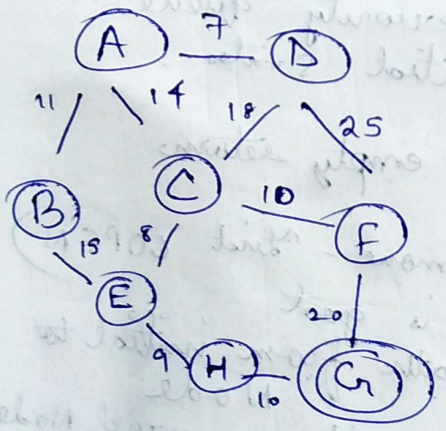
Beam Search

- Optimized version of Best First Search
- Heuristic Search Algorithm
- ~~Es~~ explores graph by expanding most promising node in limited set.
- Beam value (β): Predetermined no. of best partial solution are kept as candidates. (Will be given)
- Reduces memory requirement
- Greedy algorithm
- Constant Space Complexity

Take $\beta = 2$ (if not given)

Given Heuristic values:-
Straight line distance

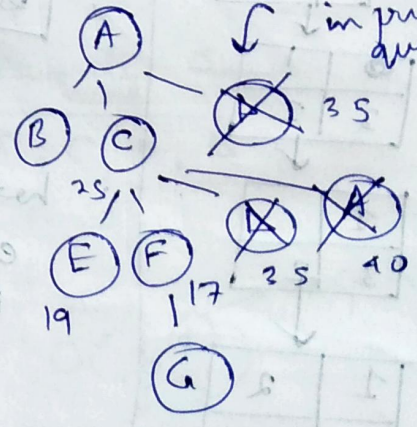
A → G = 40
B → G = 32
C → G = 25
D → G = 35
E → G = 19
F → G = 17
G → G = 0
H → G = 10



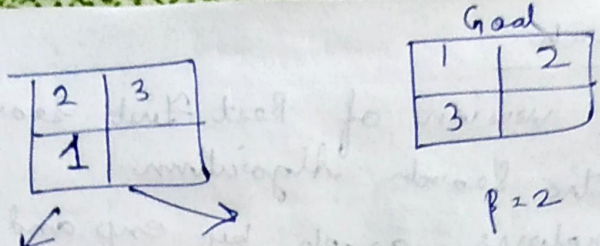
Sorting
A B C D E F

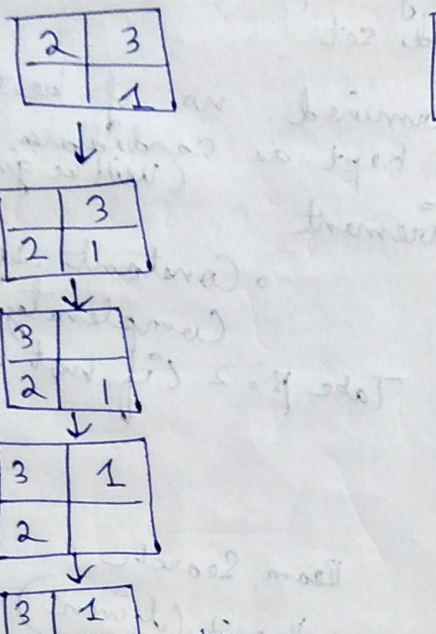
Beam Search
$\beta = 2$ (Given)

Don't keep D in priority queue



Example of local beam search





~~1 2~~

Best first search

- Informed
- Greedy
- Heuristic

Let open be priority queue containing initial states

LOOP

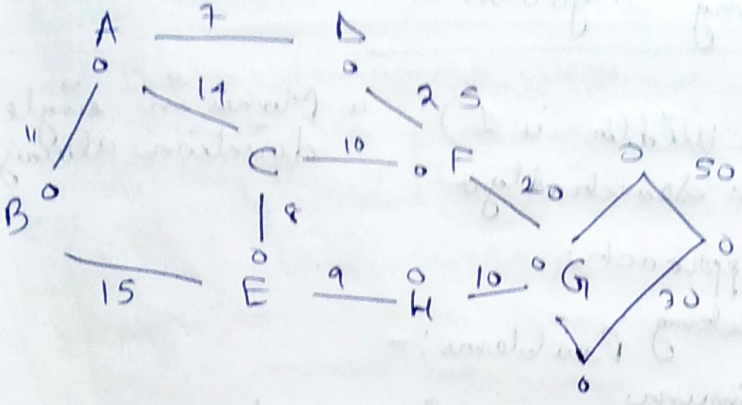
if OPEN is empty return failure

NODE ← Remove-first (OPEN)

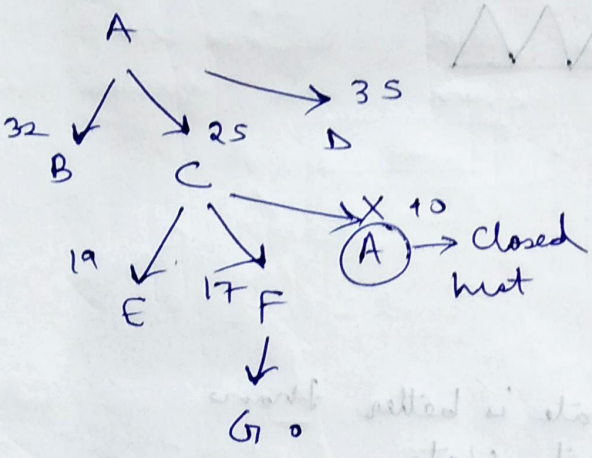
If Node is goal return path from initial to Node

else generate all successors of Node and put newly generated node into OPEN according to their heuristic values.

END LOOP



A → G = 10
B → G = 32
C → G = 25
D → G = 35
E → G = 19
F → G = 17
H → G = 10
G → G = 0



A → C → F → G

Priority Queue

A, C, B, D
F, E

$O(b^d)$ — d: depth, b: branching factor

good solution ✓

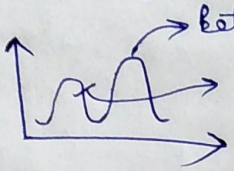
optimal solution may or may not be given

Hill Climbing Algorithm

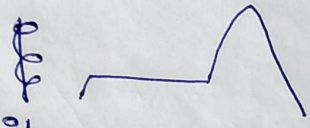
- B - 1 (Beam Width is 1)
- Local ~~Search~~ Search Algo
- Greedy Approach
- No backtracking
- Moves in single direction always

Problems :-

1) Local Maximum

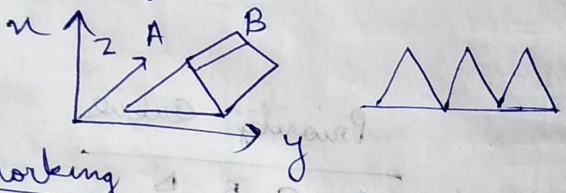


2) Plateau / Flat Maximum

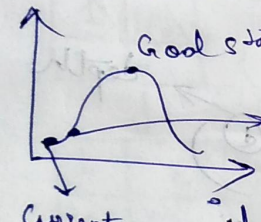


(~~Se~~ See Hill Climbing with N Queens)

3) Ridge



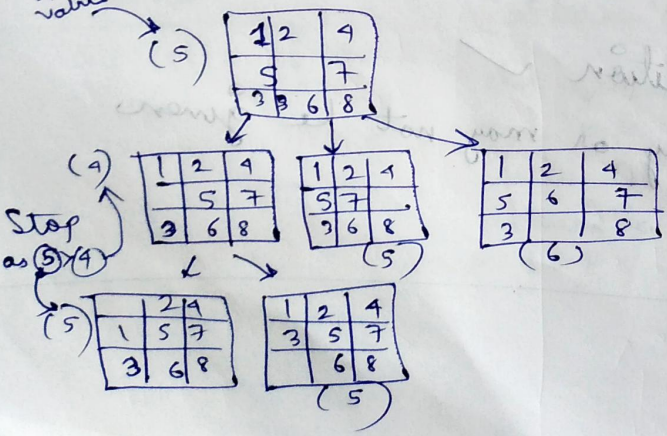
Working



if new state is better than current state

new state = Current state

Heuristic value



Simulated Annealing

- 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 strong.

Depth limited search

- Similar to DFS with predetermined limit
- DLS solves drawback of infinite path in DFS.
- Uninformed search
- Node at depth limit will treat as it has no successor nodes further.

★) Adv
- Memory efficient

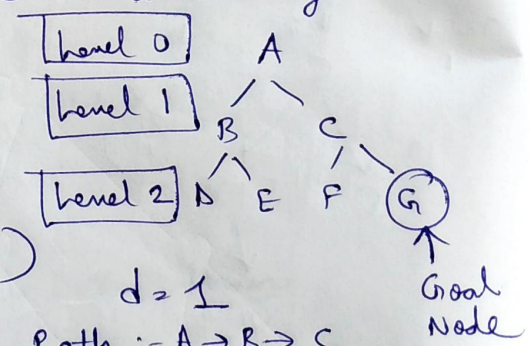
Dis Adv
- incompleteness
- may not be optimal if problem has more than 1 solution

★) DLS terminated with 2 conditions :-
- Standard failure value :- Problem has no solution
- Cutoff failure value :- Defines no solution for problem within given depth limit.

★) Not optimal

TC :- b^l

SC :- $O(b \times l)$



d = 1

Path :- A → B → C

G not found

Uninformed Searching	Informed searching
• Search without information	• Search with information
• Time Consuming	• Quick solution
• More complexity (Time, Space)	• less complexity (Time, Space)
• DFS, BFS, DLS	• A*, Best First search, Local Beam search
• Not heuristic	• Heuristic

Component of AI

learning :-
- Trial & error
- **Generalization** → Applying past experience to analog analogous new situations.
  For past tense add 'ed'.
- **Rote learning** →
  jump's past tense is jumped

Reasoning :-
Draw inferences to situation.

→ Deductive :-
Akash must be in museum or cafe.
If Akash is not in museum,
he must be in cafe.
(Truth of premise guarantees truth of conclusion)

→ Inductive :-
~~If A~~
Previous accidents of this sort was caused by instrumental failures therefore this accident was caused by instrumental failure.
Truth of Premise leads support to conclusion without absolute assurance.

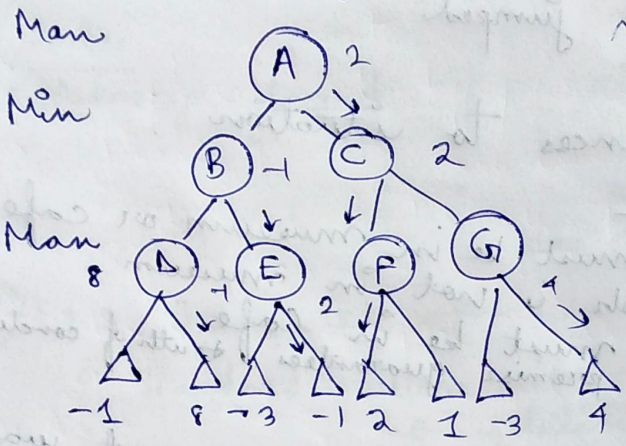
Problem Solving :-
Systematic search through a range of possible actions in order to reach predefined goal.

Perception :-
Environment is scanned by various sensory organs. Perception helps self driving cars to go at moderate speed based on its surroundings.

Language Understanding :-
System of signs having meaning by convention.
Traffic signs form a mini language.
⚠ means hazard.

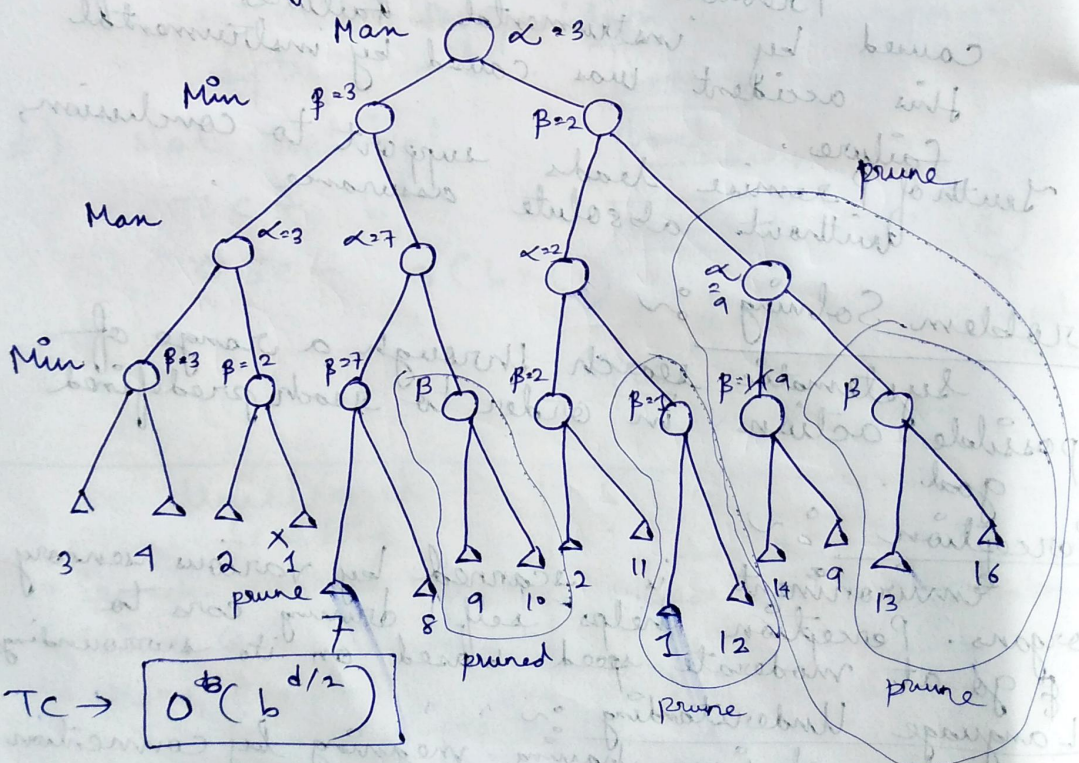
Mini Max Algorithm

- Back tracking Algorithm
- Best move strategy used
- Max will try to maximize its utility (Best Move)
- Min will try to minimize utility (Worst Move)



Time → $O(b^d)$

Game Tree



TC → $O^{\text{Best}}(b^{d/2})$

Uniform cost search

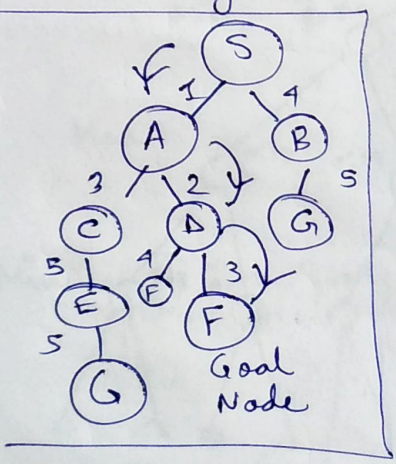
- uninformed search
- Used for traversing a weighted tree or graph
- comes in play when different cost available for each edge.
- Goal is to find path to goal node which has lowest cumulative cost
- Implemented by priority queue
- Man priority to lowest cumulative cost
- equivalent to BFS if path cost of all edges is same.

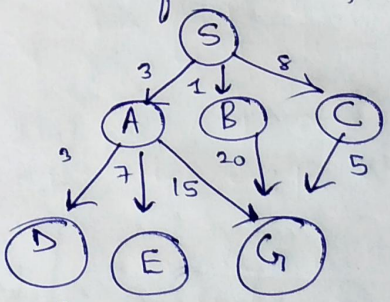
Adv

- UCS is optimal as we choose path with the least cost.

Dis Adv

- Does not care about no. of steps involved.
- Might be stuck in infinite loop





{S0}
S0 { B1, A3, C8 }
B1 { A3, C8, G20 }
A3 { GD, C8, E10, G18, G21 }
DG { C8, E10, G18, G21 }
C8 { E10, G13, G18, G21 }
E10 { G13, G18, G21 }
G 1883 { G18, G21 }

Path S → C → G
Cost → 13
Nodes expanded 7

See :~ Iterative deepening
Bidirectional search

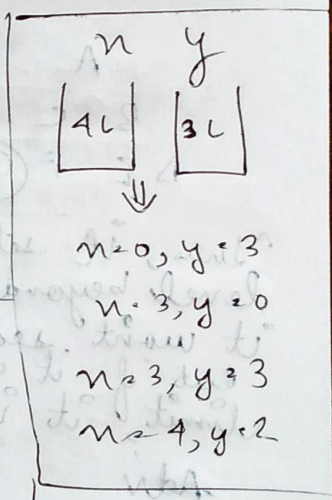
Water JUG Problem

2 jugs of different capacities are given :~
X litre, Y litre
No marking on any JUG
Goal is to fill exactly L litres of water into Y litre JUG.
State is represented as $\langle x, y \rangle$ — x: amount of water in 'x' litre jug; y: amount of water in 'y' litre jug

eg :-
two jugs: 2 litre, 3 litre; $\langle x, y \rangle$

Goal :- get exactly 1 litre water in 2 litre jug

~~$\langle x, y \rangle = (0,0)$~~
$\langle 0,0 \rangle$ {initial state}
$\langle 0,3 \rangle$
$\langle 3,0 \rangle$
$\langle 3,3 \rangle$
$\langle 4,2 \rangle$
$\langle 0,2 \rangle$
$\langle 2,0 \rangle$



x	y	Rule
0	0	-
0	3	2
3	0	8
3	3	2
4	2	6
0	2	5
2	0	8

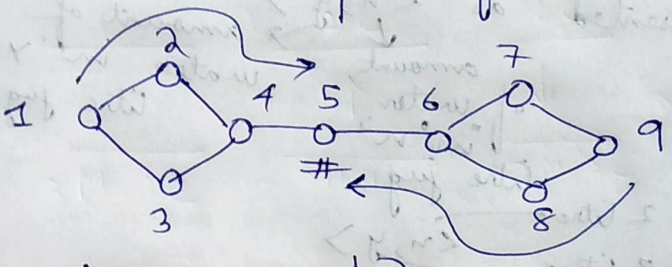
State space

1 $(x, y) \rightarrow (4, y)$ $x < 4$
2 $(x, y) \rightarrow (x, 3)$ $y < 3$
3 $(x, y) \rightarrow (x-d, y)$ $x > 0$
4 $(x, y) \rightarrow (x, y-d)$ $y > 0$
5 $(x, y) \rightarrow (0, y)$ $x > 0$
6 $(x, y) \rightarrow (4, y-(4-x))$ $y > 0, x+y >= 4$
7 $(x, y) \rightarrow (x-(3-y), 3)$ $x > 0, x+y >= 3$
8 $(x, y) \rightarrow (x+y, 0)$ $x+y <= 4, y > 0$
9 $(x, y) \rightarrow (0, x+y)$ $x+y <= 3, x > 0$

Bidirectional search

Bidirectional Search

- 2 simultaneous search from initial node to goal & backward from goal to initial, stopping when 2 meet
- Time Complexity :- $2(b^{d/2})$
- Complete in breadth first search
- Not in depth first search



SC :- $O(b^d)$

Iterative Deepening Depth First Search (gfg)

IDDFS

A

B C

D E F (G) Goal

Searches A

Searches A B backtracks to A, then searches C.

Searches A B D E C F G in dfs way.

Thus, it sets a level beyond which it won't search. But if it doesn't find goal node, within limit, it increases its level limit.

Time → b^d

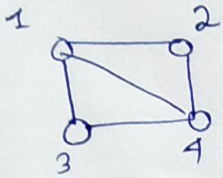
Space → $O(bd)$

~~Adv~~

DFS's space efficiency,
BFS' fast search
combined.

Constraint Satisfaction Problem

Intelligent Backtracking



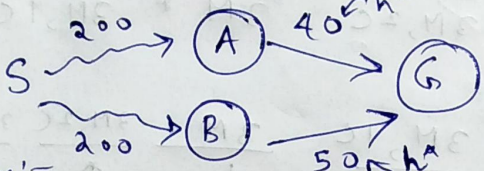
V = {1, 2, 3, 4}

D = {Red, Green, Blue}

C = {1≠2, 1≠3, 1≠4, 2≠4, 3≠4}

	1	2	3	4
Initial DOM	R, G, B	R G B	R G B	R G B
1 = R	R	G B	G B	G B
2 = G	R	G	G B	B
3 = B	R	G	B	B empty
3 = G	R	G	G	B ✓

★) How to make A^a admissible?



Given:-

$g(A) = 200, g(B) = 200$

estimated ↙ Actual

$h(n) <= h^*(n)$ underestimation

$h(n) >= h^a(n)$ overestimation

heuristic

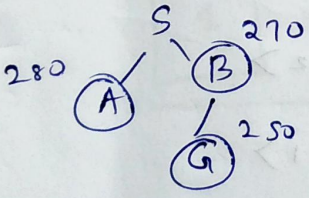
Case I overestimation

Assuming $\begin{cases} h(A) = 80 \\ h(B) = 70 \end{cases} > h^*$

$f(A) = 200 + 80 = 280$

$f(B) = 200 + 70 = 270$

$f(G) = g(G) + h(G) = 250 + 0 = 250$



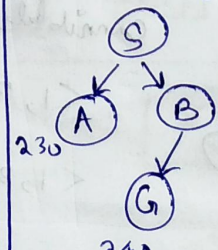
Case II underestimation

$h(A) = 30$

$h(B) = 20$

$f(A) = g(A) + h(A) = 200 + 30 = 230$

$f(B) = 200 + 20 = 220$

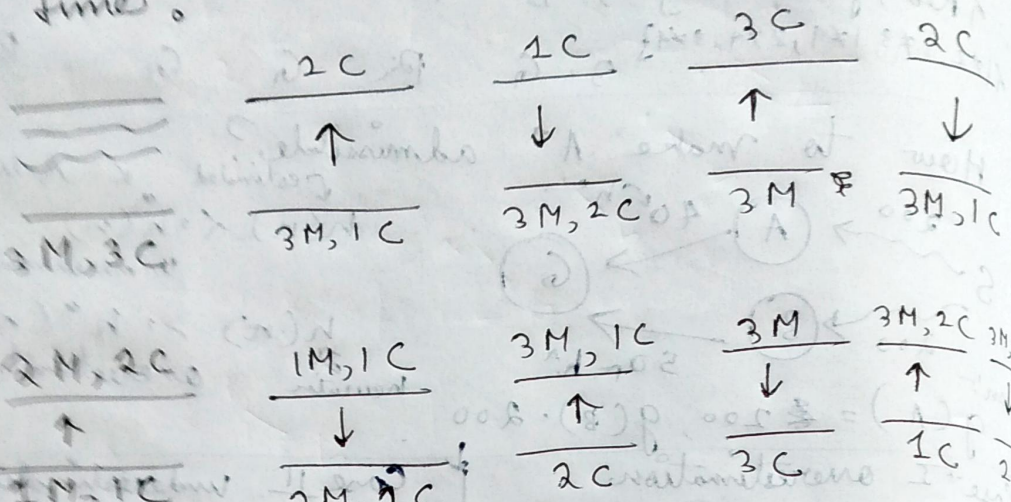


$f(G) = g(G) + h(G) = 250 + 0 = 250$

$f(G) = g(G) + h(G) = 250 + 40 = 240$

Problem

3 M, 3C were on one side of river.
→ All want to cross river
→ On same side of river
M can't be less than C.
→ 1 boat available that can hold 2 ~~peo~~ people at a time.



3M, 3C

State Representation → No. of missionaries
<R, M, C>
↓
No. of cannibals

River side
1
2 (Ending)

<1, 3, 3> <2, 0, 0>] Initial State
↓
<1, 3, 1> <2, 2, 0, 2>
↓
<1, 3, 2>, <2, 0, 1>
↓
<1, 3, 0> <2, 0, 3>
↓
<1, 3, 1>, <2, 0, 2>
↓
<1, 1, 1> <2, 2, 2>

<1,2,2>, <2,1,1>

↓

<1,0,2> <2,3,1>

↓

<1,0,3> <2,3,0>

↓

<1,0,1> <2,3,2>

↓

<1,0,2> <2,3,1>

↓

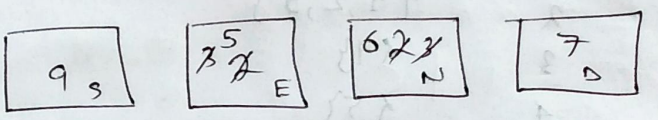
<1,0,0> <2,3,3> Goal state

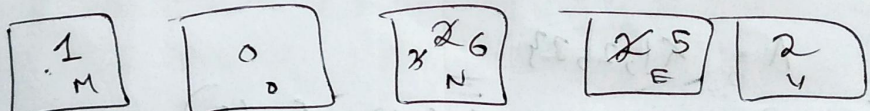
Best first vs Hill Climbing

Is uniform cost search special case of Best first search?

9567
1085
9567
1085

SEND
+ MORE
MONEY





A heuristic $h(n)$ is admissable for every node n, $h(n) \leq h^*(n)$, where $h^*(n)$ is the true cost to reach the goal state from n.

An admissible heuristic never overestimates the cost to reach the goal, i.e, it is optimistic.

A search algorithm is admissible if it returns an optimal solution path.

Admissibility of A* algorithm does not require monotonicity of the heuristic. Admissibility only requires an optimal solution to be returned for a goal node (for which $h = 0$) rather than for every node that is closed.

Plot integers from 0-9 for the alphabet specified in the crypt Arithmetic problem with following constraints.

1. No two alphabet have the same integer value
2. Having alloted the different values for different alphabet we have to perform arithmetic operations.

```
  FORTY
+ TEN
+ TEN
------
 SIXTY
```

Y → 3
N → 8
E → 5
T → 4
R → 2
X → 1

O → 8
I → 9
F → 6
S → 7

BEST FIRST SEARCH VS HILL CLIMBING

- Similar to Steepest ascent hill climbing with two exceptions:
  - In hill climbing, one move is selected and all the others are rejected, never to be reconsidered. In BFS, one move is selected, but the others are kept around so that they can be revisited later if the selected path becomes less promising
  - The best available state is selected in the BFS, even if that state has a value that is lower than the value of the state that was just explored. Whereas in hill climbing the progress stop if there are no better successor nodes.

Best-first search vs hill-climbing search

Best-first search

1. Space complexity: O(b^d), because the whole search tree is stored in the memory.
2. Time complexity: O(b^d). A good heuristic function can substantially improve this worst case.
3. Greedy search: not complete, not optimal.

   A* search: complete and optimal if the estimated cost for the cheapest solution through n, f(n), is a monotonic function.

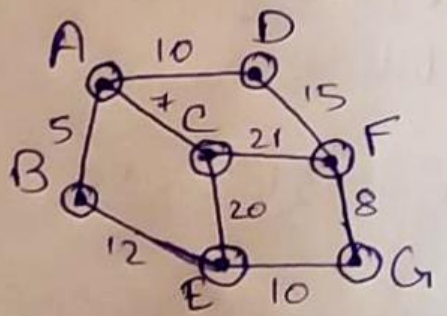
Hill-climbing search

1. Space complexity: O(1), because only a single state is maintained in the memory.
2. Time complexity: O(b^d).
3. Not complete, because of the local maxima phenomena (the goal state is not reached, but no state is better that the current state). Possible improvement: simulated annealing, which allows the algorithm to backtrack from the local maxima in an attempt to find a better continuation.
4. Not optimal.

Local Beam Search

Local Beam Search in Artificial Intelligence, is an informed search technique which works based on some heuristic values to find the goal state. It is the modified version of Best first search algorithm where a B value is given. Based on that value, we have to choose some best paths and then cut all the other paths. In that way, choosing a good path becomes much easier and the space complexity becomes lower than best first search algorithm.

Let's assume an example -



B = 2

Here we have to reach G_1 state from A state. So A is the start state and G_1 is the goal state

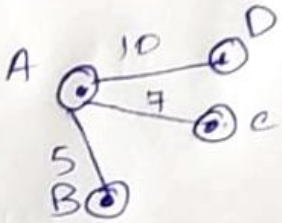
Now, Based on that given heuristic values we have to choose best path and go ahead.

Here, Beam value (B) = 2. So, in every

Heuristic values

- A - G_1 = 40
- B - G_1 = 32
- C - G_1 = 28
- D - G_1 = 31
- E - G_1 = 19
- F - G_1 = 17
- G_1 - G_1 = 0

state we have to choose 2 best paths and cut all other paths.

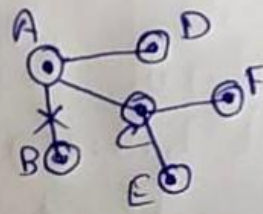


Step1: Evaluate initial state A.

Now, from state A we can traverse three another state B, C, D. According to the huristic value we go for C [Because, C-Gr = 28] and according to the Beam value we have to choose two best paths, here C and D and cut B.

A	C	D

Step2: From C we are going to check another states which are E and F but as F-Gr path has lesser huristic value 17, we go to consider F first.

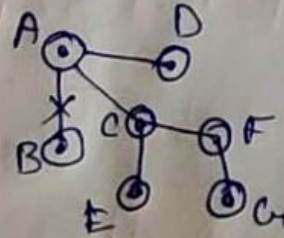


A	C	F	E	D

Sorting

Step3: From F we can go directly to the goal state Gr.

A	C	F	Gr	E	D



As we find the goal state the algorithm stopped here. And we get a good solution path A-C-F-Gr. (It may or may not be the optimal)

1) Different components of ai

2) DLS

3) Simulating Annealing algo

4) Local beam Search

5) α-β pruning

6) Uniform cost search

7) Blind search

8) Constraint Satisfaction problem

9) Minimax Problem. (with graph)

10) Show that if a heuristic is Consistent then f(n) is monotonically decreasing along any path.

11) A problem solving search is proceed either in forward or in backward direction. Justify.

12) A* search problem Solve. Define state space. Define Heuristic. Find whether this heuristic is admissible or not?

13) Distinguish between procedural and declarative knowledge.

14) What is a production System?

15) Missionary Cannibble, Water-Jug.