CS218 Design and analysis of algorithms **Jan-Apr 2024**

Exercises: Linear Programming, Approximation, Randomized, Error correction

1. Prove that the following algorithm gives 2-approximation for minimum size vertex cover. That is, the set S output by the algorithm is a vertex cover and its size is at most twice of the optimal vertex cover.

$S \leftarrow$ empty set.

While the graph is non-empty

choose an edge (u, v) and put both its endpoints in S

delete u and v and all their incident edges from the graph

delete isolated vertices

Observe that the edges chosen during the algorithm form a matching in the given graph. Prove that this is an 1/2-approximation for maximum matching. That is, the matching obtained has size at least half of the maximum size matching.

2. Consider the following (approximation) algorithm for minimum size vertex cover. Construct examples to show that the approximation factor is not bounded by any constant, that is, the approximation factor increases with the input size.

$S \leftarrow$ empty set.

While the graph is non-empty

choose a vertex u with the highest degree and put it in S

delete u and all its incident edges from the graph

delete isolated vertices

---

3. Recall the linear program we wrote for minimum weight vertex cover problem. Let OPT be the weight of the minimum weight vertex cover. Let LP-OPT be the optimal value of the corresponding linear program. Let ALG-VC be the weight of the vertex cover obtained after rounding the LP optimal solution. Recall that in the rounding procedure, we selected a vertex v if and only if we had $x_v \geq 1/2$ in the LP optimal solution. Prove that
   - LP-OPT $\leq$ OPT $\leq$ ALG-VC.
   - ALG-VC $\leq 2 \times$ LP-OPT
4. **Maximum weight matching:** Given a graph with edge weights, the goal is to find a matching (set of disjoint edges) with maximum total weight. Write an integer linear program for the maximum weight matching problem. Now, remove the integer constraint, that is, variables are allowed to take any real value. We get a linear program. Find an example (a graph with edge weights), where the optimal value of the linear program is higher than the weight of the maximum weight matching. Interestingly, if the graph is bipartite then the two values are always equal.
5. Recall the greedy algorithm for minimum makespan problem. Prove that if we go over the job in decreasing order of processing times, then the greedy algorithm gives a 3/2-approximate solution.

   Do you think your analysis is tight? Do you see an example, where the solution obtained is indeed 3/2 times the optimal?

6. (Line fitting.) Given n points $p_1, p_2, \ldots, p_n\mathbb{R}^d$, with labels $\ell_1, \ell_2, \ldots, \ell_n \in \mathbb{R}$, we want to compute a linear function that best fits with the points and labels. More precisely, find a function $h(x) = a_1x_1 + a_2x_2 + \cdots + a_dx_d + b$ so that we minimize the error function $E(h)$ defined as

$$E(h) = \max_{1 \le j \le n} \{|h(p_j) - \ell_j|\}.$$

Write a linear program for this.

7. (Curve fitting.) Given n points $p_1, p_2, \ldots, p_n\mathbb{R}^d$, with labels $\ell_1, \ell_2, \ldots, \ell_n \in \mathbb{R}$, we want to compute a quadratic function that best fits with the points and labels. More precisely, find a function $h(x) = \sum_{1 \le i \le j \le d} a_{i,j}x_ix_j$ so that we minimize the error function $E(h)$ defined as

$$E(h) = \max_{1 \le j \le n} \{|h(p_j) - \ell_j|\}.$$

Write a linear program for finding h.

8. (Classification.) Given n points $p_1, p_2, \ldots, p_n\mathbb{R}^d$, which are labeled either positive or negative, we want to compute a linear function that best fits with the points and labels. More precisely, find a function $h(x) = a_1x_1 + a_2x_2 + \cdots + a_dx_d + b$ so that we minimize the hing loss $L(h)$ defined as follows.

For a point p_j, if it's label is positive then we expect $h(p_j)$ to be (significantly) more than zero. Let's say we expect $h(p_j)$ to be at least 1. If that is not true then we consider the difference from 1 as the loss. Define loss with respect to a positively labeled point p_j as

$$L(h, p_j) \begin{cases} = 1 - h(p_j) & \text{if } h(p_j) < 1 \\ = 0 & \text{otherwise.} \end{cases}$$

Similarly, define loss with respect to a negatively labeled point p_k as

$$L(h, p_k) \begin{cases} = h(p_k) + 1 & \text{if } h(p_j) > -1 \\ = 0 & \text{otherwise.} \end{cases}$$

Finally we define the hinge loss $L(h)$ over all points as $\sum_j L(h, p_j)$. Write a linear program to find h that minimizes $L(h)$.

9. Consider random variables $X_1, X_2, \ldots, X_n$ such that for each $1 \le i \le n$, $X_i = 1$ with probability $1/i$ and 0 with probability $1 - 1/i$. What is the expectation of X_i? What is the expectation of $X = \sum_{i=1}^{n} X_i$?

10. Recall the randomized algorithm for 2-dimensional linear programming from the class. We showed that the expected number of times we need to compute intersections is less than $2n$. Prove that this number is less than $10n$ with probability at least $4/5$.

11. Instead of choosing a random permutation, suppose we simply choose a random number i between 1 and n. And then we follow the order $i, i+1, i+2, \ldots, n-1, n, 1, 2, \ldots, i-1$. Do you think the analysis of the expected running time will still work?

12. Can you design a similar randomized algorithm for 3-dimensional linear programming? In 2D case, the crucial property we used was that at any corner at most 2 lines will intersect (by the way, is it really true?).

Is such a statement true in the 3D case? If not, can we somehow perturb the input constraints so that only 3 planes intersect at a corner? Assuming this is true, what will be the probability that when we introduce the ith constraint the optimal point does not change. In case, the optimal point changes, how much time will be required to update the optimal point.

13. (Simplex algorithm simulation) Consider the following linear program.

$$\begin{aligned} \max 2x_1 + x_2 \text{ subject to} \\ x_1 &\geq 0 \\ x_2 &\geq 0 \\ x_1 &\leq 1 \\ x_1 + x_2 &\leq 2 \end{aligned}$$

- First write the LP in the standard form where we have (two) equations and (four) non-negativity constraints $x_1 \geq 0, x_2 \geq 0, x_3 \geq 0, x_4 \geq 0$.
- Start with the basic feasible solution that has $x_1 = 0, x_2 = 0$. What will be the values of x_3, x_4? At any point in the algorithm, the two variables which are zero are called the *non-basic variables* and the remaining are called *basic variables*. For example, in the beginning, x_3, x_4 are basic variables and x_1, x_2 are non-basic.
- Run the iterations of the simplex algorithm till you get an optimal solution. After each iteration write down (i) the basic feasible solution and (ii) express the objective function in terms of the current non-basic variables (i.e., which are zero). In an iteration, there may be multiple possible choices for which variable to increase. You can just make an arbitrary choice.

Below is how an iteration of simplex algorithm runs.

- Choose one of the non-basic variables to increase. It should be among those whose coefficient in the objective function is positive. If there is no such variable then output the current solution. If there is such a variable then increase it till the maximum possible value while maintaining feasibility. The other non-basic variables (except the chosen one) should remain zero in this iteration.
- This gives you a new basic feasible solution and a new set of basic and non-basic variables
- Express the objective function in terms of the non-basic variables.
- Express the basic variables in terms on non-basic variables.

14. Prove that when the simplex algorithm stops, that is, when we express the objective function in terms of non-basic variables and all coefficients turn out to be negative, then we are at an optimal solution. What is the optimal value at this point?

15. (Initial basic feasible solution.) We said that finding initial basic feasible solution itself is a challenging problem. We will solve it by framing it as another linear program. Suppose the given linear program (LP1) is

$$\begin{aligned} a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n &= b_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n &= b_2 \\ &\vdots \\ a_{k,1}x_1 + a_{k,2}x_2 + \cdots + a_{k,n}x_n &= b_k \\ x_1, x_2, \ldots, x_n &\geq 0. \end{aligned}$$

Here $a_{i,j}$s and b_is are given as input and x_js are unknowns. Without loss of generality, we can assume that $b_1, b_2, \ldots, b_k \geq 0$. Because otherwise we can simply multiply -1 on both the sides of the equation.

We write the following new linear program (LP2).

$$\begin{aligned} \min z_1 + z_2 + \cdots + z_k \text{ subject to} \\ a_{1,1}x_1 + a_{1,2}x_2 + \cdots + a_{1,n}x_n - b_1 = z_1 \\ a_{2,1}x_1 + a_{2,2}x_2 + \cdots + a_{2,n}x_n - b_2 = z_2 \\ \vdots \\ a_{k,1}x_1 + a_{k,2}x_2 + \cdots + a_{k,n}x_n - b_k = z_k \\ x_1, x_2, \ldots, x_n \geq 0. \\ z_1, z_2, \ldots, z_k \geq 0. \end{aligned}$$

Here z_is and x_js are unknowns.

Prove that LP1 has a feasible solution if and only if LP2 has an optimal value =0.

There is a trivial initial basic feasible solution for LP2, which is $z_1 = b_1, z_2 = b_2, \ldots, z_k = b_k$ and $x_1 = x_2 = \cdots = x_n = 0$.

16. Prove that modulo a prime number p, every nonzero number $\pmod p$ has a multiplicative inverse. Hint: Euclid's gcd algorithm.

    For example, modulo 11, the inverses are as follows.

a	a^{-1}
1	1
2	6
3	4
4	3
5	9
6	2
7	8
8	7
9	5
10	10

17. Prove that a degree d polynomial has at most d roots over reals/rationals/complex numbers. Prove that a degree d polynomial has at most d roots even when working with modulo a prime number. For example, if we are working modulo 7, then the polynomial $x^3 - 1$ has roots $1, 2, 4$ (here we have only $0, 1, 2, 3, 4, 5, 6$ as the field elements). As another example over modulo 7, the polynomial $x^4 - x + 1$ has no roots.

    Hint: induction on degree. Assume one root α and argue that $x - \alpha$ should divide the polynomial.

18. Prove that for any $d + 1$ points in a 2 dimensional space, there is a unique degree d curve passing through them.

19. Consider 4 distinct points in the $\mathbb{R}^2$: (0,3), (1, 2), (2, 2), (5, 1). What is the unique degree 3 curve (of the form $y = ax^3 + bx^2 + cx + d$) passing through these 4 points.

    Hint: Lagrange interpolation

20. Suppose we have 36 distinct points in the 2 D plane. We consider the unique degree 35 curve C passing through these 36 points. Consider 64 other distinct points on this curve, which makes total 100 points. Suppose 32 out of 100 points are modified arbitrarily (in the y coordinate). Still the curve C will pass through at least 68 points. Argue that there is no other curve of degree 35 that passes through some 68 points out of 100 points.