Mark Shinozaki
Cpts 350
Homework #3

1. write an algorithm that selects both the maximal element and the minimal element from an array A of n elements, using only 1.5·n comparisons.

breakdown

Find maximum and minimum with 1.5n comparisons

An Array A of n elements

The maximum and minimum elements in A

if
if n is odd, set min = max = A[0], iterate from second element with i = 1

if
If n is even, compare A[0] and A[1] to set min and max, iterate from the third element with i = 2.

For
for each pair (A[i], A[i+1]], i runs from the current start index to n-1 in:

- compare A[i] with A[i+1]

if
if A[i] > A[i+1] then,

Max = max (max, A[i])
Min = min (min, A[i+1])

else

Max = Max (max, A[i+1]
Min = min (min, A[i])

Return min and max

* This means it can be ensured that min and max elements are found with no more than 1.5 comparisons.

2. Compare the average-case complexities of the two algorithms; i.e. For the average-case complexities, under what conditions (on the choices for i), S is better than T or vice versa.

- Algorithm S: using linear Select
  - for each of the j smallest elements where $j \leq i$. linear Select has an avg complexity of $O(n)$
  - Avg-case, linear Select for each j up to i, the total avg case complexity for S is $O(in)$.

- Algorithm T: using Mergesort and Select
  - T sorts the Array A using Merge sort, which has avg case and worst case being $O(n \log n)$
  - Sorting is $O(n \log n)$ and the selection of the first i elements is $O(i)$. Total avg case complexity for T remains $O(n \log n)$.

- S is more efficient than T, when youre selecting a relatively small subset of the elements ($i < \log n$) which costs of repeatedly applying linear Select is less than sorting the entire array

- T is more efficient than S. when i is large enough $i \geq \log n$ that the amount of multiple LS operations surpass the one time cost of sorting

3. worst case complexity for LS with $k=3$ and $k=7$

For $k=5$, it has been shown in class that the worst-case time complexity is $O(n)$, since at least $\frac{3n}{10}$ elements which is less than or greater to the pivot

For $k=3$, the medians of medians will guarantee that at least $1/3$ of the elements are $<$ or $>$ the pivot. With less efficient partitioning, the constant in $O(n)$ complexity could be higher but the complexity class remains $O(n)$

For $k=7$, diving the array into groups of 7 improves the partitioning efficiency over $k=5$. to at least half of the $n/7$ groups contributing 3 elements $<$ or $>$ the pivot, at least $\frac{3n}{14}$ elements $>$ or $<$ the pivot. overall the complexity would still be $O(n)$

4.

Worst-case and Avg complexity of ilSelect Algorithm

- worst case complexity for quickSelect is $O(n^2)$, ilselect is $O(n)$ as the worst-case snario of quick Select is mitigated by linearSelect call.
- Avg, Both QuickSelect and linearSelect have an Avg complexity of $O(n)$

5. The minimal # of Scom operations for the initial Sorting phase is $(\lceil \frac{n}{5} \rceil)$ but the total,