DYNAMIC MODEL

- **Incremental** (edges only added)
- **Decremental** (edges only removed)
- **Fully-dynamic** (edges added/removed)

Presentation

Incremental BFS(s) for a given "s" ← Total update time $O(mn)$

$O(n)$ amortized update time

Ques How to maintain All-pairs distances

Ans Maintain incrementally BFS(v), $\forall v \in V$.

Total time $O(mn) * n)$

As we have m-edge insertions.

Amortized time $= O(n^2)$.

NOTE → Above sol^n works for unweighted case.

Notation

* $OUT\text{-}BFS_k(s)$ — First k-levels of $OUT\text{-}BFS(s)$

* $IN\text{-}BFS_k(s)$ — First k-levels of $IN\text{-}BFS(s)$

$\sigma = (e_1, e_2, \ldots, e_m)$ ← sequence of edge insertions

e_i = edge inserted at time $t = i$

$G_0 = (V, \emptyset)$ ← empty graph

$G_i = (V, \{e_1, e_2 \ldots e_i\})$ ← graph at time $t = i$

$\pi_{G_i}(x, y)$ ← x to y shortest path in graph G_i.

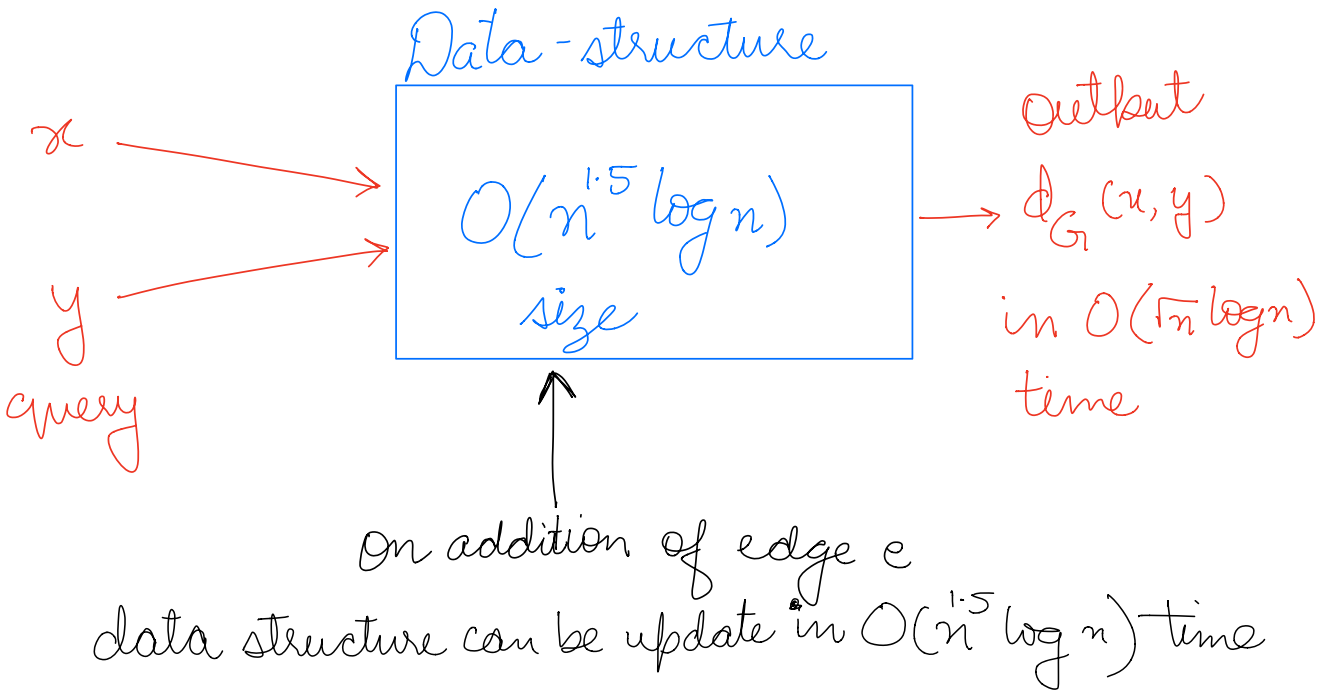
Note: $\left\{ \pi_{G_t}(x,y) \;\middle|\; 1 \le t \le m,\ x, y \in V \right\} = m \cdot n^2 \le n^2 \cdot n^2 = n^4$

Problem Statement :

Maintain a oracle (data-structure) of $O(n^{1.5} \log n)$ size incrementally (i.e. for a graph undergoing edge insertions), so that :

(i) after every edge insertion, the oracle can be updated quickly (in amortized $O(n^{1.5} \log n)$ time.)

(ii) given any query pair (x, y) the oracle reports $d_G(x, y)$ in $O(\sqrt{n} \log n)$ time.

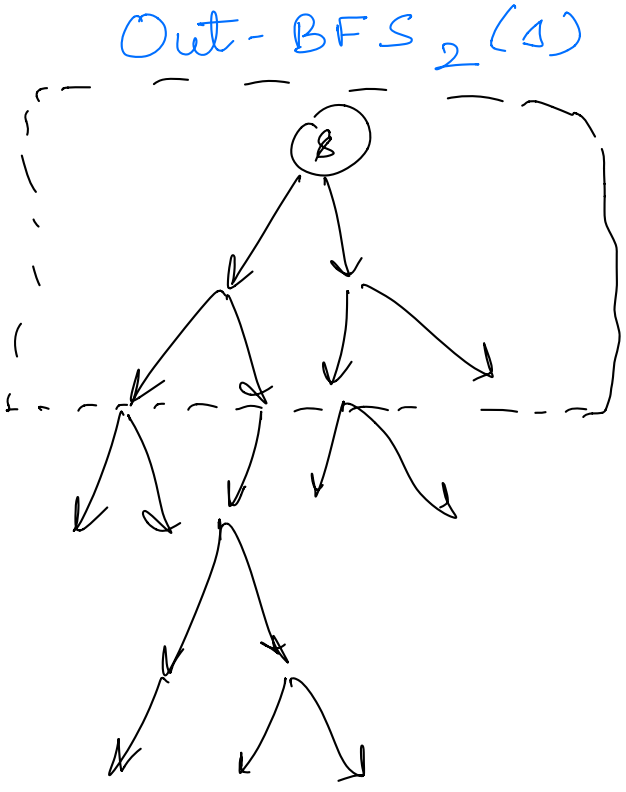


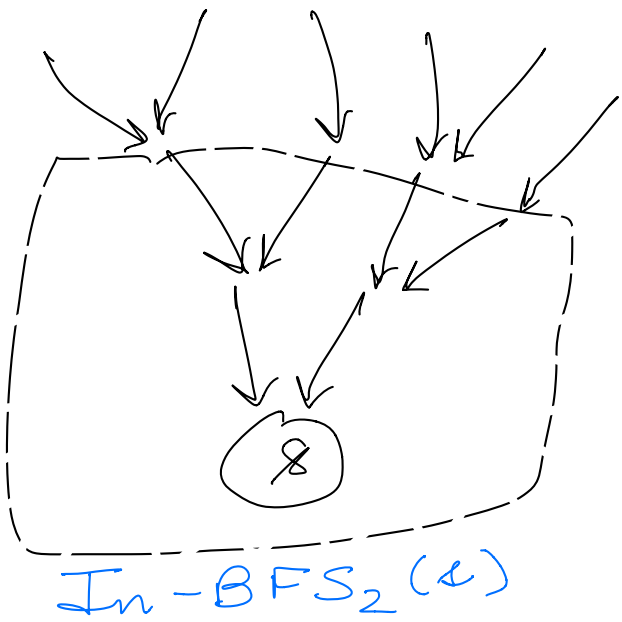
$In\text{-}BFS_k(s)$ ≡ First k levels of $In\text{-}BFS(s)$

$Out\text{-}BFS_k(s)$ ≡ First k levels of $Out\text{-}BFS(s)$

Lemma 1: For any unweighted $G = (V, E)$ and $s \in V$, we can incrementally maintain $In\text{-}BFS_k(s)$ and $Out\text{-}BFS_k(s)$ in $O(mk)$ total time.

Proof: H.W. (Hint: We only need to maintain sets $L_1, L_2 \ldots L_k$ Refer to talk!)





Data-Structure

(1) For each $v \in V$, we maintain out-$BFS_{\sqrt{n}}(v)$

(2) $R \leftarrow$ Random set of size "$16\sqrt{n}\log n$"

For each $v \in R$, we maintain in-bfs(v) & out-bfs(v)

Total update time:

$$|V| \cdot O(m\sqrt{n}) + |R| \cdot O(mn)$$

$$= O(mn\sqrt{n}\log n) \Rightarrow$$

Trivial is $O(mn)$ after every update

Amortized update time $= O(n\sqrt{n}\log n)$

Lemma 2: The following holds with high probability:

For each $1 \le i \le m$ (time stamp), $x, y \in V \times V$,

if $d_{G_i}(x,y) \ge \sqrt{n}$, then "$R \cap$ vertices $(\pi_{G_i}(x,y))$" is non-empty.

Proof: H.W.

Query Algorithm:

Given a pair (x, y) how to find $d_{G_t}(x, y)$?

(1) Check if $y \in$ out $-BFS_{\sqrt{n}}(x)$

If yes, then report it directly

(2) If $y \notin$ out $-BFS_{\sqrt{n}}(x)$

$\Rightarrow d_{G_t}(x, y) > \sqrt{n}$ (Here t = current time)

$\Rightarrow \pi_{G_t}(x, y)$ contains a vertex of R.

So, in this case,

$$d_{G_t}(x, y) = \min_{r \in R} \{ \underbrace{d_{G_t}(x, r)}_{\text{can be obtained by IN-BFS}(r)} + \underbrace{d_{G_t}(r, y)}_{\text{can be obtained by OUT-BFS}(r)} \}$$

⊛ It is easy to verify query time is

$O(|R|) = O(\sqrt{n} \log n)$

⊛ The time to update data-structure is

$O(n\sqrt{n} \log n)$ on average (per edge).

Note: Here we talk about worst-case and not amortized

Another Problem:

Ques. Can you have incremental algorithm for ALL-Pairs-distances with worst-case update time of $O(n^2)$ and query time of $O(1)$?

Sol^n: Yes, we can explicitly maintain distance matrix.

Sub-problem for (u,v) pair:

How to find

$d_{G_t}(u,v)$ from $d_{G_{t-1}}(u,v)$ in $O(1)$ time?

Suppose we added edge $e_t = (x, y)$ at time "t."

Then,

$$d_{G_t}(u, v) =$$

$$\text{MIN} \left\{ d_{G_{t-1}}(u, v), \; d_{G_{t-1}}(u, x) + wt(x, y) + d_{G_{t-1}}(y, v) \right\}$$

$\Rightarrow d_{G_t}(u, v)$ can be updated in $O(1)$ time.

As there n^2 vertex pairs, the time to update adjacency-matrix is $O(n^2)$!
