MCS-375: Algorithms: Analysis and Design Handout #G2

San Skulrattanakulchai

Gustavus Adolphus College Oct 30, 2015

Interval Graph Coloring Problem

GT: Ch 10.2

Problem

Given n lectures, each with a start time and a finish time, find a minimum number of lecture halls to schedule all lectures so that no two occur at the same time in the same hall.

Example

Given lectures $A(1,3], B(1,5], C(1,2], D(4,6], E(4,8], F(7,10], G(7,11], H(9,13], I(12,14], J(12,15]$, an optimal schedule uses 3 halls. It schedules lectures C, D, F, J in the first hall, B, G, I in the second hall, and A, E, H in the third hall.

We will number the halls by positive integers 1, 2, 3, ...

High-level Algorithm

A greedy method for solving this problem works as follows.

> **for** each lecture ℓ in order of increasing start time **do**
>   assign to ℓ the smallest hall that has not been assigned to
>   any previously assigned lectures that overlap ℓ

Low-level Algorithm

$d \leftarrow 0$ // d contains the largest hall ever used
$A \leftarrow \emptyset$ // A is an empty queue of available halls that are ever used
/* Priority queue Q contains all the endpoints (with references to their intervals). */
/* We break ties in favor of finish time; for equal finish times break the tie by start time. */
$Q \leftarrow \{\, S[i], F[i] : 1 \leq i \leq n \,\}$
while $Q \neq \emptyset$ **do** {
    $x \leftarrow$ extract-min(Q)
    if x is a start time **then** {
        if $A \neq \emptyset$ **then** // there is some hall that was used but is available right now

$c \leftarrow$ dequeue(A) // so reuse it
else { // all halls that's ever used are being used
$d \leftarrow d + 1$ // so need to get a new hall
$c \leftarrow d$
}
assign hall c to the interval of x
} **else** { // x is a finish time
$c \leftarrow$ hall of the interval of x
enqueue(c, A) // release hall c and put it in the pool
}
}

Running Time

The algorithm has an $O(n \log n)$ time bound because there are $2n$ endpoints, and we do an insert and an extract-min on each endpoint, with the work of priority queue operations dominating all other work.

Correctness

Let k be the maximum number of halls ever used at any point in time in our algorithm, i.e., k is the value of the variable d at the end of the algorithm. Consider the point in our algorithm when hall number k is assigned to a lecture. At that time, every hall from hall 1 to hall $k-1$ is assigned to some lecture. Our algorithm only lets a lecture occupy a hall from its start to finish time, but no more. This means that the set of lectures in the input instance contains k mutually overlapping lectures. This means that k is a lower bound on the number of halls required by any algorithm. Since our algorithm uses exactly k halls, it uses the least number possible!

Remarks

1. The technique used in the above proof of correctness is called a *lowerbounding argument* (or *upperbounding argument* for a maximization problem). It is one of the two common techniques of proof used to show correctness of greedy algorithms.

2. This is precisely the Minimum Graph Coloring Problem on interval graphs.

3. The queue A in the algorithm can be any data structure that supports constant time insertion and deletion.