1 Book 4.12: Video Streams

Background

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're 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 $r \cdot t$.

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.

Questions:

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 ∃ a valid schedule.

1.0.1 Question 1:

Claim: $\exists$ *a valid schedule* $\iff$ *each stream i satisfies* $b_i \leq r \cdot t_i$.

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

1.0.2 Question 2:

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.

1.1 Question 1:

False.
Proof by example, giving the case: $s_1 = (1,1)$, $s_2 = (2,1)$ and $r = 1$. We see here that this is clearly an acceptable order as there is no space and for <u>t=1</u> $\implies b_1 \leq r \cdot t$ i.e. $1 \leq 1 \cdot 1$ so $*$ holds. Similarly for <u>t=2</u> $\implies b_2 \leq r \cdot t$ i.e. $2 \leq 1 \cdot 2$ so $*$ holds. However this will fail the case since for s_2, $b_2 \nleq r \cdot t_2$ i.e. $2 \nleq 1 \cdot 1 \implies$ fails the case. $\square$

1.2 Question 2:

1.2.1 Mathematical Formulation

Given an input of n video streams, $s_1 = (b_1, t_1), s_2 = (b_2, t_1), ..., s_n = (b_n, t_n)$ where b_i = the load of stream i and t_i = the duration for how long it takes to transmit all data. We will say that in the correct order, there cannot be any delays between the end of one stream and the start of the next. i.e. time starts t=1 and goes to $\sum_{i=1}^{n} t_i$. We will denote the current time as T_i for when each step i begins where $T_1 = 1$. Additionally there is a rule that $b_i \leq r \cdot T_i$ for each stream i where r is a fixed parameter.

1.2.2 Solution

Important Confusing Data Structures:

- VideoStream[n] **streams** : Holds the video streams storing them as objects able to retrieve their (d_i, t_i)

The main functionality of this is to sort the video streams by d_i/t_i, and then check to see if the condition is met. If at any point it is not, then there does not exist a valid arrangement.

Algorithm 1 Method

```
procedure SORTSTREAMS(streams, r)
    SORT(streams, Comparator())
    $T \leftarrow 1$
    for $s_i \in streams$ do
        if $b_i > r \cdot T$ then
            return null
        $T += t_i$
    return streams
procedure COMPARE($d_a, t_a, d_b, t_b$)
    $pos_a \leftarrow d_a/t_a$
    $pos_b \leftarrow d_b/t_b$
    return $pos_a - pos_b$
```

1.2.3 Correctness

Proposition 1.
Propose that by sorting via d_i/t_i we gaurantee $\exists$ a valid ordered input.

Proof.
The motivation behind this is using the equality $d_i \leq r \cdot T_i \implies d_i/T_i \leq r$ must also hold. Therefore, we see that the i^{th} stream is depenent on the ratio between the load and the current time. $\therefore$ We develop our rational this way by saying that if we order them then in the order of their ratio, then those with lower b_i compared to t_i would appear first, i.e. $d_1/t_1 \leq d_2/t_2 \leq ... \leq d_n/t_n$. If $\exists$ a valid output then this will give us a valid output. This is due to the fact that if you were to rearrange any part, then the ratios would be inverted and then streams would be worse off. □

1.2.4 Analysis

Let N be the total number of streams being considered.

Proposition 2. *The <u>space complexity</u> of this algorithm is **O(N)***

Proof.
This is due to the fact that all of our data is stored within the object array of size N and two integers T *and* r.

Giving us a space complexity of **O(N)**

□

Proposition 3. *The <u>time complexity</u> of this algorithm is **O(N·log(N))***

Proof. This is the case because our algorithm takes $(N \cdot log(N))$ to sort based off of this comparator and then linearly searches through, (N), doing the single check statement (1)

Giving us a time complexity of **O(N·log(N))**

□