Assignment: 02 Name: Methew Guda
Due Date: 15-10-2015 A#: A00381751

Q1 - (a):

If I was interested in improving the efficiency of this software by investigating the algorithms employed within the various modules which make up the software, I would focus on those modules which takes more times to run. From self-seconds I find first four modules took (0.26 + 0.16 + 0.13 + 0.9) = 1.45 seconds and the whole program took 1.72 seconds in my virtual machine. If I could improve the four most time consuming modules by 50%, the whole program will take 0.995 seconds to run.

Q2 - (a):

[13, 1, 12, 3, 9, 5, 2, 11, 10, 8, 6, 4, 7, 0]

[13, 12, 9, 11, 10, 8, 6, 7]

[13, 11, 10, 7]

[13, 10]

[13]

The root of this tree gives us the largest element in the list which is 13.

Q2 - (b):

Since the second largest element in the tree must have lost to the largest element of the tree at some point, so let's trace back to the tree, from the root to the leaf level by replacing all instances of the largest element with $-\infty$.

[$-\infty$, 1, 12, 3, 9, 5, 2, 11, 10, 8, 6, 4, 7, 0]

[$-\infty$, 12, 9, 11, 10, 8, 6, 7]

[$-\infty$, 11, 10, 7]

[$-\infty$, 10]

[$-\infty$]

Now starting at the leaf level, at the node that just received the value −∞, work back towards the root level, repairing the tournament tree as I go.

[−∞, 1, 12, 3, 9, 5, 2, 11, 10, 8, 6, 4, 7, 0]

[1, 12, 9, 11, 10, 8, 6, 7]

[12, 11, 10, 7]

[12, 10]

[12]

So using this approach we get the second largest element of the tree which is 12.

Q2 - (c):

Since the third largest element in the tree must have lost to the second largest element of the tree at some point, so let's trace back to the tree, from the root to the leaf level by replacing all instances of the second largest element with −∞.

[1, −∞, 3, 9, 5, 2, 11, 10, 8, 6, 4, 7, 0]

[1, −∞, 9, 11, 10, 8, 6, 7]

[−∞, 11, 10, 7]

[−∞, 10]

[−∞]

Now starting at the leaf level, at the node that just received the value −∞, work back towards the root level, repairing the tournament tree as I go.

[1, −∞, 3, 9, 5, 2, 11, 10, 8, 6, 4, 7, 0]

[1, 9, 11, 10, 8, 6, 7]

[9, 11, 10, 7]

[11, 10]

[11]

So using this approach we get the third largest element of the tree which is 11.

Q2 - (d):

Using standard/sequential search method:

On average case, the cost of finding the largest element in the tree is: $(N+1)/2$. So now the cost to find five largest elements in the tree is: $5(N+1)/2$, where N is 10 million. $5(10 \text{ million} + 1)/2 = 25,000,002.5$ microseconds ≈ 25 seconds.

Using tournament tree method:

The cost to find the five largest element in the tree is:

$(N-1) + 2 \log_2 N + 2 \log_2 N + 2 \log_2 N + 2 \log_2 N$, where N is 10 million.

$(N-1) + 8 \log_2 N = (10 \text{ million} - 1) + 8 \log_2(10 \text{ million}) = 10,000, 185.027976$ microsecond ≈ 10 seconds to complete.