CMPS 2200 Introduction to Algorithms – Fall 14

11/11/14

8. Homework

Due **11/20/14** at the beginning of class

Remember, you are allowed to turn in homeworks in groups of two.

1. **Binary Counter (5 points)**
   Use aggregate analysis to show that, over a sequence of n increment operations on a binary counter, the amortized runtime of one such increment operation is $O(1)$.
   (Hint: Study the flipping behavior of every single bit $A[i]$.)

2. **Queue from Stacks (5 points)**

   [See Homework 2 from CMPS 1600 as a reference.]
   Assume we are given an implementation of a stack, in which PUSH and POP operations take constant time each. We now implement a FIFO 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()
   • Else, pop all elements from A and immediately push them onto B. Return B.POP()

   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.

3. **Decision tree (5 points)**
   Below is the code for *Bubble Sort*:

```
void bubbleSort(int A[1..n]){
  for(int i=1; i <= n; i++)
    for(int j=n; j >= i+1; j--)
      if(A[j]<A[j-1])
        swap(A[j],A[j-1]);

}
```

   Draw the decision tree for Bubble Sort for an array $A[1..3]$ of $n = 3$ elements. Annotate the decision tree with comments indicating the part of the algorithm that a comparison belongs to.

4. **Lower bound for comparison-based searching (5 points)**

   Consider the problem of searching for a given key in a sorted array of n numbers. Use a decision tree to show a lower bound of $\Omega(\log n)$ for any comparison-based search algorithm. *(Hint: The decision tree needs to represent the output of the search algorithm in its leaves. What should be stored in the leaves?)*