Search Methods for Problem Solving

Asst.Prof.Dr.Emre Özbilge

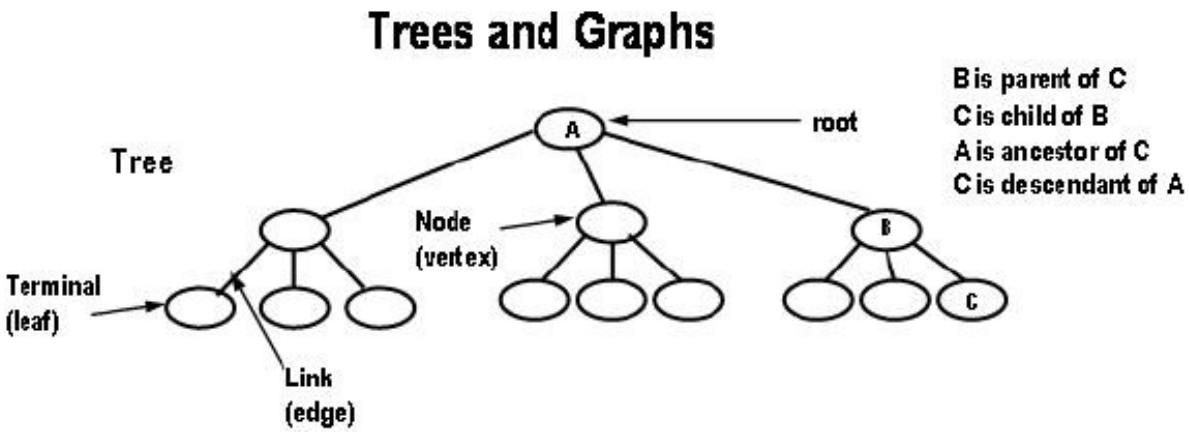
Department of Artificial Intelligence Engineering
Faculty of Engineering
Cyprus International University

October 30, 2024

CYPRUS INTERNATIONAL UNIVERSITY

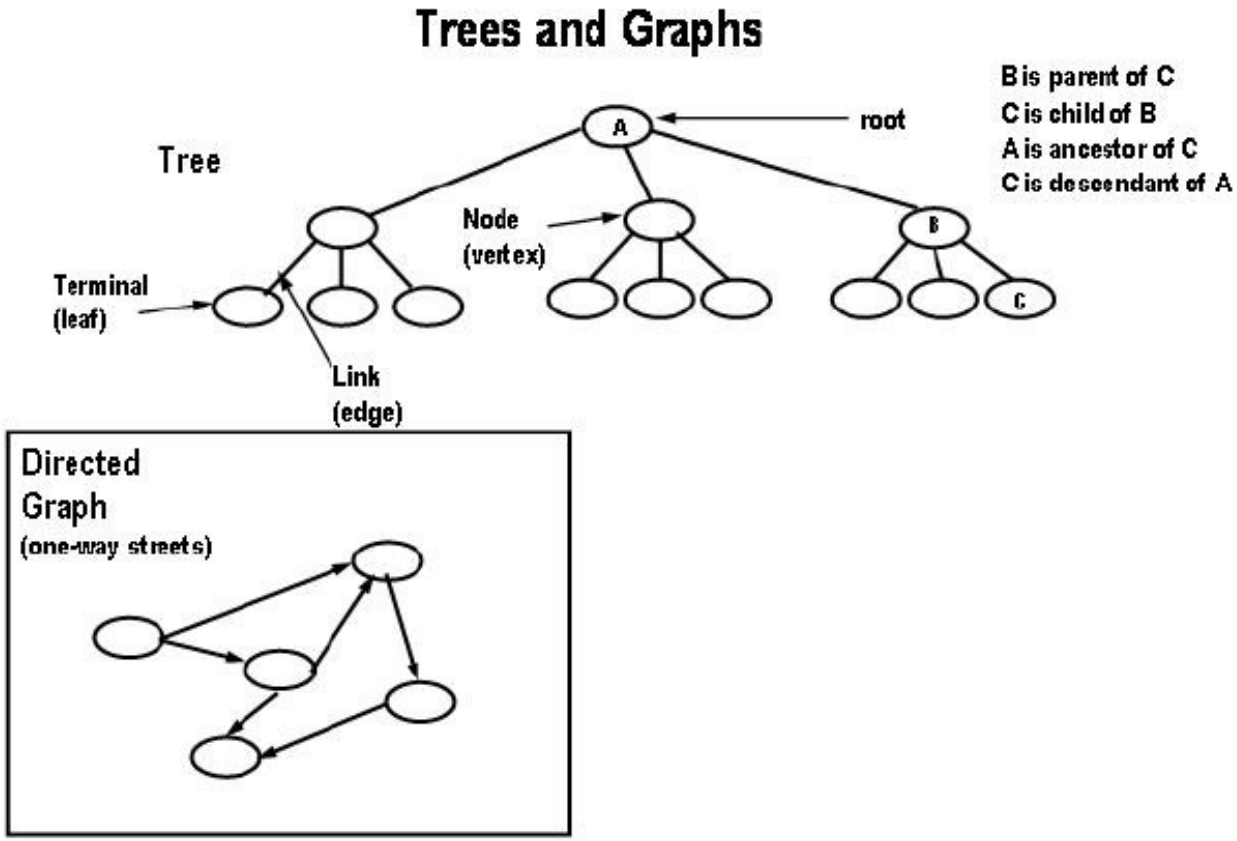
Tree and Graph

- A tree is made up of nodes and links (circles and lines) connected so that there are no loops(cycles). Nodes are sometimes referred to as vertices and links as edges (this is more common in talking about graphs).
- A tree has a root node (where the tree "starts"). Every node except the root has a single parent (aka direct ancestor). More generally, an ancestor node is a node that can be reached by repeatedly going to a parent node. Each node (except the terminal (aka leaf) nodes) has one or more children (aka direct descendants). More generally, a descendant node is a node that can be reached by repeatedly going to a child node.



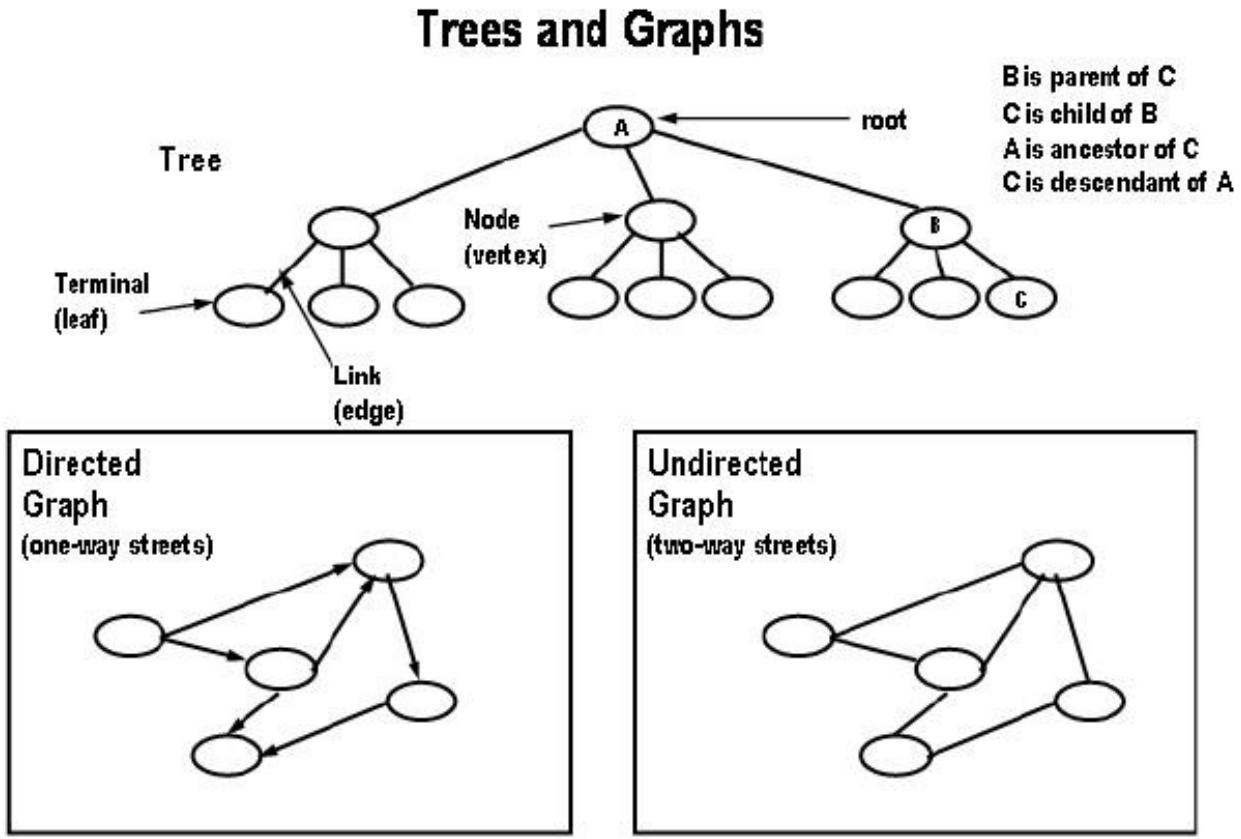
Directed Graph

- A graph is also a set of nodes connected by links but where loops are allowed and a node can have multiple parents. We have two kinds of graphs to deal with: directed graphs, where the links have direction (akin to one-way streets).



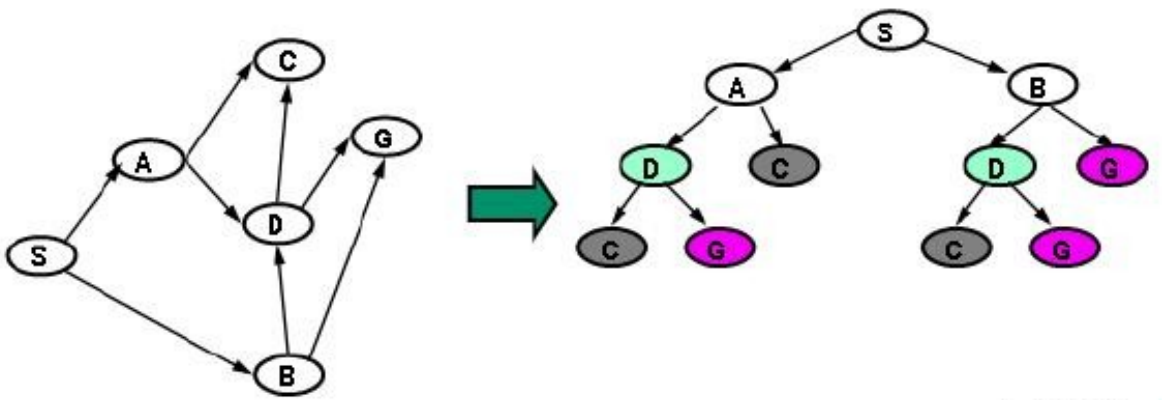
Undirected Graph

- Undirected graphs where the links go both ways. You can think of an undirected graph as shorthand for a graph with directed links going each way between connected nodes.



Graph Search as Tree Search

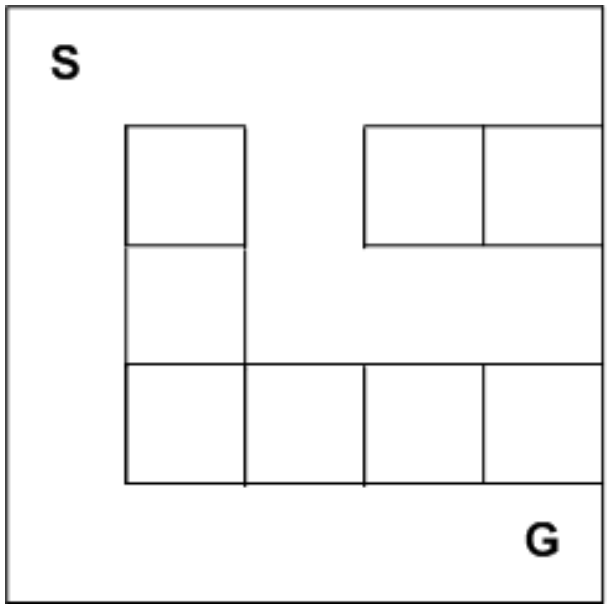
- We can turn graph search problems (from S to G) into tree search problems by:
  - Replacing undirected links by 2 directed links.
  - Avoiding loops in path (or keeping track of visited nodes globally).

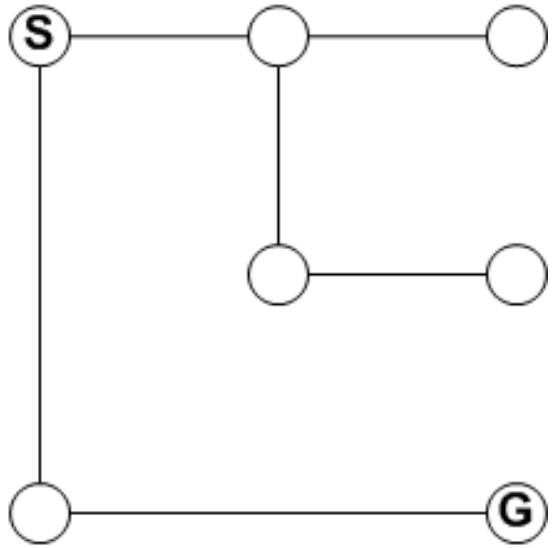


CYPRUS INTERNATIONAL UNIVERSITY

Exercise: Design Graph of Maze for Pathfinding Problem

- The S node represents the start position, whereas the G node represents the goal.
- Think about the location where the intelligent agent needs to make decision.





CYPRUS INTERNATIONAL UNIVERSITY

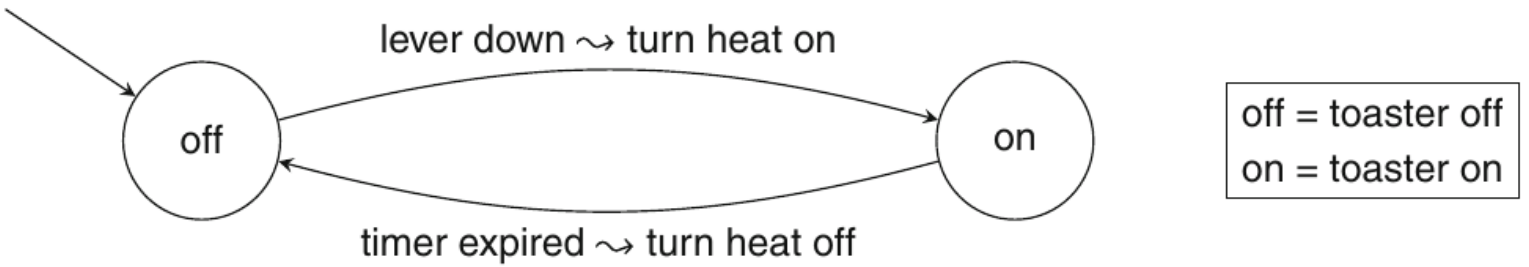
Finite State Machine (FSM)

- Graphs can also take on more abstract forms. Consider a graph structured as follows: nodes represent descriptions of various world states, such as the arrangement of blocks in a scene, while links symbolize actions that transition between these states.
- In this context, navigating from one node to another (specifically from a starting point to a desired endpoint) represents a "plan of action" for achieving a target state from an initial condition. This type of graph holds particular significance in the field of artificial intelligence.

CYPRUS INTERNATIONAL UNIVERSITY

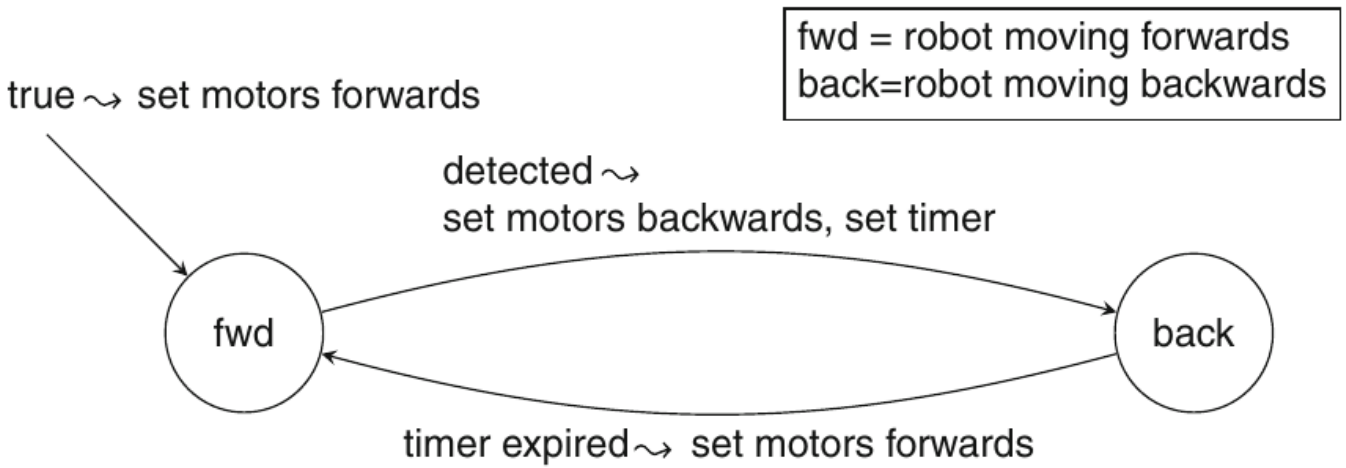
State Machines

- What is state ?
  1. For example: initially, the toaster is in the off state; when you push the lever down, a transition is made to the on state and the heating elements are turned on; finally, when the timer expires, a transition is made back to the off state and the heating elements are turned off.
  2. A finite state machine (FSM) consists of a set of states s_i and a set of transitions between pairs of states s_i , s_j. A transition is labeled condition/action: a condition that causes the transition to be taken and an action that is performed when the transition is taken.



Case Study: Persistent Behaviour I

- The robot moves forwards until it detects an object. It then moves backwards for one second and reverses to move forwards again.



Case Study: Persistent Behaviour II

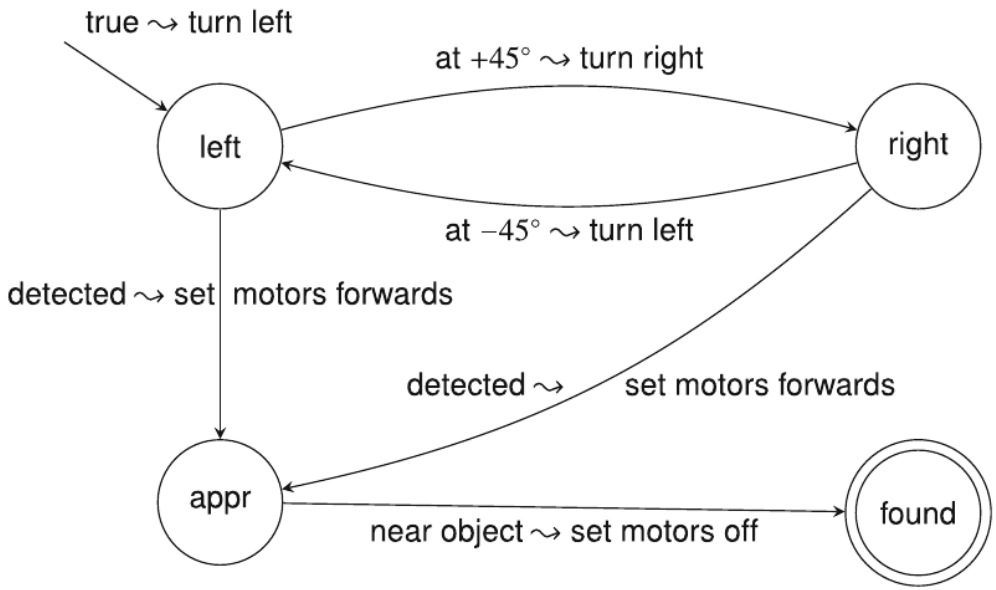
Algorithm 4.1: Persistent

```
        integer timer                                   // In milliseconds
        states current ← fwd
 1: left-motor-power ← 100
 2: right-motor-power ← 100
 3: loop
 4:     when current = fwd and object detected in front
 5:         left-motor-power ← −100
 6:         right-motor-power ← −100
 7:         timer ← 1000
 8:         current ← back
 9:
10:     when current = back and timer = 0
11:         left-motor-power ← 100
12:         right-motor-power ← 100
13:         current ← fwd
```

CYPRUS INTERNATIONAL UNIVERSITY

Case Study: Search & Approach Controller I

- The robot searches left and right ($\pm 45°$). When it detects an object, the robot approaches the object and stops when it is near the object.



left = robot turning left to search
right = robot turning right to search
appr = robot approaching object
found = robot found object

Case Study: Search & Approach Controller II

- *Final state* denoted by a double circle in the diagram.
- *Nondeterminism*: States left and right each have two outgoing transitions, one for reaching the edge of the sector being searched and one for detecting an object.
- The meaning of nondeterminism is that any of the outgoing transitions may be taken. There are three possibilities:
  - The object is detected but the search is not at an edge of the sector; in this case, the transition to appr is taken.
  - The search is at an edge of the sector but an object is not detected; in this case, the transition from left to right or from right to left is taken.
  - The search is at an edge of the sector exactly when an object is detected; in this case, an arbitrary transition is taken. That is, the robot might approach the object or it might change the direction of the search.

CYPRUS INTERNATIONAL UNIVERSITY

Case Study: Search & Approach Controller III

- Case study: Fill in the missing lines in the following Algorithm.

Algorithm 4.2: Search and approach

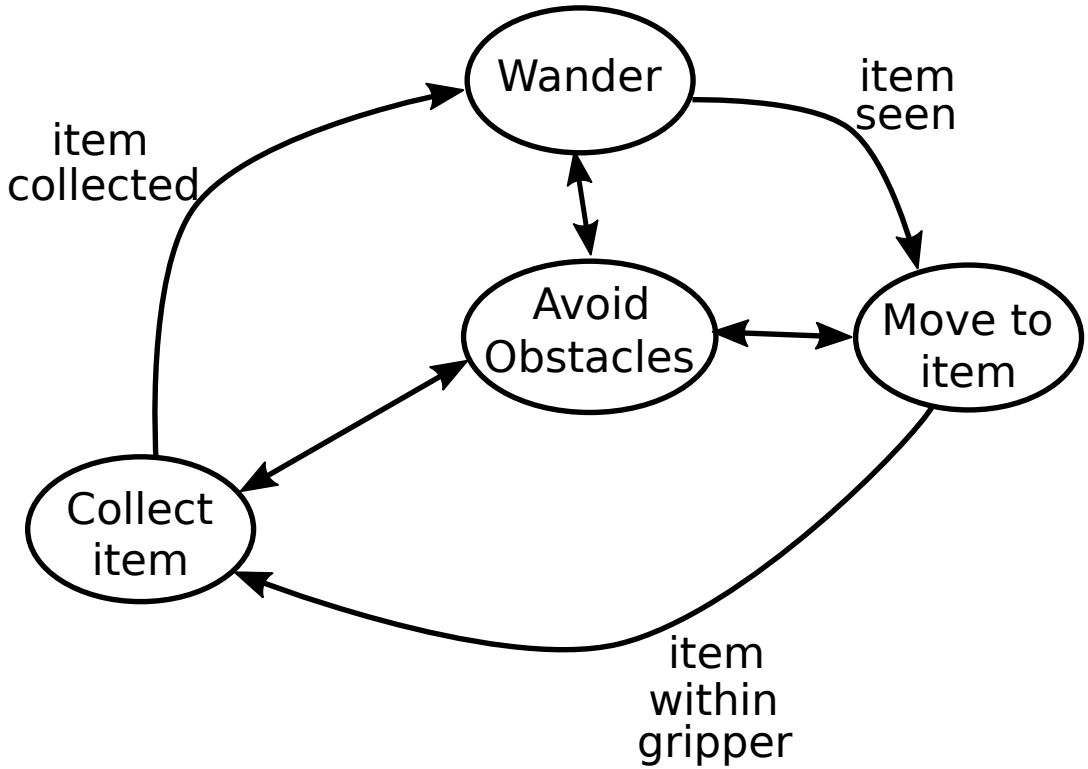
states current ← left

```
1: left-motor-power ← 50                    // Turn left
2: right-motor-power ← 150
3: loop
4:     when object detected
5:         if current = left
6:             left-motor-power ← 100       // Go forwards
7:             right-motor-power ← 100
8:             current ← appr
9:         else if current = right
10:            ...
11:    when at +45°
12:        if current = left
13:            left-motor-power ← 150       // Turn right
14:            right-motor-power ← 50
15:            current ← right
16:    when at −45°
17:        ...
18:    when object is very near
19:        if current = appr
20:            ...
```

CYPRUS INTERNATIONAL UNIVERSITY

Case Study: Rubbish Collecting Autonomous Truck

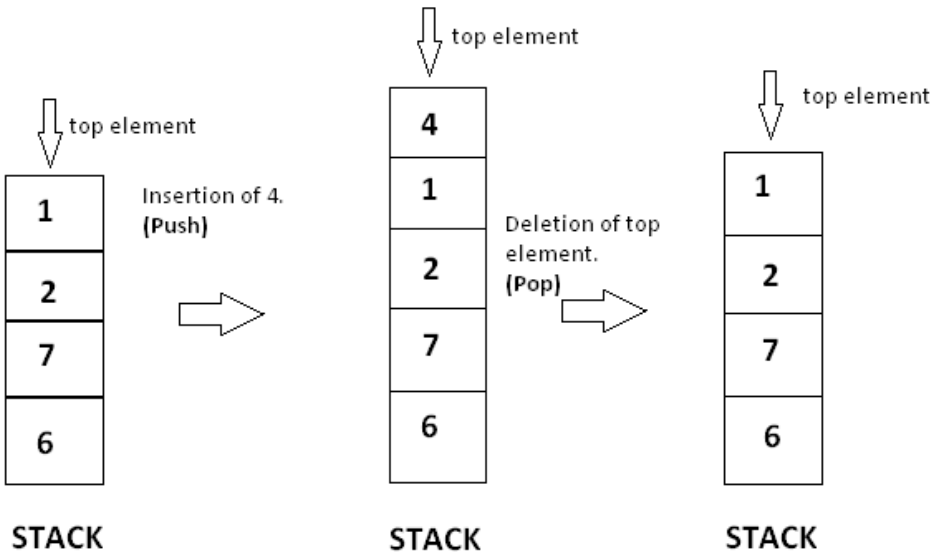
- Wandering – to search for rubbish.
- Moving towards items – for when an item is spotted and the robot wants to collect it.
- Collecting item – for dealing with collecting items.
- Avoiding obstacles – avoid any obstacles on the route of the robot.



CYPRUS INTERNATIONAL UNIVERSITY

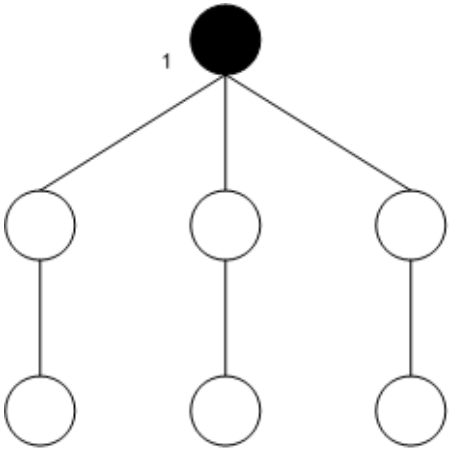
Depth-First Search I

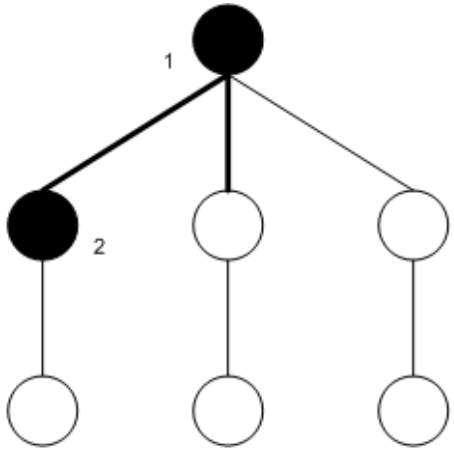
- The Depth-First Search (DFS) is one of the simplest search algorithms for directed graphs without cycles.
- It can be designed using either recursion or a stack.
- The algorithm searches each branch, all the way to the end.
- It is called uninformed search (blind search).
- In stack implementation, last in the stack, will be first out (LIFO) from the stack.

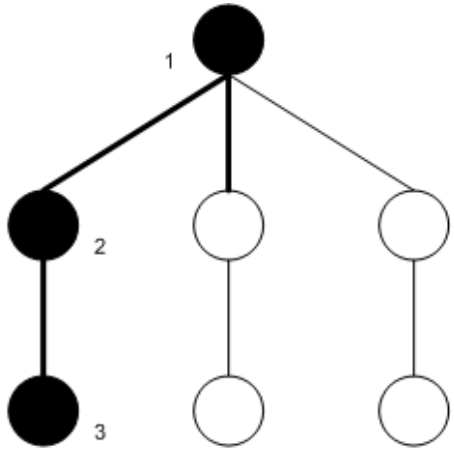


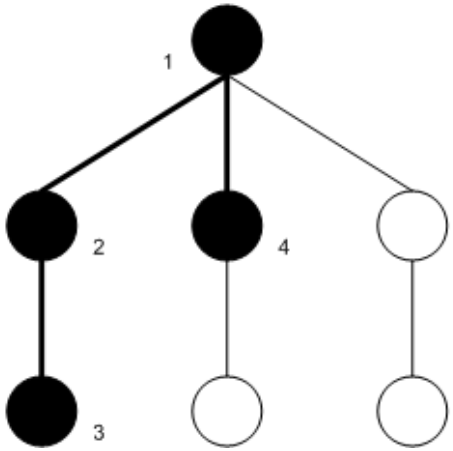
CYPRUS INTERNATIONAL UNIVERSITY

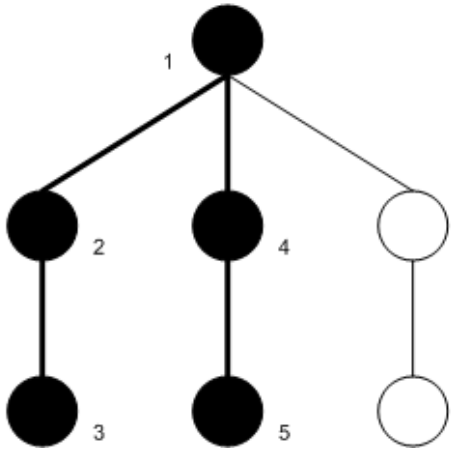
Depth-First Search II

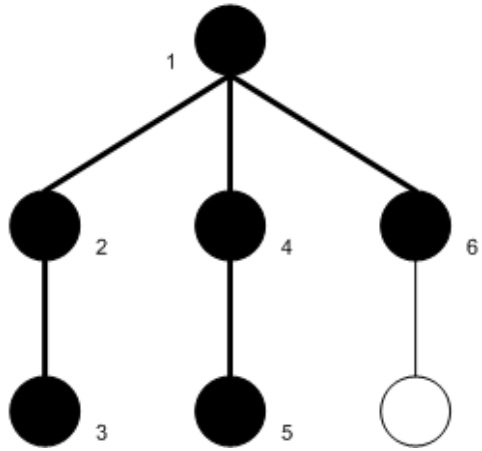












CYPRUS INTERNATIONAL UNIVERSITY

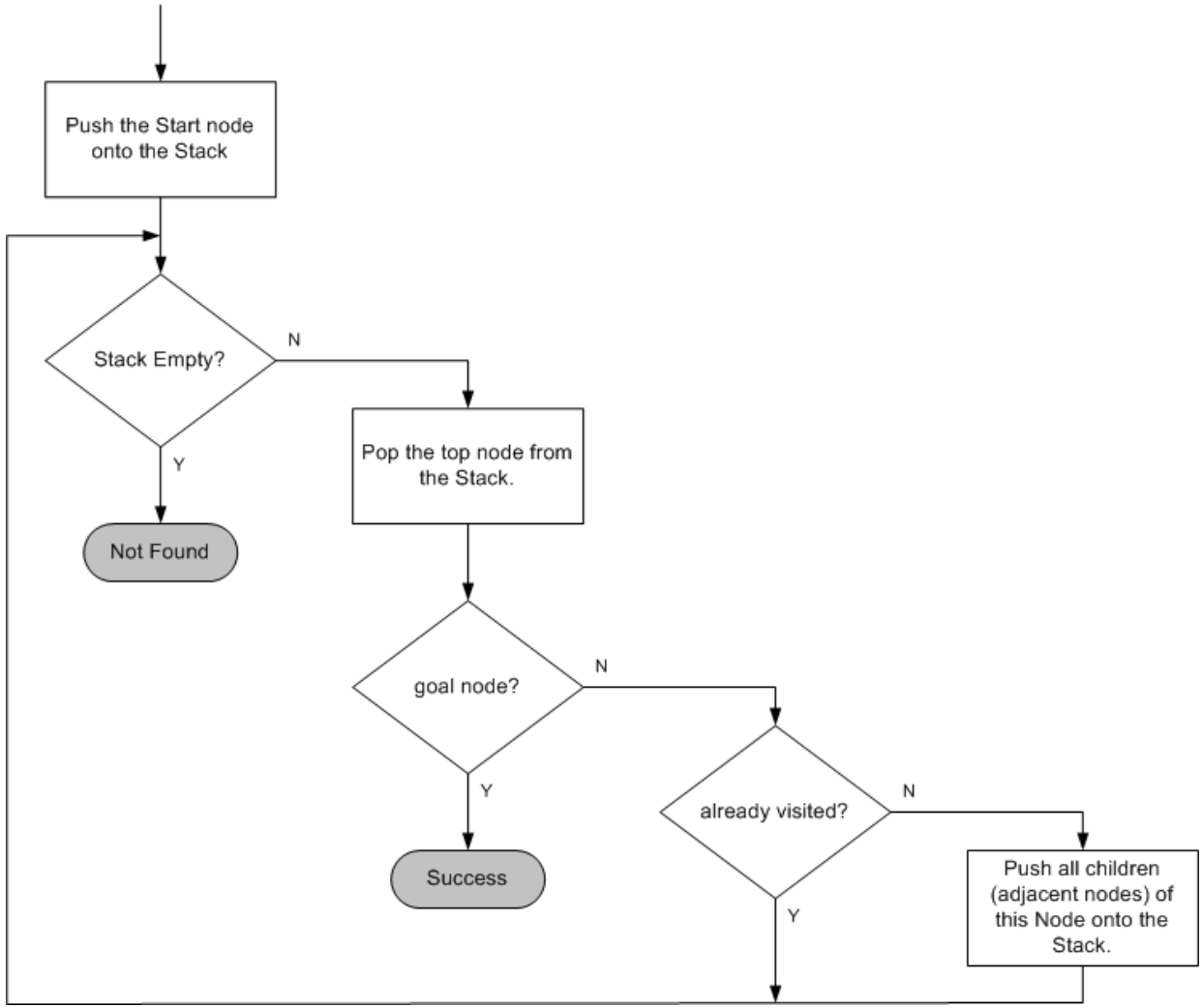
Depth-First Search III

Algorithm:

- We take the start node and push it on a stack.
- We then pop the top item off of the stack and check to see if it is the goal node.
- If the top item is goal, the search has succeeded and ended.
- Otherwise, we check to see if this node has been visited already.
- If it was visited, we ignore it and get the next item on the stack.
- Otherwise, we push all adjacent nodes of the current node onto the stack (in left-to-right order).
- Then, the process begins again.

CYPRUS INTERNATIONAL UNIVERSITY

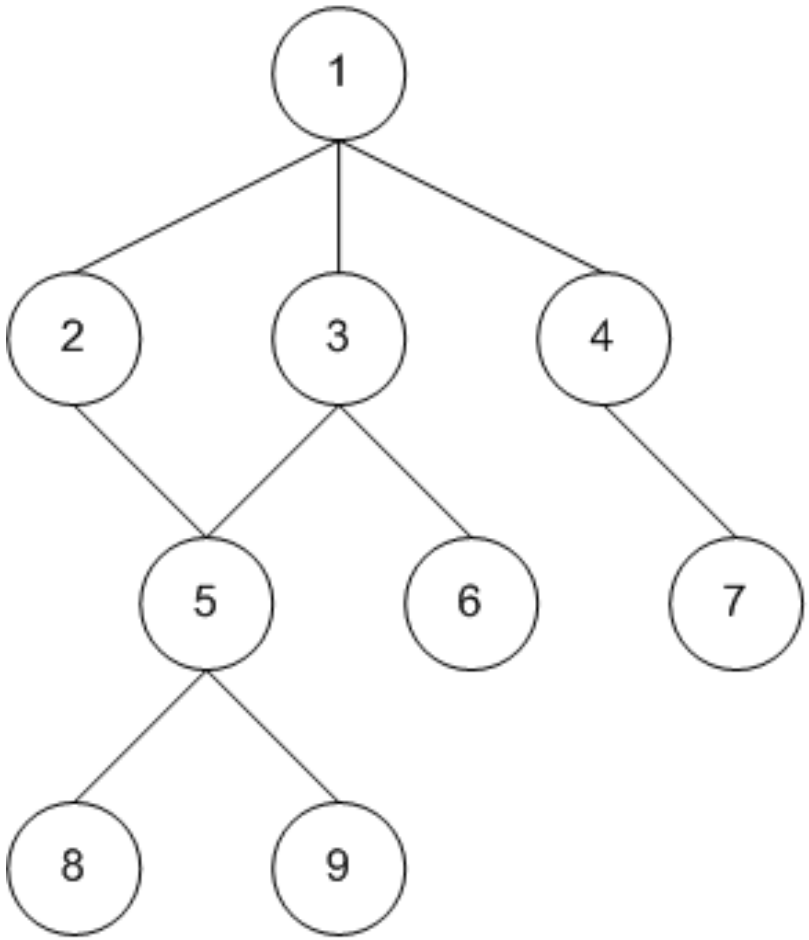
Depth-First Search IV



CYPRUS INTERNATIONAL UNIVERSITY

Depth-First Search V

- **Case study:** find the sequence of the nodes visited by DFS algorithm. Search must be performed in left-to-right.



CYPRUS INTERNATIONAL UNIVERSITY

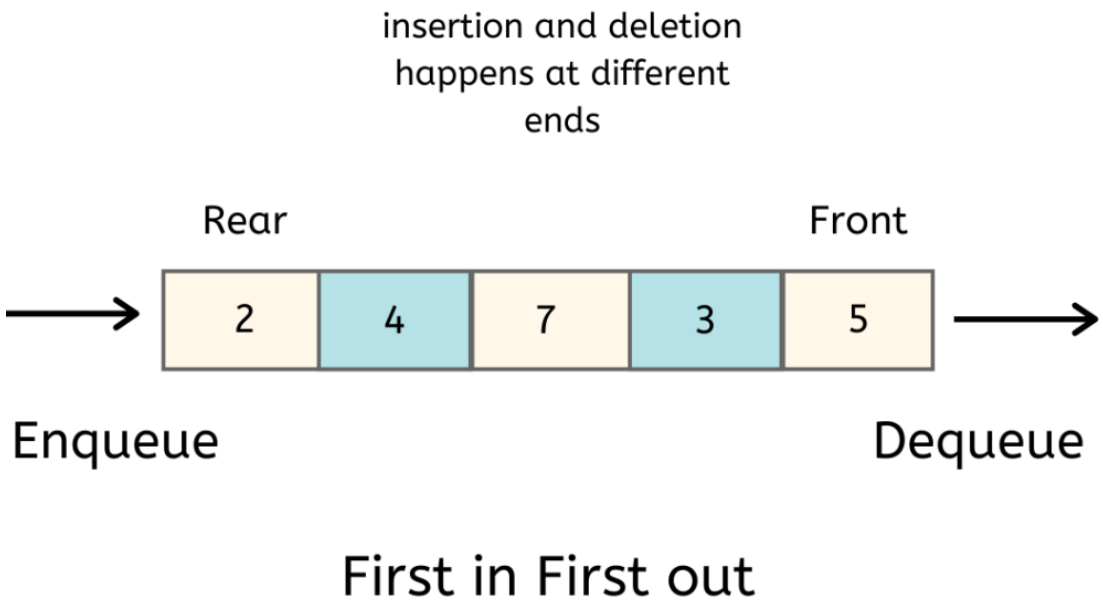
Pseudocode of DFS

```
DFS(G, u)
    u.visited = true
    for each v in G.Adj[u]
        if v.visited == false
            DFS(G,v)

init() {
    For each u in G
        u.visited = false
     For each u in G
       DFS(G, u)
}
```

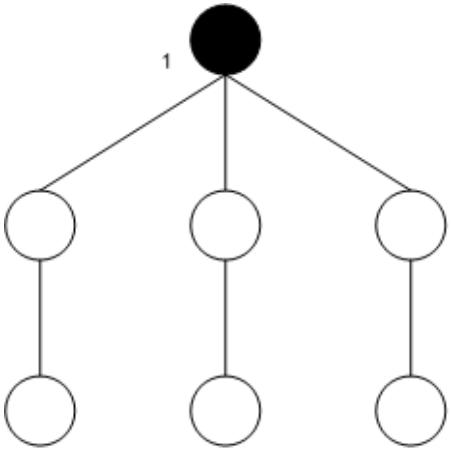
Breadth First Search I

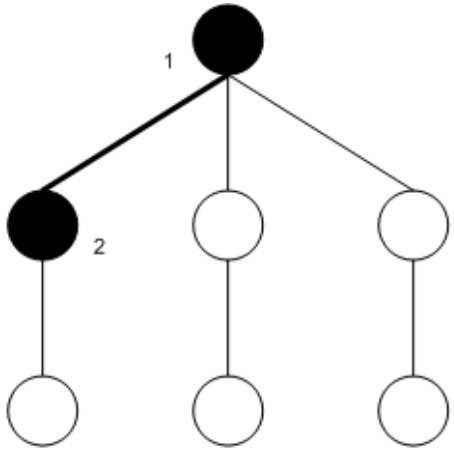
- The breadth first search (BFS) algorithm is another simple search algorithm for directed graphs without cycles.
- It can be implemented recursively or non-recursively.
- The algorithm searches all adjacent nodes to the current first, before descending down each branch.
- In queue implementation, first in the queue, will be first out (FIFO) from the queue.

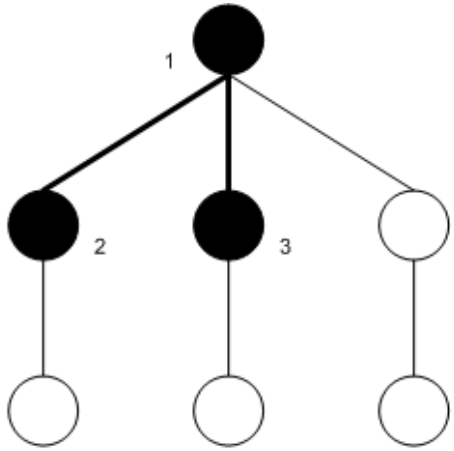


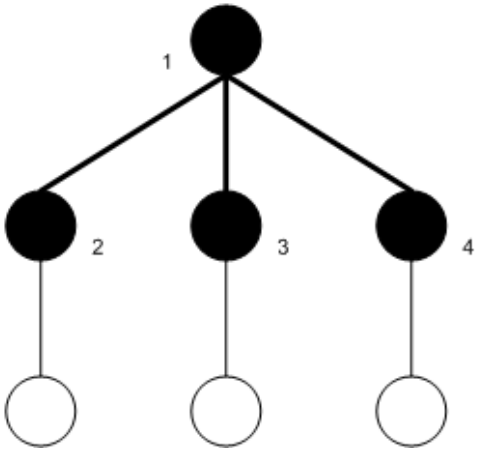
CYPRUS INTERNATIONAL UNIVERSITY

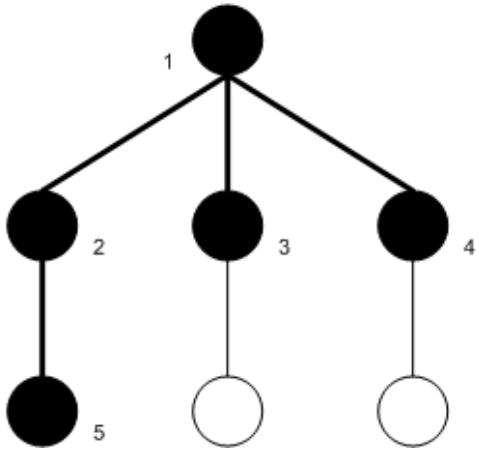
Breadth First Search II

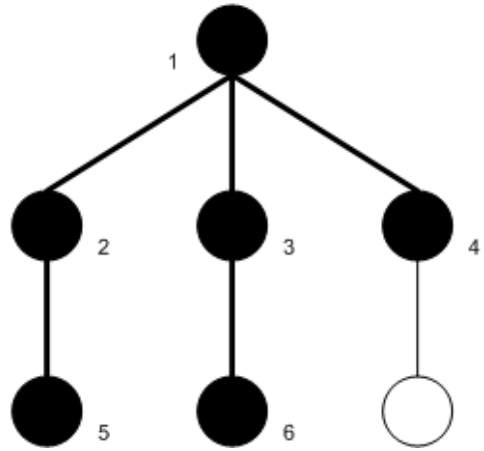












CYPRUS INTERNATIONAL UNIVERSITY

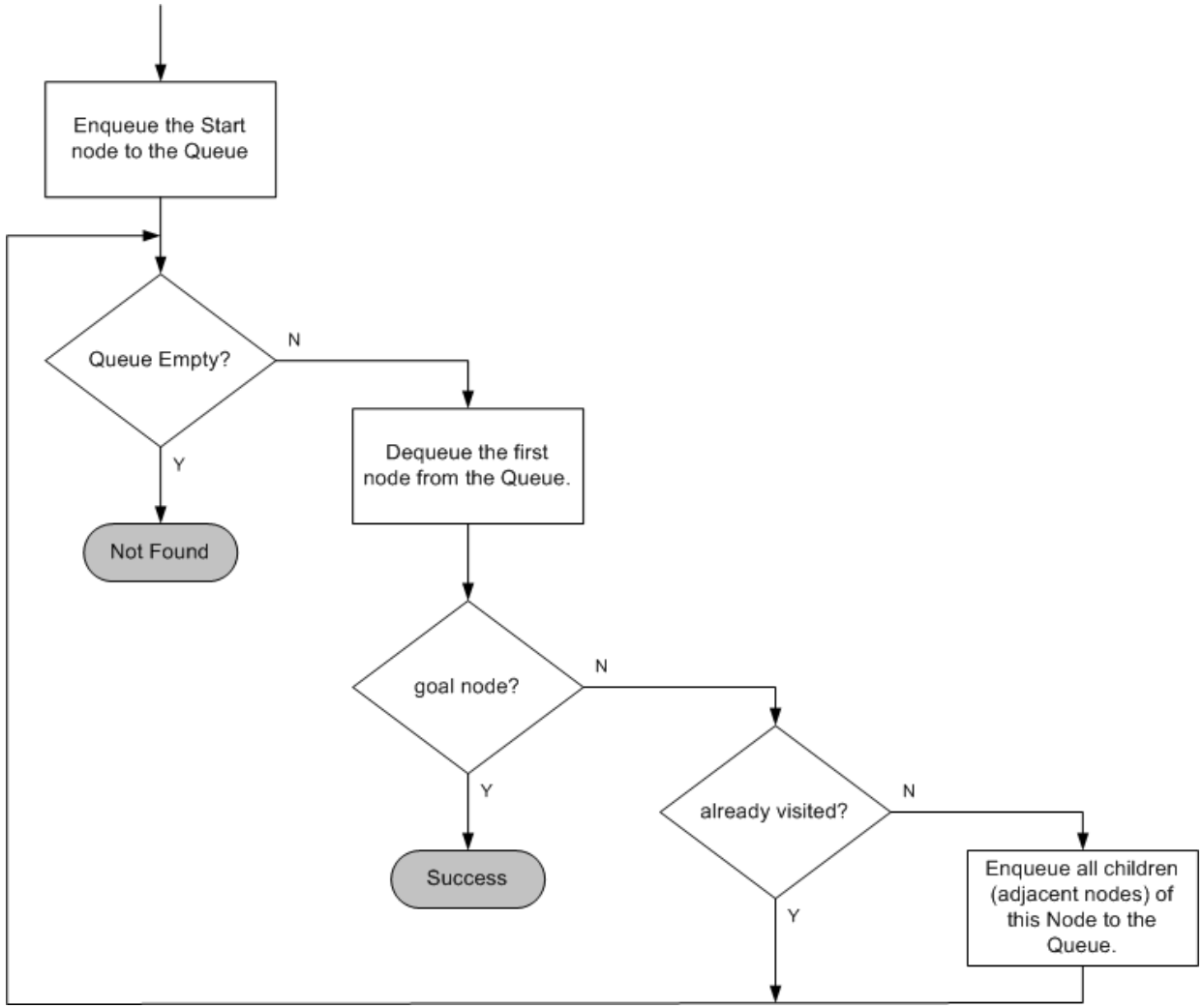
Breadth First Search III

Algorithm:

- We take our start node and enqueue it to the queue.
- We then dequeue the top item from the queue and check to see if it is the goal node.
- If it is the goal, then search has succeeded and ended.
- Otherwise, we check to see if this node has been visited already.
- If it was visited,we ignore it and get the next item from the queue.
- Otherwise, we enqueue all of the adjacent nodes of the current node (in right to left order).
- Then, the process begins again.

CYPRUS INTERNATIONAL UNIVERSITY

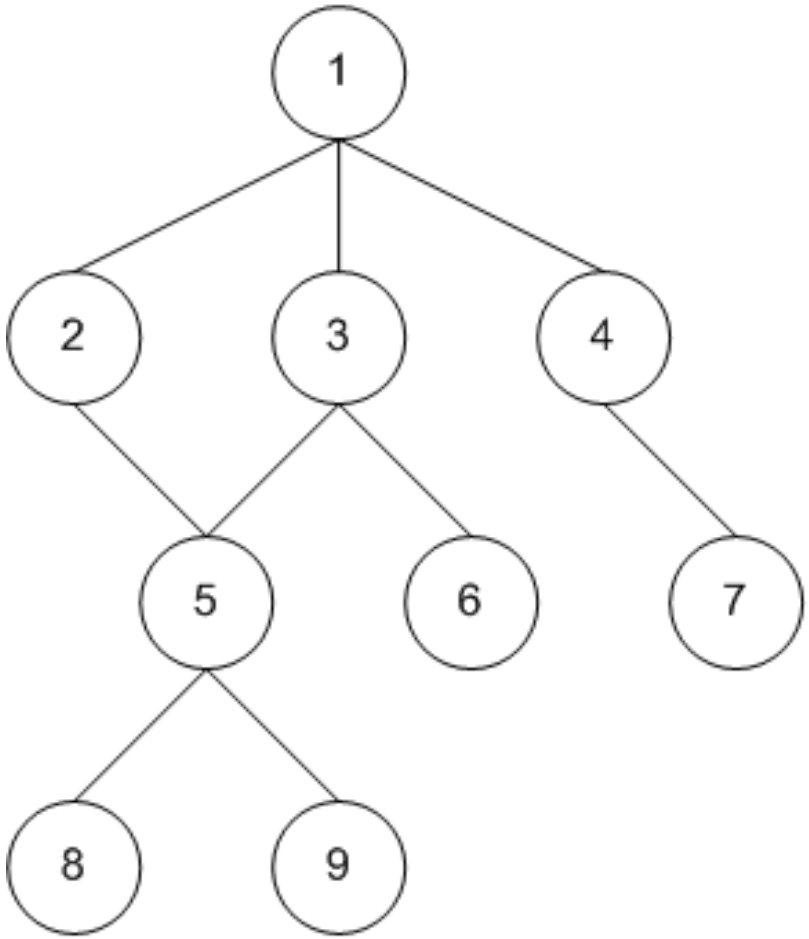
Breadth First Search IV



CYPRUS INTERNATIONAL UNIVERSITY

Breadth First Search V

- **Case study:** find the sequence of the nodes visited by BFS algorithm. Search must be performed in left-to-right.



CYPRUS INTERNATIONAL UNIVERSITY

Breadth First Search VI

- This particular problem was ideal for DFS, but it illustrates how poorly BFS operates with this problem.
- BFS algorithm tends to perform better with narrow trees.
- If the tree is wide, BFS may not operate as efficiently as its DFS counterpart.
- When BFS finds the goal node, it is the minimum depth solution.

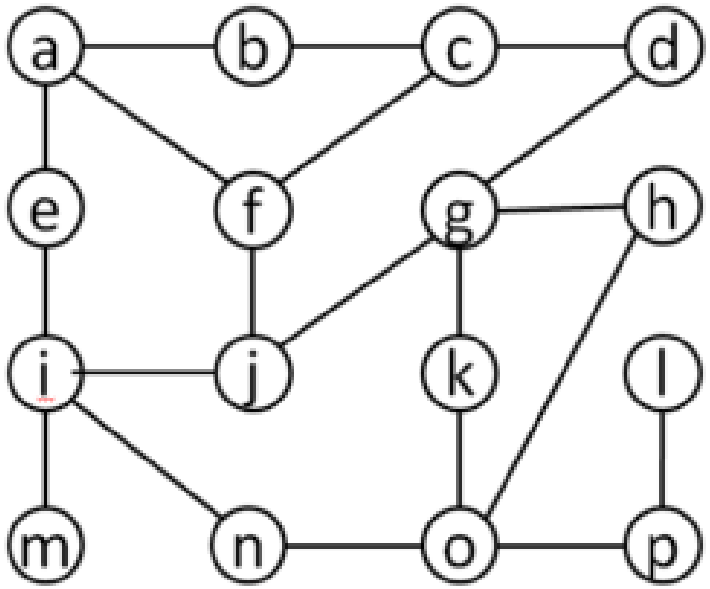
CYPRUS INTERNATIONAL UNIVERSITY

Pseudocode of BFS

```
create a queue Q
mark v as visited and put v into Q
while Q is non-empty
    remove the head u of Q
    mark and enqueue all (unvisited) neighbours of u
```

Exercises: DFS and BFS

- Search from left-to-right. Initial state is ‘g’ and goal state is ‘d’.
- Show both algorithms, using queue and stack methods, at the end provide the order of the visited states.



CYPRUS INTERNATIONAL UNIVERSITY

Difference between Informed and Uninformed Search in AI

- **Uninformed Search** algorithms have no additional information on the goal node other than the one provided in the problem definition. The plans to reach the goal state from the start state differ only by the order and length of actions.

- **Informed Search** algorithms have information on the goal state which helps in more efficient searching. This information is obtained by a function that estimates how close a state is to the goal state.

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search I

- Rather than move blindly through the state space, A-Star uses a heuristic to identify which node to test next. This is called informed search.
- This method minimises the amount of search that's actually done to find the best path to the goal.
- A-Star is most useful in state-space search problems.

Algorithm:

- A-Star algorithm begins with lists.
- The first is the OPENED list, which contains nodes that are to be examined.
- Each node, representing a location on the search space, is represented with g, h, f elements which represent the heuristic for the node.

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search II

Heuristic Element	Description
h	The cost estimate (given the heuristic function) of getting from this node to the goal node
g	The cost of getting from the parent node to this node (plus the parent's g value)
f	The cost estimate of getting from the start node to the goal node, through this node ($g + h$)

- In the initialisation, we will specify our start and goal nodes.
- The start node is then placed on the OPENED list, awaiting examination by the algorithm.
- While items exist on the OPENED list, we must work toward the goal node.
- If the OPENED list becomes empty that means we have not yet fount the goal node, then no path exists from the start to goal node an we can exit.
- While nodes exist on the OPENED list, we loop and perform the following function as given in eqs. (3), (4) and (6).
- We search and return the node on the OPENED list that has the lowest f value.

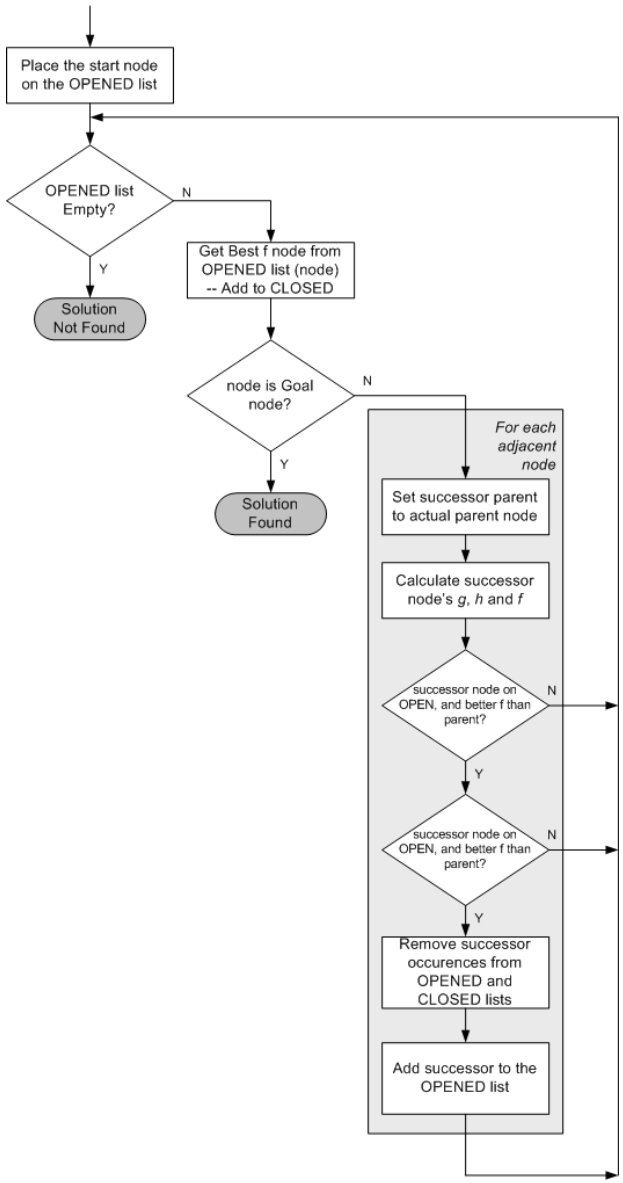
CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search III

- We checked first to see if it is the goal node.
- If it is, the search is done.
- Otherwise, we have to identify all of the adjacent nodes to the current node.
- If the node exists already on either the OPENED or CLOSED lists that means we have looked at the node in the past, and the current node's f value is lower that indicates it is better than any of the OPENED and CLOSED lists, then the node on the OPENED or CLOSED lists is removed and the current node is placed on the OPENED list.
- If the current node's f value is not better than one found on the OPENED or CLOSED list, then it is simply discarded and we continue.
- Finally, if our current node has not been found on either the OPENED or CLOSED list, we add it to the OPENED list.
- Once the goal node is found, we simply walk back through the parent nodes to identify the path from the goal node to the start node.

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search IV



CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search V

How to calculate the cost estimate to goal node heuristic **h** ?

- (a) Either calculate the exact value of h (which is certainly time consuming), or (b) approximate the value of *h* using some heuristics (less time consuming).
- (a) Exact heuristics:
  1. Pre-compute the distance between each pair of cells before running the A-star Search Algorithm.
  2. If there are no blocked cells/obstacles then we can just find the exact value of *h* without any pre-computation using the distance formula/Euclidean Distance.
- (b) Approximation heuristics:
  1. Manhattan Distance: It is nothing but the sum of absolute values of differences in the goal's x and y coordinates and the current cell's x and y coordinates respectively. When to use this heuristic? When we are allowed to move only in four directions only (right, left, top, bottom):

$$h = |curr_x - goal_x| + |curr_y - goal_y|$$

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search VI

2. Diagonal Distance: It is nothing but the maximum of absolute values of differences in the goal's x and y coordinates and the current cell's x and y coordinates respectively. When to use this heuristic? When we are allowed to move in eight directions only (similar to a move of a King in Chess):

$$dx = |curr_x - goal_x|$$
$$dy = |curr_y - goal_y|$$
$$h = D \cdot (dx + dy) + (D2 - 2 \cdot D) \cdot min(dx, dy)$$

where D is length of each node(usually $= 1$) and D2 is diagonal distance between each node (usually $= \sqrt{2}$).

3. Euclidean Distance: As it is clear from its name, it is nothing but the distance between the current cell and the goal cell using the distance formula. When to use this heuristic? When we are allowed to move in any directions.

$$h = \sqrt{(curr_x - goal_x)^2 + (curr_y - goal_y)^2} \quad (2)$$

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search VII

Example:

- First equation is our cost function. This heuristic is a standard Manhattan distance function, providing a grid distance between two points.

$$h = cost \cdot (|curr_x - goal_x| + |curr_y - goal_y|) \tag{3}$$

where the cost element defines the cost of moving along one axis and is simply defined as 1.0.

- Calculating the g value represents the cost of moving from the current node to the successor node.

$$g = 1.0 + \alpha \cdot (parent_g - 1.0) \tag{4}$$

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search VIII

- g represents the cost from the starting node to current node, so this reason the above equation only provides the cost from the current node to successor node. In order to compute overall cost, we must also add the parent's g value so in this way it yields the correct cost from starting node to the successor.

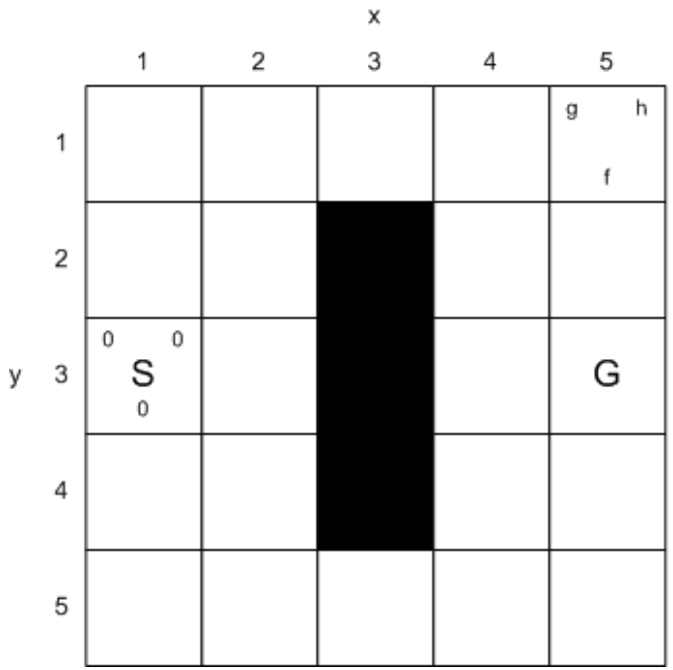
$$g = g + parent_g \tag{5}$$

- Finally we calculate the past cost of going through this node.

$$f = h + g \tag{6}$$

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search IX



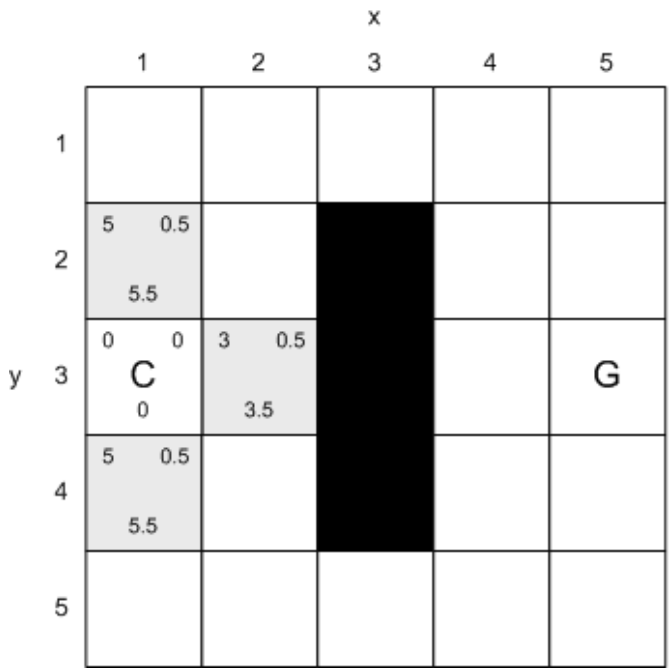
- We identify the adjacent nodes of the current node (1,3).
- We calculate the g and h values (using 1.0 as our cost constant and 0.5 as our α constant).

$$h = 1.0 \cdot (|2-5| + |3-3|) = 3.0$$
$$g = 1.0 + 0.5 \cdot (0 - 1.0)$$
$$f = 3.0 + 0.5 = 3.5$$

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search X

- Adjacent nodes of the start node are placed on the OPENED list because they are not exist already.
- Because we have checked our start node, we placed this node on the CLOSED list.
- Then we start again, picking the best node from our OPENED list, which is in the case is position (2,3) having the lowest f value of 3.5, Then we calculate the heuristic values of the adjacent nodes.

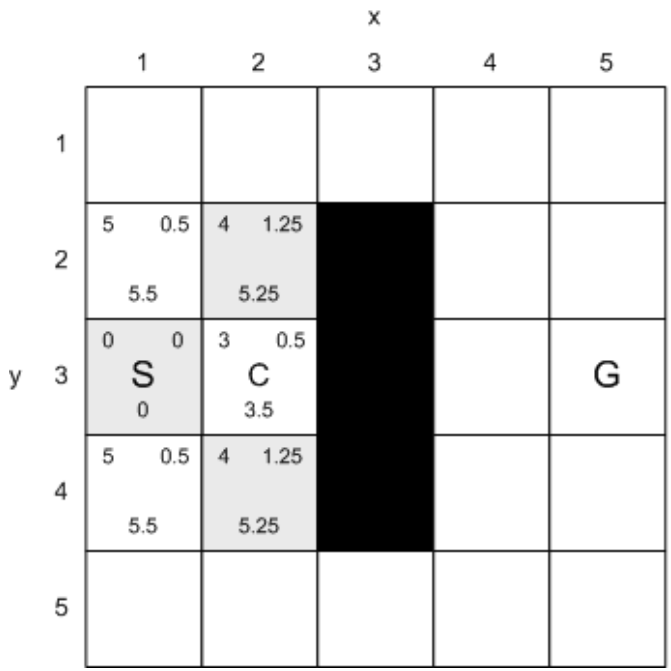


Opened List	Closed List
(1,2)	(1,3)
(1,4)	
(2,3)	

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search XI

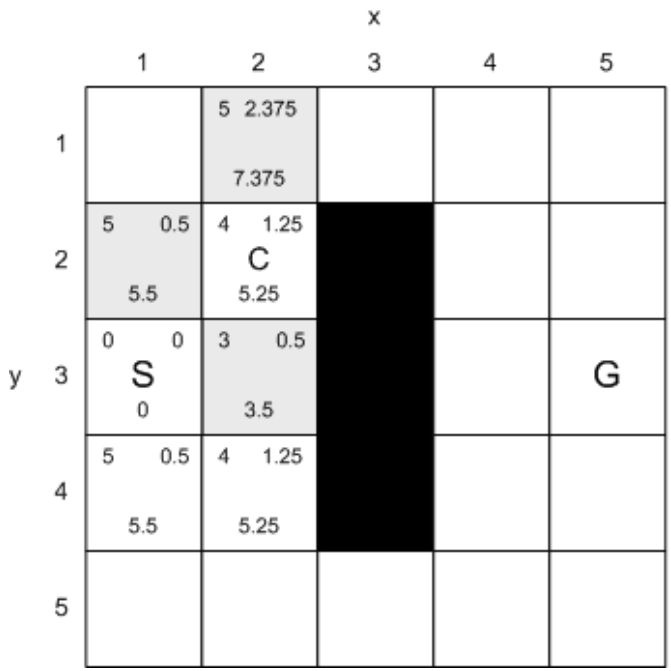
- Nodes (2,2) and (2,4) have identical f values. The algorithm choose randomly, so in this case node (2,2) is selected and is placed on the list.
- Because symmetry exists in this example, the next node to be chosen to examine will be node (2,4).



Opened List	Closed List
(1,2)	(1,3)
(1,4)	(2,3)
(2,2)	
(2,4)	

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search XII



Opened List	Closed List
(1,2)	(1,3)
(1,4)	(2,3)
(2,1)	(2,2)
(2,4)	

- Our OPENED list, which continues growing with the new nodes to examine and the CLOSED list also grows, including nodes that have already been reviewed.

CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search XIII

- The A-Star algorithm is rooted in game programming but, in most cases, is expensive in this domain, so other less computationally expensive methods are used such as *waypoints*, *pre-computed paths* or *look-up tables*.
- A-Star can be used to find paths in *communication networks*, *route-finding in geographic applications*, *robot navigation*, and *solving combinatorial problems*, such as *Rubik's cube*.

CYPRUS INTERNATIONAL UNIVERSITY

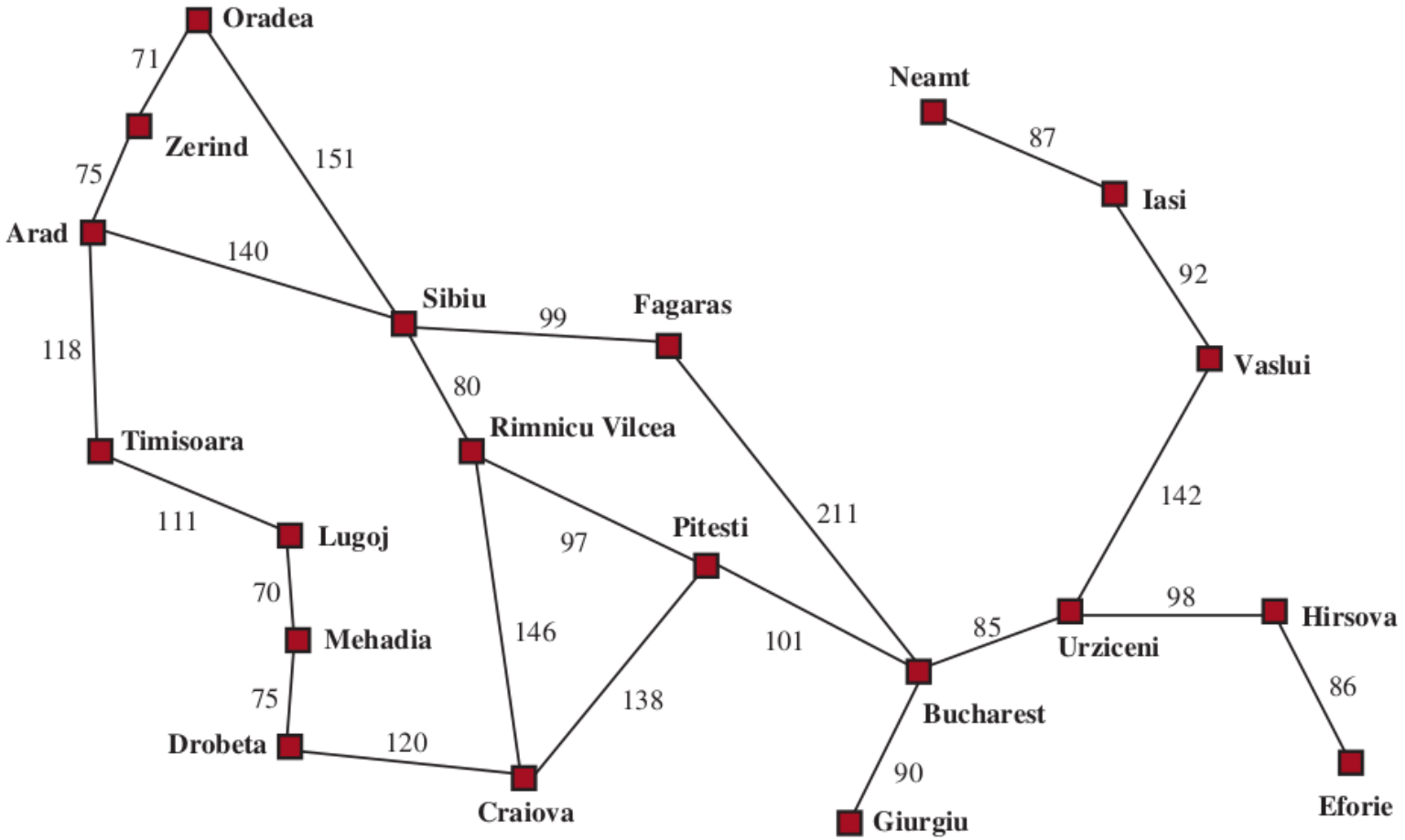
Greedy Best-First (GBF) Search I

- Greedy best-first search expands first the node with the lowest $h(n)$ value (*i.e.* cost estimate of getting from the current node to the goal node), the node that appears to be closest to the goal on the grounds that this is likely to lead to a solution quickly.
- So the evaluation function $f(n) = h(n)$.

CYPRUS INTERNATIONAL UNIVERSITY

Exercise: Find the optimal path using A-Star and GBF Algorithms I

- Road map of Romania, with road distances in miles.



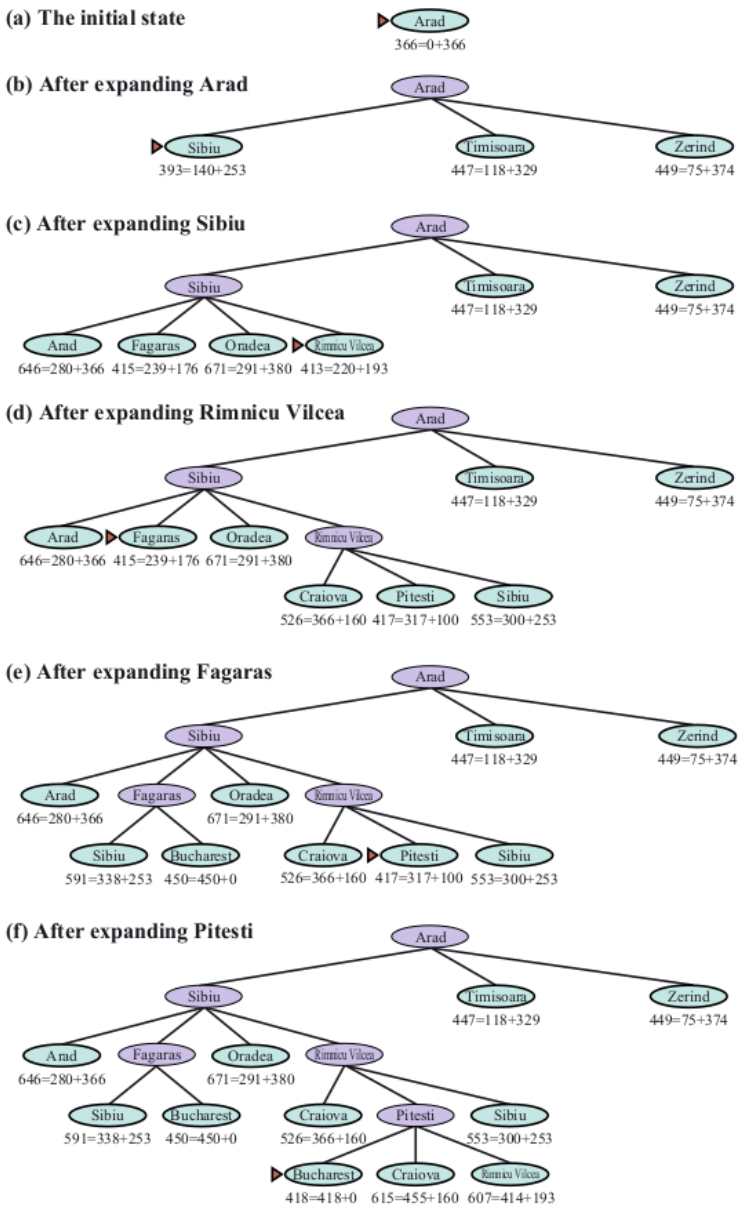
Exercise: Find the optimal path using A-Star and GBF Algorithms II

- Values of h – straight-line distances to Bucharest.

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

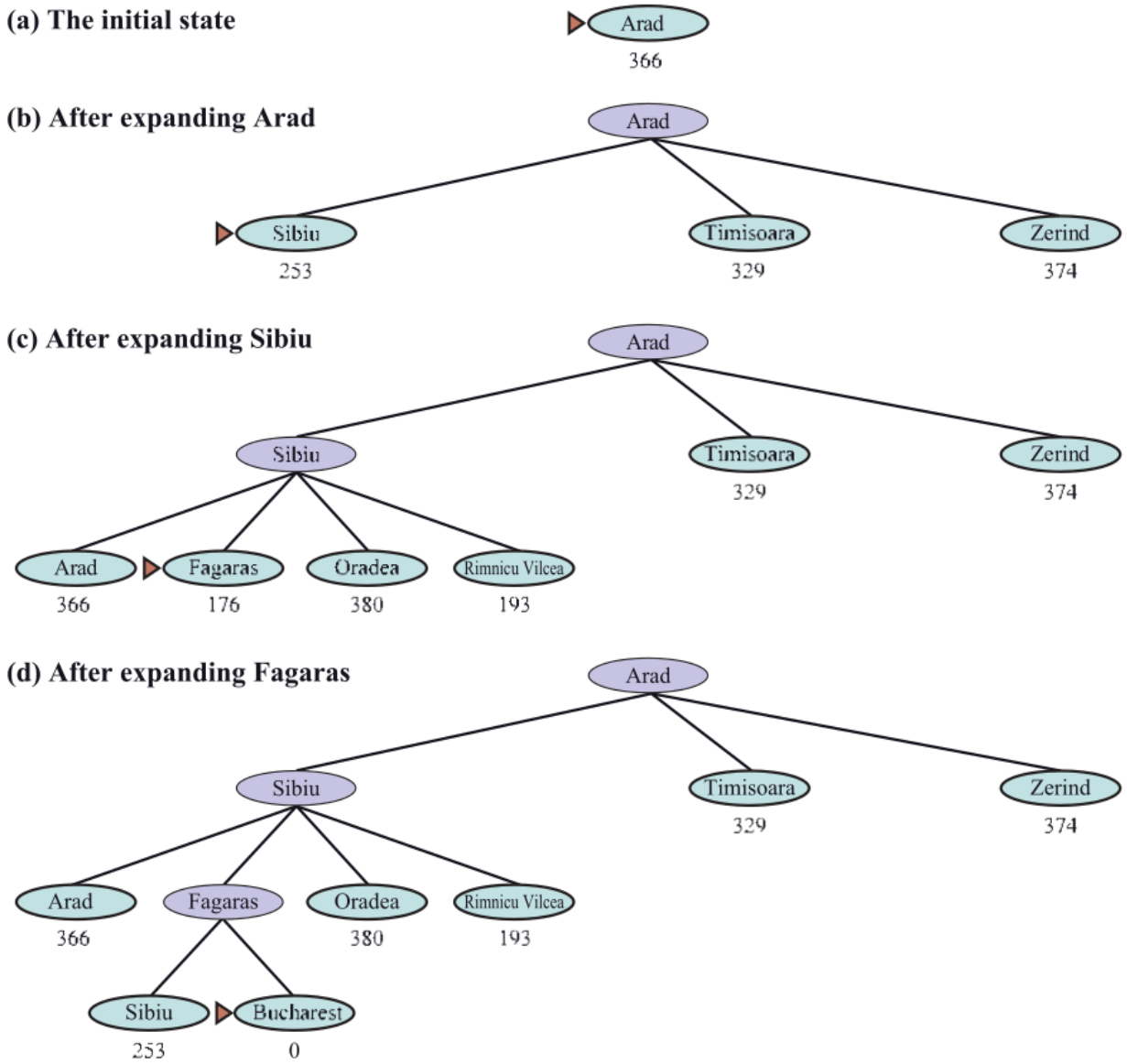
CYPRUS INTERNATIONAL UNIVERSITY

A-Star Search Solution



CYPRUS INTERNATIONAL UNIVERSITY

GBF Search Solution



CYPRUS INTERNATIONAL UNIVERSITY