3/22/12

8. Homework

Due **3/29/12** before class

1. **Aggregate Analysis**

   Consider a sequence of n operations on a data structure, in which the cost c_i of the i-th operation is defined as $c_i = i^2$ if i is a power of 2, and $c_i = 1$ otherwise.

   (a) What is the worst-case runtime of a single operation?

   (b) Use aggregate analysis to determine the amortized cost per operation.

2. **Queue from Stacks**

   Assume we are given an implementation of a stack, in which PUSH and POP operations take constant time each. We now implement a queue using two stacks A and B as follows:

   ENQUEUE(x):
   • Push x onto stack A

   DEQUEUE():
   • If stack B is nonempty, return B.POP()
   • Otherwise pop all elements from A and while doing so push them onto B. Return B.POP()

   a) Show how the following sequence of operations operates on the two stacks. Suppose the stacks are initially empty.
   Enqueue(1), Enqueue(2), Enqueue(3), Enqueue(4), Dequeue(), Enqueue(5), Enqueue(6), Dequeue()

   b) Why is the algorithm correct? *(Hint: It might help to argue which invariants hold for A and B.)*

   c) What is the worst-case runtime of a single ENQUEUE operation? What is the worst-case runtime of a single DEQUEUE operation?

   d) Prove using the accounting method that the amortized runtime of ENQUEUE and DEQUEUE each is $O(1)$. Argue why your account balance is always non-negative.

   e) Use aggregate analysis to show that the amortized runtime of ENQUEUE and DEQUEUE each is $O(1)$.

3. **Union-Find**

   (a)
   ```
   for(i=1; i<=16; i++) x[i]=MAKE-SET(i);
   for(i=1; i<=15; i+=2) UNION(x[i],x[i+1]);
   for(i=1; i<=13; i+=4) UNION(x[i],x[i+2]);
   UNION(x[12],x[13]); UNION(x[1],x[8]); UNION(x[1],x[10]);
   FIND-SET(x[4]);
   FIND-SET(x[16]);
   ```
   Assume an implementation of the Union-Find data structure with a disjoint-set forest with union-by-weight and path compression.

   Show the data structure after every line of code. What is the answer to the `FIND-SET` operation?

   (b) Describe how to construct a sequence of m `MAKE-SET`, `FIND-SET`, `UNION` operations, n of which are `MAKE-SET` operations, that takes total $\Omega(m \log n)$ time when using a disjoint set forest with union by weight only. Assume $m \geq n$. *(Hint: Construct the data structure such that m `Find-SET` operations take $\Omega(m \log m)$ time.)*