National Taiwan University
Department of Electrical Engineering
Algorithms, Spring 2024

Classes:
Prof. Shao-Hua Sun &
Prof. Hui-Ru Chiang

Homework #2

(on-line submission due: 9:10 AM, 11 April 2024; late submissions are not allowed)

TA: Hsin-Tzu Chang sheena9101019@gmail.com and Yun-Yao Tien erictien02@gmail.com

Collaboration policy: You can discuss the problems with other students, but you must write the final answers by yourself. Please specify all of your collaborators (names and student id's) or resources (websites) for each problem. If you solve some problems by yourself, please also specify "no collaborators". Homework without collaborator specification will not be graded.

1. (10 pts) The operation **HEAP-DELETE(A, i)** deletes the item in node i from heap A. Give an implementation of **HEAP-DELETE** that runs in $O(\log n)$ time for an n-element max-heap.

2. (10 pts) Show that the running time of **QUICKSORT** is $\Theta(n^2)$ when the array A contains distinct elements and is sorted in decreasing order.

3. (10 pts) Modified Exercise 8.2-1 (page 196)

   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, $A, B, ..., Z$, and thus you may work on the array of the 26 characters.

4. (10 pts) 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 $[a..b]$ in $O(1)$ time. Your algorithm should use $\Theta(n + k)$ preprocessing time.

5. (10 pts) Describe an $O(n)$-time algorithm that, given a set S of n distinct numbers and a positive integer $k \le n$, determines the k numbers in S that are closest to the median of S.

6. (20 pts) 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.

7. (30 pts) Dynamic programming implementations.

   (a) Find an optimal parenthesization of 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 $q_i = 0.03, 0.06, 0.07, 0.11, 0.08, 0.05, 0.04, i = 0, ..., 6$, 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 $X = x_1 x_2 \dots x_m$ and $Y = y_1 y_2 \dots y_n$ 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) (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.

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

Longest common ***suffix***

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

Longest common ***substring***

   (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 $LCSuff(X, Y, m, n)$).

   (c) (5 pts) Define the longest common substring length $LCSubStr(X, Y, m, n)$ in terms of $LCSuff(X, Y, i, j)$.

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

10. (20 pts) Given a set C of N chords of a circle (see Figure 1(a)), assume that no two chords of C share an endpoint. Number the endpoints of these chords from 0 to $2N-1$, clockwise around the circle (see Figure 1(c)). Let $M(i,j), i \leq j$, denote the **number** of chords in the maximum *planar subset* (i.e., no two chords overlap each other in the subset) in the region formed by the chord ij and arc ij. (Here we ignore the ordering of i and j.) As the example shown in Figure 1(b), $M(2,7) = 1$ and $M(0,11) = 3$.

   (a) (10 pts) For the case where chord $kj \in C$, $k \notin [i,j]$, the recurrence for $M(i,j)$ is given by $M(i,j) = M(i, j-1)$. Give the respective recurrences for $M(i,j)$ for the other two cases where (1) chord $kj \in C$, $k \in [i,j]$, and (2) chord $ij \in C$.

   (b) (10 pts) Based on the three cases derived in (a), sketch a bottom-up dynamic programming approach for computing the size of the maximum planar subset in a circle of N chords. What is the time complexity of your algorithm?

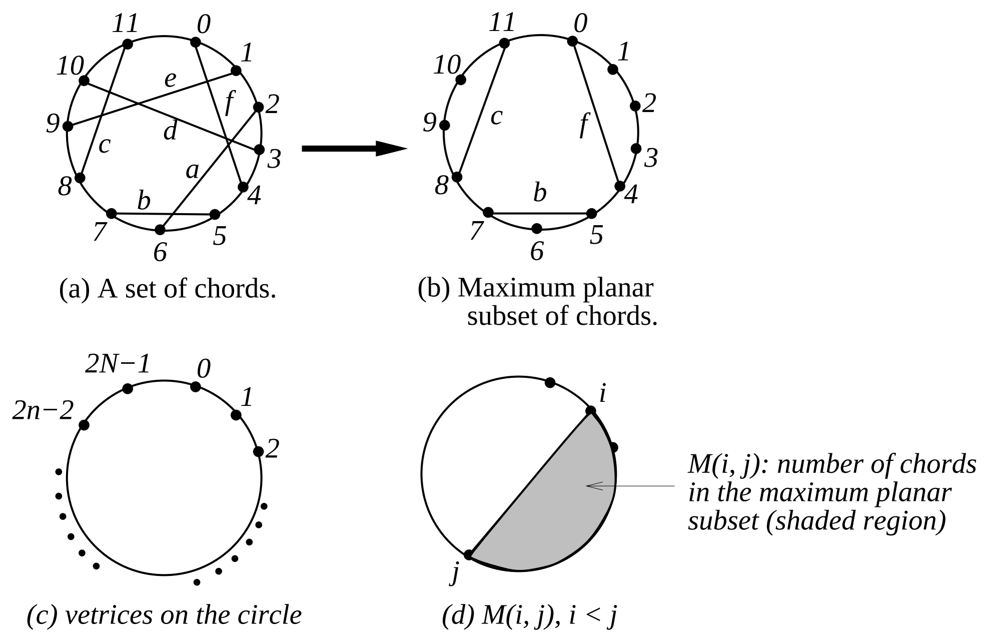
11
0
1
10
e
f
2
9
d
3
c
a
8
4
b
7
5
6
(a) A set of chords.
11
0
1
10
2
9
c
f
3
8
4
b
7
5
6
(b) Maximum planar subset of chords.
2N−1
0
1
2n−2
2
(c) vetrices on the circle
i
M(i, j): number of chords in the maximum planar subset (shaded region)
j
(d) M(i, j), i < j

Figure 1: Maximum planar subset.