CS 477/677 Analysis of Algorithms

Homework 6

Due April 21, 2014

1. [50 points] (U&G-required)

A search engine company needs to do a significant amount of computation every time it recompiles its index. For this task, the company has a single large supercomputer, and an unlimited supply of high-end PCs.

They have broken the overall computation into n distinct jobs, labeled $J_1, J_2, \ldots, J_n$, which can be performed completely independently of one another. Each job consists of two stages: first it needs to be *preprocessed* on the supercomputer, and then it needs to be *finished* on one of the PCs. Let's say that job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC.

Since there are at least n PCs available on the premises, the finishing of the jobs can be performed fully in parallel – all the jobs can be processed at the same time. However, the supercomputer can only work on a single job at a time, so the system managers need to work out an order in which to feed the jobs to the supercomputer. As soon as the first job in order is done on the supercomputer, it can be handed of to a PC for finishing; at that point in time a second job can be fed to the supercomputer; when the second job is done on the supercomputer, it can proceed to a PC regardless of whether or not the first job is done (since the PCs work in parallel); and so on.

Let's say that a *schedule* is an ordering of the jobs for the supercomputer, and the *completion time* of the schedule is the earliest time at which all jobs will have finished processing on the PCs. This is an important quantity to minimize, since it determines how rapidly El Goog can generate a new index.

Give a polynomial-time algorithm that finds a schedule with as small a completion time as possible.

Note: to prove that your greedy strategy yields the optimal solution, you have to prove that the problem has the *greedy-choice property*.

2. [50 points] (U&G-required)

Suppose you have n video streams that need to be sent, one after another, over a communication link. Stream I consists of a total of b_i bits that need to be sent, at a constant rate, over a period of t_i seconds. You cannot send two streams at the same time, so you need to determine a schedule for the streams: an order in which to send them. Whichever order you choose, there cannot be any delays between the end of one stream and the start of the next. Suppose your schedule starts at time 0 (and therefore ends at time $\sum_{i=1}^{n} t_i$, whichever order you choose). We assume that all the values b_i and t_i are positive integers.

Now, because you are just one user, the link does not want you taking up too much bandwidth, so it imposes the following constraint, using a fixed parameter r:

(*) For each natural number $t > 0$, the total number of bits you send over the time interval from 0 to t cannot exceed rt.

Note that this constraint is only imposed for time intervals that start at 0, not for time intervals that start at any other value.

We say that a schedule is *valid* if it satisfies the constraint (*) imposed by the link.

The problem. Given a set of n streams, each specified by its number of bits b_i and its time duration t_i, as well as the link parameter r, determine whether there exists a valid schedule.

Example. Suppose we have n = 3 streams, with

$(b_1, t_1) = (2000, 1)$, $(b_2, t_2) = (6000, 2)$, $(b_3, t_3) = (2000, 1)$,

and suppose the link's parameter is $r = 5000$. Then, the schedule that runs the streams in the order 1, 2, 3 is valid, since the constraint (*) is satisfied:

$t = 1$: the whole first stream has been sent, and $2000 < 5000 \bullet 1$

$t = 2$: half of the second stream has also been sent, and $2000+3000 < 5000 \bullet 2$.

Similar calculations hold for $t = 3$ and $t = 4$.

(a) [10 points] Consider the following claim:

There exists a valid schedule if and only if each stream i satisfies $b_i \leq rt_i$.

Decide whether you think the claim is true or false, and give a proof of either the claim or its negation.

(b) [40 points] Give an algorithm that takes a set of n streams, each specified by its number of bits b_i and its time duration t_i, as well as the link parameter r, and determines whether there exists a valid schedule. The running time of your algorithm should be polynomial in n.

4. [20 points] (G-required)

Exercise 16.1-2 (page 422).

5. [20 points] (Extra credit)

Exercise 16.3-1 (page 436).