CIS 606 Homework 5 Fall 2019

Name: **Reza Shisheie**

Login ID: **reshishe**

1. Exercise 9.3-8 (Textbook page 223): Let $X[1...n]$ and $Y[1...n]$ be two arrays, each containing n numbers already in sorted order. Give an $O(\log n)$-time algorithm to find the median of all $2n$ elements in arrays X and Y .

   It is assummed that there exists an array A as a union of array X and array Y in a sorted fashion:

   $$A = \text{sort}(X \cup Y)$$

   No matter n is an odd or even number, length of array A is $2n$, which is even, and thus there exist two medians in array A. Therefore, there are $n-1$ elements before and after the meddians in array A.

   Assuming M as an element of array X is a median of array A, then there exists w where

   $$M = X[w]$$

   where w is the location of M in array X. Therefore, all the first $w-1$ elements of array X and all the first $n-w$ elements of array Y must be less than or equal to M:

   $$X[1:w]_w + Y[1:n-w]_{n-w} \leq X[w] \tag{1}$$

   As you see the total elements before $X[w]$ is n elements. Likewise the second $n-w$ elements in array X to the end - after $X[w]$ - and second $w-1$ elements in array Y to the endmust be larger than M :

   $$X[w] \leq X[w:n]_{n-w+1} + Y[n-w+1:n]_{w-1} \tag{2}$$

   Combining Eq.(1) and Eq.(2), $X[w]$ must be between the following range of array Y:

   $$Y[n-w] \leq X[w] \leq Y[n-w+1] \tag{3}$$

   Now we just have to develope an algorithm that finds a value in sorted array X in such a way that satisfied Eq.(3).

```
FIND-MEDIAN(X, Y, p, r, n)
 1  // find the middle element of array X
 2  w = (p + r)/2
 3  if Y[n − w] ≤ X[w] ≤ Y[n − w + 1]
 4  // if value satisfies Eq.(3), return w
 5        return w
 6  elseif X[w] < Y[n − w]
 7  // if value is less than Eq.(3) lower bound, look up the right hand
 8        FIND-MEDIAN(X, Y, w + 1, r, n)
 9  elseif X[w] > Y[n − w + 1]
10  // if value is larger than Eq.(3) upper bound, look up the left hand
11        FIND-MEDIAN(X, Y, p, w − 1, n)
12  if p == r
13  // if reached the last element then M is not in X. Flip X and Y and start over
14        FIND-MEDIAN(Y, X, 1, n, n)
```

This algorithm is deviding X by 2 and finding for median, which is a prone and rearch method. Therefore, its complexity is:

$T(n) = T(n/2) + c.$

$a = 1,\ b = 2,\ f(n) = c$

$n^{\log_b a} = n^{(\log_2 1)} = n^0 = 1 \implies f(n) = O(n^{\log_b a})$

$\overset{\text{Case2}}{\implies} T(n) = \Theta(n^{\log_b a}.\log n) = \Theta(n^{\log_2 1}.\log n) = \Theta(\log n)$

If the algorithm does not find w in the first run, it flips X nad Y and reruns . In this case the complexity will be $T(n) = 2.\Theta(\log n)$ which is the same as $T(n) = \Theta(\log n)$

2. Exercise 9-2 c (Textbook page 225): Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm such as SELECT from Section 9.3.

The algorithm has to find the $x_i \epsilon [x_1, x_2, ..., x_n]_{1\times n}$ where:

1. $[x_1, x_2, ..., x_{i-1}] < x_i < [x_{i+1}, ..., x_n]$
2. $\sum_{x_i<x_k}(w_i) < \frac{1}{2}$
3. $\sum_{x_i>x_k}(w_i) \leq \frac{1}{2}$

To satify the first parts, median of median has to be found. This is what is already discussed in the book. To satisfy the second and third part, summation of all elements before and after element k must be checked.

For the most part it is taken from the solution on page 219 of CLRS:

1. Divide the n elements of the input array into $[n/5]$ groups of 5 elements each and at most one group made up of the remaining n mod 5 elements. Cost:$T(n/5)$

2. Find the median of each of the $[n/5]$ groups by first insertion-sorting the elements of each group (of which there are at most 5) and then picking the median from the sorted list of group elements. Cost: 10.

3. Use SELECT recursively to find the median x of the $[n/5]$ medians found in step 2. (If there are an even number of medians, then by our convention, x is the lower median.)

4. Partition the input array around the median-of-medians x using the modified version of PARTITION MM-SELECT$(A, 1, n/5, n/10)$. Let q be one more than the number of elements on the low side of the partition, so that x is the kth smallest element and there are $n - k$ elements on the high side of the partition.

5. Let:

$w_l = (w_p + w_{p+1} + ... + w_{k-1})$

$w_r = (w_{k+1} + w_{k+2} + ... + w_n)$

Now:

If $w_l < 1/2$ and $w_r \leq \frac{1}{2} \longrightarrow$ return $A[k]$

If $w_l < 1/2$ and $w_r \geq \frac{1}{2} \longrightarrow$ mm-search$(A, k+1, r, \frac{1}{2} - w_l))$

If $w_l > 1/2$ and $w_r \geq \frac{1}{2} \longrightarrow$ mm-search$(A, p, k-1, \frac{1}{2})$

FIND-WEIGHTED-MEDIAN(A, p, r, w)

1 $n = r - p + 1$
2 // dividing A into groups of 5 elements. Cost=T(n/5)
3 $A =$ DIVIDE-GROUPS$(A, p, r, 5)$
4 // Sort each column of A ascendingly. Cost=10*(n/5)
5 $A =$ SORT-GROUPS(A)
6 // finding median among the medians of subarrays.
7 $q =$ MM-SELECT$(A, 1, \frac{n}{5}, \frac{n}{10})$
8 // Summation of all weights of A from p tp $q-1$. This is top left portion of matrix. Cost $\frac{3n}{10}$
9 $wSum_l =$ SUM-WEIGHT$(A, p, q-1)$
10 // Summation of all weights of A from $q+1$ tp r. This is the rest of the matix. Cost $\frac{7n}{10}$
11 $wSum_r =$ SUM-WEIGHT$(A, q+1, r)$
12 // if weighted sum is what we expect− >return x element
13 **if** $wSum_l < w$ and $wSum_r \leq w$
14 return $A[q]$
15 // if weighted sum of left side is larger than w − > partition left hand and look for w
16 **elseif** $wSum_l > w$
17 FIND-WEIGHTED-MEDIAN$(A, p, q-1, w)$
18 // if weighted sum of right is larger than w − > partition right hand and look for the remaining of w
19 **elseif** $wSum_r > w$
20 FIND-WEIGHTED-MEDIAN$(A, q+1, r, w - wSum_l)$

Time complexity of this algorithm in worse-case scenario is:

$$T(n) = T(n/5) + T(7n/10) + O(n)$$

where:

1. $T(n/5)$ counts for dividing the matrix into 5 groups
2. $T(7n/10)$ counts for worse case-scenario of searching the larger portion of matrix
3. $O(n)$ counts for:
   3.1. Sorting all subarrays
   3.2. Summation of weights on both sides
   3.3. Partitioning

This is proved by CLRS that has $O(n)$ complexity.