Analysis:

▲theoretical WC: (n-1) + (k-1)(⌈logn⌉ - 1)

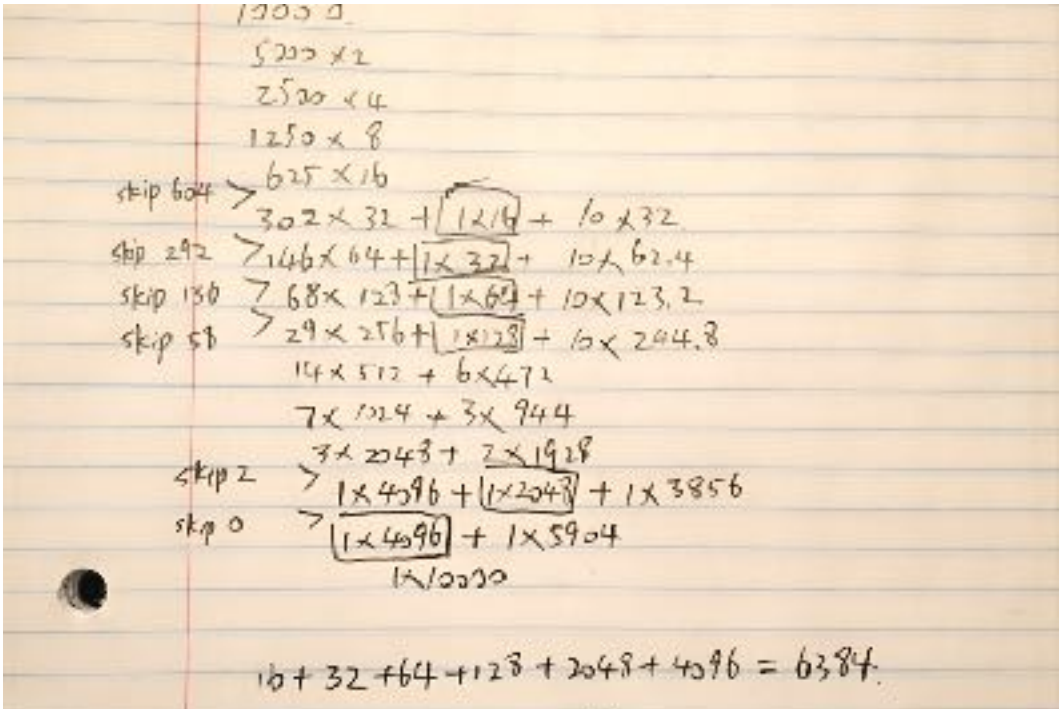
▲ theoretical expected WC for given number of runs : 10506 (comps)

for n = 10000, k = 40, WC = 9999 + 39 * 13 = 10506 comparisons

for n = 1000, k = 10, WC = 999 + 9 * 9 = 180 comparisons

We do comparisons on adjacent elements recursively and store the bigger element. For N = 10000, we have 9999 to get the largest element. Elements are stored in arrays and multiple arrays contribute to a big tree. Then we remove the biggest element and make a new championship on elements that lost to the largest, to get the next biggest and this operation requires ⌈log 10000⌉ - 1 = 13 comparisons.

▲ theoretical AVG : (16+32+64+128+2048+4096 = 6384 = 63.84%) 63.84 % of time we can have one less comparisons. 63.84%*39=24.8976 comparisons can be skipped resulting in 10506 - 24.8976= 10481.1024. See below image:



theoretical AVG: 10506 - 39 * 63.84% * 1

- observed WC and AVG for given number of runs:

  n= 100, k=10: WC = 153, AVG = 150.12

  n=10000, k=40: WC = 10491, AVG = 10481.06

```
yifanj6@woodhouse 14:37:19 ~/CS165/project1
$ ./main
n=  100,  k=10:  maximum= 153,  avg=  150.12
n=10000,  k=40:  maximum= 10491,  avg=10481.06
```

RESULT FOR RUNNING MAIN.C

Theoretical AVG is different from AVG observed because the actual distribution of elements varies, but on the large scale, it is very close to our expectation.

Our tree looks like this: