CS575-20 Assignment 2 (Summer 2020)
(Due June 28 by 11:59pm)

Objective:

(1) Design and analyze an algorithm with D&C or recursion, or dynamic programming
(2) Establish recurrence equation and solve it.
(3) Enhance concept of heap for sorting

There are two parts in this assignment: (A) Theory part and (B) programming part

[Part A] Theory [85%]:

1. (6%) We have a problem that can be solved by a direct (non-recursive) algorithm that operates in N^2 times. We also have a recursive algorithm for this problem that takes NlgN operations to divide the input into two equal pieces and lgN operations to combine the two solutions together. Show whether the direct or the recursive version is more efficient. (Note: For the recursive algorithm, the base case is T(n) = 1 if n=1.)

2. [10%]

   a. [3%]Write the recurrence equation for the code below. Use the number of comparisons as your barometer operation (The *min* operation requires 1 comparison and the *max* operation requires 1 comparison): *): (Note: you can count min and max as the comparison operation and skip the rt-lt <=1)*

```
MinMax(A,lt,rt)
// return a pair with the minimum and the maximum
   if (rt - lt ≤ 1)
       return (min(A[lt], A[rt]), max(A[lt],A[rt]));
   (min1, max1) = MinMax(A,lt, ⌊(lt+rt)/2⌋ );
   (min2, max2) = MinMax(A, ⌊ (lt+rt)/2 ⌋ +1,rt);
   return (min (min1, min2), max(max1, max2));
```

   b. [4%] Show the recursion tree and solve the recurrence equation for this code. For simplicity assume n = 2^k.

   c. [3%] Prove the solution using induction

3. (9%) Given a sorted array of distinct integers A[1, …, n], you want to find out whether there is an index i for which A[i]=i. Give a divide-and-conquer algorithm to solve this problem. Derive the time complexity. (Note: the running time much be less than O(n)).

4. (9%) An Array A[1,…,n] is said to have a majority element if more than half of its entries are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. Show how to solve this problem in O(nlgn) time.(Hint: Split the array A into two arrays A1 and A2 of half the size. Does knowing the majority elements of A1 and A2 help you figure out the majority element of A?) Note that it is required to use a divide-and-conquer approach with O(nlgn) time complexity.

5. [6%] Find the asymptotic bound of the divide and conquer recurrence T(n) using master method:

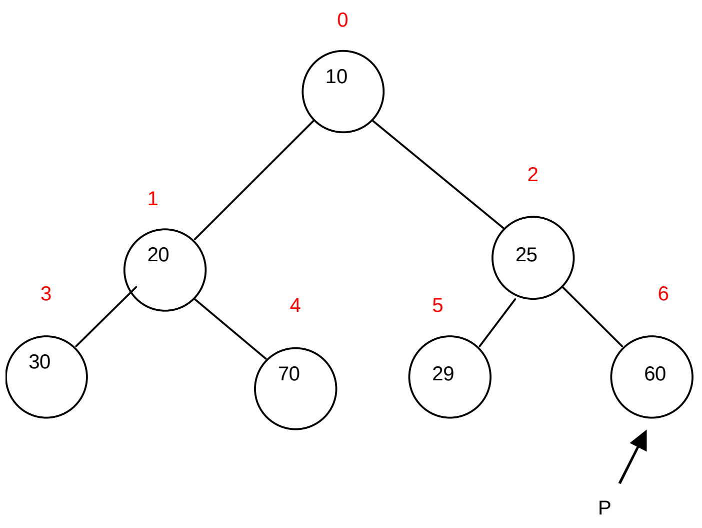
   1. $T(n) = 9*T(n/3) + n^2+4$
   2. $T(n) = 6*T(n/2) + n^2-2$
   3. $T(n) = 4*T(n/2) + n^3 +7$

6. [5%] Solve the following recurrence equation using methods of characteristic equation.

T(n)=8T(n-1)-21T(n-2)+18T(n-3) for n>2
T(0)=0
T(1)=1
T(2)=2

*Hint: if there are two same roots, the solution template is T(n) = C1*r1^n + C2*r2^n + C3*n*r3^n*

7. [6%] Perform the execution deleteMin, then insert(5) on the following heap. Show the heap and the position of *P* after the execution of *deleteMin*, then show the heap after the *insert(5).*



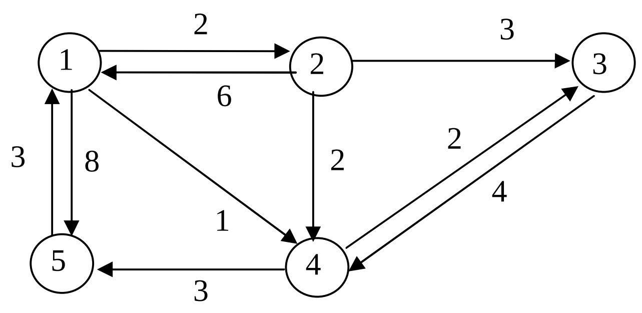
8. [8%] Given the following array:
   a. [2%] Show the essential complete binary tree for the following array:

10	11	7	15	9	3	12	10	22	8

   b. [3%] Convert the essential complete binary tree into a min heap using the fast-make-heap method. Draw the essential complete binary tree after each *siftDown*.
   c. [3%] Convert the essential complete binary tree (from (a)) into a max heap. Using the max-heap sorting algorithm to sort the above sequence in the non-increasing order. Show the heap after the value "11" has been moved to the sorted sequence.

9. [8%] Find the longest common subsequence for the strings x= CACMYCCA, and y = YMCMAMYYCMA. Show the table with the lengths and arrows.

10. (8%) Solve the all-pairs shortest-path problem for the following diagraph. Using the Floyd's algorithm to calculate the values for matrices D^0, D^1, …, D^5 and the corresponding P tables. So the shortest distance between all pairs of nodes can be found.



11. [10%]
Design a dynamic programming algorithm to solve the sum of subsets problem:

There are *n* positive integers, $p_1, p_2, ..., p_n$ and a positive integer sum *S*. Is there a subset *A* of the *n* integers $A \subseteq \{p_1, ..., p_n\}$ such that the sum of the integers in the subset is S, ($\sum_{p_i \in A} p_i = S$)? If there is such a subset the output of the program is *true* otherwise it is *false*.

Example: $p_1 = 3, p_2 = 5, p_3 = 11$ and $S = 16.$ In this case the output is *true* since $p_2 + p_3 = 16$.
Example: $p_1 = 3, p_2 = 5, p_3 = 11$ and $S = 18.$ In this case there is no solution, and the output is *false*.

The problem can be solved with dynamic programming.

A **Boolean** matrix *B* with rows 0 to *n* and columns 0 to *S* is generated. For the examples above the matrix has rows 0 to 3, and columns 0 to 16.

B[*i*, *s*] is **true** if a subset of the first *i* integers sums to *s*, otherwise it is **false**.

The solution to the original problem is **true** if *B*[*n*, *S*]= **true**, otherwise it is **false.**

(1) [4%] Write the recurrence relation for the solution.

(2) [3%] Write pseudo code to compute B.

(3) [3%] Apply dynamic programming to the following problem:
$p_1 = 1, p_2 = 2, p_3 = 4$ and $S = 6.$ Show your matrix B. Each element of B has the value *true* or *false*. Also, use arrows to show the matrix elements that were ordered, or used.

[Part B]: Programming (15%)

1. [15%] Implement the algorithm for finding the closest pair of points in two dimension plane using divide and conquer strategy.

A system for controlling air or sea traffic might need to know which are the two closest vehicles in order to detect potential collisions. This part solves the problem of finding the closest pair of points in a set of points. The set consists of points in R^2 defined by both an x and a y coordinate. The "closest pair" refers to the pair of points in the set that has the smallest Euclidean distance, where Euclidean distance between points $p_1=(x_1,y_1)$ and $p_2=(x_2,y_2)$ is simply $\text{sqrt}((x_1-x_2)^2+(y_1-y_2)^2)$. If there are two identical points in the set, then the closest pair distance in the set will obviously be zero.

Input data: n points with coordinates:

X coordinates: p[0].x, p[1].x, p[2].x,…., p[n].x

Y coordinates: p[0].y, p[1].y, p[2].y,…., p[n].y

Output: minimum distance between points p[i] and p[j] (index i and j should be identified)

Input data for test:

```
n = 10000;
for(i=0; i<n; i++)
 {
   p[i].x= n- i;
   p[i].y= n- i;
    }
```

Other Input data for test:

```
n = 10000;
for(i=0; i<n; i++)
 {
   p[i].x= i*i;
   p[i].y= i*i;
    }
```

Or:

Input:

If X>0
X=1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, ……,19995, 19997, 19999 (odd numbers)
Y=int($\sqrt{9999^2 - (X - 10000)^2}$)

Else
X=0, -2, -4, -6, -8, -10, -12, ……, -19996, -19998, -20000 (even number)
Y= int($\sqrt{10000^2 - (X + 10000)^2}$)

3. (Optional: Extra 10%) apply the close-pair algorithm to the three dimensional case.

Note 1:

Your report (.doc) of the programming part should follow the following format:

(1) Algorithm description

(2) Major codes

(3) Running results

(4) Report of any bugs

Note 2:

2.1 For Part B, your code package includes your code, makefile, executable file, and write-up (.doc).

2.2 Your program will read an input file and write an output file.

2.3 Your program should be invoked like this…

```
prompt>submission inputFile.txt outputFile.txt
```

where inputFile.txt is referring to an input file,
ouputFile.txt is referring to an output file.