EC 504 – Fall 2020 –1 Homework 3

Due Thursday, Oct. 8, 2020 in the beginning of class. Written and Coding problems submitted in the directory `/projectnb/alg504/yourname/HW3` on your SCC account by Thursday Oct 8, 11:59PM.

Reading Chapter 16-10 on SortingDataStructures.

1. (10 pts) CRLS Exercise 9.3-1. Note that this refers to the general Select problem with finds the i-th element in size for an unsorted list of N objects $a[N]$ That it find the element $x \in a[N]$ so that exactly i-1 are larger. You may answer this problem for the spacial case of **median** discussed in class: That is element such that there are N/2 larger elments.

2. (20 pts) Below you will find some functions. For each of the following functions, please provide:

   - A recurrence $T(n)$ that describes the worst-case runtime of the function in terms of n *as provided* (i.e. without any compiler optimizations to avoid redundant work).
   - The tightest asymptotic upper and lower bounds you can develop for $T(n)$.

   (a)
   ```
   int D(int n) {
         if (n <= 1) return 1;
         int prod = 0;
         for (int ii = 0; ii < n; ii++)
             prod *= D((int) sqrt(n));
         return prod;
   }
   ```

   (b)
   ```
   int E(int n) {
         if (n<= 1) return 1;
         int count = 3;
         int tmp = E(n/2);
         for (int k = 0; k < n; k++)
              for (int m = 1; m <n; m*=2)
                   if (tmp < exp(k+m))
                        count++;
         return E(n/2)*(count%2);
   }
   ```

3. (40 pts) You are given n nuts and n bolts, such that one and only one nut fits each bolt. Your only means of comparing these nuts and bolts is with a function TEST(x, y), where x is a nut and y is a bolt. The function returns +1 if the nut is too big, 0 if the nut fits, and -1 if the nut is too small.

   Design and analyze an algorithm for sorting the nuts and bolts from smallest to largest using the TEST function, such that the worst case performance of the algorithm has asymptotic complexity $O(n^2)$.

   Pseudo-code.

   You can also do a version of quicksort, by picking a bolt and pivoting the nut array, and similarly picking a nut and pivoting the bolt array.

4. (20 pts) Suppose you are given two sorted arrays A, B of integer values, in increasing order, sizes n and m respectively, and $m+n$ is an odd value (so we can define the median value uniquely). Develop an algorithm for finding the median of the combined sorted arrays, and analyze the complexity of your algorithm. It is straight forward to develop an algorithm that is $O(min(n, m))$. Explain in words. You should attempt to construct a code that is worst case complexity is $O(min(n, m))$ to get good performance.

   Implement your algorithm as function completing the code `mergeAP.cpp` in the code in `HW3_codes`. It will read in an input file form the command line. There is large input file called `input_AB.txt`

   I have provides a code `makeABlist.cpp` You can use this to generate other input files. It is recommended to make a very small input to develop and test your solution. The solution should run on `input_AB.txt` or any example that we test it on.

   Put final source code with Makefile with the output file called `input_AB.txt_out` on your top level CCS account in directory (e.g. folder!)

   `/projectnb/alg504/username/HW3`