Homework Assignment 2

Due: Thursday, May 15 in class.

General remark: whenever you are asked to provide an α-approximation algorithm, you need to prove that your algorithm outputs a feasible α-approximate solution.

1. **Metric k-cluster** In the metric k-cluster problem, we are given a complete undirected graph with non-negative weights w_e on edges satisfying the triangle inequality and an integer $k > 0$. The goal is to partition all vertices into k clusters $V_1, \ldots, V_k$, while minimizing the maximum distance of any pair of points in a cluster. So we seek to minimize the maximum, over all clusters V_i, of $\max_{u,v \in V_i} \left\{ w_{(u,v)} \right\}$.

   (a) Show a factor-2 approximation algorithm for the problem.

   (b) Prove that the problem is hard to approximate up to any factor smaller than 2.
   Hint: Reduction from 3-coloring.

2. **k-dispersion** The input to the k-dispersion problem is a complete undirected graph $G = (V, E)$ with weights $w_e \geq 0$ on edges $e \in E$, that obey the triangle inequality. Additionally, we are given an integer $k > 0$. The goal is to choose a subset $S \subseteq V$ of k vertices that are as far apart from each other as possible. Namely, we are trying to maximize $\min_{s,s' \in S} \left\{ w_{(s,s')} \right\}$.

   - Consider the following greedy algorithm: start with $S = \{v\}$ for an arbitrary node $v \in V$, and repeatedly add a node furthest from the nodes currently in S, until $|S| = k$. Show that this is a 2-approximation algorithm.
   - Prove that the problem is hard to approximate up to any factor smaller than 2.

3. **Multidimensional knapsack and bin packing**

   (a) In the multidimensional knapsack problem, the input is a set V of n non-negative integral d-dimensional vectors. Each vector $v_i \in V$ has a profit $w_i \geq 0$. Additionally, we are given a d-dimensional non-negative vector B (knapsack capacity). The goal is to find a maximum-profit subset $V' \subseteq V$ of vectors, whose sum is bounded by B coordinate-wise.
   Show a pseudo-polynomial time algorithm for multidimensional knapsack when d is a constant independent of n. What is the running time of your algorithm?

   (b) In the multidimensional bin-packing problem, we are given a set V of n non-negative d-dimensional vectors. The goal is to partition the vectors into minimum number of bins, such that the sum of vectors in every bin is bounded by 1 in every coordinate.
   Show an $O(d)$-approximation algorithm for multidimensional bin-packing.

4. **Maximum Independent Set of Disks** In this problem, we are given a set S of unit-diameter disks in the plane. The goal is to find a maximum-cardinality subset $S' \subseteq S$ of disks, such that no two disks in S' overlap. (For convenience, assume that the discs are open, so the disc boundary is not considered a part of the disc).

(a) Consider the following greedy algorithm: start with $S' = \emptyset$, and process the disks in S one-by-one, in any order. For each such disk C, if $S' \cup \{C\}$ is a feasible solution, then add C to S'. What is the approximation factor of this algorithm? (Give the best upper bound you can.)

(b) Let $\epsilon : 0 < \epsilon < 1$ be some constant. Assume that we are given a grid whose lines are spaced $\lceil 1/\epsilon \rceil$ units apart. Assume further that no disk in the input set S intersects the grid lines. Give an efficient algorithm for solving the problem exactly in this case.

(c) Assume that all input discs are contained in a bounding box of size $N \times N$, whose bottom left corner has coordinates $(0, 0)$. Let $k = \lceil 1/\epsilon \rceil$. For an integer $q : 0 \leq q < k$, a collection $L_0, L_1, L_2, \ldots, L_{\lfloor N/k \rfloor}$ of vertical lines is called a q-shifted collection of lines, iff for all $0 \leq i \leq \lfloor N/k \rfloor$, line L_i is the vertical line at the coordinate $ki + q$. Let OPT be any optimal solution to the problem. Prove that for some integral value $0 \leq q < k$, the corresponding q-shifted set of vertical lines intersects at most $|\mathsf{OPT}|\epsilon$ of the discs in OPT.

(d) Design a PTAS for the problem. What is the running time of your algorithm?