

ECE 537 - Foundations of Computing  
 Prof. Sen  
**Homework #9**  
**Solutions**

1. Work problem 17.2-2 in the Cormen et al. text.

**Solution:** Let  $c_i$  be the cost of the  $i$ th operation. Where

$$c_i = \begin{cases} i, & \text{if } i \text{ is an exact power of 2,} \\ 1, & \text{otherwise.} \end{cases}$$

If we charge \$3 per operation (i.e.,  $\hat{c} = 3$ ), then if  $i$  is not a power of 2, we'll be paying \$1 for the operation, and saving \$2 on the balance sheet, and when  $i$  is a power of 2, we'll need to pay \$i using some of the credits stored on the balance sheet. Furthermore, if there are  $m$  total operations, then  $\sum_{i=1}^m \hat{c} = 3m$ .

We can see from the table below that the balance sheet can never go negative.

operation	am. cost	actual cost	balance
1	3	1	2
2	3	2	3
3	3	1	5
4	3	4	4
5	3	1	6
6	3	1	8
7	3	1	10
8	3	8	5
9	3	1	7
⋮	⋮		

Therefore,  $\sum_{i=1}^m \hat{c} \geq \sum_{i=1}^m c_i$ , and since  $\sum_{i=1}^m \hat{c} = 3m$ , the amortized cost per operation is  $\Theta(1)$ .

2. Work problem 17.3-3 in the Cormen et al. text.

**Solution:** Let us denote the initial configuration of the binary heap data structure using  $D_0$ , and the  $i$ -th configuration using  $D_i$ . Furthermore, use  $n_i$  to denote the number of data elements stored in  $D_i$ . We have seen that the *Insert* and *DeleteMin* operations can be implemented in  $O(\log n)$  time using a binary heap.<sup>1</sup> Next, let  $d_i(x)$  denote the depth of vertex  $x$  in  $D_i$ . Consider the following potential function:

$$\begin{aligned} \Phi(D_i) &= \sum_{x \in D_i} (d_i(x) + 1) \\ &= n_i + \sum_{x \in D_i} d_i(x). \end{aligned}$$

---

<sup>1</sup>In order to be consistent with the lecture notes, I'm using the name *DeleteMin* rather than *Extract-Min*.

Note that  $\Phi(D_0) = 0$ , and  $\Phi(D_i) > 0$  for all  $i > 0$ . Thus, this is a suitably defined potential function. Now, if the  $i$ -th operation is an *Insert*, the sum in the previous potential function will only change by an amount that is equal to the depth of the new vertex, which has been inserted as the rightmost vertex on the last level. Thus, the sum will increase by  $\lfloor \lg n_i \rfloor$ , and the change in potential due to this operation is:

$$\begin{aligned}\Delta\Phi(D_i) &= 1 + \lfloor \lg n_i \rfloor \\ &= O(\log n).\end{aligned}$$

Therefore, the amortized cost of the *Insert* operation is:

$$\begin{aligned}\hat{c}_i &= O(\log n) + O(\log n) \\ &= O(\log n).\end{aligned}$$

For the *DeleteMin* operation we have:

$$\begin{aligned}\Delta\Phi(D_i) &= -(1 + \lfloor \lg n_{i-1} \rfloor) \\ &= -O(\log n),\end{aligned}$$

and the amortized cost of the *DeleteMin* operation is:

$$\begin{aligned}\hat{c}_i &= O(\log n) - O(\log n) \\ &= O(\log n).\end{aligned}$$

3. We can generalize the analysis of the MTF algorithm provided in a class lecture by considering what happens when the *MTF* algorithm moves an accessed element some fixed fraction of the way towards the head of the list, instead of all the way to the head of the list. Specifically, prove that if  $MTF(d)$  ( $d \geq 1$ ) is an algorithm which takes an element at position  $\mu$  and moves it  $p = \lceil \mu/d \rceil - 1$  elements closer to the front of the list after a *Search* or *Insert* operation, then

$$\sum_{i=1}^m c_i^{MTF(d)} \leq d \left( 2 \sum_{i=1}^m c_i^A + X^A - F^A - m \right)$$

where  $A$  is any self-organizing list algorithm that obeys the other assumptions given in class. Note that when  $d = 1$ , this reduces to the analysis provided in the lecture.

**Solution:** This proof is very similar to the one given in the lecture. Once again we simultaneously maintain two initially empty lists—one using  $MTF(d)$ , and the other using  $A$ . In this case, however, the potential function will be  $d$  times the number of inversions in  $MTF(d)$ 's list with respect to  $A$ 's list. This potential function works because it is initially zero, and can never be nonnegative. We will show that the amortized cost of any operation in  $MTF(d)$ 's list is at most  $d(2c^A - 1)$ . As before, we assume that any DICTIONARY ADT operation that takes place at position  $\mu$  in  $MTF(d)$ 's list takes place at position  $\alpha$  in  $A$ 's list. Consider first the *Search* operation. Let  $I_\alpha$  be the element stored at position  $\alpha$  in  $A$ 's list. In the  $MTF(d)$  list,  $I_\alpha$  is moved

forward past  $p$  list items after the search. Of these  $p$  list items, let  $x_\alpha$  be the number that follow  $I_\alpha$  in  $A$ 's list. For example if we have

$$\begin{aligned} A &: a, c, e, f, b, d, g \\ MTF(d) &: d, a, e, c, b, f, g \end{aligned}$$

with  $\alpha = 4$ , and  $d = 2$ ; then  $I_\alpha = f$ ,  $k = 6$ , and  $p = 2$ . After the *Search* operation the  $MTF(d)$  list will be rearranged as follows:

$$MTF(d) : d, a, e, f, c, \boxed{b}, g$$

The item enclosed in a box is the only one that  $f$  was moved past in  $MTF(d)$ 's list, and also appears after  $f$  in  $A$ 's list. The number of inversions created by moving  $I_\alpha$  past  $p$  items in  $MTF(d)$ 's list is  $p - x_\alpha$ , but  $x_\alpha$  other inversions are removed. Therefore the amortized cost of this operation to  $MTF(d)$ 's list is  $\mu + d(p - x_\alpha - x_\alpha) = \mu + d(p - 2x_\alpha)$ . Since the actual cost of a *Search* to position  $\alpha$  in  $A$ 's list is  $\alpha$ , we need to show that  $\mu + d(p - 2x_\alpha) \leq d(2\alpha - 1)$ . According to the definition of the ceiling function, we know that  $p \geq \mu/d - 1$ . Furthermore, since each of the items which  $I_\alpha$  is moved past in  $MTF(d)$ 's list is either one of the  $\alpha - 1$  items preceding  $I_\alpha$  in  $A$ 's list or one of the  $x_\alpha$  items following  $I_\alpha$  in  $A$ 's list but preceding it in  $MTF(d)$ 's list, we know that  $p \leq \alpha - 1 + x_\alpha$ . Thus we can write that  $\mu/d - 1 \leq \alpha - 1 + x_\alpha$ ; which when solved for  $\mu$  yields  $\mu \leq d(x_\alpha + \alpha)$ . Multiplying the inequality  $p \leq \alpha - 1 + x_\alpha$  by  $d$  and adding it to the last result gives us  $\mu + dp \leq d(2x_\alpha + 2\alpha - 1)$ , which implies  $\mu + d(p - 2x_\alpha) \leq d(2\alpha - 1) = d(2c^A - 1)$ .

Now consider the *Insert* operation. In this case let  $I_\alpha$  be the item we are inserting. Since  $I_\alpha$  is appended to the end of both lists,  $x_\alpha = 0$ . When  $I_\alpha$  is moved past the  $p$  items in  $MTF$ 's list,  $p - x_\alpha = p$  new inversions are created, and none are removed. Therefore the amortized cost of this operation is  $\mu + dp$ . Since this was previously bounded by  $d(2x_\alpha + 2\alpha - 1)$ , and  $x_\alpha = 0$ , we have that  $\mu + dp \leq d(2\alpha - 1) \leq d(2c^A - 1)$ .

Finally consider the *Delete* operation. Since no exchanges take place, no new inversions are created, and only those  $x_\alpha$  inversions that involve the item being deleted will be removed. Therefore the amortized cost of this operation to  $MTF$ 's list is  $\mu - dx_\alpha$ . Since  $d$  is positive,  $\mu - dx_\alpha \leq \mu - x_\alpha$ , and in the lecture we showed that  $\mu - x_\alpha \leq \alpha$ . Since  $\alpha \geq 1$ , it follows that  $\alpha \leq 2\alpha - 1 \leq d(2\alpha - 1) = d(2c^A - 1)$ . Thus, we have shown that if no exchanges occur in  $A$ 's list,

$$\sum_{i=1}^n c^{MTF(d)} \leq \sum_{i=1}^n d(2c^A - 1) = d\left(2 \sum_{i=1}^n c^A - n\right)$$

If we also allow exchanges to occur in  $A$ 's list, then each free exchange by algorithm  $A$  decreases the number of inversions by 1 (thereby decreasing the potential by  $d$ ), and each paid exchange by algorithm  $A$  increases the number of inversions by 1 (thereby increasing the potential by  $d$ ). This yields the desired bound:

$$\sum_{i=1}^n c^{MTF(d)} \leq d \left( 2 \sum_{i=1}^n c^A + X^A - F^A - n \right)$$