Homework Assignment 2 (No Collaborators)

1. **Exercise 6.5−8**

Algorithm 1 Heap-Delete Algorithm

```
function HEAP-DELETE(A, i)
    A[i] ← A[n]
    n ← n − 1
    HEAPIFY(A, i)
end function
```

2. **Exercise 7.2−3** The worst case running time of Quicksort is $\Theta\left(n^2\right)$ for a descending order array when the selection of the pivot is always the right element. In this case, each iteration of Partition reduces the size of the sub-array by 1. Thus, we have a recurrence relation equal to
$$T(n-1)+\Theta(n)=\Theta(n^2)$$

3. **Modified Exercise 8.2−1**
Using Figure 8.2 in textbook as a model, illustrate the operation of COUNTING-SORT based on the string (array of 16 characters): "NTUEECSALGORITHM". Please mark the two T 's as T_1 and T_2, and the two E 's as E_1 and E_2 according to their order in the input, and show their positions during the processing. Assume you have only the 26 characters, $A, B, \ldots, Z$, and thus you may work on the array of the 26 characters.

4. **Exercise 8.2−4**
Describe an algorithm that, given n integers in the range 0 to k, preprocesses its input and then answers any query about how many of the n integers fall into a range $[a \ldots b]$ in $O(1)$ time. Your algorithm should use $\Theta(n+k)$ preprocessing time.

5. **Exercise 9.3−7**
Describe an $O(n)$-time algorithm that, given a set S of n distinct numbers and a positive integer $k \leq n$, determines the k numbers in S that are closest to the median of S.

6. **Search Trees**
a. Give the binary search tree that results from successively inserting the keys $8, 2, 1, 6, 5, 7, 9, 10$ into an initially empty tree.
b. Label each node in the tree with R or B denoting the respective colors RED and BLACK so that the tree is a legal red-black tree.
c. Give the red-black tree that results from inserting the key 4 into the tree of (b).
d. Give the red-black tree that results from deleting the key 7 from the tree of (c)

7. **Dynamic Programming Implementations**
a. Find an optimal parenthesization o a matrix-chain product whose sequence of dimensions is ¡ 3, 5, 7, 9, 11 ¿.
b. Determine an LCS of ¡ C, A, B, A, C, B, D ¿ and ¡ A, D, B, A, C, D ¿.
c. Determine the cost and structure of an optimal binary search tree for a set of $n=6$ keys with the following probabilities: $p_i = 0.05, 0.09, 0.10, 0.05, 0.12, 0.15, i = 1, .., 6$, respectively, and $q_i = 0.03, 0.06, 0.07, 0.11, 0.08, 0.05, 0.04, i = 0, \ldots, 6$, respectively

8. **(20 pts) Given a log of wood of length k, Woody the woodcutter will cut it once, in any place you choose, for the price of k dollars. Suppose you have a log of length L, marked to be cut in n different locations labeled $1, 2, \ldots, n$. For simplicity, let indices 0 and $n+1$ denote the left and right endpoints of the original log of length L. Let the distance of mark i from the left end of the log be d_i, and assume that $0 = d_0 < d_1 < d_2 < \ldots < d_n < d_{n+1} = L$. The wood-cutting problem is the problem of determining the sequence of cuts to the $\log$ that will (1) cut the log at all the marked places, and (2) minimize your total payment to Woody.**
   a. (4 pts) Give an example with $L = 4$ illustrating that two different sequences of cuts to the same marked log can result in two different costs.
   b. (9 pts) Let $c(i, j)$ be the minimum cost of cutting a log with left endpoint i and right endpoint j at all its marked locations. Suppose the log is cut at position m, somewhere between i and j. Define the recurrence of $c(i, j)$ in terms of i, m, j, d_i, and d_j. Briefly justify your answer.
   c. (7 pts) Using part (b), give an efficient algorithm to solve the wood-cutting problem. Use a table C of size $(n+1) \times (n+1)$ to hold the values $C[i][j] = c(i, j)$. What is the running time of your algorithm?

9. **(20 pts) Let $X = x_1 x_2 \ldots x_m$ and $Y = y_1 y_2 \ldots y_n$ be two character strings. This problem asks you to find the maximum common substring length for X and Y. Notice that substrings are required to be contiguous in the original strings. For example, photograph and tomography have common substrings ph, to, ograph, etc. The maximum common substring length is 6 .**
   a. (a) (4 pts) The following gives the computation of the maximum common suffix and substring lengths on the two strings, $ABAB$ and BAB, similar to the table used for computing the length of LCS in class. Only partial results are given. Please complete all the entries in the table. Longest common suffix Longest common substring

		A	B	A	B
	0	0	0	0	0
B	0	0			
A	0				
B	0				

		A	B	A	B
	0	0	0	0	0
B	0	0			
A	0				
B	0				

   b. (5 pts) Dynamic programming can be used to find the longest common substring efficiently. The idea is to find length of the longest common suffix for all substrings of both strings. Suppose $\gamma = \alpha\beta$ is the concatenation of two strings; we say that β is a suffix. Find the optimal substructure for the longest common suffix problem (a recurrence relation for $\mathrm{LCSuff}(X, Y, m, n)$).
   c.

   (5 pts) Define the longest common substring length $\mathrm{LCSubStr}(X, Y, m, n)$ in terms of $\mathrm{LCSuff}(X, Y, i,$

   d. (6 pts) Give a dynamic programming algorithm for solving this problem. What are the time and space complexity of your algorithm?