Algorithms, Final, Dec 2022 (3 hours, total 40pts)

December 30, 2022

You can use the following NPC problems we mentioned in class or homeworks for your reductions: SAT, 3SAT, Vertex Cover, Set Cover, independent set, Clique, Subset Sum, Knapsack, 3-partition problem, 3 Dimensional Matching, Coloring, 3-Coloring, (directed or undirected) Hamitonian cycle/path, Traveling salesman problem, Dominating set problem, the max cut problem.

You can assume that (a poly-sized) linear program can be solved in poly-time.

Moreover, you can assume the min-cost matching (on bipartite or general graphs) and the min-cost flow problem (e.g., find a flow of value k and the cost of the flow is minimized) can be solved in poly-time.

We may need concentration inequalities like Chebyshev's inequality and/or Chernoff Bound.

(Chebyshev's inequality) Let X be a random variable with finite expected value μ and finite non-zero variance σ^2. Then for any real number $k > 0$,

$$\Pr(|X - \mu| \geq k\sigma) \leq \frac{1}{k^2}.$$

(Chernoff Bound) Let $X_1, \ldots, X_n$ be n i.i.d. random variables in $[0, 1]$. Let $\mu = \mathbb{E}[X_i]$. Then for any $\epsilon > 0$, we have that

$$\Pr\left[\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \geq \epsilon\right] \leq 2\exp\left(-\frac{n\epsilon^2}{2}\right).$$

Other versions should work equally well for this exam. You can use any of them.

1. (3 pts) Suppose there exists a trail with L miles and n stopping points on that trail. We assume that the stopping points are located at distances $x_1, x_2, \ldots, x_n$ from the start of the trail. Each day you can hike at most d miles and you must stop at a stopping point at that day if you haven't finish the trail. Try to design a polynomial algorithm to find the minimal days you need to go through the whole trail and prove the correctness of your algorithm. You can assume there always exists a feasible solution.

2. (3pts) (solve one of the following two problems, both are textbook exercises. you don't need to solve both) (1) Path selection problem: We are given a directed graph G and a set of directed paths $P_1, \ldots, P_c$ in G (P_is may share nodes or edges). The question is whether we can choose at least k of the paths such that no two of the selected path share any nodes. Show the problem is NPC.

   (2) In class, we learnt that finding k edge/vertex-disjoint paths from s to t can be solved using flow. However, the following variant turns out to be NPC. Given a directed graph G and k pairs of nodes $(s_1, t_1), \ldots, (s_k, t_k)$. The problem asks whether there exists k vertex-disjoint path $P_1, \ldots, P_k$ such that P_i goes from s_i to t_i. Prove the problem is NPC.

3. (4 pts) Let S be the set $\{1, 2, \ldots, mn\}$. We partition S into m sets $A_1, A_2, \ldots, A_m$ of size n. Let a second partitioning of S into m sets of size n be $B_1, B_2, \ldots, B_m$.

(a) Prove that the sets A_i can be renumbered such that $A_i \cap B_i \neq \emptyset$.

(b) Design a polynomial time that finds such a renumbering

4. (5 pts) (Inference-free schedule) We are given an undirected graph $G(V, E)$. We also have two robots A and B. Initially, A is located at node a and B is at b. We would like to find a schedule to move A to node c and move B to node d. At each time slot, a robot can move from one node to one of its neighbors in G or stays still. We also require that the schedule is *inference-free*: at any time slot, A and B must be at a distance at least r from each other. Give a polynomial time algorithm that finds a inference-free schedule. You can assume a and b are at a distance greater than r and so are c and d.

5. (5pts) Game of Local Optimum Suppose A is a $N \times N$ matrix containing N^2 different integers ranging from 1 to N^2. You can query $A(i, j)$ to get the number at row i and column j. In this problem, you need to run an algorithm to find a local optimum $(\hat{x}, \hat{y})$ in matrix A. A grid $(\hat{x}, \hat{y})$ for some $1 \leq \hat{x}, \hat{y} \leq N$ is called a local optimum if

$$A(\hat{x}, \hat{y}) < \min\{A(\hat{x}-1, \hat{y}), A(\hat{x}+1, \hat{y}), A(\hat{x}, \hat{y}-1), A(\hat{x}, \hat{y}+1)\}.$$

For convenience, you can simply assume $A(i, 0) = A(i, N+1) = A(0, i) = A(N, i) = +\infty$ for any $1 \leq i \leq N$. Now please answer the following questions.

(hint: there is no dependency between each questions, so you can take any order.)

(a) (2pt) Gradient descent is a popular method for finding local optima. Specifically, we can start from any coordinate (x_0, y_0) and, in each following step $t > 0$, find a neighboring grid (x_t, y_t) (i.e., $|x_t - x_{t-1}| + |y_t - y_{t-1}| = 1$) satisfying $A(x_t, y_t) < A(x_{t-1}, y_{t-1})$ (**If there are more than one such neighboring grids, choose an arbitrary one.**). We repeat this process until (x_t, y_t) becomes a local optimum.

Despite its simplicity, gradient descent may not be an efficient algorithm. Please construct a counter example such that the above gradient descent algorithm needs to query $O(N^2)$ grids to find a local optimum. For your convenience, you can simply assume we always start from $(x_0 = 1, y_0 = 1)$.

(b) (3pt) Propose an efficient algorithm that can find a local optimum in $o(N^2)$ time. (hint: In fact, only querying $O(N)$ grids is sufficient).

6. (5pts) There is a set of pages $P = \{1, 2, ..., n\}$ that can be broadcast by a server. Assume that time is discrete and there are T time slots. At each time slot, the broadcast server can broadcast exactly *one* page and all of the users could receive that page. There are several users. Each user u is associated with a page $p_u \in P$ and a time interval $[b_u, e_u]$. u can be satisfied if the server broadcasts page p_u any time during $[b_u, e_u]$.

Design an algorithm to schedule the broadcast server in order to satisfy as many users as possible. The problem is NP-hard (you do not have to prove it). Design a poly-time approximation algorithm that can achieve a constant factor approximation ratio.

$*$ Please answer 3 questions from the following 5 questions. If you answer more than 3 questions, we will give you scores according to the 5 problems for which you get the highest scores.

7. (5pts) (Evaluating an integral through sampling)

   (a) (2pts) Suppose we want to evaluate the value of $g = \int_0^1 f(x)dx$. You can assume that $f(x) \in [0,1]$ for all $x \in [0,1]$. However, $f(x)$ is not a function with closed-form formula, thus one cannot compute the integral using formulas we learnt in calculus class. Nevertheless, for each x, you can compute the value of $f(x)$ easily, say in $O(1)$ time. Now, we estimate the integral by

   $$\hat{g} = \frac{1}{n}\sum_{i=1}^{n} f(x_i),$$

   where x_i is an uniformly random chosen points from $[0,1]$. Argue why this is a good estimation. For fixed value ϵ and δ, show how many samples (in terms of ϵ and δ) we need to show that

   $$\Pr[|g - \hat{g}| \leq \epsilon] \geq 1 - \delta.$$

   (b) (1pts) The above problem can provide an estimation with additive error ϵ. However, for some applications, g can be extremely small (say 10^{-10}) and a usual additive error (say $\epsilon = 0.0001$) is not enough. In this case, we would like an estimation with multiplicative error. More precisely, for fixed valuex ϵ and δ, we would like an estimation $\hat{g}$ such that

   $$\Pr[|g - \hat{g}| \leq \epsilon g] \geq 1 - \delta.$$

   If we still use uniform sampling, show how many samples do we need. You can assume that you know a value v, which is a very coarse estimation of g such that $g/10 \leq v \leq 10g$ (in this case, the number of samples may depend on ϵ, δ and v^{-1}).

   (c) (2pts) Suppose we have another function $h(x)$, which is a very rough estimation of $f(x)$. In particular, we know that $f(x)/10 \leq h(x) \leq 10f(x)$ and we know the value $A = \int_0^1 h(x)dx$. Suppose we can take n i.i.d. samples x_i according to a probability density that is proportion to $h(x)$ (i.e., according to pdf $h(x)/A$). In this case, show that there is a method that can estimate g with multiplicative error ϵ (with probability $1-\delta$), and the number of samples n is polynomial in ϵ^{-1} and $\log 1/\delta$. You need to state what is your algorithm, why it is a good estimation, and the analysis of the number of samples. (sidenote: you can see from this problem that knowing a rough approximation of f can greatly reduce the sample complexity if you use a better sampling method).

8. For a finite set V of points in the Euclidean plane $\mathbb{R}^2$, the Voronoi diagram consists of regions

   $$P_v = \{x \in \mathbb{R}^2 \mid ||x - v||_2 = \min_{u \in V} ||x - u||\}$$

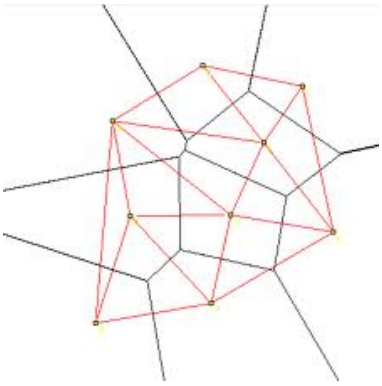
   for each $v \in V$. Essentially, P_v consists of all points that are closest to v. The Delaunay triangulation of V is the graph $D(V, E)$ where

   $$E = \{(u, v) \mid |P_v \cup P_u| > 1\}.$$

($|P_v \cup P_u| > 1$ means that the intersection of regions P_v and P_u is more than a single point.) See the figure above for an example: The black edges depict the boundaries of the regions in the Voronoi diagram while the red edges form the Delaunay triangulation. Delaunay triangulation is a planar graph (you do not need to prove this fact).

(a) (3pts) Show that a minimum spanning tree on V is a subgraph of $D(V, E)$.

(b) (2pts) Assume a Delaunay triangulation can be computed in $O(n \log n)$ time. Show that a minimum spanning tree can be also computed in $O(n \log n)$ time.



9. (Approximate Counting 0-1 KNAPSACK) For the 0-1 KNAPSACK problem, we are given a set of n objects of weight $0 \leq a_1 \leq a_2 \leq \cdots \leq a_n \leq b$, and the capacity C. All these numbers are positive integers.

   (1) (1 pts) We want to count the number N of the subset S such that $\sum_{i \in S} a_i \leq C$. Your algorithm can run in pseudopolynomial time (i.e., $\mathsf{poly}(n, C)$).

   (2) (1 pts) We would like to uniformly sample one solution from all possible solutions to $\sum_{i \in S} a_i \leq C$ (i.e., each subset S is sampled with probability $1/N$). Show how to do this in pseudopolynomial time.

   (3) (1 pts) We are given a very large set U. We can take samples uniformly from U. Suppose you want to estimate the cardinality of a subset $T \subset U$. For any element $u \in U$, you can determine whether $u \in T$ or not in $O(1)$ time. Suppose $|U|/|T| = \alpha$. Show that it is possible to estimate the cardinality of a subset T with ε relative error with probability $1 - \delta$ using $\mathrm{poly}(\alpha, \varepsilon^{-1}, \log \frac{1}{\delta})$ time.

   (4) (2 pts) Design a $\mathsf{poly}(n, \varepsilon^{-1}, \log \frac{1}{\delta})$ algorithm to approximate the number N with ε relative error with probability $1 - \delta$. (hint: first create a scaled problem as follows: Let $a'_i = \lfloor n^2 a_i / b \rfloor$. Let N' be the number of subset S such that $\sum_{i \in S} a'_i \leq n^2$. Show that $N' \leq (n+1)N$. For any solution in scaled problem, show that we can delete at most 1 element to meet the original constraint.)

10. (5pts) Consider a random bipartite graph $G(U, V; E)$ with $|U| = n$, $|V| = n$. For each pair of nodes u and v, $(u, v) \in E$ with probability $1/n$. Let $M(E)$ be the size of the maximum matching for $G(U, V; E)$. Show that

$$\mathbb{E}[M(E)] = \Omega(n).$$

(Note that E is a random set, so $\mathbb{E}[M(E)]$ makes sense).

11. (5pts) Consider the following stochastic matching problem. We have a random bipartite graph model where each possible edge e is present independently with some probability p_e. The probabilities p_e are given as input. Given these probabilities, we want to build a large matching in the randomly generated graph. However, the only way we can find out whether an edge is present or not is to query it, and if the edge is indeed present in the graph, we are forced to add it to our matching. Further, for each vertex i, we are allowed to query at most t_i edges incident on i (the numbers t_i is called the patience level which are also part of the input). Our goal is to design an adaptive query algorithm to maximize the expected size of the matching.

Our strategy is based on the following LP ($\delta(i)$ is the set of edges incident on vertex i):

$$\begin{aligned} &\max \sum_e x_e \\ s.t. &\sum_{e\in\delta(i)} y_e \le t_i, \quad \forall i \in V \\ &\sum_{e\in\delta(i)} x_e \le 1, \quad \forall i \in V \\ &x_e = p_e y_e, \quad \forall e \in E \\ &y_e \in [0,1]. \quad \forall e \in E. \end{aligned}$$

Intuitively, y_e represents the probability that the strategy probes edge e.

Our algorithm is as follows:

- First solve the LP and obtain the optimal solution (x_e, y_e) ($x_e = p_e y_e$).
- Pick a permutation π on edges uniformly at random.
- For each edge e in the ordering π, do the following: If e is not available (at least one of its endpoints is matched or has lost its patience, i.e., we have already probed t_i edges incident on that vertex i) then do not probe it. If e is still available then probe it with probability y_e/α where α is a constant which you can choose.

(a) (1pt) Note that this LP is somewhat different from the LP relaxation we studied in class. Here y_e is not the relaxation of some integer variable. Prove that the optimal LP solution is an upper bound of the expected matching size obtained by any optimal strategy.

(b) (4pt) Prove that this algorithm achieves a constant factor approximation (using appropriate constant α). (Hint: use a somewhat large constant α. Let I_e be the event that e is available when we consider to probe it in the random permutation π. Try to show $\Pr[I_e]$ is at least some constant).

Note that even if you do not know how to formally prove (a), you can still try to answer (b).