CS 522: Data Structures and Algorithms II
Homework 1

Dustin Ingram

January 18, 2013

1. **Solution:** If both Increment and Decrement operations were included in the k-bit counter, an amortized analysis of the cost of n operations would cost as much as $\Theta(nk)$ time because we would no longer be able to consider each operation as a consecutive Increment, but rather as any combination of Increments and Decrements. Thus, in a worse-case scenario, it would be possible to alternate n times between two operations which cost $O(k)$ each, resulting in a total cost of $\Theta(nk)$.

2. **Solution:** To show that the amortized cost of Table-Delete under this strategy is bounded above by a constant, we will consider two cases. We will use the potential function:

$$\Phi(T) = |2 \cdot T.num - T.size|$$

The first case is the one in which the table does not contract, and thus $num_i = num_{i-1} - 1$, $size_i = size_{i-1}$, and $c_i = 1$:

$$\begin{aligned}
\hat{c_i} &= c_i + \Phi_i - \Phi_{i-1} \\
\hat{c_i} &= 1 + |2 \cdot num_i - size_i| - |2 \cdot num_{i-1} - size_{i-1}| \\
\hat{c_i} &= 1 + |2 \cdot (num_{i-1} - 1) - size_{i-1}| - |2 \cdot num_{i-1} - size_{i-1}| \\
\hat{c_i} &= 1 + |-2| \\
\hat{c_i} &= 3
\end{aligned}$$

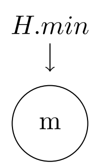
The second case is the one in which the table does contract, and thus $size_i = \frac{2}{3} size_{i-1}$, $num_{i-1} = \frac{1}{3} size_{i-1}$, and $c_i = num_i + 1$:

$$
\begin{aligned}
\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
\hat{c}_i &= (num_i + 1) + |2 \cdot num_i - size_i| - |2 \cdot num_{i-1} - size_{i-1}| \\
\hat{c}_i &= ((num_{i-1} - 1) + 1) + |2 \cdot (num_{i-1} - 1) - \frac{2}{3} size_{i-1}| - |2 \cdot num_{i-1} - size_{i-1}| \\
\hat{c}_i &= (num_{i-1}) + | - 2 + \frac{1}{3} size_{i-1}| \\
\hat{c}_i &= 2
\end{aligned}
$$

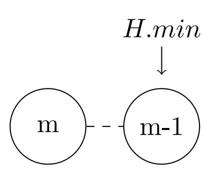
Thus we see that the amortized cost of TABLE-DELETE is at most 3 and is thus bounded.

3. **Solution:** One can use the following sequence to produce a Fibonacci heap that is only a linear chain of n nodes. Since we start at the base case (where the heap is empty) and create a chain of a single node, followed by a chain of two nodes from a chain of one node, and then a chain of three nodes from a chain of two nodes, we can repeat the process n times to produce a chain of total length n.

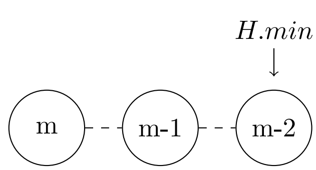
   FIB-HEAP-INSERT(H, m):

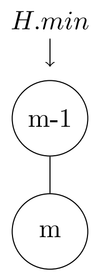
   FIB-HEAP-INSERT(H, $m - 1$):

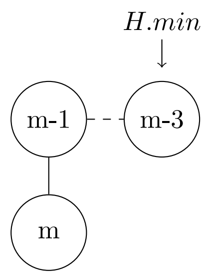
   FIB-HEAP-INSERT(H, $m - 2$):

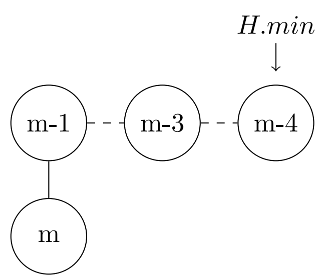
   FIB-HEAP-EXTRACT-MIN(H):



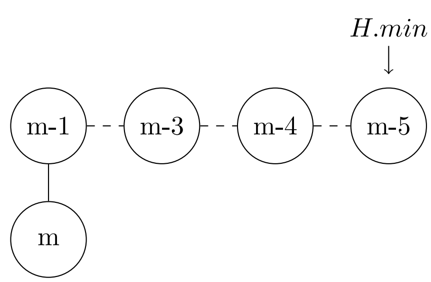
Fib-Heap-Insert(H, $m - 3$):



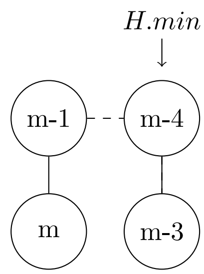
Fib-Heap-Insert(H, $m - 4$):



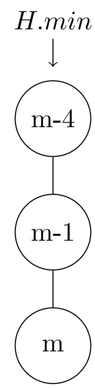
Fib-Heap-Insert(H, $m - 5$):



Fib-Heap-Extract-Min(H):



FIB-HEAP-DELETE(H, $m-3$):



4. **Solution:**

5. **Solution:**

6. **Solution:**

7. **Solution:**

8. **Solution:**