Algorithm HW9

Daoyu Wang

November 12

Contents

1 17.1-2

If we allow the operation **DECREASEMENT**, then consider this situation: the initial value is 0 which represents 0 of length k, then we **DECREASEMENT** it to -1 which represents 1 of length k, then we **INCREASEMENT** it to 0 and cycle over and over again.

In this situation, all k bits will reverse from 0 to 1 or from 1 to 0 in every operation. So the worst case running time is $O(kn)$.

2 17.3-1

Assume that:

$$\Phi'(D_i) = \Phi(D_i) - \Phi(D_0) \tag{1}$$

Obviously, when using $\Phi'(D_i)$ to calculate amortization analysis, the result is as follows:

$$\begin{aligned}\sum_{i=1}^{n} \hat{c}_i &= \sum_{i=1}^{n} c_i + \Phi'(D_i) - \Phi'(D_0) \\ &= \sum_{i=1}^{n} c_i + \Phi(D_i) - \Phi(D_0) - \Phi(D_0) + \Phi(D_0) \\ &= \sum_{i=1}^{n} c_i + \Phi(D_i) - \Phi(D_0)\end{aligned} \tag{2}$$

So the modified amortized cost is the same as the original one.

3 17.3-4

The total cost is as follows:

$$\sum_{i=1}^{n} c_n + D_n - D_0 = O(n) + s_n - s_0 \tag{3}$$

4 19.1-3

Firstly, if we traverse the order in postorder and mark the nodes in the binomial tree B_k as binary, we can find that the tree is still a binomial tree.

4.1 Problem 1

Consider node **x** which is marked as **l** in height **i** and assume $j = k - i$. We can use **Mathematical Induction** to prove that there are **j** 1's in the binary representation of **x** in height **i**. However, we can't use height **i** as the object of induction because the node at height **i+1** is only associates with its right child (its value is its right child plus one) but we can't acknowledge full information of its right child if we just know its right child is in height **i**.

Change our mind: consider the how B_{k+1} is generated. We can find that B_{k+1} is generated by merging two items. First of the items is B_k while the second one is the duplicate of B_k but change the k^{th} bit (count from right to left and the rightest is the 0^{th}) from 0 to 1. Then we connect the roots of two items and the B_k is in lower position.

So we can use the order k as the object of induction. We can prove that there are **j** 1's in the binary representation of **x** and this is suitable for all nodes in B_k.

- **Base Case:** When $k = 0$, there is only a root **x** in the binomial tree B_0, so there is no 1 in the binary representation of **x**. Actually, **j** is also 0 in this situation.

- **Inductive Hypothesis:** Assume that there are **j** 1's in the binary representation of **x** in height **i** and this is suitable for all nodes in B_k.

- **Inductive Step:** As we has discussed above, the duplicate of B_k just change the k^{th} bit from 0 to 1. So the number of 1 in all layers of B_k has been added by 1. In other words, the number of 1 in a given layer in duplicate of B_k is the same as the number of 1 in its **upper** layer in B_k. Actually, when we connect the roots of two items and put B_k in lower position, in the binomial tree B_{k+1}, a given layer in the duplicate of B_k **happens to** be on the same layer as the upper one of its original layer in B_k. As a result, in B_{k+1}, in height **i** whose corresponding **j** is $(k+1) - (i+1) = k - i$, all nodes have **j** 1's in their binary representation, same as the nodes in B_k.

4.2 Problem 2

According to the result of Problem 1, in height **i** whose corresponding **j** is $k - i$, all nodes have **j** 1's in their binary representation. So there are C_k^j binary **k** strings containing exactly **j** 1's.

4.3 Problem 3

We can also use **Mathematical Induction** to prove that among the degree of node **x** is the same as The number of 1's to the right of the rightmost 0 in the binary representation of **l**.

Use the order k as the object of induction.

- **Base Case:** When $k = 0$, there is only a root **x** in the binomial tree B_0, so the degree of **x** is 0. Actually, the number of 1's to the right of the rightmost 0 in the binary representation of **l** (each bit is 0) is also 0 in this situation.

- **Inductive Hypothesis:** Assume that the degree of node **x** is the same as The number of 1's to the right of the rightmost 0 in the binary representation of **l** and this is suitable for all nodes in B_k.

- **Induction Step:** As we has discussed above, the duplicate of B_k just change the k^{th} bit from 0 to 1. So the number of 1's to the right of the rightmost 0 in the binary representation only changes in the root the duplicate of B_k because while the number of 1's to the right of the rightmost 0 in other nodes doesn't change, only the root's binary representation is several 0 with k 1's and the change of the k^{th} bit makes the number of 1's to the right of the rightmost 0 becomes $k+1$.

  When we connect the roots of two items and put B_k in lower position, in the binomial tree B_{k+1}, only the root of the duplicate of B_k (now the new root of B_{k+1}) **happens to** change its degree from k to $k+1$ which exactly coincides with the quantitative changes in 1 above. As a result, in B_{k+1}, the degree of node **x** is the same as The number of 1's to the right of the rightmost 0 in the binary representation of **l**, same as the nodes in B_k.

5 19.2-2

The initial graph is:

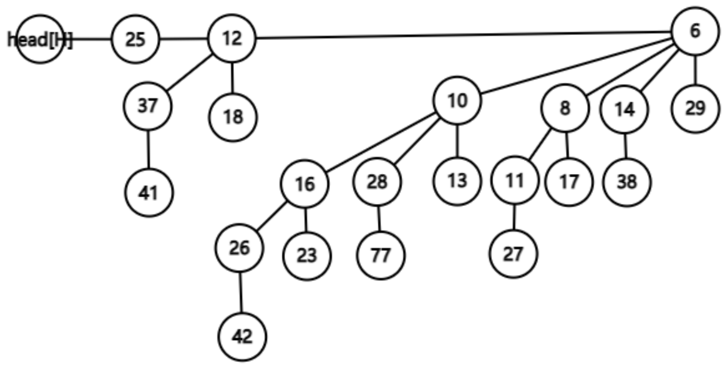

Figure 1: Initial Graph

Merge the two binomial heaps:

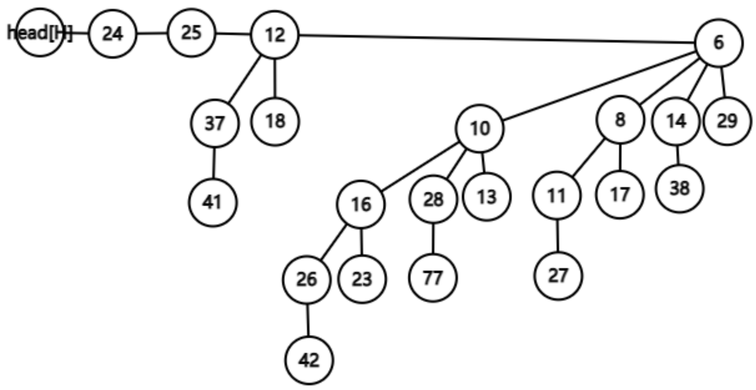

Figure 2: Merge

Union Heap:

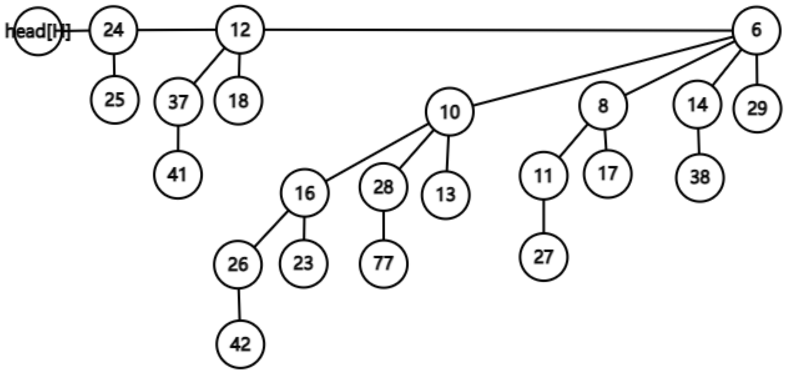

Figure 3: Union

6 19.2-6

We can find the node containing minimum key in $O(\lg n)$ time. Then we can use the **DECREASE-KEY** operation to decrease the key to the minimum key and use **EXTRACT-MIN** to delete it. The total running time is $O(\lg n)$.

Algorithm 1 BINOMIAL-HEAP-DELETE(H, x)

$x \leftarrow head[H]$
$min \leftarrow key[x]$
$x \leftarrow sibling[x]$
while $x \neq NIL$ **do**
    if $key[x] < min$ **then**
        $min \leftarrow key[x]$
    end if
    $x \leftarrow sibling[x]$
end while
$BINOMIAL - HEAP - DECREASE - KEY(H, x, min)$
$BINOMIAL - HEAP - EXTRACT - MIN(H)$