Manchester Metropolitan University

Artificial Intelligence Principles

6G7V0011 - 1CWK100

Dr. Peng Wang

Email: p.wang@mmu.ac.uk

Department of Computing and Mathematics

Wednesday, Oct. 29th, 2024

Outline

Outline

Review

Manchester Metropolitan University

- Artificial Intelligence and SOTA techniques
- Agents and Rational agents
- Four types of agent programs
  1. Simple reflex agent
  2. Model-based agent
  3. Goal-based agent
  4. Utility-based agent
- **Graph search and Tree (tree-like) search**

Search Concepts

Manchester Metropolitan University

- Frontier, Fringe, Openset
- Reached, closedset
- Expansion
- Exploration strategy - Depth first? Breadth first? etc.

Openset: is the set of nodes we choose currently from - that is, it contains all the nodes we might be interested in looking at next.

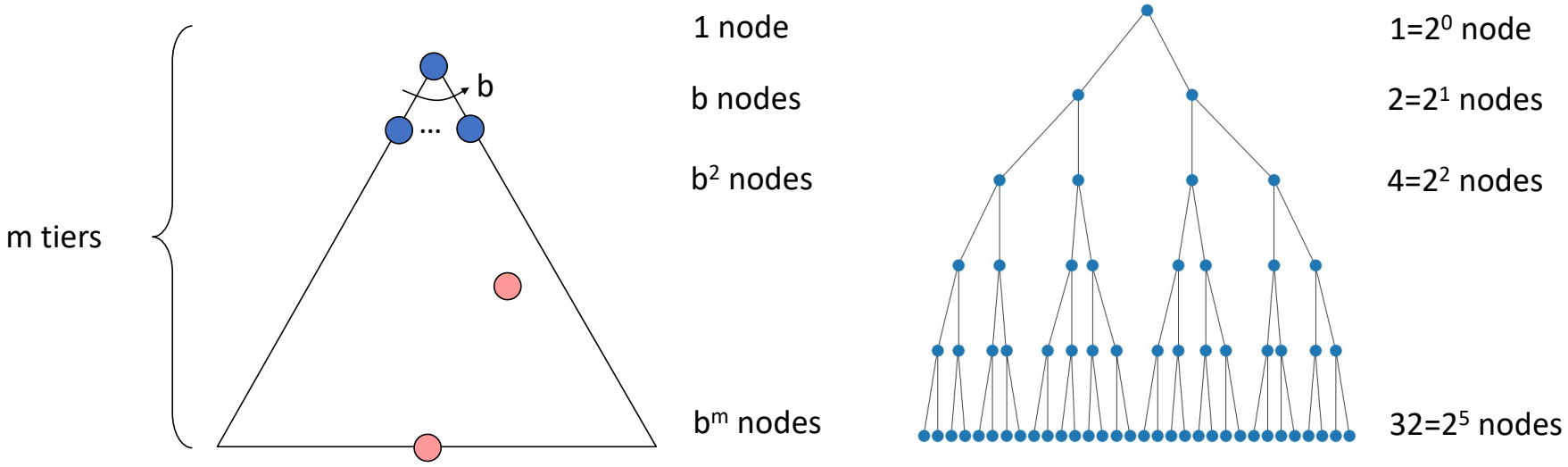
frontier or **fringe**: meaning 'a set of nodes to be expanded. Without causing confusions, we will use them all in the course.

Closedset: is the set of nodes we've already considered.

Sometimes, a Closedset is also called **reached**, meaning 'a set of nodes that are explored already'. Similarly, we will use both terms in this course without causing confusions.

Exploration strategy is the key question!

Search Algorithm Properties



- **Complete**: Guaranteed to find a solution if one exists?
- **Optimal**: Guaranteed to find the least cost path?
- **Time** complexity?
- **Space** complexity?

- Key factors of a search tree
  1. b is the branching factor
  2. m is the maximum depth
  3. solutions at various depths

Node number: $b^0 + b^1 + b^2 + \cdots + b^m = \sum_{i=0}^{m} b^i \backsim \mathcal{O}(b^m)$

Breath First Search (BFS) - graph I

Manchester Metropolitan University

Algorithm 1: Pseudocode of Breath First Search

Input: Initial state **s**, goal state **g**
Output: A *node* helps to retrieve a solution (path) $\mathcal{P}$, or failure

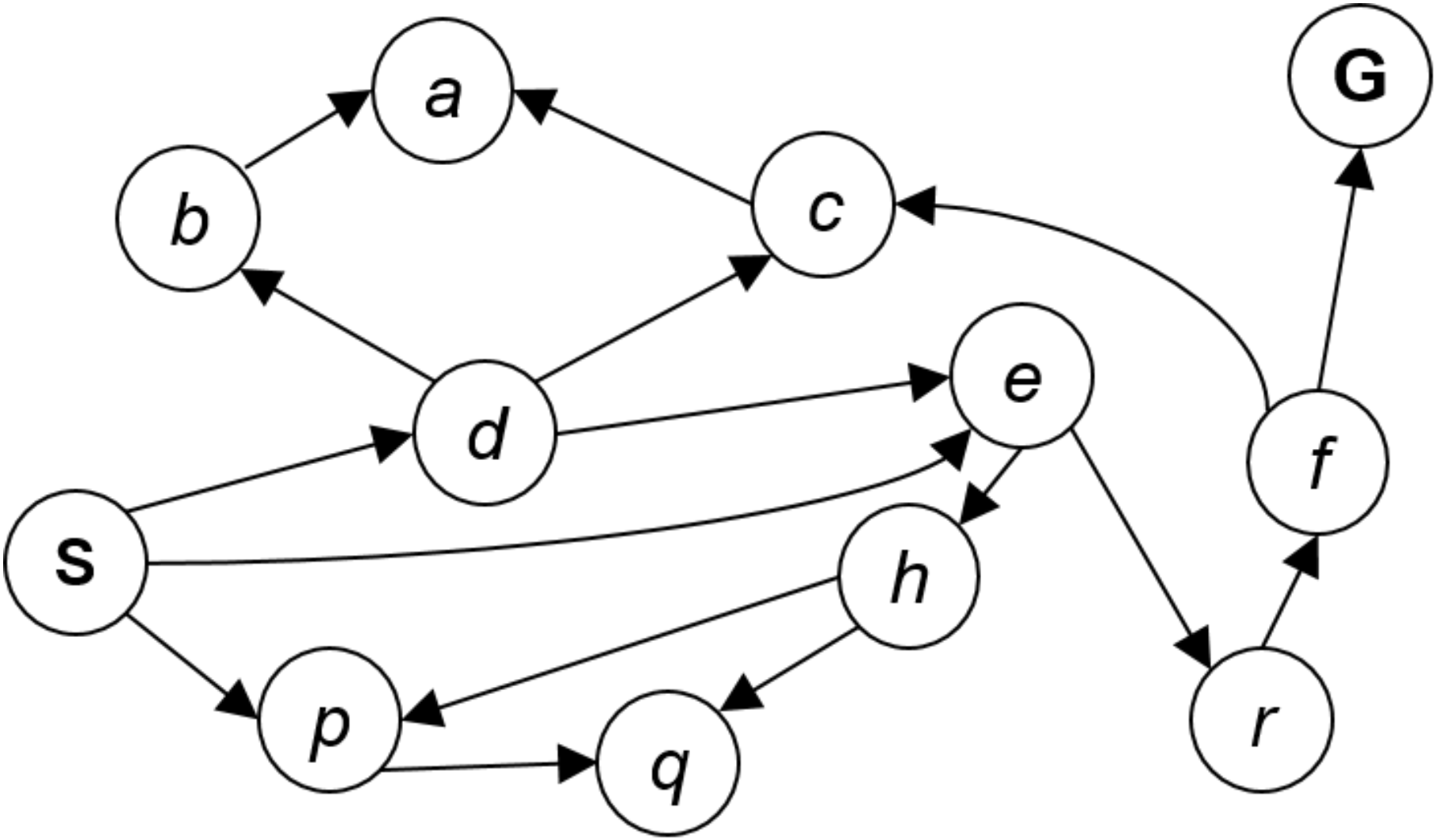
```
1:  node ← with s as state
2:  if s == g then
3:      return node
4:  end if
5:  O ← node, an FIFO queue
6:  C ← ∅
7:  while O != ∅ do
8:      parent ← the first node in O
9:      C ← parent.state
10:     for child in successor (of the current parent) do
11:         v ← child.state
12:         if v is not in C and child is not in O then
```

Breath First Search (BFS) - graph II

Manchester Metropolitan University

```
13:          if v == g then
14:             return child
15:          end if
16:          add child to the end of 𝒪
17:       end if
18:    end for
19: end while
20: return failure
```

Breadth First Search



Breadth First Search Properties

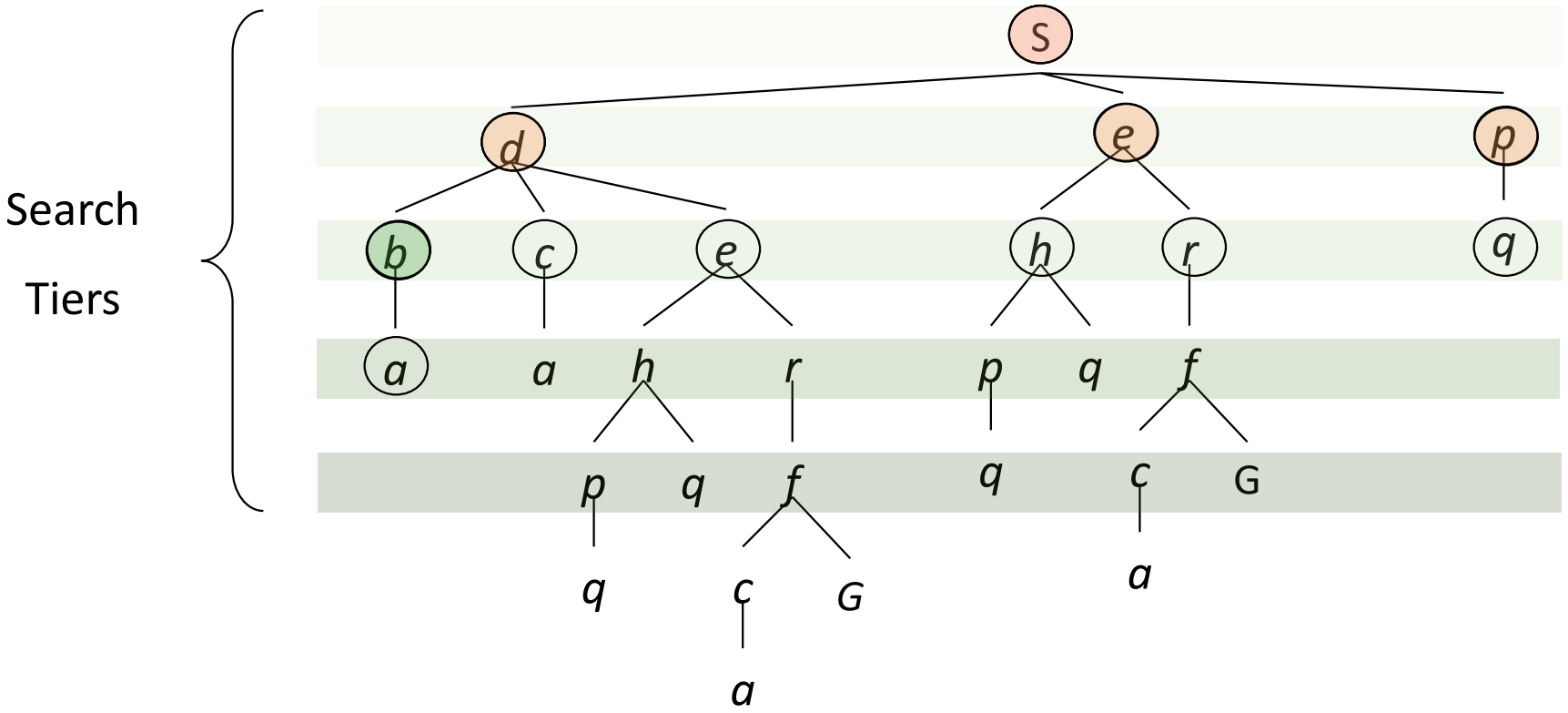
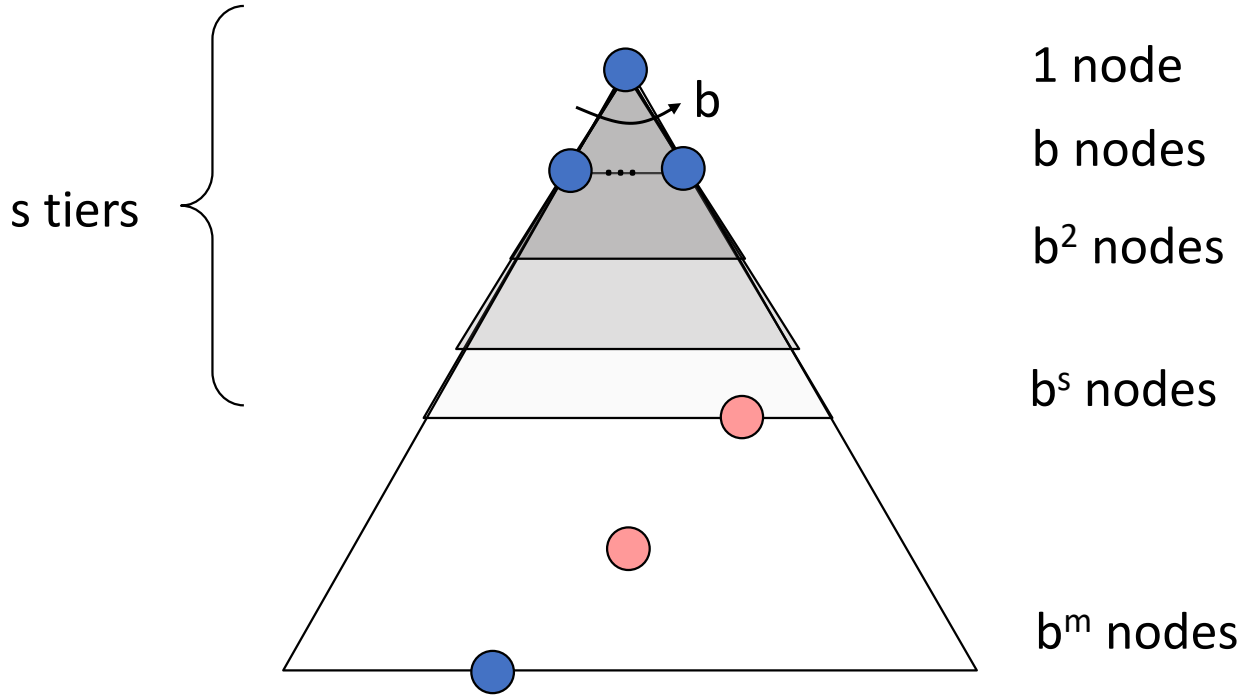


Figure 1: An example of Breadth First Search

Breadth First Search



When search reached the shallowest solution at depth s

- Has processed all nodes above depth s
- Time complexity: $\mathcal{O}(b^s)$
- Space complexity (frontier): $\mathcal{O}(b^s)$
- Complete: s must be finite if a solution exists
- Optimal: When costs of edges are equal c.

Depth First Search (DFS) - graph I

Manchester Metropolitan University

Algorithm 2: Pseudocode of Depth First Search

Input: Initial state **s**, goal state **g**.
Output: A *node* helps to retrieve a solution (path) $\mathcal{P}$, or failure

```
1:  node ← with s as state
2:  O ← node, an LIFO queue
3:  C ← ∅
4:  while O != ∅ do
5:      parent ← the last node in O
6:      v ← parent.state
7:      if v == g then
8:          return parent
9:      end if
10:     C ← parent.state
11:     for child in successor (of the current parent) do
12:         v ← child.state
```

Depth First Search (DFS) - graph II

Manchester Metropolitan University

```
13:         if v is not in C and child is not in O then
14:             add child to O
15:         end if
16:     end for
17: end while
18: return failure
```

Depth First Search

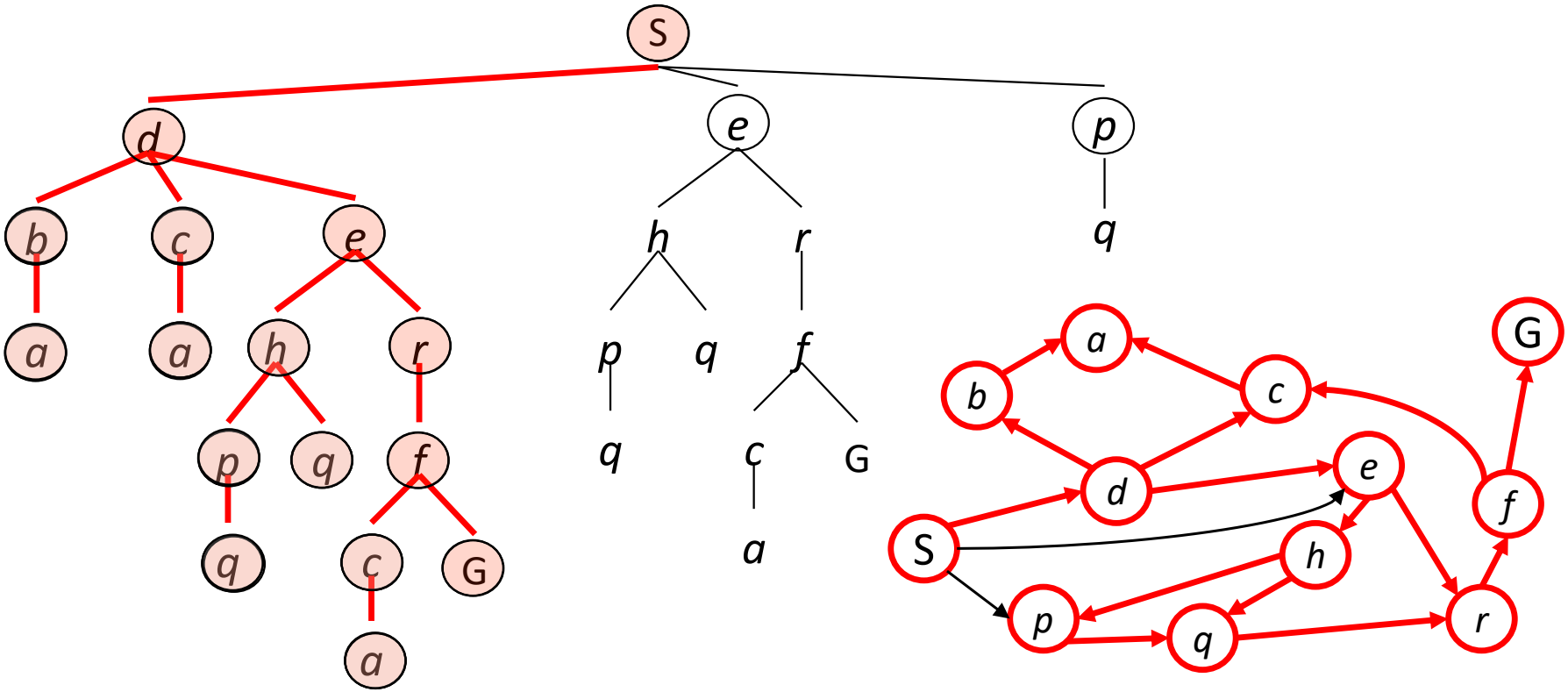
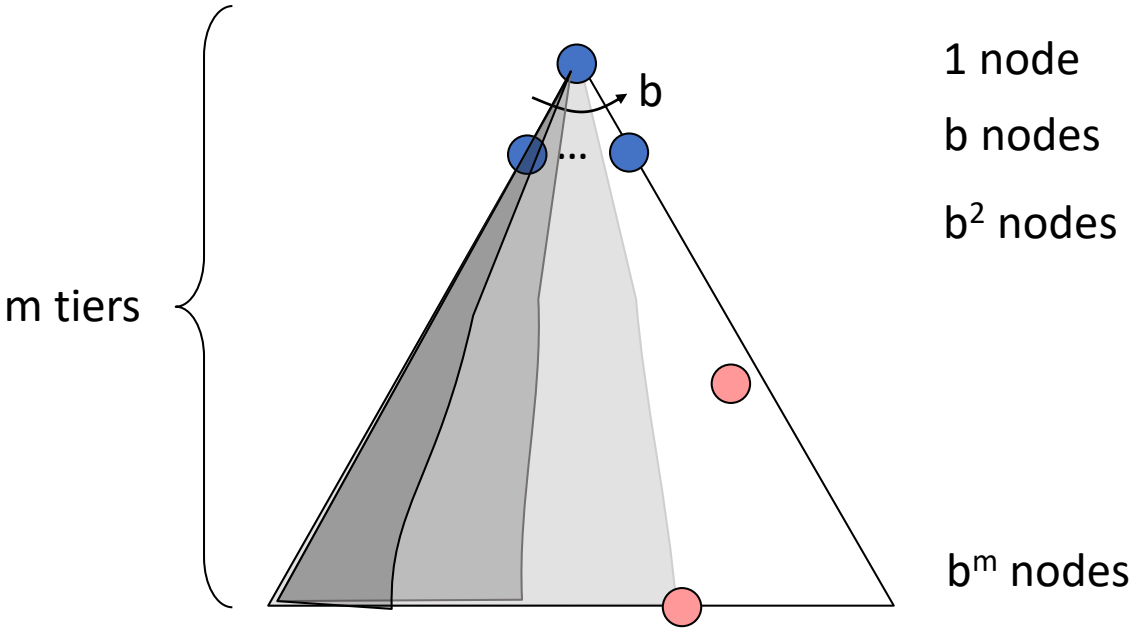


Figure 2: An example of depth first search

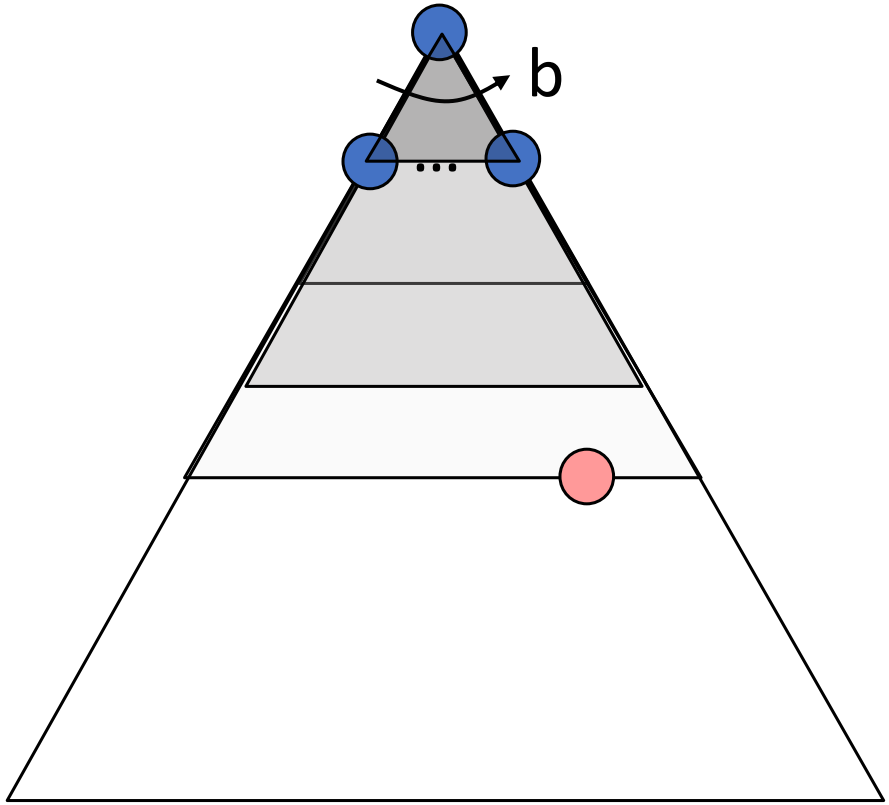
Depth First Search Properties



Reached the cheapest solution with depth s

- Has processed all nodes on the left (left prefix)
- Could have processed the whole tree
- Time complexity: $\mathcal{O}(b^m)$ with finite m
- Space complexity (frontier): $\mathcal{O}(bm)$ (only siblings along path)
- Complete: Yes without cycle
- Optimal: No, 'leftmost'
- Action coat: 0

Depth First Search Properties - continued



In depth first search, s could be infinite (cycles). This causes both time and space issues.

Comprise:Limit search depth, and then increase search depth from such as 2 to the limit. Need to judge if a solution can be found within the limited depth.

Depth First Search (DFS) - graph I

Manchester Metropolitan University

Algorithm 3: Pseudocode of Iterative Deepening Search

Input: Initial state **s**, goal state **g**, a maximum search limit M.
Output: A *node* helps to retrieve a solution (path) $\mathcal{P}$, or failure

```
1:  while m < M do
2:      node ← with s as state
3:      if s == g then
4:          return node
5:      end if
6:      O ← node, an LIFO queue
7:      result ← failure
8:      while O != ∅ do
9:          parent ← the last node in O
10:         v ← parent.state
11:         if v == g then
12:             return parent
```

Depth First Search (DFS) - graph II

Manchester Metropolitan University

```
13:       else if depth >= M then
14:          result = cutoff        % reached search limit.
15:       else if no cycle then
16:          for child in successor (of the current parent) do
17:             add child to the end of 𝒪
18:          end for
19:       end if
20:    end while
21: end while
22: return result
```

BFS vs. DFS

Manchester Metropolitan University

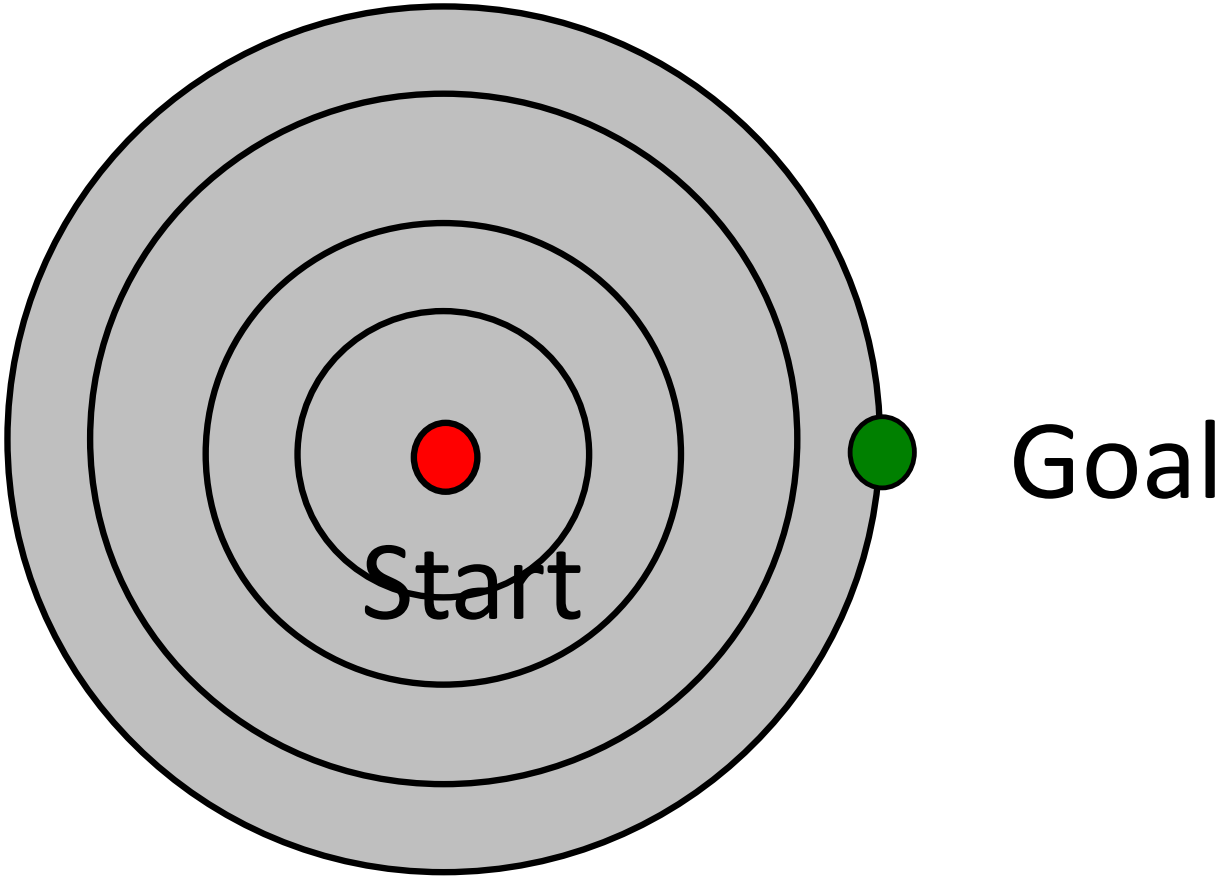


Figure 3: Search Contour of BFS

- Search in all directions
- Edge costs are equal

BFS vs. **DFS**

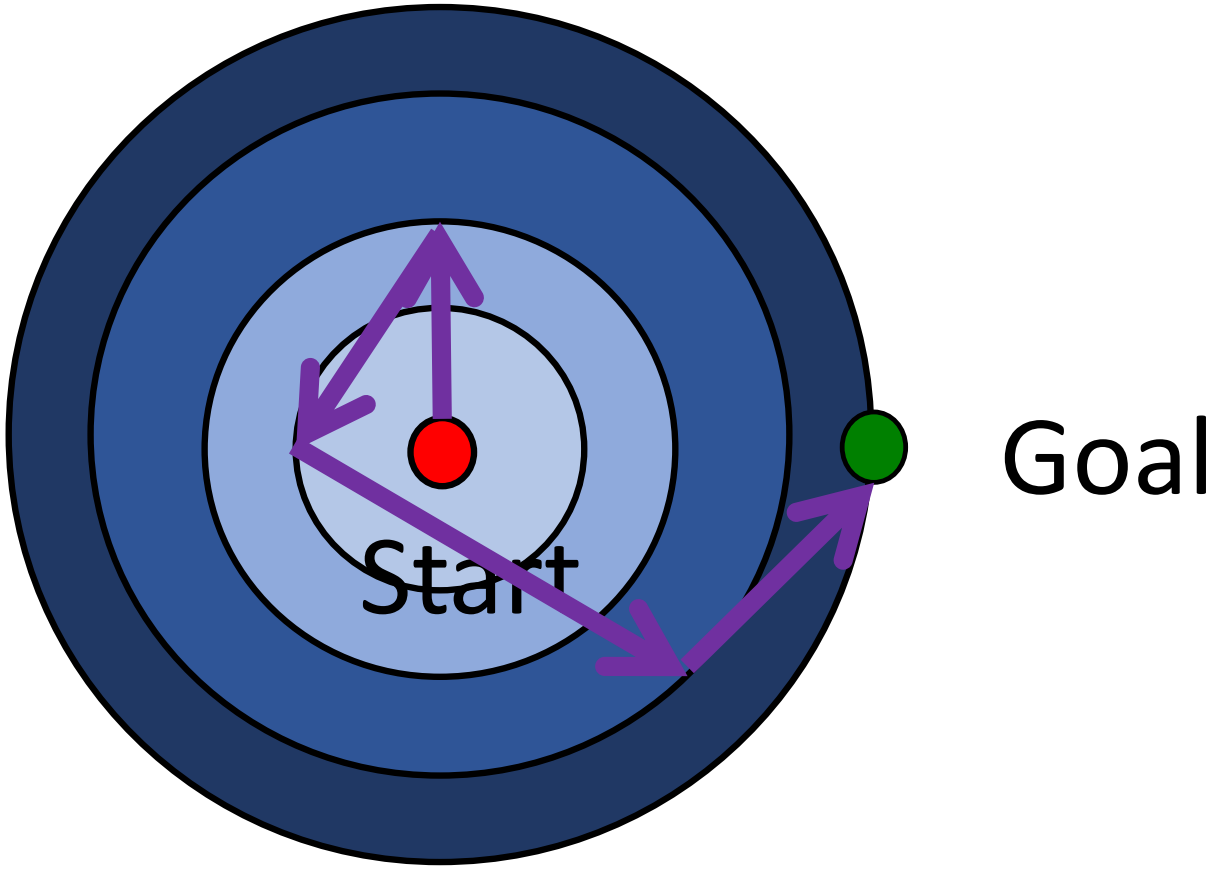


Figure 4: Search Contour of DFS

- Search in a specific direction (depth)
- Edge costs are not prioritised, not as much as depth

Summary

Manchester Metropolitan University

We have learned:

- Depth First Search
- Breadth First Search
- Pros and Cons of each algorithm