

CS 610, Spring 2009, Prof. Calvin
Homework #2 Solutions

C-1.32 Use an accounting scheme similar to the one described on pp. 40–41 in the text. Charge each addition four cyber-dollars, and don't charge deletions. With each addition, “store” the extra 3 cyber-dollars with the inserted items. If the table needs to be extended when it reaches size 2^i , this will require 2^i cyber-dollars to copy the elements over to a new array. Take \$2 each from the elements at locations 2^{i-1} through $2^i - 1$, and place the remaining \$1 at each of these 2^{i-1} locations into the first quarter of the slots in the new array, to be used to cover the cost of copying the first quarter of the array into a smaller array, should that be necessary in the future.

C-2.23 Insert the n intervals into a min priority queue keyed on the left endpoint a_i ; let A denote the array implementing the heap. Modify the method names from class from max to min.

```
computeUnion(A)
while heap is not empty
     $[a, b] = \text{HeapExtractMin}(A)$ 
     $min = a$ 
     $max = b$ 
    while  $\text{HeapMinimum}(A) \leq max$ 
         $[a', b'] = \text{HeapExtractMinimum}(A)$ 
        if  $b' > max$  then  $max = b'$ 
    output union component  $[min, max]$ 
```

C-2.28 For a min heap, consider the sequence $(n, n - 1, n - 2, \dots, 1)$. The i th insertion (of $n - i + 1$) will require time proportional to the height of the heap, or $\Omega(\lg(n - i + 1))$. Therefore, the total time is bounded below as

$$\sum_{i=1}^n \Omega(\lg(n - i + 1)) = \sum_{i=1}^n \Omega(\lg(i)) = \Omega\left(\sum_{i=1}^n \lg(i)\right) = \Omega(\lg(n!)) = \Omega(n \lg(n)).$$