HW # 5 Soln

CS 724

(2) For $i = 1, \ldots, n$ let c_i be the actual cost of insertion.

Let D_i represents the table that results after i^{th} operation performed on D_{i-1}.

The ~~cost~~ amortized cost $\hat{c}_i$ of the i^{th} operation is

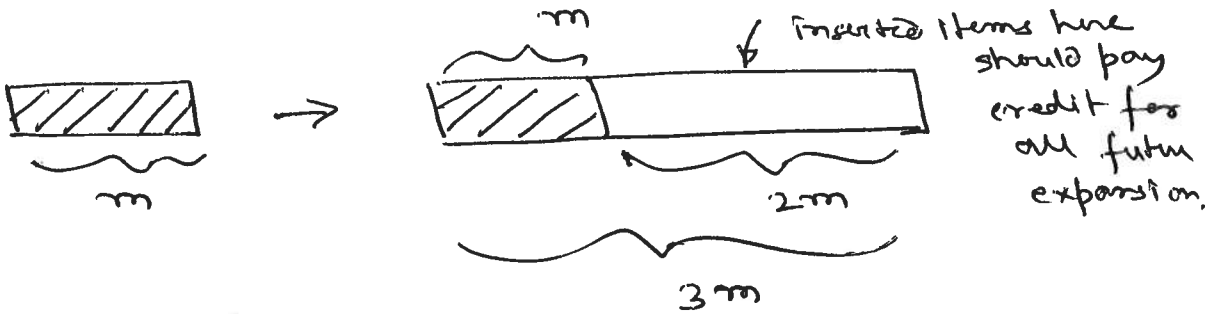
$$\hat{c}_i = c_i + \phi(D_i) - \phi(D_{i-1})$$

where $\phi(D_i)$ is the potential function after operation i.

~~We will use the notation num(T) to indicate number of elements in table T and size(T) to indicate size of the table~~

(a) Accounting method

Consider the situation when a table of size m is full. It is expanded and all m items are copied to the new table. Now these m items does not have any credit to be copied further. And new inserted items must pay for them.



Suppose we charge c unit for each insertion.

total credit = $2mc$. Out of these $2m$ inserts were paid (1 unit for each $2m$ insert).

Thus remaining credit = $2mc - 2m$. This credit must be enough for future expension when all $3m$ items need to be copied to the larger table of size $3 \times 3m = 9m$.

Thus. $2mc - 2m = 3m$

$\Rightarrow c = 5/2$

Therefore we must charge $5/2$ unit for each new item insert.

∴ For total m insert, total amortized cost is $\frac{5}{2}m$. which is $O(m)$

(b) Potential method

At any point the table is atleast ~~one~~ one third full.

Note that any item, after insert, has $5/2 - 1 = 3/2$ credit left. Therefore we define potential function to be $\phi(T) = \frac{3}{2}\left[num(T) - \frac{size(T)}{3}\right]$

Immediately after expension

$num(T) = \frac{size(T)}{3} + 1 \Rightarrow \phi(T) = \frac{3}{2} \geq 0.$

Immediately before expension

$num(T) = size(T) \Rightarrow \phi(T) = \frac{3}{2}\left[size(T) - \frac{size(T)}{3}\right]$

$= size(T) \geq 0$

At any other point $num(T) > \frac{size(T)}{3} \Rightarrow \phi(T) \geq 0.$

Let $\hat{c}_i$ be the amortized cost of the ith operation.

then $\hat{c}_i = c_i + \phi_i - \phi_{i-1}$

If ith table insert does not trigger expansion

$c_i = 1$ (only copy this item)

$\phi_i = \frac{3}{2}\left(num(T) - \frac{1}{3} size(T) \right)$

$\phi_{i-1} = \frac{3}{2}\left[num(T) - 1 - \frac{1}{3} size(T) \right]$

size of the table is same. The number of items was 1 less during last insert.

$\therefore \hat{c}_i = c_i + \phi_i - \phi_{i-1}$

$= 1 + \frac{3}{2}\left[num(T) - \frac{1}{3} size(T) \right]$

$- \frac{3}{2}\left[num(T) - 1 - \frac{size(T)}{3} \right]$

$= 1 + \frac{3}{2} num(T) - \frac{1}{2} size(T) - \frac{3}{2} num(T)$

$+ \frac{3}{2} + \frac{1}{2} size(T)$

$= 5/2.$

If ith table insert triggers table expansion

$c_i = num(T)$ need to copy all num(T) items.

size of the table was $1/3$rd of the current table size during $(i-1)$th insert.

$\therefore \phi_i = \frac{3}{2}\left[num(T) - \frac{1}{3} size(T) \right]$

$\phi_{i-1} = \frac{3}{2}\left[num(T) - 1 - \frac{1}{3} \cdot \frac{size(T)}{3} \right]$

Note that now number of items satisfy.

$$num(T) = \frac{size(T)}{3} + 1. \quad \text{— } (*)$$

$$\therefore \hat{c}_i = c_i + \phi_i - \phi_{i-1}$$

$$= num(T) + \frac{3}{2}\left[num(T) - \frac{1}{3} size(T)\right]$$

$$- \frac{3}{2}\left[num(T) - 1 - \frac{1}{3} \cdot \frac{size(T)}{3}\right]$$

$$= num(T) + \frac{3}{2} num(T) - \frac{1}{2} size(T)$$

$$- \frac{3}{2} num(T) + \frac{3}{2} + \frac{1}{6} size(T)$$

$$= \frac{size(T)}{3} + 1 - \frac{1}{2} size(T) + \frac{3}{2} + \frac{1}{6} size(T)$$

using (*)

$$= \frac{5}{2} + size(T)\left\{\frac{1}{3} - \frac{1}{2} + \frac{1}{6}\right\}$$

$$= \frac{5}{2} + size(T)\left\{\frac{2 - 3 + 1}{6}\right\} = \frac{5}{2}.$$

$\Downarrow$

0

③ Let $c[i,j]$ be the minimum cost obtainable when storing j pills using bottles 1 through i.

Using dynamic programming our solution will be

$$c[n, W]$$

We will first write solution of a problem in terms of solution of subproblems. While writing the expression of $c[i,j]$ we will consider whether the bottle i is used or not.

Case #1 : Bottle i is used. Thus optimal cost is c_i plus optimal cost of storing $j - p_i$ pill is bottles 1 through $(i-1)$.

Case #2 : Bottle i is not used. In this case optimal cost is to stor j pills in bottles 1 through $(i-1)$.

Thus we can write

$$c[i,j] = \min \{ c_i + c[i-1, j-p_i] , c[i-1, j] \} \quad (**)$$

So far so good.

Only problem is if $j - p_i$ is negative.

Thus we need to consider the case when $p_i > j$.

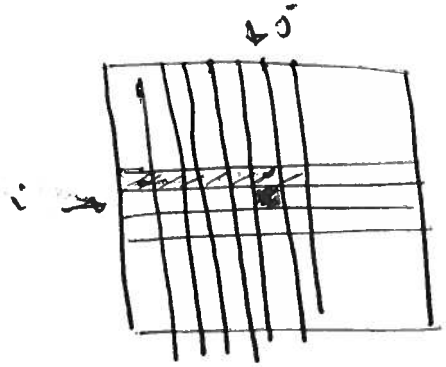
In this case the previous recurrence relation becomes

$$c[i,j] = \min \{ c_i, c[i-1,j] \}$$

Because if bottle i is used it is not filled!

Thus, we have

$$c[i,j] = \begin{cases} \min\{c_i, c[i-1,j]\} & \text{if } p_i > j \\ \min\{c_i + c[i-1, j-p_i], c[i-1,j]\} & \text{if } p \le j \end{cases}$$



c matrix

c[i,j] depends on c[i-1,j] and c[i-1,t] where t<j

Thus as long as we fill the table rowwise all sub problem solutions should be available.

```
for i ← 1 to n
    c[i,0] = 0
for j ← 1 to W
    c[0,j] = 0
for i ← 1 to n
    for j ← 1 to W
        if p_i ≤ j
            with = c_i + c[i-1, j-p_i]
        else with = c_i
        without = c[i-1, j]
        c[i,j] = min{with, without}
```

Two nested for loops. Running time $O(nW)$.

(i) The worst case is when the binary tree is a long chai

(a) and adding new elements make this chain even longer.

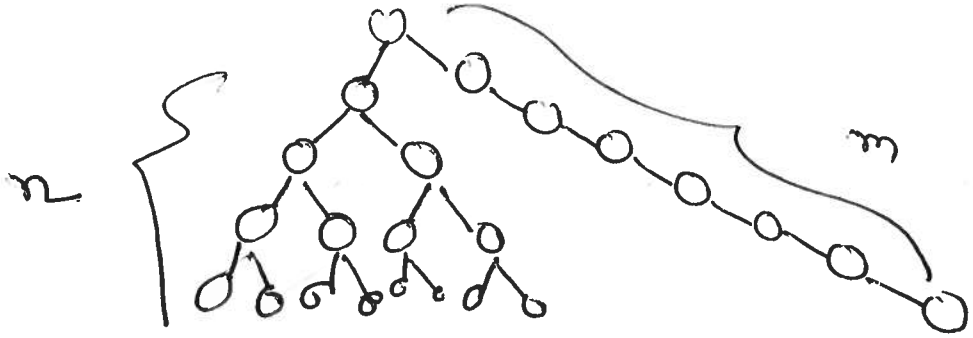
To insert the i^{th} element from A it needs to visit the complete depth of the tree before being inserted. That is cost is $(m+i)$.

∴ Total cost

$$\sum_{i=1}^{n}(m+i) = \sum_{i=1}^{n} m + \sum_{i=1}^{n} i = mn + n\frac{(n+1)}{2} = O(mn+n^2)$$

(b) ~~Best case is when the elements are inserted that it first~~

Best case is when the tree is chai and all nods form a balance binary tree starting at the other child node.



$$\frac{n}{2}\log n + \frac{n}{4}(\log n - 1) + \frac{n}{8}(\log n - 2) + \cdots$$

$$= O(n\log n - n).$$

④ (a) colored red : NO. Because it will be a red child of a red node which will violate R-B tree property.

colored black : NO. Because it will violate the property that all paths from root node will have same number of internal black node.

(b) On longest path every other node will be black, i.e., red and black nodes will alternate. On shortest path there will be no red nodes all black nodes.

(c) 2^{2R-1} ← (largest) This will happen when in every path red and black nodes alternate.

Smallest : $2^R - 1$. This will happen when the tree has only black nods. No red nodes.

(d) The largest ratio is 2, when each black node has two red children.
the smallest ratio is 0, when there are no red nodes.