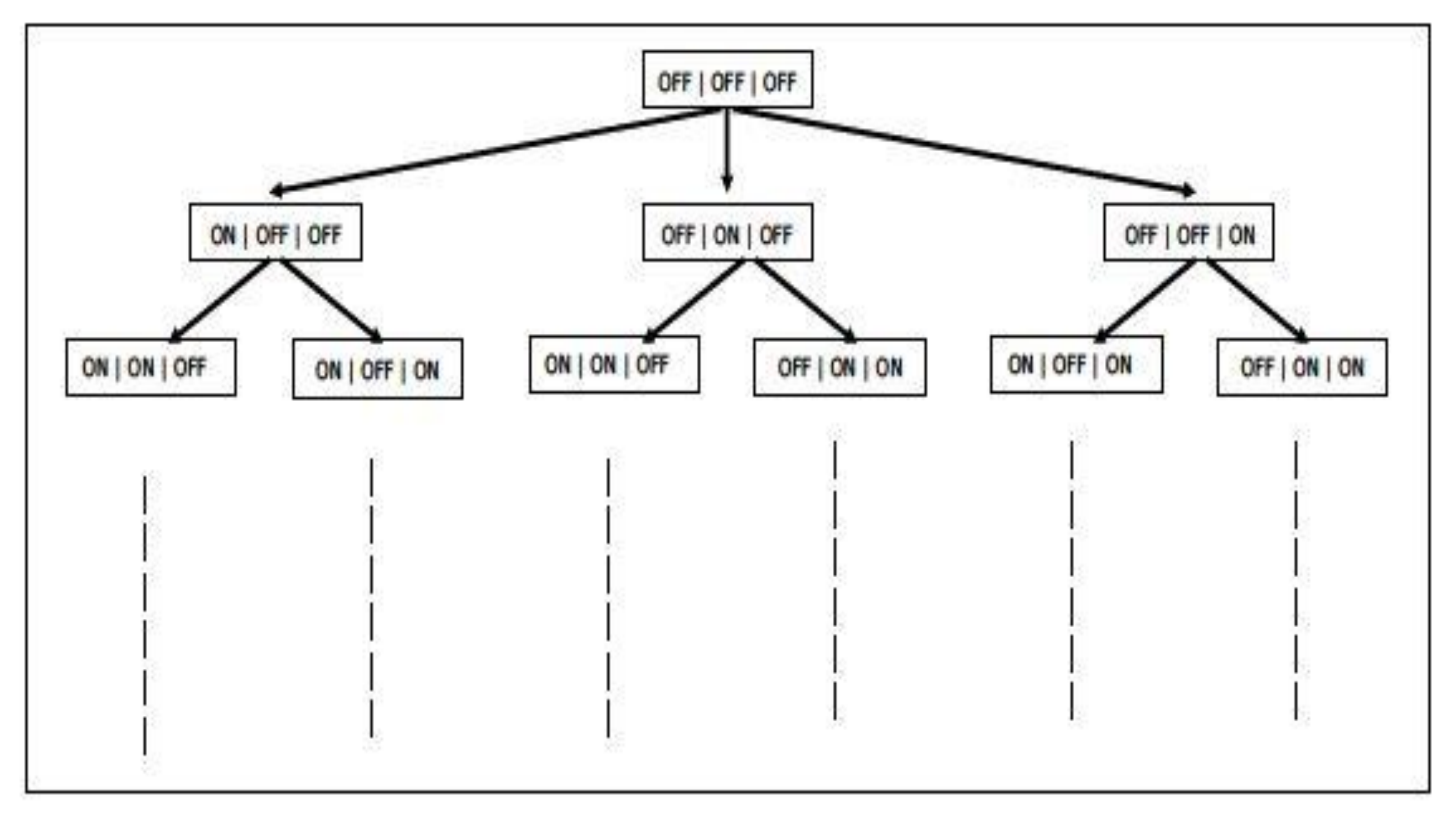
Problem Solving

Fox, Lamb and Grains Problem

Problem Solving

Toddle problem



Two – One Problem

Start

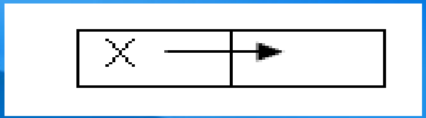
1 1 ? 2 2

Goal

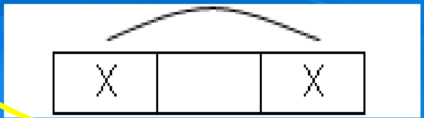
2 2 ? 1 1

Rules:
- 1s’ move right
- 2s’ move left
- Only one move at a time
- No backing up

Legal Moves:
- Slide



- Hop



Two – One Problem
Trials to solve the problem

Trial One

1 1 ? 2 2

1 1 2 ? 2

1 ? 2 1 2

1 2 ? 1 2

1 2 2 1 ?

1 2 2 ? 1

Stuck!!!

Trial Two

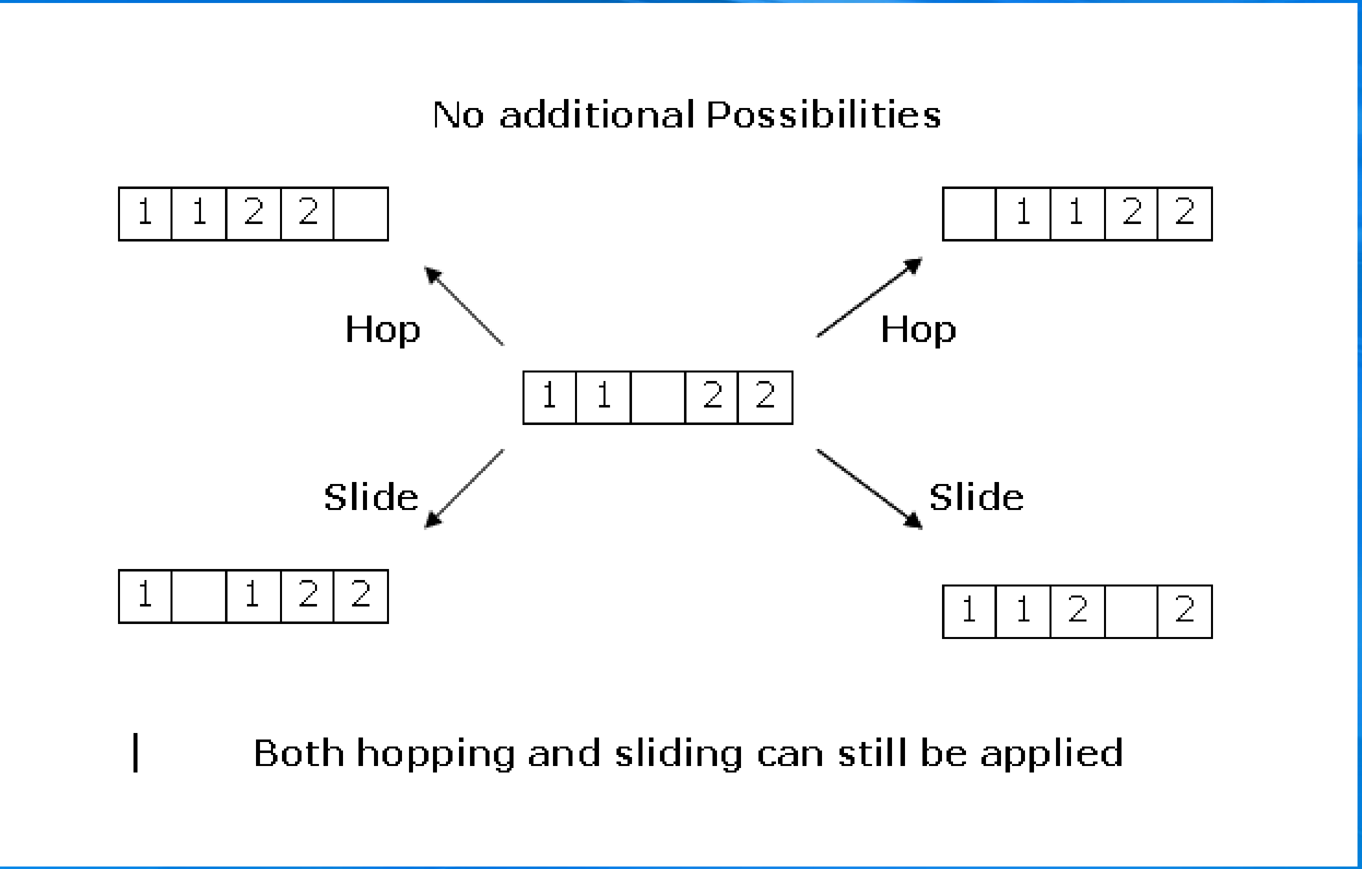
1 1 ? 2 2

1 ? 1 2 2

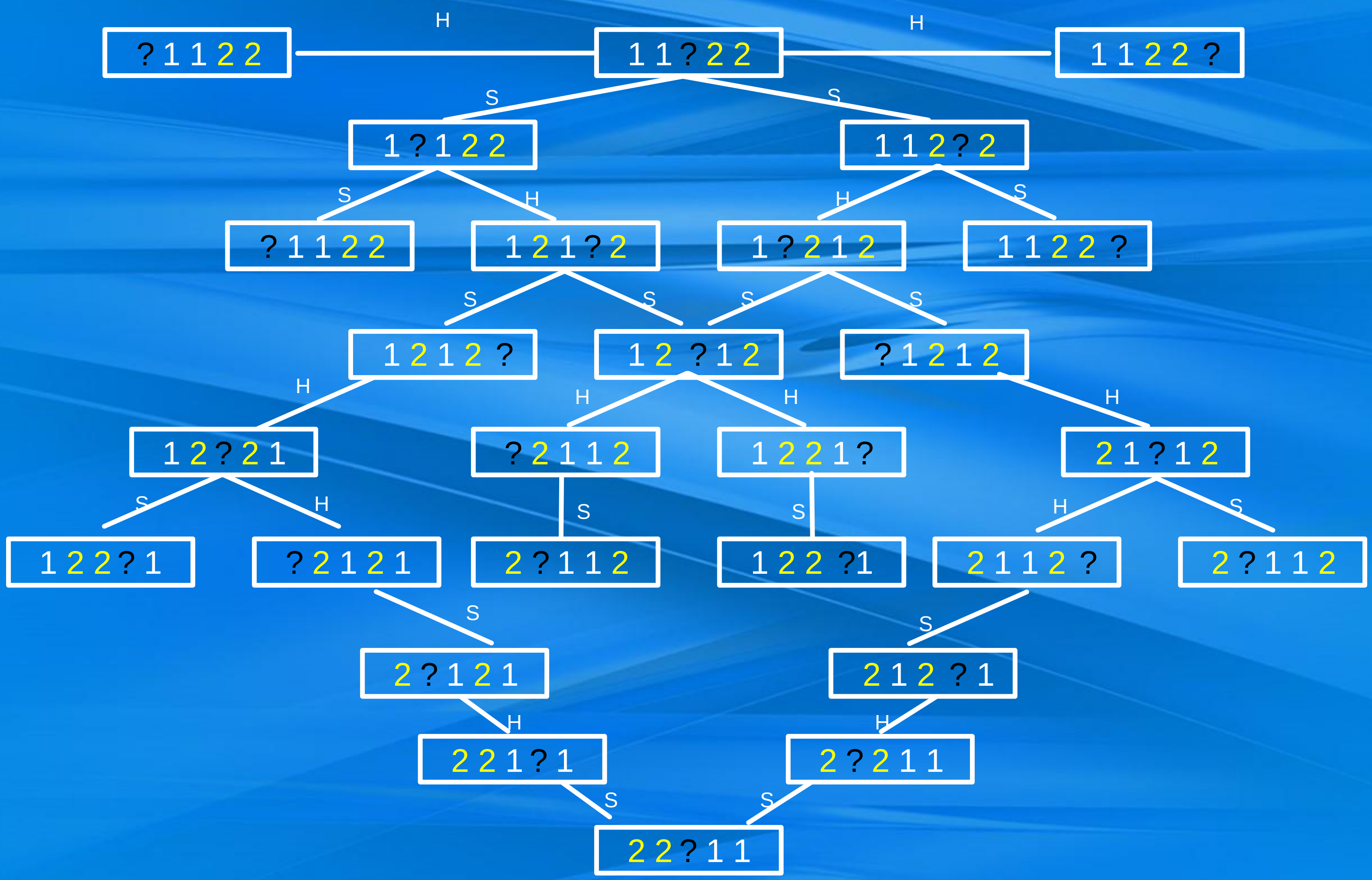
? 1 1 2 2

Stuck!!!

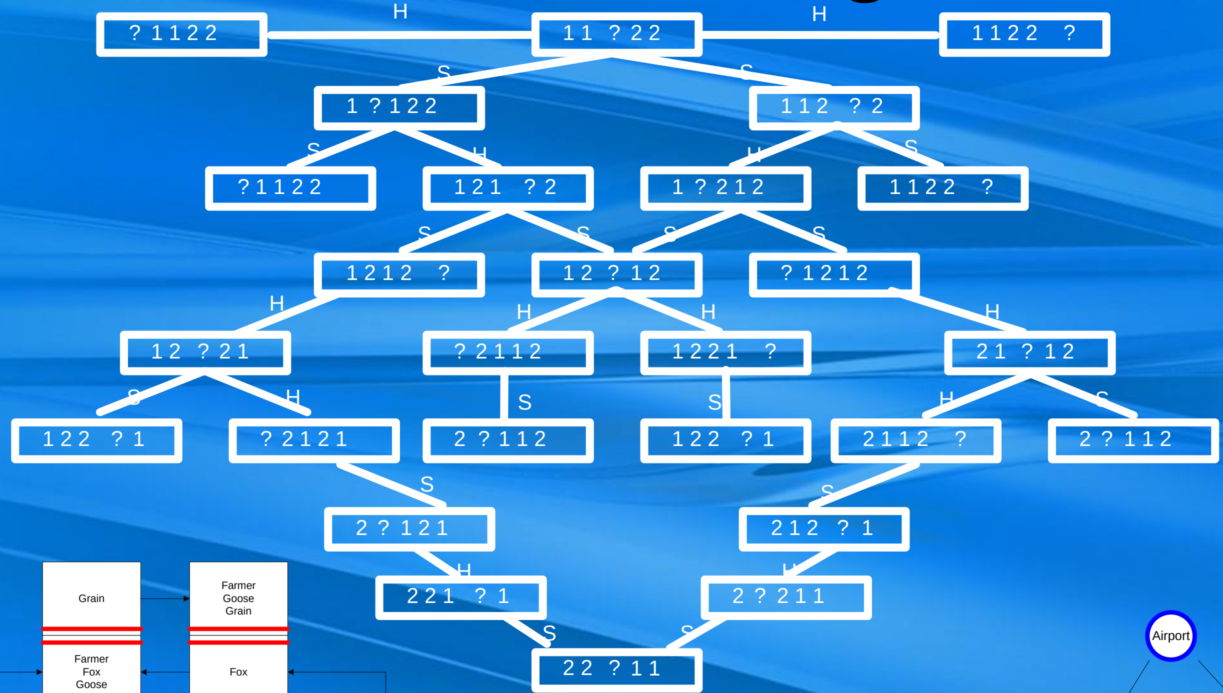
Two – One Problem
Five States

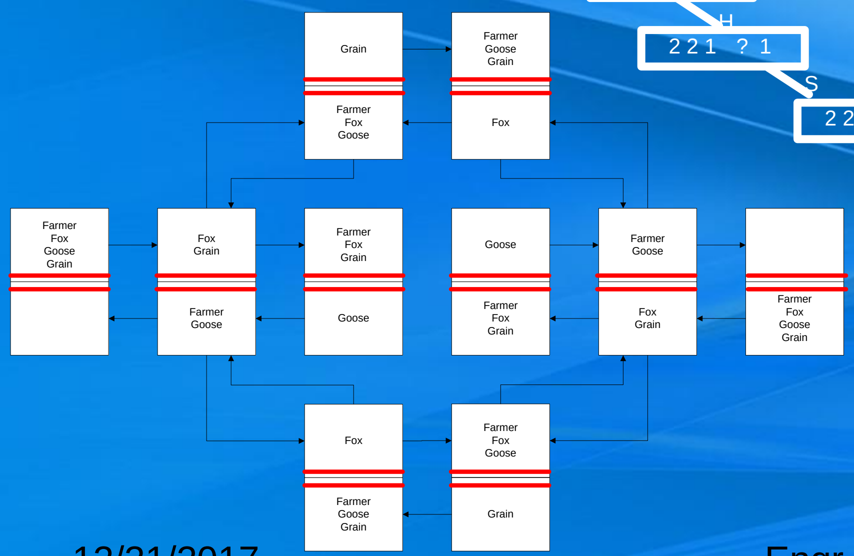


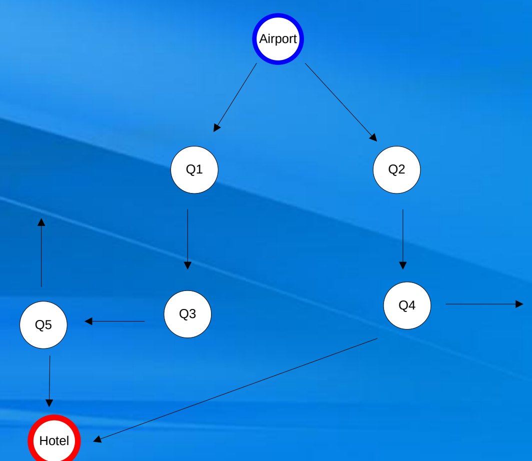
Two – One Problem
Solution Space



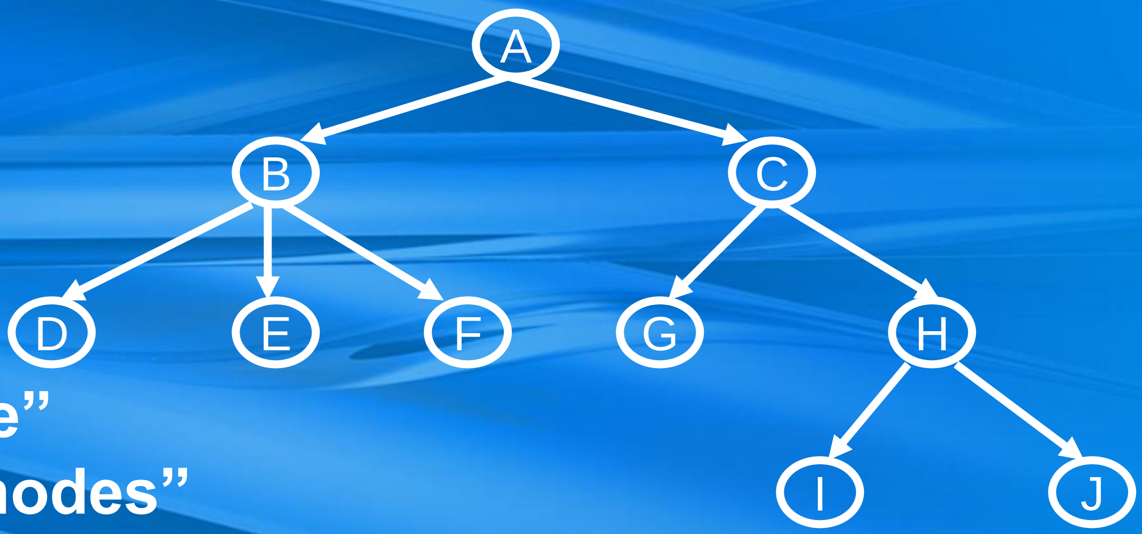
Solution to Problem Solving : Searching





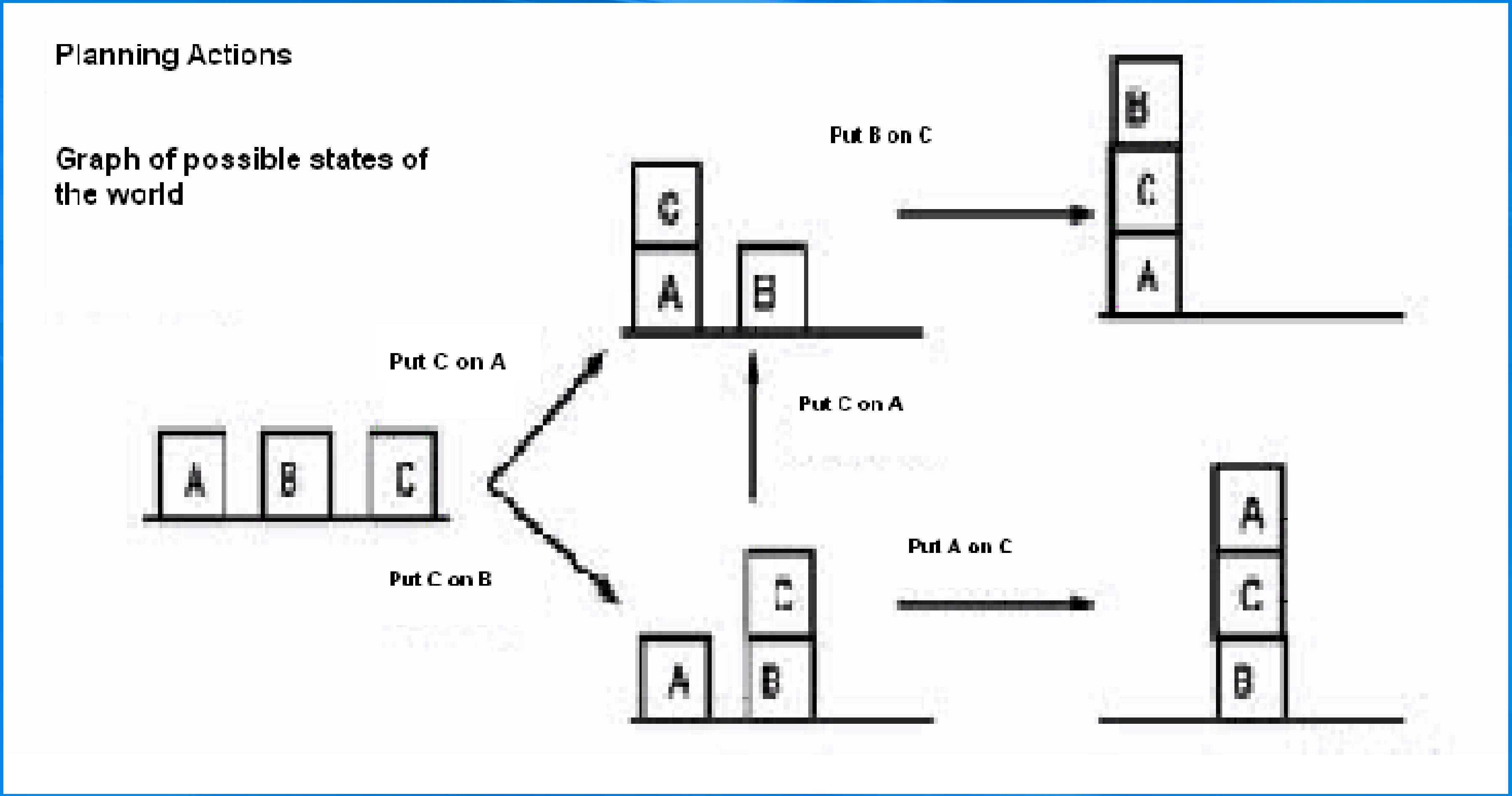


Tree and Graph Terminology



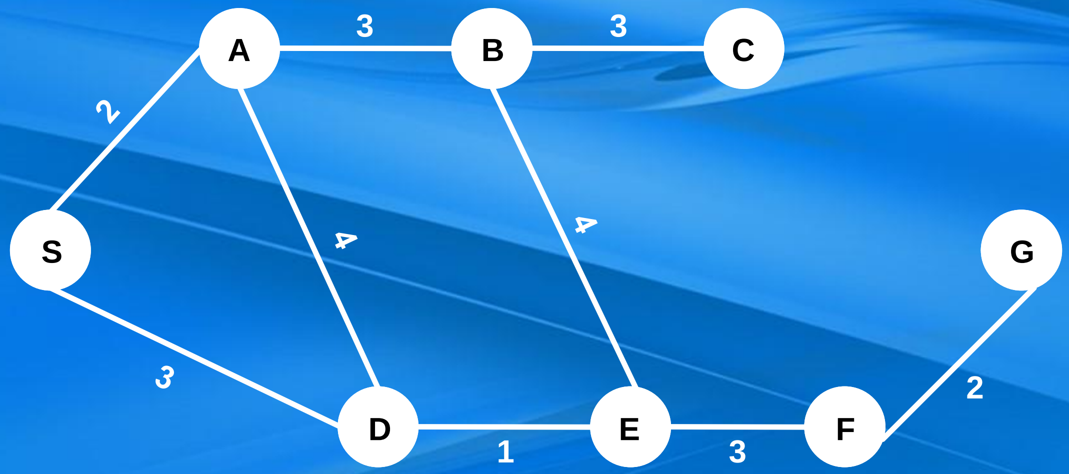
- "A" is the "root node"
- "A, B, C …. J" are "nodes"
- "B" is a "child" of "A"
- "A" is ancestor of "D"
- "D" is a descendant of "A"
- "D, E, F, G, I, J" are "leaf nodes"
- Arrows represent "edges" or "links"

Examples of Graphs



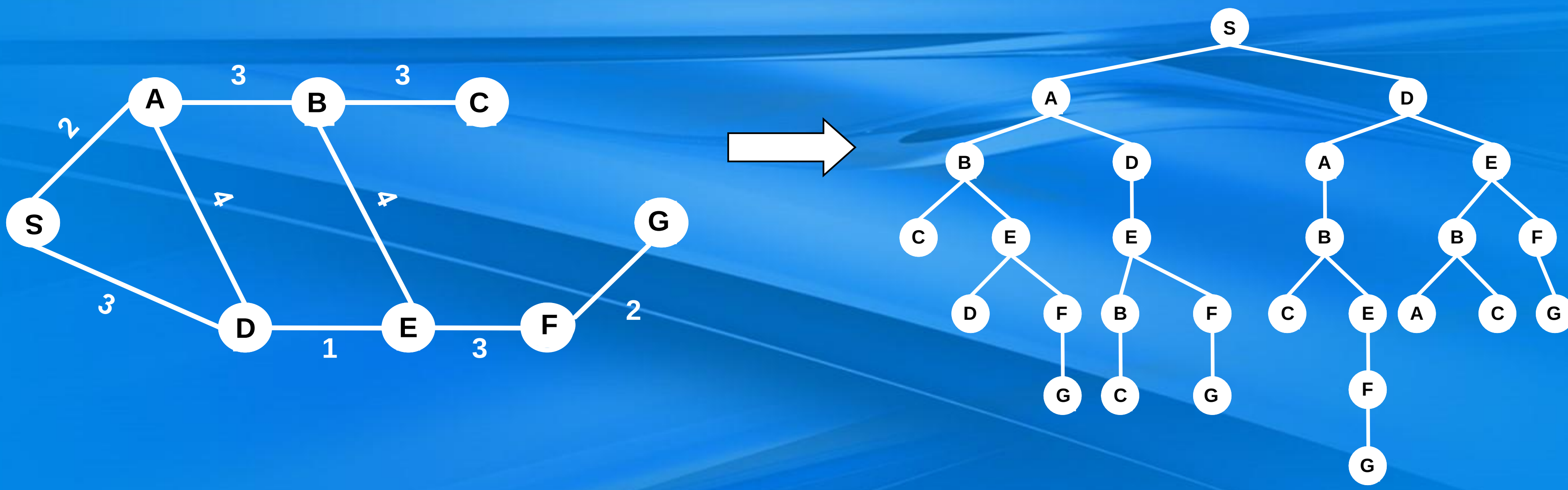
Problem Formulation using Graphs

- **The search methods we'll be dealing with are defined on trees and graphs**



Tree Search

- **Graph search is really tree search**



Simple Search Algorithm

Let S be the start state

1. **Initialize Q with the start node Q=(S) as only entry; set Visited = (S)**
2. **If Q is empty, fail. Else pick node X from Q**
3. **If X is a goal, return X, we've reached the goal**
4. **(Otherwise) Remove X from Q**
5. **Find all the children of node X not in Visited**
6. **Add these to Q; Add Children of X to Visited**
7. **Go to Step 2**

DFS

- Depth First Search dives into a tree deeper and deeper to fine the goal state. We will use the same Simple Search Algorithm to implement DFS by keeping our priority function as

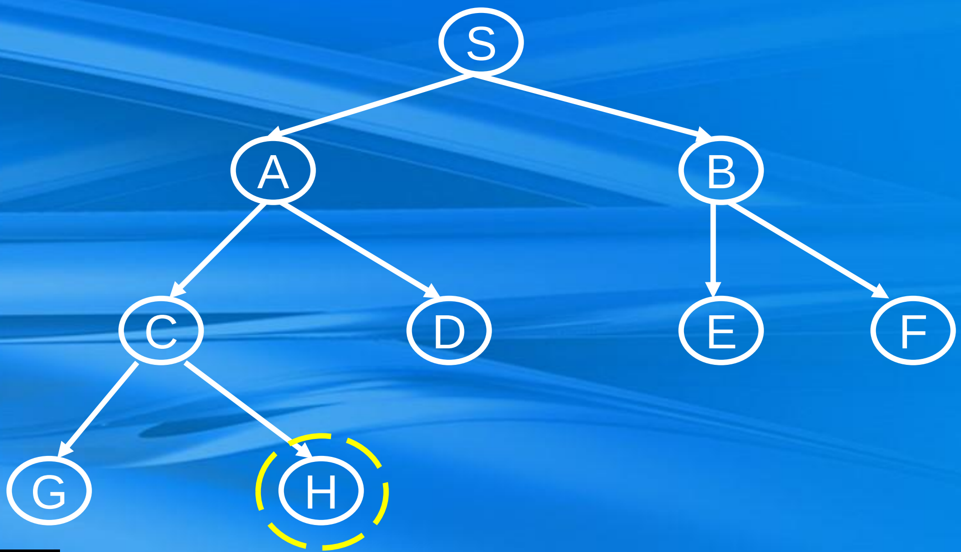
$$P(n) = \frac{1}{height(n)}$$

Simple Search Algorithm

Let S be the start state

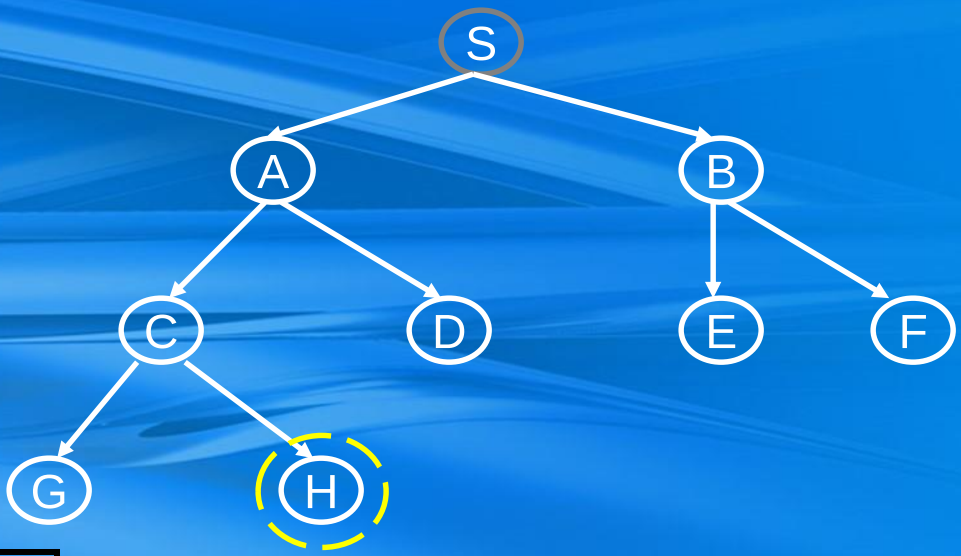
1. **Initialize Q with the start node Q=(S) as only entry; set Visited = (S)**
2. **If Q is empty, fail. Else pick node X from Q**
3. **If X is a goal, return X, we've reached the goal**
4. **(Otherwise) Remove X from Q**
5. **Find all the children of node X not in Visited**
6. **Add these to Q; Add Children of X to Visited**
7. **Go to Step 2**

DFS: Example



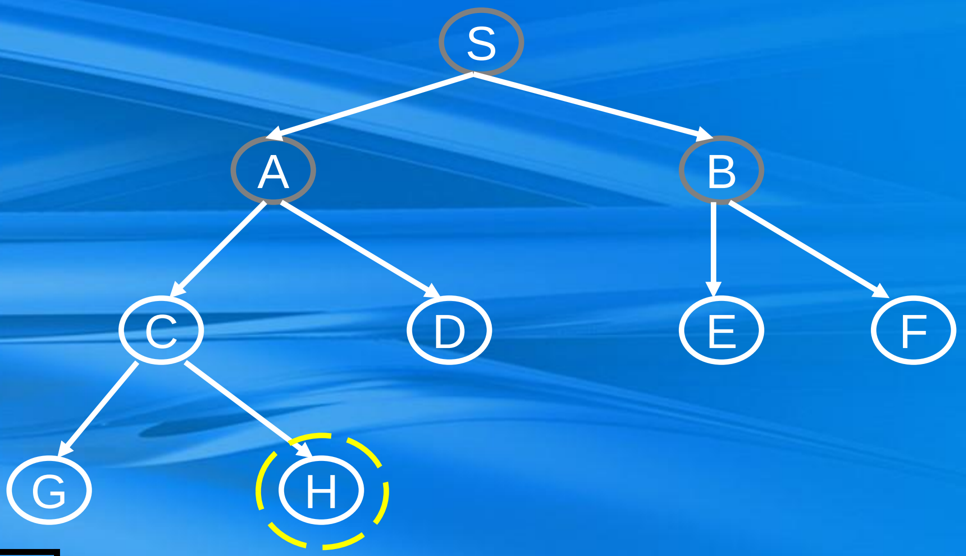
	Q	Visited
1		
2		
3		
4		
5		

DFS: Example



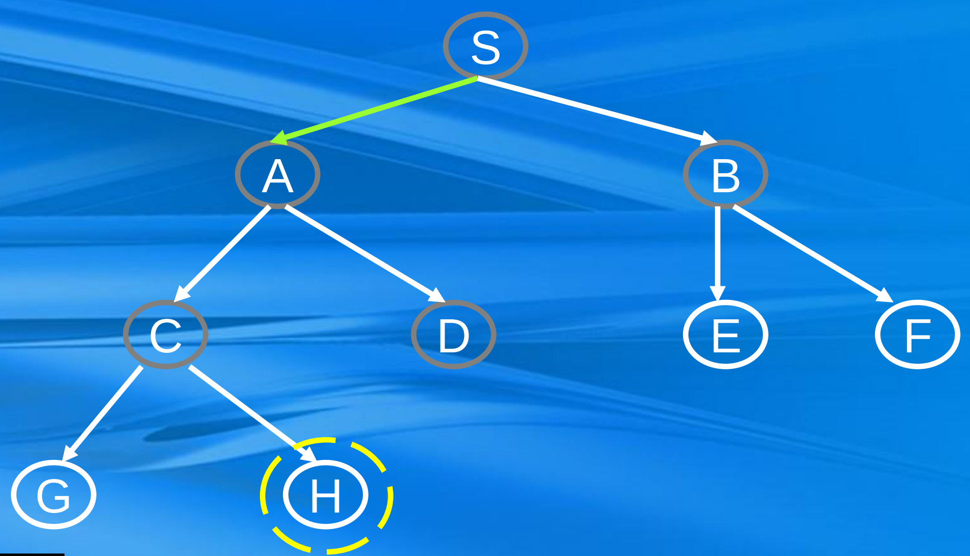
	Q	Visited
1	S	S
2		
3		
4		
5		

DFS: Example



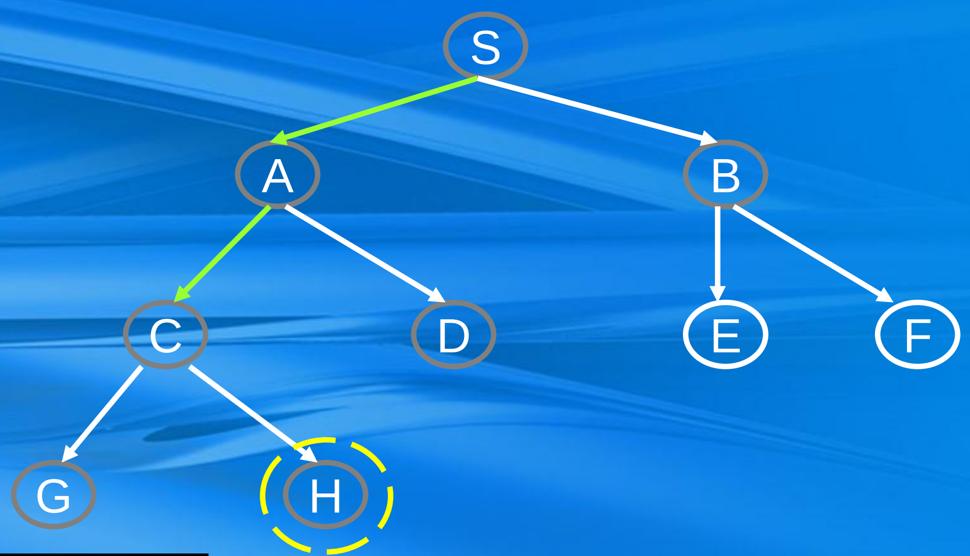
	Q	Visited
1	S	S
2	A,B	S,A,B
3		
4		
5		

DFS: Example



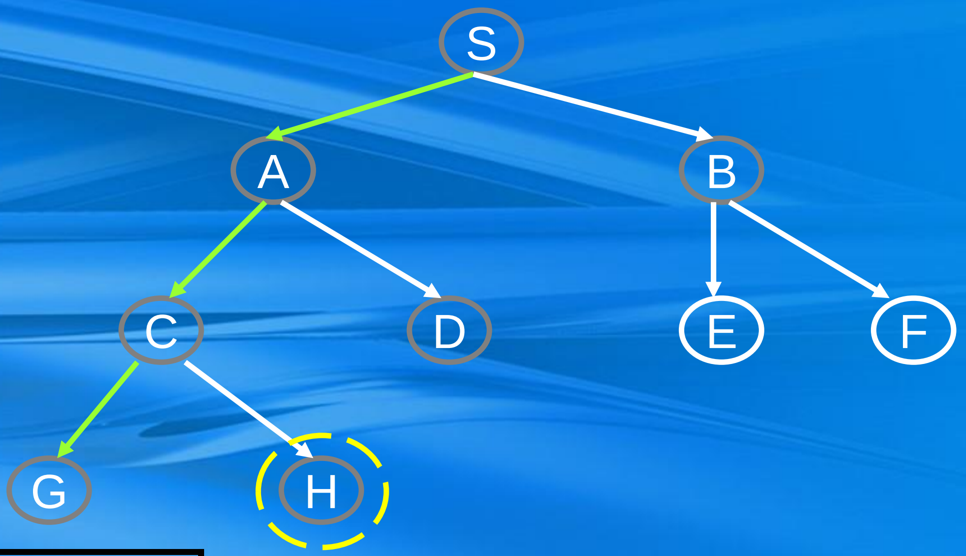
	Q	Visited
1	S	S
2	A,B	S,A,B
3	C,D,B	S,A,B,C,D
4		
5		

DFS: Example



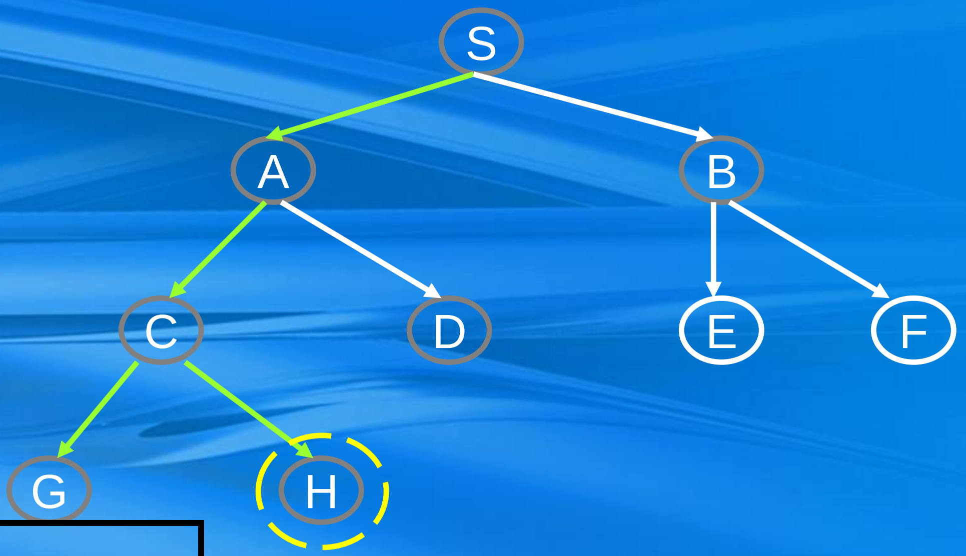
	Q	Visited
1	S	S
2	A,B	S,A,B
3	C,D,B	S,A,B,C,D
4	G,H,D,B	S,A,B,C,D,G,H
5		

DFS: Example



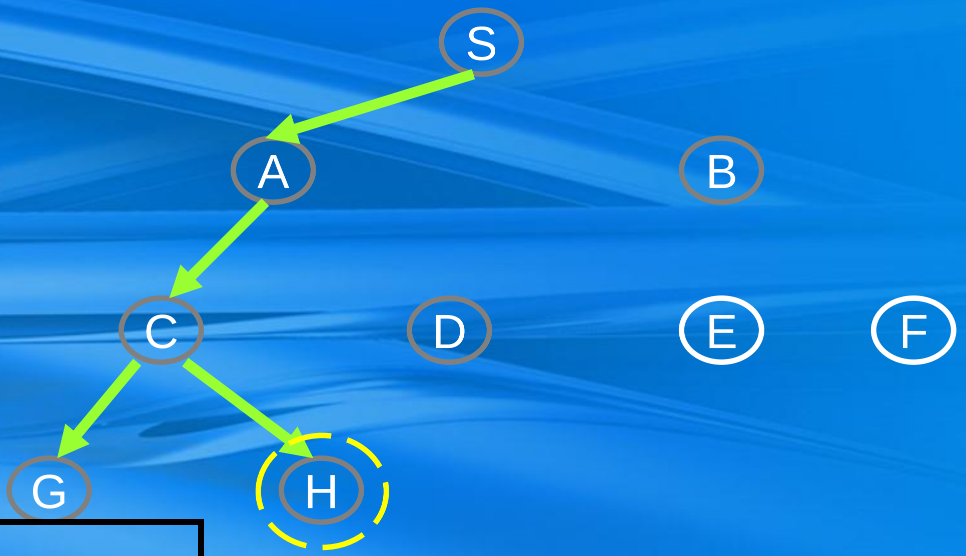
	Q	Visited
1	S	S
2	A,B	S,A,B
3	C,D,B	S,A,B,C,D
4	G,H,D,B	S,A,B,C,D,G,H
5	H,D,B	S,A,B,C,D,G,H

DFS: Example



	Q	Visited
1	S	S
2	A,B	S,A,B
3	C,D,B	S,A,B,C,D
4	G,H,D,B	S,A,B,C,D,G,H
5	H,D,B	S,A,B,C,D,G,H
6	D,B	S,A,B,C,D,G,H

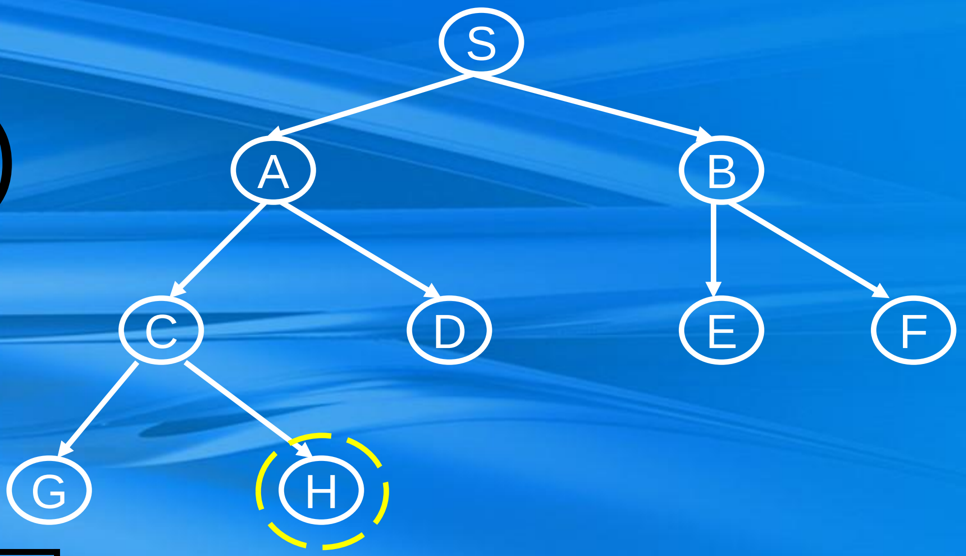
DFS: Example



	Q	Visited
1	S	S
2	A,B	S,A,B
3	C,D,B	S,A,B,C,D
4	G,H,D,B	S,A,B,C,D,G,H
5	H,D,B	S,A,B,C,D,G,H
6	D,B	S,A,B,C,D,G,H

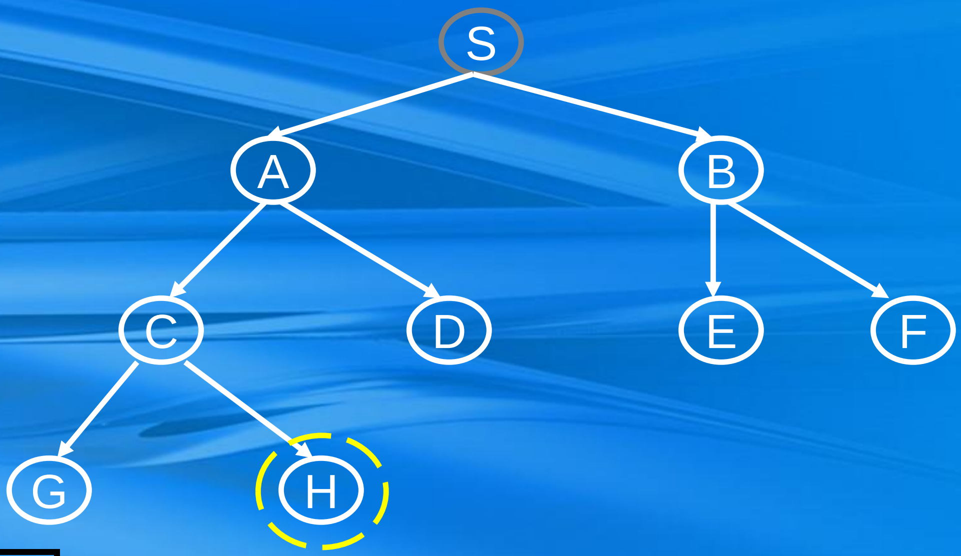
BFS: Example

P(n) = height (n)



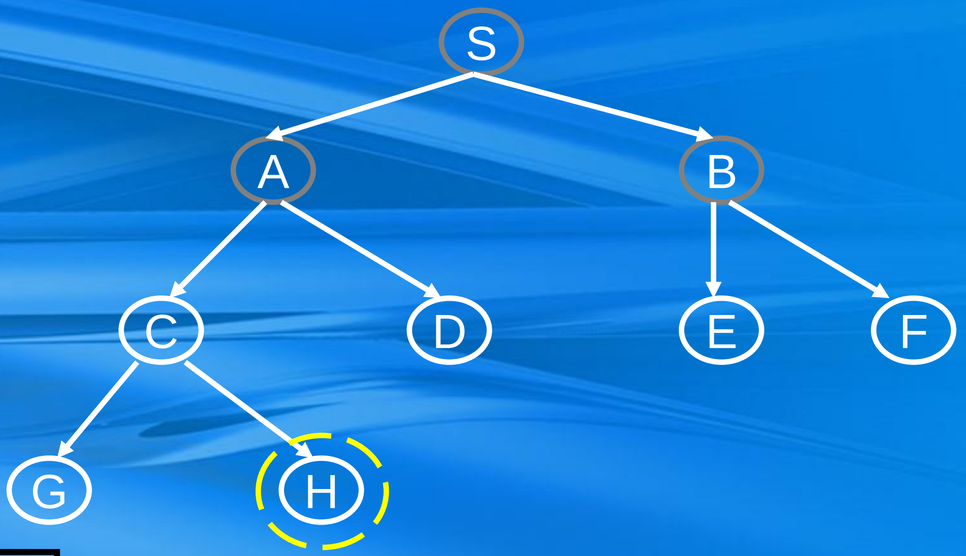
	Q	Visited
1		
2		
3		
4		
5		

BFS: Example



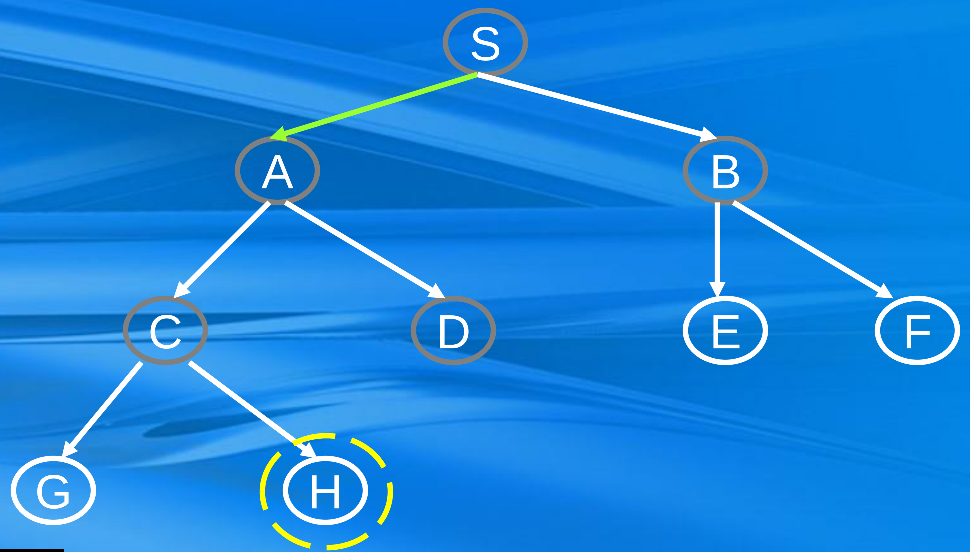
	Q	Visited
1	S	S
2		
3		
4		
5		

BFS: Example



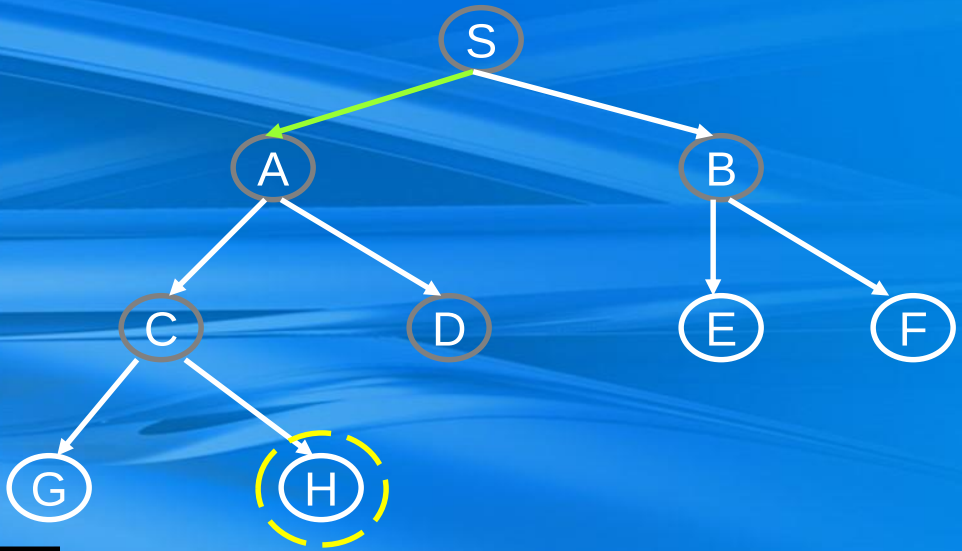
	Q	Visited
1	S	S
2	A,B	S,A,B
3		
4		
5		

BFS: Example



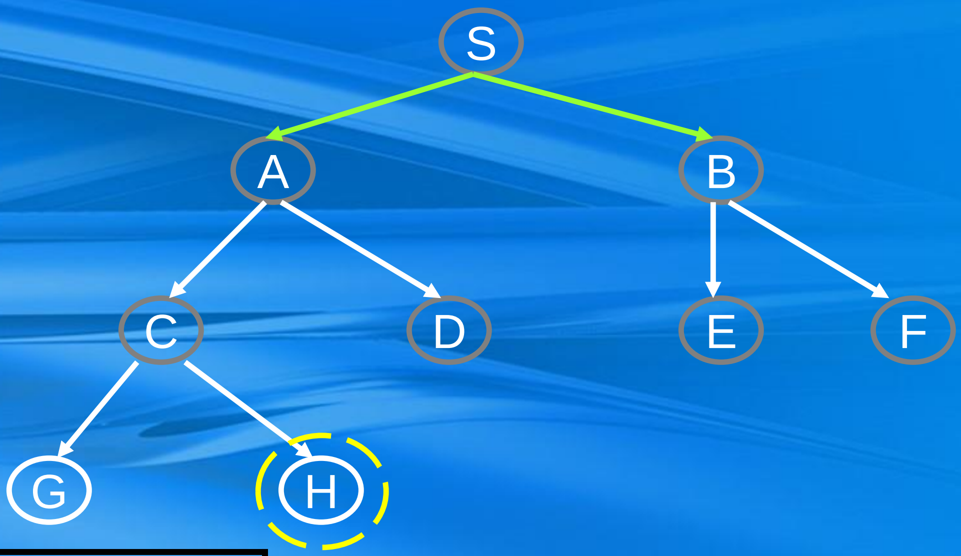
	Q	Visited
1	S	S
2	A,B	S,A,B
3	B,C,D	S,A,B,C,D
4		
5		

BFS: Example



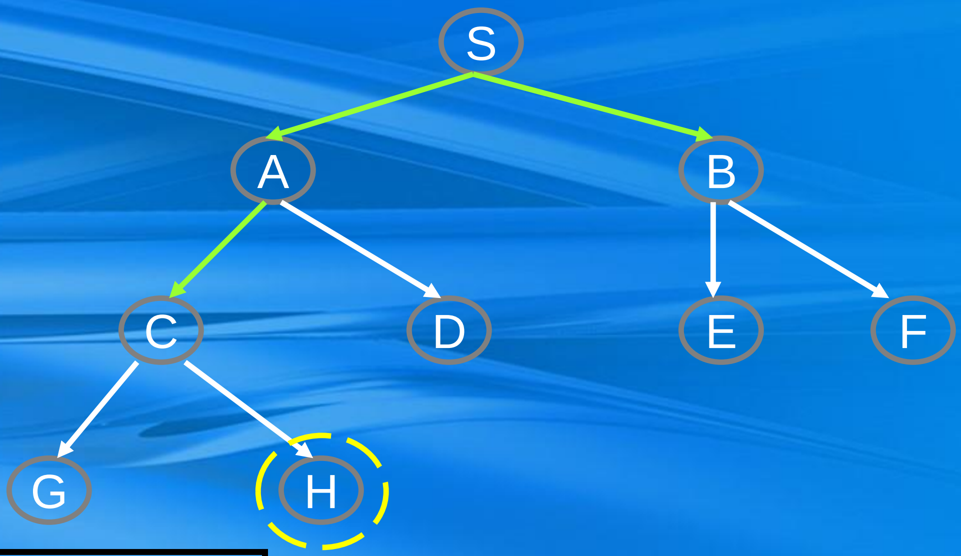
	Q	Visited
1	S	S
2	A,B	S,A,B
3	B,C,D	S,A,B,C,D
4		
5		

BFS: Example



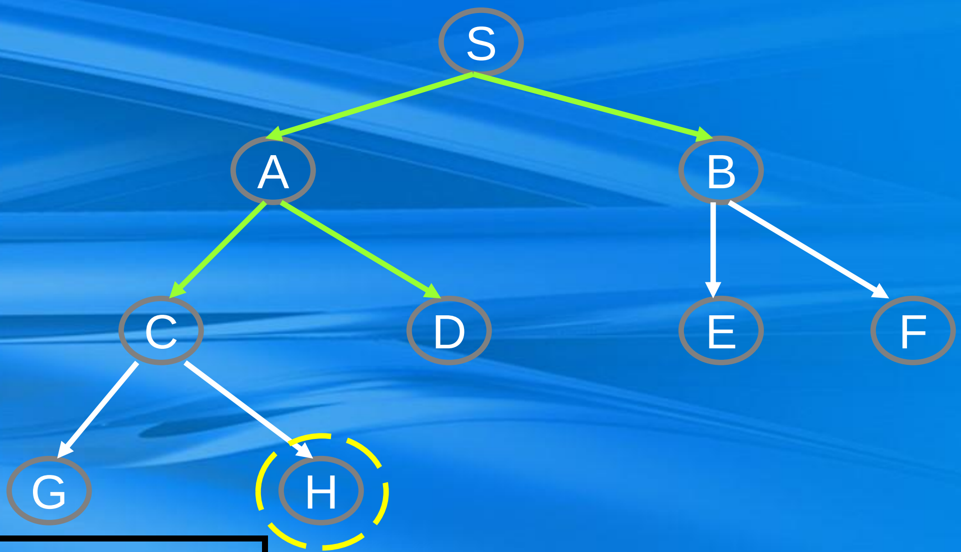
	Q	Visited
1	S	S
2	A,B	S,A,B
3	B,C,D	S,A,B,C,D
4	C,D,E,F	S,A,B,C,D,E,F
5		

BFS: Example



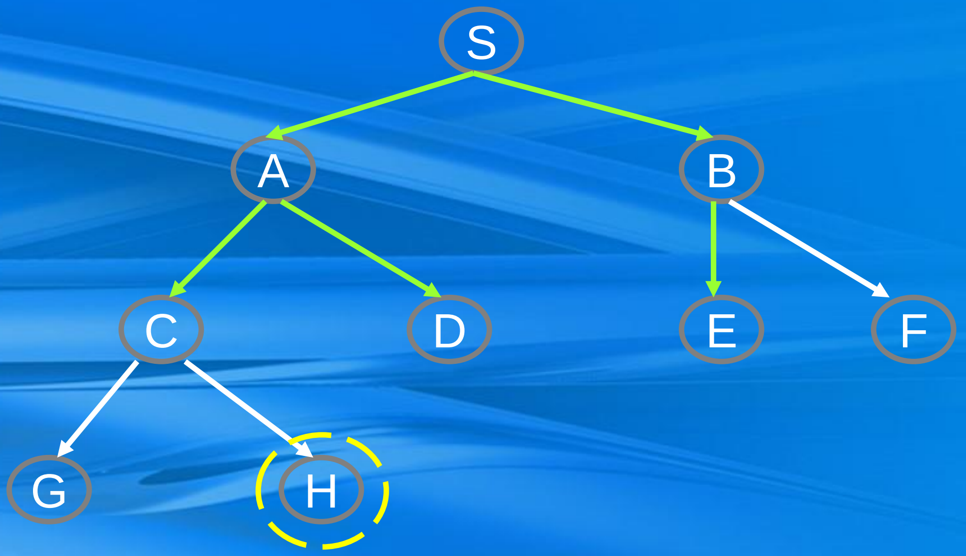
	Q	Visited
2	A,B	S,A,B
3	B,C,D	S,A,B,C,D
4	C,D,E,F	S,A,B,C,D,E,F
5	D,E,F,G,H	S,A,B,C,D,E,F,G,H
6		

BFS: Example



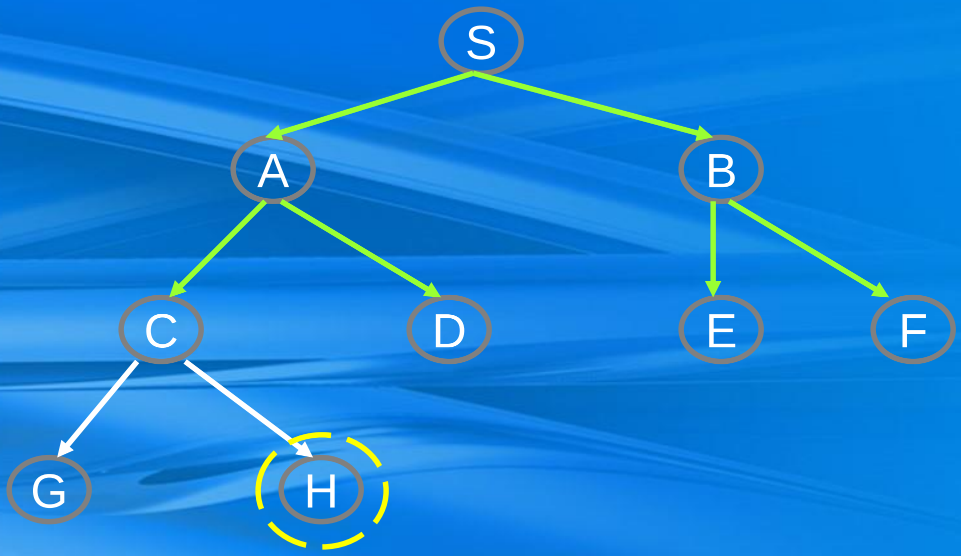
	Q	Visited
3	B,C,D	S,A,B,C,D
4	C,D,E,F	S,A,B,C,D,E,F
5	D,E,F,G,H	S,A,B,C,D,E,F,G,H
6	E,F,G,H	S,A,B,C,D,E,F,G,H
7		

BFS: Example



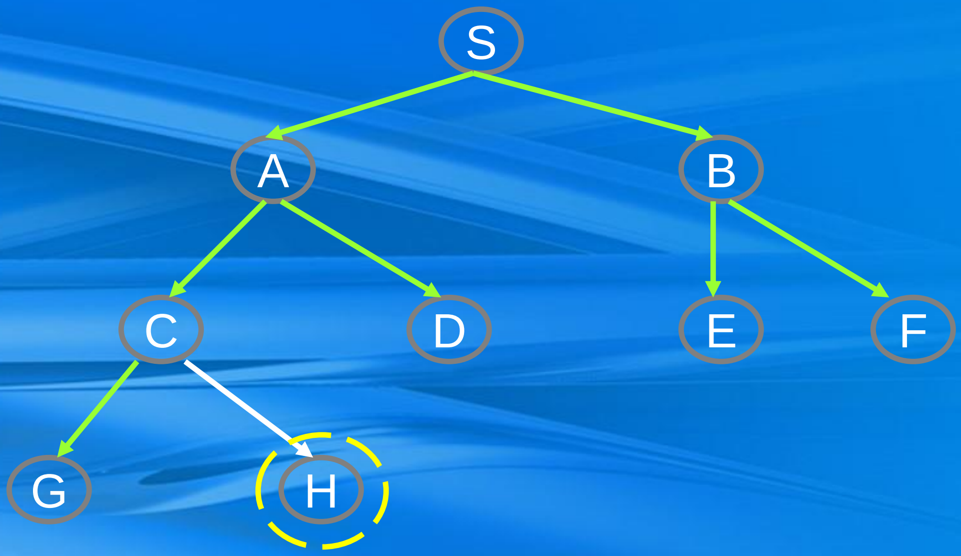
	Q	Visited
4	C,D,E,F	S,A,B,C,D,E,F
5	D,E,F,G,H	S,A,B,C,D,E,F,G,H
6	E,F,G,H	S,A,B,C,D,E,F,G,H
7	F,G,H	S,A,B,C,D,E,F,G,H
8		

BFS: Example



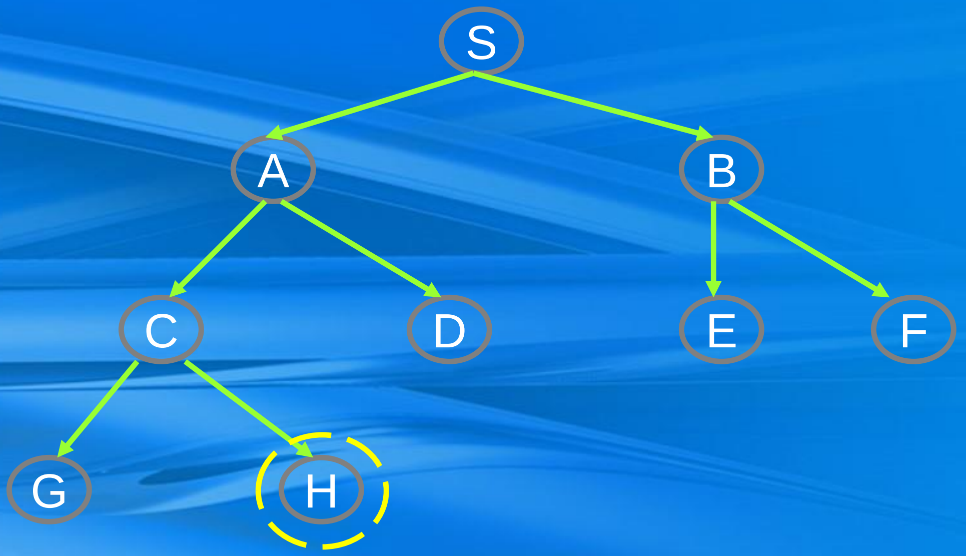
	Q	Visited
5	D,E,F,G,H	S,A,B,C,D,E,F,G,H
6	E,F,G,H	S,A,B,C,D,E,F,G,H
7	F,G,H	S,A,B,C,D,E,F,G,H
8	G,H	S,A,B,C,D,E,F,G,H
9		

BFS: Example



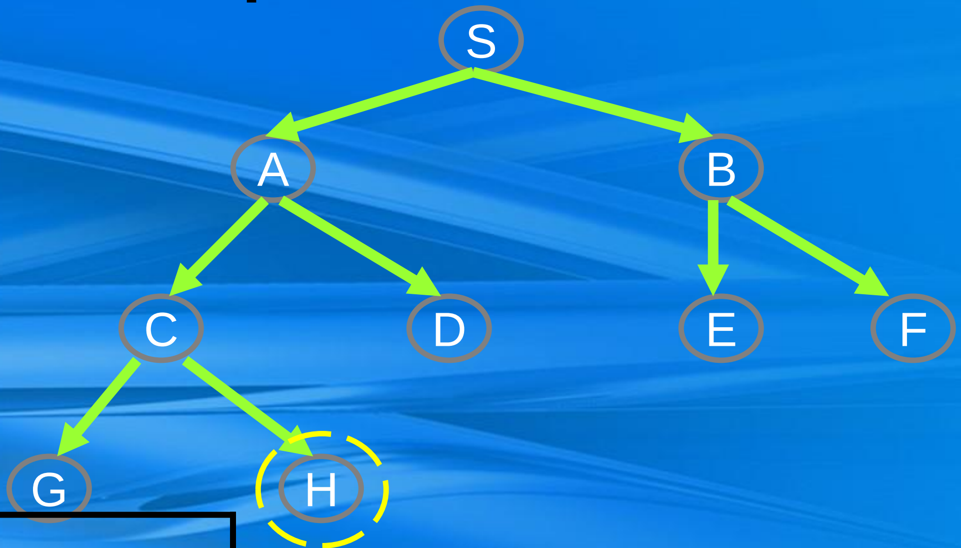
	Q	Visited
6	E,F,G,H	S,A,B,C,D,E,F,G,H
7	F,G,H	S,A,B,C,D,E,F,G,H
8	G,H	S,A,B,C,D,E,F,G,H
9	H	S,A,B,C,D,E,F,G,H
10		

BFS: Example



	Q	Visited
6	E,F,G,H	S,A,B,C,D,E,F,G,H
7	F,G,H	S,A,B,C,D,E,F,G,H
8	G,H	S,A,B,C,D,E,F,G,H
9		S,A,B,C,D,E,F,G,H
10		S,A,B,C,D,E,F,G,H

BFS: Example



	Q	Visited
1	S	S
2	A,B	S,A,B
3	B,C,D	S,A,B,C,D
4	C,D,E,F	S,A,B,C,D,E,F
5	D,E,F,G,H	S,A,B,C,D,E,F,G,H
6	E,F,G,H	S,A,B,C,D,E,F,G,H
7	F,G,H	S,A,B,C,D,E,F,G,H
8	G,H	S,A,B,C,D,E,F,G,H
9	H	S,A,B,C,D,E,F,G,H
10		S,A,B,C,D,E,F,G,H

Problem with BFS

- **Imagine searching a tree with branching factor 8 and depth 10. Assume a node requires just 8 bytes of storage. The breadth first search might require up to:**

$= (8)^{10}$ nodes

$= (2^3)^{10} \text{ X } 2^3 = 2^{33}$ bytes

= 8,000 Mbytes

= 8 Gbytes

Problems with DFS and BFS

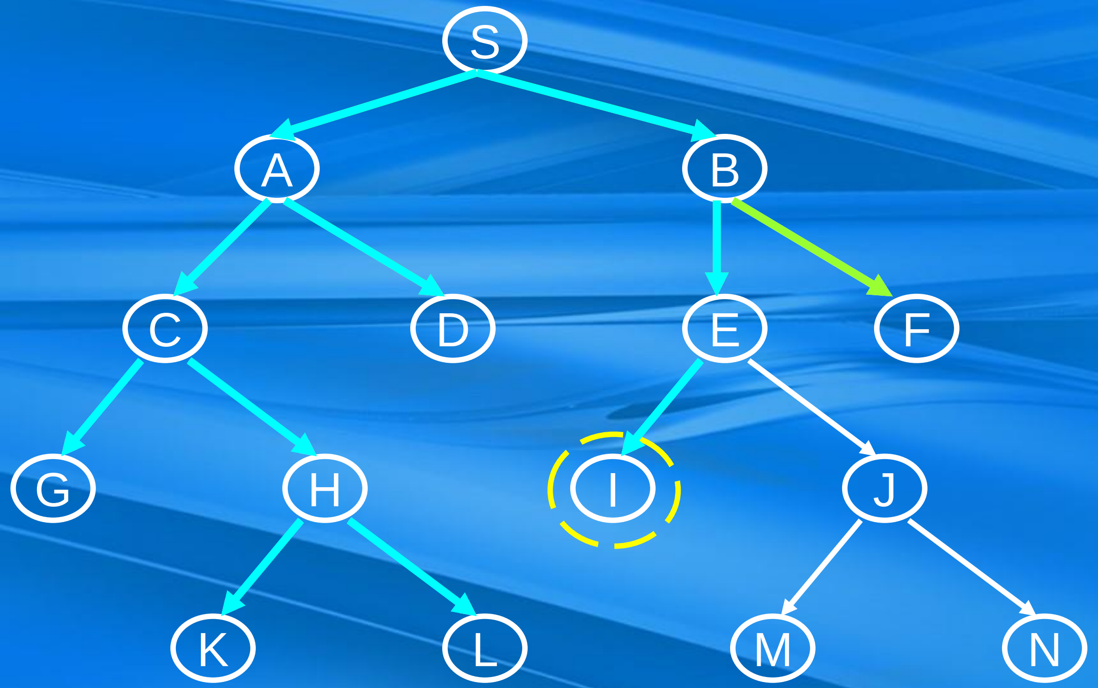
DFS has small space requirements (linear in depth) but has major problems:

- DFS can run forever in search spaces with infinite length paths
- DFS does not guarantee finding the shallowest goal

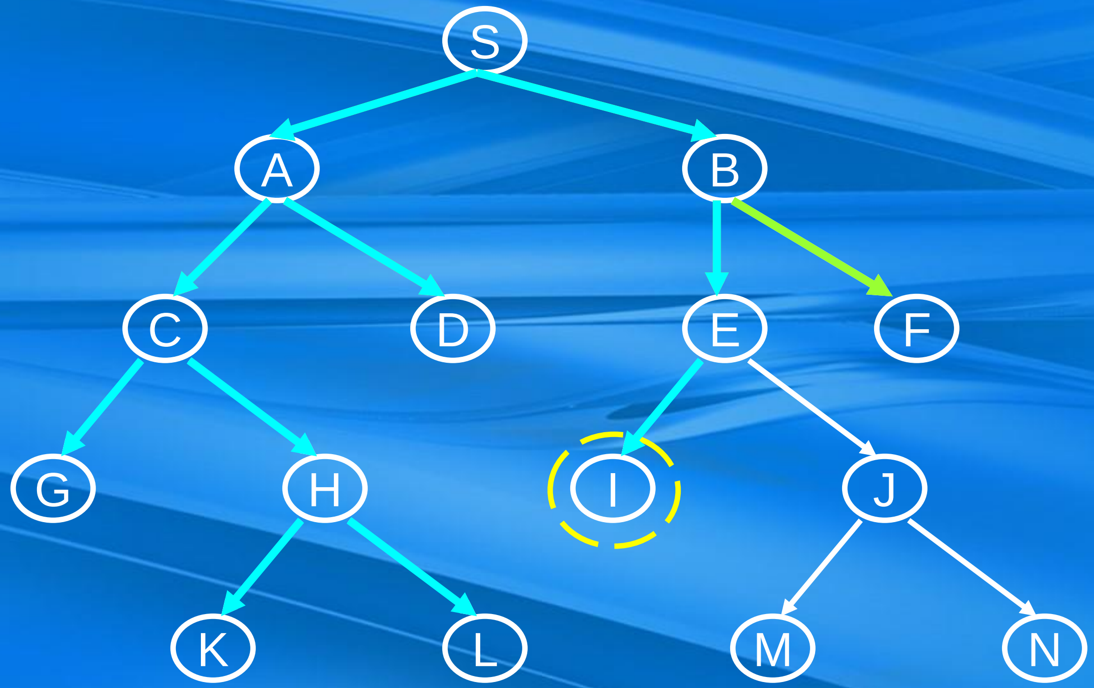
BFS guarantees finding the lowest path even in presence of infinite paths, but it has one great problem

- BFS requires a great deal of space (exponential in depth)

Progressive Deepening



Progressive Deepening

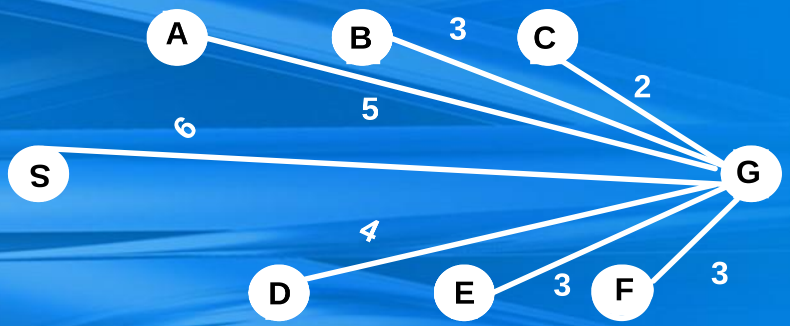


Progressive Deepening

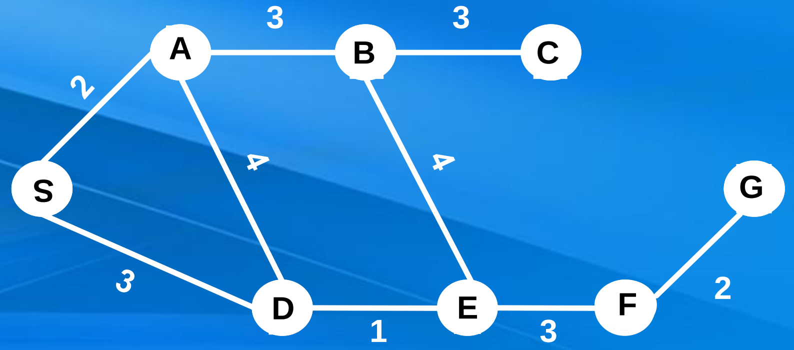
- it guarantees to find the solution at a minimum depth like BFS. Imagine that there are a number of solutions below level 4 in the tree.
- The procedure would only travel a small portion of the search space and without large memory requirements, will find out the solution.

Heuristically Informed

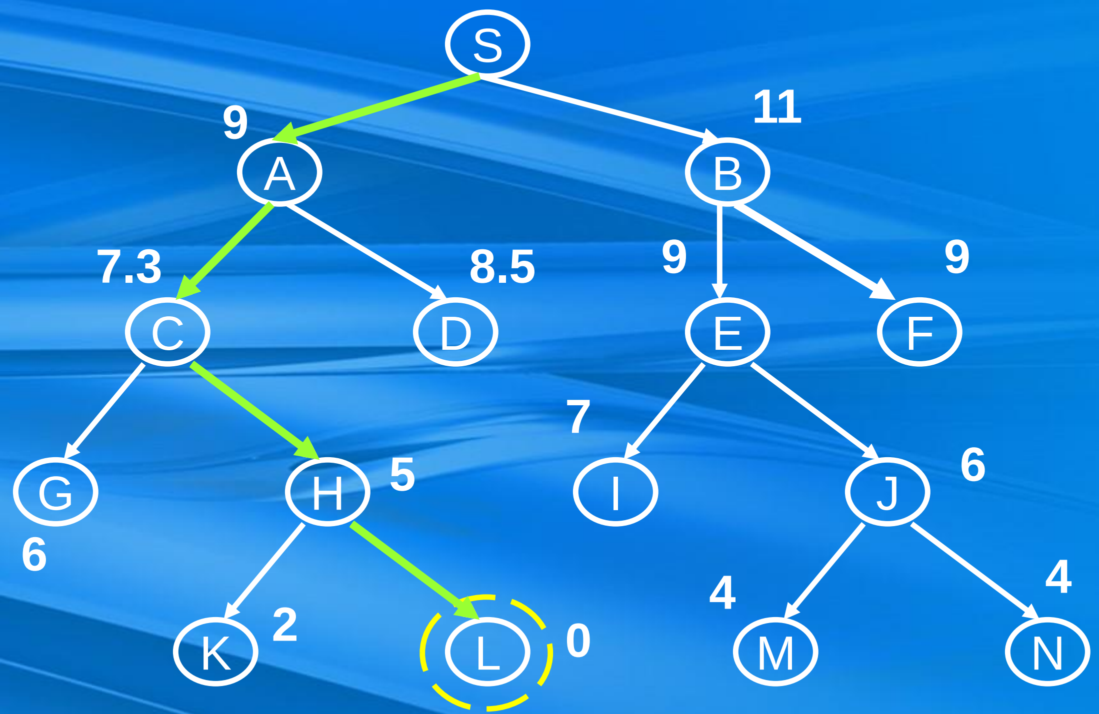
- **Heuristic Example**
  - **Here you see the distances between each city and the goal**



  - **If you wish to reach the goal, it is usually better to be in a city that is close, but not necessarily; city C is closer than, but city C is not a good place to be**



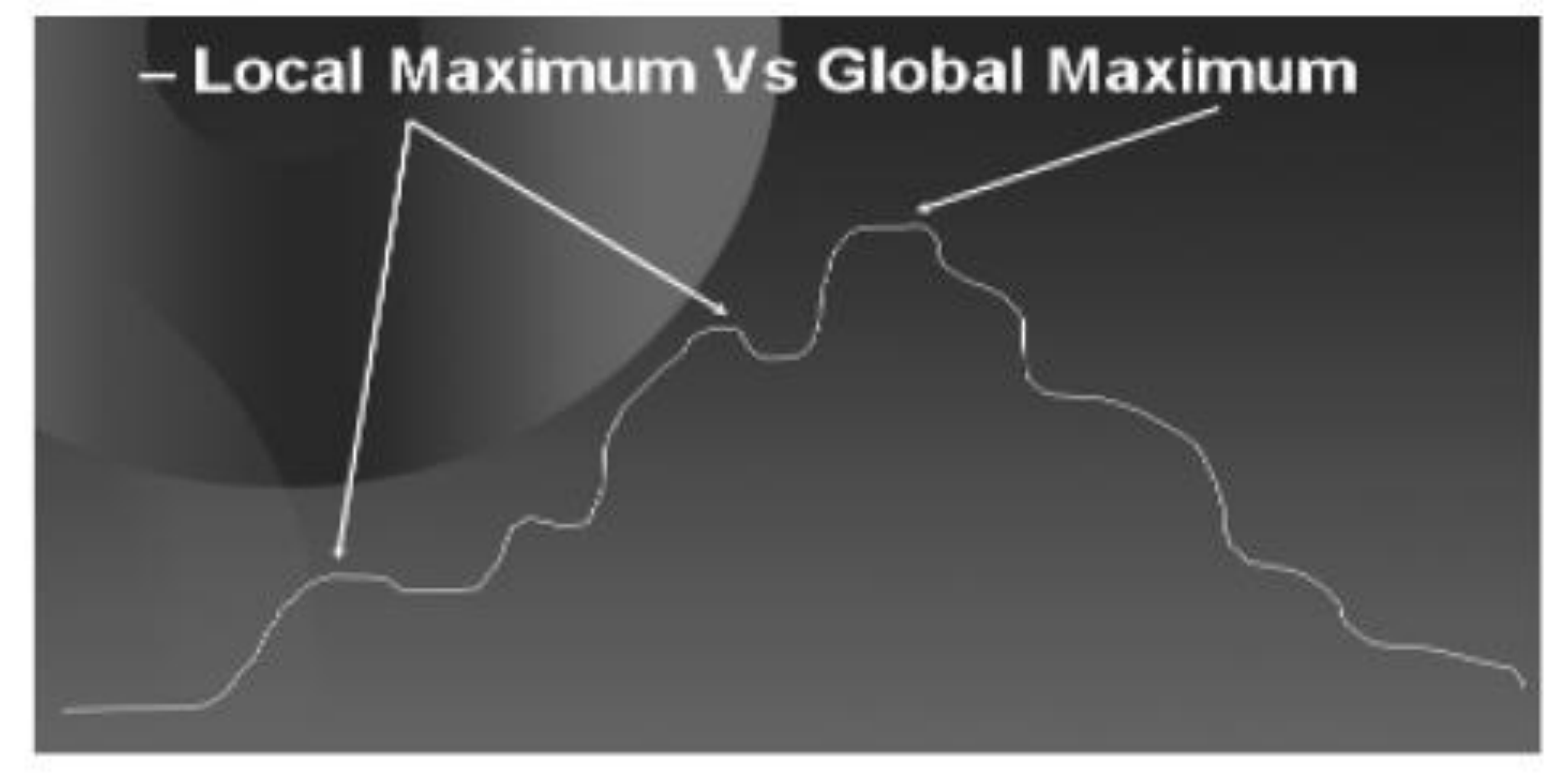
Hill Climbing



Hill Climbing is DFS with a heuristic measurement that orders choices. The numbers beside the nodes are straight-line distances from the path-terminating city to the goal city.

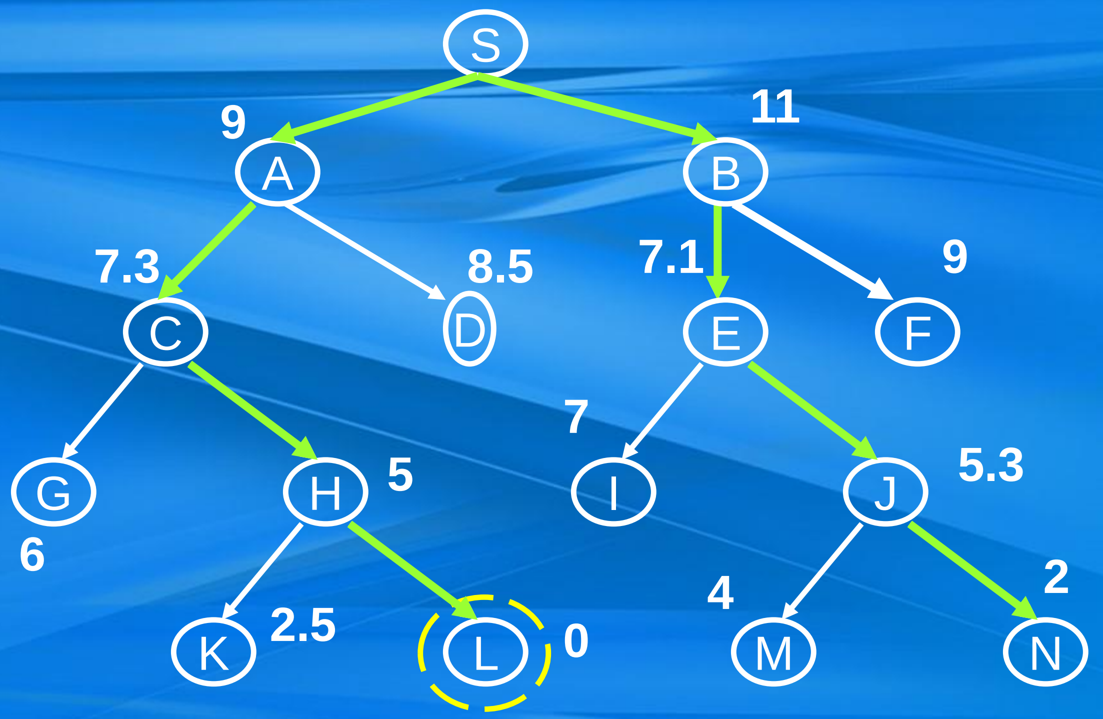
Hill Climbing

- Example:
  - Blind person climbing a hill.

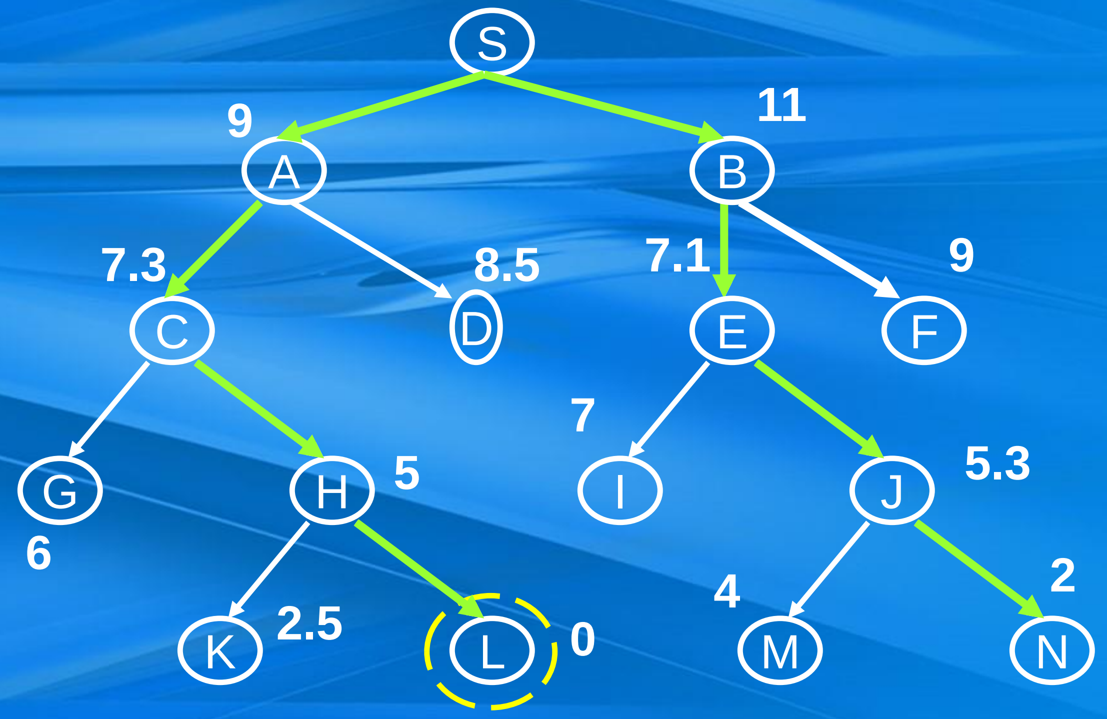


Hill Climbing

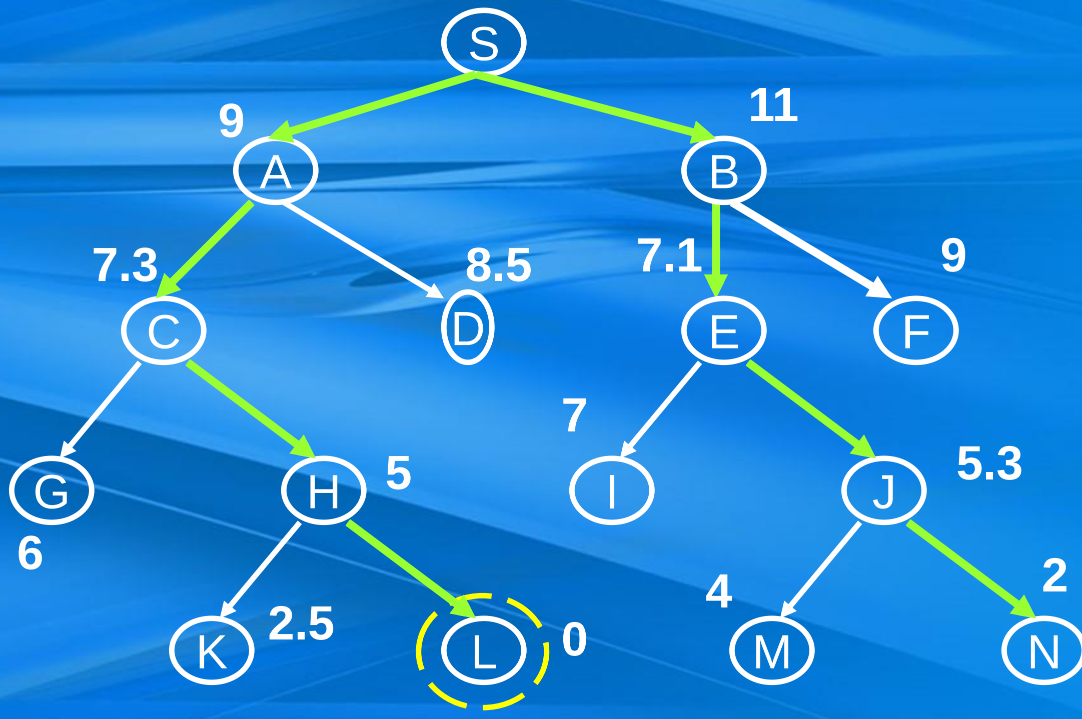
When we start at S we see that if we move to A we will be left with 9 units to travel.



Standing on A we see that C takes us closer to the goal hence we move to C.



From C we see that city H give us more improvement hence we move to H and then finally to L.



Beam Search

Degree = 2

At every level use only 2 best nodes