CS:5350 Final, Fall 2019

Due Mon, Dec 16th, 5pm

Notes: The exam is worth 200 points or 20% of your grade. The problems are each worth 50 points, though they will not necessarily take the same amount of effort and time. By submitting a solution you are vouching that all of the submitted work is completely yours, i.e., you have not collaborated with anyone and you have not used any online resources, except the ones mentioned below. As resources, you are welcome to use lecture notes posted on the CS:5350 webpage, the Williamson-Shmoys textbook, and any notes you have taken during lectures.

1. The input contains n points $p_1, p_2, \ldots, p_n$ in the plane, where each point p_i is specified by its coordinates (x_i, y_i). Additionally, the input contains m disks $D_1, D_2, \ldots, D_m$. The disks satisfy two properties: (i) they all have the same radius r and (ii) the centers of any two disks are at least r units away from each other. You can assume that the radius r is some constant and that each disk D_j is specified in the input by its coordinates. Finally, the points and disks together satisfy the property that every point is contained in some disk.

   The problem – let us call it DiskCover – is to find the fewest disks to cover all the points. More formally, we want to find the smallest set $C \subseteq \{1, 2, \ldots, m\}$ such that $p_i \in \cup_{j \in C} D_j$ for all i, $1 \leq i \leq n$.

   Design a constant-factor approximation algorithm for the DiskCover problem. Your answer should contain two parts: (a) an algorithm description, and (b) analysis showing that the algorithm is a constant-approximation to DiskCover.
   Note: The specific constant that you get for the approximation factor is not important, as long as you get a constant.
   Hint: This is a version of SetCover. Think about the different SetCover approximation algorithms we have discussed in class and use an algorithm that is likely to yield a constant-approximation.

2. Consider the MaxCut problem. Here the input is an undirected graph $G = (V, E)$ and the output is required to be partition of V into subsets U and W such that the number of edges that cross the cut (U, W) is maximized. In other words, we want to maximize the number of edges with one endpoint in U and the other in W.

   (a) In class, we have discussed a simple, randomized 1/2-approximation (in expectation) for MaxCut. Describe the deterministic 1/2-approximation algorithm you obtain by derandomizing this algorithm via the method of conditional expectations. Your answer just needs to contain an algorithm; there is no need to prove its correctness.

   (b) Now consider the following *greedy* algorithm. Suppose that the vertices are numbered $1, 2, \ldots, n$. In iteration 1, consider vertex 1 and place it in set U. In iteration k, consider vertex k and place it in U or W based on the following criterea. Look at F, the set of edges incident on k that have the other endpoint in $\{1, 2, \ldots, k-1\}$. Place k in U or W depending on which choice maximizes the number of edges in F being in the cut.

   Example: Suppose we have processed vertices 1, 2, 3, 4, and 5 and $U = \{1, 3\}$ and $W = \{2, 4, 5\}$. Suppose that vertex 6 has edges to vertices 1, 3, and 4. So the set $F = \{\{6, 1\}, \{6, 3\}, \{6, 4\}\}$. The greedy algorithm will place vertex 6 in W because this will make more of the edges in F cross the cut.

   Prove that the derandomized algorithm in (a) is equivalent to this greedy algorithm[1].

[1]**Side observation:** This implies that this greedy algorithm is a 1/2-approximation algorithm for MaxCut.

3. Suppose (for the moment) that we somehow know the radius ρ of an optimal solution to the k-Center problem. Given this optimal radius ρ, we can model the problem of finding k centers that yield this radius as the following integer program (IP). Recall that we are given n points $P = \{p_1, p_2, \ldots, p_n\}$ and $D : P \times P \to \mathbf{R}_{\geq 0}$ is a metric on P.

$$\begin{aligned} \sum_{i=1}^{n} x_i &= k & \\ \sum_{i : D(p_i, p_j) \leq \rho} x_i &\geq 1 & \text{for each } j = 1, 2, \ldots, n \\ x_i &\in \{0, 1\} & \text{for each } i = 1, 2, \ldots, n \end{aligned}$$

Here $x_i \in \{0, 1\}$, $1 \leq i \leq n$ is a choice variable indicating the selection of p_i as a center. Do not be thrown off by the fact that this integer program just has constraints and no objective function; it just means that any feasible solution, i.e., a solution satisfying these constraints, is good enough.

(a) Explain, in 2-3 sentences, how this integer program models the problem of finding k centers that realize the given radius ρ.

(b) Since we cannot solve this integer program in polynomial time, we replace the n integrality constraints $x_i \in \{0, 1\}$ by non-negativity constraints $x_i \geq 0$ and solve this LP relaxation. Let x_i^* denote the obtained solution to the LP relaxation.

Here is your task: Design a *deterministic* algorithm for rounding the x_i^* values so that the rounded values, which we will denote by $z_i \in \{0, 1\}$, satisfy the following constraints:

$$\begin{aligned} \sum_{i=1}^{n} z_i &= k & \\ \sum_{i : D(p_i, p_j) \leq 2\rho} z_i &\geq 1 & \text{for each } j = 1, 2, \ldots, n \\ z_i &\in \{0, 1\} & \text{for each } i = 1, 2, \ldots, n \end{aligned}$$

Note the use of "2ρ" in the second constraint above. This means that once we round the x_i^*'s to z_i's, we are allowing ourselves twice the radius to cover all points in the input. So this rounding algorithm gives us a set of k centers that form a 2-approximation to the k-Center problem.

Here are some suggestions for how to think about a rounding algorithm. Pick a point p_i with $0 < x_i^* < 1$. For example, x_i^* may be 0.3 telling us that p_i has been fractionally chosen to be a center. Consider all points $p_j \neq p_i$ such that $D(p_i, p_j) \leq \rho$. For each such point p_j, add x_j^* to x_i^* and then round down x_j^* to 0. In other words, we "transfer" x_j^* from p_j to p_i. Now think about the following two questions: (i) what is the value of x_i^* now? (ii) what should we do with all the points that p_j was helping cover as a "fractional" center?

(c) Finally, explain in 2-3 sentences how to get rid of the assumption that the optimal radius ρ is given to us. In other words, describe a polynomial time algorithm that allows us to generate all candidate values for the optimal radius ρ and consider each value one-by-one.

4. Consider the following optimization problem, let us call it BalancedSum. Given a set X of positive integers, our task is to partition X into disjoint subsets A and B such that

$$\max\left\{ \sum_{x \in A} x, \sum_{y \in B} y \right\}$$

is as small as possible. In other words, we want to partition X into two subsets A and B such that the sums of the elements in the two sets are as balanced, i.e., close to each other, as possible.

(a) Prove that the following algorithm yields a 3/2-approximation to BalancedSum.

```
GreedyPartition($X[1 \ldots n]$):
    $a \leftarrow 0$
    $b \leftarrow 0$
    for $i \leftarrow 1$ to $n$
        if $a < b$ then
            $a \leftarrow a + X[i]$
        else
            $b \leftarrow b + X[i]$
    return max{a, b}
```

(b) Give an example of an array X for which the cost of the output of GreedyPartition is 50% larger than the cost of the optimal partition.
