CSCI181AF: Advanced Data Structures
09/21/2022

Writeup HW #3

Introduction

We will be evaluating the performance of four different algorithms written in C++ to return an array of the k largest elements of a given C++ vector of length n by comparing the running times (ms) of each. The first algorithm sorts the array then returns the first k values. The second algorithm partially sorts the vector to find the $(n-k)^{th}$ largest number, then outputs the top k numbers. The third algorithm converts the vector to a maxheap, then pops off the top k values to a new vector to return. The fourth algorithm uses a minheap of size k. If a value is larger than the top of the minheap, then it is added, then the minheap is returned as a vector. These different algorithms were tested using values of $n = 10000000$, $k = 32, 5000000$, with each experiment being completed 30 times (each algorithm runs 30 times) across two different situations, one where the values were strictly increasing and another where the order has been shuffled.

Data

Strategy	Avg.	STDev.	95% CI
1	759	27	±10
2	76.5	20	±7
3	152	11	±4
4	16.6	0.8	±0.3

Table 1: $n = 1e7, k = 32$, Shuffled

Strategy	Avg.	STDev.	95% CI
1	747	28	±10
2	105	14	±5
3	3800	190	±70
4	4540	400	±140

Table 2: $n = 1e7, k = 5e6$, Shuffled

Strategy	Avg.	STDev.	95% CI
1	50	13	±5
2	31	4	±1
3	330	18	±6
4	140	9	±3

Table 3: $n = 1e7, k = 32$, Unshuffled

Strategy	Avg.	STDev.	95% CI
1	62.3	13	±4
2	35.2	5	±2
3	704	45	±16
4	1033	87	±30

Table 4: $n = 1e7, k = 5e6$, Unshuffled

Strategy	Avg.	STDev.	95% CI
1	394	340	±120
2	61.6	33	±12
3	1247	1496	±540
4	1431	1852	±660

Table 5: Averaged Values over All 4 Experiments

Analysis

Looking at Table 5, we can see that overall, our strategy of partially sorting the vector had on average, the best time with a confidence interval that did not overlap with any other confidence intervals. However, on a case by case basis, we can see that Table 1 indicates that for a small k, shuffled array, the min heap works best. However, in unshuffled vectors, as seen in Tables 3 and 4, the partial sort strategy works best. Thus, in a general case, we can conclude that the best algorithm for the k of n problem is to partially sort the vector to find the $(n-k)^{th}$ largest number and output the top k numbers, as it also has the least variance across each experiment.