ShanghaiTech University

CS240 Algorithm Design and Analysis

Spring 2019

Problem Set 5

Due: **1pm**, May 13, 2019

1. Submit your solutions to Gradescope (www.gradescope.com).
2. In "Account Settings" of Gradescope, set your FULL NAME to your Chinese name and enter your STUDENT ID correctly.
3. If you want to submit a handwritten version, scan it clearly. Camscanner is recommended.
4. When submitting your homework, match each of your solution to the corresponding problem number.

Problem 1:

In the game Geography on a Graph, we have a directed graph $G = (V, E)$ and a designated start node $s \in V$. Players alternate turns starting from s; each player must, if possible, follow an edge out of the current node to a node that has not been visited before. The player who loses is the first one who cannot move to a node that has not been visited. Prove that it is PSPACE-complete to decide whether the first player can force a win in Geography on a Graph. (In your proof, please draw the graph constructed in your reduction.)

Problem 2:

In the game Geography on a Graph, suppose the graph G has no directed cycles (i.e., G is a DAG). Now we can decide whether a player has a forced win in the game in polynomial time. Describe such an algorithm.

Problem 3:

We are given an undirected graph $G = (V, E)$ and costs $c(v)$ on the nodes $v \in V$. A subset $S \subseteq V$ is said to be a dominating set if all nodes $u \in V - S$ have an edge (u, v) to a node v in S. We want to minimize the sum of costs of nodes in the dominating set. Give a polynomial-time algorithm for this problem for the special case in which G is a tree.

Problem 4:

Find a 2-approximation *greedy* algorithm to solve the knapsack problem in polynomial time and prove its correctness. Suppose there are n items and the i-th item has weight w_i and value v_i. The total weight limit is B. Assume $\forall i, w_i \leq B$.

Problem 5:

Suppose that we are given a set of n objects. The size s_i of the ith object satisfies $0 < s_i < 1$. We wish to pack all the objects into the minimum number of unit-size bins. Each bin can hold any subset of the objects whose total size does not exceed 1. Give a polynomial-time 2-approximation algorithm to find the minimum number of bins required and prove its correctness.

Problem 6:

Consider the Max-Cut problem, which is the opposite of Min-Cut. Given a graph G, split its vertices into two sides to maximize the number of edges between the two sides. Give a polynomial-time 2-approximation randomized algorithm to find the Max-Cut (i.e., the expected cut produced by the algorithm is at least 1/2 times the optimal cut) and prove its correctness.