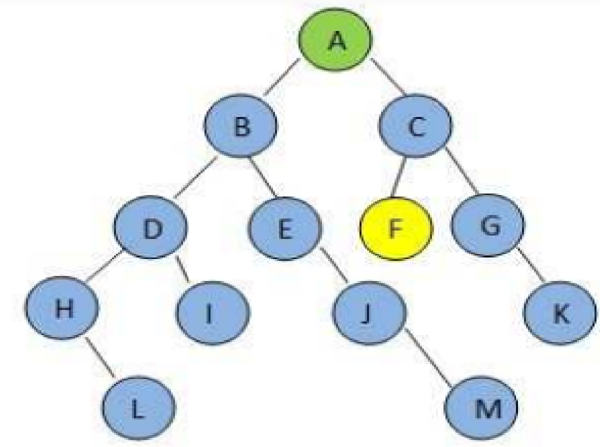
Artificial Intelligence

Solving Problems by Searching (Chapter 3)

1. In the following graphs, assume that if there is ever a choice amongst multiple nodes, both the BFS and DFS algorithms will choose the left-most node first.



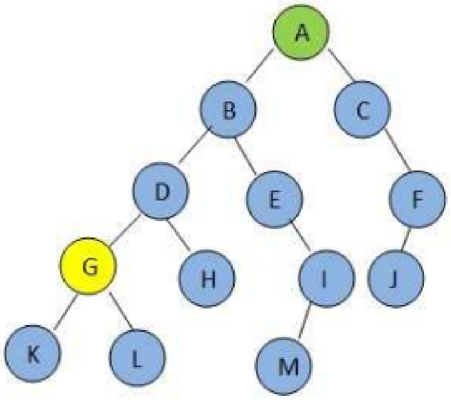
Starting from the green node at the top, which algorithm will visit the least number of nodes before visiting the yellow goal node? BFS

Before countering goal node F:

BFS algorithm encounters nodes: ABCDE

DFS algorithm encounters nodes: ABDHLIEJMC

2. In the following graphs, assume that if there is ever a choice amongst multiple nodes, both the BFS and DFS algorithms will choose the left-most node first.



Starting from the green node at the top, which algorithm will visit the least number of nodes before visiting the yellow goal node? DFS
Before countering goal node G:

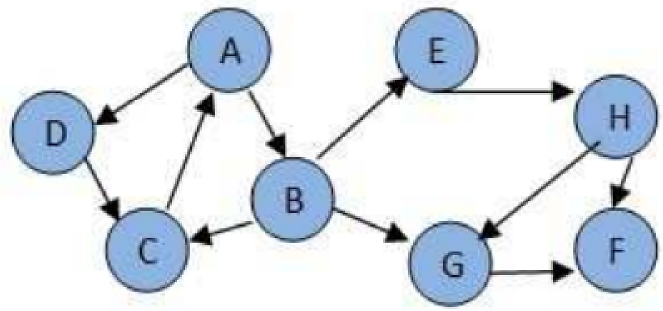
BFS algorithm encounters nodes: ABCDEF

DFS algorithm encounters nodes: ABD

For BFS algorithm, visiting a node’s siblings before its children, while in DFS algorithm, visiting a node’s children before its siblings

3. .

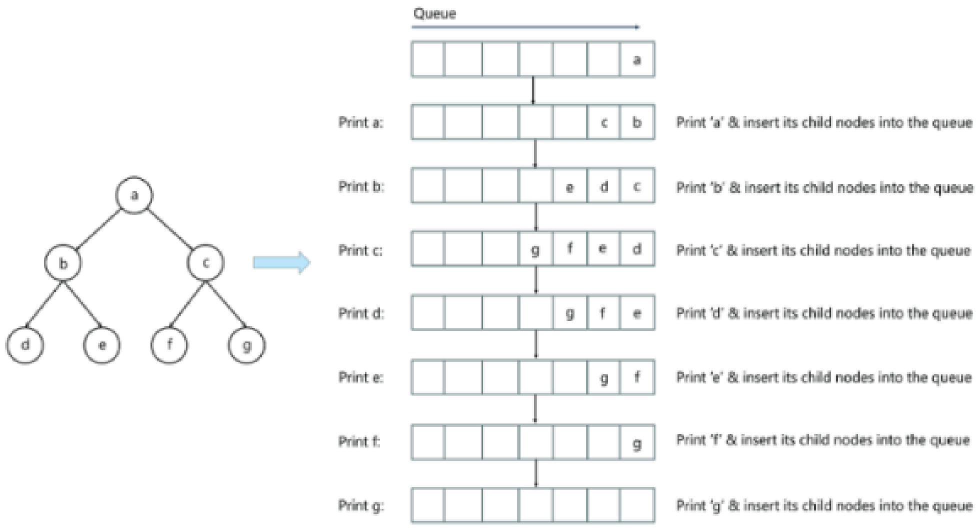
Consider the following graph. If there is ever a decision between multiple neighbor nodes in the BFS or DFS algorithms, assume we always choose the letter closest to the beginning of the alphabet first.
In what order will the nodes be visited using a Breadth First Search? In what order will the nodes be visited using a Depth First Search?



The answer is: ABDCEGHF

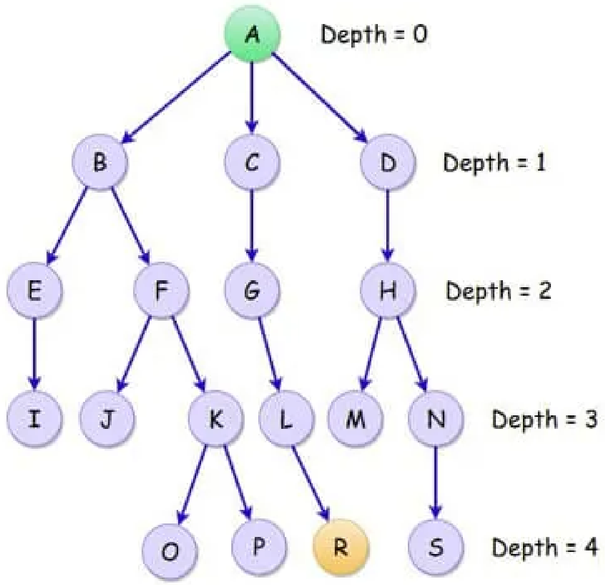
The answer is: ABCEHFGD

4. DFS



							a	
Print a:						c	b	Print 'a' & insert its child nodes into the queue
Print b:					e	d	c	Print 'b' & insert its child nodes into the queue
Print c:				g	f	e	d	Print 'c' & insert its child nodes into the queue
Print d:					g	f	e	Print 'd' & insert its child nodes into the queue
Print e:						g	f	Print 'e' & insert its child nodes into the queue
Print f:							g	Print 'f' & insert its child nodes into the queue
Print g:								Print 'g' & insert its child nodes into the queue

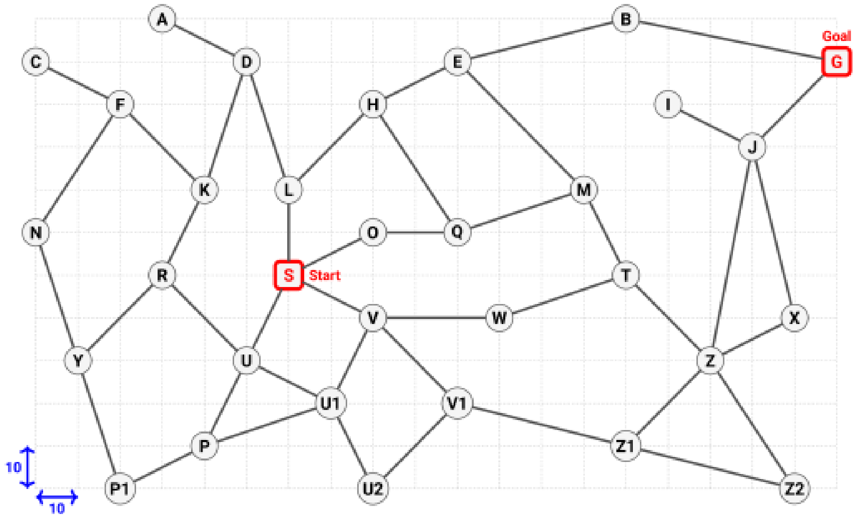
5. IDDFS



The tree can be visited as: A B E F C G D H

DEPTH = {0, 1, 2, 3, 4}

DEPTH LIMITS	IDDFS
0	A
1	A B C D
2	A B E F C G D H
3	A B E I F J K C G L D H M N
4	A B E I F J K O P C G L R D H M N S

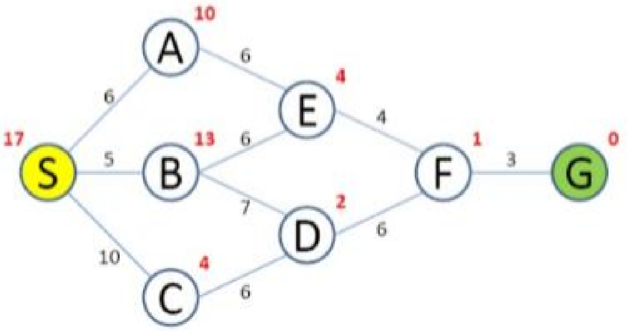


The figure shows a map with several locations on a grid where each tile is 10km x 10km in size. In this map, S is the start node and G is the goal node, the locations are connected by two way edges (or roads). The MoveGen returns nodes in alphabetical order. Assume that RemoveSeen procedure removes children already in OPEN or CLOSED list. Use Manhattan distance when needed.

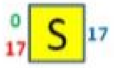
6. What is the heuristic value of node E? 90
7. In the map, starting with node S, list the first 7 nodes inspected (or expanded) by Hill Climbing algorithm. If the algorithm terminates early then list the nodes inspected. List the nodes in the order they were inspected.
   S,O,Q,M

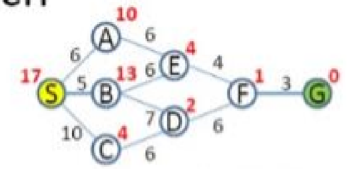
8. In the map, starting with node S, list the first 7 nodes inspected (or expanded) by Depth First Search (DFS) algorithm. List the nodes in the order they were inspected. If the algorithm terminates early then list all the nodes seen.
   S,L,D,A,K,F,C

9. In the map, starting with node S, what is the 15th node inspected (or expanded) by Depth First Search (DFS) algorithm? If the algorithm terminates early then name the last node inspected
   Z1
10. In the map, starting with node S, list the first 7 nodes inspected (or expanded) by the Breadth First Search (BFS) algorithm. List the nodes in the order they were inspected. If the algorithm terminates early then list all the nodes seen
    S,L,O,U,V,D,H
11. In the map, starting with node S, what is the 15th node inspected (or expanded) by Breadth First Search (BFS) algorithm? If the algorithm terminates early then name the last node inspected.
    K
12. A* Algorithm



A* Search

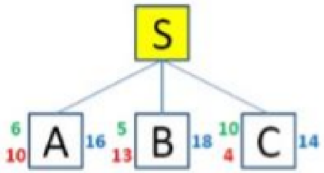


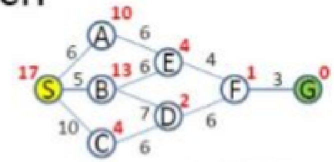


QUEUE:

S

A* Search





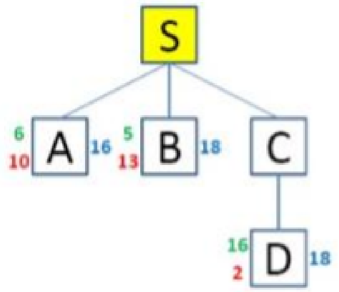
QUEUE:

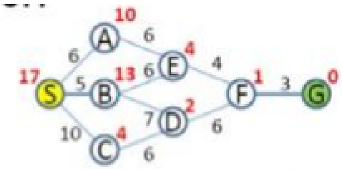
SC

SA

SB

A* Search



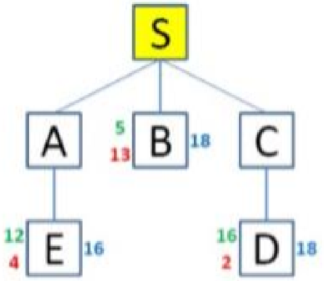


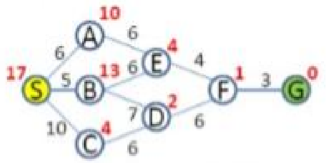
QUEUE:

SA

SCD

SB



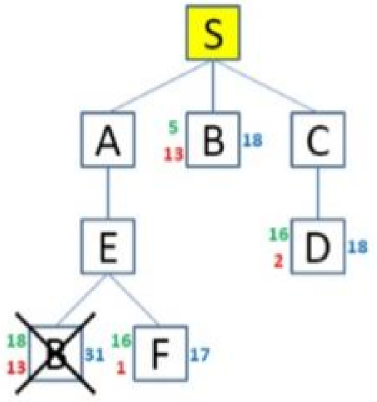


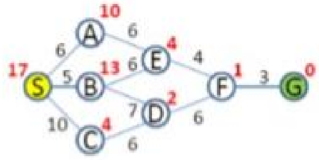
QUEUE:

SAE

SCD

SB





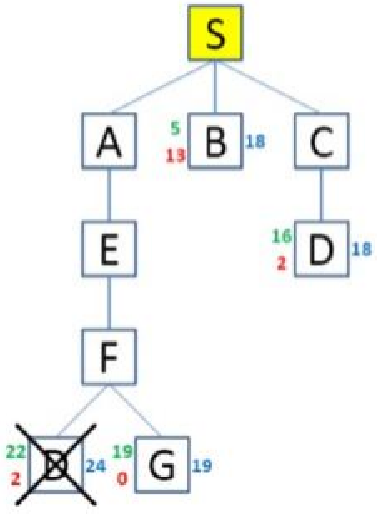
QUEUE:

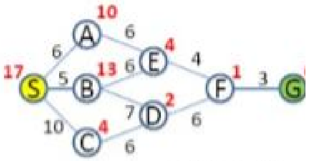
SAEF

SCD

SB

SAEB





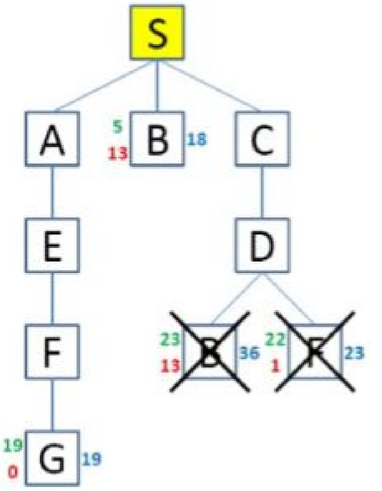
QUEUE:

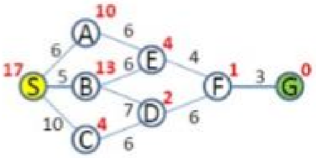
SCD

SB

SAEFG

SAEFD





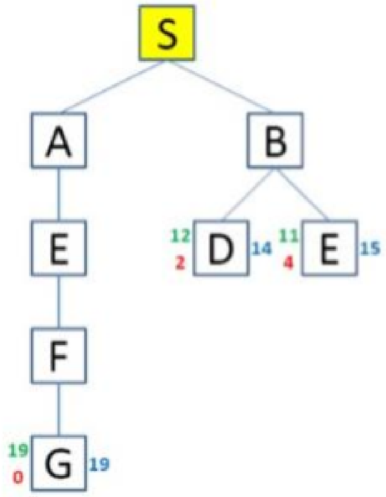
QUEUE:

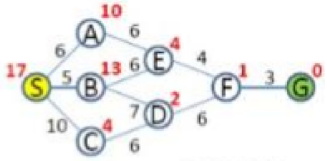
SB

SAEFG

SCDF

SCDB



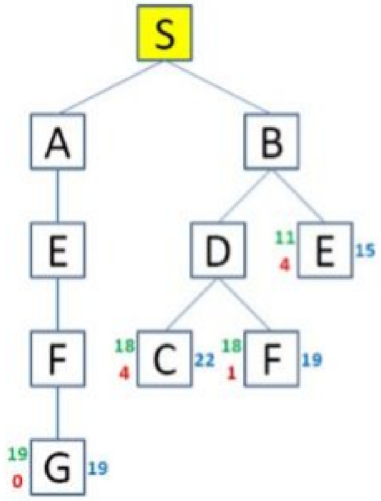


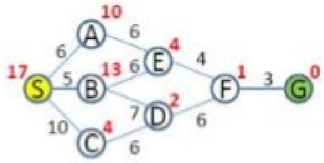
QUEUE:

SBD

SBE

SAEFG





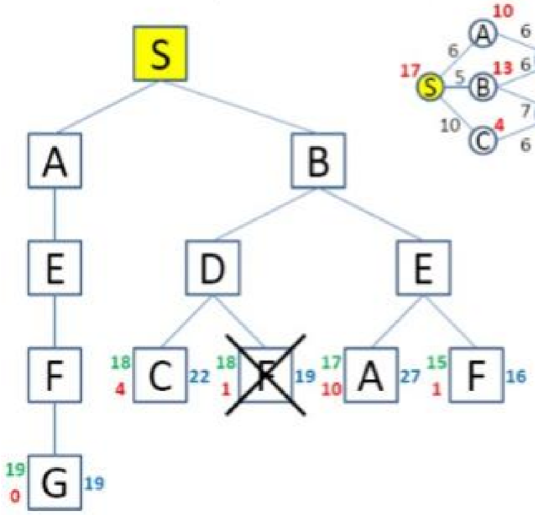
QUEUE:

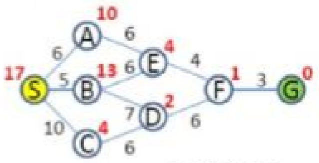
SBE

SBDF

SAEFG

SBDC





QUEUE:

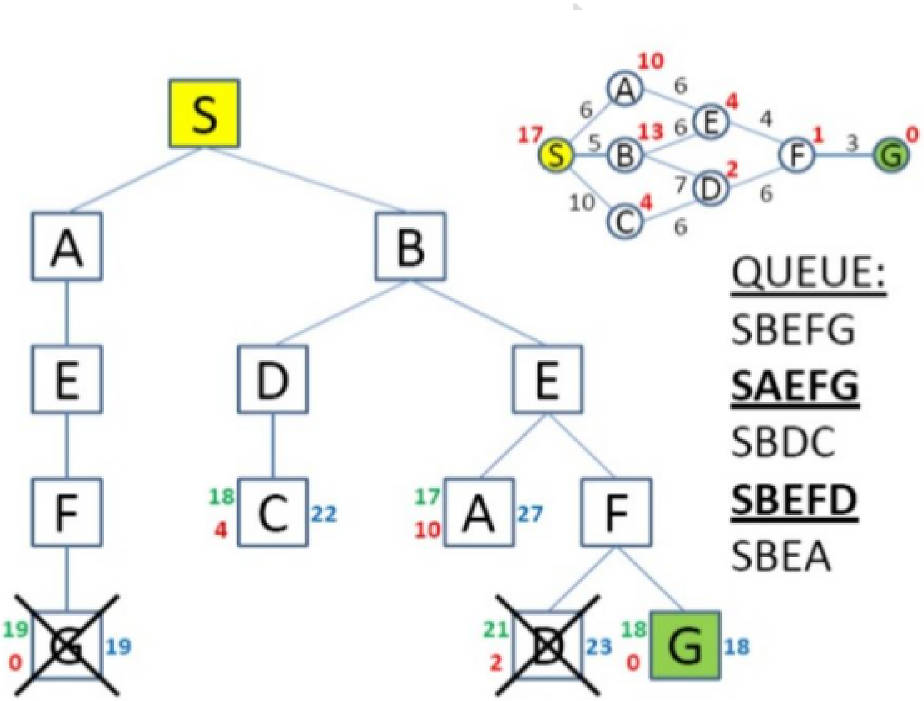
SBEF

SAEFG

SBDF

SBDC

SBEA



Given an initial state of a 8-puzzle problem and final state to be reached-

2	8	3
1	6	4
7		5

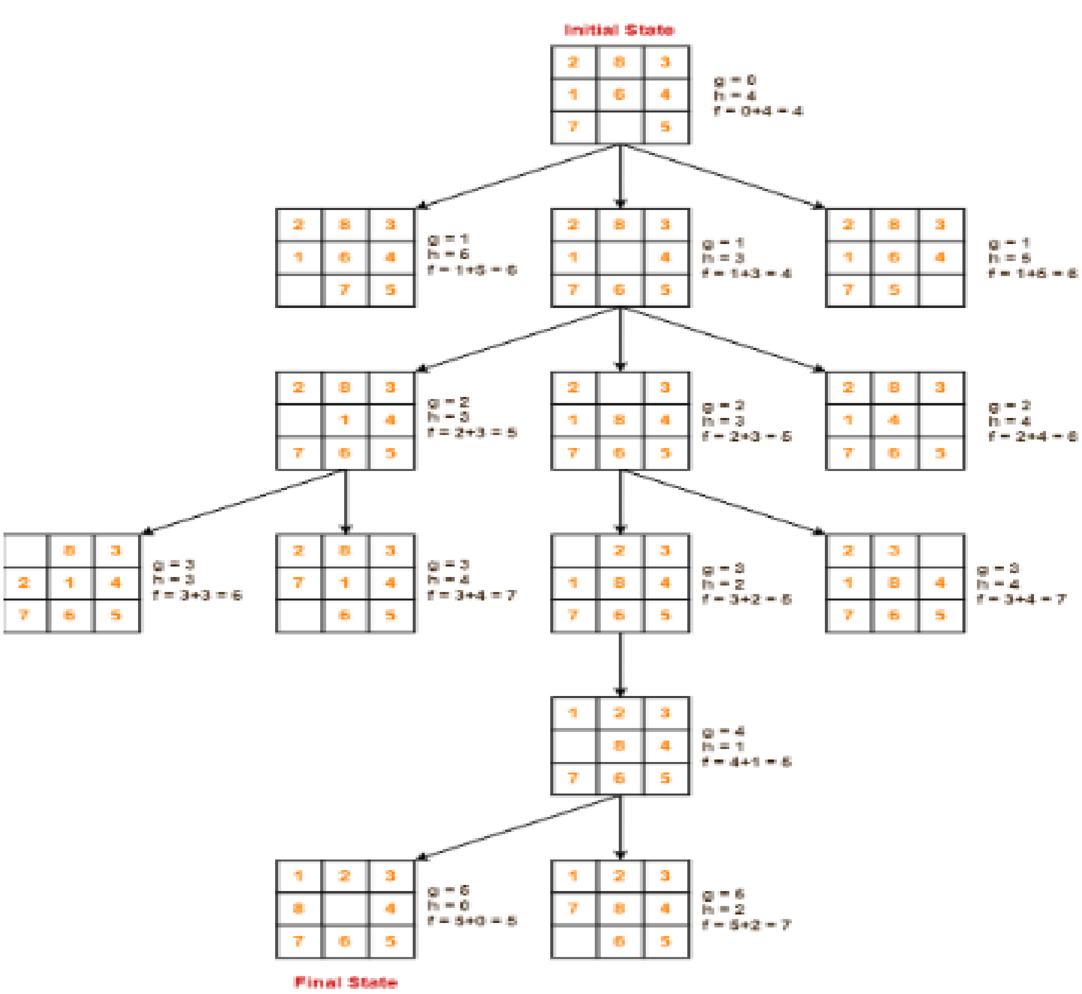
Initial State

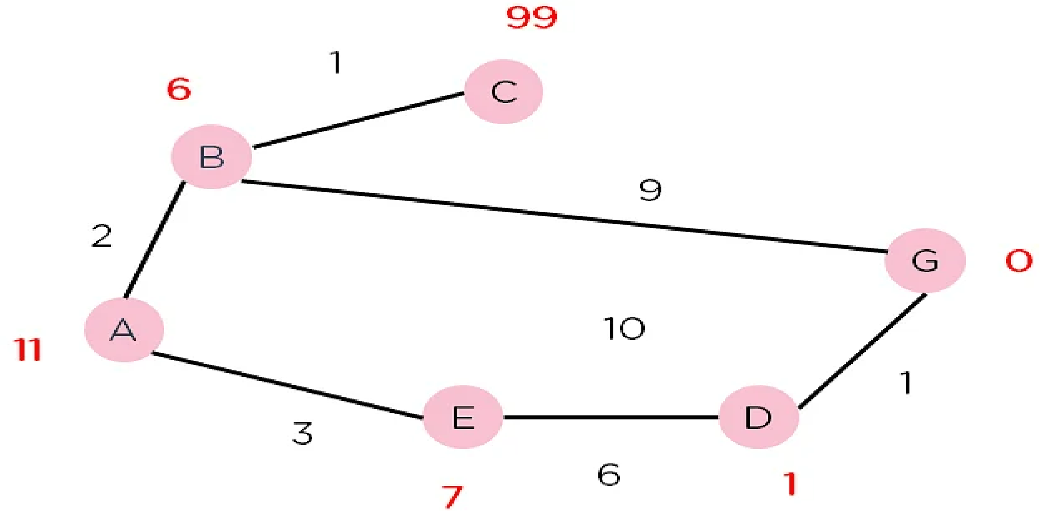
1	2	3
8		4
7	6	5

Final State

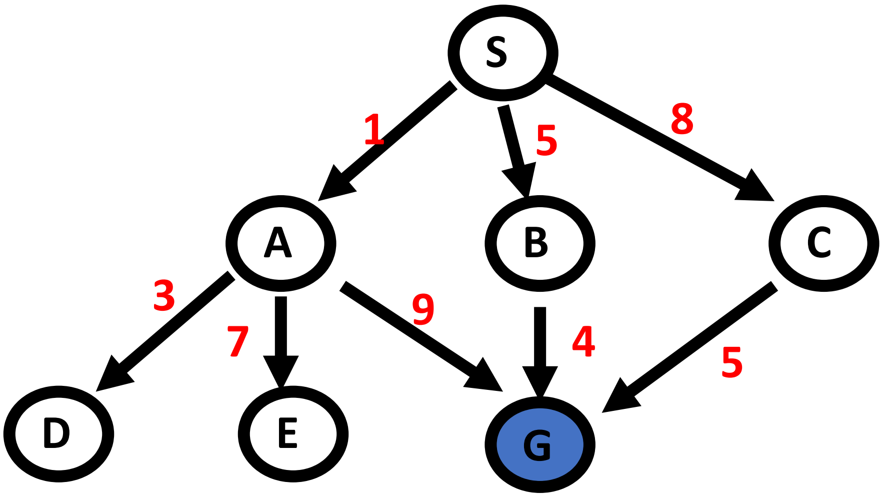
Find the most cost-effective path to reach the final state from initial state using A* Algorithm.

Consider g(n) = Depth of node and h(n) = Number of misplaced tiles.


Initial State
g = 0
h = 4
f = 0+4 = 4
g = 1
h = 5
f = 1+5 = 6
g = 1
h = 3
f = 1+3 = 4
g = 1
h = 5
f = 1+5 = 6
g = 2
h = 3
f = 2+3 = 5
g = 2
h = 3
f = 2+3 = 5
g = 2
h = 4
f = 2+4 = 6
g = 3
h = 3
f = 3+3 = 6
g = 3
h = 4
f = 3+4 = 7
g = 3
h = 2
f = 3+2 = 5
g = 3
h = 4
f = 3+4 = 7
g = 4
h = 1
f = 4+1 = 5
g = 5
h = 0
f = 5+0 = 5
g = 5
h = 2
f = 5+2 = 7
Final State


99
1
C
6
B
9
2
G
0
10
A
11
1
E
D
3
6
7
1

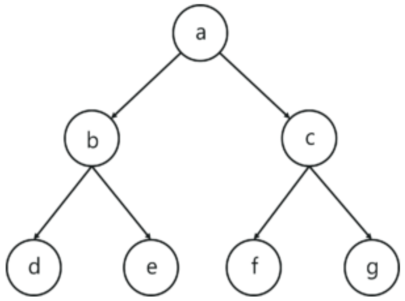
Breadth-First Search



exp. node	OPEN list	CLOSED list
	{ S }	{}
S	{ A B C }	{S}
A	{ B C D E G }	{S A}
B	{ C D E G }	{S A B}
C	{ D E G }	{S A B C}
D	{ E G}	{S A B C D}
E	{ G}	{S A B C D E}
G	{}	{S A B C D E G}
	Path	S-A-B-C-D-E-G

Number of nodes expanded (including goal node) = 7

Depth First Search

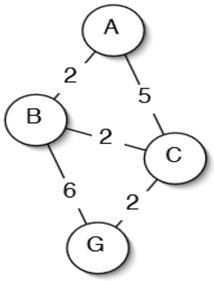


Frontier	Explored set
{A}	{}
{B,C}	{A}
{D,E,C}	{A,B}
{E,C}	{A,B,D}
{C,F,G}	{A,B,D,E}
{F,G}	{A,B,D,E,C}
{G}	{A,B,D,E,C,F}
{ }	{A,B,D,E,C,F,G}

The path explored is **A-B-D-E-C-F-G**

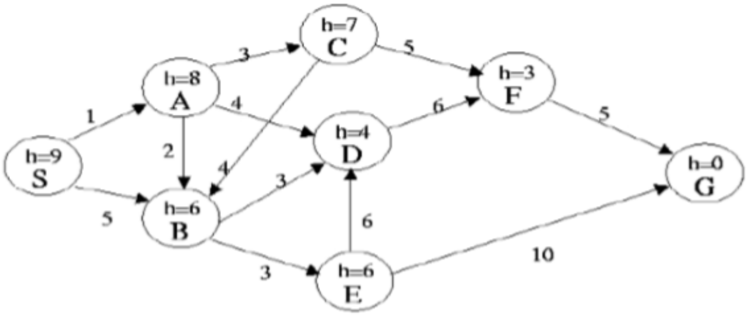
Uniform Cost Search

Simulate uniform cost search with a strict expanded list on this graph. At each step, show the state of the node that's being expanded, the length of that path, and the current value of the expanded list (as a list of states).



State Expanded	Length of Path	Expanded List
A	0	(A)
B	2	(B A)
C	4	(C B A)
G	6	(G C B A)

Q. In the graph below the start state is S, and the goal state is G. The transition costs are next to the edges, and the heuristic estimate, h, of the distance from the state to the goal is in the state's node. Assume ties are always broken by choosing the state which comes first alphabetically.



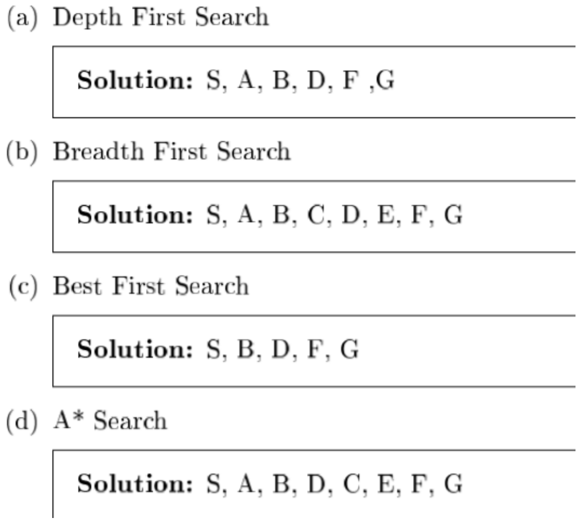
What is the order of states expanded using:

(a) Depth first search

(b) Breadth first search

(c) A* search

Solution:



(a) Depth First Search

Solution: S, A, B, D, F ,G

(b) Breadth First Search

Solution: S, A, B, C, D, E, F, G

(c) Best First Search

Solution: S, B, D, F, G

(d) A* Search

Solution: S, A, B, D, C, E, F, G

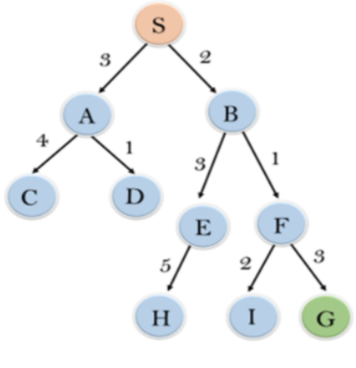
d) Is the heuristic in this example 1. admissible? Yes

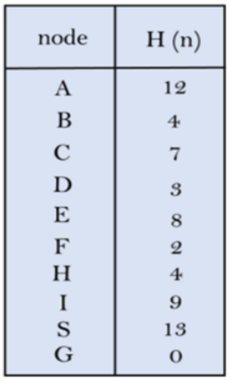
2. consistent? No Justify your answer, briefly. (3 points)

All the h values are less than or equal to actual path cost to the goal and so the heuristic is admissible.

The heuristic drops from 5 at S to 3 at B while the path cost between S and B is only 1, and so the heuristic is not consistent.

Best First Search

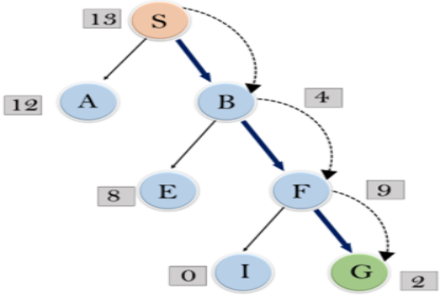


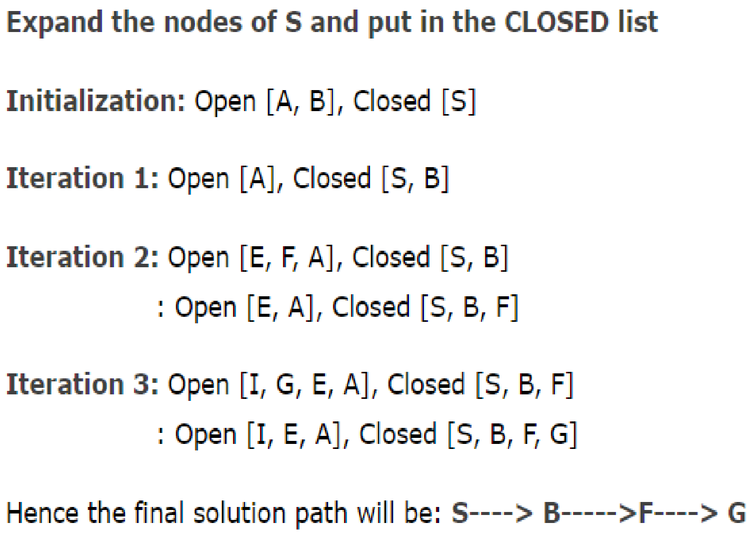


node	H (n)
A	12
B	4
C	7
D	3
E	8
F	2
H	4
I	9
S	13
G	0

Solution

- F(n)=h(n)





Expand the nodes of S and put in the CLOSED list

Initialization: Open [A, B], Closed [S]

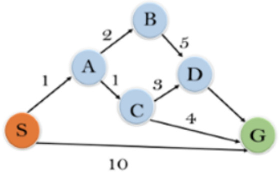
Iteration 1: Open [A], Closed [S, B]

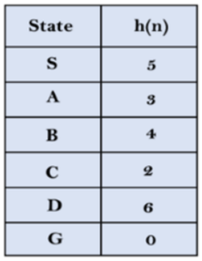
Iteration 2: Open [E, F, A], Closed [S, B]
: Open [E, A], Closed [S, B, F]

Iteration 3: Open [I, G, E, A], Closed [S, B, F]
: Open [I, E, A], Closed [S, B, F, G]

Hence the final solution path will be: **S----> B----->F----> G**

A* Search

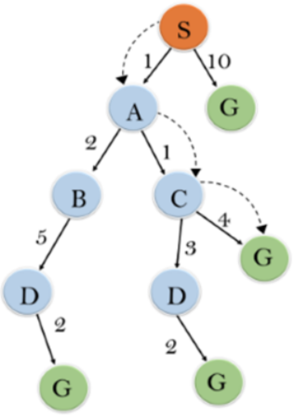


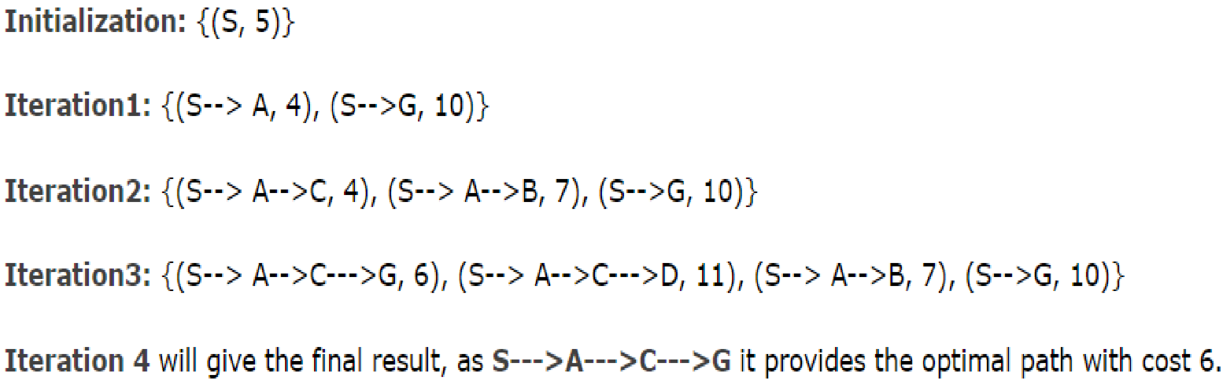


State	h(n)
S	5
A	3
B	4
C	2
D	6
G	0

Solution:

F(n)=g(n)+h(n)





Initialization: {(S, 5)}

Iteration1: {(S--> A, 4), (S-->G, 10)}

Iteration2: {(S--> A-->C, 4), (S--> A-->B, 7), (S-->G, 10)}

Iteration3: {(S--> A-->C--->G, 6), (S--> A-->C--->D, 11), (S--> A-->B, 7), (S-->G, 10)}

Iteration 4 will give the final result, as **S--->A--->C--->G** it provides the optimal path with cost 6.