MODULE-2
Heuristic Searching Techniques

Search Alg in AI.

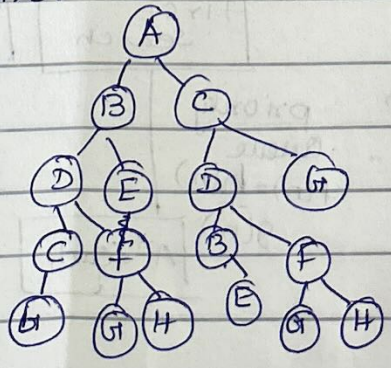
- Search forms a core component of many AI process.
- Search is a systematic examination of states to find path from start/root state to the goal state.
- for complex pblms, tradition alg are unable to find a sol. within some practical time & space limit
- Consequently, many special techniques are developed; using heuristic f^n.

Requirement of Search Alg/Technique

1. The first requirement is that it should cause motion.
2. The second requirement is that it should be systematic

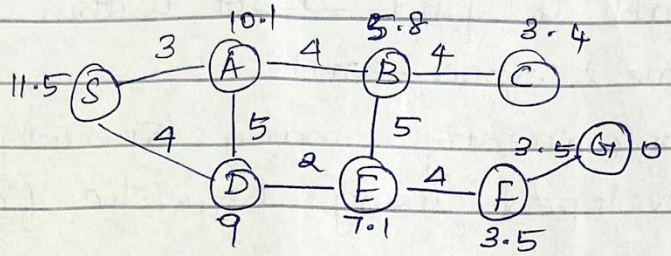
AI Search
- blind search
- unblind search.

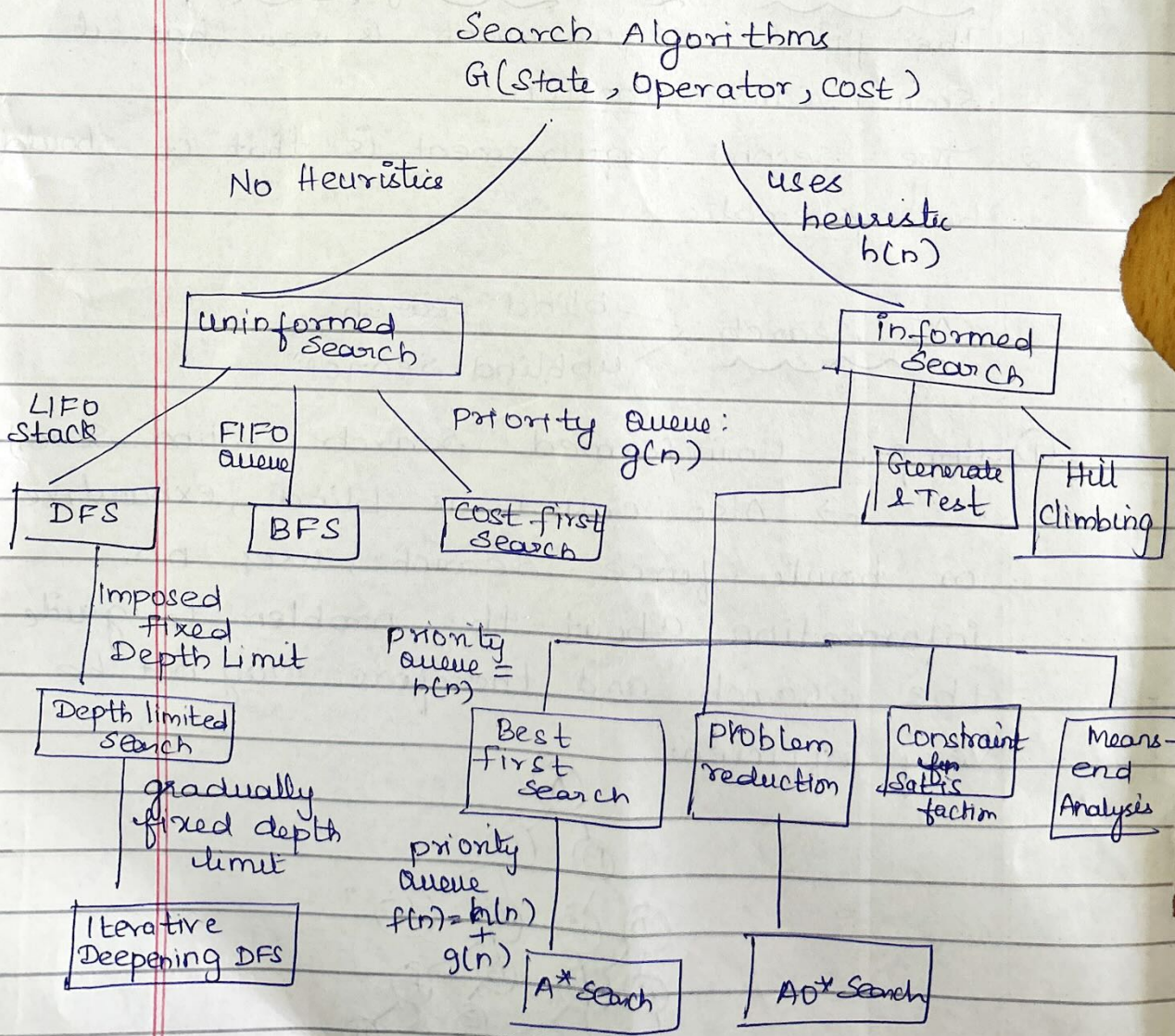
① ~~Uni for~~ Uninformed Search (Blind Search)
→ Also called as blind, exhaustive or brute-force search, uses no information about the problem to guide the search and therefore may not be very efficient.



2. Informed Search: (Heuristic 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





BFS.

BFS in AI :-

Alg :-

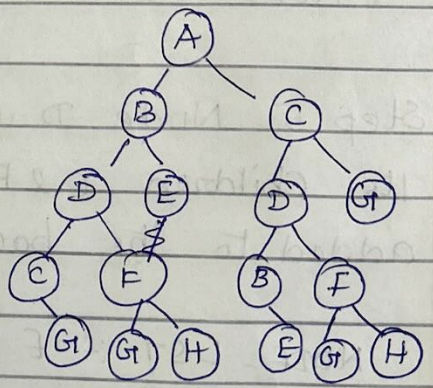
1. Create a variable called NODE-LIST and set it to the initial state
2. Until a goal state is found, or NODE-LIST is empty:
   a) Remove the first element from NODE-LIST and call it E. If NODE-LIST was empty, then quit.
   b) For element E do the following:
      i. Apply the rule to generate a new state,
      ii. If the new state is a goal state. quit and return this state
      iii. Otherwise, add the new state to the end of NODE-LIST.

Ex :-

Step 1 : Initially NODE-LIST contains only one corresponding to the source state A.

NODE-LIST : A

G1 (Goal Node)



Step 2: A is removed from NODE-LIST. The node is expanded, and its children B & C are generated. They are placed at the back of NODE-LIST

NODE-LIST: B C

Step 3: NODE B is removed from NODE-LIST. and is expanded. Its children D, E are generated and put at the back of NODE-LIST

NODE-LIST: C D E

Step 4: Node C is removed from NODE-LIST is expanded. Its children D and G are added to the back of NODE-LIST.

NODE-LIST: D E ~~G C E~~ D G

Step 5: Node D is removed from NODE-list. Its children C & F are generated and added to the back of NODE-LIST

NODE-LIST :- E D G C F

Step 6: Node E is removed from NODE-LIST. It has no children.

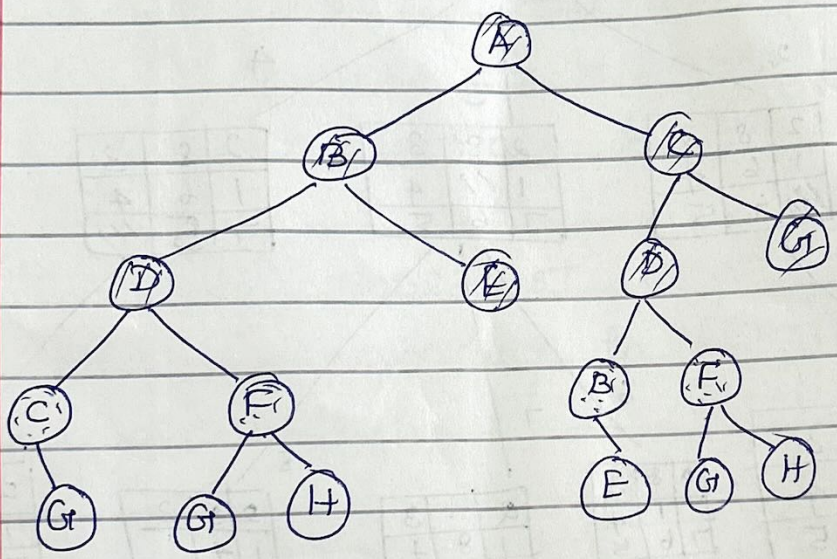
NODE-LIST :- D G C F

Step 7: D is expanded; B & F are put in OPEN

NODE-LIST : G C F B F

Step 8: G is selected for expansion. It is found to be a goal node.

Hence the alg returns the path A-C-G by the following parent pointers of the node corresponding to G.



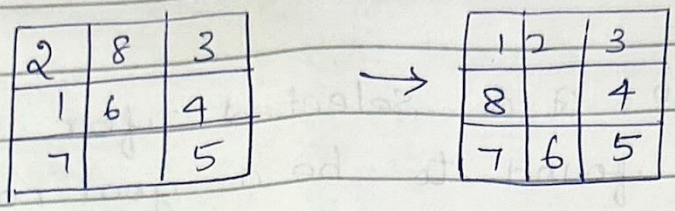
Adv :-

1. One of the simplest search strategies
2. BFS is complete. If there is solution, BFS is guaranteed to find it.
3. If there are multiple solutions, then a minimal sol. will be found.

Disadv :-

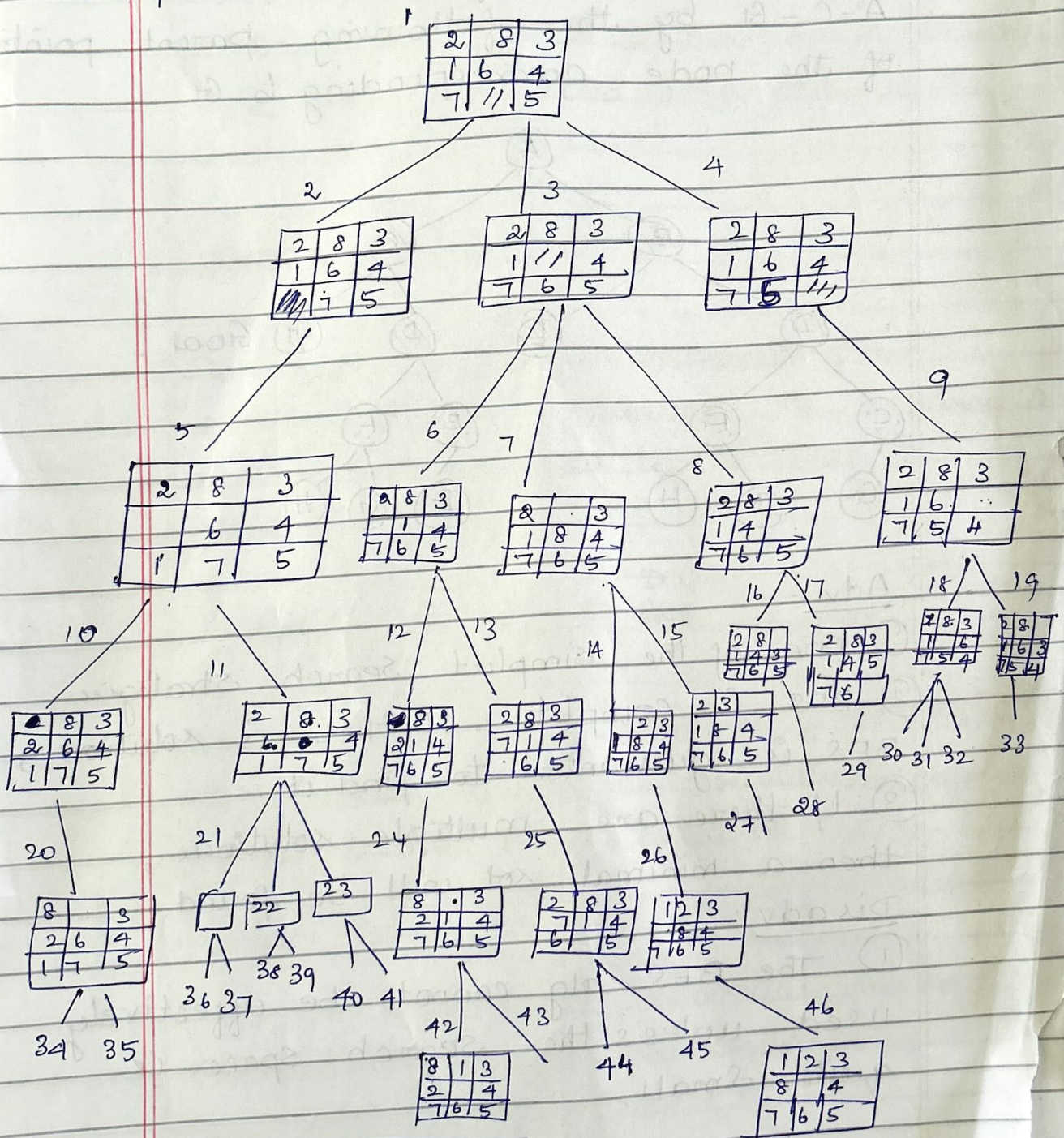
1. The BFS alg cannot be effectively used unless the search space is quite small.

Ex: 8-puzzle problem solved using BFS



Sol:-

possible moves



absent: 74, 83, 101

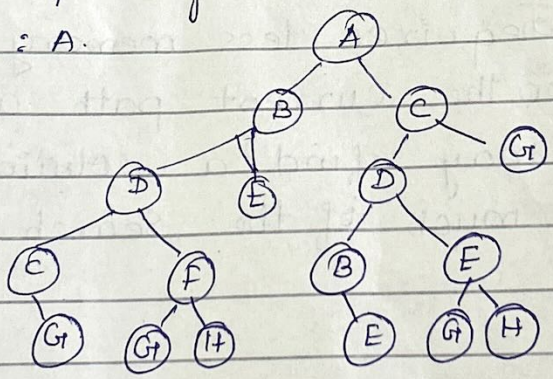
DFS

Depth first search in AI

1. If the initial state is a goal state, quit and return success.
2. Otherwise, do the following until success or failure is signaled:
   a) Generate a successor, E, of the initial state. If there are no more successors, signal failure.
   b) Call Depth-first search with E as the initial state.
   c) If success is returned, signal success. Otherwise continue in this loop.

Step1: Initially NODE-LIST contains only one node corresponding to the source state A.

NODE-LIST : A.



Step2: A is removed from NODE-LIST. A is expanded and its children B & C are put in front of NODE-LIST.

NODE-LIST: B C

Step3: B is removed. & its children D & E are pushed ~~into~~ in front of NODE-LIST.

NODE-LIST : D E C

step 4: remove D. add its children C & F. are pushed in front of NODE-LIST.

NODE-LIST: C F E C

step 5: Remove C from a NODE-LIST & its children is G is pushed in front of NODE-LIST

NODE-LIST :- G F E C

G is a goal node.

The solution path:

A - B - D - C - G is returned.

Adv :-

1. DFS requires less memory since only the nodes on the current path are stored.
2. DFS may find a solution without examining much of the search space at all.

Disadv :-

1. May find a sub-optimal solution (one that is deeper or more costly than the best solution)
2. Incomplete: Without a depth bound, one may not find a solution even if one exists