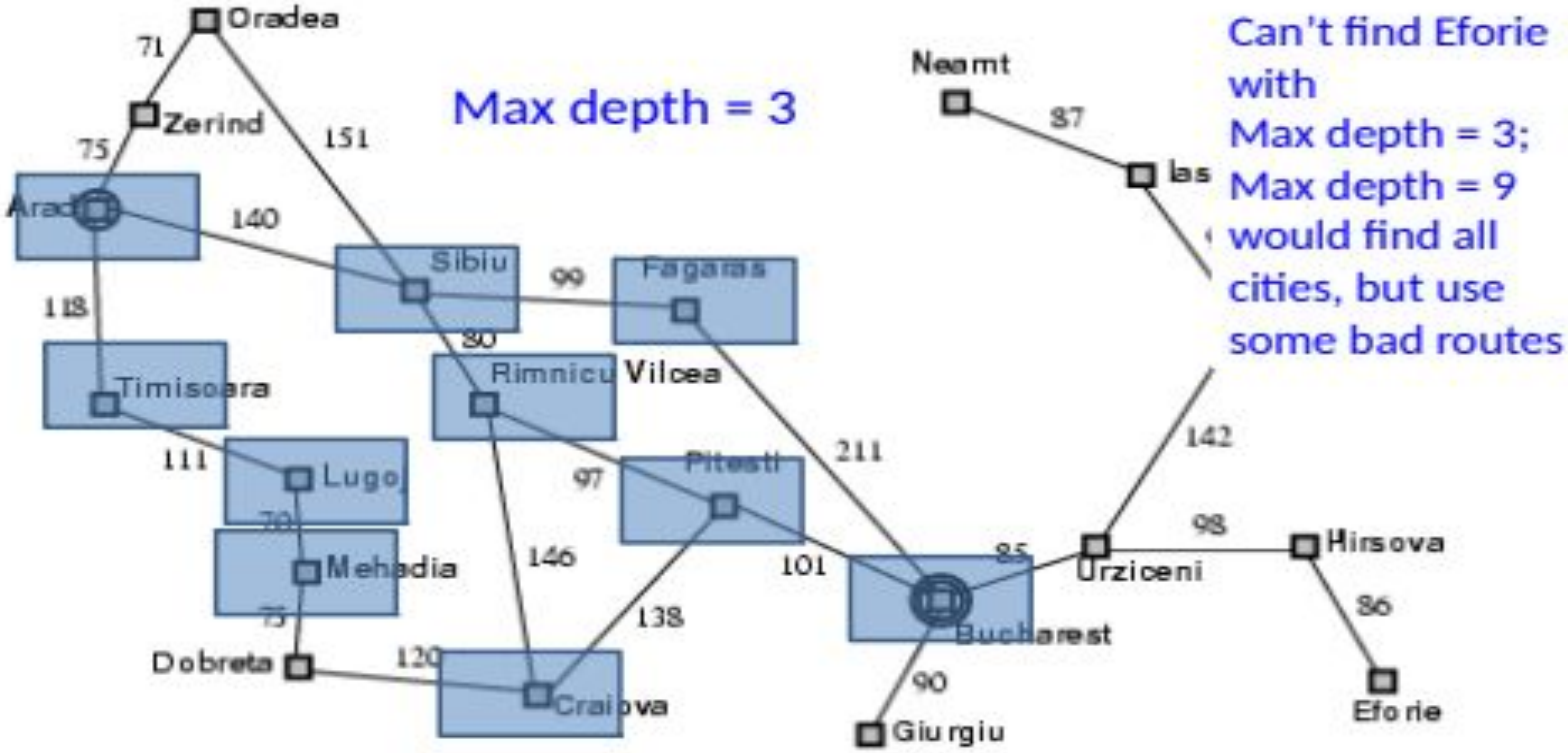


Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Depth Limited Search

Example: Romania Problem

Only 20 cities on the map, so no path longer than 19.
In fact, any city can reach any other in at most 9 steps.



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Iterative deeping DFs

- Iterative deepening search (or iterative deepening first search) is the general strategy. This is most often used in combination with DFS which finds the best depth limit. It is the depth limited search in iteration. It does this by gradually the limit from 0, then to 1 and then to 2 and so on. This happens when the depth limit reaches *d* that is the depth of the shallowest goal node.
- Iterative deepening DFS combines the benefits of DFS and BFS. As compared to DFS, its memory requirements are modest $O(bd)$.
- In iterative deepening DFS, the nodes on the bottom level (depth *d*) are generated once, the nodes on the next level are generated twice and it continues up to the children of root which are generated *d* times. Therefore, the total number of nodes generated is
- $N(IDS) = d(b) + (d - 1)b2 + \ldots + (1)bd$

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Iterative deeping DFs

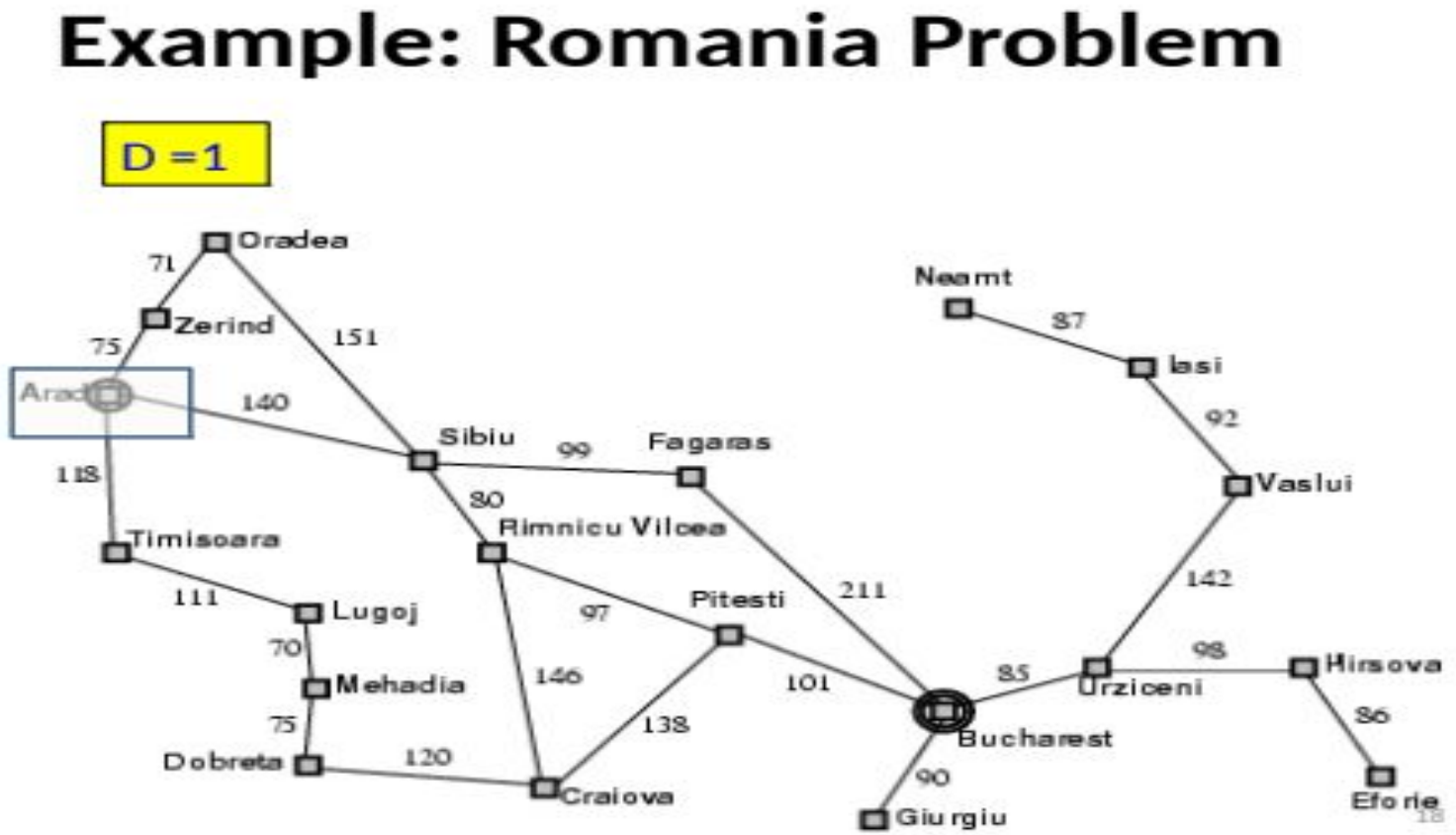
- This provides the time complexity of $O(bd)$. This can be compared with the nodes generated by BFS:
- $N(\text{BFS}) = b + b2 + \ldots + bd + (bd + 1 - b)$
- Observe that BFS generates nodes at depth, $d + 1$. Iterative deepening DFS does not. Hence, iterative deepening DFS is faster than the BFS in spite of repeated generation of states.

ALGORITHM

```
function ITERATIVE-DEEPENING-SEARCH (problem) returns a solution, or failure
    inputs: problem, a problem
    for depth ← 0 to É do
        result ← DEPTH-LIMITED-SEARCH (problem, depth)
        if result ≠ cut-off then return result
```

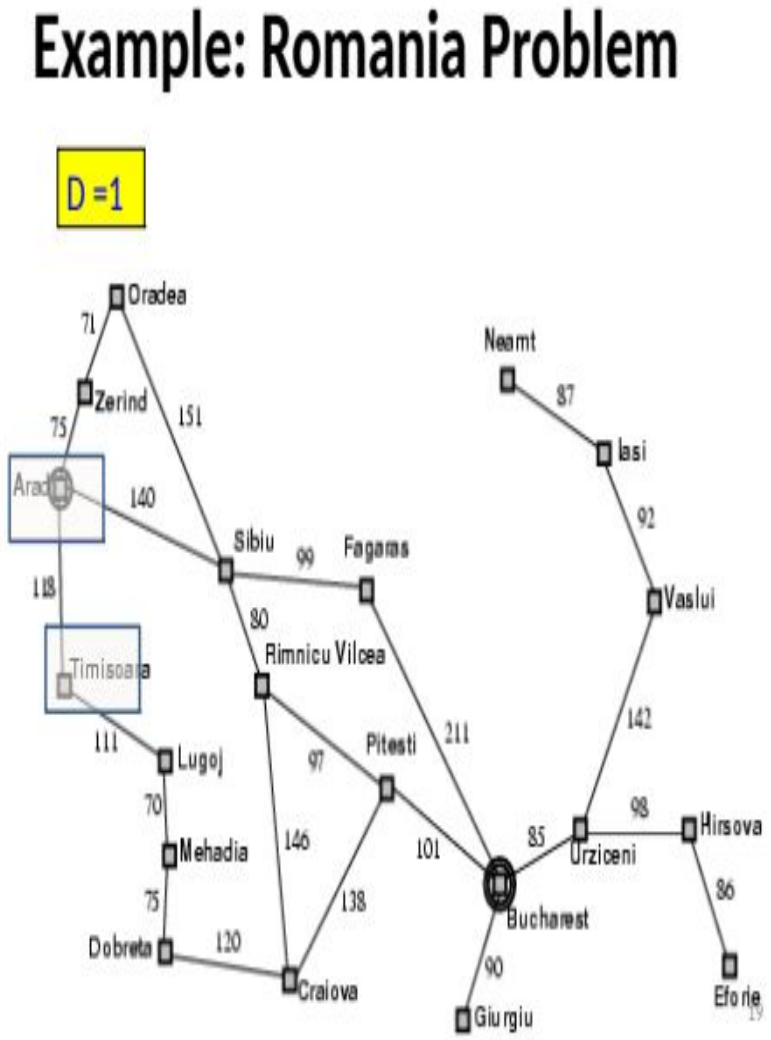
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

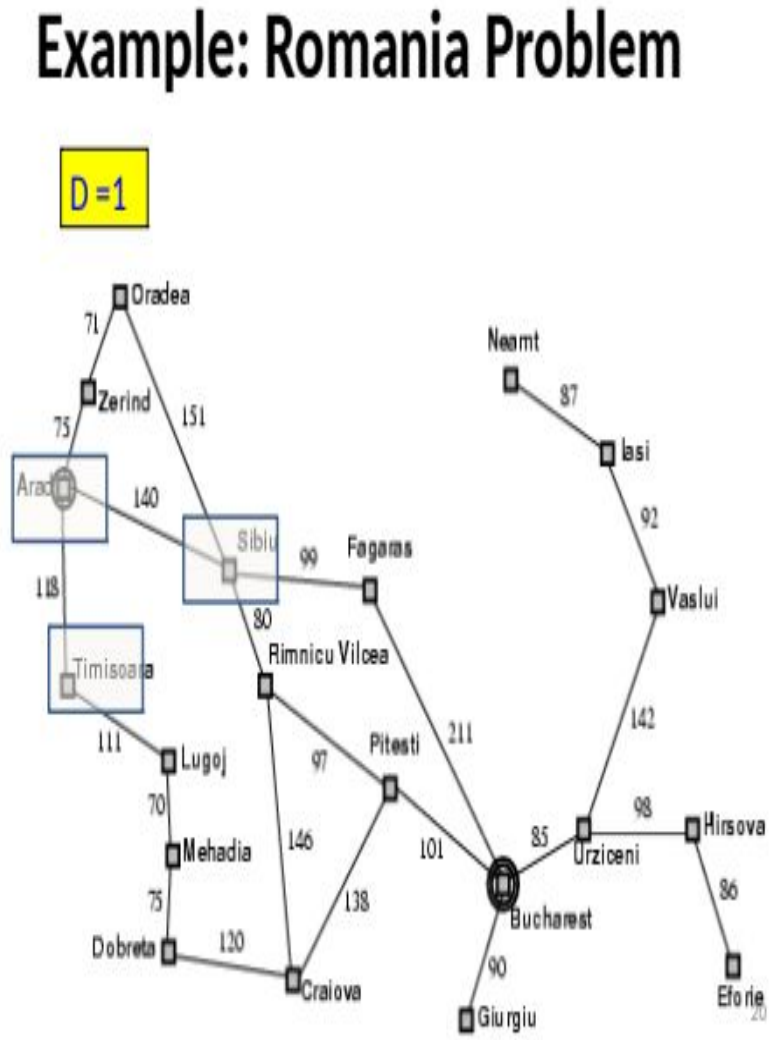
Iterative-deeping DFs



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

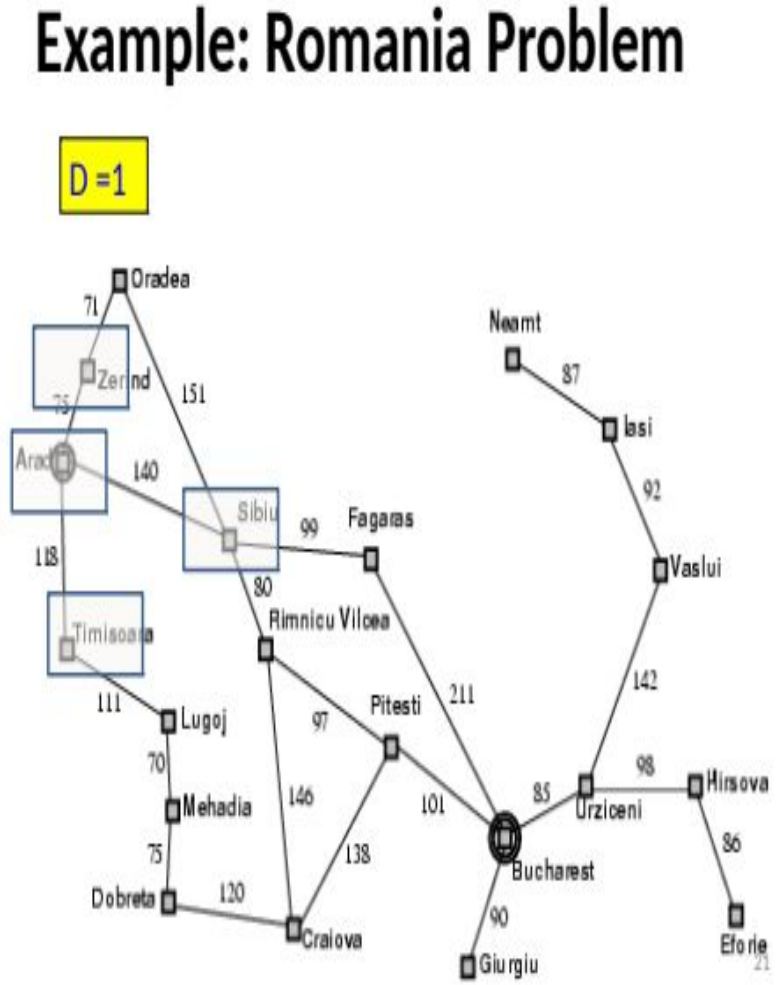
Iterative-deeping DFs

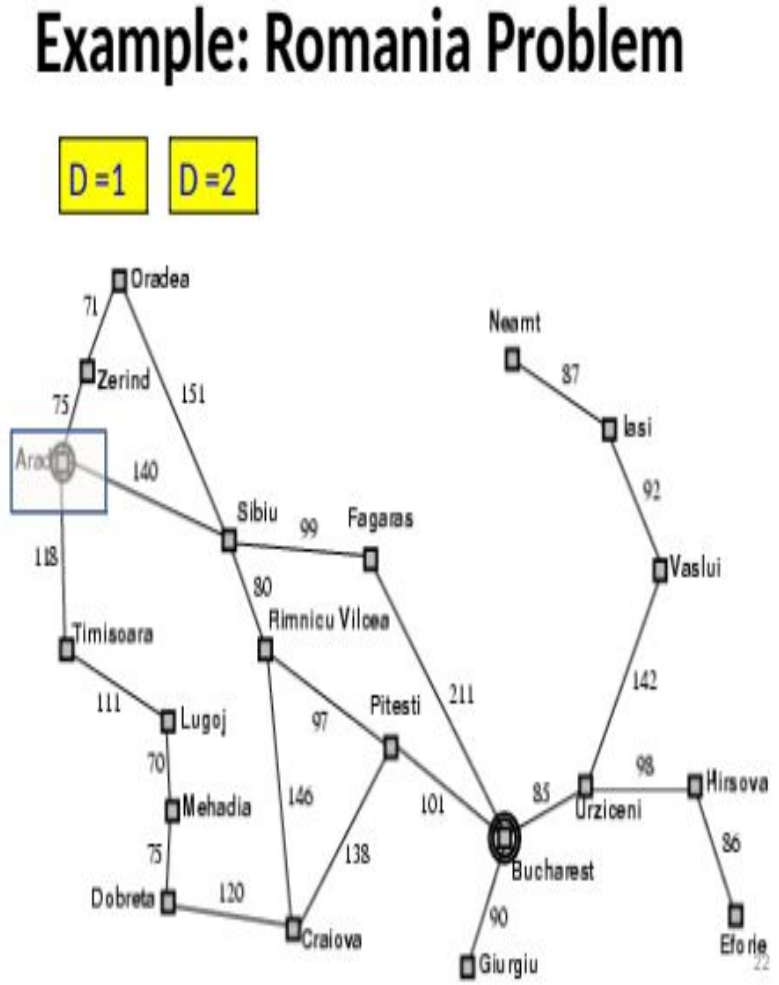




Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

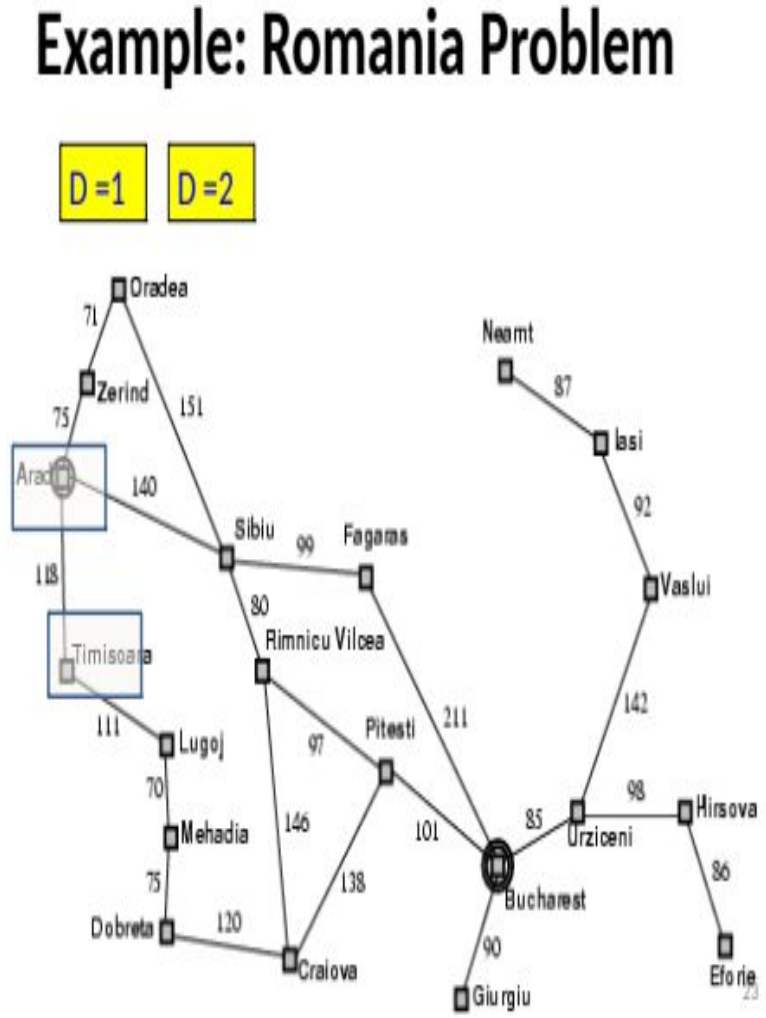
Iterative-deeping DFs

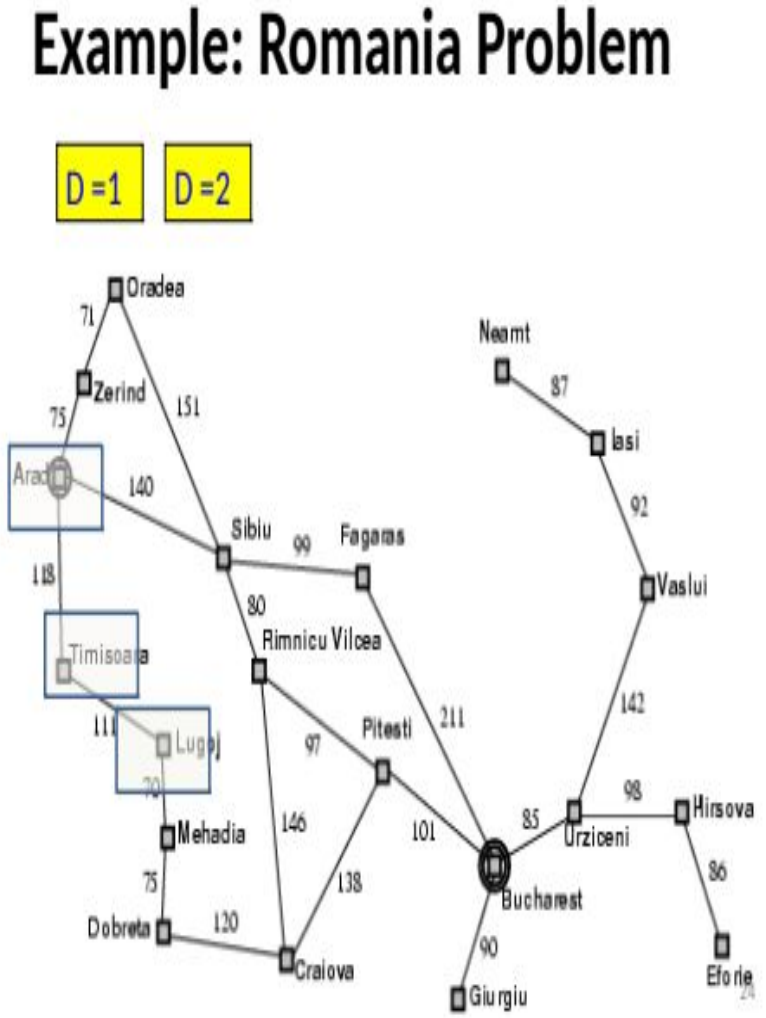




Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

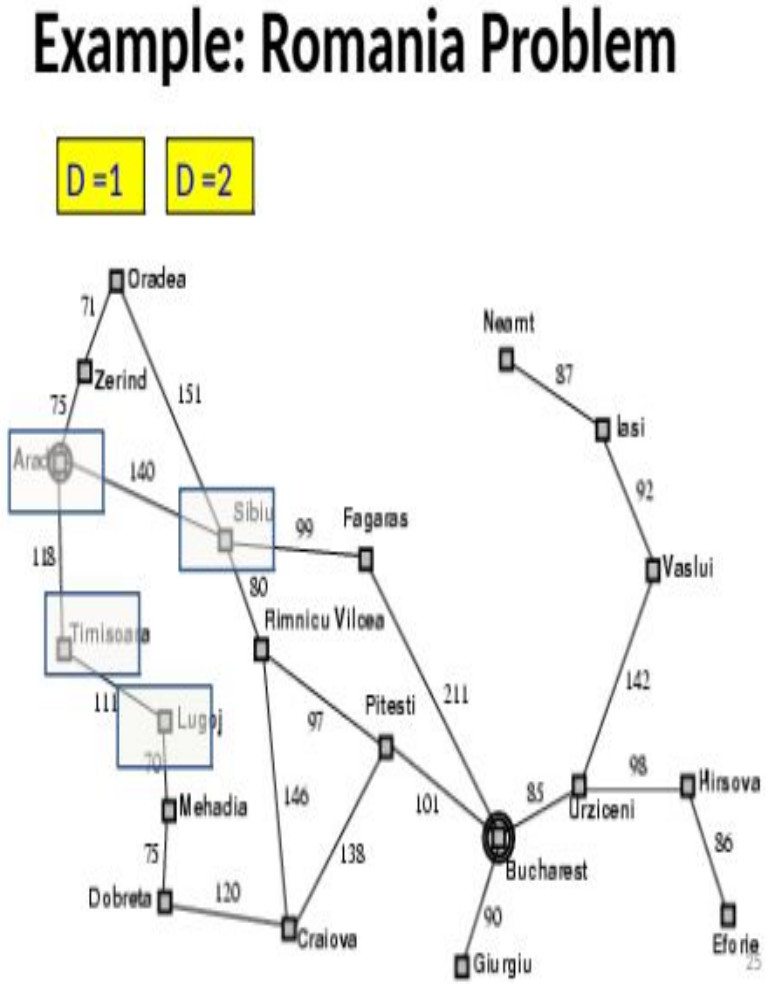
Iterative-deeping DFs

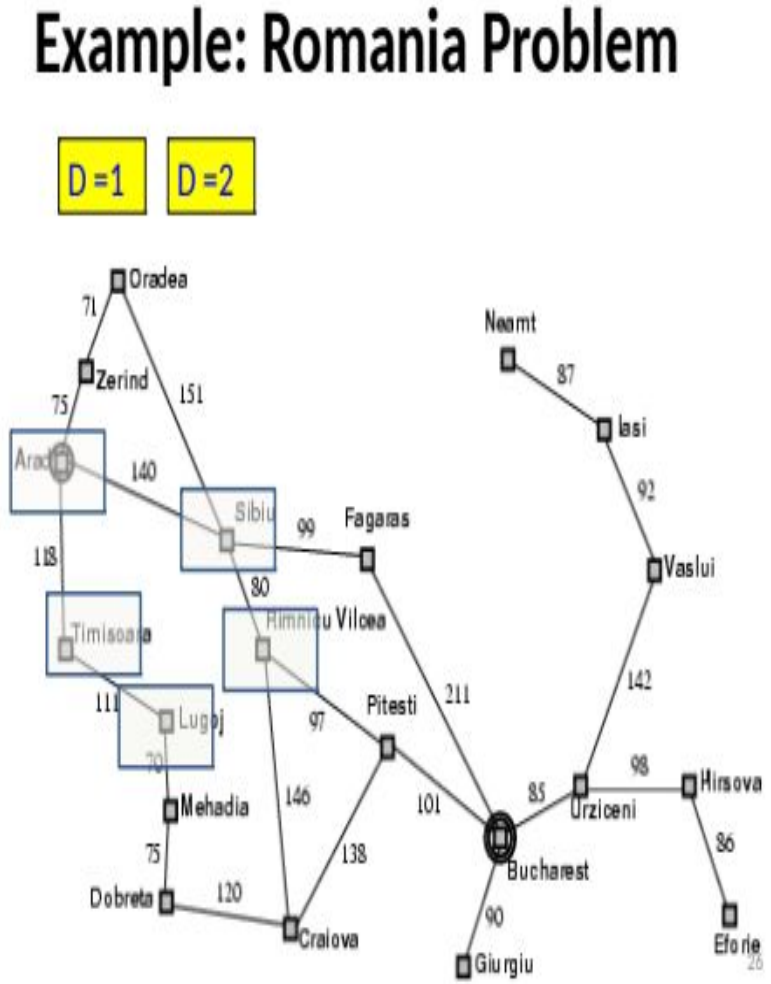




Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Iterative-deeping DFs

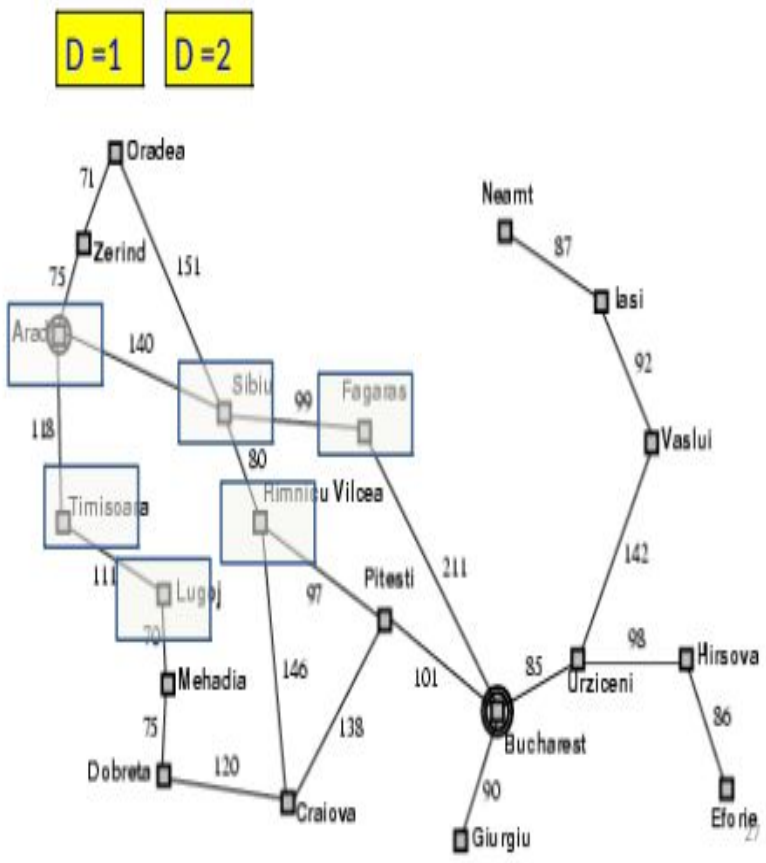




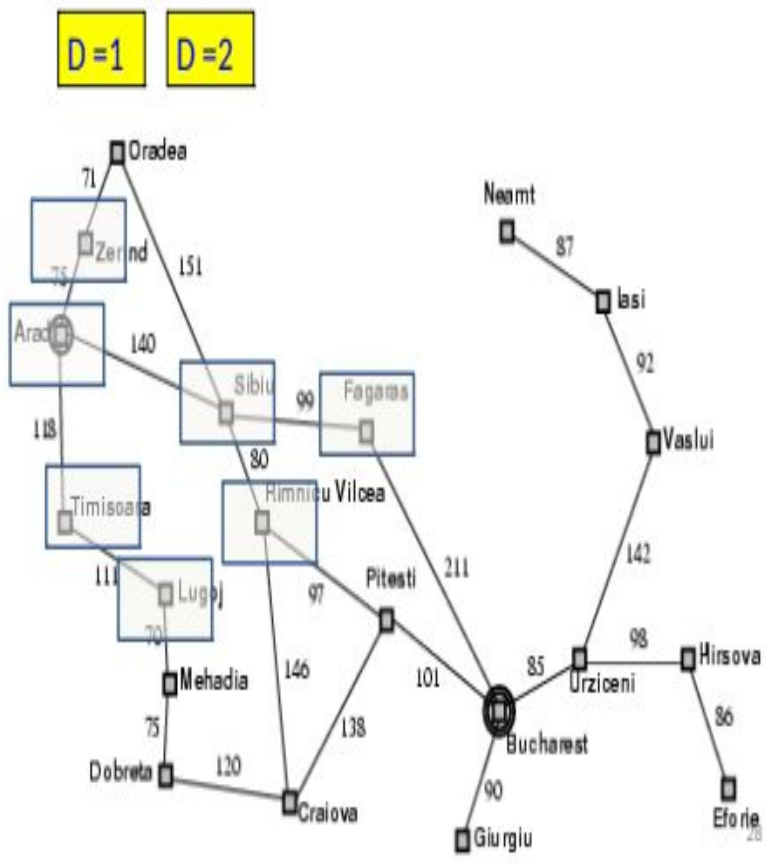
Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Iterative-deeping DFs

Example: Romania Problem

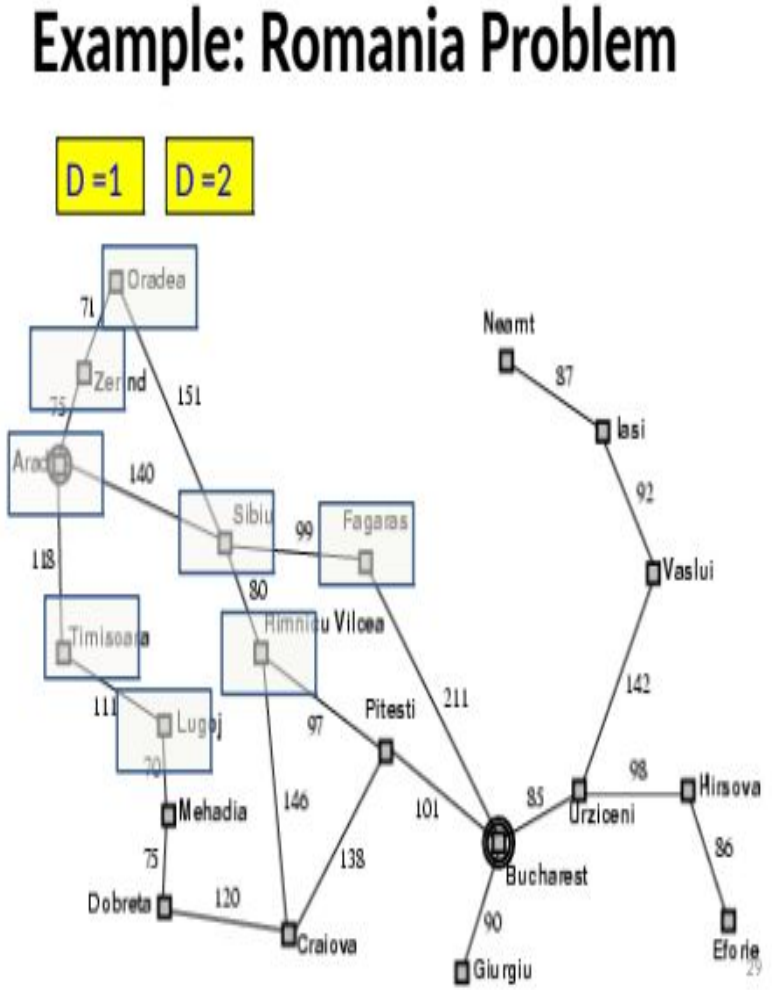


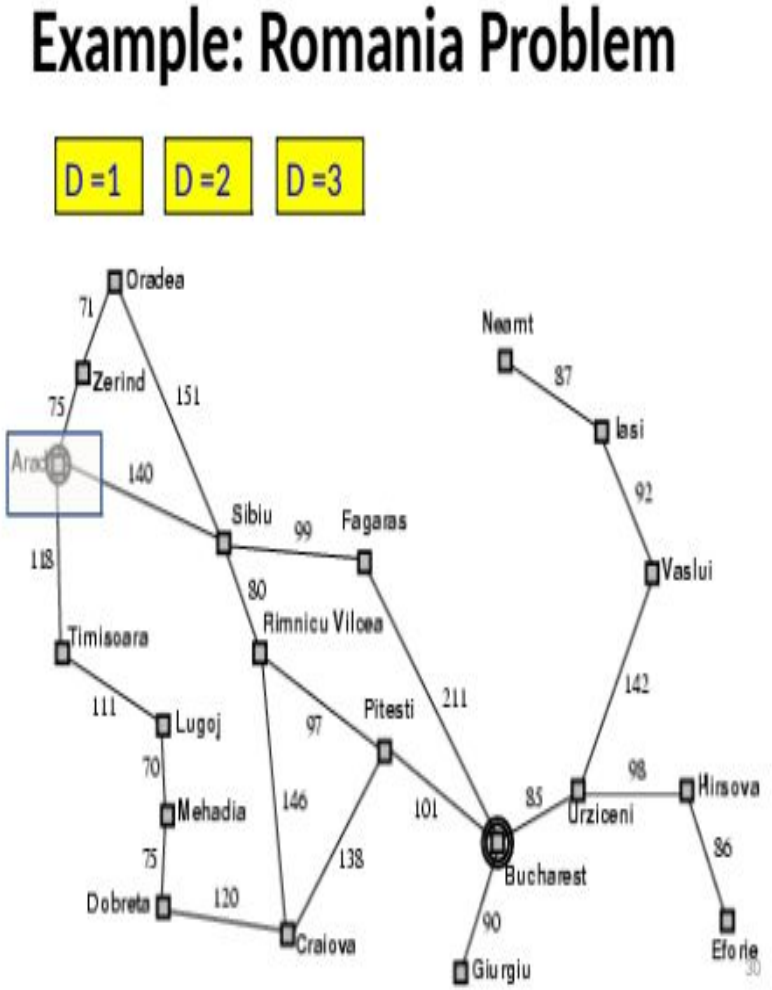
Example: Romania Problem



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

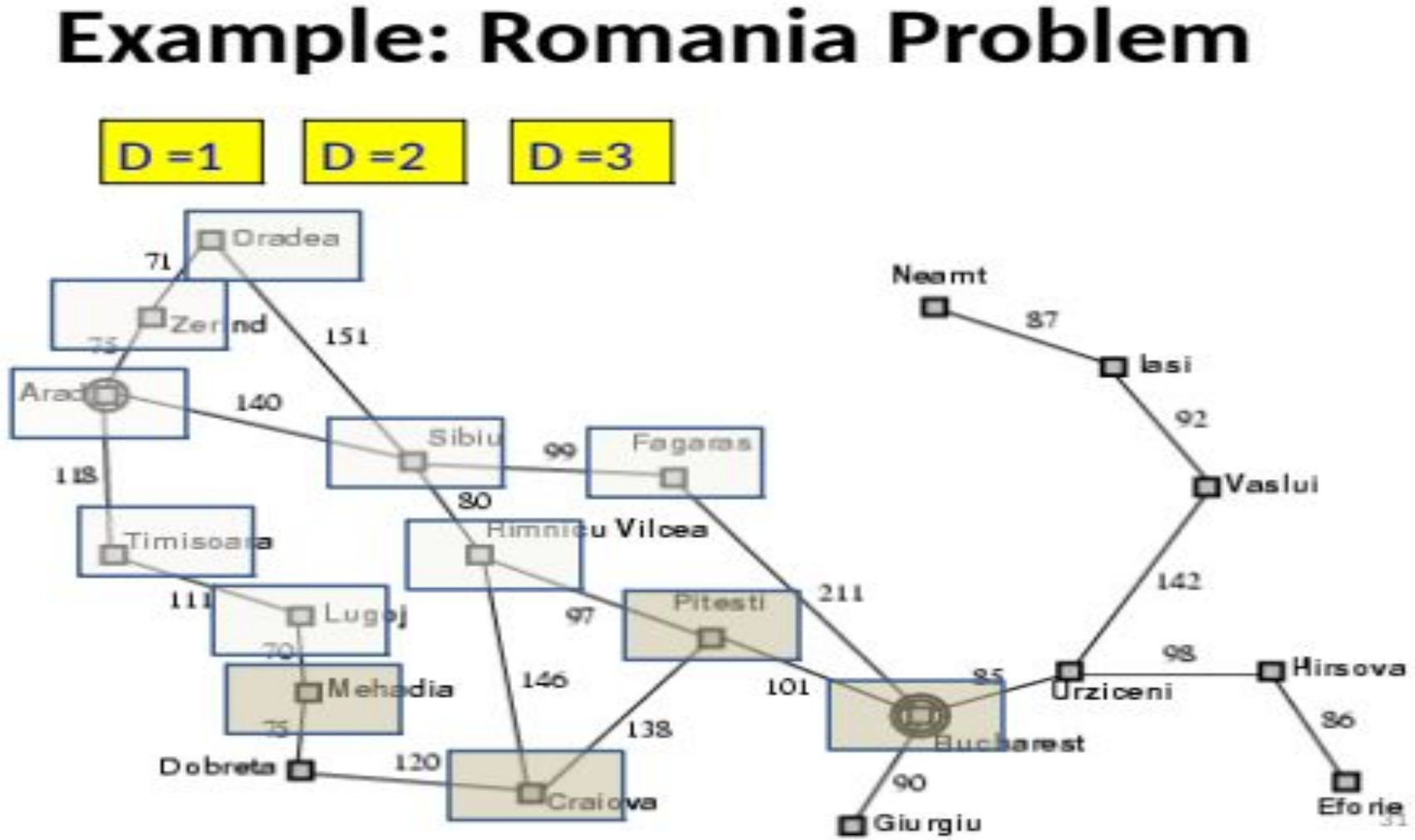
Iterative-deeping DFs





Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Iterative-deeping DFs



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

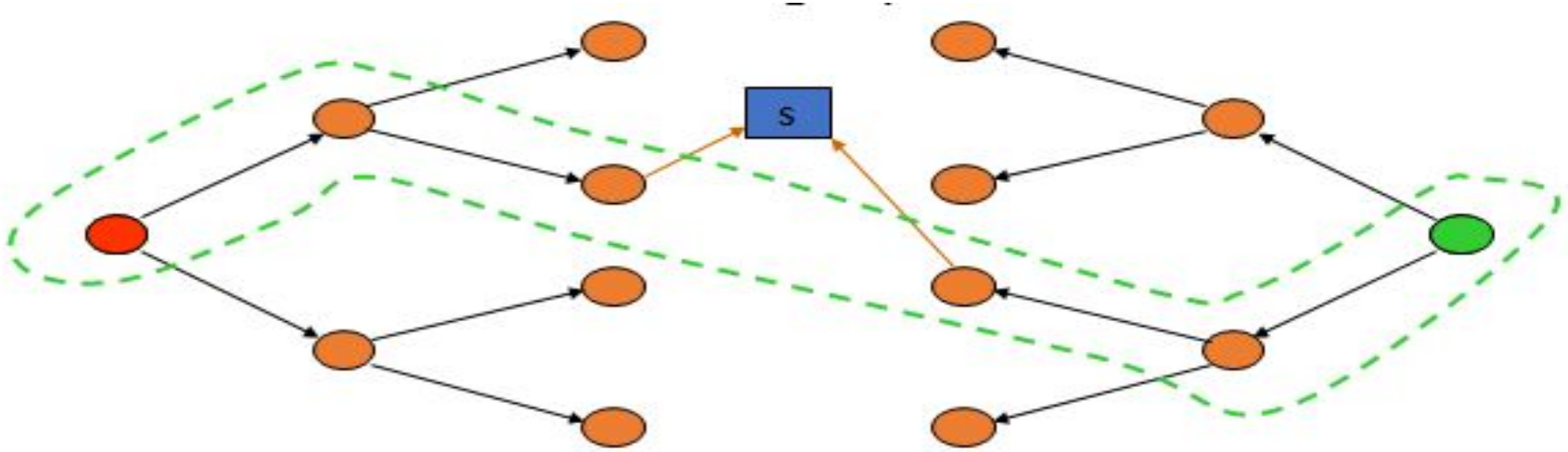
Bidirectional Search

- Search forward from the start state and backward from the goal state simultaneously and stop when the two searches meet in the middle.
- If branching factor=b, and solution at depth d, then $O(2b^{d/2})$ steps.
- B=10, d=6 then BFS needs 1,111,111 nodes and bidirectional needs only 2,222.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Bidirectional Strategy

2 fringe queues: FRINGE1 and FRINGE2



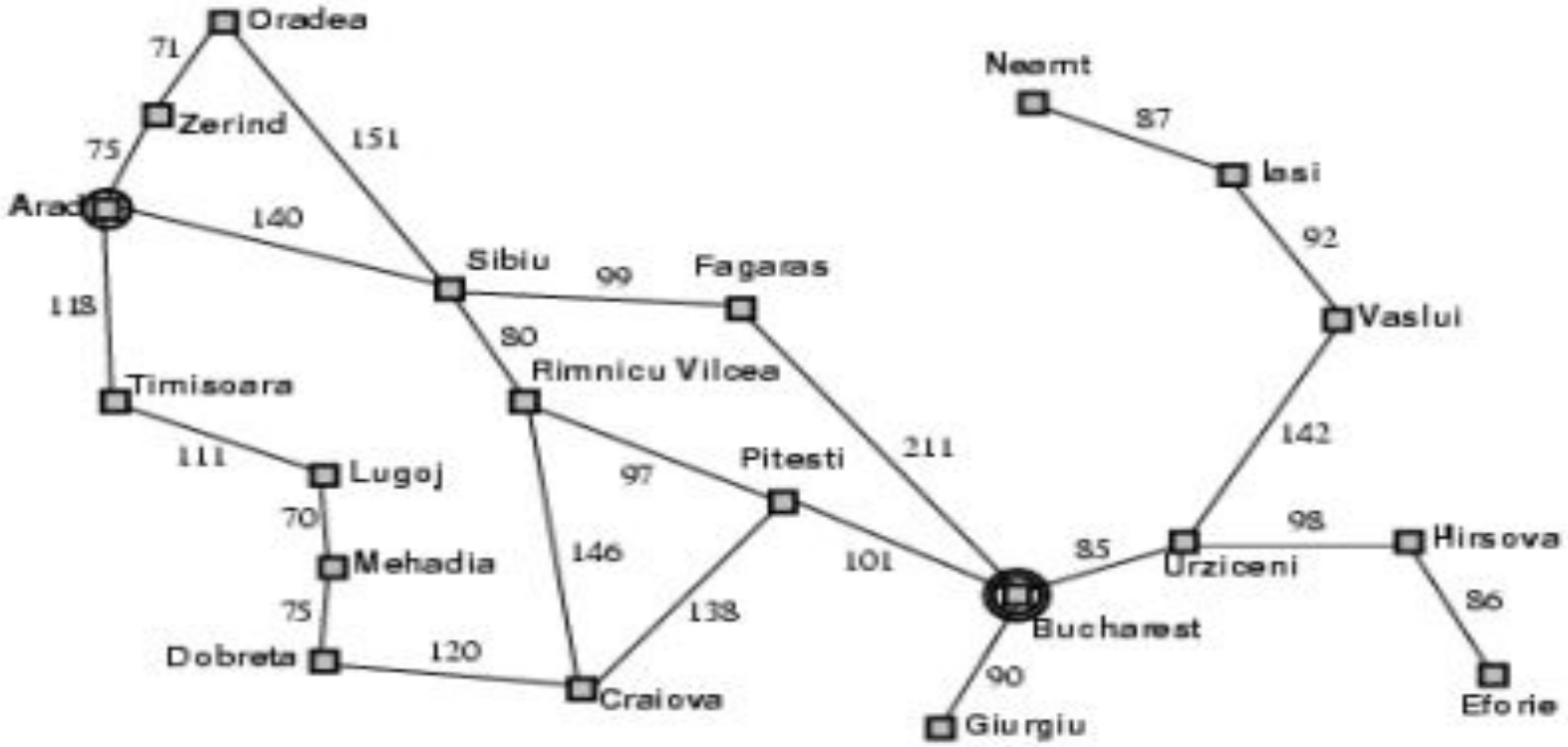
Time and space complexity is $O(b^{d/2}) << O(b^d)$
if both trees have the same branching factor b

Question: What happens if the branching factor is different in each direction?

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Example: Romania

- On holiday in Romania; currently in Arad
- Flight leaves tomorrow from Bucharest



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

4.1 Introduction

4.2 Breadth first search

4.3 Depth first search

4.4 Difference between BFS and DFS

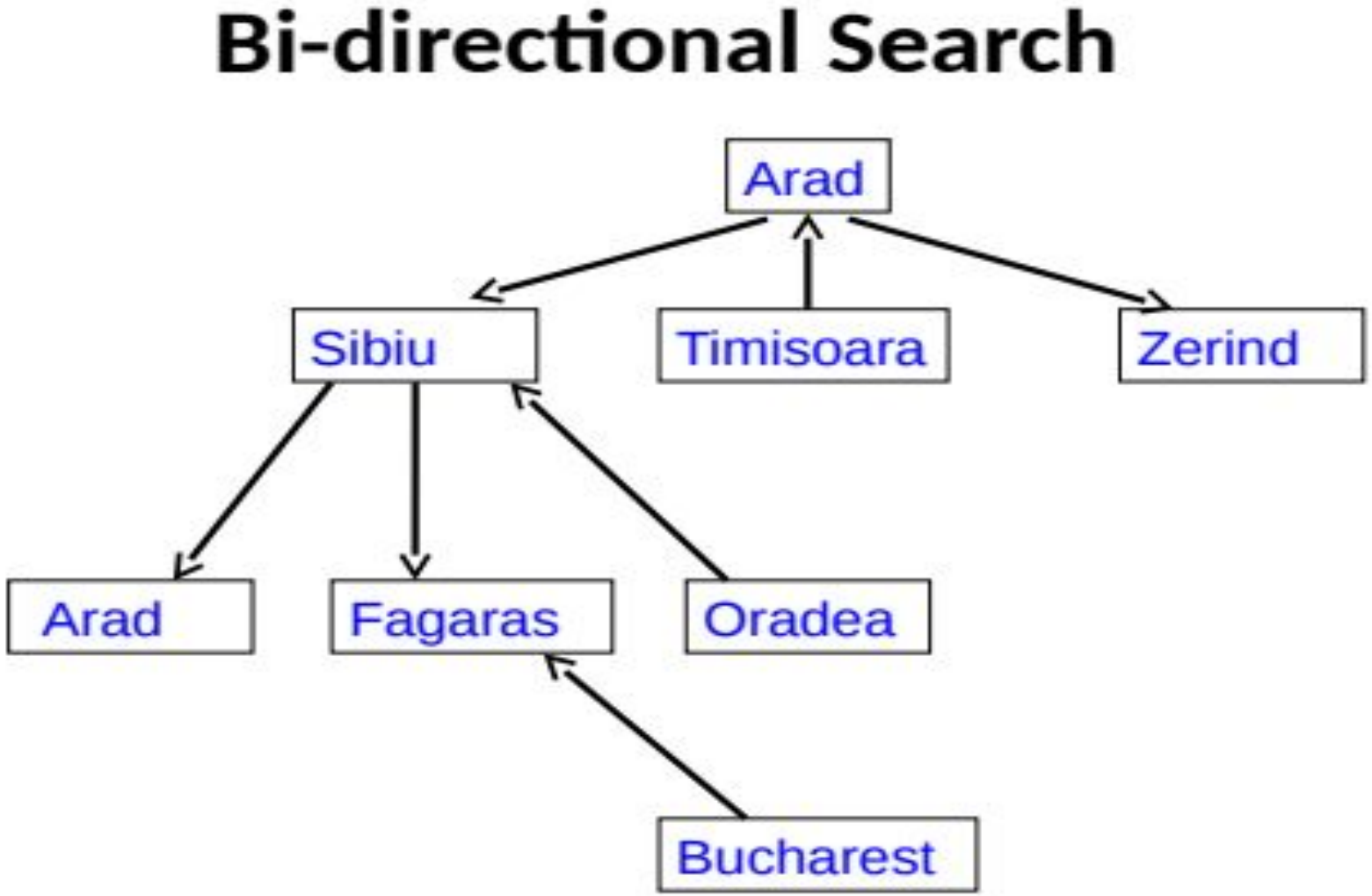
4.5 Uniform cost search

4.6 Depth-limited search

4.7 Iterative-deeping DFs

4.8 Bidirectional Search

4.9 Comparison of uninformed search



Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002

Comparison of ununiformed search

Criterion	*BFS*	*UCS*	*DFS*	*DLS*	*Iterative deepening*	*Bidirectional*
Completeness	Complete	Complete	Incomplete	Incomplete	Complete	Complete
Time Complexity (running time)	$O(b^{d+1})$	$O(b^{(C*/e)})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Space Complexity (memory)	$O(b^{d+1})$	$O(b^{(C*/e)})$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$
Optimality	Optimal	Optimal	Non-optimal	Non-optimal	Optimal	Optimal

Here, in the table,

b – Branching factor of each node (how many branches are created from each node of the search tree).

d – Depth of the shallowest solution (the depth of the place where the GOAL node is located form the root node).

m – Maximum depth of the search tree.

l – Limited depth of the search tree used in depth-limited search. C* – Cost of the optimal solution (based on the path cost).

E – Cost of every action.

Copyright © 2019 by Wiley India Pvt. Ltd., 4436/7, Ansari Road, Daryaganj, New Delhi-110002