Solution to Problem 3

Graded by: Group 3 (John Cantera, Yosong Liu, Andrew Wilson)
Adopted from a solution by: John Cantera

Problem Statement

Consider a set of intervals on the real line $I_1, I_2, \ldots, I_n$. A coloring of theses intervals is an assignment of colors to the intervals such that two intervals that overlap must be assigned different colors. Give an $O(nlg(n))$ algorithm for finding a minimum coloring of a set of intervals. **Note:** assume intervals are closed and $O(nlg(n))$ is the time required to sort n values.

Algorithm

We will present an algorithm which determines the maximum number of overlapping intervals that occurs within a set of intervals. In doing so, we will then know the minimum number of colors needed such that all overlapping intervals are assigned a different color.

Notation: Let a set of intervals $I = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where each (x_i, y_i) pair denotes the starting, x_i, and ending, y_i, boundaries for a given interval.

Algorithm 1 Interval Coloring Algorithm

```
begin
  Let
     Array A ← {x_1, ..., x_n} from set I     // array indices range from 1 to n
     Array B ← {y_1, ..., y_n} from set I
  Sort(A)
  Sort(B)
  i = j = 1
  color_max = color = 0
  while i ≤ n do
        begin
          if (A[i] ≤ B[j])
            then
                 Increment color by 1
                 Increment i by 1
                 if (color > color_max)
                   then
                        color_max = color
             else
                 Decrement color by 1
                 Increment j by 1
        end
  return color_max
end
```

Run Time Analysis

Assuming a sorting algorithm of cost $O(nlg(n))$ is used to sort arrays A and B, we need only worry about the cost of the steps in the **WHILE** loop. Since these steps essentially comprise a **MERGE** type operation between two arrays of n elements, the cost is $O(n)$ and we get,

$$T(n) = O(nlg(n)) + O(n) = O(nlg(n))$$

Grading Policy

Points

0-7	for workable algorithm
0-3	for run time analysis