Algorithms Middle Examination

Dec. 9, 2020 (10:10 ~ 12:20)

(Note that, if you design an algorithm, you must have pseudo-code to represent your algorithm. You can put comments after your pseudo-code to clarify your presentation.)

1. (12% Ch. 15) Design an O(N) algorithm to solve the following problem. Given an array A of length N, which consists of positive integers. A subset of A is called valid if any two elements in the subset are not adjacent in A. Among all valid subsets of A, find the one with the maximum sum. The algorithm only needs to return the sum, not the actual subset. For example, if $N = 7$ and $A = [3, 5, 3, 1, 2, 5, 1]$, the algorithm should return 11 since among all valid subsets, $\{A_1, A_3, A_6\}$ and $\{A_2, A_4, A_6\}$ both have the maximum sum 11. (Hint: This problem can be solved using dynamic programming. Define the state $dp[i]$ as the answer for the subarray $A[1] \dots A[i]$.)
2. (10% Ch. 15) Design an O($N + M$) algorithm to solve the following problem. Given an array A of length N, an array B of length M, and both consist of positive integers. Determine if B is a subsequence of A. A subsequence of an array is some array that can be formed by deleting zero or more elements from the array while maintaining the relative position of the remaining elements. For example, if $A = [3, 1, 4, 1]$ and $B = [3, 4, 1]$, the algorithm should return true. If $A = [3, 1, 4, 1]$ and $B = [4, 3]$, the algorithm should return false.
3. (10% Ch. 15) Consider a chessboard of size $k \times n$. How many ways are there to cover the chessboard completely with n rectangular bars, each of size $k \times 1$ or $1 \times k$? For instance, when $k = 2$, $n = 3$, there are three different ways: (a) cover the leftmost column by a vertical bar and the remaining region by two horizontal bars; (b) cover the rightmost column with a vertical bar, and the remaining region by two horizontal bars; or (c) cover each column with a vertical bar. Design an O(n)-time algorithm to compute the desired answer for any input k and n.
4. (10% Ch. 16) Suppose that in a 0-1 knapsack problem, the order of the items when sorted by increasing weight is the same as their order when sorted by decreasing value. Give an efficient algorithm to find an optimal solution to this variant of the knapsack problem, and argue that your algorithm is correct.
5. (10% Ch. 16) What is the optimal Huffman code for the following set of frequencies, based on the first 8 Fibonacci numbers? a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21. Can you generalize your answer to find the optimal code when the frequencies are the first n Fibonacci numbers?
6. (10% Ch. 17) Suppose we perform a sequence of n operations on a data structure where the ith operation costs i if i is an exact power of 2, and 1 otherwise.

a. (4%) Use aggregate analysis to determine the total amortized cost in Big-Oh notation.
b. (4%) Using the potential method, show that the potential function $\Phi(D_i) = i - 2^{\lfloor \lg i \rfloor}$ cannot yield useful result.
c. (2%) Show how to modify $\Phi(D_i)$ so that the potential method analysis gives the exact result.

7. (14% Ch. 17) Suppose that we wish to implement a dynamic table T with insertion and deletion operations. Let α be the load factor of the table T. Each insertion (deletion) operation will insert (delete) an item from the table T. If T is full before insertion of an item, we expand T by doubling its size. If T is bellow (1/4)-full after deletion, we contract T by halving its size. Let α_i denote the load factor of table T after the ith operation. Please answer the following question:

   a. (5%) What is the amortized cost of the ith operation if the ith operation is an insertion, $\alpha_{i-1} < 1/2$, and $\alpha_i < 1/2$?

   b. (5%) What is the amortized cost of the ith operation if the ith operation is a deletion and $\alpha_{i-1} > 1/2$?

   c. (4% Bonus) What is the amortized cost of the ith operation if $\alpha_{i-1} = 1/2$ and $\alpha_i < 1/2$?

8. (10%, Ch. 22) Given an undirected graph $G = (V, E)$, check if there is a way to color the nodes in red or blue such that no two adjacent nodes have the same color. The algorithm should run in $O(V + E)$-time. You have to specify the termination condition.

9. (10% Ch. 22) Give an algorithm that determines whether or not a given undirected graph $G = (V, E)$ contains a simple cycle. Your algorithm should run in $O(V)$ time, independent of $|E|$.

10. (10% Ch. 22) Prove that for any directed graph G, we have $((G^{T})^{SCC})^{T} = G^{SCC}$. That is, the transpose of the component graph of G^{T} is the same as the component graph of G.