PROBLEM SOLVING IN ARTIFICIAL INTELLIGENCE

Hello!

I am Jahanzeb Naeem

1 What is a Problem

Definition of a Problem

- Initial State and Goal State
- Actions (S) → { a_1, a_2, a_3, ……}
- Result (S, a) → S^1
- Goal Test (S) → T | F
- Path Cost $(S \xrightarrow{a} S \xrightarrow{a} S)$ → n Implement as Step Cost (S, a, S^1) → n
- State Space
- Frontier (Furthest Path Explored)
- Explored
- Un-explored

Example

- Route Finding

2

Tree Search

Tree Search

- Function TreeSearch (problem):

```
Frontier = {[initial]}
Loop:
    If frontier is empty: return Fail
    path = remove_choice (frontier)
    s = pathend
    If s is a goal: return path
    for a in actions:
        add [path+a→Result (s,a)] to frontier
```

Tree Search (Cont.)

- Function TreeSearch (problem p) returns path

```
Frontier = {path (p.initial)}
Loop:
        If frontier is empty: return Fail
        path = remove_choice (frontier)
        s = pathend
        If GoalTest (s): return path
        for a in p.Actions (s):
                add [path+a→Result (s,a)] to frontier
```

3

Breadth-First Search

Tree Search (Cont.) - BFS

- Function **Graph**Search (problem):

  Frontier = {[initial]} **; explored = {}**

  Loop:

  If frontier is empty: return Fail

  path = remove_choice (frontier)

  s = pathend **; add s to explored**

  If s is a goal: return path

  for a in actions:

  add [path+a→Result (s,a)] to frontier

  unless Result (s,a) in frontier + explored

Tree Search (Cont.) - BFS

- Works on same level first

4

Uniform-Cost Search (Cheapest-First)

Tree Search (Cont.) - CFS

- Works on weightage of nodes

5 Depth-First Search

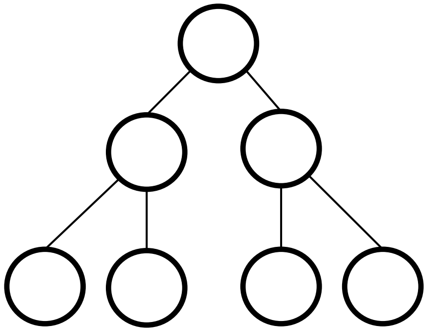
Tree Search (Cont.) - DFS

- Works on depth first

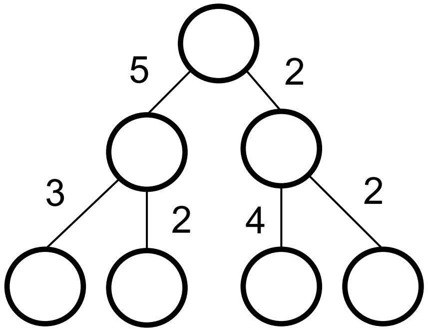
6 Search Comparison

Search Comparison

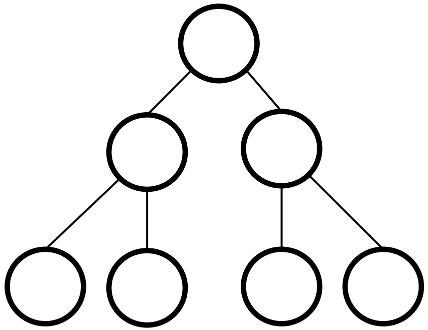
Breadth-First



Cheapest-First



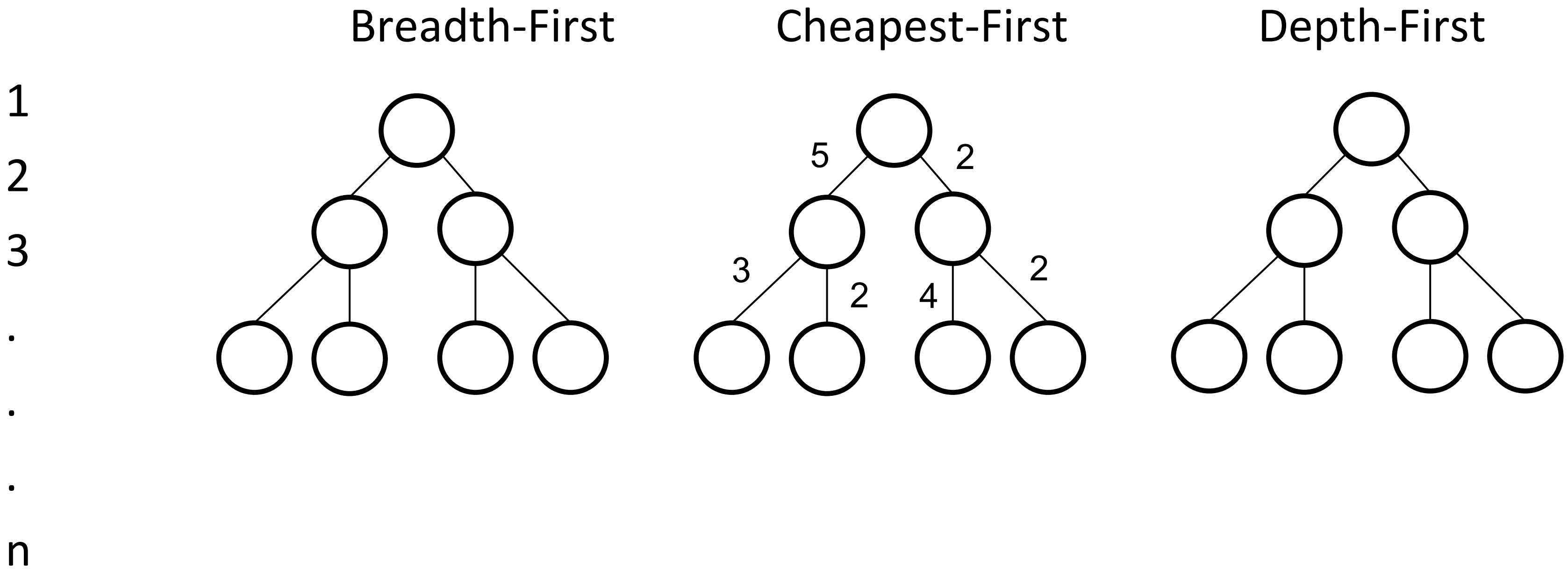
Depth-First



	Breadth-First	Cheapest-First	Depth-First
Optimal:	Yes	Yes	No

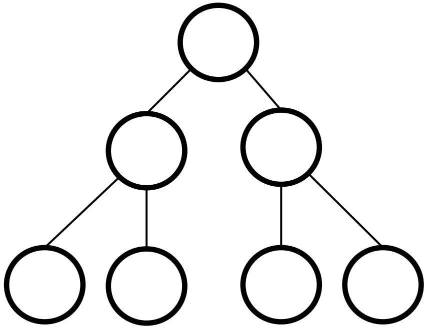
Search Comparison (Cont.)

Why to use Depth-First?

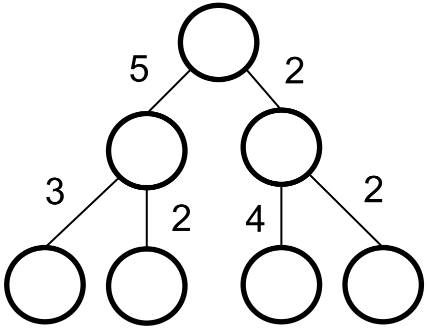


Search Comparison (Cont.)

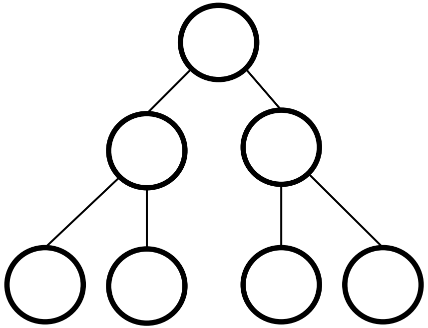
Breadth-First



Cheapest-First



Depth-First



	Breadth-First	Cheapest-First	Depth-First
Completeness:	Yes	Yes	No

7

Greedy Best-First Search

Greedy Best-First Search

- Expand paths directed toward the goal
- What if an obstacle is present?

8

A* Search

A* Search

- f = g + h
  - g (path) = path cost
  - h (path) = h (s) = estimated distance to goal
- A* finds lowest cost path if:
  - h (s) < true cost
  - h never over estimate the distance to goal
  - h optimistic
  - h admissible

Optimistic Heuristics

- Optimistic h Finds lowest-cost path
  - $f = g + h$
  - $h(s) < \text{true cost}$
  - $S \rightarrow G$

9 State Space

10 Examples

Optimistic Heuristics

- Robot Cleaning a House
- Sliding Block Puzzle (15 Puzzle)
  - h_1 = #misplaced blocks
  - h_2 = sum (distances of blocks)
  - A block can move from A→B
    - if (A adjacent to B) = h_1
    - and (B is blank) = h_2
  - $h = \max(h_1, h_2)$

11

Problems with Search

Problem Solving

- Problem-Solving works when:
  - Fully Observable
  - Known
  - Discrete
  - Deterministic
  - Static