CS 388R: Randomized Algorithms, Fall 2019 November 21st

Lecture 25: Random Walk on Undirected Graphs; Closest Pair in Plane

Prof. Eric Price *Scribe: Hongru Yang*

NOTE: THESE NOTES HAVE NOT BEEN EDITED OR CHECKED FOR CORRECTNESS

1 Markov Chain; Random Walk on Undirected Graphs (Continued)

Hitting time:

$$h_{uv} = \mathbb{E}[\text{time to reach } v \text{ starting at } u]$$

Commute time:

$$C_{uv} = h_{uv} + h_{vu} = 2mR_{\text{eff}}(u, v)$$

Cover time:

$$C_u(G) = \mathbb{E}[\text{time to visit all vertices starting at } u]$$
$$C(G) = \max_u C_u(G) \leq 2m(n-1)$$

Theorem 1. *$mR_{max} \leq C(G) \leq O(1)mR_{max} \log n$, where $R_{max} = \max_{u,v} R_{eff}(u, v)$.*

Proof. Lower bound: $C(G) = \max_u C_u(G) \geq \max_u \max_v h_{uv} \geq \max_{u,v} \frac{C_{uv}}{2} = mR_{\max}$.

Upper bound: Suppose starting at u,

$$\mathbb{P}[\text{haven't visited } v \text{ by time } t] \leq \frac{h_{uv}}{t} \leq \frac{C_{uv}}{t} = \frac{2mR_{\text{eff}}(u, v)}{t} \leq \frac{2mR_{\max}}{t}$$

If we take $t = 4mR_{\max}$, by Markov's inequality, we have $\mathbb{P}[\text{haven't visited } v \text{ by time } t] \leq \frac{1}{2}$. However, observe that the above proof holds for any starting u. Thus, we have

$$\mathbb{P}[\text{haven't visited } v \text{ between time } [t+1, 2t] \text{ regardless of state at } t] \leq \frac{1}{2}$$

which implies

$$\mathbb{P}[v \text{ not reached by } t2\log n] \leq \frac{1}{2^{2\log n}} = \frac{1}{n^2}$$
$$\mathbb{P}[\text{not cover the entire graph by } t2\log n] \leq \frac{1}{n^2}$$

□

Application

Check $s - t$ connectivity on an undirected graph in $O(\log n)$ space. $2mn$ steps per (s, t) pair. $O(\log n)$ space to tell where I am and keep track of how many steps performed.

2 Closest Pair in Plane

Suppose we have n points in plane, $P = \{p_1, p_2, \ldots, p_n\}$ and we want to find $CP(P) = \min_{i \neq j} \|p_i - p_j\|_2$. Naively we can do this in $O(n^2)$. We are going to show a deterministic algorithm that runs in $O(n \log n)$ time and a randomized algorithm that runs in expectation $O(n)$ time.

Deterministic Closest Pair in Plane

1. Recursively split all the points by its x coordinate. Then $P = L \cup R$ and $|L| = |R| = \frac{n}{2}$. Solve $CP(L), CP(R)$ and let $\sigma = \min(CP(L), CP(R))$.
2. Find the minimum distance δ of pairs across boundary $L \leftrightarrow R$.
3. Return $\min(\delta, \sigma)$.

To do step 2 in $O(n)$ time, first notice that we only need to go through all the points that are within σ distance to the left and right of the splitting boundary. Further, we can divide the region into squares of size $\sigma \times \sigma$ by y axis. Inside each square, there can be at most 4 points. Therefore, for a given point that is to the left of the boundary, we only need to look at the 3 adjacent box to the right of the boundary and there are at most 8 points.

To achieve above, we use $\lfloor \frac{x}{\sigma} \rfloor, \lfloor \frac{y}{\sigma} \rfloor$ as key to store points into a hash table. So, we can query all the points in a square in $O(1)$ time.

Randomized Closest Pair in Plane

Suppose we have correct σ beforehand, we can split the plane into $\sigma \times \sigma$ small squares and all squares have at most 4 points inside. Then the closest pair lies in adjacent cells. We can then scan through all points and for each point we check all adjacent cells and then add the point into the hash table.

New goal: given a guess σ, is $\sigma_{\text{true}} > \sigma, \sigma_{\text{true}} = \sigma, \sigma_{\text{true}} < \sigma$?

We can run the algorithm and if we have more than 5 points in a square then $\sigma_{\text{true}} < \sigma$.

New algorithm:

1. Randomly assign order to all the points and we denote the first i points in this order P_i.
2. Scan through all the points in this order. For each point p_i,
   - Find $CP(P_i)$.

- If $CP(P_i) < \sigma$, rebuild the hash table of P_i with $\sigma = CP(P_i)$.
- Else add p_i to the hash table.

3. Return σ

Notice that we can compute $CP(P_i)$ in constant time since all the points in the hash table are at least σ distance apart (and thus each square has at most 4 points) and we can query all the points that are in adjacent cells of p_i in constant time.

The running time of this algorithm in expectation is $\sum_{i=1}^{n} 1 + \mathbb{P}[\text{need to rebuild for } p_i]i = 3n$ since $\mathbb{P}[\text{need to rebuild the hash table for } p_i] = \mathbb{P}[\text{minimum distance among } P_i \text{ uses } p_i] = \frac{i-1}{\binom{i}{2}} = \frac{2}{i}$.

3 Introduction to Point Location in Regions Split by Lines in General Location

If we have a set of n lines L in general location, we will have $\binom{n}{2} + 1 = \Theta(n^2)$ regions. We can prove this by observing that whenever we add a new line we will create total number of lines more regions.

Definition 2. *The **triangulation** of L is defined by adding line segment between intersections of lines in L to split multi-sided regions so that each divided region is a triangle. (We assume parallel lines intersecting each other at infinity.)*

It can be shown that there are $O(n^2)$ triangles in a triangulation.

We assign labels to all the regions and we want to output label of a given point.

Our goal is to create a data structure with $O(n^2)$ time and space complexity beforehand so that we can do fast query later.

A Related Problem: R-ary Tree

R-ary can be created by first randomly sample R points from a set of points associated with one node. Then these points define a splitting of all the points. Create children of the node based on the splitting.

Then, if for a given node we have r branches,

$$\mathbb{E}[\text{size of a given branch}] = \frac{n}{r}$$

By Chernoff's bound this implies,

$$\mathbb{P}[\text{size of a given branch} \geq \frac{n}{r}\log r] \leq e^{-\log r} = \frac{1}{r}$$

which implies w.p. 3/4, the maximum branch size is at most $\frac{n \log r}{r}$.

Thus the expected query time can be bounded by $Q(n) \leq O(r) + Q(\frac{n \log r}{r}) \leq O(1) + Q(n/10) = O(\log n)$ if we take $r = 10$.