CS 4349.006 Homework 3

Due Thursday, September 23 on eLearning

Please solve the following **2** problems.

1. Consider the following somewhat contrived sorting algorithm:

```
StoogeSort(A[1 .. n]):
  if n = 2 and A[1] > A[2]
      swap A[1] ↔ A[2]
  else if n > 2
      m ← ⌈2n/3⌉
      StoogeSort(A[1 .. m])
      StoogeSort(A[n − m + 1 .. n])
      StoogeSort(A[1 .. m])
```

   (a) State a recurrence for the running time of StoogeSort($A[1 .. n]$). You should assume it takes constant ($O(1)$) time to initiate a recursive call StoogeSort($A[x .. y]$) for any pair of indices x and y.

   (b) State the running time of StoogeSort in big-Oh notation by solving the recurrence from part (a).

   (c) Prove StoogeSort actually sorts its input.

   Advice: Begin by arguing that after the first recursive call to StoogeSort, no element of rank $m+1$ through n lies in the subarray $A[1 .. n-m]$.

2. Suppose we are given a set S of n items, each with a distinct ***value*** and a positive ***weight***. For any element $x \in S$, we define two subsets

   - $S_{<x}$ is the set of elements of S whose value is strictly less than the value of x.
   - $S_{>x}$ is the set of elements of S whose value is strictly more than the value of x.

For any subset $R \subseteq S$, let $w(R)$ denote the sum of the weights of elements in R. The ***weighted median*** of R is the unique element x such that $w(S_{<x}) \leq w(S)/2$ and $w(S_{>x}) < w(S)/2$. In other words, if we were to sort the elements of S *by value* and then add up their weights one-by-one in order, the first element to take our sum strictly beyond half the total weight of S is the weighted median.

Describe and analyze an algorithm to compute the weighted median of a given weighted set in $O(n)$ time. Your input consists of two unsorted arrays $S[1 .. n]$ and $W[1 .. n]$, where for each index i, the ith element has value $S[i]$ and weight $W[i]$.

If you find it helpful, you are free to call the $O(n)$ time procedures MomSelect$(A[1 .. n], k)$ and Partition$(A[1 .. n], p)$ as defined in class. You may also assume any changes to $S[1 .. n]$ are accompanied by identical changes to $W[1 .. n]$ so that at all times and for any index i, value $S[i]$ and weight $W[i]$ both refer to the same weighted element.

Advice: Modify QuickSelect so you 1) are guaranteed to use the median as the pivot and 2) you recursively search the correct subarray to find the weighted *median of the input set. Be careful when you specify which element you are looking for in the recursive call.*