CSE421: Design and Analysis of Algorithms	May 9, 2019

Homework 5

Shayan Oveis Gharan — Due: May 16, 2019 at 5:00 PM

P1) Suppose you are working in the quality control of a factory. The factory produces quarters for the US government and your job is to make sure that all quarters have exactly the same weight. You are given 2^k quarters for $k \geq 2$ and you know that at most one of them can be defective. A defective quarter will weight *higher* or *lower* than normal. You are given a scale

with two trays: Each time you can put a set S of quarters in the left and a set T in the right (for disjoints sets S, T). The scale will show if S is heavier than T, or T is heavier than S, or they have exactly the same weight. Design an algorithm to find the defective quarter (if it exists) by using the scale only $O(k)$ many times. (Note that your algorithm will run by a human not a compute.)

P2) Let G be a graph with maximum degree k. Recall that a set S of vertices of G form an independent set if there is no edges between vertices of S. Design a polynomial time $O(k)$ approximation algorithm for the maximum independent set problem, i.e., the size of the independent set that your algorithm outputs must be at least $1/O(k)$ fraction of the optimum.

P3) Consider an array $a_1, \ldots, a_n$ of n integers, that is hidden from us. We have access to this array through an oracle knapsack(.,.). For a set $S \subseteq \{1, \ldots, n\}$ and an integer k, knapsack(S, k) will output "yes" if there is a subset $T \subseteq S$ such that the numbers indexed in T add up to k, i.e., $\sum_{i \in T} a_i = k$ and it will output "no" otherwise. Design an algorithm that calls knapsack only $O(n)$ times and outputs a set $S \subseteq \{1, \ldots, n\}$ such that the numbers indexed in S add up to k, if such a set exists.

For example, suppose $a_1 = 2, a_2 = 4, a_3 = 3, a_4 = 1$, and $k = 7$. Then, knapsack$(\{1, 2, 3, 4\}, 7)$ returns "yes" and knapsack$(\{1, 3, 4\}, 7)$ returns "no". In this case your algorithm can output either of the sets $\{1, 2, 4\}$ or $\{2, 3\}$.

P4) Draw the dynamic programming table of the following instance of the knapsack problem: You are 6 items with weight $1, 2, 3, 6, 7, 9$ and value $1, 3, 5, 12, 18, 25$ respectively and the size of your knapsack is 13..

P5) Suppose you are given n coins with value $v_1, \ldots, v_n$ dollars, and you want to change S dollars. You can assume $v_i \neq v_j$ for all $i \neq j$. Design a polynomial time algorithm that outputs the number of ways to change S dollars with the given n coins. For example, if for values $1, 2, 3, 4$ we can change 6 with in 2 ways as follows:

$$2 + 4, 1 + 2 + 3$$