CS 594: Representations in Algorithm Design	Spring 2022

Lecture 3: 01/18

Lecturer: Xiaorui Sun — *Scribe: Tianyu Zhu*

Relevant Reading: Henzinger-King 1999
Problem: Dynamic Connectivity

- Input changes: maintain a data structure for some problem;
- Graph updates: edge insertions / deletions, query (x, y).

Goal: Handle updates and answer queries as fast as possible.

Theorem 1 *Dynamic Connectivity is $O(\text{poly}\log n)$ time for each update and query.*

Approach: Maintain a spanning forest of the graph.

Observation 2 *Each update requires at most 1 edge change in the spanning forest.*

- *Insertion*

(1) Two vertices are in the same connected component $\Rightarrow$ No update,

(2) Two vertices are in different connected components $\Rightarrow$ Add to the spanning forest;

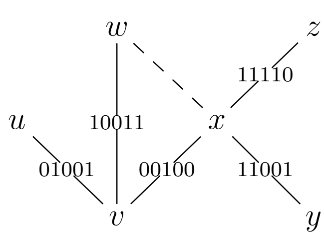
- *Deletion*

(3) Not in the spanning forest $\Rightarrow$ No update,

(4) In the spanning forest $\Rightarrow$ **Find a replacement if possible.**

Approach: Randomly assign a unique binary name for each edge. Name of a vertex is exclusive-or sum of the names of incident edges. Name of a connected component is XOR sum of the names of all its vertices.

Example 3



(w, x) *is deleted,* $S_u = 01001, S_w = 10011, S_v = 01001 \oplus 00100 \oplus 10011 = 11110.$
$CC_1 := \{u, v, w\}, S_{CC_1} = S_u \oplus S_v \oplus S_w = 00100.$

Conclusion

- If a connected component CC does not have any outgoing edges, then $S_{CC} = 0$;
- If outgoing edge e is unique, then $S_{CC} = S_e$, name of the edge;
- If outgoing edges are multiple, then $S_{CC} = \oplus_{e \in \delta(CC)} S_e$, XOR sum of their names. The sum can be 0, but only with low probability under enough long names.

Idea: It is easy when the replacement is unique. Otherwise, we make it unique by sampling some edges for a new graph G_p. Assume there are two connected components with t edges between them, then $\forall e \in E$, let $e \in E(G_p)$ with probability $\frac{1}{t}$.

$$Pr[\text{outgoing edge is unique in } G_p] = t\frac{1}{t}(1 - \frac{1}{t})^{t-1} \overset{t \to \infty}{\to} \frac{1}{e}.$$

Since t is unknown, we maintain $\log n$ many G_p for $p = \frac{1}{2}, \frac{1}{4}, ..., \frac{1}{n}$.
Claim $\exists p$ s.t. $\frac{1}{2t} < p \leq \frac{1}{t}$, so $Pr \geq \frac{1}{e^2}$. Maintaining $\log^5 n$ many G_p for each p can improve Pr to $1 - \frac{1}{n^{100}}$. See relevant reading for details.

Definition 4 (ET Tree) *(1) link two trees together by adding a new edge;*

(2) cut a tree into two trees by deleting an edge;

(3) maintain label for each vertex and tree (sum of its vertices);

(4) get root of the tree containing given vertex.

Data Structure: Spanning forest and poly $\log n$ many sampled graphs G_p, where ET trees are built with vertex labels being binary names (sum of incident edge names).
$Insert(x, y)$

- Insert (x, y) to spanning forest if needed;
- Maintain G_p: For each G_p, add (x, y) with probability p.

$Delete(x, y)$

- Maintain G_p: If $(x, y) \in E(G_p)$, then delete it and maintain ET tree;
- Find replacement: For each G_p, let CC_1 be the connected component containing x, and use ET tree (4) to get S_{CC_1}. If $\exists e \in E(G_p)$ s.t. $S_{CC_1} = S_e$, then e is the replacement to be added to the spanning forest.

$Query(x, y)$: Get root (r_x, r_y) of (x, y), return $r_x == r_y$.
Note: Above data structure is non-adaptive (fixed once built) and more likely to make mistakes as time goes. This can be modefied by rebuilding the data structure after too many updates.