

Armeanu Ana - Maria

Varianta 1

Ex 1.

Notăz intervalul initial cu $[A, B]$.

Idea: Sortez intervalele după $a_i \Rightarrow$ se renumerează în intervalele a.t. $a_1 < a_2 < \dots < a_n$.

La primul pas se alege intervalul cu $a_i < A$ și cu cel mai mare b_i . Notăz indicele intervalului ales cu "j". Inserați intervalul ales în vectorul soluție (sol). Capătul stâng al intervalului initial ~~stâng~~ ia valoarea capătului drept al intervalului inserat în soluție ultima oară. ($A = b_j$)

Se repetă primul pas până când $b_j \geq B$ sau până când verificătoarele rămase și nu găsesc niciunul pe care niciunul pot insera în soluție (Dacă $b_j \geq B \Rightarrow$ afisează soluția, altfel ~~nu~~, dacă $i \geq n \Rightarrow -1$)

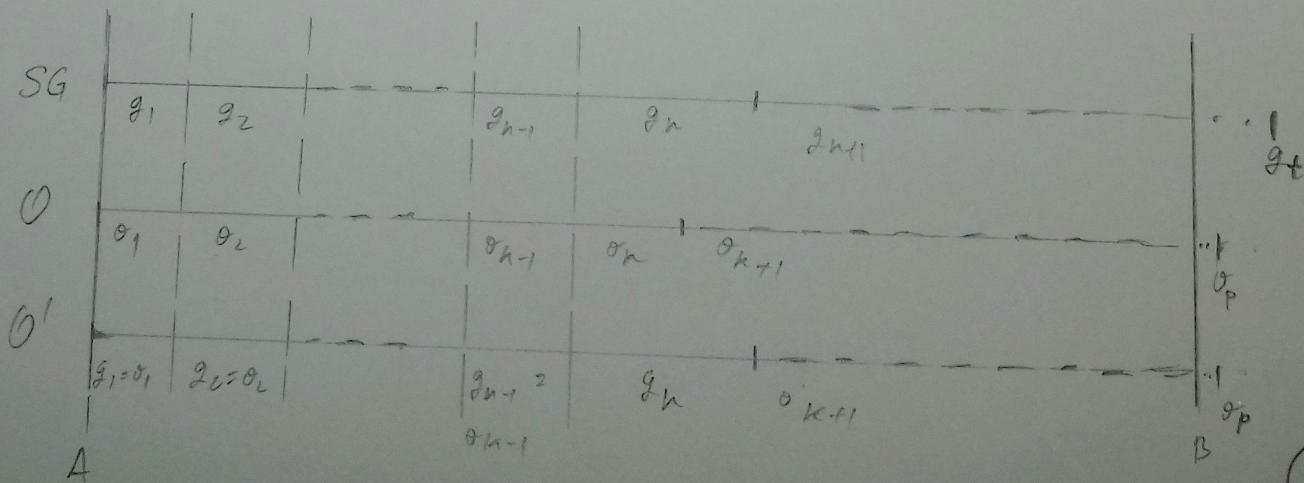
Corectitudine:

Notăz soluția greedy anunțată mai sus cu "SG".

Notăz intervalele continute de SG cu g_1 . \Rightarrow $SG = \{g_1, \dots, g_t\}$.

Fie \mathcal{O} o soluție optimă ce conține un număr maxim de elemente în comun cu SG. Notez elementele pe care \mathcal{O} le conține ca $\sigma_i \Rightarrow \mathcal{O} = \{\sigma_1, \dots, \sigma_p\}$. $\left. \begin{array}{l} \text{SG} = \{g_1, \dots, g_t\} \\ \mathcal{O} = \{\sigma_1, \dots, \sigma_p\}, \text{ cu } t \geq p. \end{array} \right\} \Rightarrow \exists \text{ un indice } k \leq t \text{ așa că } \sigma_k \neq g_k. \Rightarrow g_1 = \sigma_1, \dots, g_t = \sigma_t, \text{ dar } g_k \neq \sigma_k.$

Fie $k \leq t$ cel mai mic indice a.i. $g_k \neq \sigma_k$. La pasul la care SG a ales g_k , σ_k era neselectat, dar era o variantă compatibilă cu $g_1 = \sigma_1, \dots, g_{k-1} = \sigma_{k-1}$. Deoarece g_k a fost ales $\Rightarrow b_{g_k} \geq b_{\sigma_k} \Rightarrow g_k$ e compatibil și cu $\sigma_{k+1}, \dots, \sigma_p$ (pentru că σ_{k+1} începe înainte de b_{σ_k}). Atunci, în \mathcal{O} , σ_k poate fi înlocuit cu $g_k \Rightarrow \Rightarrow$ înălță o soluție posibilă: $\mathcal{O}' = \mathcal{O} \setminus \{\sigma_k\} \cup \{g_k\} \Rightarrow \Rightarrow |\mathcal{O}'| = |\mathcal{O}| \Rightarrow \mathcal{O}'$ e tot o soluție optimă, dar cu mai multe elemente în comun cu SG \Rightarrow contradicție \Rightarrow SG este o soluție optimă \Rightarrow algoritmul este corect.



VARIANTA 1

Ex. 2.

a) Idee: Planific activitățile după t_i , în ordine crescătoare

Corectitudine:

Se renumerează activitățile aș. $t_1 \leq t_2 \leq \dots \leq t_n \Rightarrow$ soluția greedy = permutarea identică id.

Conform soluției descrise, intervalul de desfășurare a activității $\alpha = [l_1 + \dots + l_{x-1}, l_1 + \dots + l_{x-1} + l_x]$.

Presupun prin reducere la absurd că soluția greedy nu e optimă.

Fie τ o permutare optimă cu nr minim de ~~inversii~~ inversions. Cum soluția greedy = permutarea identică $\Rightarrow \tau$ e cea mai apropiată de soluția greedy propusă.

id nu e optimă $\Rightarrow \exists (i, j)$ inversions: $i < j$ și $t_{\tau(i)} > t_{\tau(j)}$

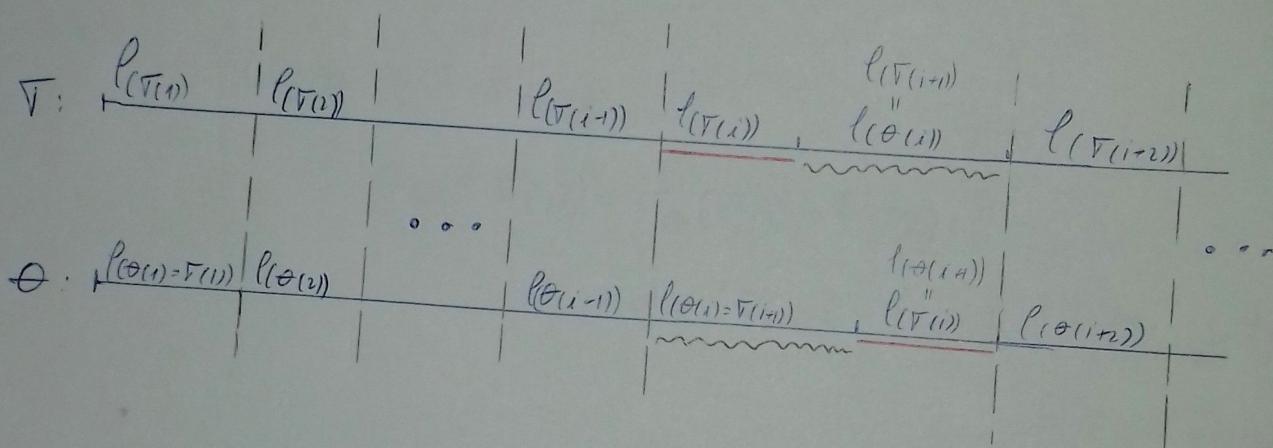
În plus, \exists cel puțin o inversions a.t. $j = i + 1$.

Fie θ permutarea obținută din τ prin interchimbarea elementelor de pe pozițile i și j .

Nr. de inversions $\tau >$ nr inversions $\theta \Rightarrow \theta$ seamăna mai mult cu id.

căsători,

$$\left. \begin{array}{l} \theta(i) = \tau(j) \Rightarrow \theta(i) = \tau(i+1) \\ \theta(j) = \tau(i) \Rightarrow \theta(i+1) = \tau(i) \\ \theta(z) = \tau(z), \forall z \neq i, j \end{array} \right\} \Rightarrow \text{Intervalele de desfășurare ale celor 2 permutări vor arăta astfel:}$$



$\theta(z) = \tau(z), \forall z \neq i, j \Rightarrow$ doar $\tau(i)$ și $\tau(i+1)$ pot avea întâlniri de ferite

Notăz durata activității $\tau(x)$, τ cu $d_{\tau}(\tau(x))$
în θ cu $d_{\theta}(\tau(x))$

$$\Rightarrow \left. \begin{array}{l} d_{\tau}(\tau(i)) = l_{(\tau(1))} + \dots + l_{(\tau(i-1))} + l_{(\tau(i))} \\ d_{\theta}(\tau(i)) = l_{(\tau(1))} + \dots + l_{(\tau(i-1))} + l_{(\tau(i+1))} + l_{(\tau(i))} \\ d_{\tau}(\tau(i+1)) = l_{(\tau(1))} + \dots + l_{(\tau(i-1))} + l_{(\tau(i))} + l_{(\tau(i+1))} \\ d_{\theta}(\tau(i+1)) = l_{(\tau(1))} + \dots + l_{(\tau(i-1))} + l_{(\tau(i+1))} \end{array} \right\} \Rightarrow \boxed{2}$$

$$\Rightarrow d_{\emptyset}(\bar{v}_{(i)}) = d_{\bar{V}}(\bar{v}_{(i+1)})$$

$$\text{Dar } d_{\emptyset}(\bar{v}_{(i+1)}) = d_{\emptyset}(\bar{v}_{(i)}) - \ell(\bar{v}_{(i)}) = \\ = d_{\bar{V}}(\bar{v}_{(i+1)}) - \ell(\bar{v}_{(i)}) < d_{\emptyset}(\bar{v}_{(i+1)})$$

Înărtierea lui $\bar{v}_{(i)}$ nu este

$$r_{\emptyset}(\bar{v}_{(i)}) = d_{\emptyset}(\bar{v}_{(i)}) - t_{(\bar{v}_{(i)})} = \\ = d_{\bar{V}}(\bar{v}_{(i+1)}) - t_{(\bar{v}_{(i)})}$$

$$\text{Dar } t_{(\bar{v}_{(i)})} > t_{(\bar{v}_{(i+1)})} \Rightarrow$$

$$r_{\emptyset}(\bar{v}_{(i)}) = d_{\emptyset}(\bar{v}_{(i)}) - t_{(\bar{v}_{(i)})} = \\ = d_{\bar{V}}(\bar{v}_{(i+1)}) - t_{(\bar{v}_{(i)})} < d_{\bar{V}}(\bar{v}_{(i+1)}) - t_{(\bar{v}_{(i+1)})} \\ \leq r_{\bar{V}}(\bar{v}_{(i+1)}) \\ \leq p_{\bar{V}} =$$

$$\Rightarrow \text{Înărtierea lui } \bar{v}_{(i)} \text{ nu este} < \text{înărtierea lui } \bar{v}_{(i+1)} \text{ nu este} \\ = \max_{1 \leq i \leq n} r_{\emptyset}(v_i)$$

\Rightarrow contradicție cu optimizarea lui $\bar{V} \Rightarrow$ id este
soluția optimă.

b) Contraexemplu:

$$n = 3$$

$$1. l_1 = 3 \quad t_1 = 7$$

$$2. l_2 = 4 \quad t_2 = 6$$

$$3. l_3 = 5 \quad t_3 = 5.$$

Planific activitățile în ordine crescătoare, după $l_i \Rightarrow$

$$\Rightarrow \text{ordinea } 1. 2. 3. \Rightarrow P_1 = 0, 1. \text{ se termină la } 3 \\ P_2 = 1, 2. \text{ se termină la } 7 \\ P_3 = 7, 3. \text{ se termină la } 12$$

$$\Rightarrow \text{întârzierea maximă} = 7. \quad (1)$$

Planific activitățile în ordine crescătoare după $t_i \Rightarrow$

$$\Rightarrow \text{ordinea } 3. 2. 1. \Rightarrow P_3 = 0, 3. \text{ se termină la } 5 \\ P_2 = 3, 2. \text{ se termină la } 9 \\ P_1 = 5, 1. \text{ se termină la } 12$$

$$\Rightarrow \text{întârzierea maximă} = 5 \quad (2)$$

Din (1) și (2) \Rightarrow soluția propusă la b) nu este corectă.

c) Contraexemplu:

$$n = 3$$

$$1. l_1 = 2 \quad t_1 = 6 \quad t_1 - l_1 = 4$$

$$2. l_2 = 3 \quad t_2 = 5 \quad t_2 - l_2 = 2$$

$$3. l_3 = 9 \quad t_3 = 12. \quad t_3 - l_3 = 3$$

Planific activitățile în ordine crescătoare după $t_i - p_i \Rightarrow$

\Rightarrow ordinea: 2. 3. 1. $\Rightarrow P_2 = 0$, 2. se termină la 3;
 $P_3 = 0$, 3. se termină la 11; } \Rightarrow
 $P_1 = 8$, 1. se termină la 14;

\Rightarrow întârzierea maximă = 8. (1)

Planific activitățile în ordine crescătoare după $t_i \Rightarrow$

\Rightarrow ordinea: 2. 1. 3. $\Rightarrow P_2 = 0$, 2. se termină la 3; }
 $P_1 = 0$, 1. se termină la 5; } \Rightarrow
 $P_3 = 2$, 3. se termină la 15;

\Rightarrow întârzierea maximă = 2 (2)

Din (1) și (2) \Rightarrow soluția propusă la c) nu este corectă.

a) Idee: Parcurește sirul de nr., numărul curent se va adăuga la vectorul al cărui ultim element este cel mai ~~mare~~^{mic} număr, mai mare decât cel curent; dacă nu există se "crează" un "vector nou".

Implementare: Voi folosi un vector "v" de vectori v_i , $i=1, n$, initial goi. ~~Pentru~~ Parcurește lista de nr. și caută locul nr. curent în vectorii care nu sunt goi (la pasul i vor fi $k+1$ vectori ce conțin nr.), prin căutare binară ($\log n$). Dacă subprogramul căutării binare returnează $-1 \Rightarrow$ nu am găsit un vector potrivit pentru nr. curent \Rightarrow bag nr. curent în următorul vector goi și incrementez k . $\Rightarrow O(n \log n)$

Initial, $k = -1$.

Corectitudine: Pentru simplitate, $k=0$, initial.

Notez sirul initial cu S . Fie s o secvență descăzătoare a lui S . $s = (x_{i_1}, x_{i_2}, \dots, x_{i_k}) \Rightarrow$

$i_1 \leq i_2 \leq \dots \leq i_k$ și $x_{i_1} < \dots < x_{i_k}$. Astfel, și decompunerea
 a lui S în subsiruri descrescătoare va conține un număr
 $\geq k$ de subsiruri (Dacă ar fi o decompunere cu mai puține
 subsiruri \Rightarrow cel puțin 2 din \bullet ar face parte din același
 subsir \Rightarrow subsirul nu e ordonat \Rightarrow contradicție)
 \Rightarrow și decompunere corectă a unui sir de numere, în
 ordine descrescătoare, va conține cel puțin m subsiruri,
 unde $m =$ numărul de numere din cel mai lung subsir
 element aflat în ordine crescătoare. (**)

Demonstrați că ultimul nr al vectorului k din

decompunere este unui sir crescător de lungime k . (*)
 \Leftrightarrow algoritmul construiește cel mult m subsiruri).

• Inducție după k :

1) $k=1 \Rightarrow (*)$ e adevarată

2) Fie H sir. Dacă algoritmul îl imparte în $k-1$ vectori
 \Rightarrow sfârșitul vectorului $k-1$ este unui sir
 crescător de lungime $k-1$.

3) Fie ℓ subsirul lui S , format din elementele vectorilor: $1 \dots k-1$. Se ~~vor~~ să pasul curent se va adăuga elementul x în al k -lea vector. Conform (*)

$\Rightarrow \ell$ conține un subîn crescător și se termină cu ultimul element al penultimului vector: $p \Rightarrow \ell$ va conține:
 $(x_{i_1}, x_{i_2}, \dots, x_{i_{k-2}}, x_{i_{k-1}}, p)$

Cum se adaugă în vectorul $k \Rightarrow$ ultimul element din vectorul $k-1 <$ ultimul element din vectorul $k_n \Rightarrow$
 \Rightarrow sirul $s_1 = (x_{i_1}, \dots, p, x) \in S$ este crescător, de lungime k .

• Sfârșitul demonstrației prin inducție.
 \Rightarrow Conform (*) și (**), soluția optimă (l minimă) va conține m subsiruri \Rightarrow algoritmul este corect.

b) Soluția este corectă, dar complexitatea acesteia este $O(n^2)$. În cazul în care doar elementele sunt în ordine crescătoare, ~~lor~~ ~~lor~~ parcurge la pasul i sunt parcursi $n-i$ elemente, unde $n = m$. total de numere).

Corectitudine:

Presupun că soluția nu este corectă \Rightarrow

- un subsir din soluție nu e ordinat crescător
- construiesc mai multe subsiruri decât este nevoie

- Din descrierea algoritmului \Rightarrow contradicție. ①
- ~~Din~~ La ex 3.a) am demonstrat că soluția optimă conține exact m subsiruri crescătoare ($m = m$. de numere din cel mai lung subsir crescător din sirul initial) (3)

Presupun că soluția de la punctul b) construiește mai multe subsiruri. Presupunerea este falsă pentru că algoritmul construiește exact același tip de soluție ca și algoritmul prezentat la punctul a). : subsiruri în care ultimul adăugat în subsir este cel mai mare număr, mai mic decât cel dinaintea sa. (în fel spus, elementul curent este plasat peste cel mai mic număr, mai mare decât el.) Conditia care creează un vector nou la punctul a) este „simulată” prin eliminarea subsirului terminat cu sirul initial și construirea următorului. (2)

Din ① și ② \Rightarrow soluția descrisă la punctul b) este corectă.