Raed Abuzaid

CSCI 335

Project 3

Time Results

	test_input	test_input2	test_input3
StdSort	0.00004088 s	0.00387578 s	0.364537 s
QuickSelect1	0.000208333 s	0.23937 s	Gave up after 30 min
QuickSelect2	0.000149779 s	0.00640623 s	0.986913 s
CountingSort	0.00113984 s	0.0127098 s	1.03979 s

Time Complexities

StdSort:

- std::sort has complexity of **O(n*logn)**.

QuickSelect1:

- quickSelect has an average case complexity of **O(n)** and a worse case of **O(n^2)**
- quickSelect here is called multiple times but ultimately the worst case is **O(n^2)**

QuickSelect2:

- Has the same average case complexity of **O(n)** and a worse case of **O(n^2)**
- Using the keys method we can avoid unnecessary recursive calls

CountingSort:

- The iteration are all O(n) but the sorting part of the function is in the worse case **O(n*logn)** since we use std::sort. We ultimately sort fewer values since we only take into consideration one of each unique value, so while **O(n*logn)** just like stdSort it is still faster in general

Discussion

Given that both counting sort and std sort used the std::sort algorithm, they had the same time complexity **O(n*logn).** Std sort had to sort a larger set because it had to account for every single copy whether duplicate or not. Although counting sort used std::sort on a smaller set since it removed the duplicate copies, it still had to iterate through the whole set multiple times first to put the vector into a map for each value, then again from the map of unique values to another vector to be sorted. Even though it allowed us to use std::sort on a smaller vector, I feel that the previous iterations were too costly to notice any benefit, this is reflected in the poor results compared to stdSort in the time table. Quick select 1 performed badly as the data set grew, to the extent that I had to crash the program after 30 minutes of my computer begging to be saved for test input 3. I feel the reason for this was the frequent recursive calls. Ultimately quick select 2 managed to save time on unnecessary recursive calls by checking if a subrange contains a key, this method even allowed it to perform better than counting sort.