Lecture two

Topics that must be covered in this lecture:

- Problem spaces; problem-solving by search
- What is Search?
- Some common terms in the search issues
- Search Algorithms
- Brute-force search (breadth-first, depth-first, depth-first with iterative deepening).

---

problem space: A problem space is represented by directed graph, where nodes represent search state and paths represent the operators applied to change the state.

Problem solving : is a process of generating solutions from observed data.

− a problem is characterized by a set of goals,

− a set of objects, and

− a set of operations.

What is Search?

Search is an important aspect of AI. Search can be defined as a problem-solving technique that enumerates a problem space from an initial position in search to a goal position (or solution). How the problem space is searched is defined by the search algorithm or strategy. Search strategies offer different ways to enumerate the search space. Ideally, the search algorithm selected is one whose characteristics match that of the problem at hand.

Some common terms in the search issues:

State: is a representation that an agent can find itself in.

State Space: is a graph whose nodes are the set of all states, and whose links are actions that take the agent from one state into another.

Search Tree: is a tree in which the root node is the start state and has a reachable set of children.

Search Node: is a node in the search tree.

Goal: is a state that the agent is trying to reach.

Action: this is something that the agent can choose to do.

Branching Factor: The branching factor in a search tree is the number of actions available to the agent.

Search Algorithms

Search algorithms are divided into:

A. Blind search:

1. Take solution some or all
2. Stop with comparison.
3. Get solutions from all available.

B. Heuristic Search:

1. Take only the best solution.
2. Stop only when get a solution near optimal.

The most search algorithms are illustrated below. It begins with two types of searches - Uninformed and Informed. Uninformed Search: Also called blind, exhaustive, or brute-force search, uses no information about the problem to guide the search and therefore may not be very efficient. Informed Search: Also called heuristic or intelligent search, uses information about the problem to guide the search usually guesses the

distance to a goal state and therefore efficient, but the search may not be always possible.

Uninformed Search (Blind Search, Brout force)

This type of search takes all nodes of the tree in a specific order until it reaches to goal.

A- Depth – First – Search:

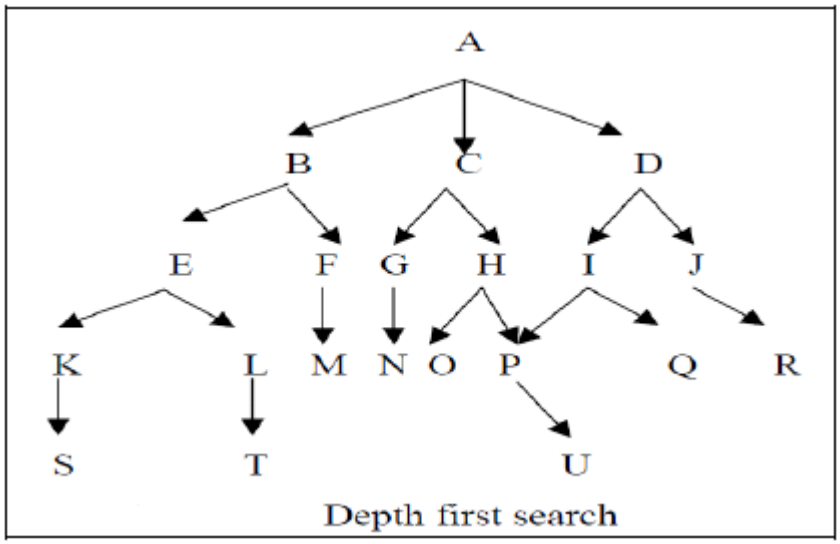
In depth–first – search, when a state is examined, all of its children and their descendants are examined before any of its siblings. depth–first search goes deeper into the search space whenever this is possible only when no further descendants of a state can be found owe it.

Depth – First – Search Algorithm

```
Begin
Open: = [start];
Closed: = [ ];
While open [ ] do
Remove leftmost state from open, call it x;
If x is a goal then return (success)
Else begin
Generate children of x;
Put x on closed;
Eliminate children of x on open or closed; put remaining children on left
end of open end
End;
Return (failure)
        End.
```

For Example

Goal: U



Depth first search

1 – Open = [A]; closed = [ ].

2 – Open = [B, C, D]; closed = [A].

3 – Open = [E, F, C, D]; closed = [B, A].

4 – Open = [K, L, F, , D]; closed = [E, B, A].

5 – Open = [S, L, F, C, D]; closed = [K, E, B, A].

6 – Open = [L, F, C, D]; closed = [S, K, E, B, A].

7 – Open = [T, F, C, D]; closed = [L, S, K, E, B, A].

8 – Open = [F, C, D,]; closed = [T, L, S, K, E, B, A].

9 – Open = [M, C, D] as L is already on; closed = [F, T, L, S, K, E, B, A].

B- Breadth – First – Search

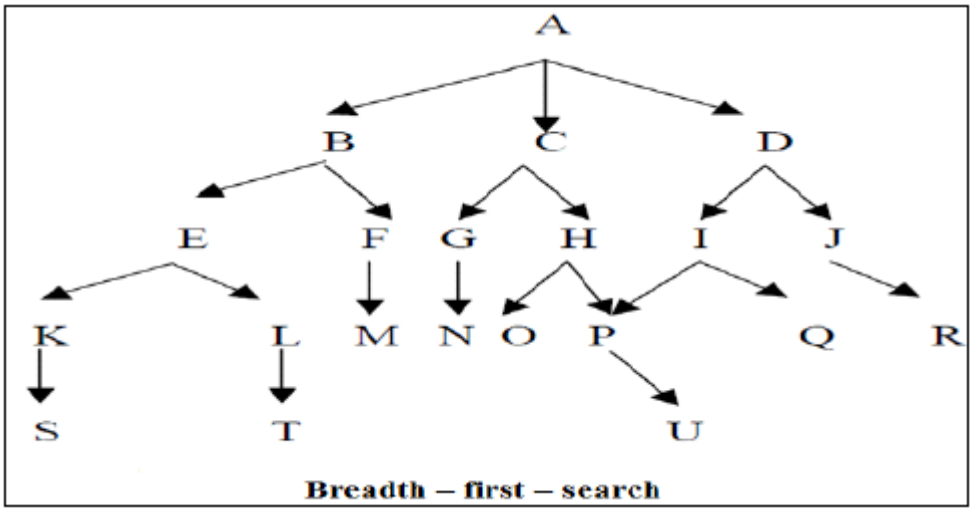
In breadth –first search, when a state is examined, all of its siblings are examined before any of its children. The space is searched level-by-level, proceeding all the way across one level before doing down to the next level.

Breadth – First – Search Algorithm

Begin
Open: = [start];
Closed: = [ ];
While open ≠[ ] do
Begin
Remove left most state from open, call it x;
If x is a goal the return (success)
Else
Begin
Generate children of x;
Put x on closed;
Eliminate children of x on open or closed;
Put remaining children on right end of open
End
End
Return (failure)
End.

For Example

Goal: U



Breadth – first – search

1 – Open = [A]; closed = [ ].
2 – Open = [B, C, D]; closed = [A].
3 – Open = [C, D, E, F]; closed = [B, A].
4 – Open = [D, E, F, G, H]; closed = [C, B, A].
5 – Open = [E, F, G, H, I, J]; closed = [D, C, B, A].
6 – Open = [F, G, H, I, J, K, L]; closed = [E, D, C, B, A].
7 – Open = [G, H, I, J, K, L, M]; closed = [F, E, D, C, B, A].
8 – Open = [H, I, J, K, L, M, N,]; closed = [G, F, E, D, C, B, A].
9 – and so on until either U is found or open = [ ].

C- depth-first with iterative deepening:

DFID is another kind of exhaustive search procedure which is a blend of depth first and breadth-first search.

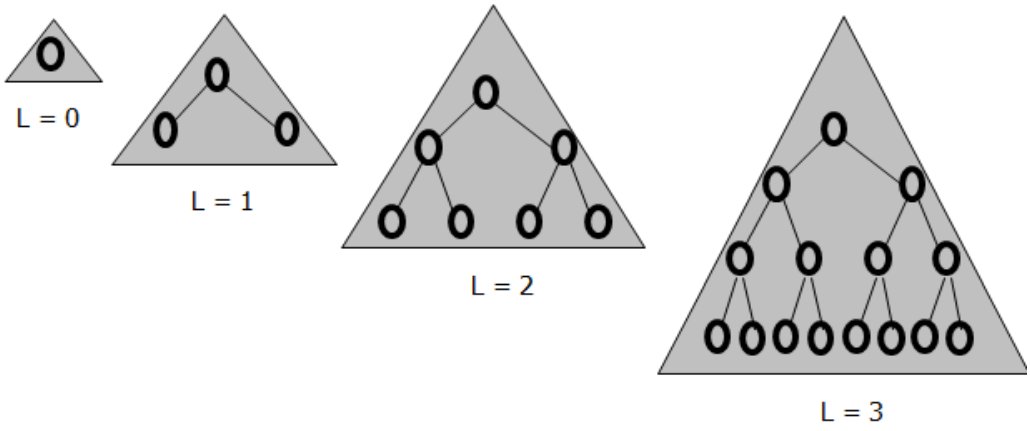
Algorithm: Steps

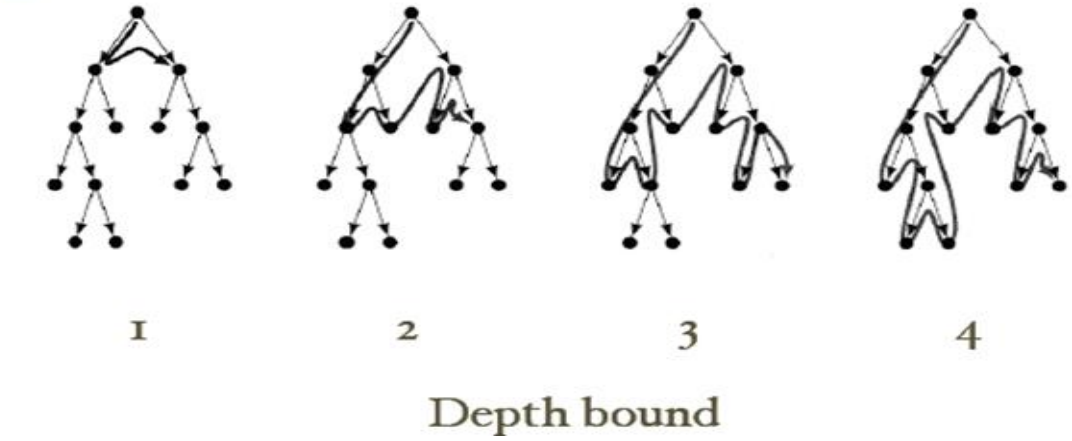
− First, perform a Depth-first search (DFS) to depth one.

− Then, discarding the nodes generated in the first search, start all over, and do a DFS to level two.

− Then three . . . . until the goal state is reached.

- In general, iterative deepening is the preferred uninformed search method when there is a large search space and the depth of the solution is not known.


L = 0
L = 1
L = 2
L = 3


1
2
3
4
Depth bound