Analysis of Algorithms

V. Adamchik CSCI 570 Fall 2016

Discussion 4 University of Southern California

Amortized Cost

Heaps

Binary counter

Given n binary numbers of log n bits each

00…000
00…001
00…010
00…011
00…100
…
111…111

Let the cost of incrementing a binary number is the number of bits flipped. What is the amortized cost per increment?

Binary counter

Clearly, the worst-case cost per increment is O(log n) - all bits are flipped.

000	
001	1
010	2
011	1
100	3
101	1
110	2
111	1

Average: 11/7 < 2

Binary counter

The aggregate method.

Lets us start with the least significant bit. Each time we increment a number, that bit is changed. Thus, the number of times this bit changes is n. Look at the next significant bit. it changes n/2 times, the next one - n/4, and so on.

Binary counter

Thus the total cost is

$n + n/2 + n/4 + \ldots + 1 =$

$= n (1 + 1/2 + 1/4 + \ldots + 1/n) =$

$= n (2 - 1/n)$

Amortized cost per increment is O(2).

Binary counter

The accounting method.

Every time you flip $0 \rightarrow 1$, pay the actual cost of 1, plus put 1 into a bank.

Now, every time you flip a $1 \rightarrow 0$, use the money in the bank to pay for the flip.

Clearly, by design, our bank account cannot go negative. Why-?

The key point now is that even though different increments can have different numbers of $1 \rightarrow 0$ flips, each increment has exactly one $0 \rightarrow 1$ flip.

Binary counter 2

Let us assume that the cost of a flip is 2^k to flip kth bit. What is the amortized cost per increment?

Now, in a sequence of n increments, a single increment could cost as much as n in the worst-case.

The aggregate method.

Binary counter 2

Lets us start with the least significant bit. The number of times this bit changes is n, with the cost 2^0.

The number of times the next bit changes is n/2, with the cost 2^1.

Thus the total cost is

$n\,2^0 + n/2\,2^1 + n/4\,2^2 + \ldots + 1\,n = n \log n$

Amortized cost per increment is $O(\log n)$.

Amortized Dictionary

The idea of this data structure is similar to a binomial heap. We will have a collection of sorted arrays, where array k has size 2^k. Each array is either empty or full. However, there will be no relationship between the items in different arrays. The issue of which arrays are full and which are empty is based on the binary representation of the number of items we are storing. For example, if we had 11 items (11 = 1 + 2 + 8), then the arrays of size 1, 2, and 8 would be full and the rest empty.

Amortized Dictionary

What is the worst-case complexity of searching in this data structure?

Do a binary search on each array!

$\log(n/2) + \log(n/4) + \ldots + \log(2) =$

$= O(\log^2(n))$

What is the worst-case complexity of searching in a binomial heap?

$O(n)$

Amortized Dictionary

What is the amortized cost of insert?

Insert requires merging arrays of the same size.

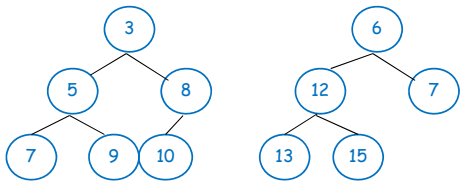
This is exactly the same as the binary counter with cost 2^k for counter k.

Amortized cost per insert is $O(\log n)$.

Amortized cost per insert in a binomial heap is $O(2)$.

Problem

Suppose you have two binary min-heaps, A and B, with a total of *n* elements between them. You want to discover if A and B have a key in common. Give a solution to this problem that takes time $O(n \log n)$ and explain why it is correct. Do not use the fact that heaps are implemented as arrays; treat them as abstract data types.



Solution

Compare the root of each min-heap. If they match, stop.

If they don't match, remove the smaller-valued root:

- we remove the smaller-valued root because we know that value isn't in the other min-heap.

Repeat until a match is found or one of the min heaps is empty.

delMin() runs in O(log n), and in the worst case, we have to run it on all n nodes, giving O(n log n)

Second Solution O(n log n)

Run heapsort on one heap. For each element of the second heap, run a binary search.

Another Solution O(n log n) amortized

Insert all items (by running delMin) of one heap into a hashtable. For each element of the second heap, run a hashtable search.