Danh Nguyen
65872969 (danhn2)

Rodney McGrath
55352053 (rjmcgrat)

Assignment 1: Selection of *k* Largest

Selection of k largest values from an array of n values. Program is compiled by typing gcc -o MAIN MAIN.c and ran by typing ./MAIN

Data Structure and Implementation:

- The data structure that was used in a tournament tree. The tournament tree acts a lot like a max heap where the highest element would be the root element. Also, when removing the root element of the tournament tree, we compute the next highest element by running the “generate tournament tree algorithm” again but only on direct children of the root node of the current tournament tree.
- So in order to get the k largest, we just have to take of root nodes of the tournament trees generated k times.
-

Valgrind memory check:

```
rjmcgrat@andromeda-20 22:49:54 ~
$ cd cs165/project1
rjmcgrat@andromeda-20 22:50:05 ~/cs165/project1
$ gcc -o MAIN MAIN.c
rjmcgrat@andromeda-20 22:50:16 ~/cs165/project1
$ valgrind ./MAIN
==32453== Memcheck, a memory error detector
==32453== Copyright (C) 2002-2015, and GNU GPL'd, by Julian Seward et al.
==32453== Using Valgrind-3.12.0 and LibVEX; rerun with -h for copyright info
==32453== Command: ./MAIN
==32453==
n=  100,  k=10:  maximum= 176,  avg=  164.80
n=10000,  k=40:  maximum= 10643,  avg=10608.09
==32453==
==32453== HEAP SUMMARY:
==32453==     in use at exit: 0 bytes in 0 blocks
==32453==   total heap usage: 14,401,773 allocs, 14,401,773 frees, 100,018,608 bytes allocated
==32453==
==32453== All heap blocks were freed -- no leaks are possible
==32453==
==32453== For counts of detected and suppressed errors, rerun with: -v
==32453== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
rjmcgrat@andromeda-20 22:51:27 ~/cs165/project1
$
```

Result of functions:

```
n=  100,  k=10:  maximum= 176,  avg=  164.80
n=10000,  k=40:  maximum= 10643,  avg=10608.09
dhcp-v074-128:code rodneymcgrath$
```

Results with varying n and k:

```
n=  100,  k=40:  maximum= 390,  avg=  358.87
n= 1000,  k=40:  maximum= 1476,  avg= 1443.63
n= 2000,  k=40:  maximum= 2535,  avg= 2497.46
n= 3000,  k=40:  maximum= 3559,  avg= 3525.85
n= 4000,  k=40:  maximum= 4583,  avg= 4545.62
n= 5000,  k=40:  maximum= 5598,  avg= 5561.40
n= 6000,  k=40:  maximum= 6611,  avg= 6573.48
n= 7000,  k=40:  maximum= 7622,  avg= 7583.73
n= 8000,  k=40:  maximum= 8632,  avg= 8592.09
n= 9000,  k=40:  maximum= 9636,  avg= 9601.71
n=10000,  k=40:  maximum= 10640,  avg=10608.35
```

```
n=10000,  k=1:  maximum= 10012,  avg=10011.65
n=10000,  k=20:  maximum= 10320,  avg=10295.31
n=10000,  k=25:  maximum= 10398,  avg=10372.46
n=10000,  k=30:  maximum= 10481,  avg=10450.25
n=10000,  k=50:  maximum= 10810,  avg=10768.73
n=10000,  k=60:  maximum= 10980,  avg=10928.81
n=10000,  k=70:  maximum= 11141,  avg=11093.47
n=10000,  k=80:  maximum= 11319,  avg=11257.59
n=10000,  k=90:  maximum= 11481,  avg=11422.92
n=10000,  k=99:  maximum= 11635,  avg=11571.37
n=10000,  k=100:  maximum= 11654,  avg=11587.31
```

```
n=  100,  k=5:  maximum= 136,  avg=  131.05
n= 1000,  k=15:  maximum= 1174,  avg= 1158.04
n= 2000,  k=25:  maximum= 2329,  avg= 2302.69
n= 3000,  k=35:  maximum= 3490,  avg= 3456.24
n= 4000,  k=45:  maximum= 4659,  avg= 4617.95
n= 5000,  k=55:  maximum= 5832,  avg= 5785.53
n= 6000,  k=65:  maximum= 7009,  avg= 6956.30
n= 7000,  k=75:  maximum= 8189,  avg= 8130.23
n= 8000,  k=85:  maximum= 9363,  avg= 9307.91
n= 9000,  k=95:  maximum= 10552,  avg=10488.80
n=10000,  k=100:  maximum= 11651,  avg=11588.23
dhcp-v074-128:code rodneymcgrath$
```

Empirical complexity analysis:

Observed worse case: n + 1.22 * klogn

Based on the result, it is cleared to see that the worse case is just a little more than n because when we increase n, the worse case increases by that factor and a little more. Next, we realized that since we are getting k largest and each time we get the next biggest value we have to do roughly log(n) comparison. Putting both of these pieces together give us an observed worse case of n + klogn. There is a constant of 1.22 to multiply the klogn because the data for the observed worse case is slightly higher than n + klogn.

Average case: **n + 1.14 * klogn**

It seems as though our average case is extremely close to the observed worse case but it seems to be slightly lower than the worse case. As a result, lowering the constant slightly seems to give relatively accurate results for the average case results.

Theoretical complexity analysis:

Theoretical Worst Case: n + klogn

In order to build the very first tournament tree in my algorithm, there must be n - 1 comparisons that takes place which means that n - 1 is an essential part in the worse case. Then, in order to calculate the next highest node in the tournament tree, we would just do comparisons on the children nodes of the current highest node in the tournament tree. The worst case possible comparison for this process would be logn and we have to do it k times. The reason why it would be logn comparisons is due to the fact that the children of a node in a tournament tree is not that much in relation to n. As a result, the number of comparisons in order to find the next highest node amongs the children should be extremely low and slow growing in relation to n (logn fits such properties). In order to find the k largest, the function of logn will have to be apply k times hence making klogn comparisons a requirement.

Theoretical Expected Worst Case:

n + klogn which is 10644 comparisons for n = 10000 and k = 40.

n = 10000 rather than 100 is necessary for the worse case to be reached and k = 40 is the standard k that will be used in this particular case. Applying the function for the worse case for this particular case would be the theoretical expected worse case.

Theoretical Average: n + klogn

The theoretical average is the same as the theoretical worst case is due to the fact that the way the element is order overall should not affect the number of comparisons during the creation of the tournament tree.

The observed worse case and average is the same as the theoretical counterpart. The only difference is that the observed worse case and average is slightly higher due to a higher coefficient in multiplication. Also, the reason why there are higher number comparisons for the observed might be due to the fact that over the course of 1000 runs, the order of the tree affects slightly the path to get the k largest slightly which results in more comparisons.