Nice Problems with Randomized Algorithm

Problem 1 (20') (Heavy-Light Partition)

Background Let $G = (V, E)$ be a directed graph with non-negative edge weights. Given such a graph, Dijkstra's algorithm can compute the shortest path from a source vertex to all other vertices in $O((n+m)\log n)$ time, where $n = |V|$ and $m = |E|$. However, Dijkstra's algorithm does not generalize to graphs with negative edge weights. In 2022, Bernstein, Nanongkai, and Wulff-Nilsen proposed a breakthrough algorithm for computing single-source shortest paths in graphs with arbitrary edge weights, running in $O((n+m)\cdot\text{polylog}(n))$ time. Their work received the Best Paper Award at FOCS 2022.

Problem A key step in their algorithm involves the following graph partitioning problem:
Given a directed graph $G' = (V, E')$ with **non-negative** edge weights and a parameter $r > 0$, define the **ball** around a vertex $v \in V$ as

$$B(v, r) = \{u \in V \mid \text{dist}_{G'}(v, u) \le r\},$$

where $\text{dist}_{G'}(v, u)$ denotes the shortest path distance from v to u in G'.
Design an algorithm, running in $O((n+m)\cdot\text{polylog}(n))$ time, that computes the set of all vertices $v \in V$ such that

$$|B(v, r)| \le \frac{7}{8}n,$$

and for all remaining vertices $v' \in V$, it holds that

$$|B(v', r)| \ge \frac{3}{4}n.$$

You may use randomization in your algorithm. The output should be correct with high probability. You need to prove the correctness of your algorithm.

Problem 2 (30') An Undergraduate Broke Yao's Conjecture

本科生推翻姚期智40年前的猜想，哈希表的平均查询时间竟与填满程度无关

机器之心 2025年02月11日 11:08 北京

选自量子杂志

作者：Steve Nadis

机器之心编译

1985 年，著名计算机科学家、图灵奖得主姚期智提出了一个与哈希表有关的猜想。现在，40 年过去了，一名本科生却成功推翻了这个猜想。而这项成就却源自一个始于 2021 年秋的故事。

Figure 1: Background of this problem

A **hash table** is a widely used data structure that stores a collection of key-value pairs and supports efficient operations such as insertion, lookup, and deletion. The core idea is to store data in an **array of size** n, where each position in the array is called a **slot**.
To insert a key (e.g., a username or student ID), we compute its **hash value**—an integer between 0 and $n-1$—using a hash function. This value determines the slot where the key should be stored.
However, two different keys may hash to the same slot. This situation is called a **collision**. A common technique to resolve collisions is called **open addressing**, where we use multiple independent hash functions $h_1, h_2, h_3, \ldots$ to generate a sequence of candidate slots. We probe these candidate slots one by one until an empty one is found.
The performance of open addressing depends heavily on the **load factor**—i.e., how full the table is. When the table is nearly full, insertions may require many probes.

In 1985, Professor Andrew Yao made the following conjecture:

Conjecture 1 (Yao's Conjecture, 1985)**.** *In any open-addressing hash table, if the table already contains* $(1-\varepsilon)n$ *keys, then any insertion must take at least* $\Omega(1/\varepsilon)$ *expected time.*

This conjecture suggests that inserting into a nearly full hash table is inherently slow.

However, in 2021, an undergraduate student constructed a new hashing scheme that **refuted** Yao's conjecture. They designed a randomized insertion algorithm in which each insertion takes expected time

$$O(\log^2(1/\varepsilon))$$

This surprising result shows that efficient insertions are possible in dense hash tables under the right design.

Problem In this problem, you will prove the following result:

Theorem 2. *There exists a randomized algorithm such that, given* n *and* ε*, we can insert* $(1-\varepsilon)n$ *keys into a hash table of size* n*, where each insertion takes at most* $O(\log^2(1/\varepsilon))$ *time in expectation.*

The insertion algorithm is described below. You may assume that all hash functions used in the algorithm generate **uniformly random and independent** values.

Algorithm 1 Efficient Hash Table Insertion

```
1: Input: Number of slots $n$, load parameter $\varepsilon$
2: Partition the $n$ slots into $k = O(\log \frac{1}{\varepsilon})$ disjoint blocks:
     • The first block has size $n/2$, the second $n/4$, the third $n/8$, and so on.
     • The last two blocks each have size $\frac{\varepsilon n}{128}$.
3: For each insertion of an element $x$:
4: Let flag $\leftarrow 0$
5: for $i = 1$ to $k$ do
6:     Let $t \leftarrow 100 \log \frac{1}{\varepsilon}$
7:     if $i = k$ then
8:         Set $t \leftarrow 100 \log n$                ▷ Use more probes in the last block
9:     for $j = 1$ to $t$ do
10:        Compute $h_{i,j}(x)$         ▷ $h_{i,j}$ is a fully random hash function into block $i$
11:        if slot $h_{i,j}(x)$ is empty then
12:            Place $x$ in $h_{i,j}(x)$
13:            Set flag $\leftarrow 1$, break
14:    if flag $= 1$ then
15:        break
16: if flag $= 0$ then
17:     Enumerate all slots to find the first empty one and insert $x$
```

We will prove the following intermediate result:

Lemma 3. *The last block is at most half full with high probability. That is, it contains at most $\frac{1}{2} \cdot \frac{\varepsilon n}{128}$ elements.*

(1) [5'] Assuming Lemma 3, prove that the expected time of the **last insertion** is at most $O(\log^2(1/\varepsilon))$.

To prove Lemma 3, we will use the following probabilistic tool:

Lemma 4 (Coupon Collector's Lemma)**.** *Suppose there are n distinct types of coupons. If we draw $X = 2n \log \frac{1}{\delta}$ independent coupons uniformly at random (with replacement), then with high probability, we will have collected at least $(1-\delta)n$ distinct types. You can assume $\delta^{-1} \le n^{0.1}$.*

Background: The **Coupon Collector Problem** is a classic result in probability theory. It describes the number of samples needed to collect a full set of n unique items when each item is sampled uniformly at random. The lemma above is a strengthened version which ensures that we collect almost all types with high probability.

(2) [15'] Prove Lemma 4.
Hint: Compute the expectation, then use concentration bound. (Like Chernoff, Azuma, McDiarmid)

(3) [10'] Prove Lemma 3.
Hint: Suppose block i is more than half full. Let its size be S_i, so it contains at least $S_i/2$ elements. These elements must have failed to insert into block $i-1$, even after trying $t = 100 \log \frac{1}{\varepsilon}$ random slots in that block. This gives a total of at least $\frac{S_i}{2} \cdot t = \frac{S_{i-1}}{4} \cdot 100 \log \frac{1}{\varepsilon}$ probes into block $i-1$. Apply Lemma 4 to show that block $i-1$ must have been almost completely full, and then show $i-2$ is also almost completely full ...

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

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

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

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

Remarks:

- The **diameter** of $G[V_i]$ is $\max_{u,v \in V_i} d_G(u, v)$.
- β balances cluster tightness (D) and separation likelihood. Lower β 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 2 Low Diameter Decomposition (LDD)

```
Require: Undirected graph $G = (V, E)$, target diameter $D > 0$
Ensure: Vertex partition $\{V_1, \dots, V_k\}$ with $\mathrm{diam}(G[V_i]) \leq D$
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:     Compute $B \leftarrow \{u \in V \mid \neg marked[u] \wedge d_G(v_0, u) \leq R_{v_0}\}$
6:     Create cluster $C \leftarrow B$
7:     $marked[u] \leftarrow$ TRUE for all $u \in C$
8:     Add $C$ to output partition
9: 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 6. *A randomized low-stretch tree of stretch α 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 7. *For any metric space $M = (V, d)$, there exists an efficiently sampleable α_B-stretch spanning tree distribution $\mathcal{D}_B$, where*

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

Δ_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 7 by the following algorithm.

Algorithm 3 Low Stretch Tree Construction, LST(M, δ)

```
Require: Metric space $M = (V, d)$, target diameter $D = 2^\delta$
Ensure: Spanning tree $T$ with low stretch. Invariant: diameter$(M) \leq 2^\delta$
1: if $|V| = 1$ then
2:     return trivial tree containing the single point
3: Partition $V$ into clusters $C_1, \ldots, C_t \leftarrow$ LDD$(M, D/2)$
4: for $j = 1$ to $t$ do
5:     Let $M_j$ be $M$ restricted to $C_j$
6:     Recursively build $T_j \leftarrow$ LST$(M_j, \delta - 1)$
7: Connect roots $r_2, \ldots, r_t$ to $r_1$ with edges of length $2^\delta$
8: return final tree $T$ rooted at $r_1$
```

Lemma 8. *If the random tree T returned by some call LST(M, δ) has root r, then*

1. *every vertex x in T has distance $d(x, r) \leq 2^{\delta+1}$*
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 β is the approximate factor in LDD)*

If we can prove Lemma 8, 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 7 can be proved.

b. (20') Prove the Lemma 8.

c. (10') Prove the following theorem.

Theorem 9. *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 10. *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 $\geq 1 - \frac{1}{poly(n)}$.*

Problem 4 (50') (A Combinatorial Proof of Chernoff Bound)

In the lecture, you learned how to prove Chernoff Bound by Generating Function. In this problem, you will learn another way to prove Chernoff Bound. Our goal is to prove the following theorem:

Theorem 11. *Suppose $X_1, \ldots, X_n$ are i.i.d random variables from $\{-1, 1\}$, each w.p. $\frac{1}{2}$, and k be a positive integer with $k \leq \sqrt{n}$. Then*

$$\Pr\left[\sum_{i=1}^{n} X_i \geq k\sqrt{n}\right] \leq e^{-\Theta(k^2)}$$

You will achieve this goal step by step.

a. (10') Prove the following lemma:

Lemma 12. *(Poor Man Chernoff Bound) Suppose $X_1, \dots, X_n$ are i.i.d random variables from $\{-1, 1\}$ each w.p. $\frac{1}{2}$, and k be a positive integer. Then*

$$\Pr\left[\sum_{i=1}^{n} X_i \geq 2k\sqrt{n}\right] \leq 2^{-k}$$

Hint: First, you can use Chebeyshev's Inequality to prove the following fact:

Fact 13. *Suppose $X_1, \dots, X_n$ are i.i.d random variables from $\{-1, 1\}$ each w.p. $\frac{1}{2}$. Then*

$$\Pr\left[\sum_{i=1}^{n} X_i \geq 2\sqrt{n}\right] \leq \frac{1}{4}$$

Then, consider $S_i = \sum_{j=1}^{i} X_j$, let p be the first point where $S_p \geq 2\sqrt{n}$, then $\Pr[S_n \geq 2k\sqrt{n}] = \Pr[p \text{ exists}] \cdot \Pr[S_n - S_p \geq 2(k-1)\sqrt{n} \mid p \text{ exists}]$. If you can prove $\Pr[p \text{ exists}] \leq \frac{1}{2}$, you can prove the lemma by induction.

b. (10') Prove the following lemma:

Lemma 14. *(Chernoff Bound for Geometric Distribution)*

Suppose $X_1, \dots, X_n$ are i.i.d random variables, that $X_i \geq 0, \Pr[X_i \geq j] \leq p^j, \forall j = 1, 2, \dots$ for a $p < \frac{1}{4}$. Then

$$\Pr\left[\sum_{i=1}^{n} X_i \geq 2n\right] \leq (4p)^n$$

.

Hint: If we can prove $\Pr[\sum_{i=1}^{n} \lfloor X_i \rfloor \geq n] \leq (4p)^n$, then it's easy to see the lemma will hold. Suppose $\sum_{i=1}^{n} \lfloor X_i \rfloor \geq n$, then there exist $Y_1, \dots, Y_n$, that $\forall 1 \leq i \leq n, X_i \geq Y_i$ and $\sum_{i=1}^{n} Y_i = n$. Fix the sequence $Y_1, \dots, Y_n$, calculate the probability of sequence X_i satisfies $\forall i, X_i \geq Y_i$. Then use union bound for all the possible sequence of Y.

c. (10') Prove the following lemma:

Lemma 15. *(Lowerbound for Chernoff Bound)*

Suppose $X_1, \dots, X_n$ are i.i.d random variables from $\{-1, 1\}$ each w.p. $\frac{1}{2}$, and k be a positive integer and $k \leq \sqrt{n}$, Then

$$\Pr\left[\sum_{i=1}^{n} X_i \geq \frac{k}{2}\sqrt{n}\right] \geq (\frac{1}{4})^{k^2}$$

.

You can use the fact:

Fact 16. *Suppose $X_1, \dots, X_n$ are i.i.d random variables from $\{-1, 1\}$ each w.p. $\frac{1}{2}$. Then*

$$\Pr\left[\sum_{i=1}^{n} X_i \geq \frac{1}{2}\sqrt{n}\right] \geq \frac{1}{4}$$

.

Hint: Divide $X_1, \dots, X_n$ into $m = k^2$ groups, use the Fact 16 on each group.

d. (20') Prove Theorem 11.