November 1, 2023 10:50 AM

GRAPHS

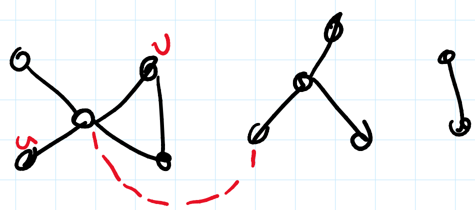
Problem (Dynamic Connectivity)

maintain undirected graph $G = (V, E)$ to support

connectivity queries: given $u, v \in V$, are u & v connected?

insert edge

delete edge



Rmk: insert-only case

$\Rightarrow$ union-find

$O(\alpha(n))$ amort. update/query

fully dynamic?

Without amort:

Frederickson '83: $O(\sqrt{m})$ update time, $O(1)$ query time

Eppstein et al. '92: $O(\sqrt{n})$ "

Kapron-King-Mountjoy '13: $O(\log^5 n)$ rand. (assume obliv. adversary)

Chuzhoy, Gao, Li, Nanongkai, Peng, Saranurak '20: $O(n^{\varepsilon})$ det. (no assumption)

with amortization:

- Henzinger-King '97: $O(n^{1/3})$ rand. update, $O(1)$ query

Henzinger-King '95: $O(\log^3 n)$ rand. update, $O\left(\frac{\log n}{\log\log n}\right)$ query

Henzinger-Thorup '96: $O(\log^2 n)$ rand. update "

Holm-de Lichtenberg-Thorup '98: $O(\log^2 n)$ det., "

Thorup '00: $O(\log n (\log\log n)^3)$ rand. $O\left(\frac{\log n}{\log\log\log n}\right)$ query

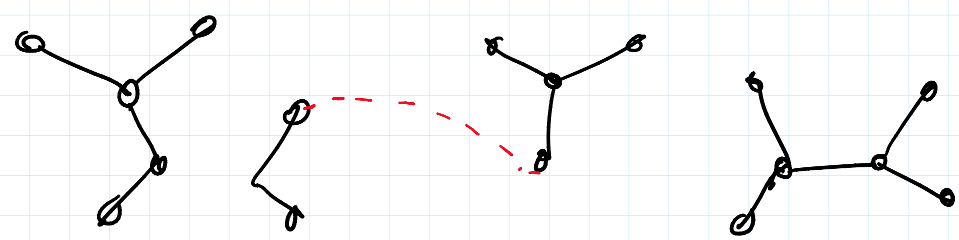
Thorup '00: $O(\log n (\log\log n)^3)$ rand. $O\left(\frac{\log n}{\log\log\log n}\right)$ query

– – '17: $O(\log n(\log\log n)^2)$

Warm-Up: Case of Forest

acyclic graphs

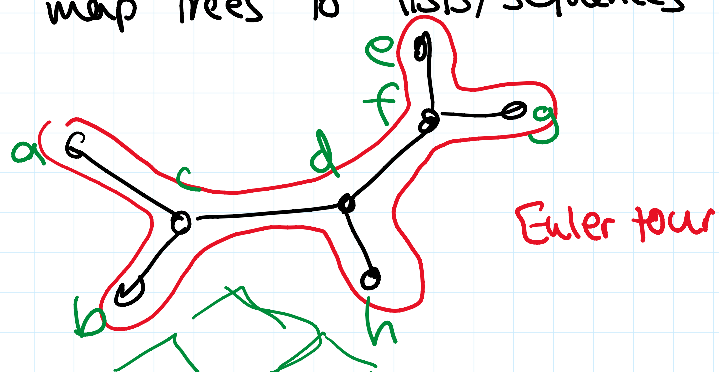
(Henzinger-King)
"E-T Tree"



insert: link 2 trees

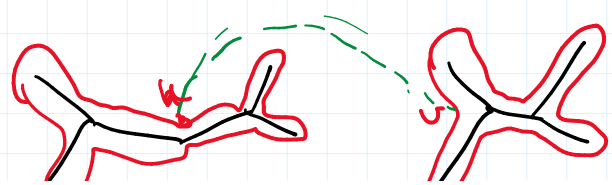
delete: cut tree into 2

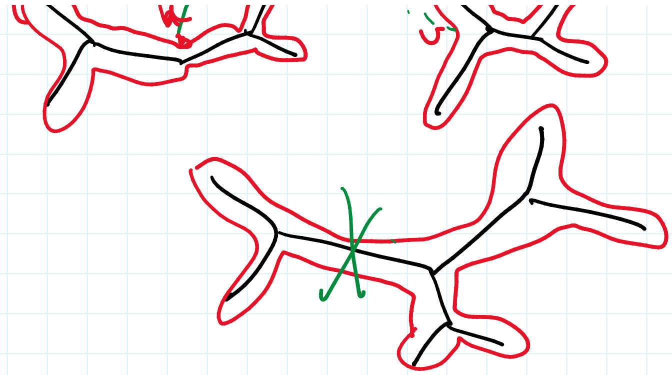
idea - map trees to lists/sequences – –

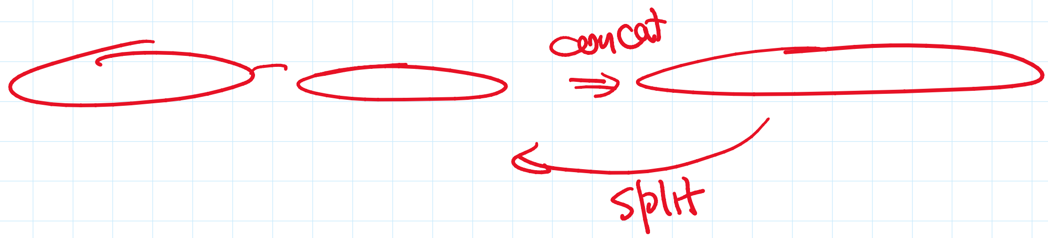


a c b c d h d f g f e f d c a

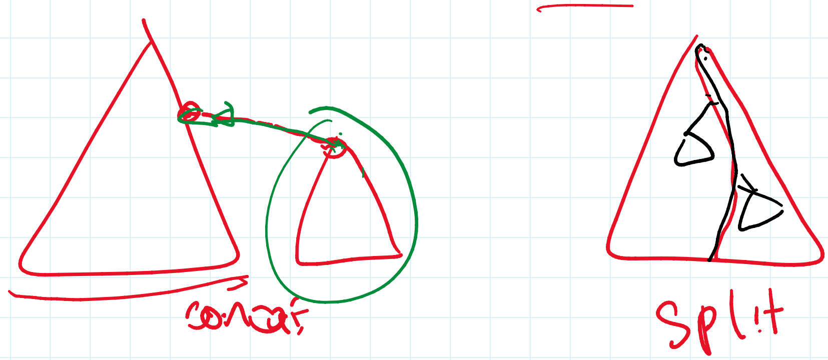
link/cut on trees reduces to $O(1)$ splits/concats on lists







concats/splits $O(\log n)$ time
by balanced search trees
e.g. 2-3 trees



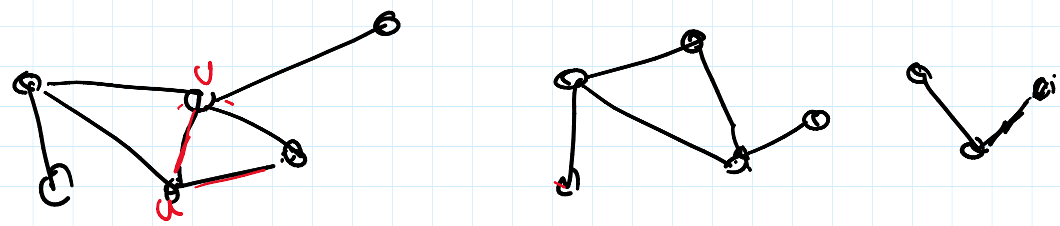
query(u,v): find u's root & v's root & check equal. ← $O(\log n)$

⇒ update/query $\boxed{O(\log n)}$ time

(can reduce query time $O\left(\frac{\log n}{\log\log n}\right)$ if update time $O(\log^{1+\epsilon} n)$ by increasing degs to $(\log^{\epsilon} n)$).

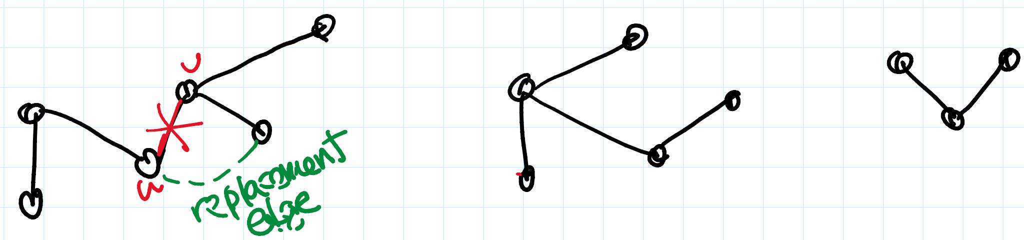
General Case. (Holm, de Lichtenberg, Thorup '98)

G



idea – maintain a spanning forest F of G

F



store F in ET trees $\Rightarrow$ query $\boxed{O(\log n)}$

insert(uv): if u, v in same tree, ok ✓
else add uv to F (link)

delete(uv): if $uv \notin F$, ok ✓
else remove uv from F (cut)
find a replacement edge. xy
add xy to F (link)
(if exists)

HOW??

"Hierarchical" Method

(initially, s = n)

Assume G has max component size $\leq s$

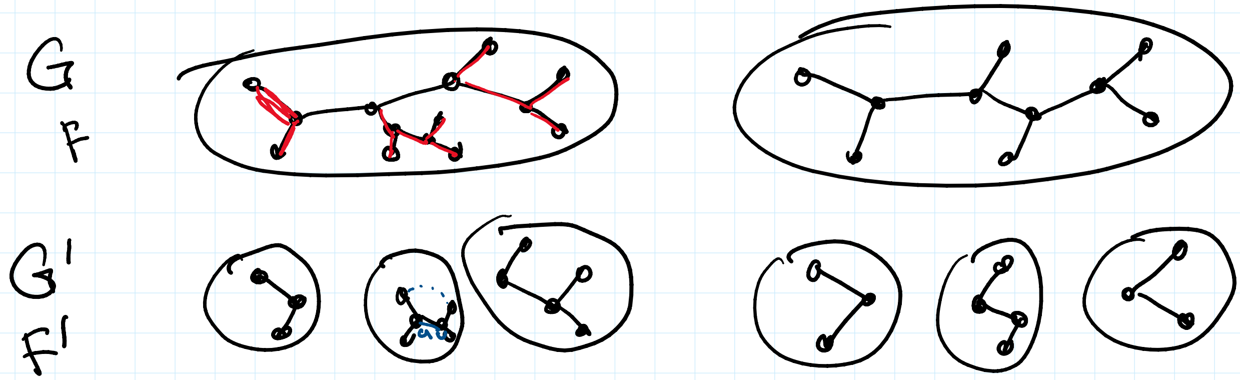
Maintain a subgraph $G' \subseteq G$
with max comp size $\leq s/2$

$\Rightarrow$ # levels in recursion $= O(\log n)$

with max comp size $\le s/2$

& spanning forest F' of G' with $F' \subseteq F$
($F' = F \cap G$)

recurse on (G', F')



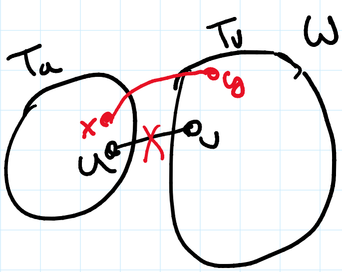
Replace-edge (G, F, uv):

// Case 1: $\exists$ replacement edge xy in G'

if $uv \in F'$, $\quad xy \leftarrow$ Replace-edge(G', F', uv)
if xy exists, return xy ✓

// Case 2: $\exists$ replacement edge xy in $G - G'$

let T_u, T_v be trees of F containing u, v
w.l.o.g. say $|T_u| \le |T_v|$.

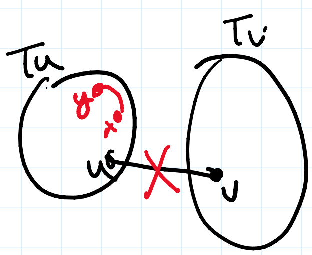


add $T_u^{-F'}$ to F' & G' // $|T_u| \le s/2$

for each $xy \in G - G'$ with $x \in T_u$

if $y \notin T_u$ return xy ✓

else add xy to G' // T_u remains a tree in F'



Analysis: over m updates,

Analysis: over m updates,

\# edges added to $G' \le m$.

total \# links/cuts $= O(m \log n)$ ← because $\log n$ levels of recursion

$\Rightarrow$ total update time $= O(m \cdot \log^2 n)$

$\Rightarrow$ amort update $\boxed{O(\log^2 n)}$

query $\boxed{O(\log n)}$