Bradley Lamitie

2/28/2017

CMPT 435

Assignment 2

Idea behind my algorithm:

My algorithm will go through the list of intervals and choose the one with the earliest ending time to start with. It will update the current point to the ending time of said interval. It will then remove all conflicting intervals and repeat the process again.

Input: Array A which has n intervals in it, each with a start and end time.

Classes = 1; //1

CurrentPoint = A[0] //1

A[0] delete //1

While A isnt empty //Worst case(n)

For[i in A] //executed n \* n

if(currentPoint < A[i].end && currentPoint > A[i].start)

Remove A[i] //1 \* n\*n

End if

End For

Current point = A[0].ending //1

A[0].delete //1

Classes += 1; //2

End While

Return Classes //1

O(n^2)

Proof:

The greedy Algorithm is not optimal, x: # of lectures in classroom.

Optimal solution is optimal d

G = {g1, g2 ... gk} gi = i-th lecture selected by G

O = {o1, o2 ... om} oi = i-th lecture selected by O

Lecture r is the first different lecture selected by greedy solution,

Claim Gr must finish before Or

If claim is true replace Or with Gr, Since Gr finishes before Or O is not optimal

So O = g1 ... gk | Ok+1 Ok+2 ... Om

G must collide with O1, Ok+1 must be incompatible with g1 otherwise Ok+1 must have been examined by g1

**Contradiction**

**2.**

The idea behind my algorithm is that we will analyze each segment and sort them by ending times descendingly. We pick the first segment covering the first point specified. The ending time of that segment is set to be the newest righthand point. We continue this process until we reach the end point.

Input Sorted Array A of n segments with a left endpoint l, and a right endpoint r. , a segment to be covered [0, M]

Rightpoint = 0 ; //1

segmentCount = 0; //1

while(rightpoint < M) //n

for [i in A.size] //n\*n

if(A[i].start < rightpoint && rightpoint < A[i].end)

Rightpoint = A[i].end //1 \* n\* n

segmentCount +=1 //2 \* n\* n

A[i].remove // 1 \* n\* n

End if

End for

End while

Return segment count //1

O(n^2)

Proof:

Op does not include s1

Let Sk be the segment in Op that covers 0

r1>=rk

S1 = the segment that covers 0 and has the rightest end point

Op’ = Op - {sk} + {s1}

Op’ has same amount of pieces as OP

Op’ is also optimal

**QED**

**The other two cases are solved similarly.**