CS 580 Algorithm Design, Analysis, And Implementation
Fall 2021 Vassilis Zikas HW 6

Due Monday December 6 at 11:59 p.m.

Important Note: *You are **only allowed one late day** for this assignment i.e. last date to submit the assignment (with late penalty) is 11:59 pm Tuesday, December 7. Any submission made after 11:59 pm Monday, December 6 and before 11:59 pm Tuesday, December 7 will receive a 10% reduction in the grade.* **No submissions will be accepted/graded after 11:59 pm Tuesday, Dec. 7!**

1. (30 points) Consider the following clustering problem known as the k-center problem. The input is an undirected, complete graph $G = (V,E)$ with a distance metric $d(.,.)$ on the vertex set and an integer k. In this problem, the goal is to partition the graph's vertex set into k clusters each of which has a cluster center. We aim to *minimize the maximum distance* of a vertex to its cluster's center, call this distance the cost of the clustering.

   We assume that the distance metric has the following properties: symmetric $\forall i,j\ d(i,j) = d(j,i)$, follows triangle inequality $\forall i,j,k\ d(i,j) \leq d(i,k) + d(k,j)$, and $\forall i, d(i,i) = 0$.

   Consider the following greedy strategy to build a set S of k cluster centers. First, we pick one vertex arbitarily and include it in S. While size of S is less than k, we pick the vertex $v \in V$ that maximizes the following $\min_{j \in S} d(v,j)$, and add it to S. That is, we start with arbitrary vertex v_1. Then we pick the vertex v_2 that has the maximum distance from v_1. Then we pick the vertex v_3 that is the furthest from v_1 and v_2, i.e. that maximizes the minimum distance from v_1 and v_2, and so on. For example, consider the instance of k-center problem given below with $k = 3$ and the distances between any pair of points is the Euclidean distance between them. The optimal solution is $\{1^*, 2^*, 3^*\}$ but the greedy algorithm outputs $\{1,2,3\}$.

   Prove that the above algorithm returns a clustering (i.e. a set of cluster centres) of cost atmost two times the cost of any optimal clustering (optimal is the one that minimizes the maximum distance of any vertex to it's cluster center).

Answer: Let r^* be the optimal radius i.e. the minimum (over all possible cluster center sets) maximum distance of any vertex to its cluster center. Let $S^* = \{j_1, \ldots, j_k\}$ denote the optimal solution. We assign

each vertex to the closest centre in S^*. For $u \in V$ define S_u to be the set of all vertices in u's cluster i.e. $S_u = \{v \in V : u \text{ and } v \text{ are assigned to the same cluster center}\}$. Each pair of vertices in a cluster are at most $2r^*$ apart, by triangle inequality.

Consider the set $S \subseteq V$ obtained by running the given algorithm. If one centre in S is selected from each cluster of the optimal solution S^*. Then the radius is atmost $2r^*$ by the observation above.

Next, suppose that the greedy algorithm is run on the input instance and in some iteration the algorithm chooses a vertex u, such that another vertex $v \in S_u$ is already chosen. Again, the distance between u and v must be less than $2r^*$ and since the greedy algorithm selected v, the minimum distance from all other to a vertex in the set S must be less than $2r^*$. Therefore, the maximum radius in the future iteration is atmost $2r^*$.

Note that in the case that no vertex from a cluster (in the optimal clustering) is picked, then there must exist a cluster (in the optimal clustering) such that two vertices are chosen in it by the greedy algorithm.

2. (40 points) Suppose that A is an algorithm that outputs k cluster centres such that cost of clustering is at most 1.5 times the optimal cost for any instance of the k-center problem (defined in the previous question). Prove that existence of A implies a polynomial time algorithm for 3-SAT.

   Hint: It might be easier to prove that the existence of A implies a polynomial time algorithm for some other known NP-Hard problem. You can now use this to get an algorithm for 3-SAT.
   Answer: We reduce from the dominating set problem. Recall that in the dominating set problem, we are given a graph $G = (V, E)$ and an integer k, and we must decide if there exists a set $S \subseteq V$ of size k such that each vertex is either in S or adjacent to a vertex in S. Given an instance of the dominating set problem, we set distance of adjacent vertices 1 and and non adjacent vertices 2. Then, there is a dominating set of size k if and only if maximum distance of the k-centre instance is 1.

   First suppose that there is a dominating set of size k, then the vertices in the set become cluster centre and all vertices can be assigned to atleast one cluster centre. The max distance of any vertex to its cluster is thus 1. Similarly, given k clusters such that the maximum distance of a vertex from its centre is 1, we can form a dominating set of size k i.e. the cluster centres.

   If we can find a solution of size cD ($c < 2$) of the k-centre problem, then we can use this to find a dominating set of size k. This can be seen as follows. Suppose that we get a solution in which the maximum distance is $c \cdot 1$, then it implies that the maximum distance is 1 (since distances can only be integral), and we can recover a dominating set of size k as shown above. If the max distance in the solution returned using this is greater than or equal to 2, then the max distance of is 2 and no dominating set of size k exists.

3. (30 points) Consider the MAX-CUT problem defined as follows. The input is an undirected graph $G : (V, E)$ and the goal is to find a cut of maximum size. In other words, the goal is to find a partition of the vertex set into two parts U and $W = V \setminus U$ such that number of edges with one end point in U and other other in W is maximized.

   Consider the following algorithm: for each vertex $v \in V$, include it in U independently with probability $1/2$. Prove that we obtain a randomized $1/2$-approximation algorithm for the MAX-CUT problem.

   Recall that a randomized algorithm for a maximisation problem has approximation ratio $\alpha(n) \leq 1$, if for any instance of size n, the *expected* cost C of the returned solution and the optimal cost C^* satisfy $C \geq C^* \alpha$.

   Answer: For each edge e, define the indicator variable $X_e = 1$ if edge e is cut by the algorithm and 0 otherwise.

   Let S denote the size of the cut returned by the algorithm, i.e.

$$S = \sum_{e \in E} X_e$$

By linearity of expectation we get the following,

$$\begin{aligned} E[S] &= \sum_{e \in E} E[X_e] \\ &= \sum_{e \in E} \Pr[e \text{ is cut}] && \text{(Since } X_e \text{ is a indicator random variable)} \\ &= \sum_{e \in E} \frac{1}{2} = \frac{|E|}{2} \geq OPT/2 \end{aligned}$$

4. (2 points) Have you assigned pages in Gradescope?

5. **Bonus Problem.** (30 points, 10 each) Suppose that you are given an algorithm A that can analyze whether a person has a disease called Senioritis-580 by analyzing their DNA (assume that a person's DNA is a length n string). However, A is not always correct. Consider the following scenarios:

   (a) If a person has Senioritis-580 , then A outputs YES (has Senioritis-580) with probability $4/5$. However, if the person does not have Senioritis-580 , then A always outputs NO (does not have Senioritis-580). Give a polynomial time algorithm that has the following properties. For $p = 1 - 1/poly(n)$ where $poly(n)$ is a polynomial in n, if the person has Senioritis-580 , then it should output YES with probability atleast p. If the person does not have Senioritis-580 , then it should output NO with probability atleast p.

   (b) Now, suppose that if a person has Senioritis-580 , then A outputs YES with probability $q = 4/5$. However, if the person does not have Senioritis-580 , then A outputs NO with probability $q = 4/5$. Give a polynomial time algorithm that has the following properties. For $p = 1 - 1/poly(n)$ where $poly(n)$ is a polynomial in n, if the person has Senioritis-580 , then it should output YES with probability atleast p. If the person does not have Senioritis-580 , then it should output NO with probability atleast p.

   (c) In the previous scenario, prove or disprove: For all values of $q > 1/2$, there exists an algorithm that runs in polynomial time such that it is correct with probability atleast $3/4$ (i.e. $p = 3/4$).

   Assume that A runs in $O(n)$ time. Any polynomial time solution will be accepted for full credit in the above questions. **Answer:**

   (a) The proposed algorithm A' is as follows. We run A on the input person's DNA k independent times. If in any of these k independent iterations A outputs YES, then the algorithm outputs Yes, otherwise NO. Next, we calculate the error probabilities of A'. Given that the person has Senioritis-580 , A' outputs NO if in all the independent iterations A outputs NO. Therefore, the following holds:

$$\begin{aligned} \Pr[A' \text{ outputs NO}| \text{ has Senioritis-580 }] &= \prod_{i \in [k]} \Pr[A \text{ returns NO in iteration } i | \text{has Senioritis-580 }] \\ &=\leq \frac{1}{5^k} \end{aligned}$$

   Given that the person does not have Senioritis-580 , A' always outputs NO.
   Therefore, if $k \geq 2\log_5 n$, then p would be atleast $1 - 1/poly(n)$.

   (b) The proposed algorithm A' is as follows. We run A on the input person's DNA k independent times. A' returns Majority of A's returned responses. Next, we calculate the error probabilities of A'. Given that the person has Senioritis-580 , A' outputs NO if majority of the independent iterations of A output NO. By Chernoff bounds, the probabilities are bounded.

Define $X_i = 1$ if A outputs NO in iteration i if the person has Senioritis-580 . Then, A' outputs NO if $X = \sum_{i \in [k]} X_i > k/2$.

By linearity of expectation, $E[X] = k/5$.

Then, the following holds:

$$\begin{aligned} \Pr[X > k/2] &= \Pr[X - k/5 > 3k/10] \\ &\leq 10e^{-k/10}. \end{aligned}$$

Therefore, if $k \geq 20 \ln n$, then p would be atleast $1 - 1/poly(n)$.

(c) Consider the case when $q = 1/2 + 1/2^n$, in this case k needs to be atleast exponential for the error probabilities to be reduced to any constant.