

Homework Assignment 2 (No Collaborators)

1. Exercise 6.5–8

Algorithm 1 Heap-Delete Algorithm

```

1: function HEAP-DELETE( $A, i$ )
2:    $A[i] \leftarrow A[n]$ 
3:    $n \leftarrow n - 1$ 
4:    $HEAPIFY(A, i)$ 
5: end function
  
```

2. **Exercise 7.2–3** The worst case running time of Quicksort is $\Theta(n^2)$ 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

Do Half with Friend! Using Figure 8.2 in textbook as a model, illustrate the operation of COUNTING-SORT based on the string (array of 16 characters): "NTUEEC SALGORITHM". 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, \dots, Z , and thus you may work on the array of the 26 characters.

4. **Exercise 8.2–4** Given n integers in the range of 0 to k as array A , create an k sized auxiliary array B initialized to 0. For each element i in A increment $B[A[i]]$. Compute the running sum for each element i in B and obtain the number of elements less than or equal to i . At this point, this is the initial steps in counting sort. To compute the number of elements in range $[a..b]$ we now have an $\Theta(1)$ computation: $B[b] - B[a]$ in range $[a..b]$
5. **Exercise 9.3–7** With the assumption that the set S is a sorted array of elements, we have the median given by index $n/2$. Then the closest k elements must be within the range of $(n-k)/2 \leq n/2 \leq (n+k)/2$. We can use linear search to traverse through these indices to find elements greater than $(n-k)/2$, not equal to $n/2$, and less than $(n+k)/2$.

6. Search Trees

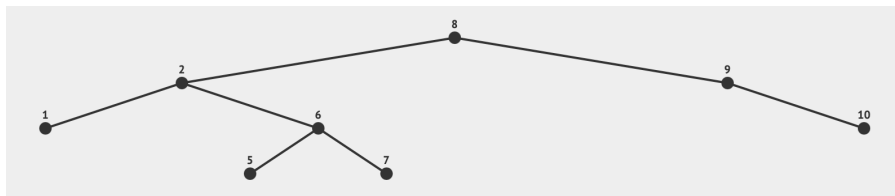


Figure 1: Binary Search Tree

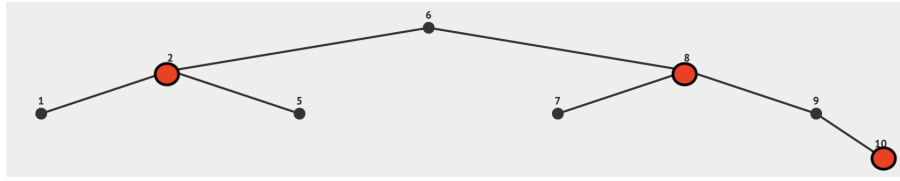


Figure 2: Red Black Tree

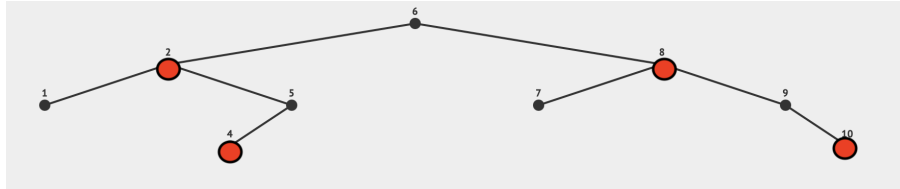


Figure 3: Red Black Tree Add 4

7. Dynamic Programming Implementations

- Find an optimal parenthesization to a matrix-chain product whose sequence of dimensions is $\langle 3, 5, 7, 9, 11 \rangle$.
- Determine an LCS of $\langle C, A, B, A, C, B, D \rangle$ and $\langle A, D, B, A, C, D \rangle$.
- Determine the cost and structure of an optimal binary search tree for a set of $n = 6$ keys with the following probabilities: $p_i = 0.05, 0.09, 0.10, 0.05, 0.12, 0.15, i = 1, \dots, 6$, respectively, and $q_i = 0.03, 0.06, 0.07, 0.11, 0.08, 0.05, 0.04, i = 0, \dots, 6$, 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 $1, 2, \dots, n$. 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 $0 = d_0 < d_1 < d_2 < \dots < d_n < d_{n+1} = L$. 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.

- (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.
- (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.
- (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

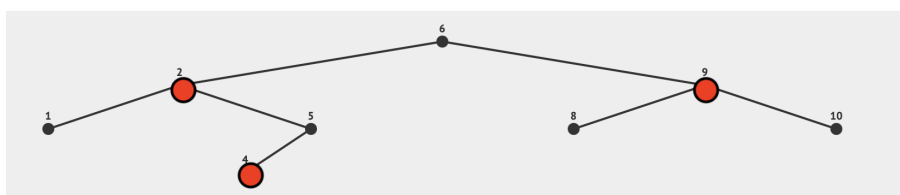


Figure 4: Red Black Tree Remove 7

running time of your algorithm?

9. (20 pts) Let $X = x_1x_2 \dots x_m$ and $Y = y_1y_2 \dots y_n$ 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.