Junzle Zheng

# CS430 HW5

A20254388

## Problem I.

(a): Prove by ~~introduction~~.

Base Case: a tree with zero nodes. this is unique.

~~Brt~~

General Case: Assume the a treap with $k-1$ nodes or less are unique. In this case a node X item with minimum priority must be at root. The root and left-subtree and right-subtree of the root are all unique. Because ~~they are~~ their size $\leq k-1$.

Assume a treap are inserted ~~and~~ and maintained by ~~thei~~ their priority ~~Now cos~~ now for a k nodes treap with ~~a~~ smallest priority. A treap is ~~o~~ has a structure of BST. ~~T~~ thus a k nodes treap ~~is~~ ~~eeo~~ equals to a $k-1$ nodes treap after inserting ~~k(i)~~ kth item. There is only one possible position for kth node. Thus K-nodes treap ~~is~~ is unique.

(b) A treap on n nodes is equivalent to a randomly ~~built~~ built binary ~~search~~ search tree on n nodes.

When we assign priorities randomly, which can be seen as a random permutation of n inputs

h is height of the tree.

$$E(h) = O(\lg n)$$

(c) Do ~~the~~ a usual binary ~~search~~ search tree insert and then perform rotations ~~t~~ to maintain min - heap order property

```
Treep - Insert ( T, X )   <--- insert X to T.
    Tree - Insert (T, X)
    while  x ≠ root(T)  and  priority[x] < priority[P[x]]   <--- parent
        do if  x = left[P[x]]
            then  Right - Rotate (T, P[x])
            else  Left - Rotate (T, P[x])
```

(d) The ~~t~~ time to insert an item into a treap is proportional ~~to~~ the height of a random generated ~~bin~~ BST, which ~~big~~ height is $O(\lg n)$

If a new item with a minimum priority than any other nodes, then we have to do maximum number of rotation

In each rotation, we ~~so~~ move node ~~from leaf~~ to its parent which means move to a ~~heiger~~ higher height.

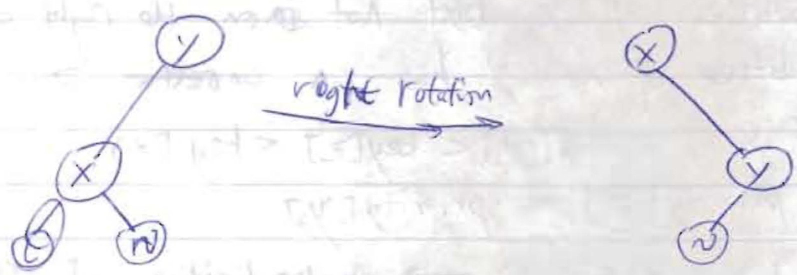
~~To~~ The height is $O(\lg n)$

Thus running time is $O(\lg n + \lg n) = O(\lg n)$

(e) Prove by induction

Base Case ~~x is~~ ~~x = p(y)~~ $x = y.p$ ~~after root~~
this rotation make y is new root and y has only one child x. so $C+D = 1$

Induction. k rotations performed during the insertion of x, is equals to C+D. If x is a leaf node, then 0 rotation is needed. $C+D = 0$
Assuming after k−1 rotations $C+D = k-1$, then after k rotations $C+D = k$ left.




This means the left spine of right subtree of x before rotation is now on y. the lenght of spine of left subtree of y is increased by 1. The left subtree of x is unaffected by right rotation. Left rotation is just a symmetric, therefore
$k = C+D$

(f) Prove "If".
Assume: $X_{ik} = 0$, which means y is not on the right spine of left subtree of x. if in this situation. y colld not be in:

3

① $y$ is ~~sub~~ in right subtree of $x$, ~~co~~ conflicts between
$key[y] < key[x]$

② $y$ is not in one of ~~sb~~ subtree of $x$, then $x, y$ ~~$~~ have a same ancestor $z$. $key[y] < key[x]$

$y$ is in left subtree of $z$ and $x$ is in right subtree.

$key[y] < key[z] < key[x]$

$priority[z] < priority[y]$

$priority[z] < priority[x]$

From condition (3) $X_{ik} = 1$ $\longrightarrow$ contradiction of assumption

③ $y$ is the left subtree of $x$ but not ~~in~~ on the right spine of the left subtree of $x$. $y$ has an ancestor $z$ in left ~~o~~ subtree of $x$: $key[y] < key[z] < key[x]$.

$priority[z] < priority[y]$

From condition (3) $X_{ik} = 1$ $\longrightarrow$ contradiction of assumption.

Prof of "Only if"

$X_{ik} = 1$, $y$ is in right spine of left subtree of $x$.

~~Prior~~ $y$ is inserted after $x$. thus $priority[y] > priority[x]$, proving ①,

$y$ is in the left subtree of $x$. $key[y] < key[x]$ proving of (2)

For (3) Suppose $X_{ik} = 1$ and there is a $z$ that

$key[y] < key[z] < key[x]$ and $priority[z] < priority[y]$

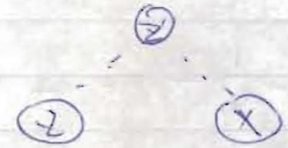
For ~~pro~~ $key[z] < key[~~x~~]$. there $\exists$ ~~pos~~ posistion.

1. ~~z~~ z is in the right spine of the left subtree of X. For y to be inserted into the right spine of the left subtree of x, it will have to be inserted into right subtree of z but key[y] < key[z]. ~~contra~~ contradiction.

2. Suppose z is in the left ~~n~~ subtree of x but not in the right spine, this ~~means~~ means z is in the left subtree of some nodes z' in the ~~right~~ right spine of x, for y, it must be in right subtree of z'

$$\text{key}[y] < \text{key}[z] \longrightarrow \text{contradiction}$$

3. z is and x have a ~~common~~ common anscestor z'



$$\text{key}[z] < \text{key}[z'] < \text{key}[x].$$

But we have $\text{key}[y] < \text{key}[z] < \text{key}[z']$, y cannot be inserted into right subtree of z' and can not be in a subtree of x. which contradiction.

(9) The keys of the items in question are $i, i+1, i+2 \dots k-1, k$.

There are $(k-i+1)!$ permutations of the priorites of these items. Of these permutations the ones ~~o~~ for $X_{ik}=1$ are those where $i$ has minimum priority, $k$ has second smallest priority, and the priority of the remaining $k-i-1$ items follow in any order. There are $(k-i-1)!$

Thus $$Pr\{X_{i,k}=1\} = \frac{(k-i-1)!}{(k-i+1)!} = \frac{1}{(k-i+1)(k-i)}$$

(5)

(h) $E[C]$ is the expected number of nodes in the right spine of left subtree of $X$, so $X_{ik} = 0$, for all $i \geq k$. We consider $i < k$

$$E[C] = \sum_{i=1}^{k-1} E[X_{i,k}] = E\left[\sum_{i=1}^{k-1} X_{i,k}\right]$$

$$= \sum_{i=1}^{k-1} \Pr\{X_{i,k} = 1\}$$

$$= \sum_{i=1}^{k-1} \frac{1}{(k-i)(k-i+1)}$$

$$= \sum_{i=1}^{k-1} \frac{1}{j(j+1)}$$

For $\frac{1}{j(j+1)} = \frac{1}{j} - \frac{1}{j+1}$

$$E[C] = \frac{1}{1} - \frac{1}{2} + \frac{1}{2} - \frac{1}{3} + \cdots + \frac{1}{k-1} - \frac{1}{k+1}$$

$$= 1 - \frac{1}{k}$$

(i) Let $T$ be a binary tree ~~BST~~ ~~using or~~ we get when inserting the items using original key. Once we remap the keys and insert them into a new binary search tree, we get $T'$ is ~~minor~~ mirror of $T$. Item $x$ with key $k$ in $T$ now is with key $n-k+1$ in $T'$, ~~the~~

The length of left spine of $x$'s right subtree in $T$ has become the ~~leg~~ length of the right spine of ~~a left~~ $x$'s left subtree in $T'$, the expected length of right spine of the left subtree of $x$ in $T'$ is: ~~1 - 1/~~

$$1 - \frac{1}{n-k+1}$$

6 Thus $E[D] = 1 - \frac{1}{n-k+1}$

(j)

$$E[\cancel{\text{nuth}}\ \text{num of Rotations}] = E[C+D]$$

$$= E[C] + E[D]$$

$$\cancel{= 1 + 1}$$

$$= 1 - \frac{1}{k} + 1 - \frac{1}{n-k+1} \leq 2.$$

Jinzhe Zhang

~~CS 430~~ CS430 HW5 A20254389.

Problem 2.

Suppose the hash function $[h_1(k) + h_2(k)] \bmod m$, $i = 0, 1, \ldots$ repeats a location at jth step, $j < m$. Will the hasing function generate m different locations

Suppose that:

$$h_1(k) + j h_2(k) = h_1(k) + i h_2(k) \pmod m$$

$$(j - i) h_2(k) = 0 \pmod m$$

$$(j - i) h_2(k) = c \cdot m \quad \text{where } c \text{ is more integer}$$

If $h_2(k)$ and $m$ are not relatively prime, then above equation is holds. Thus the hash function repeats at the i-th and j-th step, and hence the hasing function will not generate m different locations

Junzle Zheng

CS430 HW5 A20254389

(3)

Problem 3

a) The probe sequence is $\langle h(k), h(k)+1, h(k)+1+2, \ldots, h(k)+1+2+\cdots i \ldots \rangle$. Starting the probe number from 0, the ith probe is offset (modulo m) from h(k) by

$$\sum_{j=0}^{i} j = \frac{i(i+1)}{2} = \frac{1}{2}i^2 + \frac{1}{2}i$$

Thus $h'(k,i) = (h(k) + \frac{1}{2}i + \frac{1}{2}i^2) \bmod M$.

This is a special case of quadratic.

b) Let $h'(k,i)$ denote the ith probe.

$$h'(k,i) = \left(h(k) + i(i+1)/2\right) \bmod M.$$

We want to show for a given key k ~~each of the first m probe~~ and for any probe number i and j such that $0 \le i < j < M$, We have $h'(k,i) \ne h'(k,j)$

Assume that there is a key k and probe numbers i and j satisfying $0 \le i < j < M$ for which $h'(k,i) = h'(k,j)$. Then

$$h(k) + i(i+1)/2 \equiv h(k) + j(j+1)/2 \pmod M$$

$$\Downarrow$$

$$j(j+1)/2 - i(i+1)/2 \equiv 0 \mod M$$

$$\Downarrow$$

$$(j-i)(j+i+1)/2 \equiv 0 \mod M$$

we have

$$(j-i)(j+i+1)/2 = r \cdot m \quad \text{for } \cancel{r \in \mathbb{N}} \text{ some integer } r.$$

Because $m = 2^p$

$$(j-i)(j+i+1) = r \cdot 2^{p+1}$$

But $j-i < m < 2^{p+1}$

and $j+i+1 \le (m-1)+(m-2)+1 = 2m-2 < 2^{p+1}$

Thus $2^{p+1}$ can not divides $j-i$ or $j+i+1$.

We conclude that $h'(k,i) \neq h'(k,j)$