Test 3

Parssa Kyanzadeh, kyanzad2, 1006163508
Thursday April 8, 2021

Question 1

a) Since we know that A is a sorted array, we use binary search to find element x in A, if it's not found, we return -1, otherwise, we check the value in array B matching the index of which x was found in A, and if that value is 1 we return the index, else we return -1. By definition of binary search, we know that calling Search on our ADT will have a tight upper bound of $O(log(n))$.

b) We will say that:

- Calling Search on an element costs: $log(n)$, but for the sake of simplicity we will leave this as 1
- Removing an element from array A costs: 1
- Changing an element in array B costs: 1

Thus the "cheap" cost for calling Delete(S,x), costs 3.

The expensive cost for calling Delete(S, x), would consist of 3, for deleting the element x from S, but would also cost an additional $3n$, where n is the remaining elements in the set S, since all of those elements would need to be removed from the array A, which costs n, then put into the new array A of size n, which costs n, and then would also require populating a new array B of size n, where all the values are 1. This results in an expensice cost for Delete(S, x) of $3 + 3n$.

Let us charge 6 units per operation.

Take S = {1, 2, 3, 4}. A = [1, 2, 3, 4], B = [1, 1, 1, 1]

Calling Delete on an arbitrary element in S, where the resulting set's length n ≠ 2^{k-1}, would result in a surplus of 3.

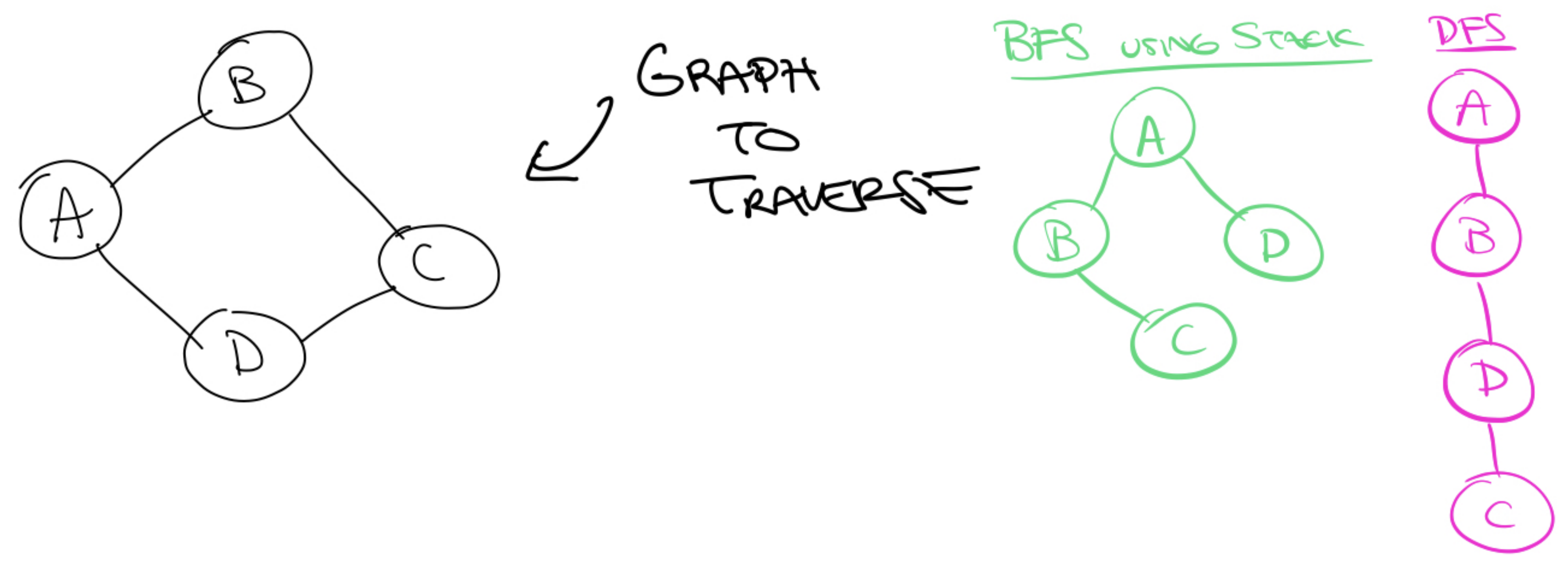
This would continue until we call Delete on S = {1, 2, 3}, where it would cost an additional 3n = 6. However, from the Delete calls that came prior, there was a surplus of exactly 6, so the total remaining is 0.

So given a total of n operations, we have a worst case running time of $O(log(n) + 2 + 3n) = O(3n)$, and so we get an amortized cost of $\frac{3n}{n} = 3$.

Question 2

a) This claim is correct, because we know that in this directed graph, there is an explicit path from u to v in the graph, and we also know that d[u] < d[v]. This means that we can apply the intuition given by the Parenthesis Theorem, to conclude that v must be a descentant of u in the depth-first forest.

b) This claim is incorrect, as replacing a stack with a queue in the breadth-first search algorithm does not entirely replicate the resulting search tree that would result from depth-first search. This is because although the breadth-first search would now be operating FILO instead of FIFO, it would still be popping from the stack immediately when an element is put into the stack, and branch out each step, as normal. A counter example to show this is as follows:



Question 3

Take G to be an arbitrary weight graph with distinctly weighted edges. Assume A and B, to be minimum spanning trees of the graph G, and take a is the cheapest edge of A.

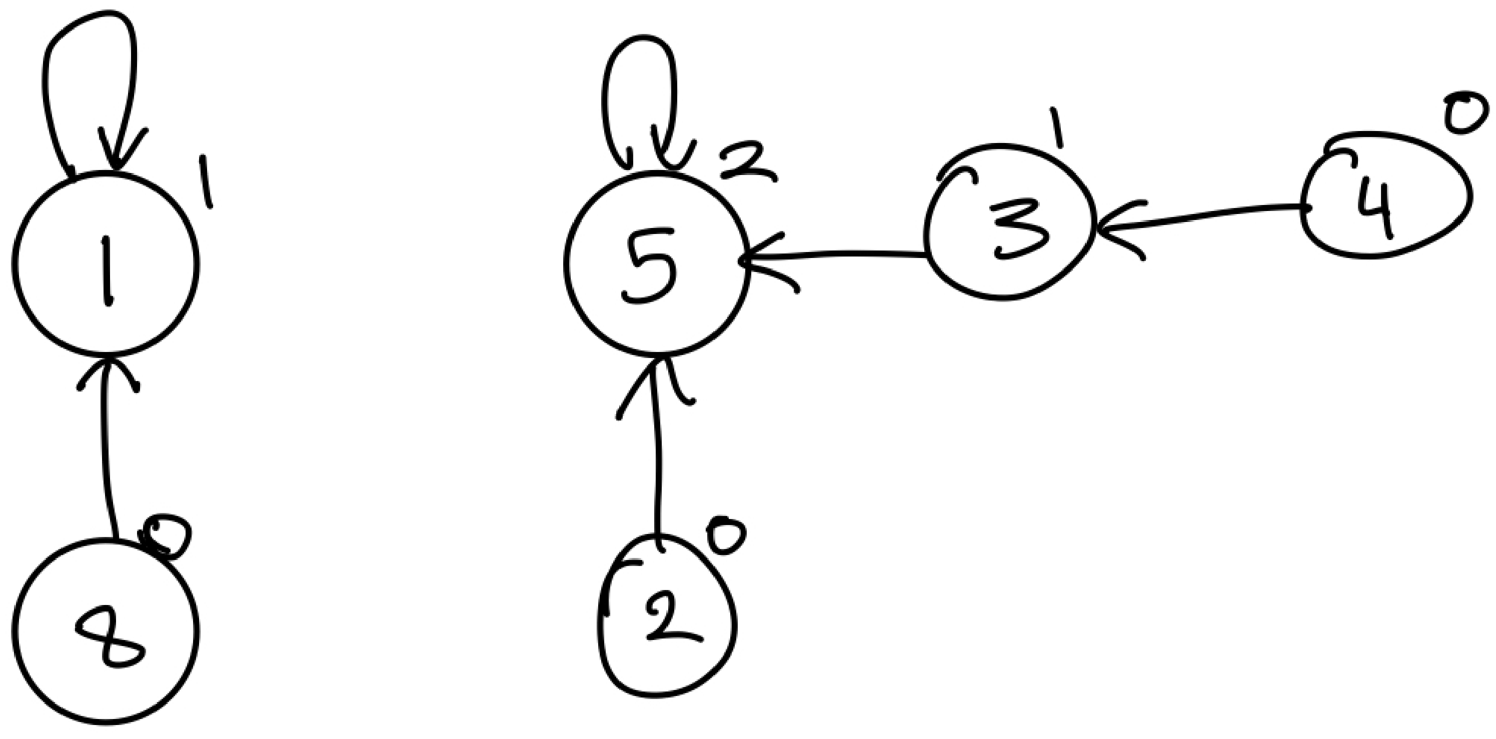
Assume that B contains more than one edge and contains nodes x and y, that does not consist of edge a.

Assume edge a is the edge from nodes x and y, this means that incorportating edge a in B would result in B becoming cyclic.

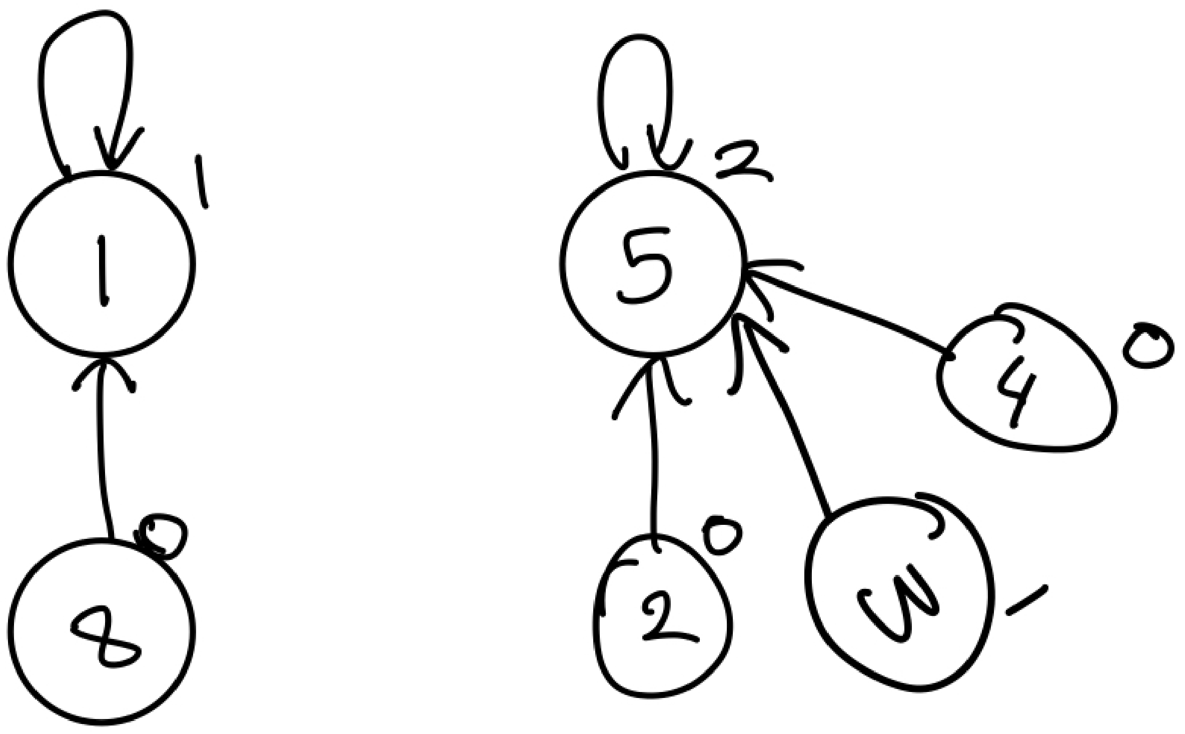
By definition, all edges of G have distinct weights, thus we know that G must contain of at least one MST.

Question 4

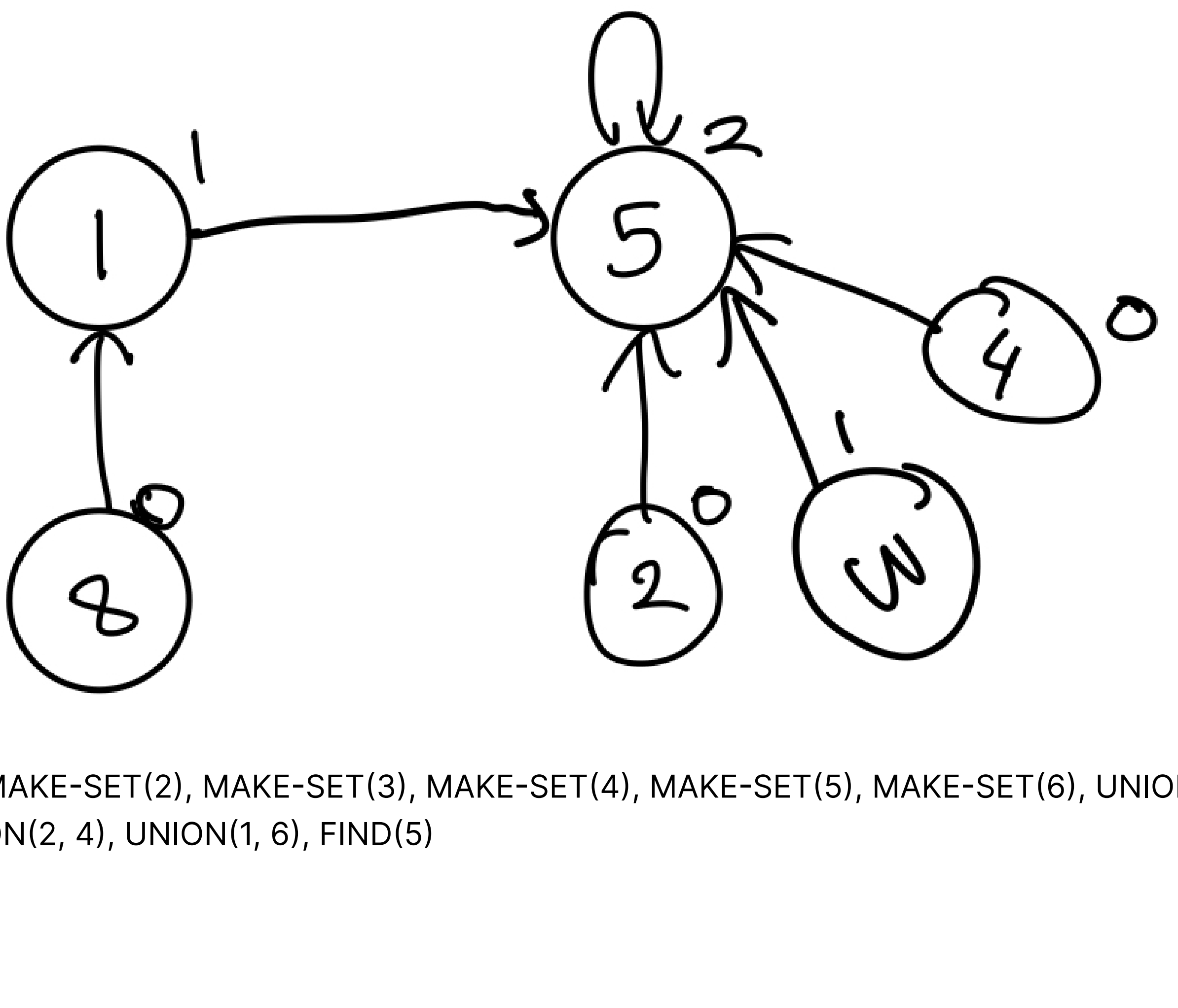
ii)



iii)



iv)



v) MAKE-SET(1), MAKE-SET(2), MAKE-SET(3), MAKE-SET(4), MAKE-SET(5), MAKE-SET(6), UNION(1, 2), UNION(3, 4), UNION(5, 6), UNION(2, 4), UNION(1, 6), FIND(5)