Search strategies.

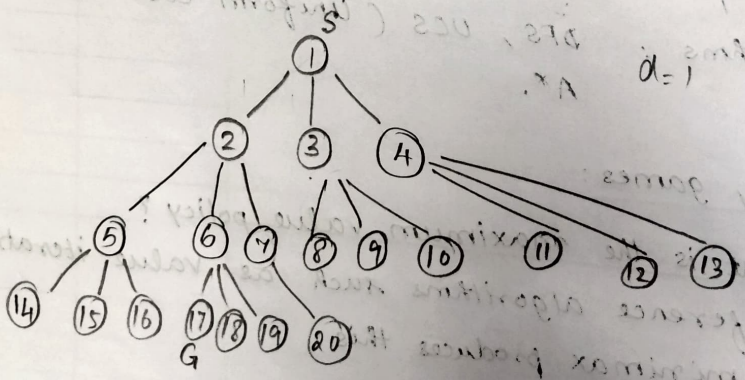
i. Uninformed search / Blind search.

ii. Informed search / Heuristic search.

Consider the state space where the start state is 1 and the successor function for the state i returns 3 states namely $3i-1$, $3i$, $3i+1$.

i. Draw the state space graph for 1 to 20. Let the goal be 17.

ii. List the nodes generated according to BFS, DFS.



ii

BFS

~~1~~, ~~2~~, ~~3~~, ~~4~~, ~~5~~, ~~6~~, 7, 8, 9, 10, 11, 12, 13, ~~14~~, ~~15~~, ~~16~~, 17

DFS

1

2

~~5~~

~~14~~

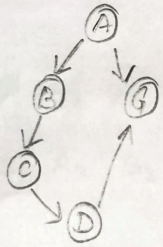
~~15~~

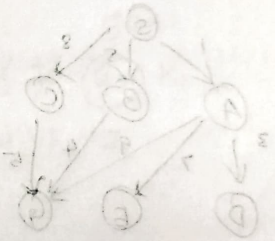
~~16~~

6

17 G

BFS on





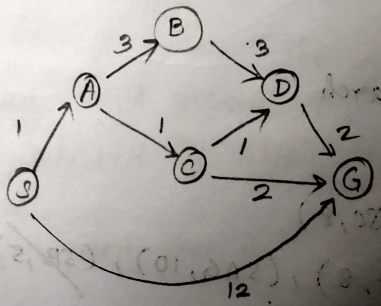
Frontier

F : ~~A~~ B G

E A

↓

Explored.



(S, 0) (SA, 1) (SG, 12)

(SAB, 4) (SAC, 2) (SG, 12).

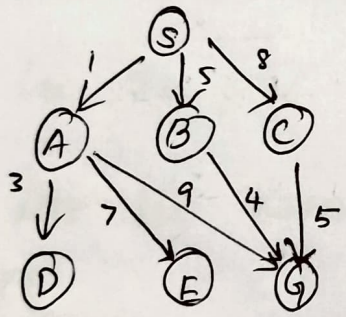
(SAB, 4) (SACD, 3) (SACG, 4) (SG, 12)

(SAB, 4) (SACDG, 5) (SACG, 4) (SG, 12).

(SABD, 7), (SABCDG, 5) (SACG, 4) (SG, 12).

S → A → C → G

Write the sequence of nodes visited for DFS and BFS.



DFS :

S A D E G

BFS :

S A B C D E G

Uniform cost search :

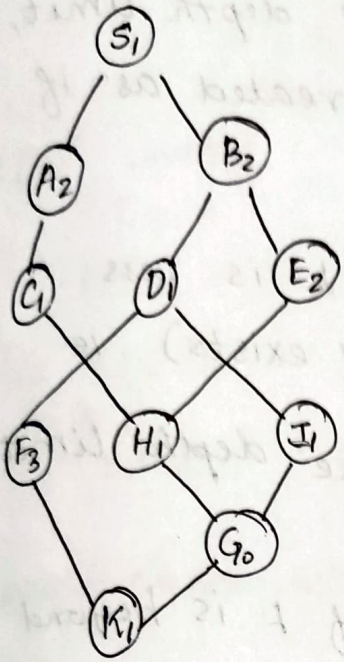
(S, 0)

(SA, 1) (SB, 5) (SC, 8).

(SAD, 4), (SAE, 8), (SAG, 10), (SB, 5), (SC, 8)

~~(SAE,8), (SAG,10), (SBG,9), (SC,8)~~

(SAG,10) , (SBG,9), (SCG,13).



BFS.

goal

Frontier ~~S1~~ ~~A2~~ ~~B2~~ ~~C1~~ ~~D1~~ ~~E2~~ ~~H1~~ ~~F3~~ ~~I1~~ (G0) K1

Explored S_1 A_2 B_2 C_1 D_1 E_2 H_1 F_3 I_1

DFS.

TOP

Frontier ~~E~~ ~~G~~ ~~H~~ ~~C~~ ~~A~~ B S

Explored. SACHEG

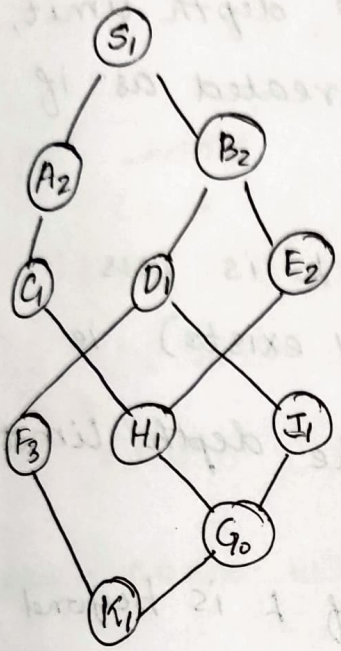
i. All algorithms don't generate paths with loops

ii. Nodes are selected in alphabetical order in case of tie.

iii. A node is visited when it is at the front of the queueue.

~~(SAE, 8)~~, (SAG, 10), (SBG, 9), ~~(SC, 8)~~.

(SAG, 10), (SBG, 9), (SCG, 13).



BFS.

goal.

Frontier ~~S1~~ ~~A2~~ ~~B2~~ ~~C1~~ ~~D1~~ ~~E2~~ ~~H1~~ ~~F3~~ ~~I1~~ G_0 K_1

Explored S_1 A_2 B_2 C_1 D_1 E_2 H_1 F_3 I_1

DFS.

Top

Frontier ~~S~~ ~~G~~ ~~H~~ ~~E~~ ~~A~~ ~~B~~ ~~S~~

Explored: SACHEG

i. All algorithms don't generate paths with loops

ii. Nodes are selected in alphabetical order in case of tie.

iii. A node is visited when it is at the front of the queueue.

Drawback of DFS is.

i. the infinite space.

The infinite space can be overcome by supplying DFS with predefined depth limit, L. ie. nodes at depth L are treated as if they have no successors.

It becomes incomplete if L is less than D (~~level~~ depth at which the goal exists). ie the shallowest goal is beyond the depth limit.

It becomes non optimal if L is beyond D.

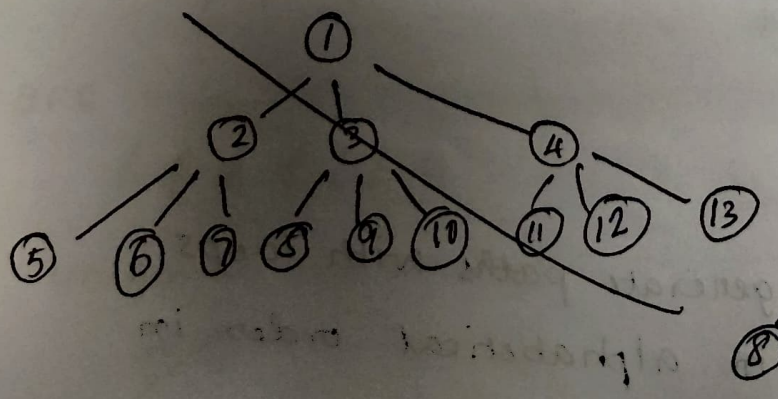
Depth Limited Search

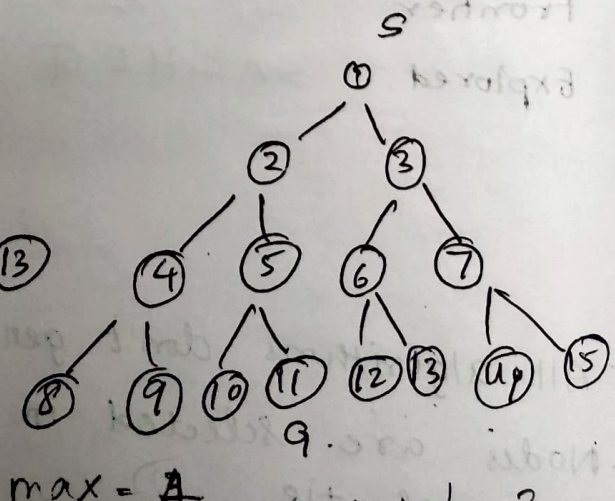
Iterative deepening search :

i. To have DFS with best depth limit.

ii. It does increase gradually the depth limit until goal is reached.

Consider





max = 4

b = 2.

d = 3

BFS : 1 2 3 4 5 6 7 8 9 10 11

DFS : 1 2 4 8 9 5 10 11

DLS with $l = 3$: same as DFS.

$l = 2$: 1 2 4 5 3 6 7 X

IDS ~~with~~

$L = 0$: 1

$L = 1$: 1 2 3

$L = 2$: 1 2 4 5 3 6 7

$L = 3$: 1 2 4 8 9 5 10 11

seq. nodes visited : 1; 123; 1245367;
1248951011.

The total no. of nodes generated in the worst case is

$$N(IDS) = db + (d-1)b^2 + (d-2)b^3 + \ldots + 1b^d.$$

consider $b = 10$, $d = 5$.

$$N(IDS) = 50 + 4 \times 100 + 3 \times 1000 + 2 \times 10000 + 100000$$

$$= 1,23,450.$$

	Complete	Optimal	Time	Space
BFS	Y	Y^{C_1}	$O(b^d)$	$O(b^d)$
DFS	N	N	$O(b^m)$	$O(bm)$
DLS	Y^C	N	$O(b^l)$	$O(bl)$
IDS	Y	Y	$O(b^d)$	$O(b^d)$
UCS	Y	Y		
BDS			$O(b^{d/2})$	$O(b^{d/2})$

$C = l \geq D$

C_1 = same cost for all edges

Bidirectional Search (BDS)

Search from start & search from goal simultaneously

The idea behind BDS is to run 2 simultaneous searches; one in the forward direction from the start state and the other in the backward direction from the goal hoping that they meet.

For example ~~cost~~ consider a tree with $b = 10$ and $d = b$. The total number of nodes generated in BFS.

is $10 + 100 + 1000 \ldots + 1000000 = 11,11,110.$

but in case of BDS. it is.

$$\begin{matrix} 1000 \\ 100 \\ 10 \end{matrix} \Bigg| \begin{matrix} 1000 \\ 100 \\ 10. \end{matrix} = 2,220.$$

BFS :

i. complete - yes (if the branching factor is finite).

ii. optimality - yes (if all step costs are identical).

iii. time - $O(b^d)$. d : depth of the shallowest goal.

iv. space - $O(b^d)$.

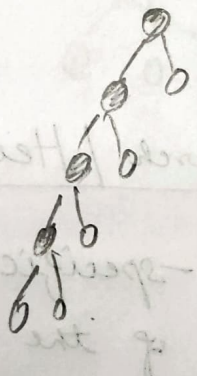
DFS :

i. complete - ~~yes (if the depth~~ No.

ii. optimality - No.

iii. time - $O(b^d)^m$ where m is the max. depth of the tree. (height).

iv. space - $O(b \times m)$.



DLS :

i. complete - yes (if $l \geq d$).

ii. optimality - No.

iii. time - $O(b^l)$

iv. space - $O(b \times l)$.

IDS :

comp: yes (if b is finite)

op : yes (if step cost are equal)

T : $O(b^d)$.

S : $O(b^d)$.

BDS :

comp : yes (if ~~branch for~~ bf. 'b' is finite)

optimal : yes (if all step costs are identical and both are bfs).

T : $O(b^{d/2})$

S : $O(b^{d/2})$.

Informed search / Heuristic search :

uses problem-specific knowledge beyond the definition of the problem itself.

expand closer seeming node first.

Heuristic function $h(n)$ gives an estimate of the distance from n to the ~~goal~~ goal.

$h(n) = 0$ for goal nodes.

examples : for TSP., straight line distance.

In Informed search we are adding something new to the problem.

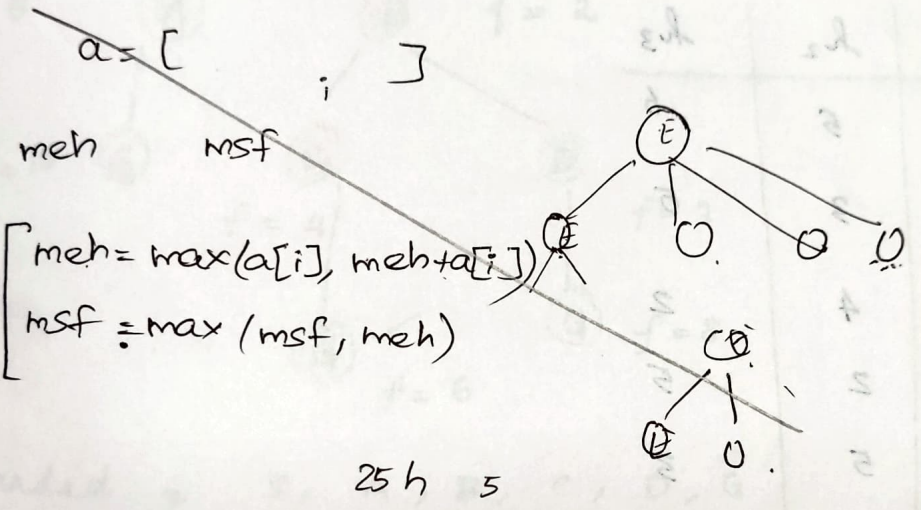
Best First Search.

The most widely used & known form of best first search is A* search.

evaluation fn. $f(n) = g(n) + h(n)$.

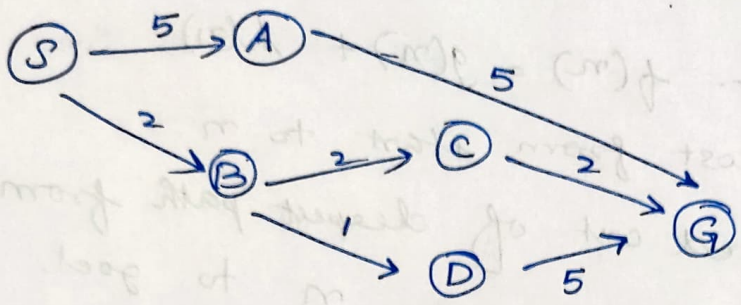
$g(n)$ → path cost from start to n.

$h(n)$ → estimated cost of deepest path from n to goal.



Greedy : evaluation function $f(n) = h(n)$.

A* : " " $f(n) = g(n) + h(n)$



	h_1	h_2	h_3
S	0	5	6
A	0	3	5
B	0	4	2
C	0	2	5
D	0	5	3
G	0	0	0

Find out the path according to uniform cost search.

$h(n) = 0$. $\Rightarrow$ in A* $f(n)$ becomes $g(n)$.

A* behaves like uniform cost search.

i. ~~(S, 0)~~

ii (SA, 5), ~~(SB, 2)~~

iii (SA, 5), (SBC, 4), ~~(SBD, 3)~~

iv (SA, 5), ~~(SBC, 4)~~, (SBDG, 8)

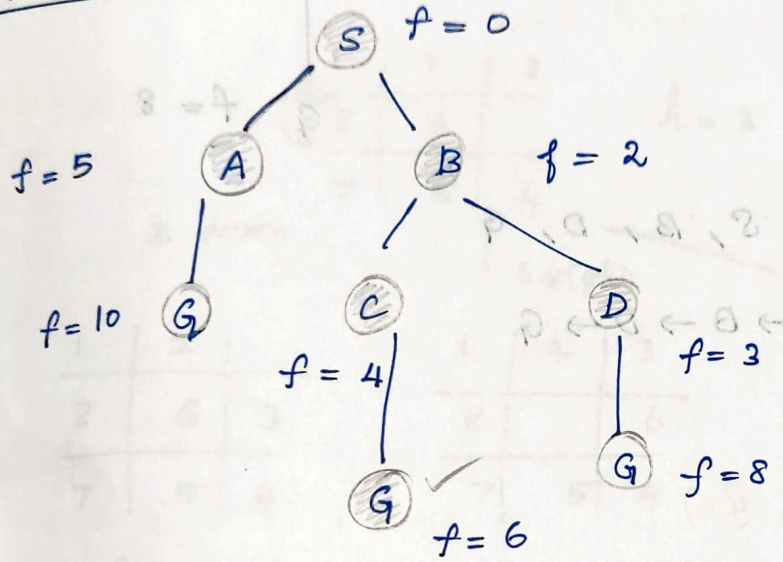
v ~~(SA, 5)~~, (SBCG, 6), (SBDG, 8)

vi (SAG, 10), (SBCG, 6), (SBDG, 8).

SBCG , 6 .

Implement A* with all the three heuristic function.

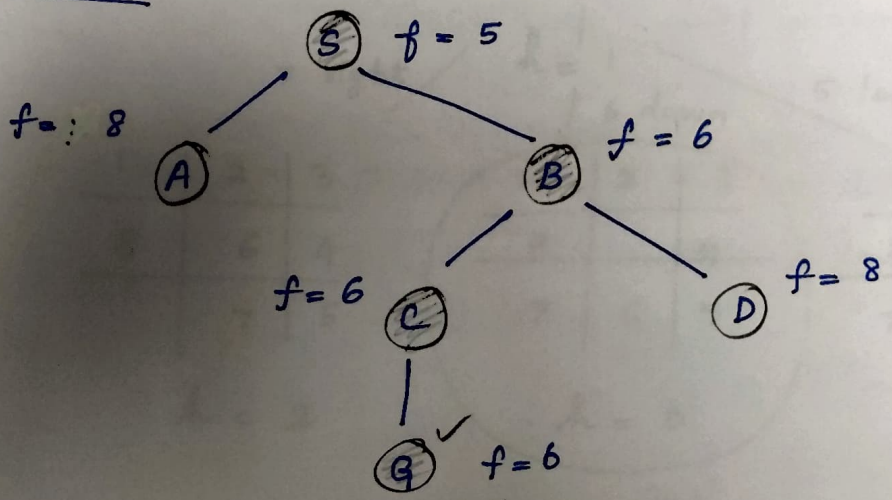
with h_1



expanded = S, A, B, C, D, G

path = S → B → C → G .

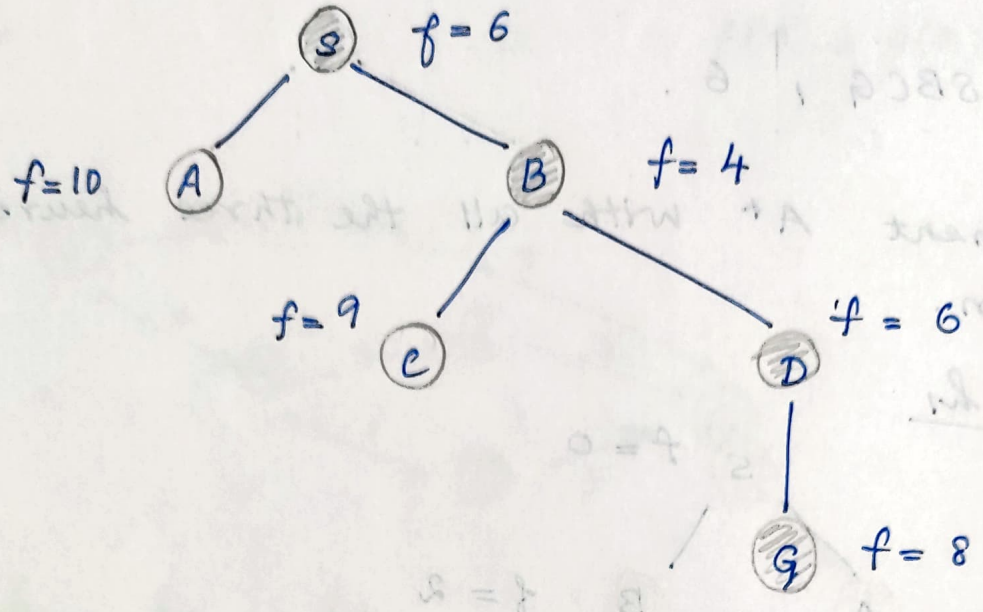
with h_2



expanded = S, B, C, G

path = S → B → C → G

with h_3 :



expanded : S, B, D, G

path : S → B → D → G

UCS: $f(n) = g(n)$

Greedy best : $f(n) = h(n)$

A* : $f(n) = g(n) + h(n)$.

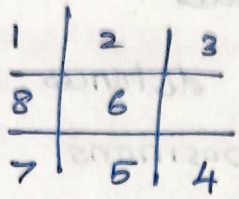
S

1	2	3
8	6	
7	5	4

G

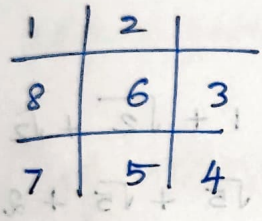
1	2	3
8		4
7	6	5

h : no. of misplaced tiles.

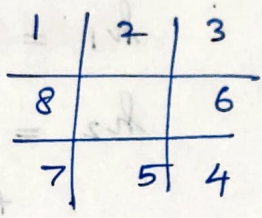


h = 3

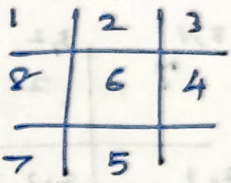
3 down | 6 right | 4 up



h = 4

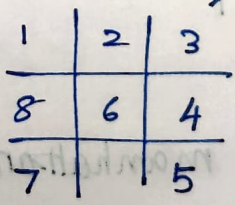


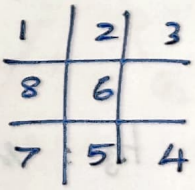
h = 3



h = 2

5 right | 4 down

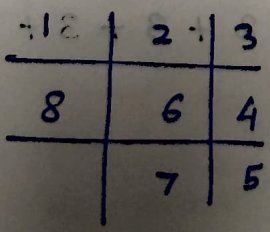




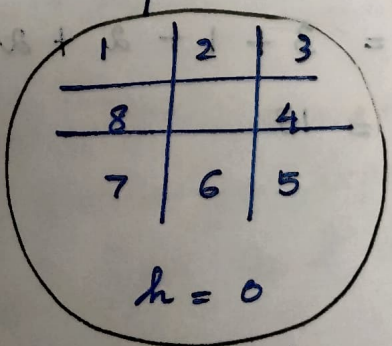
×

h = 1

7 right | 6 down | 5 left

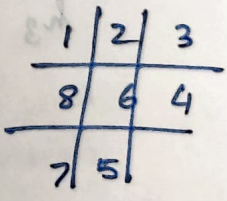


h = 2



h = 0

goal



×

4 up → 5 right → 6 down.

Start

7	2	4
5		6
8	3	1

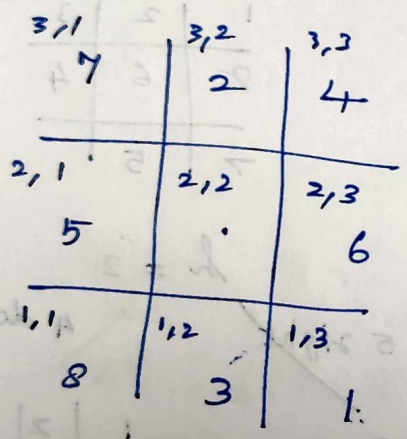
Goal

	1	2
3	4	5
6	7	8

Consider 3 heuristic functions.

H_1 = no. of misplaced tiles.

H_2 = sum of euclidean distances of the tiles from their goal positions.



$h_1 = 8$

$h_2 = \sqrt{5} + 1 + \sqrt{2} + \sqrt{2} + 2 + \sqrt{5} + \sqrt{5} + 2.$

$= 5 + 2\sqrt{2} + 3\sqrt{5}$

$= 14.54.$

H_3 : sum of manhattan distances of the tiles from their goal positions.

$h_3 = 3 + 1 + 2 + 2 + 2 + 3 + 3 + 2$

$= 18$

i. A heuristic function $h(n)$ is admissible if for every node n $h(n) <= h^*(n)$ where $h^*(n)$ is the true cost to reach the goal from n.

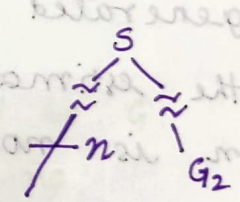
ii. An admissible heuristic function never over estimates the cost to reach the goal. ie. if n is the optimal solution reachable from n'

$$g(n) \geq g(n') + h(n').$$

iii. A* is admissible (optimal) if it uses an admissible heuristic.

Proof of optimality:

Let s be the starting state and g be the optimal goal state.



A* $f(n) = g(n) + h(n)$.

$$f(G) = g(G) = f^* \quad (\because h(G) = 0).$$

$$f(G_2) = g(G_2) \quad (\because h(G_2) = 0).$$

$$> f^* \longrightarrow (1)$$

Suppose for a contradiction that A* has selected G_2 from the queue for expansion. This will terminate A* to have G_2 as the goal.

Let us consider any node n on the optimal path to G. Since h is admissible, $f^* \geq f(n)$

if n is not chosen over G_2. Then we should have, $f(n) \geq f(G_2)$.

$$\therefore f^* \geq f(n) \geq f(G_2).$$

$$\Rightarrow f(G_2) \leq f^*$$

a contradiction to ①.

$\therefore$ A* is optimal.

Dominance:

if h_1 and h_2 are 2 admissible heuristic, if $h_2(n) \geq h_1(n)$. for all n, then h_2 dominates h_1.

Monotonicity of heuristic:

A heuristic $h(n)$ is called consistent/ monotonic if $\forall$ node n. and $\forall$ n' child n' generated by an action, a from the node n the estimated cost of reaching the goal from n is not greater than the step cost of n and n' and estimated cost of reaching the goal from n'.

$$h(n) \leq c(n, a, n') + h(n').$$

If h is consistent, then the value of $f(n)$ along any path is non decreasing.

$$g(n') = g(n) + c(n, a, n')$$

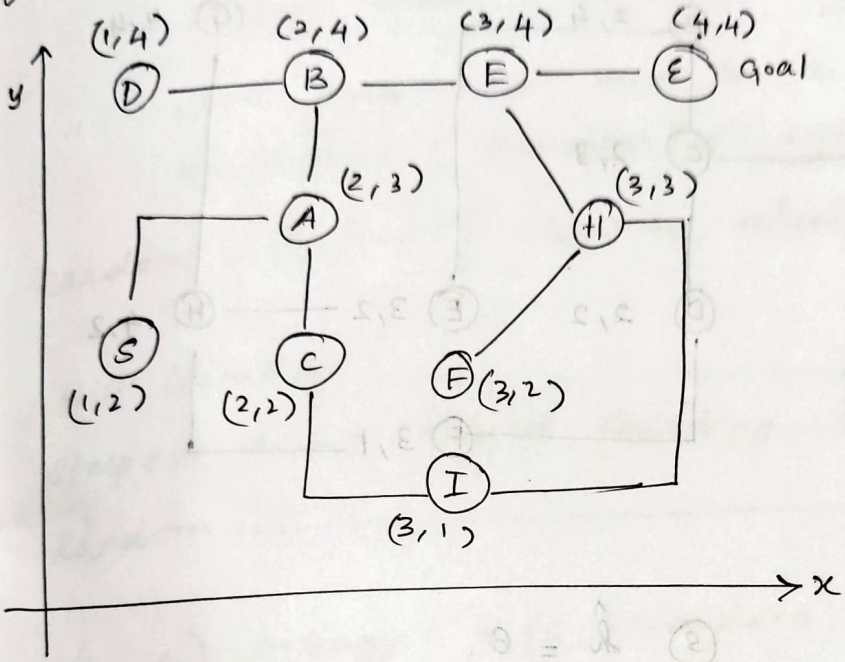
$$f(n') = g(n') + h(n')$$

$$= g(n) + c(n, a, n') + h(n').$$

$$\geq g(n) + h(n) = f(n).$$

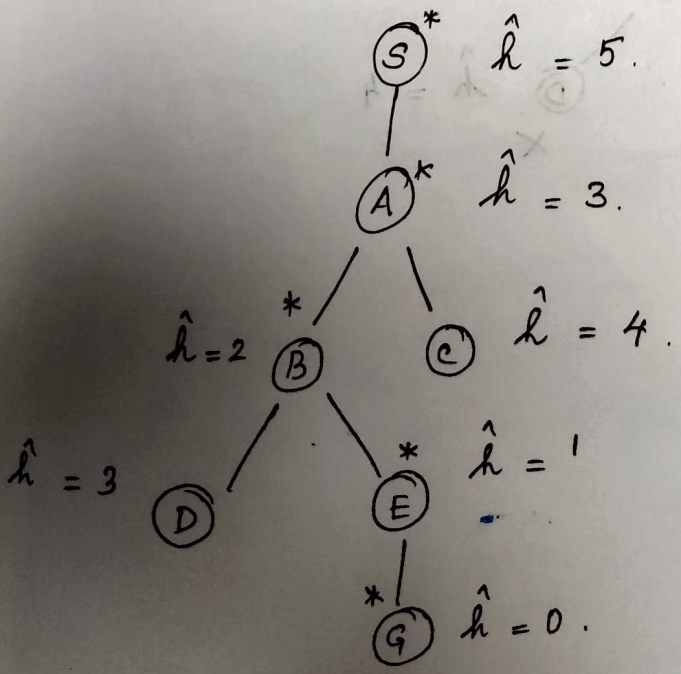
Difference between gradient descent and hill climbing approach.

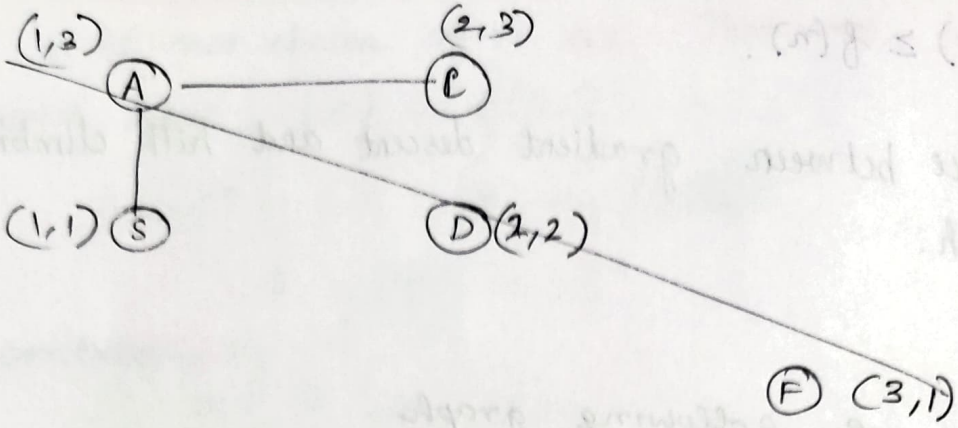
consider the following graph

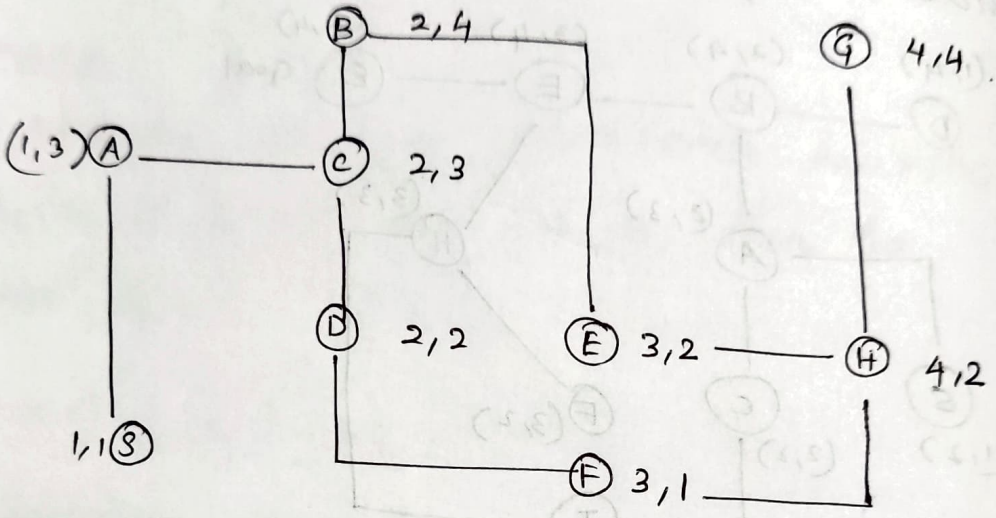


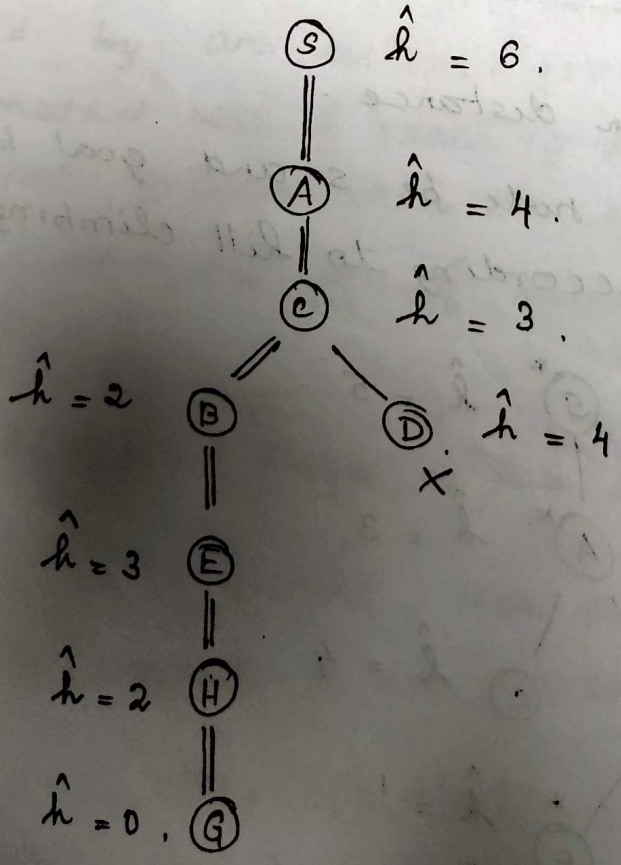
$\hat{h}$ = Manhattan distance.

Let the starting node be S and goal be G. Find the goal according to hill climbing.




(1,3)
A
(2,3)
C
(1,1)
S
D (2,2)
F (3,1)


B 2,4
G 4,4
(1,3) A
C 2,3
D 2,2
E 3,2
H 4,2
1,1 S
F 3,1


S ĥ = 6.
A ĥ = 4.
C ĥ = 3.
ĥ = 2
B
D ĥ = 4
X
ĥ = 3
E
ĥ = 2
H
ĥ = 0.
G

Hill Climbing Algorithm :

When the state-space landscape has local minima, any search that moves only in the greedy direction cannot be complete.

At each step do one of 2.

i. With prob 'p' move to neighbour with largest value. (Greedy). exploitation.

ii. With prob '1-p' move to a random neighbour (Random). exploration.

Random selection / Greedy selection.

Hill Climbing ——— } not complete.
Steepest Ascent / Descent climbing ——— }

Random ——— complete.

(1 - ε) greedy + (ε) random
is called ε optimal.

Simulated Annealing :

Constraint Satisfaction Problem.

ex. Map colouring and other assignment problems.

Variables.

$A \in \{1, 2, 3\}$

$B \in \{1, 2, 3\}$

$C \in \{1, 2, 3\}$.

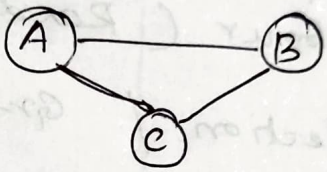
Constraints.

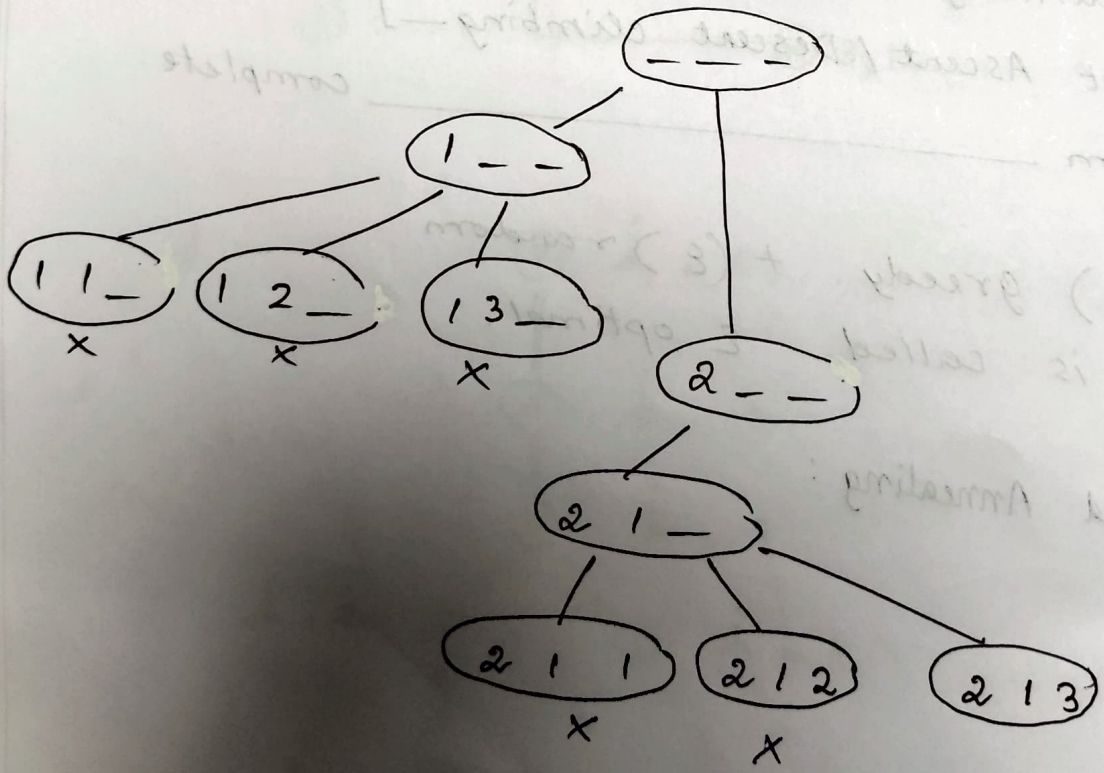
$C_1 : A > B$

$C_2 : B \neq C$

$C_3 : A \neq C$.

Constraint graph.





$C_3\ C_2\ C_1$

TWO

+ TWO

FOUR

The main variables are

F, O, T, U, W, R.

The carriers are C_1, C_2, C_3.

The constraints

Adverserial search and games.