Joao Pereira
19354106

Q1 a)

DFS traverses through the left subtree's first and after the right subtrees
BFS traverses through a level at a time (children) and then through the next level (grandchildren) and it goes on and on down.

- Example - In 1b BFS is displayed going down level by level whilst if it were performed by DFS it would start at the top and traverse through the left subtrees before going to the right subtrees.

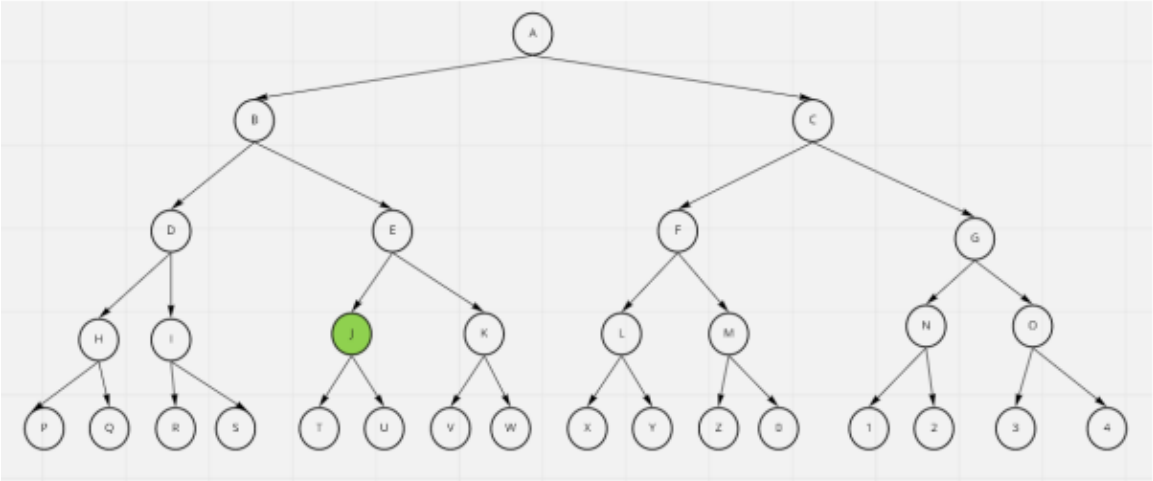
In BFS we consider every node at each level before going deeper into the graph's space.
In DFS, it gets into a deep search space very quickly.

- Example - For example in the diagram on 1b), BFS is performed and it searches through each level at a time and if it was DFS it would have to search through every node from the left to find the goal and in cases where the graph is quite large, DFS could get lost in a deep search space for a path considering the number of nodes it would need to go through.

While in DFS the children of the current node are placed at the front of the queue, in BFS The children are placed at the end.

- Example - BFS uses a queue system and DFS uses a stack system.

Q1b)



Step	Closed	Open
1		a
2	a	b,c

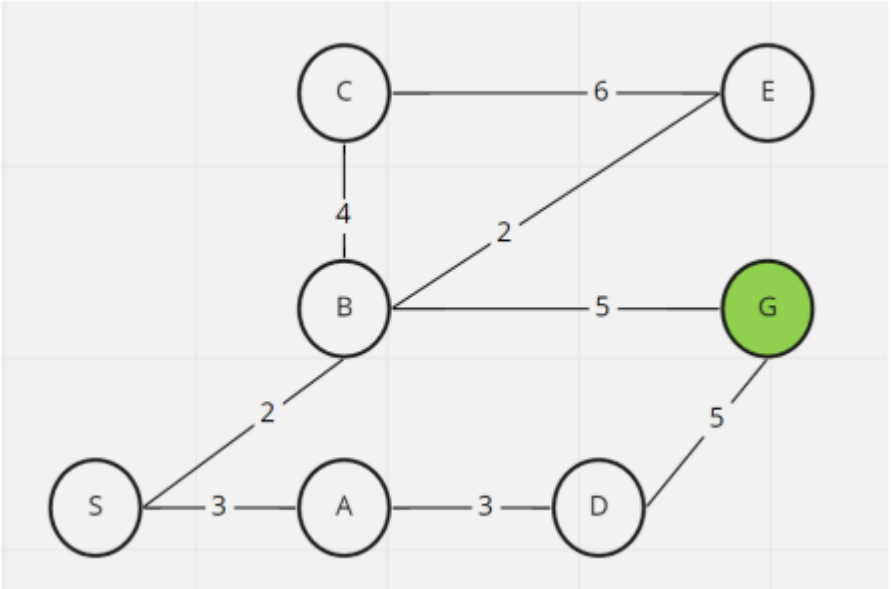
3	a,b	c,d,e
4	a,b,c	d,e,f,g
5	a,b,c,d	e,f,g,h,i
6	a,b,c,d,e	f,g,h,i,j,k
7	a,b,c,d,e,f	g,h,i,j,k,l,m
8	a,b,c,d,e,f,g	h,i,j,k,l,m,n,o
9	a,b,c,d,e,f,g,h	i,j,k,l,m,n,o,p,q
10	a,b,c,d,e,f,g,h,i	j,k,l,m,n,o,p,q,r,s
11	a,b,c,d,e,f,g,h,i,j	k,l,m,n,o,p,q,r,s,t,u
STOP	Goal J has been reached	

Q2 a)

It goes through every node before it reaches the end node. It technically does not output the shortest path which admissible heuristics in most cases would.

- Example - in 2 b) below, we have to go through each node to find the shortest path to G but if we had admissible heuristics the distance could be shorter since it would be a direct path but in some cases, it could fail.

Q2 b)

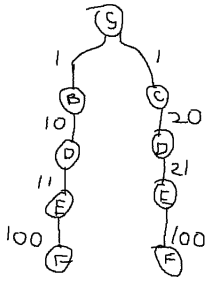


Step	Current Paths under consideration	Action
1	S-A (3), S-B (2)	Expand A
2	S-A-D (6), S-B (2)	Expand B
3	S-A-D (6), S-B-C (6), **S-B-G (7)**, S-B-E (4)	Expand D
4	S-A-D-G (11), S-B-C (6), **S-B-G (7)**, S-B-E(4)	Expand C
5	S-A-D-G (11), S-B-C-E (12), **S-B-G (7)**, S-B-E (4)	Expand E
6	S-A-D-G (11), S-B-C-E (12), **S-B-G (7)**, S-B-E-C (10)	Expand C
7	S-A-D-G (11), S-B-C-E (12), **S-B-G (7)**, S-B-E-C-B (14)	STOP
8	STOP solution is **S-B-G (7)**	

Q3 a)

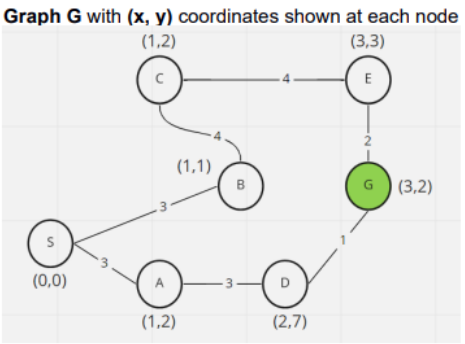
An A* algorithm may fail when using admissible heuristics if it is being operated through a non-euclidean context. This defines that we have a path however there exists a shorter path.

For example, S-C–D-E-F would be 163 but there exists a shorter path that is S-B-D-E–F which results in a path of 122



Q3 b)

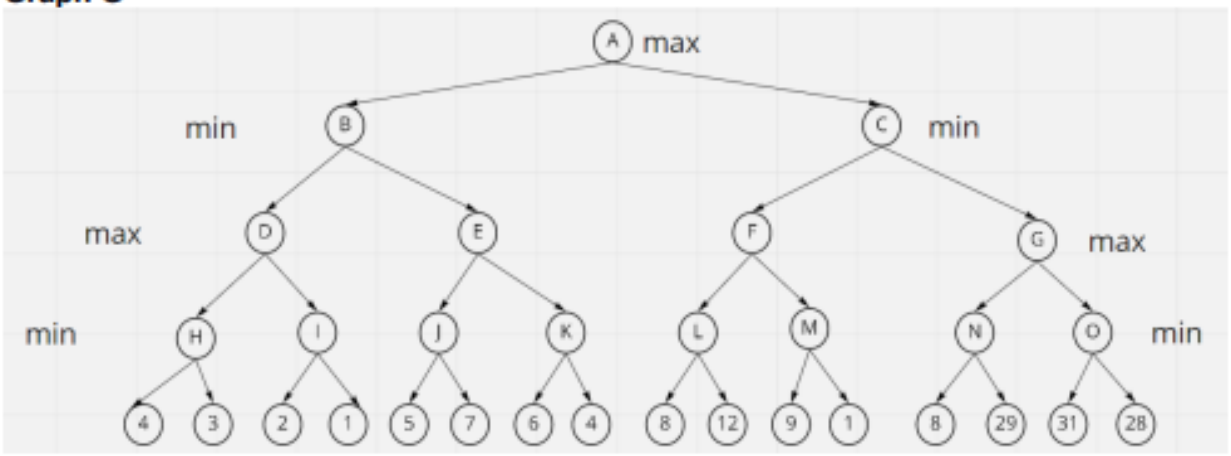
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$



Node Pair	Heuristic Distance	Admissable? Y/N
A-G	2	Yes
B-G	Root or 2.23	Yes
C-G	2	Yes
D-G	Root 26 or 5.10	N
E-G	1	Yes
S-G	Root 13 or 3.6	Y

Q4 a)

Graph G



Node	Move Generation Y/N	Static Evaluations
H	Y	4,3
I	Y	2
J	Y	5,7
K	N	-
L	Y	8,12
M	Y	9,1
N	Y	8,29
O	N	-

Path = ACFL = 8

Q4 B)

Static value = 8

Most successful move = A-C-F-L

Q5 a)

Memoisation makes the computation to the final solution quicker and memory complexity is increased. It removes unnecessary computations and stores the results of the already computed data.

An example would be the Fibonacci numbers. If Fibonacci was to be computed by memoisation the time complexity would be O(n) while if it did not use memoisation it would be of time complexity of O(2^n)

Q5 b)

Problem: Amount (A) and Coin set (C): **A = 11, C = {1, 5, 6, 9}**

Use this table to show your calculations:

	0	1	2	3	4	5	6	7	8	9	10	11
0	0	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
1												
5												
6												
9												

	o	1	2	3	4	5	6	7	8	9	10	11
o	o	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf
1	o	1	2	3	4	5	6	7	8	9	10	11
5	o	1	2	3	4	1	2	3	4	5	2	3
6	o	1	2	3	4	1	1	2	3	4	2	2
9	o	1	2	3	4	1	1	2	3	1	2	2