Dynamic Graph Algorithms

Solve standard algorithmic problems in the setting where graph is changing over time.

Eg. dynamic connectivity

- insert (u,v). ← insert only
- delete (u,v). ← delete only

} fully dynamic ?

- connected (u,v) ? connected (G) ?

Basic question, many results, but not completely resolved yet.

~~Worst~~ Clearly if arbitrary updates @ each operation → can run DFS each time.
So ~~but~~ minimize update time ~~vs~~ vs query time.

some results:

Frederickson SICOMP 85	$O(m^{1/2})$ update	$O(1)$ query time	deterministic / worst case.
Eppstein et al.	$O(n^{1/2})$	"	HW.
Holm de Lichtenberg Thorup '98	$O(\text{polylog})$ update	$O(\log n)$ query	~~Holm~~ amortized det. ~~de Lichtenberg~~ ~~Thorup~~
Kapron King Mountjoy	$O(\text{polylog})$ update	"	random // worst case !

Let's see some ideas used in these papers today.

BTW : LB : Demaine Patrascu : $T_{update} \times \log (T_{query}/T_{update})$
$T_{query} \times \log (T_{up}/T_q)$ $\geq \Omega(\log n)$.

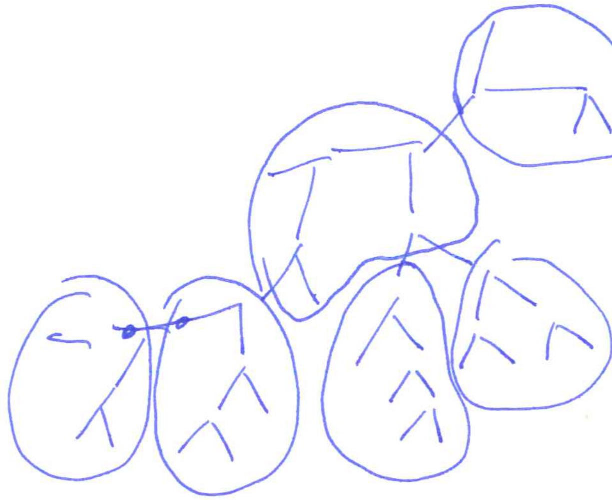
even randomized algo (Monte Carlo whp).

Frederickson (A weaker $m^{2/3}$ result).

. Sps insert-only. Then union-find does the job. Insert = Union. } implicitly maintains a spanning forest.
Query = two finds.

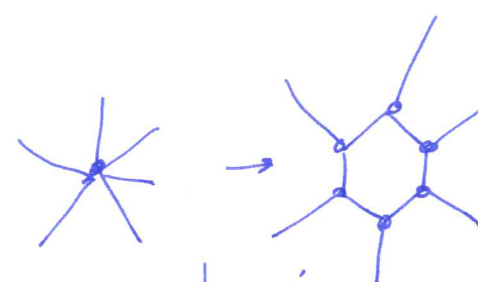
. But delete: hmm: suppose we maintain a spanning forest and delete an edge in tree. how to find a replacement edge. (if delete off-tree edge, no worries).

Idea: Cluster the tree into connected pieces of size $\simeq z$. (some parameter).

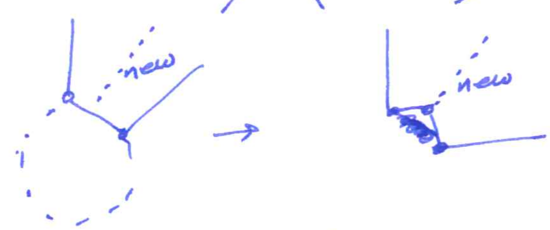


Assumption 1: G has degree ≤ 3.

Replace high degree nodes by gadgets of cycles



now if new edge then



insert/delete of 1 edge in original graph

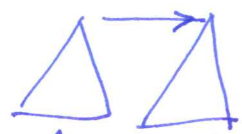
$\Leftrightarrow$ insert delete of $O(1)$ edges in new graph.

New graph has $m = \Theta(n)$ and degree ≤ 3.

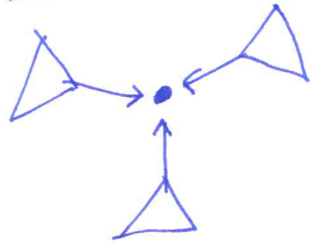
Fact 2: Given a tree with max degree ≤ 3 and parameter z, can ~~delete~~ mark set of edges so that non-marked edges form components of size $\in [z, 3z)$ ← # vertices.

assume tree of size $> z$

Pf: Direct each edge from smaller subtree to larger (break ties arbitrarily).



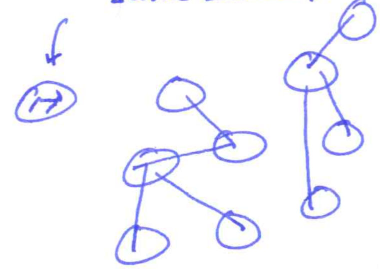
Then ~~at~~ ~~most~~ one vertex is outdegree $= 0$.



if tree has size $\geq 3z$ then ~~deleting~~ for at least one of these in-edges, deleting it gives components of size $\geq z$ on both sides. Recurse.

Actually can do this in time linear in the tree. (How? Exercise).

Call this a z-bounded clustering of the tree. (if tree has $< z$ nodes, all vtcs in the same cluster)



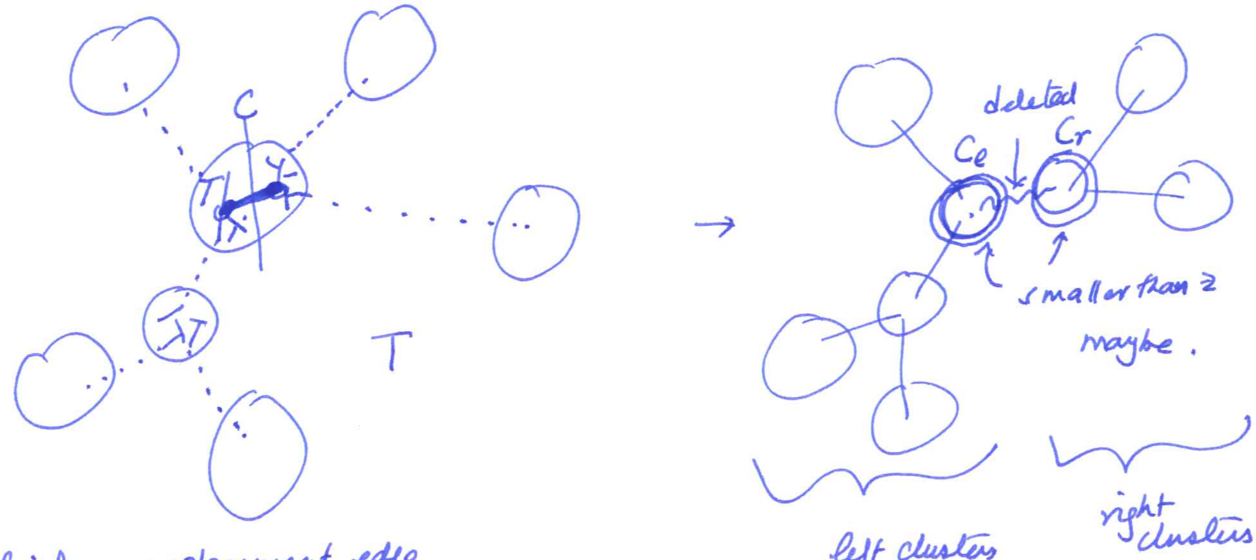
We maintain:
- a spanning forest F of G.
- a z-bounded clustering of each tree.
- for each pair of clusters of same tree, 1 bit for whether $\exists$ edge from C_i to C_j not in F. (and a list of these edges)

Inserts are easy (exercise!) so look at deletes.

If off-tree edge deleted, just update the bit for the clusters (C_i, C_j) this edge goes ~~next~~ between. (Must belong to same tree in F, since F spanning forest).

So delete $e \in F$. (say $e \in T$).

Sps e within cluster (other case similar)



Try to find a replacement edge b/w left & right clusters.

- At most $O(m/z)$ clusters $\Rightarrow$ for each look up bit $\Rightarrow$ $O((m/z)^2)$ time.

But careful! No record of (C_ℓ, C') edges since $C_\ell \cup C_r = C$ was original cluster.

No prob! only $O(z)$ edges hit $C_\ell \cup C_r = C$, so scan all of them in $O(z)$ time.

- Finally, either replacement edge or not. Either case C_ℓ, C_r maybe too small. So merge with some neighbor cluster and re-partition if too big. Again $O(z)$ time.

Total time: $O((m/z)^2 + z)$. Balance to get $z = O(m^{2/3})$. ☺.

Using more careful data structures, get $O(m^{1/2})$. [same paper].

Eppstein et al gave clever technique (very general) to reduce to $\tilde{O}(n^{1/2})$. [Homework]

Randomized Idea (Very high level sketch)

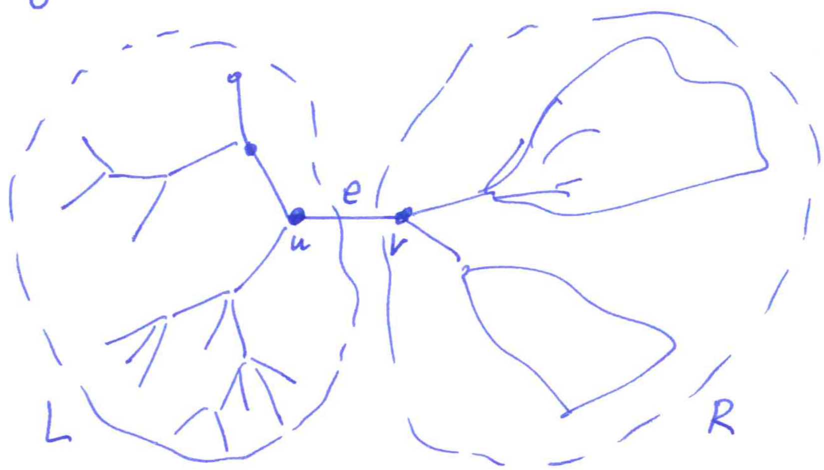
Suppose we have just one deletion. Want fast $\overbrace{\text{worst case}}^{\text{wp 1.}}$ time, only that error probability is $o(1)$.

Here's idea: each vertex name is $O(\lg n)$ bits : $\ell(x) \leftarrow$ "label of x".

edge $e = (u,v)$ represented by $\langle \ell(u), \ell(v) \rangle \stackrel{\text{def}}{=} \ell(e)$

(assume $u < v$ according to some total order on vertex names).

Maintain spanning forest on G. Now get delete($e=(u,v)$) operation (wlog assume tree T)



Need replacement edge between L & R.

Suppose: Only one edge f in $Cut(L,R) \setminus \{e\}$. Unique replacement edge.

Then: $$\underbrace{\bigoplus_{x \in L}^{\text{XOR}} \bigoplus_{\substack{\text{all edges } e' \\ \text{in } G \\ \text{incident to } x}} \ell(e')} = \ell(f) = \bigoplus_{x \in R} \bigoplus_{e' \in \partial_G(x)} \ell(e').$$ ☺

~~What if many replacement edges?~~

How do you compute this in polylog time? ~~Maintain the XOR of~~

Use some smart data structures (link-cut trees or Euler-tour trees).

OK: What if there are many edges crossing from $L \to R$ in $G \setminus e$?

Sample! For each j, sample edges in G wp $1/2^j$ and maintain the $\bigoplus_{e' \in \partial_G(x) \cap \text{sample}} \ell(e')$

for each such sample. (let $\bigoplus_{e' \in \delta_G(x) \cap \text{sample}} \ell(e')$ be the "signature" of x.)

This will work if $\exists$ a unique edge from $L \to R$ in the sample. So if $\exists \le 2^j$ edges across the cut, $\Pr[\text{unique edge survives}] \cong \sum_{e \in \delta_G(x)} \frac{1}{2^j} \cdot \left(1 - \frac{1}{2^j}\right)^{2^j - 1} \cong \Theta(1)$.

Now if repeat $\Omega(\log n)$ times, one sample succeeds w.p. $1 - \frac{1}{\text{poly}(n)}$!!

Does not quite work immediately for multiple deletes. B/c the tree after the first delete depends on the sample, and now subsequent calculations cannot use independence the same way.

But Kapron King Mountjoy [SODA 12] find a way around it.

Won't do this in class

The Holm et al Idea for amortized polylog (deterministic) bound

Again, how to find replacement edges? Can scan all edges of $L \cup R$ but may do this repeatedly! Need some way to measure progress.

<u>Idea</u>: Each edge is kept @ some "level" $\in [0, \log_2 n]$. "Charge" scanning edges to raising the level of some edges. (L = $\log_2 n$)

<u>Insert</u>: edge added @ level 0.

<u>Delete</u>: · if e @ level ℓ ← highest level for e

· look at tree $T \ni e$, splits into L & R. Say $|L| \le |R|$.

(if e not in any tree, ignore.)

· raise edges of L to level $\ell+1$ (those edges of level ℓ).

// Maintains Invariant II b/c $|L| \le |T|/2 \le n/2^\ell \cdot 1/2$ by induction.

<u>Invariants</u>: $E_0 \supseteq E_1 \supseteq \cdots E_i \supseteq \cdots E_L$ ← edges @ level i or higher

(I) F_i is spanning forest of E_i

$F_{i-1} \supseteq F_i$

i.e. if $x,y \in E_i$ connected $\Rightarrow$ x,y connected in F_i

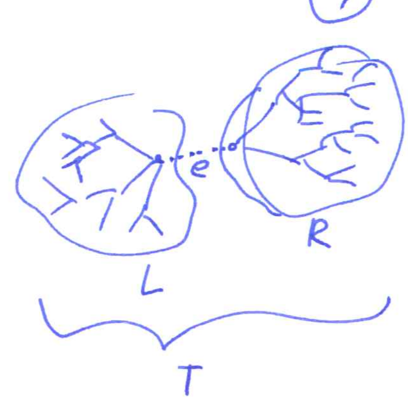
(II) each component in F_i has $\le n/2^i$ nodes

▶ Now start scanning nontree edges incident to $V(L)$. ~~is the~~

those of level ℓ.

If it goes within L, raise to level $\ell+1$.

If it goes to R, done. Stop, and add to $F_\ell, F_{\ell-1} \dots F_0$.



▶ Sps. run out of level-ℓ edges, then go to level $\ell-1$.

Now $F_{\ell-1}$ has tree $T_{\ell-1}$ containing e that has split into $L_{\ell-1}$ & $R_{\ell-1}$.

Again if $|L_{\ell-1}| \le |R_{\ell-1}|$, raise edges of level $\ell-1$ in $L_{\ell-1}$ to level ℓ,

and start scanning level $\ell-1$ edges incident to $V(L_{\ell-1})$.

Note we maintain Invariants.

(Raising levels of edges does not harm Inv I, may mess with Inv II, so that is why we raise levels carefully ensuring that size of comp halves when levels rise)

Time Analysis:

We were kinda vague about how we maintain the forests. Basically each F_i is maintained by Sleator/Tarjan Link/Cut trees. When edge added, use Link. When delete, use Cut. And then for each edge during the scan, use membership tests of L/C trees to test if edge goes across (and use Link if it does).

↖ oh, size of subtree calculations also OK in L/C trees.

Fact: Amortized cost of these operations $O(\lg n)$.

. Each edge ~~handled~~ charged $O(\lg n)$ times, ~~since~~ $\Rightarrow O(\lg^2 n)$ per update.

. query time: check if (x,y) in same component, $O(\lg n)$ time.

Takeaway Ideas: "charge to smaller side". for Holm et al algo.

: "power of XOR" and "power of sampling" for KKM.

: clustering and tree separator for Frederikson.