Lecture 18

Tree decompositions

- preferred paths → Tango trees, link-cut trees
- heavy-light decomp. → link-cut trees
- * ART/leaf-trimming decomp. → level-ancestors
- * separator decomp.

TODAY

many applications

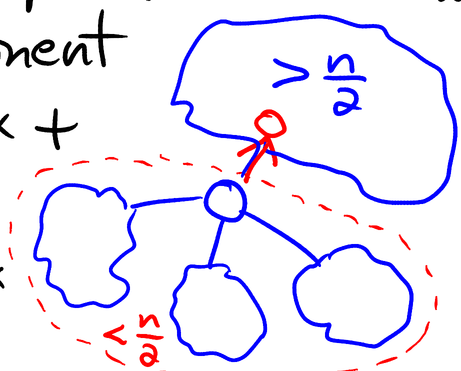
Separator theorem on trees: [Jordan 1864 – "Sur les assemblages de lignes" – Journal Reine Angew. Math.]

Any tree on n vertices has a vertex whose removal disconnects the tree into components of size $\leq \frac{n}{2}$

Proof:
- pick any vertex
- if not done, exactly one component has size $> \frac{n}{2}$
- walk one step into that component
- new component of old vertex + other pieces has $< \frac{n}{2}$ vertices

$\Rightarrow$ never go back to old vertex

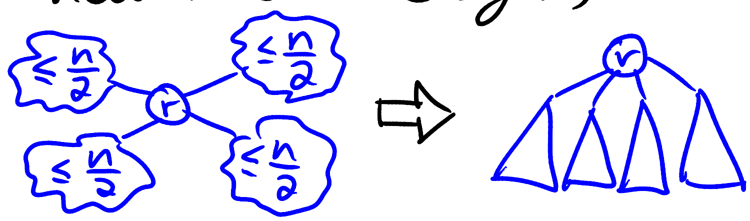
$\Rightarrow$ terminate □



Separator decomposition:

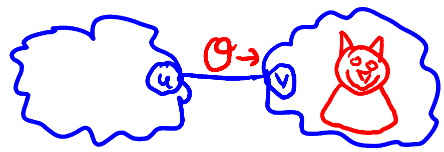
- apply separator theorem → root of new tree
- recurse on components → children subtrees

⇒ depth of new tree $= O(\lg n)$

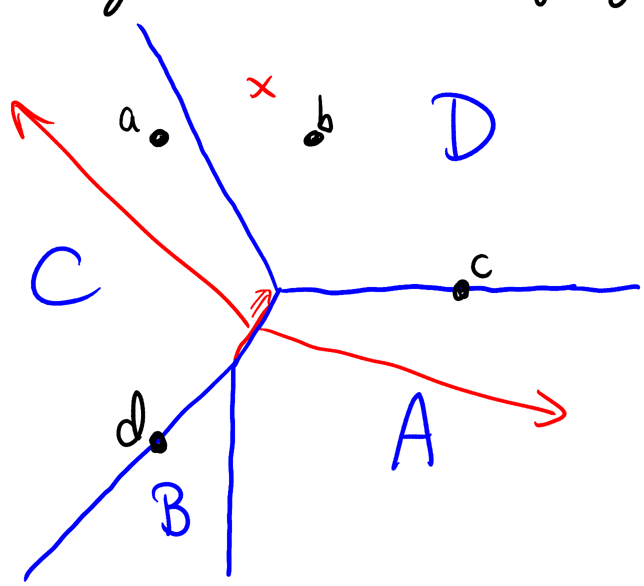


Application: Oracle Search [Aronov et al. - LATIN 2006]

- suppose you're looking for a cat (node) in a tree
- if you ask Oracle about an edge (u, v), it tells you which subtree contains cat



- separator decomposition tree lets you find cat in $O(\lg n)$ calls to Oracle (if bounded degree)
- example: maintain set of points subject to finding farthest from query point



ART decomposition:

- define bottom tree rooted at each maximally high node with $\le \lg n$ leaves below
  $\Rightarrow$ compressed size $O(\lg n)$, disjoint
- top tree on remaining nodes $\Rightarrow \le n/\lg n$ leaves
  (charge each top leaf to $> \lg n$ leaves in subtree)
- recurse in top tree

Marked ancestor problem [Alstrup, Husfeldt, Rauhe - FOCS 1997]

- rooted tree ~here, static
- each node either marked or unmarked
- updates: mark(v) & unmark(v)
- query: lowest marked ancestor of v

Bounds:

* $O\left(\frac{\lg n}{\lg\lg n}\right)$ query, $O(\lg\lg n)$ update TODAY
- $\Omega\left(\frac{\lg n}{\lg\lg n}\right)$ query [chronogram technique]
- $\Omega(\lg\lg n)$ update, even in path:
  colored predecessor problem with $u=n$

Application: dynamic method dispatching in OOP

- tree = inheritance among classes
- mark = class implements method x
- query = call to x

$\boxed{\text{OPEN: DAGs from multiple inheritance}}$

Marked ancestor upper bound:

- recursive ART decomposition

$\Rightarrow O(\frac{\lg n}{\lg \lg n})$ levels of recursion

Each bottom tree:

- maintain bitvector of which nonbranching paths have a marked node, ordered by depth
- each node stores bitmask of its ancestor paths

$\Rightarrow$ in $O(1)$ time (mask + LSB), find right path

- maintain predecessor DS on each nonbranching path

$\Rightarrow O(\lg \lg n)$ time/op.: $u = n$

- only used at end of query: if no marked ancestor in bottom tree, recurse with parent(root) in top tree

$\Rightarrow O(\frac{\lg n}{\lg \lg n} + \cancel{\lg \lg n})$ query

- each node stores recursion depth at which it's in bottom tree

$\Rightarrow$ update = predecessor update for path + bitvector update for bottom tree

$\Rightarrow O(\lg \lg n)$

Decremental connectivity in a tree: $O(1)$ amortized
(assuming all $n-1$ edges deleted)

① $O(\lg n)$ amortized update (or use link-cut/Euler-tour)
- each node stores explicit component i.d.
- delete(v, w):
  - run DFS from both v & w in parallel
  - stop when one DFS stops ⇒ smaller component
  - update all nodes in smaller component to new i.d.
- component containing updated node shrinks by $2\times$

⇒ $O(\lg n)$ updates to any node
- can also store mapping from i.d. to root of component

② $O(1)$ amortized for path
- split into chunks of length $\lg n$ ⇒ $\le \frac{n}{\lg n}$ chunks
- store each chunk as bitvector
- use $O(\lg n)$ structure to store which chunks have cut
- query: find right chunk, shift + LSB within chunk

③ $O(1)$ amortized for top tree
- use $O(\lg n)$ structure on $O(\frac{n}{\lg n})$ nonbranching paths
- use path structure on each nonbranching path

④ $O(1)$ amortized for bottom trees
- maintain bitvector of which $\overbrace{\text{nonbranching paths}}^{\le \lg n}$ have cut
- preprocess mask for ancestors of each node (1 word)
- order bits by depth ⇒ query = mask + shift + LSB + path
- use path structure on each nonbranching path

↳ NONRECURSIVE ART DECOMPOSITION