day8 2/4/98 "I'd do it, but it probably doesn't work."

1 Treaps, continued

1.1 timing analses for search

- $p\{i = ancestor(j)\} = \frac{1}{j-i+1}$ with $i \leq j$ = number of right turns in descending the tree = In a random ordering among $\{i, i+1, ..., j\}$, the probability that i is first.
- $p\{i = ancestor(j)\} = \frac{1}{j-i+1}$ with $j \leq i$ = number of left turns in descending the tree = In a random ordering among $\{j, j+1, ..., i\}$, the probability that i is first.

Then the expected number of ancestors of j is $\sum_{i=1}^{n} \frac{1}{|j-i|+1} \leq 2\sum_{i=1}^{n} \frac{1}{i} = 2 * ln(n)$

1.2 number of rotations in a delete.

The expected number of rotations in a delete is less than 2. The number of rotations is less than the sum of the rightmost branch (right spine) of the left child and the leftmost branch (left spine) of the right child.

analyse the right spine.

$p_{i \leq j}\{j = ancestor(i) \wedge i \in rightspine\}$ = random order of $\{i, ..., j\}$ with j first and i second = $\frac{1}{(j-i+1)(j-i)}$

let size of right spine = G_j. Then $E(G_j) = \sum_{i=2}^{j-1} \frac{1}{i(i-1)} = \frac{j-1}{j}$

1.3 dynamic treaps

When a particular element is accessed adjust it's priority so the elements float towards the top of the treap.

old $p(k)$ uniform in $[0, 1]$.

new $p(k) = min(p(k), random \in [0, 1])$

2 Binomial and Fibonacci heaps

2.1 An algorithm using these structures

- $G = (V, E)$ an undirected graph
- $|V| = n$
- $|E| = m$
- $l : E \rightarrow R^+$

- extend l as $l(v,w) = \infty$ if $(v,w) \notin E$, 0 if $v = w$, and $l(e)$ where $e = (v,w)$ otherwise
- P = path = $e_1, ..., e_k$.
- $l(P) = \sum_{i=1}^{k} l(e_i)$
- $d(u,v)$ = length of minimum weight path

2.1.1 Dijkstra's algorithm

1. $W = V - \{s\}$, $D(s) = 0$
2. $\forall u \in W$ do $D(u) = l(s,u)$
3. while $W \neq 0$
4. $u = Extractmin(W, D)$
5. $\forall (u,v) \in E$ with $v \in W$, $D(v) = min(D(v), D(u) + l(u,v))$
6. endwhile

2.1.2 timing analyses

with a good data structure you can get:

	#	$cost/op$	$totalcost$
$step2(insert)$	n	$O(1)$	$O(n)$
$step4(deletemin)$	n	$O(log(n))$	$O(n * log(n))$
$step5(decrement)$	m	$O(1)$	$O(m)$

2.2 binomial heap operations

- makeheap(i) O(1)
- findmin(h) O(1)
- insert(h,i) O(1)
- deletemin(h) O(log(n))
- meld(h,h') O(1) or O(log(n))
- decrement(h,i,Δ) (only efficient with fibonacci heaps) O(1)
- delete(h,i) (only efficient with fibonacci heaps) O(log(n))

2.3 definition of binomial trees

- $B_0 = 1$ element tree
- $B_1 = 2$ element tree with one a root of the other
- $B_2 = 4$ element tree composed of 2 B_1 trees joined at the root.
- In general $|B_i| = 2^i$ and the degree of the root node is i

Since the number of elements in each B_i doubles with i, you can have a heap of arbitrariy (integer) size represented as a set of B_i trees of different sizes.

- a set of trees is in heap order if each tree is ordered and points to the min.
- rank(x)=# of children of x

2.4 operation implementations

2.4.1 meld(h,h')

Essentially, you just do binary addition between the sets of trees in h and h'. The amortized cost of adding an element to the heap is 2.