Recitation: Week 7

EE4033 Algorithms, Fall 2019

Instructor: Yao-Wen Chang, James Chien-Mo Li, and Iris Hui-Ru Jiang

2019-10-23

Presenter:
蔡宇傑 Yu-Jie Cai r07943111@ntu.edu.tw
吳辰鋐 Chen-Hung Wu r07943150@ntu.edu.tw

HW2

- Deadline: 11/1(Fri.) 6:00 PM @ BL406

- Collaboration policy

  Please specify all of your collaborators (name and student id) for each problem. If you solve some problems by yourself, please also specify “no collaborators”. Problems without collaborator specification will not be graded.

1. Problem 7-1(a)

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.

- Input is "NTUEECSALGORITHM"
- Only PARTITION
- Show your steps
- Mark two T's and two E's

```
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 $A[j] \le x$
8       repeat
9           $i = i + 1$
10      until $A[i] \ge x$
11      if $i < j$
12          exchange $A[i]$ with $A[j]$
13      else return $j$
```

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

- Input is "NTUEECSALGORITHM"
- COUNTING-SORT
- Follow the steps in figure 8.2
- Mark two T's and two E's

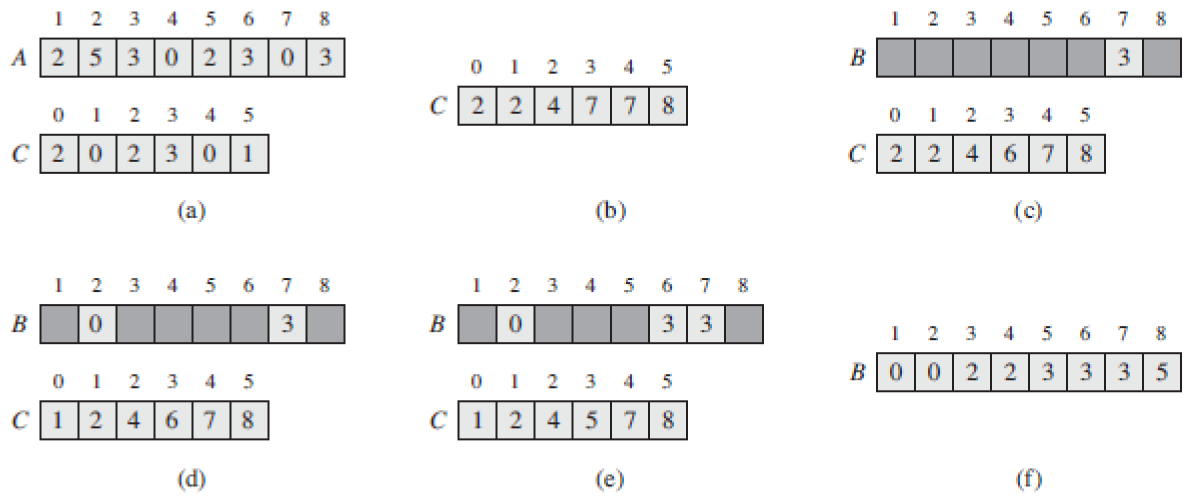


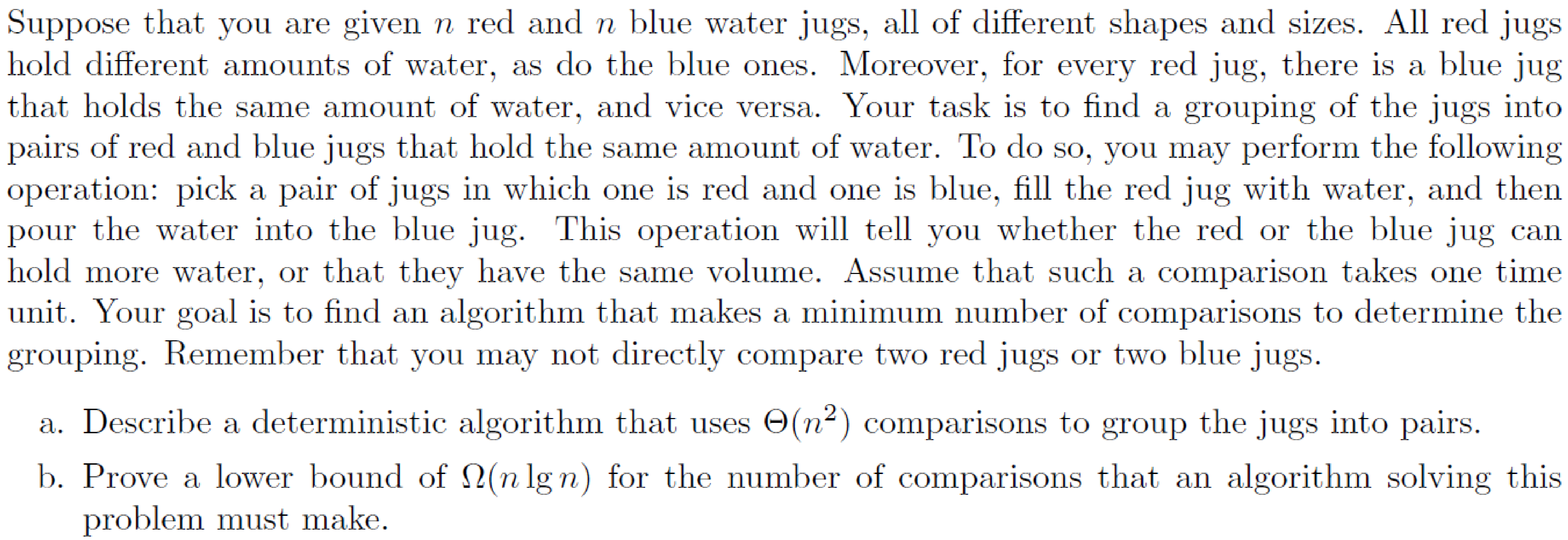
Figure 8.2

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

- Please explain why the runtime of your algorithm is $\Theta(n+k)$ and answer in $O(1)$

4. Problem 8-4 (a), (b)



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.

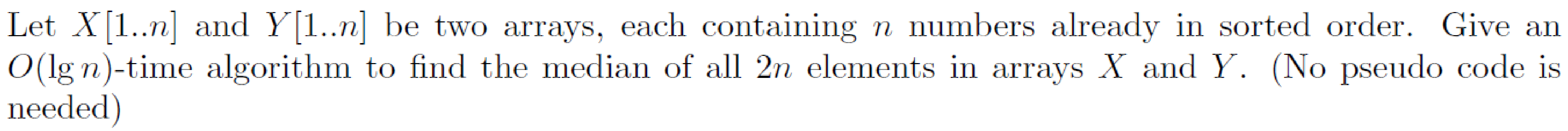
- Please explain why the lower bound for number of comparison is Ω*(nlgn)*

5. Problem 9.1-1

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.)

- Please explain why the comparison of your algorithm can less than $n + [lg] - 2$

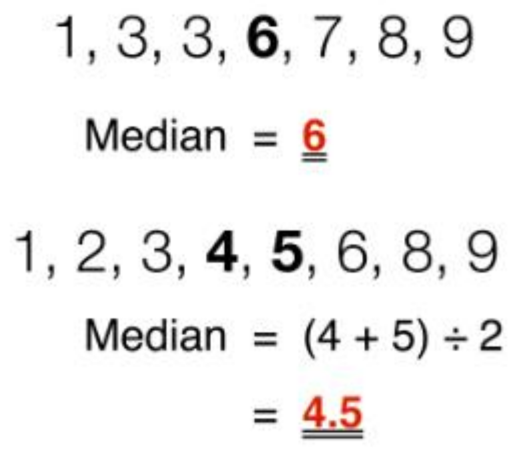
6. Exercise 9.3-8



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)

- Please explain why the runtime of your algorithm is *O(lgn)*

- Median



1, 3, 3, **6**, 7, 8, 9

Median = $\underline{\underline{6}}$

1, 2, 3, **4**, **5**, 6, 8, 9

Median = (4 + 5) ÷ 2

= $\underline{\underline{4.5}}$

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

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.

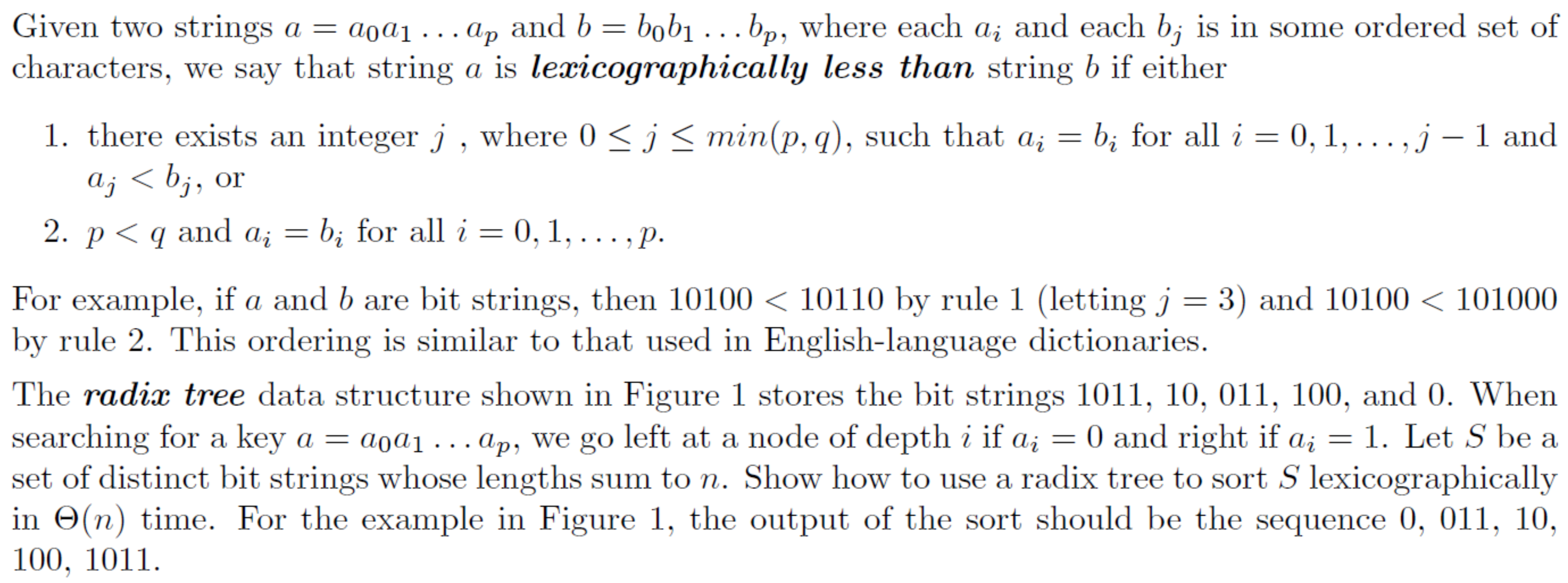
(b) 924, 220, 911, 244, 898, 258, 362, 363.

(d) 2, 399, 387, 219, 266, 382, 381, 278, 363.

- Binary-search-tree property

- If the answer is NO, please explain

8. Problem 12-2



Given two strings $a = a_0a_1 \ldots a_p$ and $b = b_0b_1 \ldots b_p$, 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 $a = a_0a_1 \ldots a_p$, 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.

- N means **sum** of bit strings length
- Explain the algorithm, and prove its complexity

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).

- In (b), please **directly colorize** the BST generated form (a)
- In (c) and (d), please use the algorithm taught in class

10. Problem 13-2

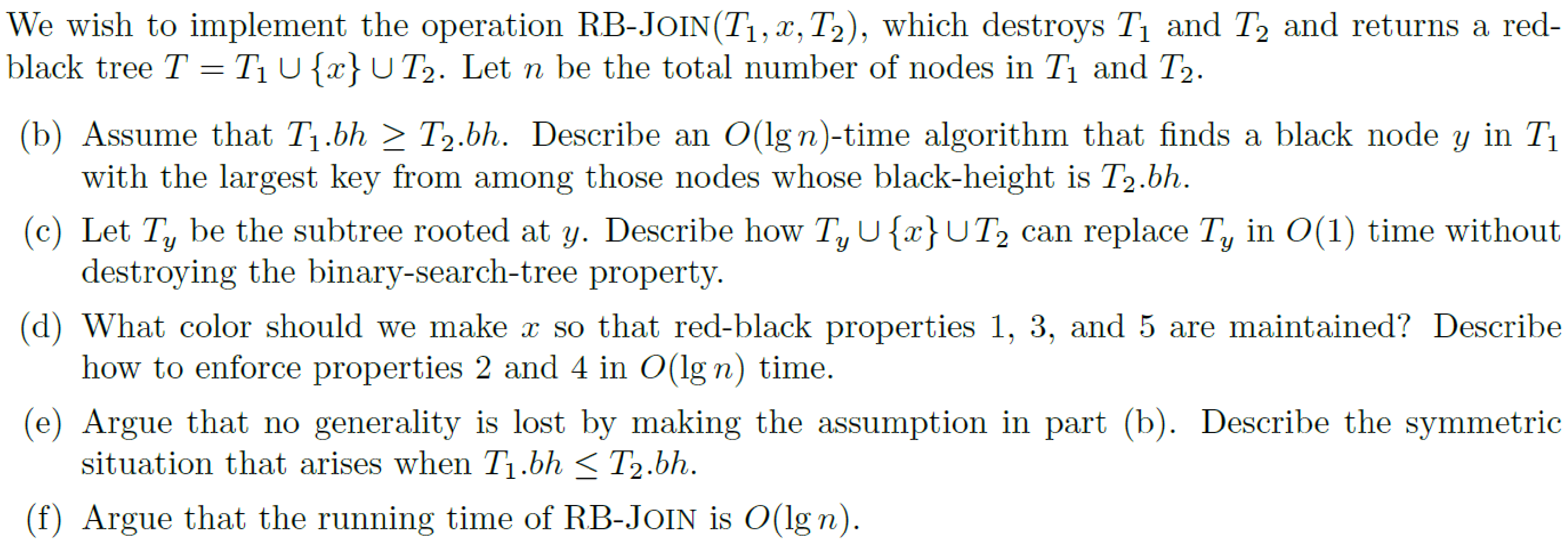
The ***join*** operation takes two dynamic sets S_1 and S_2 and an element x such that for any $x_1 \in S_1$ and $x_2 \in S_2$, we have $x_1.key \leq x.key \leq x_2.key$. It returns a set $S = S_1 \cup \{x\} \cup S_2$. In this problem, we investigate how to implement the join operation on red-black trees.

- In (a), we only discuss RB-Insert and RB-Delete

(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.

10. Problem 13-2

- From (b) to (d), the final purpose is construct “legal” RB tree



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 $O(\lg n)$.

- Discuss the complexity for each sub-problem

11. Dynamic Programming Implementations

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.

- In (a), please give the matrices *m* and *s*
- In (b), please draw the table to compute the LCS
- In (c), please give the matrices *e*, *w* and *root*. Don’t forget to show the final BST result

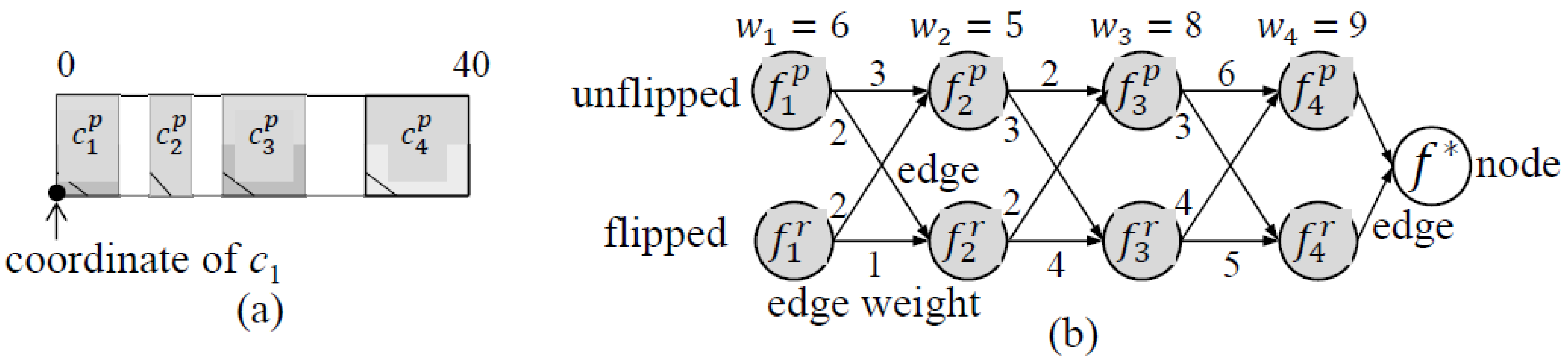
12. Woodcutter Problem

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.

- Use the terms of *l*, *m*, *j*, d_i, and d_j to construct the recurrence formula

- Prove the complexity of your algorithm

13. Flipped and Unflipped Cell Placement



- In (c), please notice that *f** and *f*$^{\alpha}$ are different
- In (d), please prove the complexity

$(\log n)!$ vs. $(\log n)^{\log n}$

- We know that $\log(n!) = \Theta(n \log n)$

- After taking log
  - $\log((\log n)!) = \Theta(\log n \log\log n)$
  - $\log((\log n)^{\log n}) = \log n \log\log n = \Theta(\log n \log\log n)$

- Thus, $(\log n)! = \Theta((\log n)^{\log n})$? Wrong!
  - Think about taking log on 2^n and 4^n