6.851

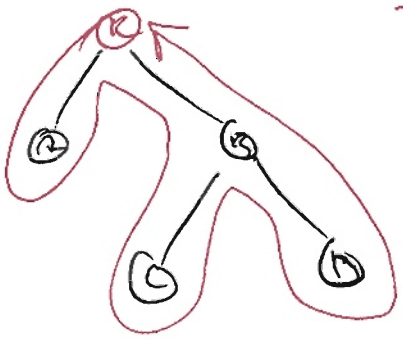
Lecture 17

André Schulz Notes

Dynamic Graphs

Euler-Tour-Trees [Henzinger & King '95]

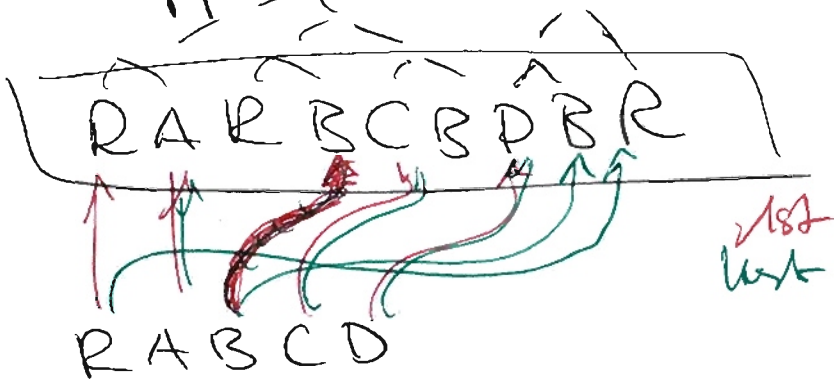
- alternative to link-cut-trees
- simpler, but difficult to store information on paths in ds
  ↳ better for storing information of subtrees
- Idea: Store the Euler-tour (DFS sequence) of the tree



Tour: RARBCBDBR

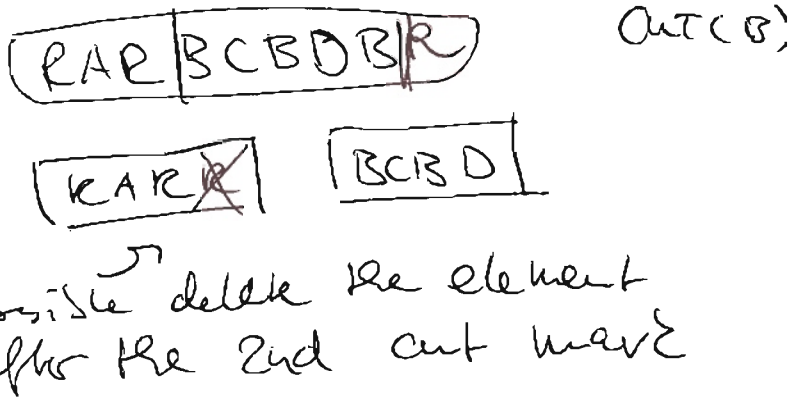
↑ Store this tour in a BST

- each node has a pointer to its first & last appearance in the tour



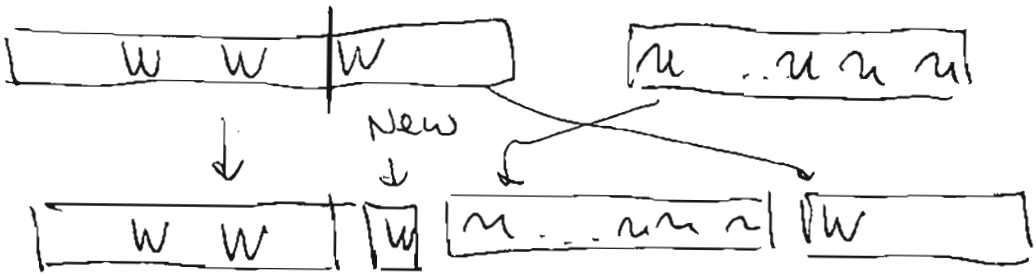
- Ops: Find root(v) $(up)^*(left)^*$
walk to the root of the tour BST

Cut(v)
Split tour BST at first and last appearance of v, glue the two parts together



link (v,w):

Split tour BST of w before last visit of w, and insert tour BST of v



At most 3 Split/merge Opn -> $O(\log n)$/Opn

~~$O(\log n)$~~ for BST

$O(\log n)$

We can speed up by storing the second Tree as B-Tree with fan-out $\Theta(\log w)$

Dynamic Connectivity

-> $O(\log^3 / \log\log w)$* update
$O(\log / \log\log n)$** find

- operations:
  - INSERT / DELETE edges
  - INSERT / DELETE vertices without edges
  - Connectivity (v,w): $\exists\, v \to^* w$ in G?

* depth $\log n / \log\log n$ branching
** depth $\log n$

Known bounds

- $O(\log n)$ for trees [link-cut / Euler tour]
- $O(\log n)$ for planar graphs [Eppstein et al. '92]

- Open: $O(\log n)$ general graphs
  - $O(\log n (\log\log n)^3)$ update*
    $O(\log n / \log\log\log n)$ query* [Thorup 2000]
  - $O(\log^2 n)$ update, $O(\log n / \log\log n)$ query
    TODAY
    [Holm, de Lichtenberg, Thorup 2001]

- $O(x \log n)$ update $\Rightarrow \Omega(\log n / \log x)$ query
- $O(x \log n)$ query $\Rightarrow \Omega(\log n / \log x)$ update
  [Fredman, Pătraşcu 04]

- $O(\sqrt{n})$ worst case update, $O(1)$ query*
  [Eppstein, Galil, Italiano, Nissenzweig '97]
- Open: polylog W.C. update & polylog query

* match lower bounds

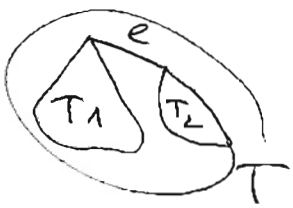
DYNAMIC CONNECTIVITY IN $O(\log^2 n)$

(Holm et al. 2001)

IDEA: STORE Spanning Forest of G as Euler Tour

↳ Queries: 2 x FindRoot + Compare the roots

↳ Insert (u,v): • if $(u,v) \in$ Tree, update adjacency information
• "store edge"
• if u,v disconnected, add edge to spanning forest (join)



↳ But what about Delete?
- if edge $\notin$ Forest just delete

How to do this?
- if edge $\in$ TreeT & no other edge between the subtrees, split the tree $T \to T_1 / T_2$
- otherwise delete edge and join $T_1 \leftrightarrow T_2$ with replacement edge

- To find replacement edges we store a hierarchy of spanning forests
- every edge gets a level:
  - when inserted → level(e) = log n
  - level only decreases, always $\in \{t, \ldots, 0\}$
- G_i = subgraph from all edges with level $\leq i$
- F_i = Spanning Forest of G_i

INVARIANT 1: Every conn. component of G_i has $\leq 2^i$ vertices

INVARIANT 2: $F_0 \subseteq F_1 \subseteq \ldots F_{\log n}$, i.e.

$F_i = F_{\log n} \cap G_i$ & F_i is minimum SF for G_i

Delete $(e=(u,v))$

- update adjacency information
- if $e \in F_{\log n}$:
  - delete e from $F_k \ldots F_{\log(n)}$ (A)
    where $k = \text{level}(e)$

- reconnect v & u by replacement edge if possible

replacement must be level > k because of Inv 2

(A) FOR $i = level(e) \dots \log(n)$ ← try to add edge with min level to assure Inv 2

- let T_v, T_u trees of F_i that contain u/v, assume $|T_v| \leq |T_u|$
- Due to invariant 1: $|T_v| + |T_u| \leq 2^i$

$$\Rightarrow |T_v| \leq 2^{i-1}$$

↳ We can afford to push all of T_v * to level $(i-1)$ (D)

(C) - for each edge $(x,y) \in T_v$ at level i

found replacement ↓

↳ if $y \in T_u$:

add (x,y) to $F_i, F_{i-1} \dots F_{\log n}$ (B)

↳ DONE

↳ else level $(x,y) \leftarrow i-1$ (D)

*(Invariant 1 tells us that we never have to push below level 0)

Costs

(A) Delete $i \leq \log n$ Trees : $O(\log^2 n)$

(B) Insert $i \leq \log n$ Trees : $O(\log^2 n)$

(C) ~~(Loop $\leq \log n$)~~ · (Find next (x,y))

Charged to level drop

(D) Prop a level : $O(\log n)$, can only happen $O(\log n)$ times / edge $\Rightarrow O(\log^2 n)$ } amortized over levels

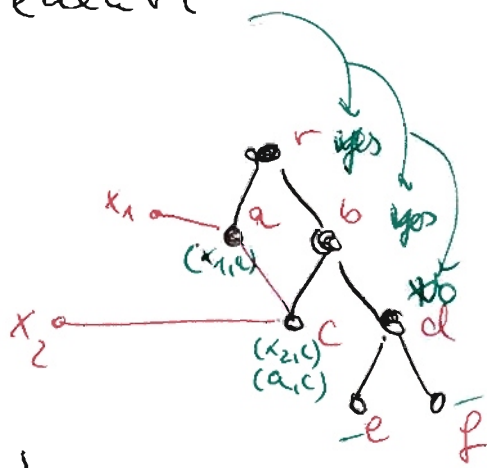
We have to augment the Euler Trees, such that

→ they can report if $|T_u| > |T_v|$ (easy)

→ they can report all non-tree edges with level i, incident to T_v

↳ Store in every node if there are level i non-tree edges beneath

↳ traverse those part of the tree that contains level i non-tree edges (takes $O(\log n)$ / edge)



→ update Adjacency information: $O(\log n)$

↳ (C) => $O(\log^2 n)$

($n \leq \log n$ trees / used, delete)

Related work

① Simpler dynamic connectivity

- incremental (insertions only)
  - $O(\alpha)$ amortized via union find
  - worst case: $\Theta(x)$ updates $\Rightarrow \Theta(\log n / \log x)$ queries
- decremental (deletions only)
  - $O(m \log n + n \operatorname{polylog} n)$ for m updates
    $O(1)$ query
    [Thorup 1999 JACM]

② Other dynamic graph problems

- minimum spanning forest (MST & connected components)
  - $O(\log^4 n)$ update [Holm, de Lichtenberg, Thorup 2001, JACM]
  - worst case $O(\sqrt{n})$ update
  - plane graphs $O(\log n)$ } [Eppstein et al. '92 '97 JACM]
- bipartiteness (is G 2-colorable)
  - reducible to MST

- planarity testing (insert edge & report violation)
  - $O(n^{2/3})$ [Galil, Italiano, Sarnak '97 JACM]
  - fixed embedding $O(\log^2 n)$ [Eppstein et al 97 JACM]
  - incremental: $O(n\,\alpha(m,n)+n)$ [La Poutre 94 SODA]

OPEN: test for fixed Minor

- k-connectivity (vertex/edge)

(How many vertices / edges do I have to delete to disconnect the graph)

↑ Can be also expressed in terms of disjoint paths between vertex-pairs by Menger's Thm

FOR NOW: Only consider # disjoint paths between 2 vertices

- $O(\text{polylog } n)$ for $k=2$ [Holm et al 2001 JACM]
- planar decremental: $O(\log^2 n)$ for 3-edge connect. [Giammarresi, Italiano Algorithmica 96]
- worst case:
  - $O(\sqrt{n})$ [Eppstein et al '97] JACM 2 edge conn.
  - $O(n)$ 2 vertex, 3 vertex conn.
  - $O(n^{3/2})$ 3 edge conn.
  - $O(n\,\alpha(n))$ $k=4$
  - $O(n \log n)$ for $O(1)$-edge conn.

OPEN: polylog n for $k=O(1)$? $k=$ polylog n?

- Whole graph (~~whole~~ ≙ min-cut = max flow)
  - $O(\sqrt{n}\ \text{polylog}\ n)$ for $O(\text{polylog}\ n)$-edge-conn. (& min cut up to that size) [Thorup 2001 STOC]

OPEN: polylog n for $k = O(1)$? $k = \text{polylog}\ n$?

(3) Dynamic directed graphs

- Transitive closure (path from $u \to v$)
  - bulk update := (insert / delete vertex + incident edges)
  - $O(n^2)$ amortized bulk update, $O(1)$ w.c. query
    [Demetrescu, Italiano 2000 Focs | Roditty 2003 SODA]
  - Same but worst case [Sankowski 2004 Focs]

OPEN $O(n^2)$ update worst case

  - $O(M\sqrt{n} \cdot t)$ amortized bulk update, $O(\sqrt{n}/t)$ w.c. queries ($\uparrow \in O(\sqrt{n})$)
    [Roditty, Zwick 2002 Focs]
  - $O(n + n\log n)$ amort. bulk update, $O(n)$ w.c. queries
    [Roditty Zwick 2004 STOC]

OPEN: Full Trade-off with update·query $= O(n \cdot n)$ or $O(n^2)$

  - acyclic: $O(n^{1.575} \cdot t)$ update $O(n^{0.575}/t)$ query
  - decremental: $O(n)$ amort. update, $O(1)$ w.c. queries — $t = O(\sqrt{n})$

[Demetrescu & Italiano 2000]