Varianta 2

3) Fie $A = \{ a_i = (p_i, h_i) \mid i = \overline{1,n} \}$

$a_i$ -> ~~un~~ activitate

$p_i$ -> profit

$h_i$ -> ultima oră la care poate fi programată

Definim forma soluțiilor $S = \{ (a_1, a_2, \dots, a_k) \mid k \leq n$ și $a_i$ este programată la ora $i \}$

Definim următorul algoritm pentru a determina o soluție cu profit maxim al activităților alese:

Definim operația „$>$" : $a_i > a_j \iff t_i > t_j$ sau ($t_i = t_j$ și $p_i > p_j$)

Algoritm: sortăm elementele mulțimii A descrescător conform operației „$>$".

soluție.pushback(primul element din A); priority-queue Q.

~~pentru i = 1, n~~ // priority queue este un heap cu proprietatea că în vf se află $a_i$ cu profitul cel mai mare dacă există $a_i, a_j$ cu $p_i = p_j$ și $t_i > t_j$, $a_i$ are prioritate

ultima_oră_alocată = ora primului

pentru i = 1, n-1

dacă ora evenimentului curent diferă de ultima oră alocată

cât timp ultima oră diferă de ora ev. curent ~~sau~~ și Q nu e vidă

extrag din Q un eveniment și îl aloc ultimei ore;

ultima oră --;

Q.push(evenimentul curent)

extrag un eveniment din Q și îl ~~adu~~ atribui orei curente;

altfel Q.push(evenimentul curent)

cât timp (ultima_oră_aloc > 0)

~~Q.pop~~ extrag din Q

adaug elementul extras din Q în soluție și îl aloc pentru ultima_ora_aloc

ultima_oră_aloc --;

scrie soluția.

Algoritmul oferă ca soluție o configurație corectă: ~~alocăm~~ mereu alocăm o ~~a~~ activitate unei ore $\leq$ ora maximă și fiecare oră are maxim o activitate alocată.

Demonstrăm că soluția oferită de algoritm este optimă.

$P(i)$: pentru o singură activitate $a_i$, răspunsul dat este chiar cel (răspuns corect)

P(2): pentru 2 activități $a_1 = (p_1, t_1)$
$a_2 = (p_2, t_2)$

I ~~$p_1 > p_2$~~ $t_1 > t_2$

~~sortarea~~ i) $p_1 > p_2$

soluția dată de algoritm este $a_1\ a_2$ (soluția corectă)

ii) $p_1 = p_2$

iii) $p_1 < p_2$ analog

II $t_1 = t_2$

dacă $t_1 > 1$ atunci soluția este $a_1 a_2$

dacă $t_1 = 1$ atunci soluția este $a_i$ cu proprietatea că $p_i$ este maxim. (corect)

III $t_2 > t_1$ analog I

Presupunem că soluția noastră coincide cu soluția optimă până la pasul $u$.

Demonstrăm pentru pasul $u+1$.

Presupunem că la pasul $u+1$ soluția noastră diferă de soluția optimă.

soluția noastră : $a_{k_1 max_1} a_{k_2 max_2} \dots a_{k_{max_{k-u}}} a_{k_{1u+1}}$

$a_{k_2 1}, a_{k_2 max_1} a_{k_2 max_2} \dots a_{k_2 u} a_{k_2 u+1}$

unde $a_{k_1 u+1}$ este activitatea pe care o alegem pentru următoarea oră

Deoarece alegem activitatea de profit maxim din activitățile

neluoate cu oră maximă mai mare ~~decât~~ sau egală cu ora curentă, înseamnă

~~Dacă $hr_{(max-u)} > hr_{(max}$~~ că I $p_{k_1 u+1} \geq p_{k_2 u+1}$ (care este neselectată)

dar atunci soluția optimă nu e optimă $\Rightarrow \not\Rightarrow$

II $p_{k_1 u+1} = p_{k_2 u+1}$

algoritmul nostru alege în acest caz activitatea cu oră mai mare $\Rightarrow hr_{k_1 u+1} \geq hr_{k_2 u+1}$

dacă $hr_{k_1 u+1} > hr_{k_2 u+1}$ $\Rightarrow$ soluția optimă nu e optimă (ne dă ia din pos. de ore pentru ulterioare alocări) $\Rightarrow \not\Rightarrow$

$\Rightarrow hr_{k_1 u+1} = hr_{k_2 u+1}$

$\Rightarrow$ În acest caz avem 2 activități de același profit și aceeași oră

$\Rightarrow$ optimalitatea coincide și alegerea activității curente nu este inoportună.

III $p_{k_1 u+1} < p_{k_2 u+1}$ $\Rightarrow \not\Rightarrow$ cu felul în care alge algoritmul

$\Rightarrow$ imposibil

$\Rightarrow$ $hr_{k_1 u+1} = hr_{k_2 u+1}$, $p_{k_1 u+1} = p_{k_2 u+1}$ $\Rightarrow$ deși ex. avem activități diferite, optimalitatea lor coincide

$\Rightarrow$ alegerea făcută de alg. e cel puțin la fel de optimă ca sol. optimă.

Complexitate spațiu: $O(n)$

timp: sortare $O(n \log n)$, selecție $O(n \log n)$ $\Rightarrow O(n \log n)$