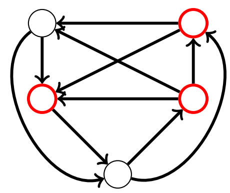
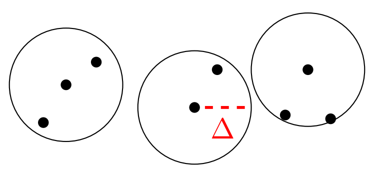
| | |
|---|---|
| **CSE421: Design and Analysis of Algorithms** | May 5th, 2020 |
| Homework 5 | |
| *Shayan Oveis Gharan* | Due: May 14, 2029 at 23:59 PM |

P1) A directed graph $G$ is a tournament if for any pair of vertices $u, v$ there is exactly one directed edge between $u$ and $v$, either from $u$ to $v$ or from $v$ to $u$. Given a tournament with $2^n$ vertices prove that it has a subtournament with at least $n+1$ vertices which is acyclic.

For example, in the following picture the red vertices show an acyclic sub-tournament of the given tournament.



P2) We want to design an $O(1)$ approximation algorithm for the following clustering problem. Given a set of $n$ points $p_1, \ldots, p_n \in \mathbb{R}^d$, and an integer $1 \leq k \leq n$, find the minimum radius $\Delta$ and a set of balls of radius $\Delta$ centered at $k$ of the given points such that all of the $n$ points lie in these balls. Note that the balls have radius $\Delta$ with respect to the Euclidean distance.




a) (15 points) Assume that we know the optimum radius $\Delta$. Design a polynomial time algorithm that finds at most $k$ balls of radius $O(\Delta$ centered at $k$ of the points covering all of the given points.

**Hint.** Recall the triangle inequality: For any triple points $a, b, c \in \mathbb{R}^d$, $\|a-c\|_2 \leq \|a-b\|_2 + \|b-c\|_2$.

b) (5 points) Now, assume that we do not know $\Delta$. Instead suppose we know that the optimum $\Delta$ is in the interval $[1, R]$[1]. Use part (a) to design an algorithm that runs in time polynomial in $n, \log R$ to find the $k$ balls of radius $O(\Delta)$.

P3) Recall that maximum independent set is NP-complete. Suppose a friend of yours is came up with an efficient algorithm to solve this problem. Unfortunately, their code does not output the independent set. Instead, for any graph $G$ and any integer $k$, if $G$ has an independent set of size $k$ it will output "yes" and "no" otherwise. Now, given a graph $G$ with $n$ vertices and an integer $1 \leq k \leq n$, design a polynomial time algorithm (that only runs their code

[1] In practice you can take $1, R$ correspond to be the minimum/maximum pairwise of all of the given points

polynomially many times) and outputs an independent set of size $k$ in $G$ if it exists, and outputs "Impossible" otherwise.

P4) Draw the dynamic programming table of the following instance of the knapsack problem: You are given 6 items with weight $1, 2, 4, 6, 7, 9$ and value $1, 3, 6, 12, 18, 25$ respectively and the size of your knapsack is 13.

P5) **Extra Credit:** Design an $O(\log n)$ approximation algorithm for weighted set cover problem, i.e., given $m$ sets $S_1, \ldots, S_m \subseteq \{1, \ldots, n\}$ such that $S_i$ has cost $c_i \geq 0$, choose a minimum cost family of these sets that cover all of the ground elements $\{1, \ldots, n\}$.