COMP 251 Assignment 5
Richard Ni
260674646

1.

Using Aggregate Analysis to determine the amortized cost per operation:

We have the sequence of n operations to have floor(log(n)) + 1 ops, where i is sequence for the power of 2, the sequence being (1, 2, 4, 8, ..., 2^(floor(log(n))))

Find the total cost:

= Summation from i = 0 -> log(n) of 2^i
= 2^(floor(log(n) + 2)) - 1
<= 2^(log(n) + 1)
= 2n

The rest of the operations cost 1, and because there are less than n of these operations, we have:

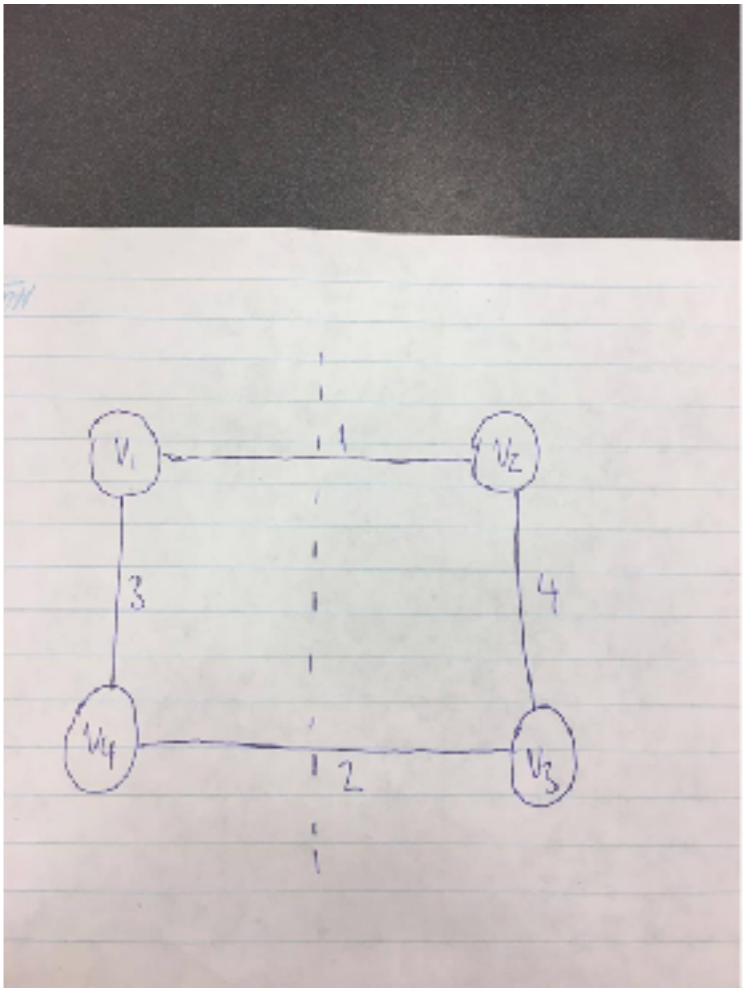
T(n) <= 2n + n
= 3n
= O(n)

Finally...

3n / n = 3
Therefore, the amortized cost per operation is **O(1)**

2.

The illustration below is an example where the algorithm fails to produce MST



G1 = (V1, E1), where V1 = {v1, v4}
G2 = (V2, E2), where V2 = {v2, v3}

MST for G1 is the weight of E1 = 3
MST for G2 is the weight of E2 = 4

The light edge crossing v1 and v2 has weight 1
The light edge crossing v3 and v4 has weight 2

Using Toole's algorithm:

MST(v4 <-> v1 <-> v2 <-> v3) = 3 + 1 + 4 = **8**

This algorithm does not produce the true MST, where the true MST is:

MST(v2 <-> v1 <-> v4 <-> v3) = 1 + 3 + 2 = **6**

Therefore, this algorithm fails to produce the MST of G

3.

Worst Case:

When paritioning in the worst case, we always produce arrays 0 -> n - 1

T(n) = T(n - 1) + 1
= **Theta(n)**

Best Case:

When partitioning in the best case, we produce arrays of equal sizes, where the max size difference is 1

T(n) = 2 T(n / 2) + 1

Using property 1 of the master theorem: T(n) <= a T(n / b) + O(n^d)

a = 2
b = 2
d = 0

where a > b^d

T(n) = Theta(n^(log_2(2)))
= **Theta(n)**