



## Descrierea soluției - risc

Autor: prof. Constantin Gălățan  
C. N. „Liviu Rebreanu” Bistrița

### Cerința 1. 20 puncte

Se determină înălțimea **hmax** și poziția **pmax** a celei mai înalte persoane.  
Pentru fiecare poziție  $2 \leq i \leq \mathbf{pmax}$  se verifică dacă  $\mathbf{h[i]} > \mathbf{h[i-1]}$  și similar, pentru fiecare poziție  $i > \mathbf{pmax}$  se verifică dacă pentru persoana **i** aflată în dreapta acesteia se verifică dacă  $\mathbf{h[i]} < \mathbf{h[i-1]}$ . Pentru a determina dacă toate persoanele aflate în dreapta poziției **pmax** sunt mai înalte decât cele aflate în stânga se verifică condiția  $\mathbf{h[pmax-1]} < \mathbf{h[n]}$ .

### Cerința 2. Total - 80 de puncte

#### Soluția 1 – Complexitate $O(n*n)$ ,

- prof. Carmen Popescu, C. N. „Gheorge Lazăr” Sibiu

Se marchează valorile care trebuie mutate folosind un vector boolean. Simultan cu aceste mutări vom calcula și următoarele valori:

**mx** = maximul valorilor ce se mută la sfârșit

**mn** = minimul valorilor ce vor fi în dreapta maximului după mutări

Pentru fiecare înălțime  $\mathbf{h[i]}$ ,  $i < \mathbf{k}$  căutăm toate valorile din fața sa care sunt mai mari, adică  $\mathbf{h[j]} > \mathbf{h[i]}$  cu  $j < i$  (este încălcată prima condiție de mai sus), acestea vor fi ”mutate”.

Vom ”muta” apoi la sfârșit toate valorile aflate în dreapta maximului care nu respectă cerința a doua de mai sus, deci pentru fiecare înălțime  $\mathbf{h[i]}$  cu  $i > \mathbf{k+1}$  vom ”muta” la sfârșit valorile din fața sa, care sunt mai mici ca  $\mathbf{h[i]}$ , adică mutăm valorile  $\mathbf{h[j]}$  cu  $j > \mathbf{k}$  și  $j < i$  pentru care  $\mathbf{h[j]} < \mathbf{h[i]}$ .

”Mutăm” apoi valorile care au rămas în stânga maximului și care sunt mai mari decât minimul valorilor din dreapta (condiția a treia de mai sus), deci mai mari decât **mn**.

Acum toate valorile sunt în ”jumătatea” potrivită, însă s-ar putea ca elementele care se aflau inițial în dreapta maximului și nu au fost mutate să fie mai mici decât maximul valorilor mutate la sfârșit (încalcă condiția a doua de mai sus) și să trebuiască să le ”mutăm” la sfârșit. Vom ”muta” așadar la sfârșit valorile  $\mathbf{h[i]} < \mathbf{mx}$  cu  $i > \mathbf{k}$ .

O altă posibilitate de a ”aranja” șirul dat este să mutăm maximul la sfârșitul șirului și apoi să reluăm ”mutările” ca mai sus. Se va alege cea mai favorabilă din cele două variante.

Această soluție obține **30 de puncte** din **80** posibile.

#### Soluția 2 - Complexitate $O(n * \log n)$

- prof. Constantin Gălățan, C. N. „Liviu Rebreanu” Bistrița

Persoanele care nu se mută trebuie alese în așa fel încât să existe certitudinea că celelalte persoane pot fi mutate la capătul din dreapta al rândului. Astfel, va trebui să găsim lungimea **celui mai lung subșir crescător**, format din **cele mai mici numere din șir**. De asemenea, începând cu poziția **pmax** a maximului înălțimilor, vom determina un număr **Desc** reprezentând lungimea **celui mai lung subșir descrescător** format din **cele mai mari numere din șir**.

Atenție, **nu se aplica algoritmul clasic de programare dinamica!**

Se face o copie **c** a șirului **h** al înălțimilor și se sortează copia. Apoi, pentru fiecare poziție **i** din șirul **h**, se contorizează numărul de valori întâlnite în parcurgerea șirului **h** de la început spre sfârșit și care sunt cele mai mici valori și totodată consecutive în șirul **c**.



Deci pentru fiecare poziție  $i$ , în șirul  $h$ , se determină numărul  $Asc[i]$ , reprezentând numărul de valori aflate în ordine crescătoare, din intervalul de poziții  $1..i$ , astfel încât acestea să fie **cele mai mici valori din șir**. Pornind de la poziția  $pmax$ , mergând către sfârșitul șirului se determină un singur număr:  $Desc$ , reprezentând numărul de valori din intervalul de poziții  $[pmax, n]$ , aflate în ordine descrescătoare astfel încât acestea să fie **cele mai mari valori din șir**.

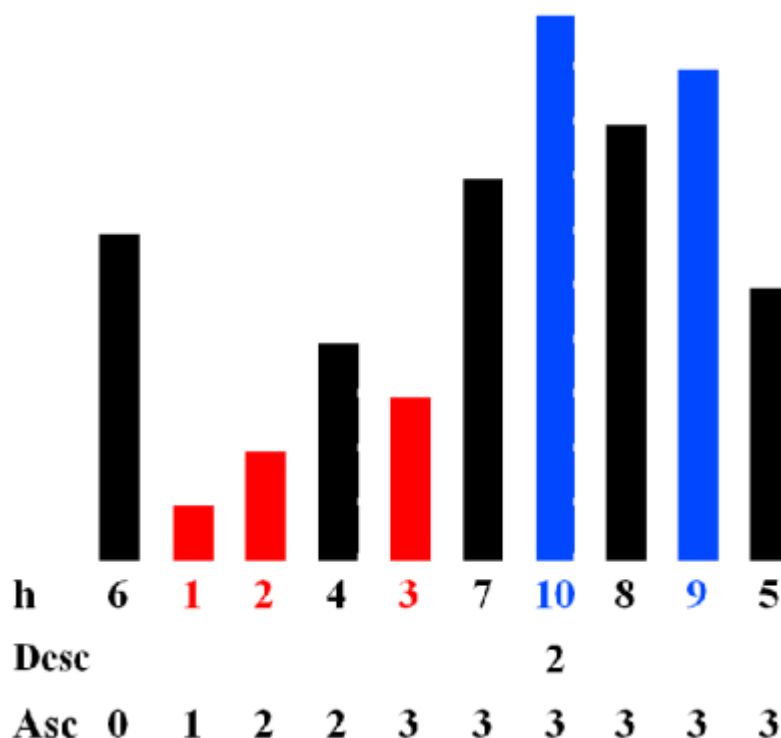
Fie  $M$  numărul de elemente care trebuie mutate. Dacă  $Asc[pmax] + Desc > Asc[n]$ , atunci maximul nu se mută la capătul randului și  $M = Asc[pmax] + Desc$ .

Un caz special este acela în care lungimea  $Asc[n]$  celui mai lung subșir crescător format din cele mai mici numere din șir, este mai mare decât  $Asc[pmax] + Desc$ . În acest caz, rămân pe loc toate valorile din acest subșir și se mută toate celelalte, inclusiv maximul.

În concluzie numărul maxim de persoane care nu se mută este:  $\max(Asc[n], Asc[pmax] + Desc)$ , iar

$$M = n - \max(Asc[n], Asc[pmax] + Desc)$$

Să luăm un prim exemplu:



Cu roșu s-au reprezentat cele mai mici numere din șir aflate în ordine crescătoare în parcurgerea șirului de la stânga la dreapta. Numărul acestora este  $Asc[i] = 3$  (pentru  $i \geq 5$ ). Cu albastru sunt figurate cele mai mari numere din șir aflate în ordine descrescătoare în parcurgerea șirului de la stânga la dreapta. Numărul lor este  $Desc = 2$ , iar capătul din stânga al acestui subșir începe la poziția  $pmax = 7$ .

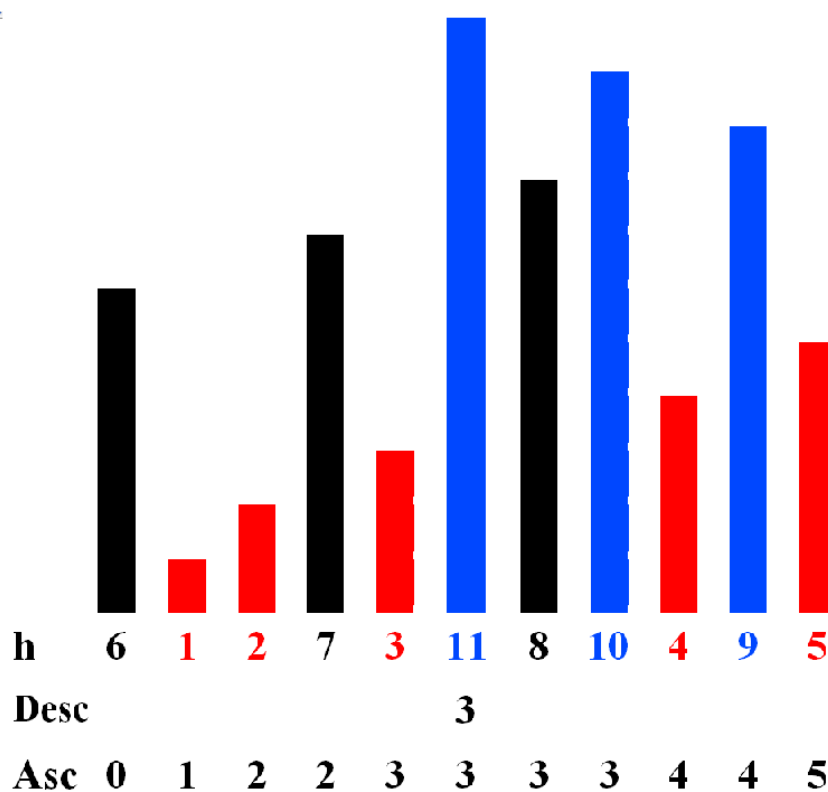
Numerele marcate cu negru trebuie să fie mutate. Ordinea mutării poate fi aleasă astfel încât să respecte cerința de *risc minim*.

$$pmax = 7, Asc[7] + Desc = 3 + 2 > Asc[n] = 3.$$

$$\text{Prin urmare, } M = n - (Asc[pmax] + Desc) = 10 - 5 = 5$$



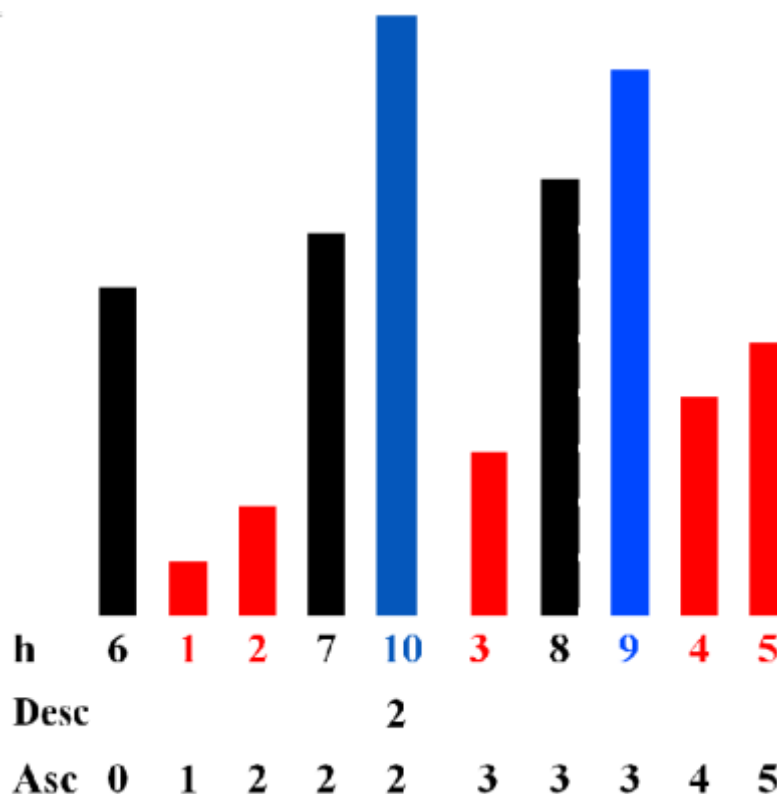
O altă situație este exemplificată mai jos:



Se observă că  $p_{\max} = 6, \text{Asc}[6] + \text{Desc} = 3 + 3 > \text{Asc}[n] = 5$ .  
 Așadar și în acest caz  $M = n - (\text{Asc}[p_{\max}] + \text{desc}) = 11 - 6 = 5$



Un ultim exemplu:



În acest caz,  $p_{\max} = 5$ ,  $\text{Asc}[5] + \text{Desc} = 2 + 2 < \text{Asc}[n] = 5$ , iar  $M = n - \text{Asc}[n]$ .

Așadar, pentru toate situațiile:  $M = n - (\max(\text{Asc}[p_{\max}] + \text{desc}), \text{Asc}[n])$

Pentru determinarea șirurilor **Asc** și a variabilei **Desc**, este necesară copierea șirului **h** și sortarea copiei, urmate de parcurgerea șirului **h**.

Determinarea valorii numărului **M** de persoane care trebuie mutate se face în timp constant conform relației de mai sus.

Complexitatea algoritmului depinde de sortare, deci  $O(n * \log n)$ .

O asemenea soluție obține **70** de puncte din **80** posibile.

### Soluția 3 - Complexitate $O(n)$

Soluția este de fapt identică cu **Soluția 2**, dar se coboară la  $O(n)$  complexitatea algoritmului de sortare cu o sortare prin numărare.

O asemenea soluție obține **80** de puncte din **80** posibile.