COMP4804 Assignment 4: Due Tuesday April 6th, 23:59EDT

Print this assignment and answer all questions in the boxes provided. Any text outside of the boxes will not be considered when marking your assignment.

1 Maximum-Area Independent Set of Squares

You are given a set $S = \{s_1, \ldots, s_n\}$ of axis-aligned squares. They are of different sizes and some of them overlap. The goal is to find a subset $S' \subseteq S$ such that the elements of S' are disjoint and their total area is maximized.

1. Consider the optimal solution S^* to this problem, and let s_i be some square in S. What is the maximum number of elements in S^* that are larger than s_i and that can overlap with s_i?

2. Describe a greedy approximation algorithm for solving this problem. State the algorithm and give an argument that it gives a constant factor approximation. [Hint: this problem is similar to the maximum-independent set of squares problem discussed in class.]

2 Self-Reducibility of Vertex-Cover

Suppose we have an algorithm $\mathcal{A}$ for Vertex-Cover that runs in $O(T(n,m))$ time on graphs with n vertices and m edges.[1] More precisely, the algorithm takes a graph G and an integer k and outputs *true* if G has a vertex cover of size k and *false* otherwise. *The algorithm doesn't output anything else.*

1. Using the above algorithm, describe an $O(T(n,m)\log n)$ time algorithm to determine the smallest value k' such that G has a vertex cover of size k'.

2. Describe how to improve the running time of the above algorithm to $O(T(n,m)\log k')$, where k' is the size of the smallest vertex cover of G.

3. Notice that the above algorithms don't tell us the actual vertices in the vertex cover, just whether or not one exists. Show how, if we already know some value k for which G has a vertex cover of size k we can find the vertices of a vertex cover of size k in $O(nT(n,m))$ time.

[1] Assume $T(n,m)$ is a non-decreasing function of both n and m.

3 A Fast Algorithm for Max-Clique in Graphs of Bounded Degree

In class we saw that the Clique decision problem is NP-complete. In this question you are asked to show that finding the largest clique in a graph G with n vertices can be done in $O(n)$ time if all vertices of G have constant degree.

1. Let G be a graph with maximum degree 3. Give the best upper bound you can come up with on the size of the largest clique in G. Show that your bound is optimal, i.e., if your bound is k then give an example of a graph with max-degree 3 that contains a clique of size k

2. Let G be a graph with maximum degree 3. Give an $O(n)$ time algorithm to find the largest clique in G.

3. Let G be a graph with maximum degree d. Give an $O(nd^2 2^d)$ time algorithm to find the largest clique in G.

4 NP-Hardness of and Algorithms for 3-Hitting-Set

The 3-Hitting-Set problem is the following: You are given a set $T = \{T_1, \ldots, T_n\}$ of n triples, where each triple $T_i = (a_i, b_i, c_i)$ consists of 3 integers in the set $\{1, \ldots, m\}$. A *hitting set* for T is a set of integers such that every triple in T contains at least one of the integers. For example the set $\{1, 2, 3\}$ is a hitting set for

$$\{(1, 2, 3), (3, 4, 7), (1, 5, 6), (2, 7, 8)\} \ .$$

1. Show that the decision problem: "Does T have a hitting set of size k?" is NP-complete. (Hint: it is a generalization of one of the NP-complete problems described in class.)

2. Give an algorithm for the 3-Hitting-Set decision problem that runs in $O(n3^k)$ time.

3. Give a fast 3-approximation algorithm for finding the smallest hitting set.

4. Using linear programming we can find real numbers $x_1, \ldots, x_m$, with $0 \leq x_i \leq 1$, such that

$$x_{a_i} + x_{b_i} + x_{c_i} \geq 1 \tag{1}$$

for all $1 \leq i \leq n$ and

$$\sum_{i=1}^{m} x_i$$

is minimized.

Given $x_1, \ldots, x_m$, describe how to find $x'_1, \ldots, x'_m$ that satisfy (1), such that each x'_i is either 0 or 1, and

$$\sum_{i=1}^{m} x'_i \leq 3 \sum_{i=1}^{m} x_i \ .$$

5. [Bonus!] Suppose that, independently, for each $i \in \{1, \ldots, m\}$, we set

$$x'_i = \begin{cases} 1 & \text{with probability } x_i \\ 0 & \text{with probability } 1 - x_i \end{cases}$$

For a particular triple $T_j = (a_j, b_j, c_j)$, give an upper-bound on the probability that $x'_{a_j} = x'_{b_j} = x'_{c_j} = 0$. [Hint: If $x, y, z > 0$ and $x + y + z \leq c$, then the product $xyz \leq (c/3)^3$]

6. [Bonus!] Suppose that the answer to your previous question was p. Then, give a randomized algorithm that produces a hitting set of expected size at most $k + pm$ where k is the size of the smallest hitting set for T.