PROBLEMA (CODEFORCES)

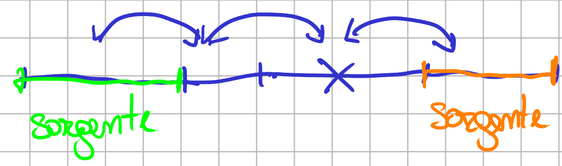
Ci sono N task. Ogni task ha una durata d_i e una deadline t_i.

Inoltre ci sono M precedenze: il task x_j va svolto prima di y_j.
(DAG)

Si può fare?

Idea: "Dijkstra" ordinando per $t_i - \cancel{d_i}$

$\underbrace{(1,3),\ (4,5)\ \text{no precedenze}}_{\text{controesempio}}$



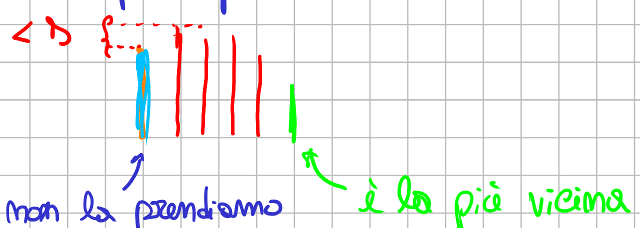
Serve preprocessing: $t_i := \min\{t_j : j \text{ discendente di } i\}$

- Principio dell'estremale: guardare l'oggetto che minimizza/massimizza qualche quantità
- Exchange argument: scambiare oggetti per trasformare una configurazione.

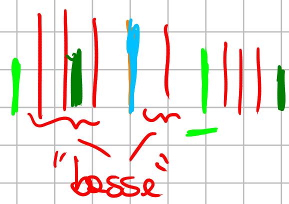
Riprendiamo towers

Ricordiamo: la greedy funziona.

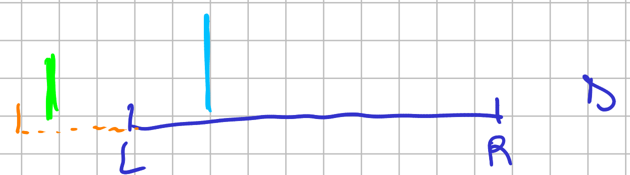
Idea: cercare di capire quando una torre non viene presa.



Claim: una torre NON viene presa se e solo se



Per ogni torre i, precalcoliamo S_i.



Conto gli $i \in [L, R]$ t.e. $S_i \le D$.

B(alkan)OI 2022 PEAKS

Data una matrice N righe, M colonne, contare il # di sottoinsiemi di righe tale che su ogni colonna si ottiene

$$a_{c r_1} < a_{c r_2} < \dots < a_{c r_k} > \dots > a_{c r_\ell}$$

dove k dipende dalla colonna

condizione $\Leftrightarrow$ non esistono $x<y<z$ t.c. $a_{cx} > a_{cy} < a_{cz}$

Una tripletta è compatibile $\Leftrightarrow$ $\nexists\, c$ t.c. $\nearrow$

Vogliamo scegliere righe t.c. triplette consecutive siano compatibili

$r_1 \quad r_2 \quad r_3 \quad r_4 \quad r_5 \quad \dots \quad r_{77}$

$(r_1, \underbrace{r_2, r_3}) \qquad (\underbrace{r_2, r_3}, r_4) \qquad \dots$

$(r_1, r_2) \to (r_2, r_3)$ per ogni tripletta compatibile

Aggiungiamo una riga -1 e N in cima e in fondo t.c. ogni tripletta che le contiene è compatibile

Grafo con le coppie (i, j) di righe con $i < j$ $(i, j = 1 \dots N)$ come nodi.

Come archi: $\exists$ arco tra (i, j) e (j, k) $\iff$ (i, j, k) sono compatibili

Path $\iff$ insieme di righe che va bene. DP su DAG.

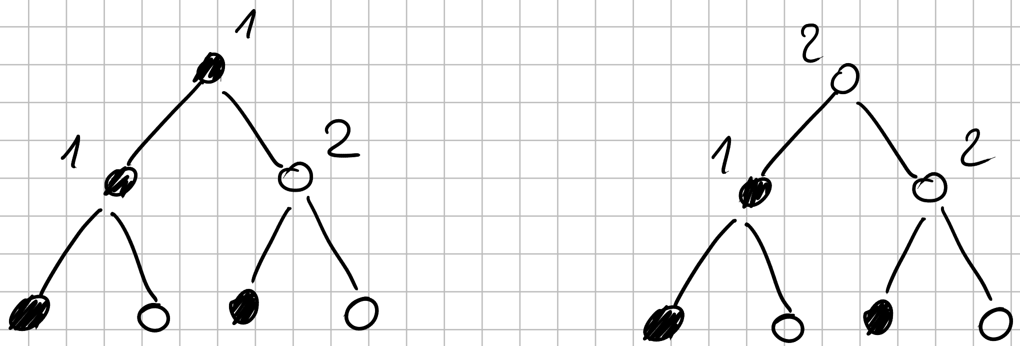
$O(N^3 M)$ per costruire il DAG

IOI 2022 CIRCUITS

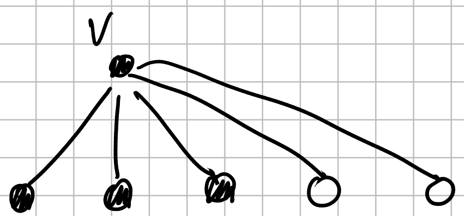
È dato un albero radicato con M foglie e N nodi interni, e un assegnamento di colori B/N alle foglie.

a ogni nodo interno assegnamo un threshold t_i (tra 1 e # figli): il nodo è nero se ha almeno t_i figli neri, ed è bianco altrimenti.

Quanti modi ci sono di assegnare i t_i in modo che la radice sia nera?

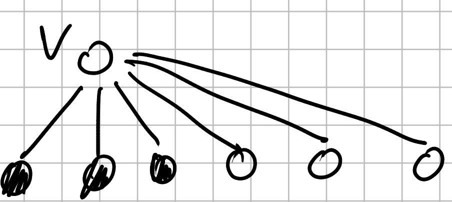


Step 1: data una colorazione, quanti assegnamenti la realizzano?



figli neri

1, 2, ... #N



figli bianchi

figli .... # figli − #B+1 = #N+1

(*) $\prod_v$ (# figli dello stesso colore di v)

Step 2 $\quad A_v = \{ \text{figli di } v \text{ dello stesso colore di } v \}$

$$(*) = \prod_v |A_v| = \prod_v \sum_{A_v} 1 = \sum_X 1 =$$

X = insieme dei modi di scegliere un arco $(v, \text{figlio di } v$ dello stesso colore$)$

per ogni v

Step 3 $\quad \text{RISPOSTA} = \sum_{\substack{\text{colorazioni} \\ \text{con 2 NERO}}} \sum_{X_C} 1 =$

\# modi di scegliere un arco padre-figlio in cui la radice ha un path a una foglia nera.

Step 4: dividiamo queste scelte in base alla foglie.

Per una foglia x, $\prod_{\substack{v \text{ non} \\ \text{antenato di } x}} \#\text{figli di } v = P_f$

La risposta è $\sum_{f \text{ neri}} P_f$.