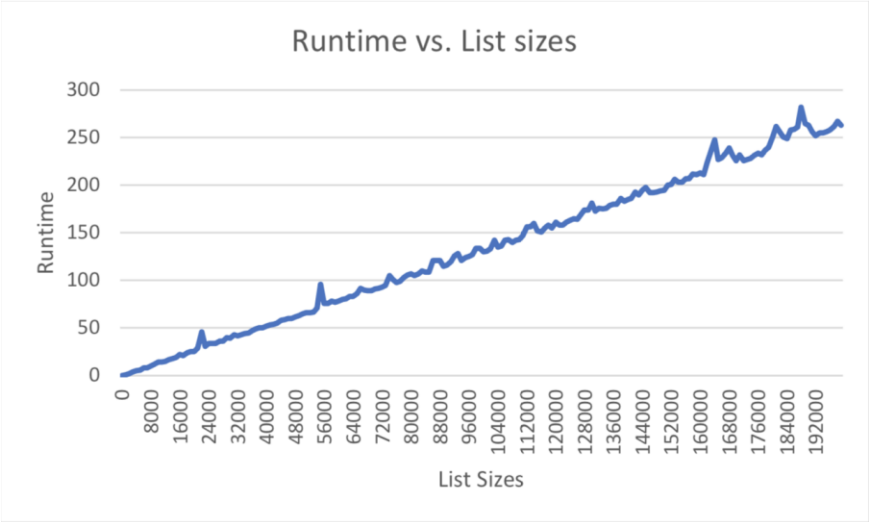
Yen-Chu, Yu
CSE 373
Shrirang Mare

HW5 Part.1e: Individual writeup

Experiment1:

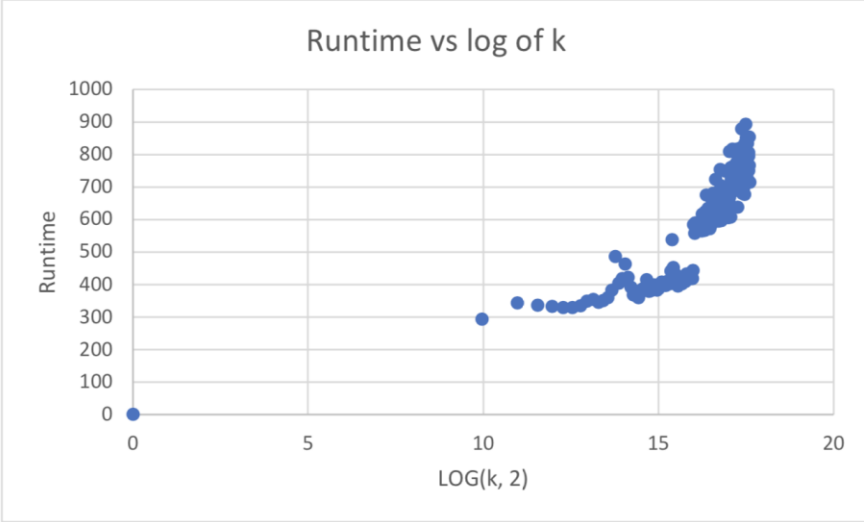
In experiment 1, it's testing the runtime of KTopSort with different input list size, and I expect it growing at a linear rate proportional to n.



As the result, the graph does show a linear relationship with list size n, and this is because the method has runtime O(n*log(k)) and the k has been fixed at 500, thus, the runtime become O(n).

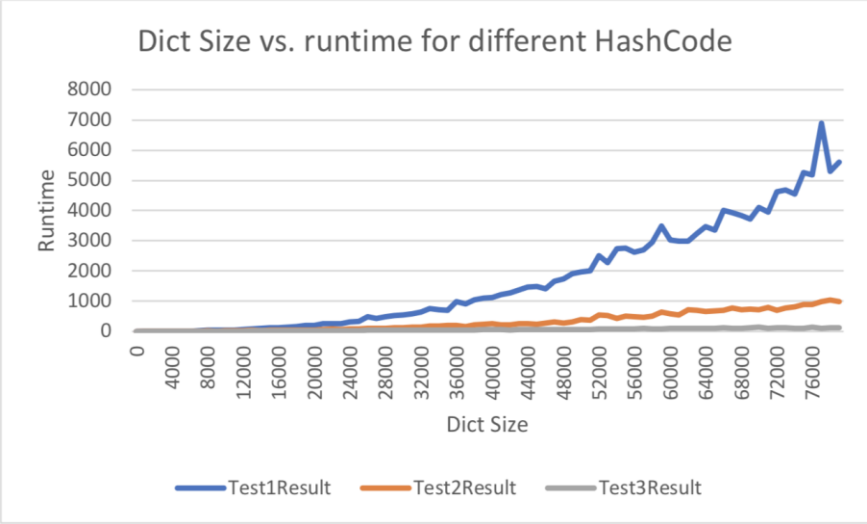
Experiment 2:

In experiment 2, it's testing the runtime of KTopSort with different input k with list size fixed at 20000, and I expect it growing in a log scale.



As the result, the graph does show a growth in log scale though it's not very clear inn the graph, the growth rate should be proportional to log2(k). This is because the method has runtime O(n*log(k)) and the n has been fixed, thus, the runtime become O(log(k)).

Experiment 3:
In Experiment 3, it's comparing the runtime of put method of ChainedHashDictionary using different functions for hash code. The data structure to put in the dictionary is a kind of string with a char array internally, and test 1, 2, 3 alters its hash code function differently. Test 1 hash code is the sum of the first 4 element in the char array. Test 2 hash code is the sum of all elements in the char array. Test 3 hash code function is similar to test 2, but instead, each time summing, it first multiply itself with 31 then add the next element. I expect that the speed of put is test 1 > test 2 > test 3 according to the uniqueness of hash code function.



As I expected, the put speed for test 1 is the slowest, test 2 is the next, and test 3 is the fastest. This is because of the uniqueness of the hash code function. Also, a hash code function that lacks uniqueness will cause clustering and slow down the speed, and good hash code functions would give fewer collisions to reduce the operation time.