COL351: Analysis and Design of Algorithms

Tutorial Sheet - 5

September 9, 2022

Question 1 Alice, Bob, and Charlie have decided to solve all exercises of the Algorithms Design book by Jon Kleinberg, Éva Tardos. There are a total of n chapters, $[1, \ldots, n]$, and for $i \in [1, n]$, x_i denotes the number of exercises in chapter i. It is given that the maximum number of questions in each chapter is bounded by the number of chapters in the book. Your task is to distribute the chapters among Alice, Bob, and Charlie so that each of them gets to solve nearly an equal number of questions. Design a polynomial time algorithm to partition $[1, \ldots, n]$ into three sets S_1, S_2, S_3 so that $\max\{\sum_{i \in S_1} x_i, \sum_{i \in S_2} x_i, \sum_{i \in S_3} x_i\}$ is minimized.

Question 2 Let $X = (x_1, \ldots, x_n)$ be a binary string of size n. Design an $O(n)$ time algorithm to compute the largest even integer $k \leqslant n$ that satisfies $X[1, k] \oplus X[n - k + 1, n] = (01)^{k/2}$.

Question 3 Let T be a tree with n nodes, and let C be a map from the nodes of T to positive integers such that node i has $C(i)$ coins attached with it. Design a polynomial time algorithm to compute a subset of nodes such that no two adjacent nodes (i.e. nodes connected directly by an edge) are chosen and the sum of the coins attached with nodes in the chosen subset is maximum.

Question 4 You are given set of k precious gemstones and a function $F : \mathbb{N} \to \mathbb{N}$ where $F(i)$ denotes the market price of a box of i gemstones (for $1 \leq i \leq k$). Your task is to find the optimal way to partition k gemstones into smaller sets that on selling provides you maximum overall profit.

Design an $O(k^2)$ time algorithm to output the optimal partitioning for k gemstones in a 'list'.

Question 5 You are given a set of k denominations, $d[1], \ldots, d[k]$. Example for Rs. 1, 2, 5, 10, 20, 50, 100, you have d(1) = 1, d(2) = 2, d(3)=5, d(4) = 10, d(5)=20, d(6)=50, and d(7)=100.

1. Design a polynomial time algorithm to count the number of ways to make change for Rs. n, given an infinite amount of coins/notes of denominations, $d[1], \ldots, d[k]$.
2. Design a polynomial time algorithm to find a change of Rs. n using the minimum number of coins (again you can assume you have an infinite amount of each denomination).