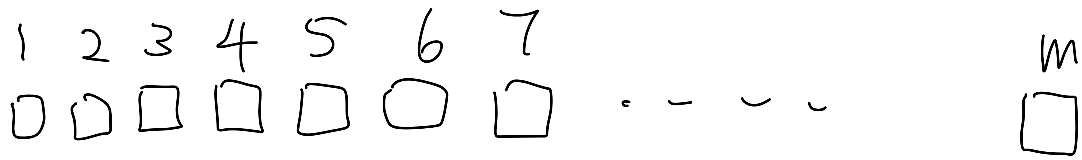
1 2 3 4 5

□ □ □ □ □

$$[(d_0, l_0), (d_1, l_1), (d_2, l_2), (d_3, l_3) \cdots (d_{n-1}, l_{n-1})]$$

$(d_i, l_i) =$ this course takes d_i slots and should stay in $[1, l_i]$

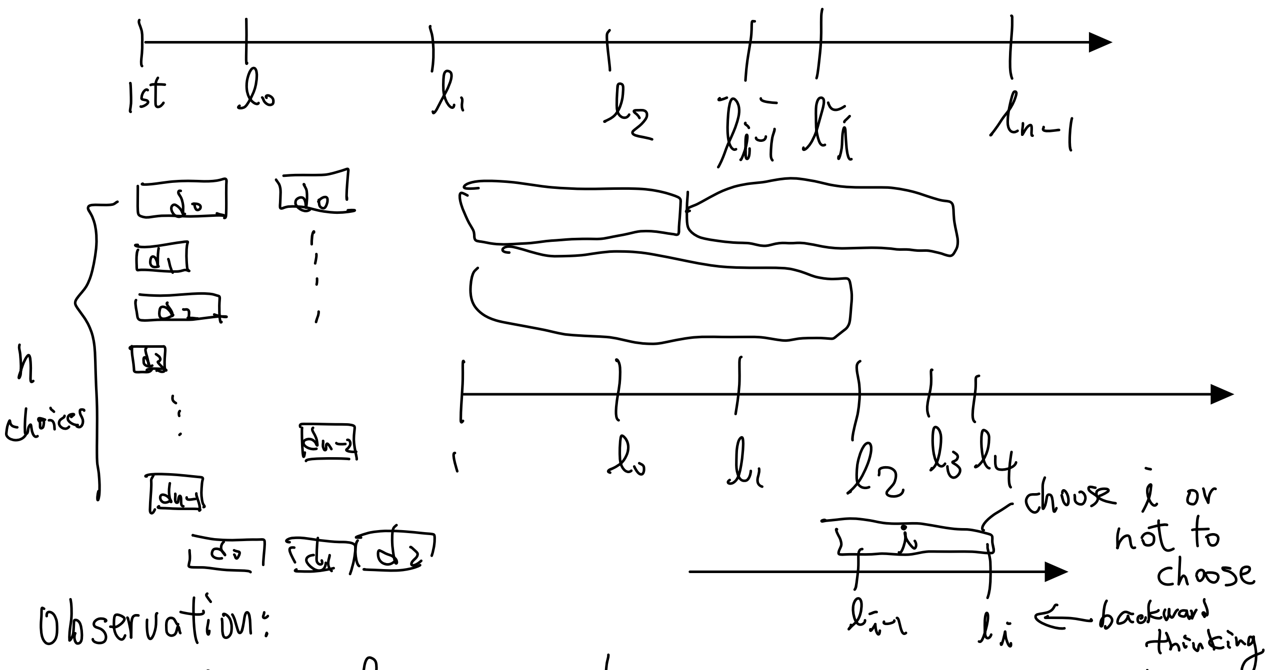
n courses



dp[t] = # of tasks we can complete before and on t sort the courses by their l_i

$$[(d_0, l_0), (d_1, l_1) \text{ - - - - } (d_{n-1}, l_{n-1})]$$

$$\Downarrow$$



Observation:

before l_0: we have courses: $0 \sim n-1$

before l_1: we have courses: $1 \sim n-1$

⋮

before l_{n-1}: we have courses: $n-1 \sim n-1$

define:

$$S_\ell = \{ i \mid \ell_i \le \ell \}$$

(index)

= all courses that should be finished before ℓ

$C \in 2^{S_\ell}$: C is a subset of S_ℓ

and $|C|$ is the number of courses

$$D(C) = \sum_{i \in C} d_i$$ = the duration of all courses in this subset

$$f(\ell) = \underset{C \in 2^{S_\ell} \text{ and for all } i \in C,\ i \text{ is finished before } \ell_i}{\operatorname{argmax}} |C| \to \text{\# of courses}$$

= all combinations of S_ℓ that can achieve the maximal number of courses completed and every course i is finished before its deadline ℓ_i

$$f(l_i) = \{ c \cup \{i\} \mid \underline{c \in f(l_{i-1})} \text{ and } D(c \cup \{i\}) \le l_i \}$$

if this is ϕ, then we assign $f(l_i)$

$$= \{ c \cup \underbrace{\{i\} / \{j\}}_{\text{substitute } j \text{ with } i} \mid c \in f(l_{i-1}) \text{ and } j \in c \cup \{i\} \; \underbrace{D(c \cup \{i\} / \{j\}) \le l_i}_{\text{also finished before } l_i} \}$$

Observe:

for $f(l)$, we actually only need one $c \in 2^{S_l}$ that has $\underline{\text{the minimum } D(c)}$

because if $c^* = \arg\min_{c \in f(l)} D(c)$

and $D(\underbrace{c^* \cup \{i\}}_{\text{add } \{i\} \text{ course}}) > l$

then other $c \in f(l)$ and $D(c) > D(c^*)$ will also cause $D(c \cup \{i\}) > l$

minimize $D(c)$

i.e. define: $$f_2(l) = \underset{c \in 2^{S_l} \text{ and for all } i \in C, \ i \text{ is completed before } l_i}{\arg\max} (|C|, \underbrace{-D(c)})$$

then $f_2(l)$ will produce only 1 combination C

(we can randomly pick one if there are 2 combinations having same $D(c)$)

The $f_2(l_i)$ recursion becomes

let $c^* = f_2(l_i)$

$$f_2(l_i) = \begin{cases} C^* \cup \{i\} & \text{if } D(c^* \cup \{i\}) \le l_i \\ \text{choose } j = \underbrace{\underset{j \in C^* \cup \{i\}}{\arg\max} D(\{j\})} \text{ else} \\ \text{and assign } C^* \cup \{i\} / \{j\} \end{cases}$$

goal is to minimize $D(C)$

One thing to Take care is we previously define $S_l = \{ i \mid l_i \le l \}$

however, $l_i = l_j$ where $i \neq j$ might happen.

In this case, just define S_l as S_{l_i}

$$S_i = \{ j \mid j < i \}$$

and $$f(i) = \underset{\substack{C \in 2^{S_i} \text{ and for all } i \in C \\ i \text{ is completed before } l_i}}{\operatorname{argmax}} (|C|, -D(C))$$

$c^* = f(i-1)$

and $$f(i) = \begin{cases} C^* \cup \{i\} & \text{if } D(C^* \cup \{i\}) \le l_i \\ C^* \cup \{i\} / \underset{j \in C^* \cup \{i\}}{\operatorname{argmax}} d_j & \text{else} \end{cases}$$