National Taiwan University
Department of Electrical Engineering
Algorithms, Fall 2019

Handout #5
October 1, 2019
James Chien-Mo Li & Yao-Wen Chang

Homework #2 (due 6pm November 1, 2019 @ BL 406)

Instructions: Submit your Homework #2 6pm November 1, 2019 @ BL 406. 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) for each problem. If you solve some problems by yourself, please also specify "no collaborators". Homework without collaborator specification will not be graded.

1. Modified Problem 7-1(a) (page 185)

   The version of Partition given in Pages 171–174 of the textbook is not the original partitioning algorithm. Here is the original partition algorithm, which is due to C. A. R. Hoare:

```
HOARE-PARTITION($A, p, r$)
1   $x = A[p]$
2   $i = p - 1$
3   $j = r + 1$
4   while TRUE
5       repeat
6           $j = j - 1$
7       until $A[j] \leq x$
8       repeat
9           $i = i + 1$
10      until $A[i] \geq x$
11      if $i < j$
12          exchange $A[i]$ and $A[j]$
13      else return $j$
```

   Demonstrate the operation of Hoare-Partition based on the string (array of 16 characters): "NTUEECSALGORITHM", showing the values of the array and auxiliary values after each iteration of the while loop in lines 4-13. 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.

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

3. Exercise 8.2-4 (page 197)

   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.

4. Problem 8-4 (a), (b) (pages 206-207)

Suppose that you are given n red and n blue water jugs, all of different shapes and sizes. All red jugs hold different amounts of water, as do the blue ones. Moreover, for every red jug, there is a blue jug that holds the same amount of water, and vice versa. Your task is to find a grouping of the jugs into pairs of red and blue jugs that hold the same amount of water. To do so, you may perform the following operation: pick a pair of jugs in which one is red and one is blue, fill the red jug with water, and then pour the water into the blue jug. This operation will tell you whether the red or the blue jug can hold more water, or that they have the same volume. Assume that such a comparison takes one time unit. Your goal is to find an algorithm that makes a minimum number of comparisons to determine the grouping. Remember that you may not directly compare two red jugs or two blue jugs.

a. Describe a deterministic algorithm that uses $\Theta(n^2)$ comparisons to group the jugs into pairs.
b. Prove a lower bound of $\Omega(n \lg n)$ for the number of comparisons that an algorithm solving this problem must make.

5. Problems 9.1-1 (page 215)

   Show that the second smallest of n elements can be found with $n + \lceil \lg n \rceil - 2$ comparisons in the worst case. (Hint: Also find the smallest element.)

6. Exercise 9.3-8 (page 223)

   Let $X[1..n]$ and $Y[1..n]$ be two arrays, each containing n numbers already in sorted order. Give an $O(\lg n)$-time algorithm to find the median of all $2n$ elements in arrays X and Y. (No pseudo code is needed)

7. Exercise 12.2-1 (a), (b), and (d) (page 293)

   Suppose that we have numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. Which of the following sequences could *not* be the sequence of nodes examined?

   (a) 2, 252, 401, 398, 330, 344, 397, 363.
   (b) 924, 220, 911, 244, 898, 258, 362, 363.
   (d) 2, 399, 387, 219, 266, 382, 381, 278, 363.

8. Problem 12-2 (page 304)

   Given two strings $a = a_0 a_1 \ldots a_p$ and $b = b_0 b_1 \ldots b_p$, where each a_i and each b_j is in some ordered set of characters, we say that string a is ***lexicographically less than*** string b if either

   1. there exists an integer j , where $0 \leq j \leq min(p, q)$, such that $a_i = b_i$ for all $i = 0, 1, \ldots, j-1$ and $a_j < b_j$, or
   2. $p < q$ and $a_i = b_i$ for all $i = 0, 1, \ldots, p$.

   For example, if a and b are bit strings, then $10100 < 10110$ by rule 1 (letting $j = 3$) and $10100 < 101000$ by rule 2. This ordering is similar to that used in English-language dictionaries.

   The ***radix tree*** data structure shown in Figure 1 stores the bit strings 1011, 10, 011, 100, and 0. When searching for a key $a = a_0 a_1 \ldots a_p$, we go left at a node of depth i if $a_i = 0$ and right if $a_i = 1$. Let S be a set of distinct bit strings whose lengths sum to n. Show how to use a radix tree to sort S lexicographically in $\Theta(n)$ time. For the example in Figure 1, the output of the sort should be the sequence 0, 011, 10, 100, 1011.

9. 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-block 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).

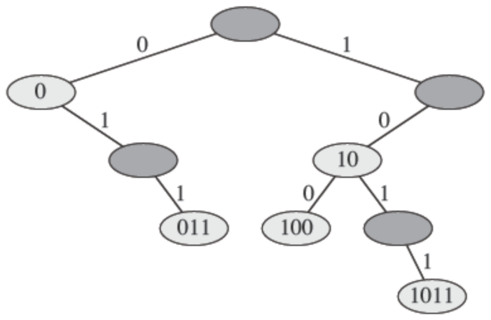

Figure 1: A radix tree storing the bit strings 1011, 10, 011, 100, and 0. We can determine each node's key by traversing the simple path from the root to that node. There is no need, therefore, to store the keys in the nodes; the keys appear here for illustrative purposes only. Nodes are heavily shaded if the keys corresponding to them are not in the tree; such nodes are present only to establish a path to other nodes.

10. Problem 13-2 (pages 332–333)

    The ***join*** operation takes two dynamic sets S_1 and S_2 and an element x such that for any $x_1 \in S_1$ and $x_2 \in S_2$, we have $x_1.key \leq x.key \leq x_2.key$. It returns a set $S = S_1 \cup \{x\} \cup S_2$. In this problem, we investigate how to implement the join operation on red-black trees.

    (a) Given a red-black tree T, let us store its black-height as the new attribute $T.bh$. Argue that RB-INSERT and RB-DELETE can maintain the bh attribute without requiring extra storage in the nodes of the tree and without increasing the asymptotic running times. Show that while descending through T, we can determine the black-height of each node we visit in $O(1)$ time per node visited.

    We wish to implement the operation RB-JOIN(T_1, x, T_2), which destroys T_1 and T_2 and returns a red-black tree $T = T_1 \cup \{x\} \cup T_2$. Let n be the total number of nodes in T_1 and T_2.

    (b) Assume that $T_1.bh \geq T_2.bh$. Describe an $O(\lg n)$-time algorithm that finds a black node y in T_1 with the largest key from among those nodes whose black-height is $T_2.bh$.

    (c) Let T_y be the subtree rooted at y. Describe how $T_y \cup \{x\} \cup T_2$ can replace T_y in $O(1)$ time without destroying the binary-search-tree property.

    (d) What color should we make x so that red-black properties 1, 3, and 5 are maintained? Describe how to enforce properties 2 and 4 in $O(\lg n)$ time.

    (e) Argue that no generality is lost by making the assumption in part (b). Describe the symmetric situation that arises when $T_1.bh \leq T_2.bh$.

    (f) Argue that the running time of RB-JOIN is $O(\lg n)$.

11. Dynamic programming implementations. (a) Find an optimal parenthesization of a matrix-chain product whose sequence of dimensions is $< 3, 5, 10, 6, 8, 30 >$. b) Determine an LCS of $< A, B, C, D, A, B >$ and $< B, D, A, C, D, B >$. (c) Determine the cost and structure of an optimal binary search tree for a set of $n = 6$ keys with the following probabilities: $p_i = 0.07, 0.09, 0.10, 0.04, 0.12, 0.14, i = 1, .., , 6$, respectively, and $q_i = 0.04, 0.06, 0.07, 0.09, 0.08, 0.07, 0.03, i = 0, ..., 6$, respectively.

12. 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) 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) 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) 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?

13. You are given a sequence of n circuit cells $C =< c_1, c_2, \ldots, c_n >$ in a single row with the **fixed** cell order from left to right, $c_1 c_2 \ldots c_n$, each cell c_i with its **bottom-left** coordinate x_i and the width w_i, and the minimum spacing ϕ_{c_i,c_j} for a pair of cells c_i and c_j, $i \neq j$. For example, Figure 2(a) shows an initial placement of four circuit cells.

You are ask to minimize the total length of placing the n cells, flipping or not flipping each cell. That is, cell c_i can have two orientations, **not flipped (unflipped)** and **flipped**, denoted by c_i^p and c_i^r, respectively. A cell flipping graph can be constructed to visualize the cell flipping problem, as shown in Figure 2(b) where the nodes f_i^p and f_i^r respectively represent the two orientations c_i^p and c_i^r of the cell c_i, and the number beside each edge between two nodes (i.e., edge weight) denotes the minimum spacing of the two corresponding cell boundaries.

(a) For the example shown in Figure 2, the initial unflipped cell placement gives the total row length of 40, with $w_1 = 6, w_2 = 5, w_3 = 8$, and $w_4 = 9$, and the minimum spacing $\phi_{c_1^p,c_2^p} = 3$, $\phi_{c_1^p,c_2^r} = 2$, and so on. Find the optimal cell flipping with the minimum total row length for these four cells c_1, c_2, c_3, and c_4. What is the optimal row length? Which cell(s) should be flipped to achieve the optimal solution?

(b) This problem exhibits the optimal substructure with overlapping subproblems. Explain the properties of (1) the optimal substructure, and (2) overlapping subproblems.

(c) Let f^* denote the optimal cell flipping, and T denote the cost function of the nodes f_i^p, f_i^r, and f^*; i.e., $T(f_i^p)$ gives the x-coordinate of the unflipped cell i. Find the recurrences for $T(f_i^\alpha)$, $\alpha \in \{p, r\}$, and $T(f^*)$. **(Hint: in terms of x_i, w_i, $T(f_i^\alpha)$, $\phi_{c_i^\alpha,c_j^\alpha}$, etc., with appropriate indices.)**

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

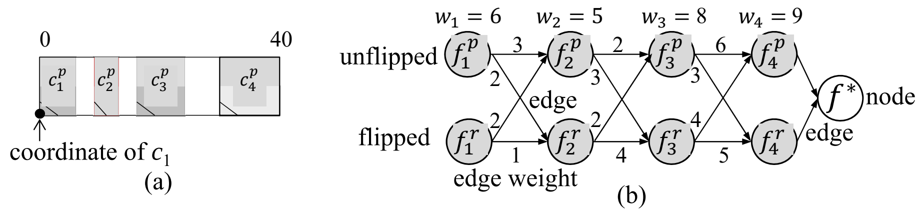

Figure 2: (a) An initial placement without any cell being flipped. (b) Cell flipping graph.

14. (DIY Problem) For this problem, you are asked to design a problem *set* related to Chapter(s) 6–9, 12, 13, and/or 15 and give a sample solution to your problem set. Grading on this problem will be based upon the *quality* of the designed problem as well as the *correctness* of your sample solution.