

## Homework Assignment 3 (No Collaborators)

1. **Exercise 16.1–4** Suppose that we have a set of activities to schedule among a large number of lecture halls, where any activity can take place in any lecture hall. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient greedy algorithm to determine which activity should use which lecture hall. (This problem is also known as the interval-graph coloring problem. We can create an interval graph whose vertices are the given activities and whose edges connect incompatible activities. The smallest number of colors required to color every vertex so that no two adjacent vertices have the same color corresponds to finding the fewest lecture halls needed to schedule all of the given activities).
2. **Exercise 16.2–2** Prove that the fractional knapsack problem has the greedy-choice property.
3. **Exercise 16.2–7** Suppose you are given two sets  $A$  and  $B$ , each containing  $n$  positive integers. You can choose to reorder each set however you like. After reordering, let  $a_i$  be the  $i$ th element of set  $A$ , and let  $b_i$  be the  $i$ th element of set  $B$ . You then receive a payoff of  $\prod_{i=1}^n a_i^{b_i}$ . Give an algorithm that will maximize your payoff. Prove that your algorithm maximizes the payoff, and state its running time.
4. **Exercise 16.3–8** Suppose that a data file contains a sequence of 8-bit characters such that all 256 characters are about equally common: the maximum character frequency is less than twice the minimum character frequency. Prove that Huffman coding in this case is no more efficient than using an ordinary 8-bit fixed-length code.
5. **Exercise 17–2** Binary search of a sorted array takes logarithmic search time, but the time to insert a new element is linear in the size of the array. We can improve the time for insertion by keeping several sorted arrays. Specifically, suppose that we wish to support SEARCH and INSERT on a set of  $n$  elements. Let  $k = \lceil \lg(n+1) \rceil$ , and let the binary representation of  $n$  be  $\langle n_{k-1}, n_{k-2}, \dots, n_0 \rangle$ . We have  $k$  sorted arrays  $A_0, A_1, \dots, A_{k-1}$ , where for  $i = 0, 1, \dots, k-1$ , the length of array  $A_i$  is  $2^i$ . Each array is either full or empty, depending on whether  $n_i = 1$  or  $n_i = 0$ , respectively. The total number of elements held in all  $k$  arrays is therefore  $\sum_{i=0}^{k-1} n_i 2^i = n$ . Although each individual array is sorted, elements in different arrays bear no particular relationship to each other.
  - a. Describe how to perform the SEARCH operation for this data structure. Analyze its worstcase running time.
  - b. Describe how to perform the INSERT operation. Analyze its worst-case and amortized running times.
  - c. Discuss how to implement DELETE.
6. **Exercise 22.2–7** There are two types of professional wrestlers: “babyfaces” (“good guys”) and “heels” (“bad guys”). Between any pair of professional wrestlers, there may or may not be a rivalry. Suppose we have  $n$  professional wrestlers and we have a list of  $r$  pairs of wrestlers for which there are rivalries. Give an  $\mathcal{O}(n+r)$  time algorithm that determines whether it is possible to designate some of the wrestlers as babyfaces and the remainder as heels such that each rivalry is between a babyface and a heel. If it is possible to perform such a designation, your algorithm should produce it.
7. **Exercise 22.3–6** Show that in an undirected graph, classifying an edge  $(u, v)$  as a tree edge or a back edge according to whether  $(u, v)$  or  $(v, u)$  is encountered first during the depth-first search is equivalent to classifying it according to the ordering of the four types in the classification scheme

8. **Exercise 22.4–3** Give an algorithm that determines whether or not a given undirected graph  $G = (V, E)$  contains a cycle. Your algorithm should run in  $O(V)$  time, independent of  $|E|$ .
9. **Exercise 22.5–6** Given a directed graph  $G = (V, E)$ , explain how to create another graph  $G' = (V, E')$  such that (a)  $G'$  has the same strongly connected components as  $G$ , (b)  $G'$  has the same component graph as  $G$ , and (c)  $E'$  is as small as possible. Describe a fast algorithm to compute  $G'$ .
10. **Exercise 23.1–10** Given a graph  $G$  and a minimum spanning tree  $T$ , suppose that we decrease the weight of one of the edges in  $T$ . Show that  $T$  is still a minimum spanning tree for  $G$ . More formally, let  $T$  be a minimum spanning tree for  $G$  with edge weights given by weight function  $\omega$ . Choose one edge  $(x, y) \in T$  and a positive number  $k$ , and define the weight function  $\omega'$  by

$$\omega'(u, v) = \begin{cases} \omega(u, v) & \text{if } (u, v) \neq (x, y) \\ \omega(x, y) - k & \text{if } (u, v) = (x, y) \end{cases}$$

Show that  $T$  is a minimum spanning tree for  $G$  with edges weights given by  $\omega'$ .

11. **Exercise 23.2–4** Suppose that all edge weights in a graph are integers in the range from 1 to  $|V|$ . How fast can you make Kruskal's algorithm run? What if the edge weights are integers in the range from 1 to  $w$  for some constant  $w$ ?