Problema torturi - Descrierea soluției

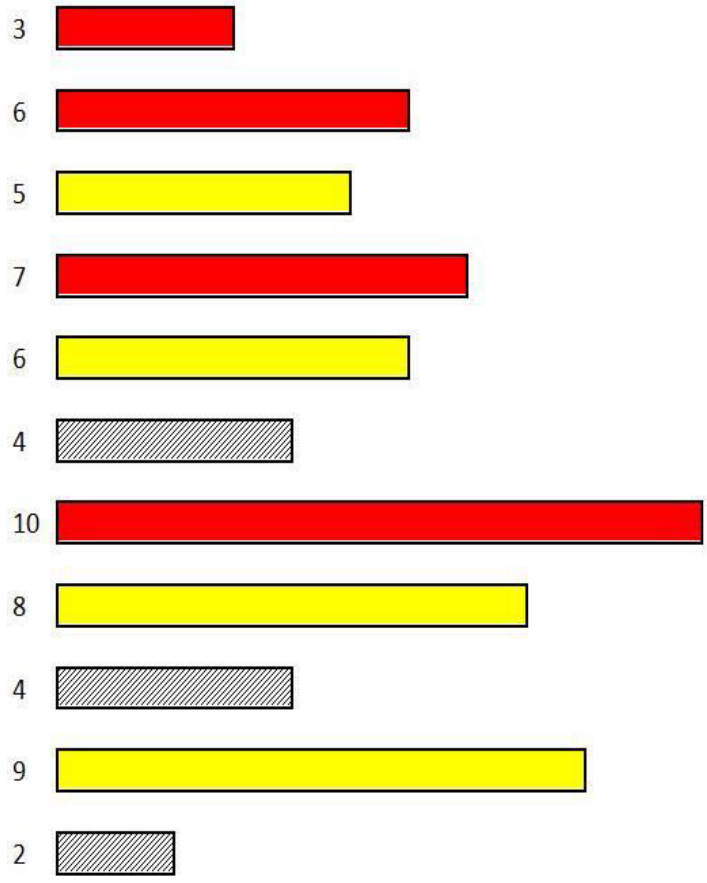
Autor: *prof. Constantin Gălățan*
C. N . „*Liviu Rebreanu*” Bistriţa

Soluție O(2^n) – 30 puncte

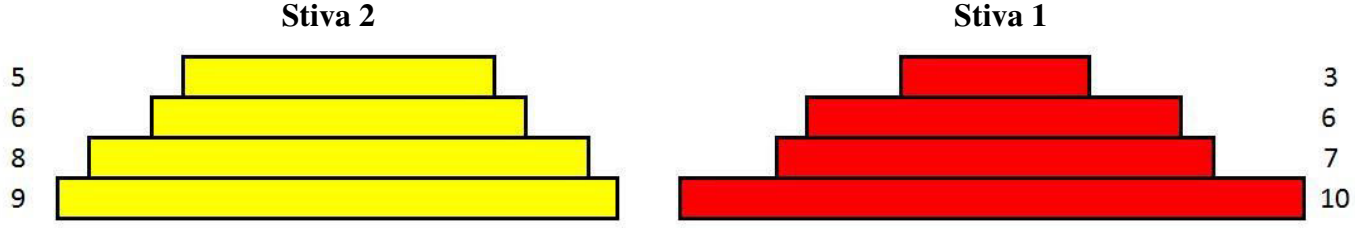
Se aplică metoda backtracking fără optimizări. Se generează toate posibilitățile de a plasa fiecare blat în prima sau a doua stivă.

Soluții O(n * n) și O(n * log n)

Vom descrie o abordare de tip greedy.
Fie următorul șir de valori, pentru diametrele blaturilor: **2**, **9**, **4**, **8**, **10**, **4**, **6**, **7**, **5**, **6**, **3**
Fiecare valoare trebuie așezată în una din cele două stive, începând cu valoarea 2.



O plasare optimă este următoarea:



Prima obsevație este aceea că valoarea v_{max1} cea mai mare din șir, (în exemplu $v_{max1}=10$) va elimina din stiva sa toate valorile plasate anterior pe acea stivă. Fie aceasta stiva 1. Pentru că avem interesul să salvăm cât mai multe dintre acestea, vom plasa în stiva a doua cât mai multe valori de pe poziții mai mici decât poziția maximului, aflate în ordine descrescătoare.

O **a doua observație**: toate valorile aflate între v_{max1} și a doua cea mai mare valoare aflată în șir după poziția maximului ($v_{max2} = 7$) pot fi salvate sau eliminate la alegere, în funcție de stiva în care decidem să le plasăm. Același lucru se poate spune și despre valorile aflate între $v_{max2} = 7$ si $v_{max3} = 6$.

Prin urmare vom determina cele mai mari valori, în ordinea din șir, începând cu poziția maximului v_{max1}. Găsim astfel valorile v_{max1}, v_{max2}, …, pe care le plasăm în prima stivă, împreună cu valorile intermediare care dorim să dispară. Pentru a obţine un număr maxim de valori pentru stiva a doua, căutăm cel mai lung subşir descrescător, format din cele mai mari valori care nu au fost plasate anterior în prima stivă (blaturile colorate cu galben). Valorile care care „strică" cel mai lung subşir descrescător vor fi plasate în stiva 1, pentru a fi distruse (blaturile haşurate).
Valorile v_{max1}, v_{max2}, … se detemină în `O(n*log n)`, prin sortare sau `O(n)` dacă se construieşte în mod direct stiva 1 ca o stivă de minime (sursa `torturi_100p.cpp`). Cel mai lung subsir descrescător se poate determina în `O(n * n)` sau `O(n*log n)`.

Soluție O(n*n) – 80 puncte

Ideea de rezolvare este cea descrisă anterior, iar determinarea celui mai lung subșir crescător se face în `O(n * n)`.

Soluție O(n * log n) – 100 puncte

Aceeași idee de rezolvare dar se utilizează un algoritm în complexitate `O(n * log n)` pentru determinarea celui mai lung subșir crescător.

Soluție O(n * log n) – 100 puncte

O altă abordare greedy. Parcurgem blaturile de la dreapta spre stânga. În prima stivă la bază va fi întotdeauna blatul maxim (de până la acel moment). Notăm pentru prima stivă înălțimea cu **`carry`**, **`nr`** - numărul maxim de blaturi la un moment dat în a doua stivă. Ne concentrăm atenția pe blatul de la baza stivei a doua. Blatul **`d[i]`** nou citit poate îmbunătăți înălțimea uneia din cele două stive, în caz contrar îl colapsăm așezându-l sub cel cu baza maximă, în prima stivă. Notăm cu **`x[k]`** dimensiunea blatului de la baza stivei a doua, de înălțime **`k`**, și pentru fiecare înălțime **`k`** atinsă de a doua stivă păstrăm în **`x[k]`** dimensiunea cea mai bună (cât mai mică). La fiecare pas:

a) poate să crească înălțimea primei stive dacă **`d[i]`** este maxim

sau

b) poate să crească înălțimea maximă a celei de a doua stive, dacă blatul cel nou se poate așeza (fără colapsare) sub blatul care dă înălțimea maximă **`nr`** și în acest caz se păstrează în **`x[nr+1]`** dimensiunea blatului curent

sau

c) poate îmbunătăți baza **`x[i]`** a uneia dintre stivele (stările) posibile pentru stiva a doua

Deoarece valorile din **`x[]`** sunt ordonate crescător (din modul cum este definit și gestionat tabloul **`x`**) se poate efectua căutare binară pe acest tablou pentru cazul c) .