CMPS 6610 Algorithms – Fall 18

10/15/18

6. Homework

Due **10/22/18** at the beginning of class

Justify all your answers.

1. **Covering points (10 points)**

   Let $A = \{a_1, a_2, \ldots, a_n\}$ be a set of n real numbers. Assume $a_1 \leq a_2 \leq \ldots \leq a_n$. We can consider these numbers to be points on the real line. The task is to determine the smallest set of unit-length (closed) intervals so that the union of the intervals covers (i.e., contains) all of the input points. Consider the following two greedy approaches:

   (a) Let I be an interval that covers the most points in A. Add I to the solution, remove the points covered by I from A, and repeat.

   (b) Add the interval $I = [a_1, a_1 + 1]$ to the solution, remove the points covered by I from A, and repeat.

   Prove or disprove the correctness of these greedy approaches.
   (Hint: One of these approaches is correct, the other one is not.)

2. **Binary search in multiple arrays (12 points)**
   While binary search runs efficiently on a sorted array, inserting a new number into the array takes linear time. We are going to see that we can store n numbers in a set of sorted arrays, such that search as well as insertion can be implemented to run efficiently.

   (a) As a warmup, use aggregate amortized analysis to analyze the amortized runtime of incrementing a binary counter. (It helps to look at the flipping behavior of each bit.)

   (b) Now consider the following data structure for storing n numbers:
   Let $n_{k-1}n_{k-2}\ldots n_1 n_0$ be the binary representation of n, using $k = \lceil \log(n+1) \rceil$ bits. The data structure stores k sorted arrays $A_0, \ldots, A_{k-1}$, where A_i stores exactly 2^i numbers if $n_i = 1$, and A_i is empty if $n_i = 0$. With this setup the data structure does indeed store $\sum_{i=0}^{k-1} n_i 2^i = n$ numbers.

      i. Please describe how to efficiently search in this data structure, and analyze the worst-case running time.

      ii. Please describe how to insert a number into this data structure. Analyze the worst-case running time and as well as its amortized running time.