CSCI 3104 **Prof. Chen, Hoenigman**
Problem Set 4 **Fall 2017, CU-Boulder**

1. (10 pts) You are given n metal balls $B_1, \ldots, B_n$, each having a different weight. You can compare the weights of any two balls by comparing their weights using a balance to find which one is heavier.

   (a) Consider the following algorithm to find the heaviest ball:

      i. Divide the n balls into $\frac{n}{2}$ pairs of balls.

      ii. Compare each ball with its pair, and retain the heavier of the two.

      iii. Repeat this process until just one ball remains.

      Illustrate the comparisons that the algorithm will do for the following $n = 8$ input:

$$B_1 : 3, \quad B_2 : 5, \quad B_3 : 1, \quad B_4 : 2, \quad B_5 : 4, \quad B_6 : \tfrac{1}{2}, \quad B_7 : \tfrac{5}{2}, \quad B_8 : \tfrac{9}{2}$$

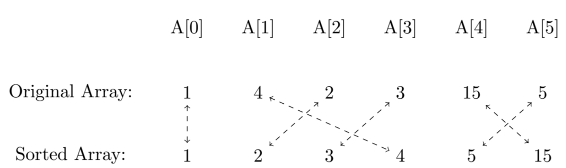
   (b) Show that for n balls, the algorithm (1a) uses at most n comparisons.

   (c) Describe an algorithm that uses the results of (1a) to find the *second* heaviest ball, using at most $\log_2 n$ additional comparisons. There is no need for pseudocode; just write out the steps of the algorithm like we have written in (1a).
   Hint: if you follow sports, especially wrestling, read about the *repechage*.

   (d) Show the additional comparisons that your algorithm in (1c) will perform for the input given in (1a).

2. (10 pts) An array is *almost k sorted* if every element is no more than k positions away from where it would be if the array were actually sorted in ascending order.

   As an example, here is an almost 2-sorted array:



   (a) Write down pseudocode for an algorithm that sorts the original array in place in time $n\,k\log k$. Your algorithm can use a function $\texttt{sort}(A, \ell, r)$ that sorts the sub-array $A[\ell], \ldots, A[r]$ **Note: you will be working on this problem in recitation this week.**

3. (20 pts) Consider the following strategy for choosing a pivot element for the `Partition` subroutine of QuickSort, applied to an array A.

- Let n be the number of elements of the array A.
- If $n \leq 15$, perform an Insertion Sort of A and return.
- Otherwise:
  - Choose $2\lfloor\sqrt{n}\rfloor$ elements at random from n; let S be the new list with the chosen elements.
  - Sort the list S using Insertion Sort and use the median m of S as a pivot element.
  - Partition using m as a pivot.
  - Carry out QuickSort recursively on the two parts.

(a) If the element m obtained as the median of S is used as the pivot, what can we say about the sizes of the two partitions of the array A?

(b) How much time does it take to sort S and find its median? Give a Θ bound.

(c) Write a recurrence relation for the worst case running time of QuickSort with this pivoting strategy.

4. (20 pts) Let A and B be arrays of integers. Each array contains n elements, and each array is in sorted order (ascending). A and B do not share any elements in common. Give a $O(\lg n)$-time algorithm which finds the median of $A \cup B$ and prove that it is correct. This algorithm will thus find the median of the $2n$ elements that would result from putting A and B together into one array. (Note: define the median to be the average of the two middle values of a list with an even number of elements.)