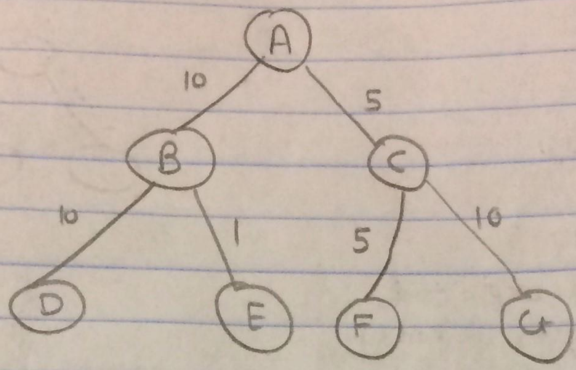
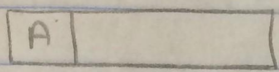
Q1]



* Breadth first search.

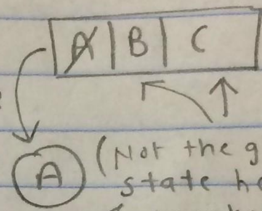
1.



Initial start node in queue

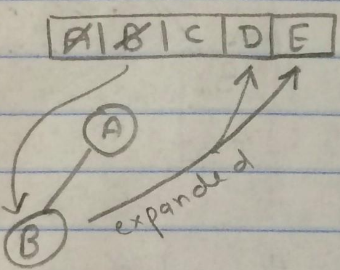
2.

deque



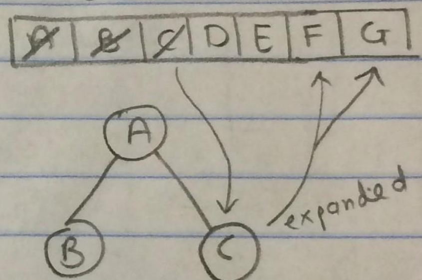
(Not the goal ~~state hence~~ expanded) & marked explored.

3.



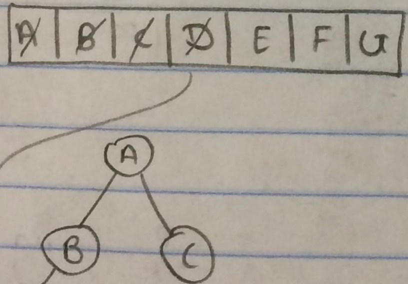
(Not goal state) marked explored

4.



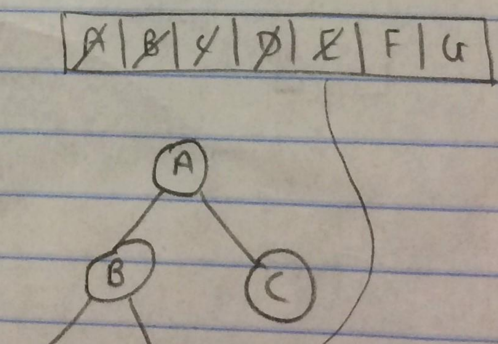
(not goal state, marked explored).

5.



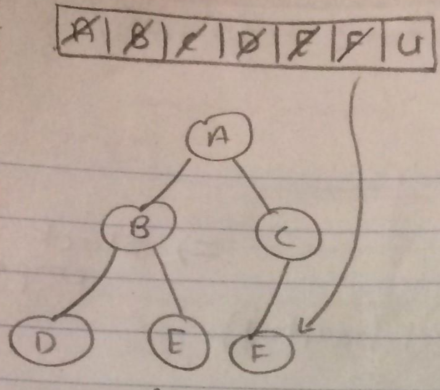
(not goal state, marked explored) nothing to expand

6.



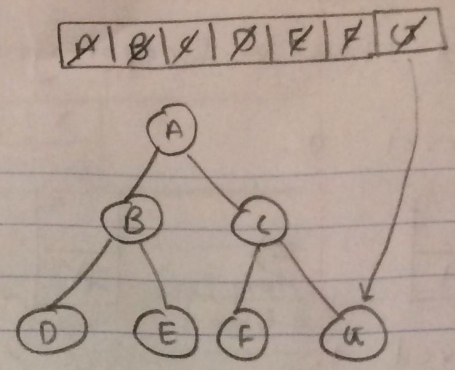
(not goal, marked explored nothing to expand).

7.



(not goal, marked explored, nothing to expand)

8



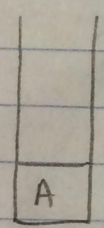
(goal state found breaking the loop)

Note: from step 2, keep dequeing the queue till it is empty.

Sequence →
A → B → C → D → E → F → G

* Depth First Search.

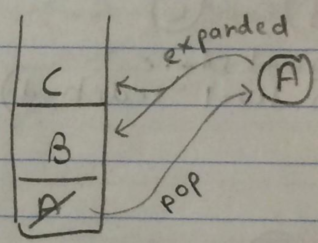
1.



LIFO stack

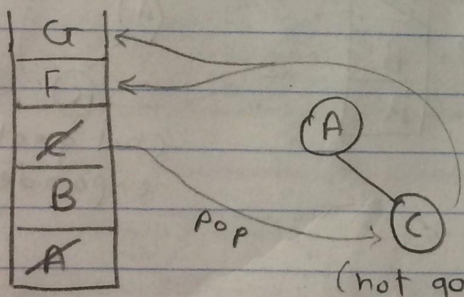
Initially start node in stack.

2.



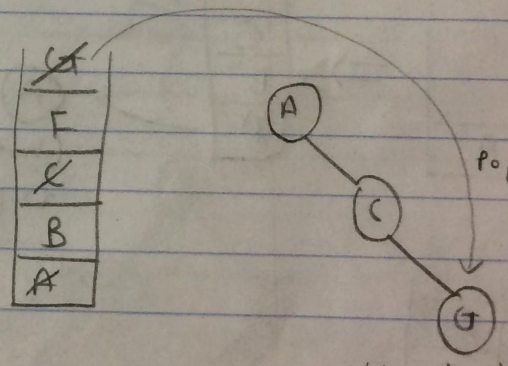
(not goal state marked explored

3.



(not goal state, marked explored, expanded)

4.



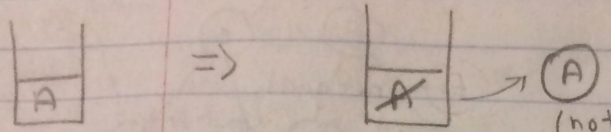
(goal state found).

Note: from step 2 start popping stack till it is empty or till goal state is found.

Sequence:
A → C → G

* Iterative Deepening Search.

1. level 0.

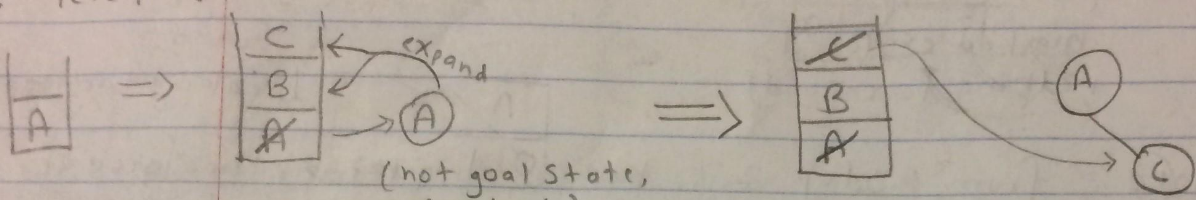


=> stack empty.

(not goal state, level haulted).

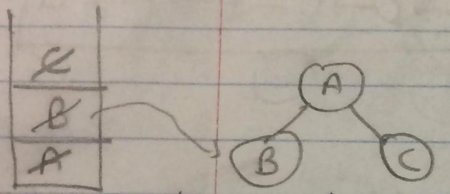
level 1.

2. level 1



(not goal state, expanded).

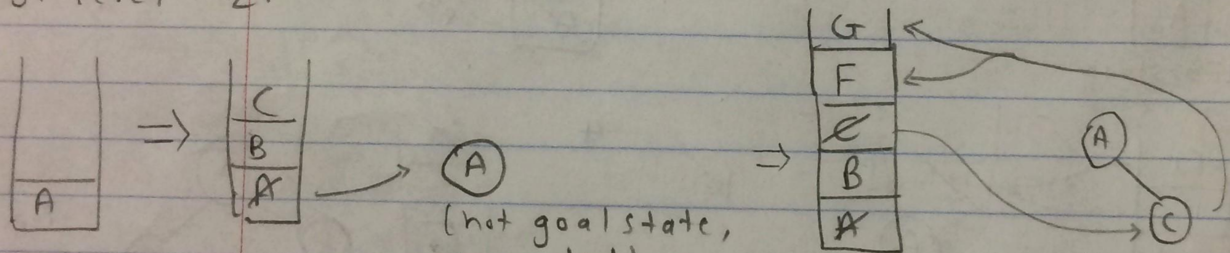
(not goal, depth level reached.



=> Stack empty.

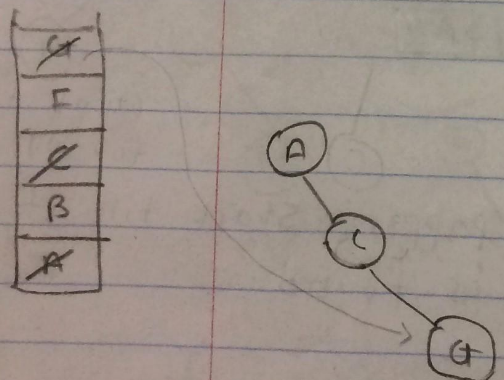
(not goal state depth level reached)

3. level 2.



(not goal state, expanded)

(not goal state, expanded).



=> Stop popping from the queue

(goal state found)

Path traversed.

A → C → G

Note :- We increase level, l from 0 till depth 'd' of the given graph.
We do not expand nodes beyond level l.

* Uniform Cost Search

1.

A	
0	0

fringe

2.

A		B		C	
0	0	10	10	5	5

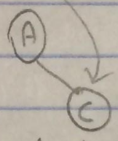
{A}.

(A) (not goal state, add in closed set, expand.

3. sorting fringe

A		C		B		F		G	
0	0	5	5	10	10	5	10	10	15

cumulative cost calculat^n



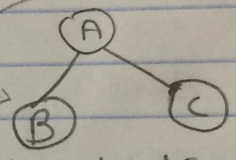
{A, C}

(not goal state, add in closed set, expand)

sorting fringe

4.

A		C		B		F		G		D		E	
0	0	5	5	10	10	5	10	10	15	10	20	1	11

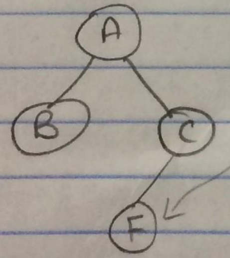


{A, C, B}

(not goalstate, add in closed set, expand)

5. Sort fringe

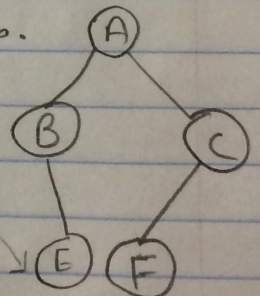
A		C		B		F		E		G		D	
0	0	5	5	10	10	5	10	1	11	10	15	10	20



{A, C, B, F}

(not goal state add in closed set leaf node)

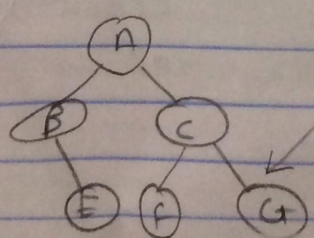
6.



{A, C, B, F, E}

(not goal state, add in closed set, leaf node)

7.



(goals state found)

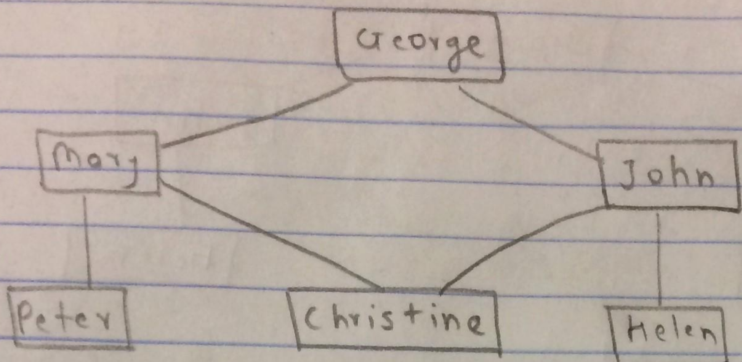
Nodes Visited in seq

A → C → B → F → E → G

backtrack path:

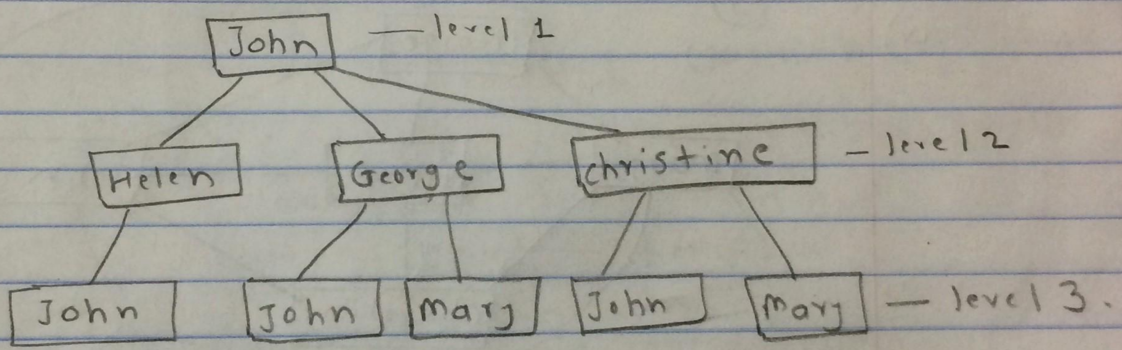
G → C → A

Question 2]



i) BFS, Iterative Deepening Search, Uniform cost search for above given graph does provide correct number of degrees of separation.

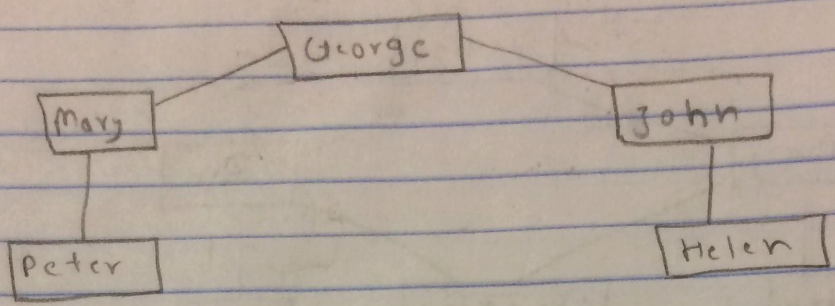
ii)



Search tree

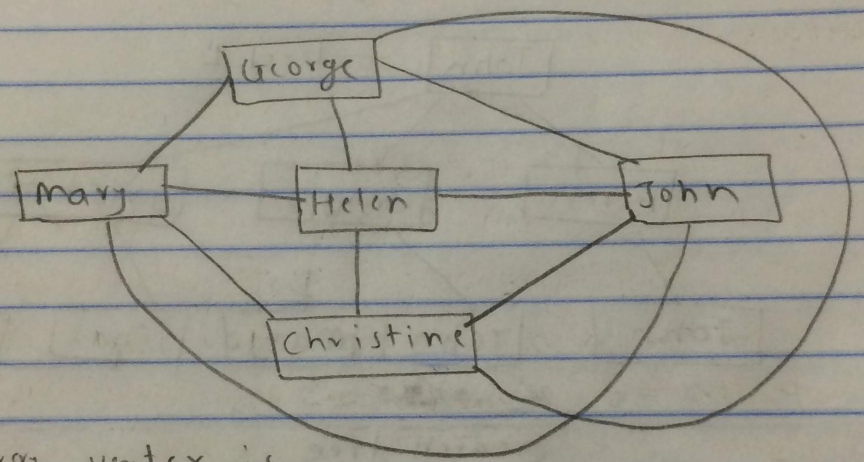
Nodes in above search tree doesnot have one to one corrospondence with vertices in SNG, meaning that though vertices in SNG are nodes in search tree SNG and search tree are not similar/analogous

Search tree doesnot have cyclic dependencies as in SNG. Search tree redundently specifies paths between two nodes, hence there are multiple nodes represen-ting same vertex in SNG iii)



In above SNG Peter has 4 degree of separation from Helen.
Helen has 4 degrees of separation from Peter.

iv)

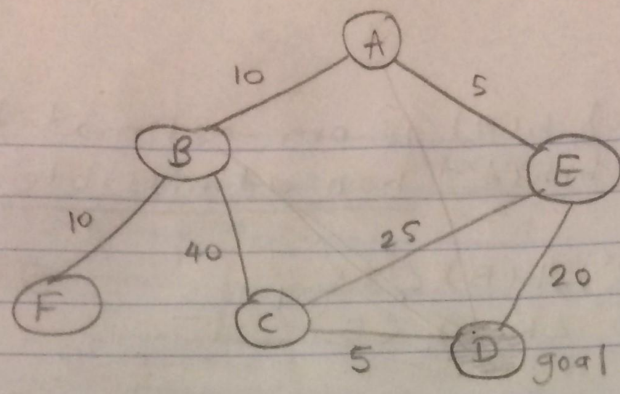


Every vertex is directly connected to every other vertex
fully connected graph.

v) For BFS search tree, if we have branching factor 'b' & depth as 'm'
total number of nodes $= b^m$
we want to store 1 million i.e 10^6 nodes, each representing a person

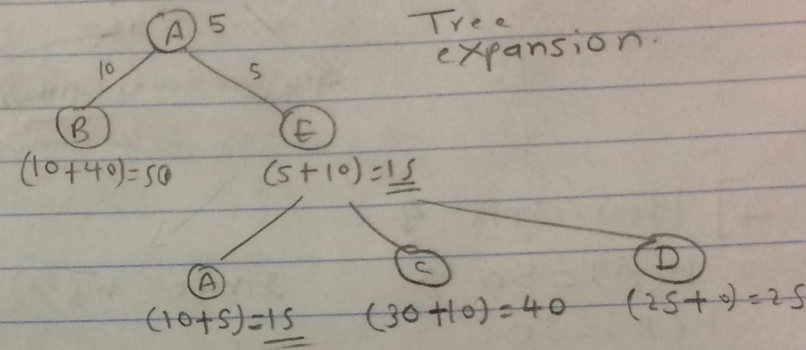
$\therefore b^m = 10^6$
$\therefore$ we keep branching factor = 10
& depth = 6 .

Q 3]



Heuristic 1]

$h(A) = 5$
$h(B) = 40$
$h(C) = 10$
$h(D) = 0$
$h(E) = 10$
$h(F) = 0$



since C is directly connected to D & with distance 5 hence h(C) is over-estimated hence

Non admissible

make $\boxed{0 < h(C) < 5}$

Heuristic 2]

$h(A) = 8/11$
$h(B) = 5$
$h(C) = 3$
$h(D) = 5/0$
$h(E) = 5$
$h(F) = 0$

Since node 'D' is goal state h(D) should be '0' & hence h(D) = 5 is over-estimated.

∴ Not Admissible.

update $\boxed{h(D) = 0}$ to make it admissible

3] Heuristic 3

$h(A) = 35$
$h(B) = 30$
$h(C) = 20$
$h(D) = 0$
$h(E) = 0$
$h(F) = 50$

$h(C), h(A)$ is over-estimated.
hence <u>non admissible</u>

$0 < h(A) < 25$
$0 < h(C) < 5.$

4] Heuristic 4

$h(A) = 50$
$h(B) = 50$
$h(C) = 50$
$h(D) = 50/\underline{0}$
$h(E) = 50$
$h(F) = 50$

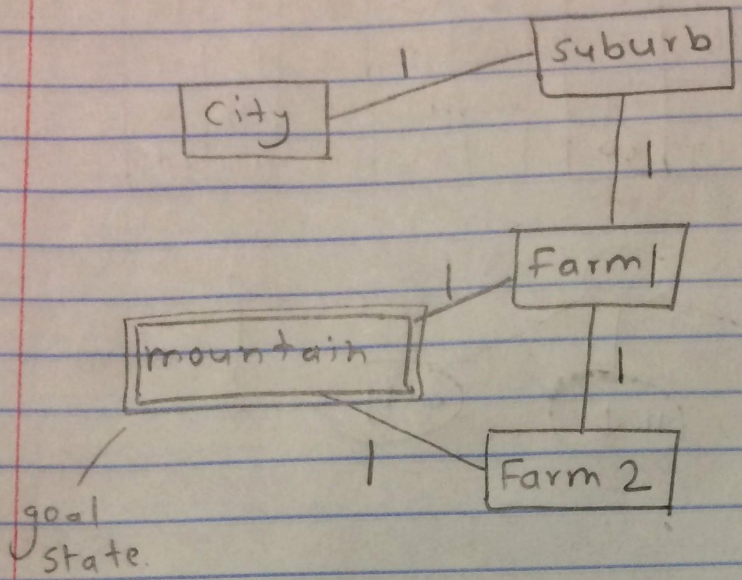
since $h(D) \neq 0$ it is not <u>Admissible</u>
~~ut~~ make $h(D) = 0$
$0 < h(A) < 25$
$0 < h(B) < 45$
$0 < h(C) < 5$
$0 < h(E) < 20$

5] Heuristic 5

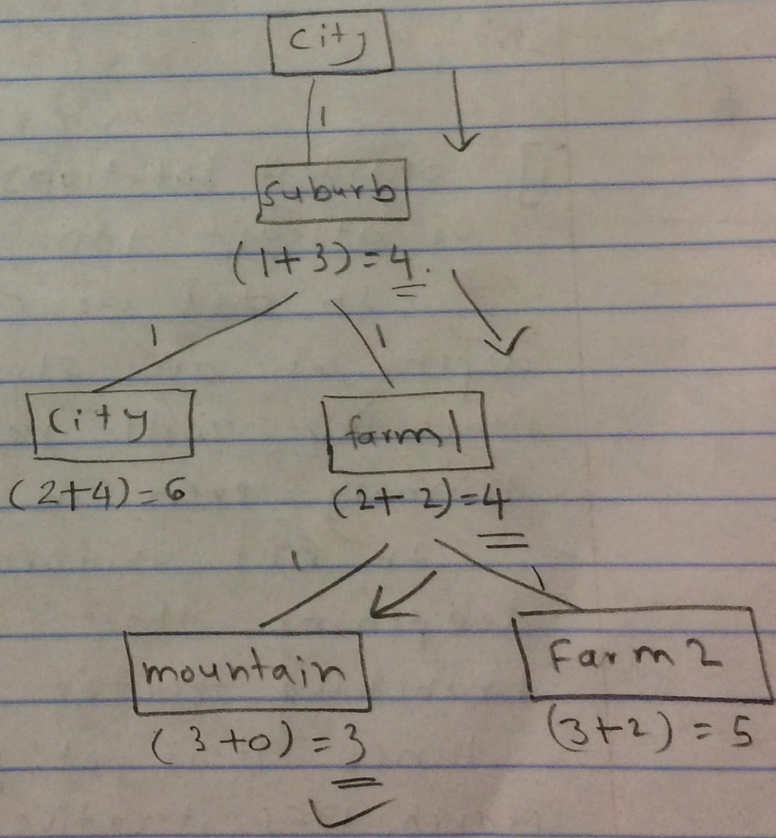
$h(A) = 0$
$h(B) = 0$
$h(C) = 0$
$h(D) = 0$
$h(E) = 0$
$h(F) = 0$

Since no cost is over-estimated this heuristic is Admissible.

Question 4]



→ h(mountain) = 0
h(city) = 4
h(farm1) = 2
h(suburb) = 3



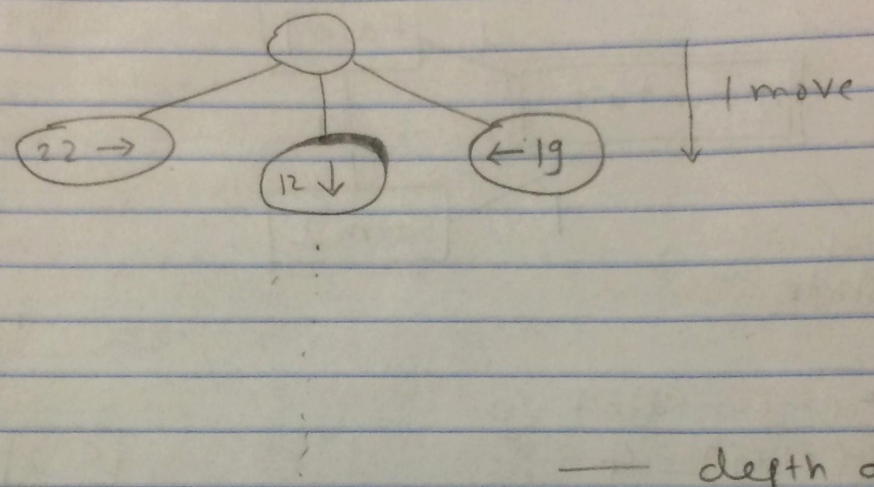
Question 5]

15	16	17	24	18
6	7	8	9	10
1	2	3	4	5
23	11	12	13	14
21		22	19	20

15	16	17	24	18
6	7	8	9	10
1	2	3	4	5
23	11		13	14
21	22	12	19	20

15	16	17	24	18
6	7	8	9	10
1	2	3	4	5
23	11	12	13	14
21	22	19		20

BFS :-



1] Shortest solution also needs more than or atleast 100 moves.

In dfs we can consider that at each depth we are storing only one node hence we will need atleast 100 kb of memory even in dfs.

In BFS, uniform cost search & iterative deepening there will be definitely branching factor more than 1 is needed.

hence it is not possible for BFS, Uniform cost search, DFS, iterative deepening search to store search nodes in 50 kb.

2] for all initial states, it takes <u>at most</u> 208 moves
hence depth of the search tree = 208
there are 4 possibilities to move tile
up, down, right, left.
hence branching factor would be $\underline{\underline{4}}$.

i) BFS → Space needed to store all nodes in search tree would be $O(b^d)$
$= 4^{208}$ Kb.
hence cannot guarantee.

ii) dfs → Since in dfs if a path from root to leaf isnt the path that includes goal state we can discard it hence, dfs needs space of $O(bm)$.
here $O(4 \times 208) = O(832)$ Kb.
But since the time complexity of dfs depends on size of state space which can be infinite, it is not guaranteed to obtain solution in finite time.

iii) Iterative deepening search - It needs space complexity of $O(bm) = O(832)$ Kb
It is same as dfs only with incremental levelling.

iv) Uniform Cost Search:- Space complexity needed is $O(b^d) = O(4^{208})$
since it same as BFS with priority queue

Hence none of the above BFS, DFS, Uniform cost search, iterative deepening search can guarantee that we will never need more than 1200 kb space to store search nodes.