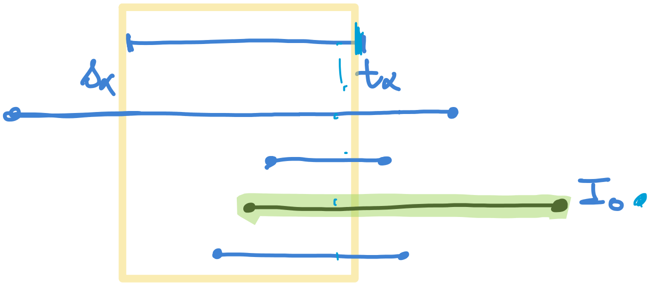
Question 1

Given: $\boxed{n \text{ intervals } I_1 .. I_n}$

Find: $\boxed{S \subseteq \{I_1 .. I_n\}}$ s.t. $\forall j \le n$, I_j overlaps with some interval in S.
and $|S|$ is minimum.

Observation:

Let $I_\alpha = [s_\alpha, t_\alpha]$ be interval with least finish time



Let Z = overlap(I_α)

$I_0 \in Z$ be interval with largest finish time

Then, it is better to keep I_0 in sol^n.

Key Property: $\exists$ an opt sol^n that contains I_0.

Hint: exchange idea.

Algo sketch:

1. Let $J = \{I_1 .. I_n\}$ be input
2. Let $I_\alpha = [s_\alpha, t_\alpha]$ be interval with least finish time
3. Let Z = overlap(I_α)
4. Let $I_0 \in Z$ be interval with largest finish time
5. Solve $J^* = J \setminus$ overlap(I_0) s.t. end time of the interval is less than I_0's end time **.

- Return opt(J^*) $\cup \{I_0\}$

Correctness: Show opt(J) = opt(J^*) + 1

Claim 1: $opt(J) \leq opt(J^*) + 1$

Show that if a committee of size $opt(J^*)$ exists for J^* then we can create a committee of size $opt(J^*) + 1$ for J.

- All intervals w/ start time > I_0's end time can be covered by a committee of size $opt(J^*)$
- All intervals w/ start time < I_0's end time can be covered by I_0 because their end time must be at least I_0's end time.

$\therefore \exists$ a committee of size $OPT(J^*) + 1$.

Claim 2: $opt(J) \geq opt(J^*) + 1$

- None of the students in $\{ I : I \in Overlap(I_0) \wedge end(I) < end(I_0) \}$ can cover any student in J^*. Thus, the best way to cover students in J^* is $opt(J^*)$
- No student in J^* can cover any student in $J \setminus J^*$ b'coz start times of students in J^* are greater than $end(I_0)$ and end times of students in $J \setminus J^*$ are at most $end(I_0)$. Thus, at least one committee member is required to cover them.

$\therefore opt(J) \geq opt(J^*) + 1$.

Tutorial-2

Question 3

The final tree (all leaves synchronized) would have signal propagation time for each leaf be the same as maximum sum of delays of all edges on path from root to some leaf in the original tree (call it D). This is necessary as well as sufficient.

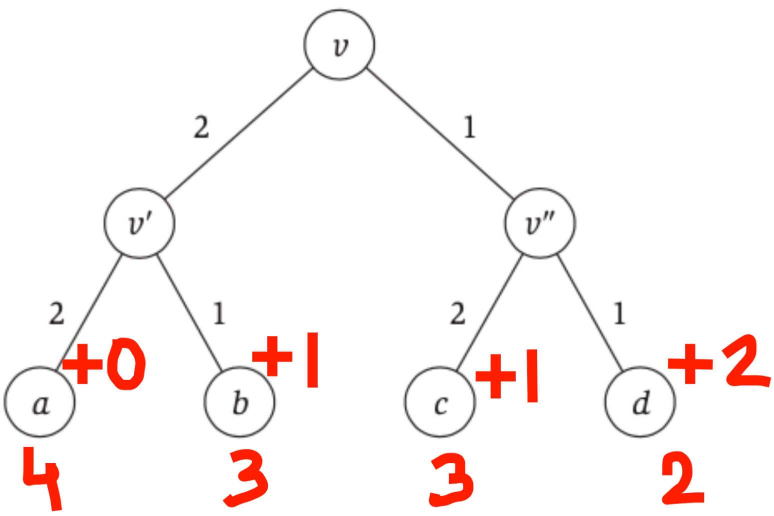
For every leaf, note down the "increase required" in total delay while reaching this leaf in order to make it equal to the desired value, D.

<u>IMPORTANT OBSERVATION:</u>

Now, for any two leaves with a common parent node, their root-to-leaf path differs only in the last edge. So, if they have a different value of "increase required", then this difference can only be created at the last edge on their paths.

Eg. in the diagram, D = 4, the "inc. reqd." for a, b, c, d is 0, +1, +1, +2. For parent v', we need to create a difference of 1 in the paths from v to a and v to b. So increase edge delay (v', b) by 1. Propagate 0 "increase required" above. Similarly for v", increase edge delay (v", d) by 1 and propagate 1 "increase required" above.

So the **greedy step** is to only change the delays of leaf edges, and propagate the "increase required" to nodes in the level above.



Let I be the input instance (binary tree with "n" leaves). Then **reduced input instance**, I', would be I without its leaf nodes or their corresponding edges. (i.e. I' will have n/2 leaves)

Thus the **algorithm** is simply to modify value of edge delays for the last layer and then reduce the size of the tree (removing last layer) till no edges remain. At each step we would maintain this value of "change required" for each <u>node</u>.

PROOF: The above important observation (that two siblings' root-to-leaf path differs only at the last edge) along with induction is a sufficient proof that this method would give the optimal answer.

Question 4

Given: 'n' different pieces of s/w
- Can obtain only 1/month
- cost/license = 100 INR
- Becoming expensive using exp growth rate (r_j)
- r_i ∀ license

Observation: Cost of all license at start is same. To reduce cost, the ...

$\ldots\ r_i > r_j \ldots$
$(K-1)$ K
↓
higher

Obs → strategy

Say r_j scheduled at K^{th} time
r_i scheduled at $(k-1)^{th}$ time

Cost of scheduling these 2 ignoring others

$100\, r_i^{k-1} + 100\, r_j^{k}$

Objective min $100(r_i^{k-1} + r_j^{k})$

$\boxed{Cost(r_i > r_j) < cost(r_i < r_j)}$

$\therefore \boxed{r_i > r_j}$

Key Property:- For any 2 licenses, purchased at order $\boxed{i \ \& \ j}$
- $r_i > r_j$ for the cost to be minimum

Algo:- Sort the licenses in order of decreasing cost of appreciation
- Buy the licenses in that order

$n \log n < O(n^2)$