link/cut trees lecture

Define the problem:

Maintain a collection of rooted trees, each node of which has a cost under the following operations:

find cost(v) — highest one
find root(v)
find min(v)
add cost(v, x)
link (v, w) — w is a root of a tree
cut (v) — remove edge

Applications to network flow:

Dinitz algorithm

sequence of layered networks
find blocking flow on each one

sequence of augmenting paths
(each eliminates an edge)
nm time to block flow

reduce this to m log n

Shift to a different type of tree,

Splay tree.

A binary search tree to represent a list. (symmetric order traversal of tree is the list) define predecessor & successor.

Rotates maintain symm. order.

Splay operation: Rotate an item to the root.

Examples: .....

Assign each item a <u>weight</u> $W(X)$
the <u>size</u> of a node = total wt of its desc. $S(X)$
Access lemma: the amortized # rotations splay an item to the root is:

$$3(\log S(r) + \log S(x)) + 1$$

Example, 9. 1. etc.

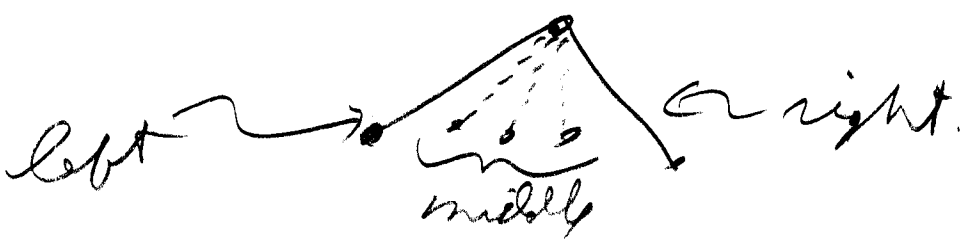
Virtual trees:

T is a tree of our forest

V is a virtual tree representing T.

V has the same nodes of T, but different structure.

in V, a node may have left, right, or non-middle children

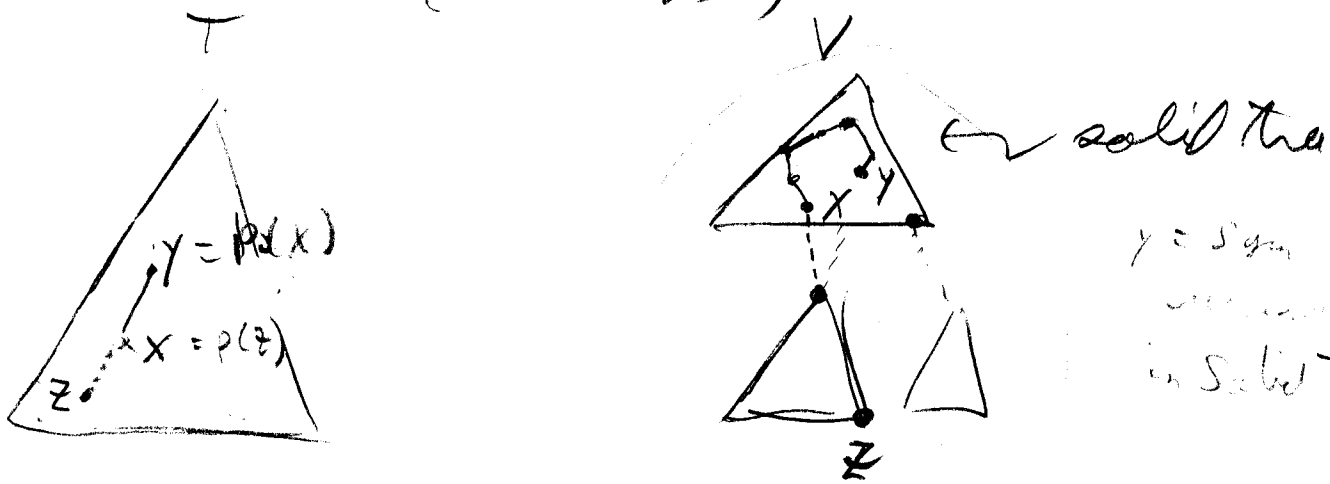


Middle edges "dashed", others "solid."

solid subtrees are binary trees.

real tree.

Relation between V and T
(two cases)



y = sign [illegible] in solid tree

most predecessor in representation
① selection of virtual tree (paths in T)
② binary tree structure

Parent & left & right pointers to store V.

cost(x) = the cost of node x.

min cost(x) = the minimum cost of any descendant of x in the same solid subtree.

could store these directly, then find cost and find min will be efficient. add cost won't.

show how to traverse the virtual tree to find these. also note that rotations are easy to do.

(5)

to make add cost fast, we store these costs in "difference form".

$$\Delta\text{cost}(x) = \begin{cases} \text{cost}(x) & \text{if } x \text{ is a solid subtree root} \\ \text{cost}(x) - \text{cost}(p(x)) & \text{otherwise} \end{cases}$$

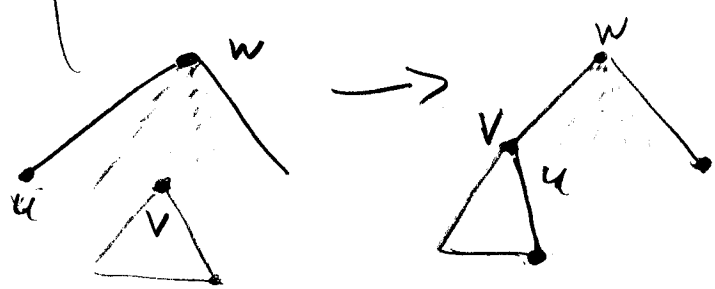
virtual tree parent

$$\Delta\text{min}(x) = \text{cost}(x) - \text{min cost}(x)$$

the virtual tree can be modified with two constant time primitives

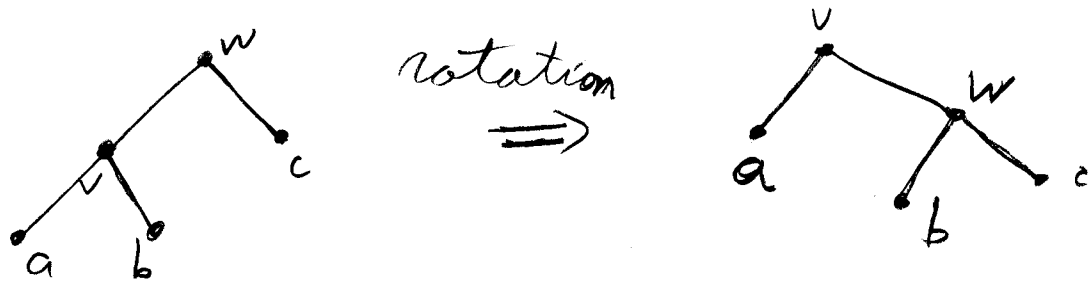
rotation & splicing.

same as rotation for solid trees.



the actual / virtual correspondence is maintained.

we must show how to update Δcost and Δmin.



how do we update Δ cost & Δ min

$$\Delta cost'(v) = cost(v) - cost(p(w))$$
$$= cost(v) - cost(w) + cost(w) - cost(p(w))$$
$$= \Delta cost(v) + \Delta cost(w)$$

$$\Delta cost'(w) = cost(w) - cost(v)$$
$$= -\Delta cost(v)$$

$$\Delta cost'(b) = cost(b) - cost(w)$$
$$= cost(b) - cost(v) + cost(v) - cost(w)$$
$$= \Delta cost'(b) + \Delta cost(v)$$

$$\Delta min'(w) = cost(w) - min\ cost'(w)$$
$$= cost(w) - min\{cost(w), min\ cost(b), min\ cost(c)\}$$
$$= max\{0, cost(w) - mincost(b), cost(w) - mincost(c)\}$$

$$= \max\{0,\ \mathrm{cost}(w) - \mathrm{cost}(b) + \mathrm{cost}(b) - \mathrm{mincost}(b),\ \mathrm{cost}(w) - \mathrm{cost}(c) + \mathrm{cost}(c) - \mathrm{mincost}(c)\}$$

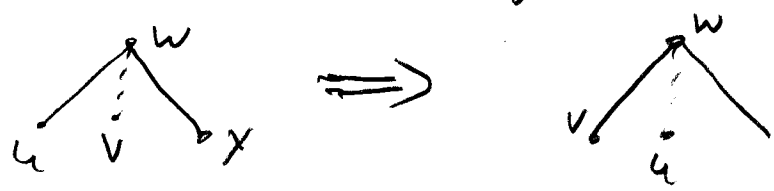
$$= \max(0,\ -\Delta\mathrm{cost}'(b) + \Delta\min(b),\ -\Delta\mathrm{cost}(c) + \Delta\min(c)\}$$

$$\Delta\min'(v) = \max\{0,\ \Delta\min(a) - \Delta\mathrm{cost}(a),\ \Delta\min'(w) - \Delta\mathrm{cost}'(u))$$

similar.

---

similar updates must for splice.



$$\Delta\mathrm{cost}'(v) = \Delta\mathrm{cost}(v) - \Delta\mathrm{cost}(w)$$
$$\Delta\mathrm{cost}'(u) = \Delta\mathrm{cost}(u) + \Delta\mathrm{cost}(w)$$

$$\Delta'\min(w) = \max\{0,\ \Delta\min(v) - \Delta\mathrm{cost}'(v),\ \Delta\min(x),\ -\Delta\mathrm{cost}(x)\}$$

The splay operation on a node in a virtual tree:

a sequence of rotations & splices.

3 passes.

Pass 1: Walk up from x, splaying in each solid subtree.

Pass 2: Walk up the (now) dashed path from x, splicing all the way

Pass 3: splay x to the root of its solid tree (the whole tree)

The amortized time of splay is $O(\log n)$ (an n node tree)

Proof: Assign a "mass" of one to each node.

the weight of a node in the V tree is # number of descendants below the dashed edge [illegible] w x.

$\Rightarrow$ the size of a node = # descendents of it in V

later

Proof: Recall the access lemma:

AL: Amort # rot is $\le 3(r(\text{root}) - r(x)) + 1$

the proof uses the $\Phi = \sum_x r(x)$

where $r(x) = \text{rank}(x) = \log_2 \text{size}(x)$

Let's modify this using a potential that is twice this. $\Phi = 2 \sum r(x)$.

For the zig-zig & zig-zag steps this at most doubles the amortized cost bound. So its $6(r'(x) - r(x))$ for the step.

In the zig case we get

$2(r'(x) - r(x))$ pays for the $\Delta\Phi$

1 pays for the rotation

$$2(r'(x) - r(x)) + 1 \le 6(r'(x) - r(x)) + 1$$

So we have this new access lemma:

AL': Amort # rot is $\le 6(r(\text{root}) - r(x)) + 1$
where $\Phi = 2 \sum r(x)$.

Proof cont'd:

[see pattern of Pag 8 for weight assignment]

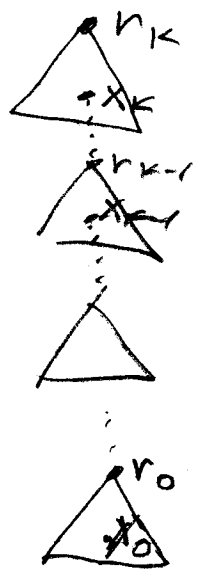
the potential we use in the L/C tree is

$$\Phi = 2 \sum_{\text{all } x \in V} r(x)$$

Lets analyze the amortized cost of each pass of the splay in V.

cost measure: one per rotation
one per splice.
(this covers everything)
Let k be the number of splices.

pass 1: Let the nodes spliced be $x_0 x_1 \cdots x_k$
Let then the roots be $r_0 r_1 \quad r_k$



cost of splay i is

$$6(r(r_i) - r(x_i)) + 1$$

Total is $\sum_{0 \le i \le k} [6(r(r_i) - r(x_i)) + 1]$

$$k+1 + 6 \sum (r(r_i) - r(x_i))$$

note $r(x_i) \ge r(r_{i-1})$ $\forall 0 \le i \le k$ (define $r(r_{-1}) = r(x_0)$)

$$\text{Pass 1} \le k+1 + 6 \sum_{0 \le i \le k} r(r_i) - r(r_{i-1})$$

$$= k+1 + 6\left(r(r_k) - r(r_{-1})\right)$$

$$= k+1 + 6\left(\underset{\log_2 n}{\underset{\parallel}{r(r_k)}} - \underbrace{r(x_0)}_{\le 0}\right)$$

$$\le k+1 + 6 \log_2 n$$

Pass 2: Splicing does not change the Φ. (the sizes of the nodes do not change. (the weights do)

pass 2 = k

---

pass 3: Replays a path of length k.
$\Rightarrow$ the cost is k.

But, by the analysis of $\overbrace{\text{splaying}}^{AL'}$ the amortized cost of pass 3 is at most $6 \log n + 1$

Putting it all together:

In the "amortized sense" we can replace K by $6 \log n + 1$

Pass 1 $\leq 2 + 12 \log n$

pass 2 $\leq 1 + 6 \log n$

Pass 3 $\leq 1 + 6 \log n$

$24 \log n + 4$

QED.

The proof improves this slightly as follows:

1) The cost is defined to be the path length to x = # rotations done. This covers the splices to OLD.

2) $\Rightarrow$ Pass 2 has 0 cost

3) Use the "doubled" access lemma: $2 * \#$ rotations $\leq 6 \log n + 2$

use one half of them to carry the "K" of pass 1. The other half to carry for pass 3.

Thus our total becomes:

$$1 + 6\log n + 6\log n + 2 = 12\log n + 3$$

Note: The paper says 1 here. I think its a bug.

If there are k dashed edges then we get

Pass 1 $\quad O(\log n + k)$

Pass 2 $\quad O(k)$

Pass 3 $\quad O(\log n)$

alternative:
[illegible] at
$O(\log n)$
[illegible] splay [illegible]
QED.

but $\sum k_i \leq$ total pass 3 cost

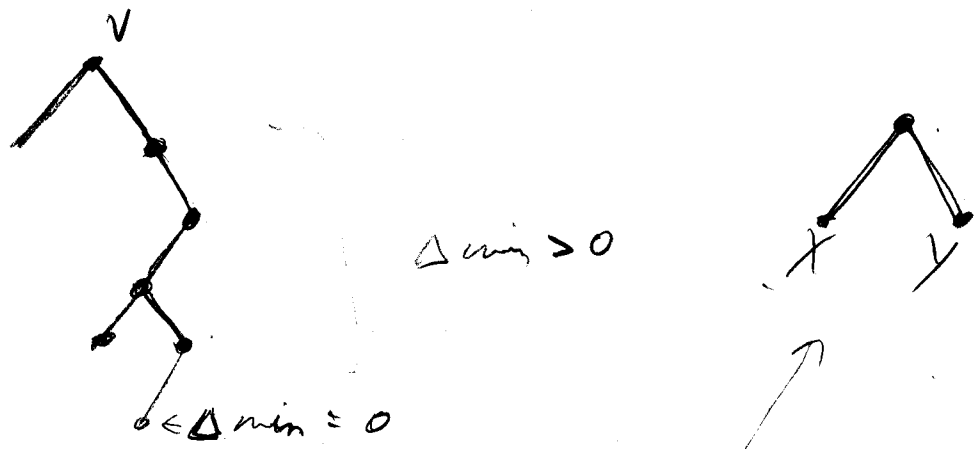
$\Rightarrow$ the operation time is $O(\log n)$
(sort of) QED

How do we use V trees & splay to implement the operations?

find cost (v) : splay at v, and return the cost(v)

find root (v) : splay at v
go to rightmost node of the solid tree say x,
splay x, return x.

find min(v) * splay at v
* walk down the solid tree from v using Δcost and Δmin to the last minimum cost node after v. call it x
* splay at x, and return it.



off by some constant from actual min cost.

$$\Delta cost(x) - \Delta min(x) = cost(x) - cost(P(x)) - cost(x) + mincost(x)$$

$$= mincost(P(x)) - cost(P(x))$$

$$\Rightarrow mincost(x) \le mincost(y)$$

$$\Updownarrow$$

$$\Delta cost(x) - \Delta min(x) \le \Delta cost(y) - mincost(y)$$

easier to explain, then by observing that we can compute and reach [illegible] we can compute its cost() and mincost().

addcost(v, x): splay(v)
add x to Δcost(v)

if left(v) ≠ null
⇒ subtract x from
Δcost(left(v))

link(v w): splay at v and at w

make v a middle child of w

cut(v): splay at v

add Δcost(v) to Δcost(right(v))

& Break the link between v
and right(v).

Thm: A sequence of m operations on n
nodes is O(m log n)

Proof: all the operations have log n
amortized time. the initial potential
is zero. the potential is always ≥ 0.

Q.E.D.