Washington University in St. Louis

Engineering

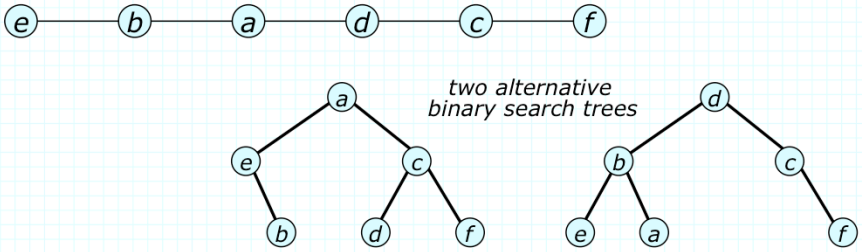
A Fast Implementation of Dynamic Trees

Jon Turner
Computer Science & Engineering
Washington University

www.arl.wustl.edu/~jst

Representing Path Sets as Search Trees

- A path set can be represented by a set of search trees
  - » each node on a path corresponds to a node in a search tree
- No key is used
  - » search tree structure just defines the order of nodes in the path
  - » so, to find the tail of a path, follow right pointers from tree root to a node with no right child



2

Differential Costs

- The node costs are represented implicitly by two fields added to each tree node
- Let *cost*(*x*) be cost of *x* and let *mincost*(*x*) be the minimum cost of any descendant of *x* in the tree.
- Each node has fields $\Delta cost(x)$ and $\Delta min(x)$ defined as

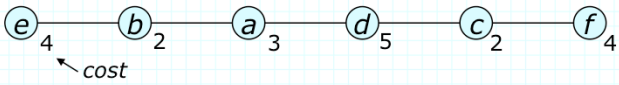
$$\Delta cost(x) = cost(x) - mincost(x)$$

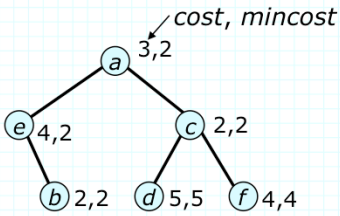
$$\Delta min(x) = \begin{cases} mincost(x) - mincost(p(x)) & \text{if } x \text{ has a parent} \\ mincost(x) & \text{otherwise} \end{cases}$$

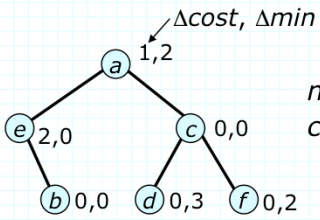
- *mincost*(*x*)=sum of Δmin fields of nodes on the path from *x* to the root, and *cost*(*x*)=*mincost*(*x*)+$\Delta cost(x)$
- This representation makes it possible to change cost of all nodes in subtree by the same amount in *O*(1) time

3

Example of Path Set Representation







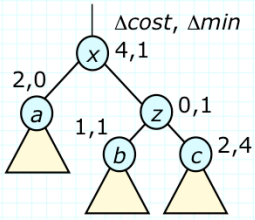
$mincost(d)=\Delta min(a)+\Delta min(c)+\Delta min(d)=5$

$cost(d)=mincost(d)+\Delta cost(d)=5$

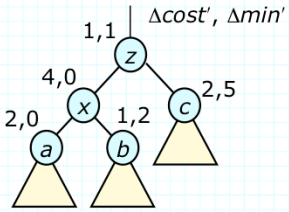
4

Updating Costs During Rotations

- Rotation operations must update differential cost fields in order to maintain actual cost values



	a	b	c	x	z
cost	4	5	9	6	3
mincost	2	4	7	2	3
mincost′	2	4	7	2	2



$\Delta cost'(z) = \Delta cost(z) + \Delta min(z)$

$\Delta min'(z) = \Delta min(x)$

$\Delta cost'(x) = \Delta cost(x) - \Delta min'(x)$

$\Delta min'(x) = \min\{\Delta cost(x), \Delta min(a)^{*}, \Delta min(b) + \Delta min(z)^{\dagger}\}$

$\Delta min'(a) = \Delta min(a) - \Delta min'(x)$

$\Delta min'(b) = \Delta min(b) - \Delta min'(x) + \Delta min(z)$

$\Delta min'(c) = \Delta min(c) + \Delta min(z)$

*drop term if *a* is missing

†drop term if *b* is missing

5

Implementing Path Set Ops

```
node function findtail(path p)
    do right(p) ≠ null ⇒ p := right(p) od;
    return splay(p);
end;
procedure addpathcost(path p, real x)
    Δmin(p) := Δmin(p) + x;
end;
[path, path] function split(item i, path s)
    path s1,s2;
    splay(i);
    s1,s2 := left(i),right(i); p(s1),p(s2) := null;
    left(i), right(i) := null;
    Δmin(s1) := Δmin(s1) + Δmin(i); Δmin(s2) := Δmin(s2) + Δmin(i);
    Δmin(i) := Δmin(i) + Δcost(i); Δcost(i) := 0;
    return [s1,s2];
end;
```

time bounded by number of splay steps

constant time

time bounded by number of splay steps

6

```
path function findpath(node v)
    x = v; do p(x) ≠ null ⇒ x = p(x) od; splay(v); return x
end;
[node, real] function findpathcost(path p);
    do right(p)≠null and Δmin(right(p))=0 ⇒
        p := right(p)
     | (right(p)=null or Δmin(right(p))>0) and Δcost(p)>0 ⇒
        p := left(p)
    od;
    p := splay(p);
    return [p, Δmin(p)];
end;
path function join(path r,v,q);
    real costv;
    left(v),right(v) := r,q;   p(r),p(q) := v;
    costv:= Δmin(v);
    Δmin(v) := min{Δmin(r), Δmin(v), Δmin(q)};
    Δcost(v) := costv – Δmin(v);
    Δmin(r) := Δmin(r) – Δmin(v);  Δmin(q) := Δmin(q) – Δmin(v);
    return v;
end;
```

time bounded by number of splay steps

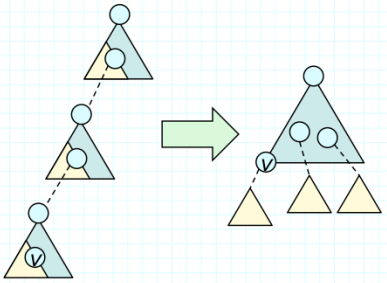
ditto

constant time

Washington University in St. Louis — Engineering

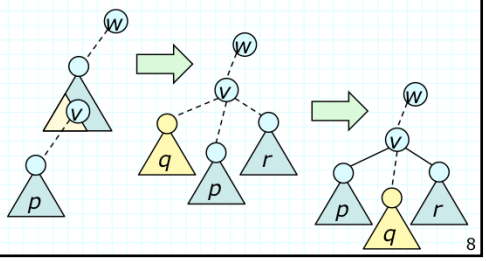
Revisiting Expose

```
path function expose(node v);
    path p; p := null;
    do v ≠ null ⇒
        [p,v] := splice(p,v);
    od;
    succ(p) := null;
    return p;
end;
```




```
[path, node] function splice(path p, node v);
    path q,r; node w;
    w := succ(findpath(v));
    [q,r] := split(v);
    if q≠null ⇒ succ(q) := v fi;
    return [join(p,v,r),w]
end
```

time bounded by number of splay steps



8

Analysis of Dynamic Trees

- *Theorem 5.2*. A sequence of m dynamic tree ops on a set of trees with n nodes can be done in $O(m \log n)$ time

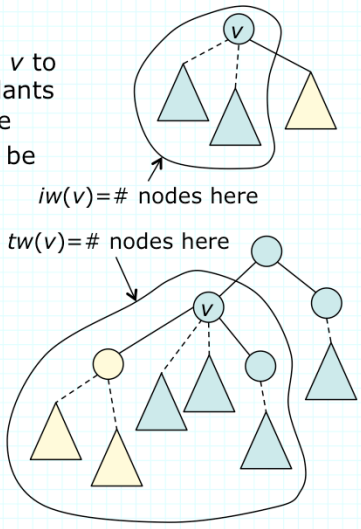
  Proof. Running time is proportional to m plus the number of splices (which we already know is $O(m \log n)$) plus number of splay steps
  - » account for the splay steps using a credit-based analysis

  Types of trees
  - » define an *actual tree* to be the abstract tree implemented by the dynamic trees data structure
  - » define a *virtual tree* to be the linked set of binary search trees that implements an actual tree
  - » define a *solid tree* to be one of the binary search trees in a virtual tree

9

Node weights

- define the *individual weight, iw*(*v*) of a node *v* to be 1 plus the number of actual tree descendants of *v* in subtrees linked to *v* by a dashed edge
- define the *total weight*, *tw*(*v*) of a node *v* to be the sum of the individual weights of the descendants of *v* in its *solid tree*
- note: *tw*(*v*) is the number of descendants of *v* in the virtual tree containing *v*; consequently, *tw*(*v*)≤*n* for all nodes



Let *rank*(*v*) = ⌊lg *tw*(*v*)⌋ and define the following credit invariant

every node *x* contains *rank*(*x*) credits

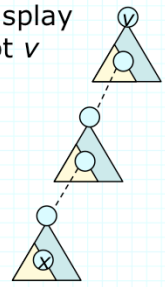
For each tree op, use one credit per splay step plus any additional credits needed to maintain the credit invariant

Lemma 4.2, which bounds the number of credits needed to perform a splay operation while maintaining a credit invariant, holds for this credit invariant, as well as for the simpler invariant used earlier

10

Now, consider the number of credits used to account for splay steps in an expose at a vertex x in a virtual tree with root v

- by Lemma 4.2, the number is at most $3(rank(v)–rank(x))$ plus 1 per splice



By the earlier analysis, the total number of exposes is at most $2m$, and the number of splices is $O(m \log n)$

- so total number of credits used by expose ops is $O(m \log n)$

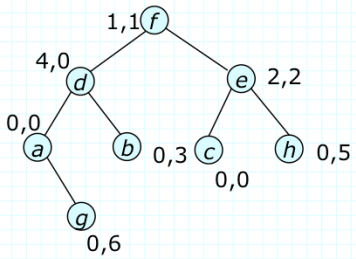
To complete the analysis, we must determine the number of credits used by the tree ops, in addition to those used by the exposes

- the only operation that requires additional credits is the *join* operation, which combines three paths into one
- the root node in the tree representing the new path acquires $O(\log n)$ credits in this operation

Combining the various estimates, the total number of credits used (and thus the running time) is $O(m \log n)$ ■

Exercises

1. Suppose that the tree shown below represents one path in a pathset data structure, where the numbers labeling the nodes are the Δ*cost* and Δ*min* values.



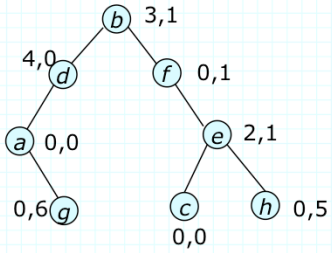
List the vertices in the order in which they appear on the path.

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

What are the costs of vertices *a, b, e* and *h*?

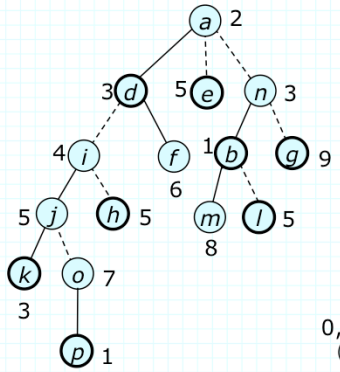
1, 4, 5 and 8

Show how the tree changes following a splay at node *b*. Show the updated Δ*cost* and Δ*min* values.



12

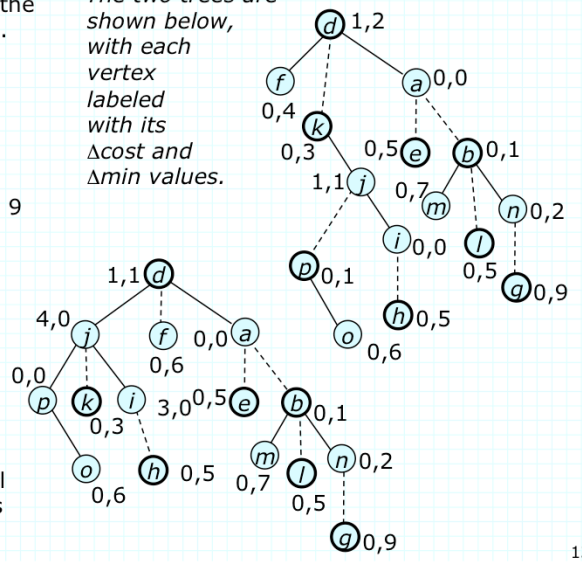
2. The diagram below shows a dynamic tree represented as a collection of paths that are linked together. The nodes with a heavy outline are the canonical nodes for their paths.



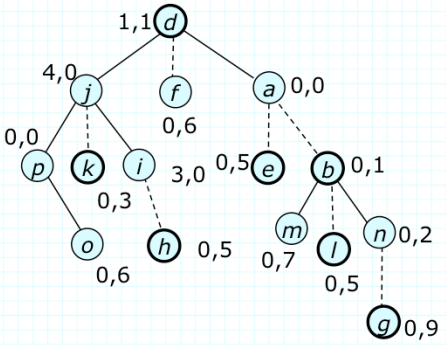
Show the corresponding *virtual tree* that implements this actual tree. For each node, include its Δ*cost* and Δ*min* values.

Then, show how the virtual tree changes after an expose at node *p*.

The two trees are shown below, with each vertex labeled with its Δcost and Δmin values.



3. The diagram below shows a virtual tree that implements an actual tree in the dynamic trees data structure. for each node in the tree with root *d*, list its individual weight and its total weight, as defined in the analysis of the dynamic trees data structure.



Nodes p and o each have an individual weight of 1. Nodes i,j and d all have an individual weight of 2, while node a has an individual weight of 7. Node o has a total weight of 1, node p has 2, node i has 2, node j has 6, node a has 7 and node d has 15.

What is the rank of every node in the entire virtual tree?

The leaves all have a rank of 0. Nodes p,i and n all have a rank of 1. Nodes j, a and b all have a rank of 2 and node d has a rank of 3.

How many credits are needed to satisfy the credit invariant?

$3+3\times2+3=12$

14