# Midterm Exam

Due: 2025-4-10 11:59 | 4 Questions, 100+10 Pts

Name: Jiacong Fang, ID: 2200017849

## Problem 1 (10')

For today's dinner, you are making a decision between two restaurants, *Yannan* and *Shaoyuan*. You know that one of them is a strictly better choice under all circumstances, and everyone prefers it to the other. However, you are the only person in the world who has no idea *which* is the better one. You decide to ask some folks on `treehole.pku.edu.cn` to give scores for the two restaurants. According to the following responses, which one will you choose?

- Alice gives a score of 84 for *Yannan*.
- Bob gives a score of 3.92 for *Shaoyuan*.
- Carol gives a score of 99999 for *Yannan*.
- Dave gives a score of 666 for *Shaoyuan*.
- Eve gives a score of $-\pi/6$ for *Yannan*.
- ...

At first glance, it seems impossible that one can do better than random guessing. However, we are going to show that this is not the case.
Formally, consider two score sequences $(x_1, \cdots, x_n), (y_1, \cdots, y_n) \in D^n$, where $D \subseteq \mathbb{R}$, such that either of the two cases is true:

- Case 0: $x_i < y_i,\ \forall i \in [n]$.
- Case 1: $x_i > y_i,\ \forall i \in [n]$.

We need to design a potentially randomized algorithm $\mathcal{A}$, whose output is either 0 or 1, to distinguish the two cases. Notably, we are restricted to "half-blind" algorithms: For each $i \in [n]$, algorithm $\mathcal{A}$ can only query one of the two values of $\{x_i, y_i\}$; once it decided to query one value, the other one in the $i$'th pair becomes inaccessible.

a. (3') Suppose $n = 1$ and $D = \mathbb{R}$. Design a half-blind algorithm $\mathcal{A}_1$ such that for any real numbers $x, y$,

$$\Pr[\ \mathcal{A}_1 \text{ is correct} \mid x_1 = x,\ y_1 = y\ ] > \frac{1}{2}.$$

b. (7') Suppose $D = [0, 100]$, $|x_i - y_i| \geq 1$. Design a half-blind algorithm $\mathcal{A}$ for sufficiently large $n$, and show that it has $1 - 1/2^{p(n)}$ success probability given any feasible input, where $p(n) = \Omega(n)$.

**Direction:** *For each $i$, your half-blind algorithms can decide freely which one of $\{x_i, y_i\}$ to observe. However, it can only observe one of the two values. You should give proofs for the error bounds.*

**Answer.** a). Consider the following algorithm $\mathcal{A}_1$:

**Algorithm 1** Half-bind Algorithm $\mathcal{A}_1$

**Input:** $x_1, y_1 \in \mathbb{R}$
**Output:** Case 1 ($x_1 > y_1$) or Case 0 ($x_1 < y_1$).

Randomly query $x_1$ or $y_1$ with equal probability
**if** $x_1$ is queried **then**
    Randomly sample $z_x \sim \mathcal{N}(0,1)$
    **if** $x_1 > z_x$ **then**
        Output Case 1 ($x_1 > y_1$).
    **else**
        Output Case 0 ($x_1 < y_1$).
**else**
    Randomly sample $z_y \sim \mathcal{N}(0,1)$
    **if** $y_1 > z_y$ **then**
        Output Case 0 ($x_1 < y_1$).
    **else**
        Output Case 1 ($x_1 > y_1$).

For any real numbers $x, y$, without loss of generatelity, we assume $x > y$ here.

$$\begin{aligned}\Pr[\ \mathcal{A}_1 \text{ is correct} \mid x_1 = x,\ y_1 = y\ ] &= \frac{1}{2}\left(\Pr_{z_x\sim\mathcal{N}(0,1)}[x > z_x] + \Pr_{z_y\sim\mathcal{N}(0,1)}[y < z_y]\right) \\ &= \frac{1}{2}\left(\Phi(x) + 1 - \Phi(y)\right) > \frac{1}{2}.\end{aligned}$$

where $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. The last inequality holds because $\Phi(x) - \Phi(y) > 0$ when $x > y$.

b). For any $i \in [n]$, we first define the following algorithm $\mathcal{A}_i$:

**Algorithm 2** Half-bind Algorithm $\mathcal{A}_i$

**Input:** $x_i, y_i \in [0, 100]$
**Output:** Case 1 ($x_i > y_i$) or Case 0 ($x_i < y_i$).

Randomly query $x_i$ or $y_i$ with equal probability
**if** $x_i$ is queried **then**
    Randomly sample $z_x \sim \mathcal{U}(0, 100)$ ▷ Uniform distribution over $[0, 100]$
    **if** $x_i > z_x$ **then**
        Output Case 1 ($x_i > y_i$).
    **else**
        Output Case 0 ($x_i < y_i$).
**else**
    Randomly sample $z_y \sim \mathcal{U}(0, 100)$
    **if** $y_i > z_y$ **then**
        Output Case 0 ($x_i < y_i$).
    **else**
        Output Case 1 ($x_i > y_i$).

Then we can design the following algorithm $\mathcal{A}$:

**Algorithm 3** Half-bind Algorithm $\mathcal{A}$

**Input:** $(x_1, \cdots, x_n), (y_1, \cdots, y_n) \in [0, 100]^n$
**Output:** Case 1 ($x_i > y_i), \forall i \in [n]$ or Case 0 ($x_i < y_i), \forall i \in [n]$.

$\text{num}_1 \leftarrow 0,\ \text{num}_0 \leftarrow 0$ ▷ Count of Case 1 and Case 0
**for** $i = 1$ to $n$ **do**
    Run Algorithm $\mathcal{A}_i$ on $(x_i, y_i)$
    **if** Algorithm $\mathcal{A}_i$ outputs Case 1 **then**
        $\text{num}_1 \leftarrow \text{num}_1 + 1$
    **else**
        $\text{num}_0 \leftarrow \text{num}_0 + 1$
**if** $\text{num}_1 > n/2$ **then**
    Output Case 1 $(x_i > y_i), \forall i \in [n]$.
**else**
    Output Case 0 $(x_i < y_i), \forall i \in [n]$.

**Error Bounds Analysis:** Without loss of generality, we assume $x_i > y_i$ for all $i \in [n]$ and $n = 2k + 1$ for $k \in \mathbb{N}^*$. (When $n$ is even, we can simply ignore $x_n$ and $y_n$ and only run $\mathcal{A}$ on the first $n - 1$ pairs, it won't affect the error bounds to much.)

For $i \in [n]$, let random variable $X_i := \mathbf{1}[\mathcal{A}_i \text{ outputs } x_i > y_i]$. We first analyse Algorithm $\mathcal{A}_i$, similar to the analysis in part a, we have:

$$\begin{aligned}\Pr[\ \mathcal{A}_i \text{ is correct } | x_i, y_i] &= \frac{1}{2}\left(\Pr_{z_x \sim \mathcal{U}(0,100)}[x_i > z_x] + \Pr_{z_y \sim \mathcal{U}(0,100)}[y_i < z_y]\right) \\ &= \frac{1}{2}\left(1 + \frac{|x_i - y_i|}{100}\right) \geq \frac{1}{2} + \frac{1}{200} = \frac{101}{200}.\end{aligned}$$

Thus, we have $\mathbb{E}[X_i] \geq 101/200$. Therefore, the probability that $\mathcal{A}$ makes a mistake can be bounded as follows:

$$\begin{aligned}\Pr[\mathcal{A} \text{ is wrong }] &= \Pr\left[\sum_{i=1}^{n} X_i \leq \frac{n}{2}\right] = \Pr\left[\frac{1}{n}\sum_{i=1}^{n} X_i - \frac{101}{200} \leq -\frac{1}{200}\right] \\ &\leq \Pr\left[\frac{1}{n}\sum_{i=1}^{n} X_i - \mathbb{E}[X_i] \leq -\frac{1}{200}\right] \\ &\leq \exp\left(-2n \cdot \frac{1}{200^2}\right) \quad \text{(Chernoff Bound)} \\ &\leq \frac{1}{2^{p(n)}} \quad \text{(for sufficiently large } n, \text{ where } p(n) = \Omega(n)).\end{aligned}$$

Then we can conclude that:

$$\Pr[\mathcal{A} \text{ is correct }] = 1 - \Pr[\mathcal{A} \text{ is wrong }] \geq 1 - \frac{1}{2^{p(n)}}. \tag{1}$$

where $p(n) = \Omega(n)$.

**Note:** The error bound can be analysed by **Median Trick**. Each run of $\mathcal{A}_i$ is a Bernoulli trial with success probability $p_i$ greater than 101/200. Then

$$\Pr[\mathcal{A}_i \text{ success}] \geq \frac{101}{200} \implies \Pr\left[\#\{\mathcal{A}_i \text{ success in } \mathcal{A}\} \leq \frac{n}{2}\right] \leq \left(\frac{101}{200}\right)^{n/2}$$

Thus, (1) holds for $p(n) = \Omega(n)$ with sufficiently large $n$.
**Q.E.D.**

◁

## Problem 2 (15') (Knapsack Counting)

Recall the 0-1 Knapsack problem you have encountered before. We are given $n$ objects of weight $0 \le a_1 \le \cdots \le a_n \le b$ where $b$ is the total volume of the knapsack, and we are asked to find a 0-1 assignment $x_1, \cdots, x_n$ such that $\sum_{i=1}^n a_i x_i \le b$. In the counting version, we will compute the number of valid assignments, the set of which is denoted as $S$, instead of finding the optimal solution. The key idea here is similar to #DNF; that is, sampling from a restricted space. Suppose $b$ is much larger than $n^2$.

We will use the *scaling method*. Define $a'_i := \lfloor n^2 a_i / b \rfloor$, then $0 \le a'_i \le n^2$. Note the floor operator may cause error when trying to recover $a_i$ from $a'_i$, but the total error is no more than $n(b/n^2) = b/n$. Let $S' := \left\{ \boldsymbol{x} : \sum_{i=1}^n a'_i x_i \le n^2 \right\}$ be the set of solutions to the rescaled problem.

a. (2') Prove that $S \subseteq S'$.

Apparently, the converse does not necessarily hold. However, we claim that any $\boldsymbol{x} \in S'$ can be shoehorned into $S$ by setting at most one of the $x_i$ from 1 to 0.

b. (5') Prove the claim above, which implies $|S'| \le (n+1)|S|$.

In other words, with probability at least $1/(n+1)$, a uniformly random element of $S'$ is in $S$. So the remaining task here is to compute $|S'|$ and sample from $S'$. Let $C(k, m) := \left| \left\{ \boldsymbol{x} : \sum_{i=1}^k a'_i x_i \le m \right\} \right|$.

c. (3') Design an algorithm based on dynamic programming that computes $C(i, j)$ for all $0 \le i \le n$ and $0 \le j \le n^2$. Your algorithm should take $O(n^3)$ time.

Definitely, we can fetch the value of $|S'|$ from $C(n, n^2)$. But in fact, we can also use this table to sample uniformly from $S'$.

d. (5') Based on the values of $C(\cdot, \cdot)$, design an algorithm which outputs a uniform sample of $S'$. Your algorithm should run in $O(n)$ time (excluding the time cost by the random generator).

**Answer.** a). For any $\boldsymbol{x} = (x_1, x_2, \cdots, x_n) \in S$, we have

$$\sum_{i=1}^n a_i x_i \le b \iff \sum_{i=1}^n \frac{n^2 a_i}{b} x_i \le n^2$$

And notice that

$$\sum_{i=1}^n a'_i x_i = \sum_{i=1}^n \left\lfloor \frac{n^2 a_i}{b} \right\rfloor x_i \le \sum_{i=1}^n \frac{n^2 a_i}{b} x_i \le n^2$$

Then we have $\boldsymbol{x} \in S'$ which implies $S \subseteq S'$.

b). Define $k$ be s.t. $a_j \le b/n$ for $j \le k$ and either $k = n$ or $a_{k+1} > b/n$. Let $C = \{0,1\}^k \times \{0\}^{n-k}$. If $x \in C$ then

$$\sum_{i=1}^n a_i x_i \le \sum_{i=1}^k a_i x_i \le \frac{bk}{n} \le n \implies C \subseteq S$$

For any $\boldsymbol{x} \in S' \setminus S$, then there exists $p = p(x) \in \mathbb{N}^*$, s.t. $x_p = 1$ and $p \notin [k]$ (Otherwise $x \in C \subseteq S$ which is a contradiction). Then the sheohorn(or a mapping) from $S'$ to $S$ is defined as:

$$f : S' \to \{0,1\}^n$$
$$f(\boldsymbol{x}) \mapsto \begin{cases} \boldsymbol{x} & \text{if } x \in S \\ \boldsymbol{y} & \text{if } x \in S' \setminus S \end{cases}$$
$$\text{where } y_j = x_j, \forall j \ne p \text{ and } y_p = 0.$$

We now prove that for any $\boldsymbol{x} \in S' \setminus S$, $\boldsymbol{y} = f(\boldsymbol{x}) \in S$. Let $\delta_i = a_i - a'_i$, then we have

$$\begin{aligned}\sum_{i=1}^{n} a_i y_i &= \frac{b}{n^2}\sum_{i=1}^{n}(a'_i + \delta_i)y_i = \frac{b}{n^2}\left(\sum_{i=1}^{n} a'_i y_i + \sum_{i=1}^{n}\delta_i y_i\right) \\ &= \frac{b}{n^2}\left(\sum_{i=1}^{n} a'_i x_i - a'_p + \sum_{i=1}^{n}\delta_i y_i\right) \\ &\leq \frac{b}{n^2}(n^2 - n + n) = b.\end{aligned}$$

The last inequality holds because $\delta \in [0, 1)$ and $p \notin [k] \implies a_p > b/n \implies a'_p \geq n$. Therefore, $\boldsymbol{y} \in S$ which implies $|S'| \leq (n+1)|S|$.

c). The dynamic programming algorithm is as follows:

**Algorithm 4** Dynamic Programming for $|S'|$

**Input:** $a'_1, a'_2, \cdots, a'_n \in [0, n^2]$
**Output:** $C(i, j)$ for all $0 \leq i \leq n$ and $0 \leq j \leq n^2$.

$\forall 0 \leq j \leq n^2$, initialize $C(\cdot, \cdot)$ with:

$$C(1, j) = \begin{cases} 2 & \text{if } j \geq a'_1 \\ 1 & \text{otherwise} \end{cases}$$

**for** $i = 2$ to $n$ **do** ▷ Compute the table $C$ using following recursion.
  **for** $j = 0$ to $n^2$ **do**
    **if** $j < a'_i$ **then**
      $C(i, j) = C(i-1, j)$
    **else**
      $C(i, j) = C(i-1, j) + C(i-1, j - a'_i)$

The time complexity of the algorithm is $O(n^3)$.

d). We can determine a uniform sample from $S'$ by tracing back probabilistically from $C(n, n^2)$ as following:

**Algorithm 5** Uniform sampling from $S'$

**Input:** $C(i, j)$ for all $0 \leq i \leq n$ and $0 \leq j \leq n^2$.
**Output:** A uniform sample from $S'$.

$j \leftarrow n^2$
**for** $i = n$ to $2$ **do**
  $x_i \leftarrow 0$ with prob. $C(i-1, j)/C(i, j)$ else $x_i \leftarrow 1$ with the remaining prob.
  **if** $x_i = 1$ **then**
    $j \leftarrow j - a'_i$ ▷ Update the remaining capacity
  **else**
    $j \leftarrow j$
**if** $C(1, j) = 2$ **then** ▷ Handle the special case when $i = 1$
  $x_1 \leftarrow 0$ and $x_1 \leftarrow 1$ with equal prob.
**else**
  $x_1 \leftarrow 0$
**Output** $(x_1, x_2, \cdots, x_n)$.

The time complexity of the algorithm is $O(n)$ beacuse each iteration of the for loop takes $O(1)$ time, and the loop runs $n$ times.

**Q.E.D.** ◁

## Problem 3 (50') (Low Diameter Decomposition)

In this problem, you'll learn Low Diameter Decomposition, and use it to give approximate algorithm for some fundamental graph problem like Tree Embedding, All Pair Shortest Path (APSP).
In this problem, the graph can be seen as weighted graph with all the weight being positive integer.

**Definition 1** (Low Diameter Decomposition (LDD))**.** *Given an undirected graph $G = (V, E)$, a **Low Diameter Decomposition (LDD)** scheme with approximation factor $\beta$ and diameter bound $D$ is a randomized algorithm that partitions $V$ into disjoint clusters $V_1, V_2, \ldots, V_k$ satisfying:*

1. ***Bounded Diameter**: For each $V_i$, the diameter of $G[V_i] \leq D$.*
2. ***Separation Probability**: For any $x, y \in V$,*

$$\Pr\left[x \text{ and } y \text{ lie in different clusters}\right] \leq \beta \cdot \frac{d_G(x,y)}{D},$$

*where $d_G(x, y)$ denotes the shortest-path distance between $x$ and $y$ in $G$.*

**Remarks**:

- The **diameter** of $G[V_i]$ is $\max_{u,v \in V_i} d_G(u, v)$.
- $\beta$ balances cluster tightness ($D$) and separation likelihood. Lower $\beta$ implies better decomposition quality.

a. (10') Prove that the following algorithm gives a LDD with approximation factor $\beta = O(\log n)$, with probability $\geq 1 - n^{-1}$.

**Algorithm 6** Low Diameter Decomposition (LDD)

**Input:** Undirected graph $G = (V, E)$, target diameter $D > 0$
**Output:** Vertex partition $\{V_1, \ldots, V_k\}$ with $\text{diam}(G[V_i]) \leq D$

Initialize $marked[v] \leftarrow$ FALSE for all $v \in V$
**while** $\exists v \in V$ with $\neg marked[v]$ **do**
  Select arbitrary unmarked vertex $v_0 \in V$
  Sample $R_{v_0} \sim \text{Geometric}(p)$ with $p = \min\left(1, \frac{4 \log_e n}{D}\right)$.
  Compute $B \leftarrow \{u \in V \mid \neg marked[u] \wedge d_G(v_0, u) \leq R_{v_0}\}$
  Create cluster $C \leftarrow B$
  $marked[u] \leftarrow$ TRUE for all $u \in C$
  Add $C$ to output partition
**return** the computed clustering

**Remarks**: The naive way to run LDD takes time $O(n^3)$, since one need to run Dijsktra for $n$ times. However, there are ways to do LDD in $\tilde{O}(n^2)$ time. You will get 10 Bonus Point if you can find out.

We will use this tool to solve approximate APSP problem. First, we will introduce Low Stretch Tree.

**Definition 2.** *A randomized low-stretch tree of stretch $\alpha$ for a graph $G = (V, E)$ is a probability distribution $\mathcal{D}$ over spanning trees of $G$ s.t.*

1. *$d_G(x, y) \leq d_T(x, y)$, for all $T$ in the support $\mathcal{D}$.*
2. *$\mathbb{E}_{T \sim \mathcal{D}}[d_T(x, y)] \leq \alpha \cdot d_G(x, y)$, $\forall x, y \in V$*

**Theorem 3.** *For any metric space $M = (V, d)$, there exists an efficiently sampleable $\alpha_B$-stretch spanning tree distribution $\mathcal{D}_B$, where*

$$\alpha_B = O(\log n \log \Delta_M)$$

*$\Delta_M$ is defined as $\max_{x,y} d(x, y)$, we assume $\forall x \neq y, d(x, y) \geq 1$.*

We will prove theorem 3 by the following algorithm.

**Algorithm 7** Low Stretch Tree Construction, $\mathrm{LST}(M, \delta)$

**Input:** Metric space $M = (V, d)$, target diameter $D = 2^\delta$
**Output:** Spanning tree $T$ with low stretch. **Invariant**: $\mathrm{diameter}(M) \leq 2^\delta$

```
if |V| = 1 then
    return trivial tree containing the single point
Partition V into clusters C_1, ..., C_t ← LDD(M, D/2)
for j = 1 to t do
    Let M_j be M restricted to C_j
    Recursively build T_j ← LST(M_j, δ − 1)
Connect roots r_2, ..., r_t to r_1 with edges of length 2^δ
return final tree T rooted at r_1
```

**Lemma 4.** *If the random tree $T$ returned by some call LST$(M, \delta)$ has root $r$, then*

1. *every vertex $x$ in $T$ has distance $d_T(x, r) \leq 2^{\delta+1}$*
2. *the expected distance between any $x, y \in T$ has $\mathbb{E}[d_T(x, y)] \leq 8\delta\beta d(x, y)$. (Recall that $\beta$ is the approximate factor in LDD)*

If we can prove Lemma 4, then one can see from Problem a that if $\beta = O(\log n)$, then with high probability, $d_T(x, y) \geq d(x, y)$ for any $x, y$, thus theorem 3 can be proved.

b. (20') Prove the Lemma 4.

c. (10') Prove the following theorem.

**Theorem 5.** *There's an algorithm that output $O(\log n \log \Delta)$ approximation of APSP on an undirected graph in $\tilde{O}(n^2)$ time, and success with probability $\geq 1 - \frac{1}{poly(n)}$.*

(This means, the algorithm output $d'(x, y)$ for any pair $x, y$, and satisfies $d(x, y) \leq d'(x, y) \leq \log n \log \Delta \cdot d(x, y)$, where $d(x, y)$ is the length of shortest path between $x, y$. )

d. (10') Prove the following theorem.

**Theorem 6.** *There's an algorithm that output $O(\log n)$ approximation of ASAP on an undirected graph in $\tilde{O}(n^2)$ time, and success with probability $\geq 1 - \frac{1}{poly(n)}$.*

**Answer.** a). We first prove the diameter bound by showing that $R_{v_0} \le D/2$ with high probability and then using the triangle inequality to bound the diameter of cluster $V_i$, i.e. for any $u, v \in V_i$, we have

$$d_G(u,v) \le d_G(u,v_0) + d_G(v_0,v) \le \frac{D}{2} + \frac{D}{2} = D.$$

Now compute the probability that $R_{v_0} > D/2$ for a given cluster $V_i$:

$$\Pr[R_{v_0} > D/2] = (1-p)^{D/2} \le e^{-pD/2} \le e^{-2\log n} = \frac{1}{n^2}.$$

Here we use $1 - x \le e^x$ for all $x \in \mathbb{R}$. Then by union bound, we have

$$\begin{aligned}\Pr\left[\exists \text{a cluster with diameter} > D\right] &= 1 - \Pr\left[\exists v \in V, R_{v_0} > D/2\right] \\ &\ge 1 - \frac{n}{n^2} = 1 - \frac{1}{n}.\end{aligned}$$

Then we bound the separation probability using the memoryless property of geometric distribution. For any $x, y \in V$, assume the center vertex picked is $v$. Sampling $R_v$ from Geometric($p$) can be seen as flip a coin of bias $p$ until we get a head. Without loss of generality, assume $x$ will lie in the current ball first. At this time, we have flipped $d_G(v,x)$ coins without seeing a head.

Then the case that $x$ and $y$ lie in different clusters will happen if we see a head within the next $d_G(v,y) - d_G(v,x) \le d_G(x,y)$ flips. Therefore,

$$\begin{aligned}\Pr[x, y \text{ lie in different clusters}] &= \Pr[R_v < d_G(v,y) | R_v \ge d_G(v,x)] \\ &= 1 - \Pr[R_v \ge d_G(v,y) | R_v \ge d_G(v,x)] \\ &= 1 - \Pr[R_v \ge d_G(v,y) - d_G(v,x)] \\ &= 1 - (1-p)^{d_G(v,y) - d_G(v,x)} \\ (d_G(v,y) - d_G(v,x) \le d_G(x,y)) \quad &\le 1 - (1-p)^{d_G(x,y)} \\ (\forall p \in [0,1], (1-p)^x \ge 1 - px) \quad &\le 1 - (1 - p \cdot d_G(x,y)) = p \cdot d_G(x,y) \\ &\le 4\log n \cdot \frac{d_G(x,y)}{D} = O(\log n) \cdot \frac{d_G(x,y)}{D}.\end{aligned}$$

Notice that if $(x,y) \notin E$, we can repeat the above argument along the shortest path from $x$ to $y$ in $G$, and the results still hold.

**Bonus: Improve the time complexity to $\tilde{O}(n^2)$.**

The basic idea is reducing redundant queries to the same vertex (i.e. line 5 of Algorithm 6).

**Algorithm 8** Improved Low Diameter Decomposition (LDD) with $\tilde{O}(n^2)$ time complexity

**Input:** Undirected graph $G = (V, E)$, target diameter $D > 0$
**Output:** Vertex partition $\{V_1, \ldots, V_k\}$ with $\mathrm{diam}(G[V_i]) \le D$

1: Initialize $marked[v] \leftarrow$ FALSE for all $v \in V$
2: **while** $\exists v \in V$ with $\neg marked[v]$ **do**
3: Select arbitrary unmarked vertex $v_0 \in V$
4: Sample $R_{v_0} \sim$ Geometric($p$) with $p = \min\left(1, \frac{4\log_e n}{D}\right)$
5: $marked[v_0] \leftarrow$ TRUE
6: **Initialize an empty priority queue $Q$ and add $v_0$ with distance 0.** ▷ Use BFS
7: **while** $Q$ is not empty **do**
8: **Pop the vertex $u$ with the smallest distance $d_u$ from $Q$**
9: **if** $\neg marked[u]$ **then**
10: $marked[u] \leftarrow$ TRUE

11: Add $u$ to cluster $C$
12: **for** each neighbor $w$ of $u$ **do**
13: **if** $\neg marked[w]$ and $d_G(v_0, w) + d_u \leq R_{v_0}$ **then**
14: **Add $w$ to $Q$ with distance $d_G(v_0, w) + d_u$**
15: **return** the computed clustering

**Time complexity**: notice that each vertex will be added and poped from the priority queue only once. And in line 12, each vertex will be queried by its neighbors at most $\deg(u)$ times and $\sum_{v\in V} \deg(v) = O(m) = O(n^2)$. We use heap-based priority queue whose time complexity of poping and adding is $O(\log n)$.

Therefore, the total time complexity of the algorithm is $O(m+n\log n) = O(n^2+n\log n) = \tilde{O}(n^2)$.

b). We will prove the lemma by induction on $\delta$.

**Base case**: when $\delta = 0$, we have $|V| = 1$ and the tree is a single vertex. There is nothing to prove. Assume the lemma holds for $\delta - 1$ and consider the case $\delta$ as follows:

**Induction step**: We prove the fisrt part of the lemma first. Consider $x$ lies in cluster $C_i$ (denote the corresponding tree as $T_i$) and by the induction hypothesis, $d_{T_i}(x, r_i) \leq 2^\delta$. And the distance to the new root $r$ is at most $2^\delta$ more. $d_T(x, r) \leq d_{T_i}(x, r_i) + 2^\delta \leq 2^{\delta+1}$. So the first part of the lemma holds.

Now we prove the second part of the lemma. According to the part(a), we know that for any $x, y$,

$$\Pr[x, y \text{ is separated }] \leq \beta \cdot \frac{d_G(x, y)}{2^{\delta-1}}$$

And if $x$ and $y$ are in the same cluster(denote as $T_i$), by the induction hypothesis, we have

$$\mathbb{E}[d_{T_i}(x, y)] \leq 8\beta(\delta - 1)d_G(x, y).$$

Therefore, the expected distance between $x$ and $y$ is

$$\begin{aligned}
\mathbb{E}[d_T(x, y)] &\leq \Pr[x, y \text{ is separated }] \cdot (d_T(x, r) + d_T(y, r)) \\
&\quad + \Pr[x, y \text{ is not separated }] \cdot d_{T_i}(x, y) \\
&\leq \beta \cdot \frac{d_G(x, y)}{2^{\delta-1}} \cdot \left(2^{\delta+1} + 2^{\delta+1}\right) + 8\beta(\delta - 1)d_G(x, y) \\
&= 8\delta\beta \cdot d_G(x, y).
\end{aligned}$$

Therefore, the lemma holds for all $\delta$. Then we use the lemma to prove the theorem 3.

When $\beta = O(\log n)$, we first prove that for $\mathcal{D}_B$ over spanning trees, $d_G(x, y) \leq d_T(x, y)$ for any $T$ in $\text{supp}(\mathcal{D}_B)$. Notice that $d_G(x, y) \leq \Delta_M = 2^\delta$ here. Fix $x$ and $y$, and let $i$ such that $d_G(x, y) \in (2^{i-1}, 2^i]$. Then consider the invocation of $\text{LST}(U, i)$ such that $x \in U$.

- If $y \in U$, then by the definition of LDD, $x, y$ will fall into separate clusters with high probability. Therefore, by the procedure of LST, $d_T(x, y) \geq 2^i$, which is the length of the edge connecting different subtrees.
- If $y \notin U$, then $x$ and $y$ must be separated at a higher level $i' > i$ during the recursion of LST, hence $d_T(x, y) \geq 2^{i'} > 2^i$.

In sum, $d_G(x, y) \leq d_T(x, y)$ for any $T$ in $\text{supp}(\mathcal{D}_B)$.

And due to $\delta = \log(\Delta_M)$, then by the lemma 4, we have

$$\mathop{\mathbb{E}}_{T\sim\mathcal{D}_B}[d_T(x, y)] \leq 8\delta\beta d_G(x, y) = O(\log n \log \Delta_M)d_G(x, y). \forall x, y \in V.$$

which concludes the proof of theorem 3.

c). We have proven that theorem 3 holds for any metric space $M$. Then the **Algorithm 6** is an approximate algorithm for APSP with factor $O(\log n \log \Delta)$ where $\Delta := \max_{x,y} d(x,y)$. The correctness is guaranteed by theorem 3.

**Time complexity**: We use the improved LDD algorithm whose time complexity is $\tilde{O}(n^2)$. Denote the total time complexity of the algorithm as $T(n)$, then

$$T(n) = \tilde{O}(n^2) + \sum_{i=1}^{t} T(n_i) \text{ where } \sum_{i=1}^{t} n_i = n$$

which implies that each level of recursion takes at most $\sum_{i=1}^{t} \tilde{O}(n_i^2) = \tilde{O}(n^2)$ time.

Notice that the depth of recursion is at most $\lceil \log \Delta_M \rceil + 1$. Therefore, the total time complexity is $\tilde{O}(n^2 \log \Delta_M) = \tilde{O}(n^2)$.

d). We try to improve the Algorithm 7 from $\alpha_B = O(\log n \log \Delta_M)$ to $\alpha_B = O(\log n)$ by introducing a better graph decomposition method called **CKR Decomposition** [1].

**Algorithm 9** CKR Decomposition

**Input:** Undirected graph $G = (V, E)$, distance metric $d$ and $D$
**Output:** Vertex partition $\{V_1, \ldots, V_k\}$.

1: Randomly sample $R \sim \text{Uniform}[D/4, D/2]$.
2: Choose a random permutation $\pi : V \to V$ uniformly at random.
3: Initialize $marked[v] \leftarrow \textsc{False}$ for all $v \in V$
4: **for** $v_0$ in the order given by $\pi$ **do**
5: **if** $marked[v_0]$ **then**
6: continue ▷ Skip if already covered by previous clusters
7: Compute $C \leftarrow \{u \in V \mid \neg marked[u] \wedge d_G(v_0, u) \le R_{v_0}\}$
8: $marked[u] \leftarrow \textsc{True}$ for all $u \in C$
9: Add $C$ to output partition
10: **return** the computed clustering

Similar with Algorithm 6, we can implement CKR Decomposition in $\tilde{O}(n^2)$ time with priority queue instead of the naive Dijkstra algorithm [3].

Then we try to prove that CKR Decomposition has **Bounded Diameter** and even better **Separation Probability** than LDD.

Bounded diameter is satisfies by the definition of CKR Decomposition directly(see line 1), i.e. we have $\forall C_i$, there exists $c_i \in C_i$ such that $d_G(c_i, u) \le D/2$ for all $u \in C_i$ which implies that $d_G(u, v) \le D$ for all $u, v \in C_i$.

Then for any $u, v \in V$, reorder the vertexs as $w_1, w_2, \cdots, w_n$ according to $\min\{d(w_i, u), d(w_i, v)\}$. Without loss of generality, denote $d(w_j, u) := a_j$, $d(w_j, v) := b_j$ and assume $a_j < b_j$. Then the case that $u$ is in $w_j$'s cluster and $v$ is not need to satisfy (1) $R \in [a_j, b_j]$, (2) $w_j$ appears before $w_1, \cdots, w_{j-1}$ in the order given by $\pi$ (Otherwise, $u$ will be assigned cluster before $w_j$'s). Thus,

$$\begin{aligned}
\Pr[u, v \text{ is separated}] &\le \sum_{i=1}^{n} \Pr[u \text{ is in } w_i\text{'s cluster and } v \text{ is not}] \\
&\le \sum_{i=1}^{n} \Pr[R \in [a_i, b_i]] \cdot \Pr[w_i \text{ appears before } w_1, \cdots, w_{i-1}] \\
\text{(triangle inequality.)} &\le \sum_{i=1}^{n} \frac{d_G(u,v)}{D/2 - D/4} \cdot \frac{1}{i} = \frac{d_G(u,v)}{D/4} \cdot \sum_{i=1}^{n} \frac{1}{i} \\
&= O(\log n) \frac{d_G(u,v)}{D}.
\end{aligned}$$

In order to achieve $O(\log n)$ approximation in LST, we can analyze the probability more carefully to get a better bound[1].

$$\Pr[u, v \text{ is separated}] \leq \frac{d_G(u,v)}{D/4} \cdot \log\left(\frac{\#B(u,D)}{\#B(u,D/8)}\right) \tag{2}$$

where $B(x,r) := \{y : d_G(x,y) \leq r\}$. The proof idea is that suppose $d_G(u,v) < D/8$ (Otherwise, the probability of separability is at most $8d_G(u,v)/D$ which is greater than 1 already.) Assume $u$ is closer to $w_j$ than $v$, then $d_G(u,w_i) \leq D$ and $d_G(u,w_i) \geq D/4 - d_G(u,v) \geq D/8$. Combine the new restriction and two requirements above, we can get the better bound.

Then we replace the LDD in Algorithm 7 with CKR Decomposition and get a new algorithm for APSP, and we can prove $\alpha_B = O(\log n)$ here.

The first part, i.e. $d_G(x,y) \leq d_T(x,y)$, for all $T$ in the support $\mathcal{D}$. The proof is almost the same as the one in part (b) of this problem (see the last few lines in page 9, marked in blue).

Then we prove the following theorem:

**Theorem 7** ([2]). *Replace the LDD in Algorithm 7 with CKR Decomposition, then for all $x, y \in V$, we have*

$$\mathop{\mathbb{E}}_{T\sim\mathcal{D}}[d_T(x,y)] \leq O(\log n) \cdot d_G(x,y)$$

The proof is similar to the one in part (b) of this problem, here we only show the most important part. Using induction we have: (level $i$ means $\mathrm{LST}(M,i)$)

$$\begin{aligned}
\mathop{\mathbb{E}}_{T\sim\mathcal{D}}[d_T(x,y)] &= \sum_{i=\delta}^{0} \Pr[x,y \text{ are separated at level } i] \cdot \left(2^{i+1} + 2^{i+1}\right) \\
\text{(Apply (2))} &\leq \sum_{i=\delta}^{0} \frac{d_G(u,v)}{2^{i-1}/4} \cdot 4 \cdot 2^i \cdot \log\left(\frac{\#B(u,2^{i-1})}{\#B(u,2^{i-4})}\right) \\
&= 32 \cdot d_G(x,y) \sum_{i=\delta}^{0} \log\big(\#B(u,2^{i-1})\big) - \log\big(\#B(u,2^{i-4})\big) \\
&= 32 \cdot d_G(x,y) \sum_{i=\delta}^{\delta-2} \log\big(\#B(u,2^{i-1})\big) \\
&\leq 96 \cdot d_G(x,y) \log n
\end{aligned}$$

where the last inequality is due to the fact that $\#B(u,2^{i-1}) \leq n$ for all $i$.

In sum, theorem 7 holds. Therefore, the LST algorithm with CKR Decomposition can achieve $O(\log n)$ approximation for APSP. And due to the **Algorithm 9 (CKR Decomposition)** can be implemented in $\tilde{O}(n^2)$ time just like LDD, the total time complexity of the algorithm is $\tilde{O}(n^2)$ as well.

**Q.E.D.** ◁

[1]The detailed analysis is interesting but hard. Here I refer many materials like [1], [3] and notes of CMU course.

## Problem 4 (25') (Sub-Network Approximation for Two-Layer Neural Networks)

**Definition 8** (Two-Layer Neural Network)**.** *Consider a wide neural network $f_M(x;\theta) : \mathbb{R}^d \to \mathbb{R}$ of the form*

$$f_M(x;\theta) := \sum_{j=1}^{M} a_j \sigma(w_j^\top x),$$

*where $\theta = \{a_1, \ldots, a_M, w_1, \ldots, w_M\}$ denotes the parameters of the network, $a_j \in \mathbb{R} \setminus \{0\}$ are the non-zero coefficients, and $w_j \in \mathbb{R}^d$ are the weights associated with the $j$-th neuron. Here, $\sigma : \mathbb{R} \to \mathbb{R}_{\geq 0}$ denotes ReLU activation function that is defined as $\sigma(\cdot) = \max\{0, \cdot\}$.*

**Definition 9** (Parameter Norm)**.** *The norm of the parameters $\theta$ is defined as*

$$\|\theta\|_{\mathcal{P}} := \sum_{j=1}^{M} |a_j| \|w_j\|_2,$$

*where $\|w_j\|_2$ denotes the Euclidean norm of the weight vector $w_j$, and $\|\theta\|_{\mathcal{P}}$ is assumed to be finite.*

Let $\pi = \mathrm{Unif}(\mathbb{S}^{d-1})$ denote the uniform distribution over the unit sphere $\mathbb{S}^{d-1} \subset \mathbb{R}^d$. The goal is to approximate the network $f_M(x;\theta)$ with a sparse subset of the neurons, while controlling the error in expectation. Indeed, please prove the following result:

**Theorem 10.** *For any integer $m \in \mathbb{N}$, there exists a sparse vector $\tilde{a} = (\tilde{a}_1, \ldots, \tilde{a}_M)$ such that $\|\tilde{a}\|_0 = m$, i.e., the vector $\tilde{a}$ contains exactly $m$ non-zero entries, which satisfies the following approximation bound:*

$$\mathop{\mathbb{E}}_{x\sim\pi} \left( \sum_{j=1}^{M} \tilde{a}_j \sigma(w_j^\top x) - \sum_{j=1}^{M} a_j \sigma(w_j^\top x) \right)^2 \lesssim \frac{\|\theta\|_{\mathcal{P}}^2}{dm}.$$

**Remark 11.** *In essence, this result demonstrates that it is possible to approximate the network with a sparse representation without incurring significant loss in performance, provided the number of selected neurons $m$ is sufficiently large relative to the network's parameter norm $\|\theta\|_{\mathcal{P}}$, and the size of the input space $d$.*

**Answer.** We try to sample $\tilde{a}$ and construct an unbiased estimator whose expectation is $f_M(x;\theta)$. For $j \in [M]$, we define the following probability mass function (Denote as distribution $\mathcal{D}_M$).

$$p_j = \frac{|a_j| \cdot \|w_j\|_2}{\|\theta\|_{\mathcal{P}}} \quad \text{and} \quad \sum_{j=1}^{M} p_j = 1.$$

Then we can sample $m$ neurons from $M$ neurons with $\{p_j\}_{j=1}^M$ independently, i.e.,

$$\text{sample } j_1, j_2, \cdots, j_m \overset{i.i.d}{\sim} \mathcal{D}_M.$$

Then we can define the following estimator, for $t \in [m]$,

$$Y_t(x) := \frac{\mathrm{sign}(a_{j_t})}{\|w_{j_t}\|_2} \cdot \sigma(w_{j_t}^\top x) \quad \text{and } \tilde{f}(x) := c \cdot \sum_{t=1}^{m} Y_t(x).$$

where $c$ is a constant waiting to be determined to makes sure that $\tilde{f}(x)$ is unbiased. Then we can assign $\tilde{a}$ as follows:

$$\tilde{a}_i = c \cdot \frac{\operatorname{sign}(a_i)}{\|w_i\|_2} \text{ for } i \in \{j_t\}_{t=1}^m, \text{ and } \tilde{a}_i = 0 \text{ otherwise.}$$

Notice that $j_1, j_2, \cdots, j_m$ may be not distinct, but we can merge those terms into a single nonzero entry. Thus, here we have $\|\tilde{a}\|_0 \le m$ (it's a weaker condition than $\|\tilde{a}\|_0 = m$ actually).
Now we determine the constant $c$ first. Given $x$, we have:

$$\begin{aligned}
\mathop{\mathbb{E}}_{j_t \sim \mathcal{D}_M}[Y_t(x)] &= \sum_{j=1}^M p_j \cdot \frac{\operatorname{sign}(a_j)}{\|w_j\|_2} \sigma(w_j^\top x) \\
&= \sum_{j=1}^M \frac{|a_j| \|w_j\|_2}{\|\theta\|_{\mathcal{P}}} \cdot \frac{\operatorname{sign}(a_j)}{\|w_j\|_2} \sigma(w_j^\top x) \\
&= \frac{1}{\|\theta\|_{\mathcal{P}}} \sum_{j=1}^M a_j \sigma(w_j^\top x) \quad (\text{use } \operatorname{sign}(a_j) \cdot |a_j| = a_j) \\
&= \frac{1}{\|\theta\|_{\mathcal{P}}} f_M(x;\theta).
\end{aligned}$$

Then (let $c = \|\theta\|_{\mathcal{P}}/m$)

$$\mathop{\mathbb{E}}_{j_t \sim \mathcal{D}_M}\left[\tilde{f}(x)\right] = c \cdot \sum_{t=1}^m \mathop{\mathbb{E}}_{j_t \sim \mathcal{D}_M}[Y_t(x)] = \frac{c \cdot m}{\|\theta\|_{\mathcal{P}}} f_M(x;\theta) = f_M(x;\theta).$$

Therefore, $\tilde{f}(x)$ is an unbiased estimator of $f_M(x;\theta)$ over $\mathcal{D}_M$.
Look back the error term we want to bound, notice that:

$$\begin{aligned}
&\mathop{\mathbb{E}}_{x \sim \pi}\left(\sum_{j=1}^M \tilde{a}_j \sigma(w_j^\top x) - \sum_{j=1}^M a_j \sigma(w_j^\top x)\right)^2 \\
&= \mathop{\mathbb{E}}_{x \sim \pi}\left[\left(\tilde{f}(x) - \mathop{\mathbb{E}}_{j_t \sim \mathcal{D}_M}[\tilde{f}(x)]\right)^2\right] = \mathop{\mathbb{E}}_{x \sim \pi}\left[\mathop{\operatorname{Var}}_{j_t \sim \mathcal{D}_M}\left(\tilde{f}(x)\right)\right] \\
&= \mathop{\mathbb{E}}_{x \sim \pi}\left[\frac{\|\theta\|_{\mathcal{P}}^2 \cdot m}{m^2} \mathop{\operatorname{Var}}_{j_t \sim \mathcal{D}_M}(Y_t(x))\right] \quad (t \text{ is a fixed index here}) \\
&= \frac{\|\theta\|_{\mathcal{P}}^2}{m} \cdot \mathop{\mathbb{E}}_{x \sim \pi}\left[\mathop{\mathbb{E}}_{j_t \sim \mathcal{D}_M}\left[(Y_t(x))^2\right] - \left(\mathop{\mathbb{E}}_{j_t \sim \mathcal{D}_M}[Y_t(x)]\right)^2\right] \\
&\le \frac{\|\theta\|_{\mathcal{P}}^2}{m} \cdot \mathop{\mathbb{E}}_{x \sim \pi}\left[\mathop{\mathbb{E}}_{j_t \sim \mathcal{D}_M}\left[(Y_t(x))^2\right]\right] \\
&= \frac{\|\theta\|_{\mathcal{P}}^2}{m} \cdot \mathop{\mathbb{E}}_{j_t \sim \mathcal{D}_M}\left[\mathop{\mathbb{E}}_{x \sim \pi}\left[(Y_t(x))^2\right]\right].
\end{aligned}$$

The inequality holds because $\mathbb{E}_{j_t \sim \mathcal{D}_M}[Y_t(x)]^2 = f_M^2(x;\theta)/\|\theta\|_2^2 \ge 0$.

Then given $j_t$, we try to calculate

$$\begin{aligned}
\mathop{\mathbb{E}}_{x\sim\pi}\left[\left(\sigma(w_{j_t}^\top x)\right)^2\right] &= \int_{\mathbb{S}^{d-1}} \left(\max\{0, w_{j_t}^\top x\}\right)^2 \cdot d\pi(x) \\
&= \int_{\{w_{j_t}^\top x \geq 0\}} \left(w_{j_t}^\top x\right)^2 \cdot d\pi(x) \\
&= \frac{1}{2}\int_{\mathbb{S}^{d-1}} \left(w_{j_t}^\top x\right)^2 \cdot d\pi(x) \quad (w_{j_t}^\top x \text{ is symmetric over } \mathbb{S}^{d-1}) \\
&= \frac{1}{2}\mathop{\mathbb{E}}_{x\sim\pi}\left[\left(w_{j_t}^\top x\right)^2\right] \\
&= \frac{\|w_{j_t}\|_2^2}{2} \cdot \mathop{\mathbb{E}}_{x\sim\pi}\left[\left(u^\top x\right)^2\right] \quad (\text{where } u = \frac{w_{j_t}}{\|w_{j_t}\|_2}).
\end{aligned}$$

We have $\mathbb{E}_{x\sim\pi}\left[\left(u^\top x\right)^2\right] = 1/d$ (it will be proved at the end, skip it for now). Therefore,

$$\mathop{\mathbb{E}}_{x\sim\pi}\left[\left(\sigma(w_{j_t}^\top x)\right)^2\right] = \frac{\|w_{j_t}\|_2^2}{2d}. \implies \mathop{\mathbb{E}}_{x\sim\pi}\left[Y_t(x)^2\right] = \frac{\|w_{j_t}\|_2^2}{2d} \cdot \frac{\operatorname{sign}(a_{j_t})^2}{\|w_{j_t}\|_2^2} = \frac{1}{2d}.$$

Then we have

$$\mathop{\mathbb{E}}_{x\sim\pi}\left(\sum_{j=1}^M \tilde{a}_j \sigma(w_j^\top x) - \sum_{j=1}^M a_j \sigma(w_j^\top x)\right)^2 \leq \frac{\|\theta\|_{\mathcal{P}}^2}{m} \cdot \mathop{\mathbb{E}}_{j_t\sim\mathcal{D}_M}\left[\frac{1}{2d}\right] = \frac{\|\theta\|_{\mathcal{P}}^2}{2dm} \lesssim \frac{\|\theta\|_{\mathcal{P}}^2}{dm}.$$

At the end, we prove that

$$\mathop{\mathbb{E}}_{x\sim\pi}\left[\left(u^\top x\right)^2\right] = 1/d.$$

Expand $u$ to $d$-orthonormal basis of $\mathbb{R}^d$, denoted as $\{e_1, e_2, \cdots, e_d\}$, then for any given $x \in \mathbb{S}^{d-1}$, we have

$$\begin{aligned}
&x = \sum_{i=1}^d \alpha_i e_i \quad \text{where } \sum_{i=1}^d \alpha_i^2 = 1, \implies \sum_{i=1}^d (e_i^\top u)^2 = \sum_{i=1}^d \alpha_i^2 = 1 \\
\implies 1 = \mathop{\mathbb{E}}_{x\sim\pi}\left[\sum_{i=1}^d (e_i^\top u)^2\right] &= d \cdot \mathop{\mathbb{E}}_{x\sim\pi}\left[\left(u^\top x\right)^2\right] \implies \mathop{\mathbb{E}}_{x\sim\pi}\left[\left(u^\top x\right)^2\right] = \frac{1}{d}.
\end{aligned}$$

**Q.E.D.** ◁

## References

[1] Gruia Calinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0-extension problem. *SIAM Journal on Computing*, 34(2):358–372, 2005.

[2] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, page 448–455, 2003.

[3] Manor Mendel and Chaya Schwob. Fast c-k-r partitions of sparse graphs, 2009.