Algorithms I

Tutorial 4

September 2, 2016

Problem 1

Try it yourself.

Problem 2

This can be done using counting sort. We create an array *cnt* of size $k+1$ all values initialized by 0. For each integer x in the input, we increment $cnt[x]$ by 1. After this, we update $cnt[i] = cnt[i-1] + cnt[i]$ for $1 \leq i \leq k$ (in order from 1 to k). So, now $cnt[i]$ denotes number of integers in the input which are less than or equal to i. So, given a query $[a, b]$ there are two cases:

- $a = 0$, answer is $cnt[b]$
- $a > 0$, answer is $cnt[b] - cnt[a-1]$

Problem 3

We can solve radix sort to sort the numbers in the range 0 to $n^2 - 1$ by representing each number as two digit base-n number. We can use the same algorithm here (using $x - 1$ instead of x for ordering).

Problem 4

The key idea is that it is sufficient to consider only the end points. So, if we take the maximum number of active intervals over all end points, that will also be equal to the maximum number over all other points. Let's say we have a list of pair of integers P. For each interval $[l, r]$, we add $(l, START)$ and (r, END) to P. $START$ and END can be any distinct integers. We sort P using the comparison: $(x, y) < (x', y')$ iff $x < x'$ or ($x = x'$ and $y = START$ and $y' = END$). Assuming P is already sorted, here is the pseudo code:

```
$answer = 0, activeCount = 0$
for all $(x, y) \in P$ do
    if $y = START$ then
        $activeCount := activeCount + 1$
        if $answer < activeCount$ then
            $answer = activeCount$
        end if
    else
        $activeCount := activeCount - 1$
    end if
end for
```

Problem 5

Ignore this problem.