CISC 3220 Homework Chapter 9

Rachel Friedman

April 26, 2020

Exercise 9.1

Question 9.1-1

Show that the second smallest of n elements can be found with $n + \lceil \log n \rceil - 2$ comparisons in the worst case.

The smallest element is found in $n - 1$ comparisons, through a tournament style comparison where each one of the "winners" advances to the next round. This results in $\lceil \log n \rceil$ rounds. The second smallest element is one of those $\lceil \log n \rceil$ elements. We now make $\lceil \log n \rceil - 1$ comparisons with those elements. So, in total, the amount of comparisons is $n - 1 + (\lceil \log n \rceil - 1)$ which is equal to $n + \lceil \log n \rceil - 2$ comparisons.

Exercise 9.2

Question 9.2-3

Write an iterative version of RANDOMIZED-SELECT.

```
PARTITION(A, p, r)
    x = A[r]
    i = p
    for k = p - 1 to r
       if A[k] < x
           i = i + 1
           swap A[i] with A[k]
    i = i + 1
    swap A[i] with A[r]
    return i
```

```
RANDOMIZED-PARTITION(A, p, r)
    x = RANDOM(p - 1, r)
    swap A[x] with A[r]
    return PARTITION(A, p, r)
```

```
RANDOMIZED-SELECT(A, p, r, i)
    while true
        if p == r
            return A[p]
        q = RANDOMIZED-PARTITION(
           A, p, r)
        k = q - p + 1
        if i == k
            return A[q]
        if i < k
            r = q
        else
            p = q
            i = i - k
```

Exercise 9.3

Question 9.3-7

Describe an $\mathcal{O}(n)$ running time algorithm that, given a set S of n distinct numbers and a positive integer $k \leq n$, determines the k numbers in S that are closest to the median of S.

For simplicity's sake, assume n is odd and k is even and the set S is sorted:

1. Use linear time selection to find the median in position $n/2$.
2. Use linear time selection to find the element in position $(n-k)/2$.
3. Use linear time selection to find the element in position $(n+k)/2$.
4. Then traverse the set S to find the elements that are less than $(n-k)/2$, and greater than $(n+k)/2$, and not equal to $n/2$.

Thus, the algorithm takes $\mathcal{O}(n)$ times, since we use linear time selection exactly three times, and traverse the set once.

Question 9.3-8

Let $X[1..n]$ and $Y[1..n]$ be two arrays, each containing n numbers already in sorted order. Give an $\mathcal{O}(\log n)$-time algorithm to find the median of all $2n$ elements in arrays X and Y.

Repeat the following until length of array is 1:

1. Get the median of each array.
2. Take the upper portion of the array with the lower median.
3. Take the lower portion of the array with the higher median.
4. Is there more than one element left in each array?
   (a) Yes - go back to step 1.
   (b) No - return the lower number of the two.

```python
def find_median_two_arrays(a, b):
    if len(a) == 1:
        return min(a[0], b[0])

    median = len(a)/2
    i = median + 1
    if a[median] < b[median]:
        return find_median_two_arrays(a[-i:], b[:i])
    else:
        return find_median_two_arrays(a[:i], b[-i:])
```