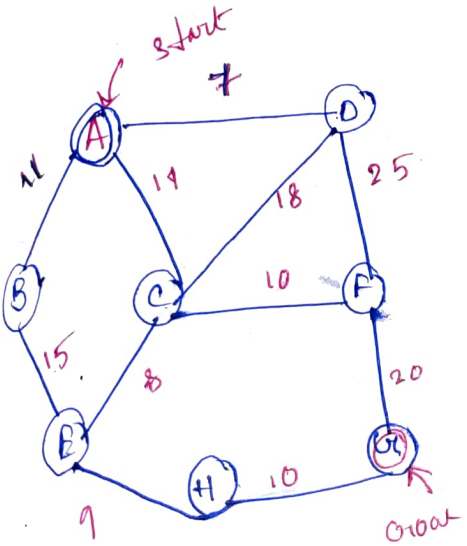
Beam Search

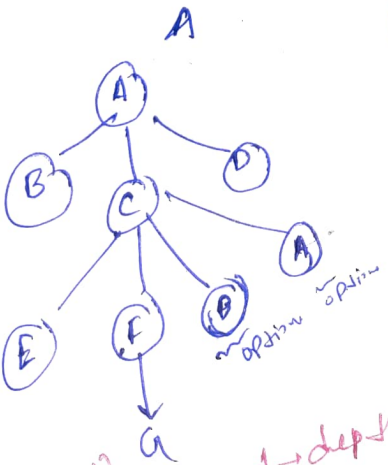
- optimize version of Best first search.
- Heuristic search Algorithm.
- Explores a graph by expanding the most promising node in a limited set.
- Reduces memory requirement — [Only Predetermined no of Best Partial Solutions are kept as candidates.]
- Greedy algorithm

Beam(β) value = Predetermine no of best partial sol^n are kept as candidate



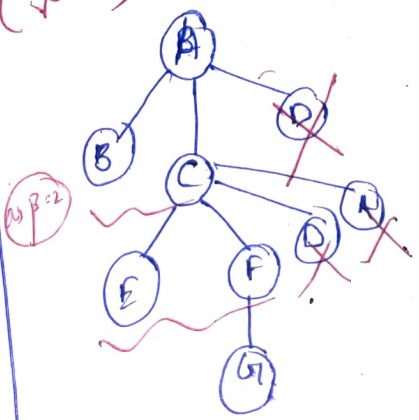
$A \to G = 40$
$B \to G = 32$
$C \to G = 25$
$D \to G = 35$
$E \to G = 19$
$F \to G = 17$
$H \to G = 10$
$G \to G = 0$

Best first search



T.C → $O(b^d)$ — d → depth
S.C → ~~$O(b^d)$~~ $O(b^d)$ • Complete

Beam search ($\beta = 2$)



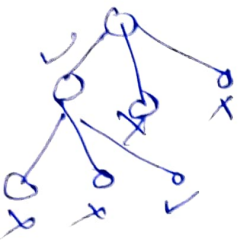
S.C → ~~O(b^d)~~ constant

when $\beta = 1$ → nothing but hill climbing algo

Good sol^n Not Optimal

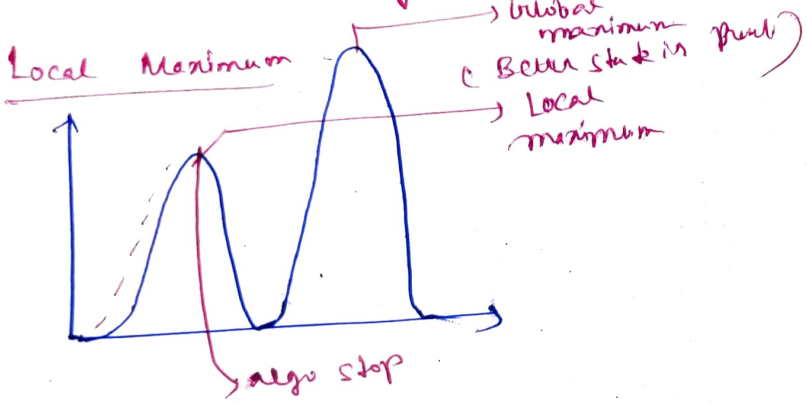
Hill climbing algo (local search, greedy method, no backtracking)

1. Evaluate the initial state.
2. Loop until a solution is found or there are no operators left.
   - Select and apply new operator
   - Evaluate the new state
   - if goal then quit
   - if better than current state then it is new current state.



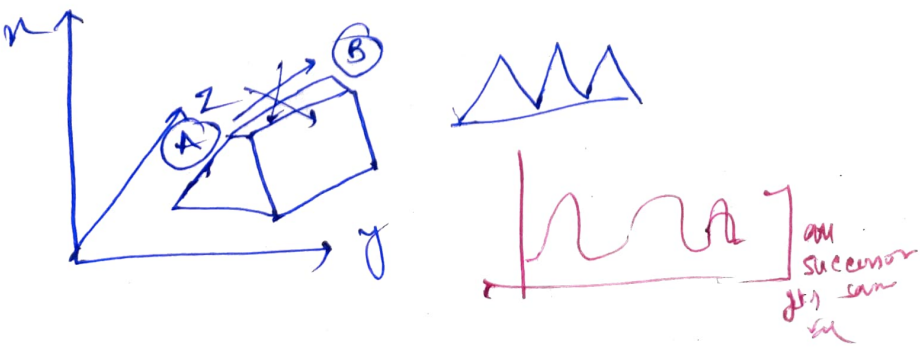
Problem in Hill climbing :-

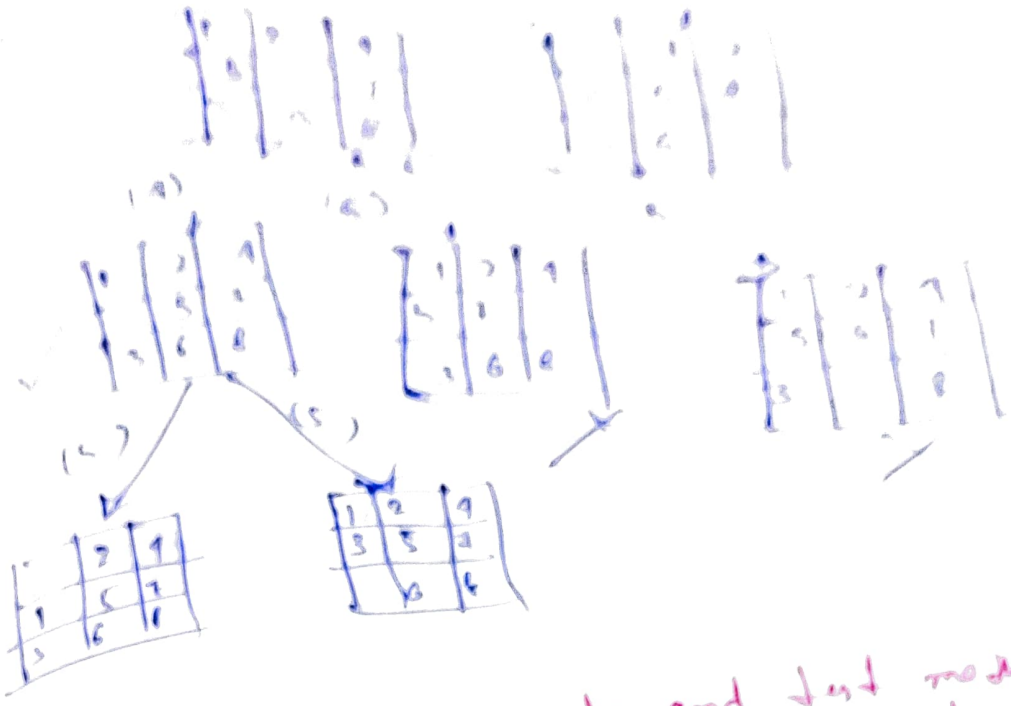
1) Local Maximum



2) ~~Plateau~~ Plateau / Flat Maximum :-

3) Ridge :-





- variant of generate and test method in which feedback from test procedure is used to help generator decide which dir. to move in search space.
- always moves in a single direction
- It is like DFS

if new state is better than current state
⇓
new state

Simulated Annealing
- checks all neighbors
- simulated annealing (SA) allows downward steps

↳ Annealing is process in metallurgy where metals are slowly cooled to make them reach a state of low energy where they are very strong.

Adv

→ Easy to code for Complex Problems also Gives good soln

→ Statistically guarantees — (finding optimal Soln).

Disadv

→ Slow Process.

→ can't tell whether the optimal Soln is found

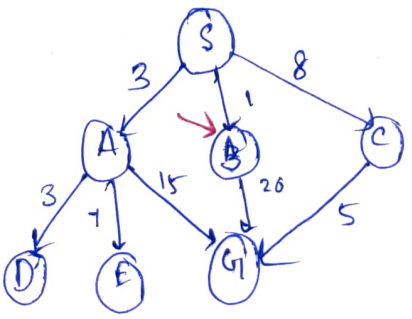
↳ some other method is also req

Simulated annealing

If fill climbing

Uniform-cost search

- use Priority queue instead of simple queue.
- Insert node in the increasing order of the Cost of the path so far.
- Guaranteed to find optimal Solution
- This is called uniform-cost search.



$g(n)$ = Cost of the path from start node to current node n.

$g(A) = 3$

$g(B) = 1$

$g(C) = 8$

by increasing value of g

- Expand nodes, discard nodes list

→ [So]

→ So [B1, A3, C8]

→ B1 [A3, C8, G121]

→ ~~A3 [C8, G1~~

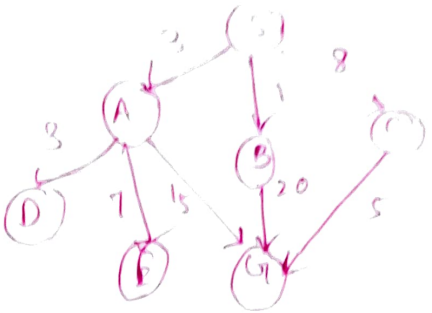
→ A3 [D6, C8, E10, G118, G121]

→ D6 [C8, E10, G118, G121]

→ C8 [E10, G113, G118, G121]

→ E10 [G113, G118, G121]

→ (G113) [G118, G121]



- Solution path found S→C→G1 cost = 13.
- Number of nodes expanded (including goal node) = 7 ??

Depth limited Search

↳ In DFS there is a Problem called infinite loop / path where, if we start from a particular direction (sequence) it can be a infinite path and solution must be in other direction to overcome this problem we can use depth limited search.

↳ working is similar to DFS but with predetermined limit.

→ Help in solving the problem DFS: infinite path

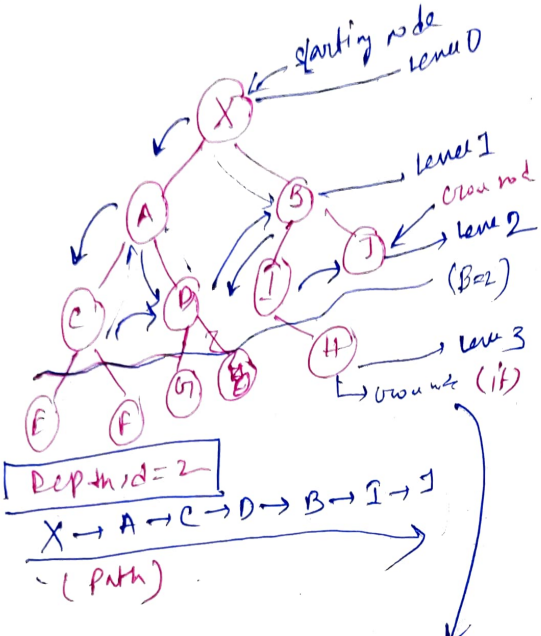
Termination cond →
- (i) failure value: There is no soln.
- (ii) Imp cut-off Failure: Terminates on reaching ~~Predetermine~~ predetermined depth.

- Advantages : Memory efficient

Disadvantage : (1) can be terminated without finding soln.

(2) Not optimal.

[Time complexity - $O(b^d)$ → $O(b \times d)$
Space]



if goal node H
Then depth will result no Soln

Uniformed Searching vs Informed Searching

Uniformed Search

(1) Search without Information. (heuristic method)

(2) No knowledge

(3) Time Consuming

(4) More complexity

(5) BFS, DFS etc

Informed Search

(1) Search with information

(2) use knowledge to find steps to Soln.

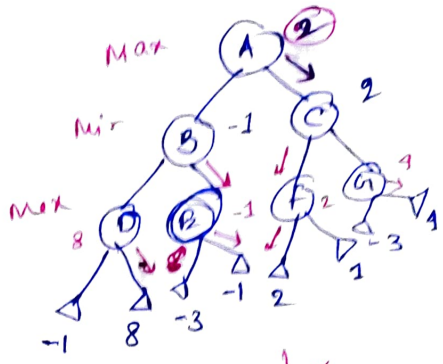
(3) Quick Soln.

(4) Less complexity (time, Space)

(5) A*, Heuristic DFS, Best first search

Minimax - Algo

- Backtracking algo
- Best move strategy used
- Max will try to maximize (Best move)
- Min will try to minimize utility (worst move).



Game tree

A → C → F → 2

T.C → $O(b^d)$

d → depth

b → branching factor (choices)

(35 → very high value

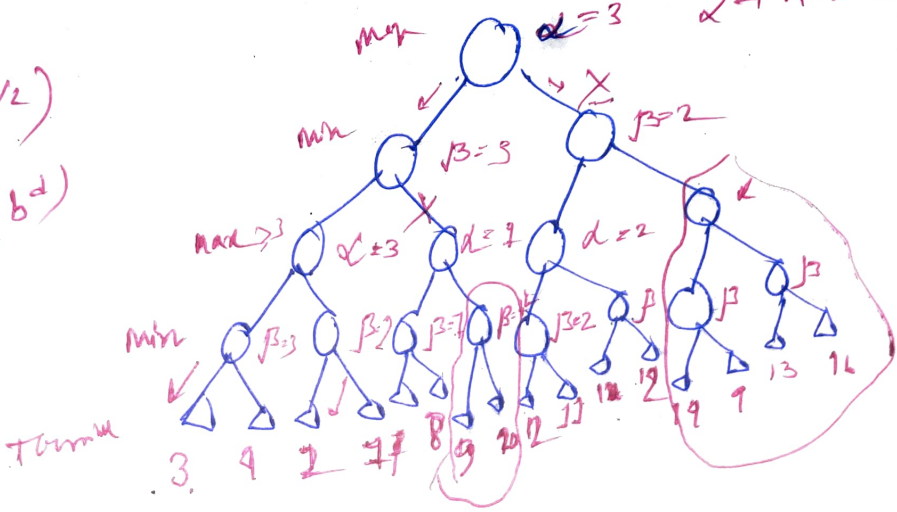
α-β Pruning

α-β Pruning -

- cut off search by exploring less no of nodes

β - min

α → max



T.C → $O(b^{d/2})$

worst case → $O(b^d)$