Dynamic trees (Steator and Tarjan 83)
Operations that we do on the trees

Maketree(v)

w = findroot(v)

(v,c) = mincost(v)

addcost(v,c)

link(v,w,r(v,w))

cut(v)

findcost(v)
Simple case -- paths

Assume for a moment that each tree $T$ in the forest is a path. We represent it by a virtual tree which is a simple splay tree.
Findroot(v)

Splay at v, then follow right pointers until you reach the last vertex w on the right path. Return w and splay at w.
Mincost(v)

With every vertex $x$ we record $\text{cost}(x) = $ the cost of the edge $(x, p(x))$

We also record with each vertex $x$ $\text{mincost}(x) = $ minimum of $\text{cost}(y)$ over all descendants $y$ of $x$. 
Mincost(v)

Splay at v and use mincost values to search for the minimum

Notice: we need to update mincost values as we do rotations.
Addcost(v,c)

Rather than storing cost(x) and mincost(x) we will store
\[ \Delta \text{cost}(x) = \text{cost}(x) - \text{cost}(p(x)) \]
\[ \Delta \text{min}(x) = \text{cost}(x) - \text{mincost}(x) \]

Addcost(v,c) :
Splay at v,
\[ \Delta \text{cost}(v) += c \]
\[ \Delta \text{cost(left(v))) -= c \]
similarly update \( \Delta \text{min} \)
Addcost(v,c) (cont)

Notice that now we have to update $\Delta\text{cost}(x)$ and $\Delta\text{min}(x)$ through rotations

$$\Delta\text{cost}'(v) = \Delta\text{cost}(v) + \Delta\text{cost}(w)$$
$$\Delta\text{cost}'(w) = -\Delta\text{cost}(v)$$
$$\Delta\text{cost}'(b) = \Delta\text{cost}(v) + \Delta\text{cost}(b)$$
Addcost(v,c) (cont)

Update \( \Delta \text{min} \):

\[
\Delta \text{min}'(w) = \max \{0, \Delta \text{min}(b) - \Delta \text{cost}'(b), \Delta \text{min}(c) - \Delta \text{cost}(c)\}
\]

\[
\Delta \text{min}'(v) = \max \{0, \Delta \text{min}(a) - \Delta \text{cost}(a), \Delta \text{min}'(w) - \Delta \text{cost}'(w)\}
\]
Link(v,w,c), cut(v)

Translate directly into catenation and split of splay trees if we talk about paths.

Lets do the general case now.
The virtual tree

• We represent each tree $T$ by a virtual tree $V$.

The virtual tree is a binary tree with middle children.

Think of $V$ as partitioned into solid subtrees connected by dashed edges

What is the relation between $V$ and $T$?
Actual tree
Path decomposition

Partition T into disjoint paths
Virtual trees (cont)

Each path in T corresponds to a solid subtree in V

The parent of a vertex x in T is the successor of x (in symmetric order) in its solid subtree or the parent of the solid subtree if x is the last in symmetric order in this subtree.
Virtual trees (cont)
Virtual trees (representation)

Each vertex points to $p(x)$ to its left son $l(x)$ and to its right son $r(x)$.

A vertex can easily decide if it is a left child a right child or a middle child.

Each solid subtree functions like a splay tree.
The general case

Each solid subtree of a virtual tree is a splay tree.

We represent costs essentially as before.

$\Delta \text{cost}(x) = \text{cost}(x) - \text{cost}(\text{p}(x))$ or $\text{cost}(x)$ is $x$ is a root of a solid subtree

$\Delta \text{min}(x) = \text{cost}(x) - \text{mincost}(x)$ (where mincost is the minimum cost within the subtree)
Splicing

Want to change the path decomposition such that v and the root are on the same path.

Let w be the root of a solid subtree and v a middle child of w

\[
\Delta \text{cost}'(v) = \Delta \text{cost}(v) - \Delta \text{cost}(w)
\]

\[
\Delta \text{cost}'(u) = \Delta \text{cost}(u) + \Delta \text{cost}(w)
\]

\[
\Delta \text{min}'(w) = \max\{0, \Delta \text{min}(v) - \Delta \text{cost}'(v), \Delta \text{min}(\text{right}(w)) - \Delta \text{cost}(\text{right}(w))\}
\]
Splicing (cont)

What is the effect on the path decomposition of the real tree?

What is the effect on the path decomposition of the real tree?
Splaying the virtual tree

Let x be the vertex in which we splay.

We do 3 passes:

1) Walk from x to the root and splay within each solid subtree

After the first pass the path from x to the root consists entirely of dashed edges

2) Walk from x to the root and splice at each proper ancestor of x.

Now x and the root are in the same solid subtree

3) Splay at x

Now x is the root of the entire virtual tree.
Dynamic tree operations

\( w = \text{findroot}(v) \) : Splay at \( v \), follow right pointers until reaching the last node \( w \), splay at \( w \), and return \( w \).

\((v,c) = \text{mincost}(v)\) : Splay at \( v \) and use \( \Delta \text{cost} \) and \( \Delta \text{min} \) to follow pointers to the smallest node after \( v \) on its path (its in the right subtree of \( v \)). Let \( w \) be this node, splay at \( w \).

\( \text{addcost}(v,c) \) : Splay at \( v \), increase \( \Delta \text{cost}(v) \) by \( c \) and decrease \( \Delta \text{cost}(\text{left}(v)) \) by \( c \), update \( \Delta \text{min}(v) \)

\( \text{link}(v,w,r(v,w)) \) : Splay at \( v \) and splay at \( w \) and make \( v \) a middle child of \( w \)

\( \text{cut}(v) \) : Splay at \( v \), break the link between \( v \) and \( \text{right}(v) \), set \( \Delta \text{cost}(\text{right}(v)) \) += \( \Delta \text{cost}(v) \)
Dynamic tree (analysis)

It suffices to analyze the amortized time of splay.

An extension of the access lemma.

• Assign weight 1 to each node. The size of a node is the total number of descendants it has in the virtual tree. Rank is the log of the size.

Potential is $c$ times the sum of the ranks for some constant $c$. (So we can charge more than 1 for each rotation)

$$
\begin{align*}
&c \log \left( \frac{s(T_k)}{s(x_k)} \right) + \ldots + c \log \left( \frac{s(T_1)}{s(x_1)} \right) + c \log \left( \frac{s(T_x)}{s(x)} \right) + k \leq \\
&c \log \left( \frac{s(T_k)}{s(x)} \right) + k
\end{align*}
$$
Dynamic tree (analysis)

pass 1 takes $3 \log n + k$

pass 2 takes $k$

pass 3 takes $3 \log n + 1 - (c-1)(k-1)$

$k = \# \text{dashed edges on the path}$
Proof of the access lemma (cont)

(1) zig - zig

\[ \text{amortized time(zig-zig) = } 2 + \Delta \Phi = \]
\[ 2 + r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z) \leq \]
\[ 2 + r'(x) + r'(z) - r(x) - r(y) \leq 2 + r'(x) + r'(z) - r(x) - r(x) = \]
\[ 2 + r(x) - r'(x) + r'(z) - r'(x) + 3(r'(x) - r(x)) \leq \]
\[ 2 + \log(s(x)/s'(x)) + \log(s'(z)/s'(x)) + 3(r'(x) - r(x)) \leq \]
\[ 2 + \log([s'(x)/2]/s'(x)) + \log([s'(x)/2]/s'(x)) + 3(r'(x) - r(x)) = 3(r'(x) - r(x)) \]
Proof of the access lemma (cont)

(2) zig - zag

amortized time(zig-zig) = 2 + ΔΦ =

2 + r’(x) + r’(y) + r’(z) - r(x) - r(y) - r(z) ≤

2 + r’(y) + r’(z) - r(x) - r(y) ≤ 2 + r’(y) + r’(z) - r(x) - r(x) =

2 + r’(y) - r(x) + r’(z) - r(x) + 2(r’(x) - r(x)) ≤

2 + log(s’(y)/s(x)) + log(s’(z)/s(x)) + 2(r’(x) - r(x)) ≤

2 + log([s(x)/2]/s(x)) + log([s(x)/2]/s(x)) + 2(r’(x) - r(x)) ≤ 3(r’(x) - r(x))