

## Problema - clepsidru

### Varianta 1

autor prof. Cheșcă Ciprian  
Liceul Tehnologic "Costin Nenitescu" Buzău

Soluție în  $O(n \cdot \sum s_i)$  – C++

Valoarea cerută de primul experiment, adică după câte secunde ajung toate "boabele" de nisip în incinta de jos a ultimei clepsidre, se determină cu formula  $n + b - 1$ . Justificarea este următoarea : la fiecare secundă câte un "bob" de nisip cade în clepsidra următoare, așadar ultimul "bob" de nisip din incinta de sus a primei clepsidre întârzie  $b-1$  secunde până începe să cadă, iar pentru a cădea are nevoie de  $n$  secunde, câte clepsidre sunt.

Pentru a răspunde cerinței celui de-al doilea experiment, adică pentru a preciza numărul de "boabe" de nisip din incintele clepsidrelor, după cele  $k$  așezări consecutive, am făcut o simulare la nivel de secundă. La fiecare secundă am simulat curgerea boabelor de nisip într-unul sau în celălalt sens de curgere, în funcție de poziție. La citirea unei noi poziții am simulat, dacă era cazul, răsturnarea clepsidrului și reșezarea "boabelor" de nisip.

### Varianta 2

autor prof. Szabo Zoltan  
Inspectoratul Școlar Județean Mureș

Soluție de 100 puncte (rezolvare în  $O(n+k)$ ) – C++

În starea inițială toate boabele de nisip sunt în incinta de SUS. Știm, că în fiecare secundă, un bob de nisip trece printr-un orificiu, indiferent de starea 1 sau 2.

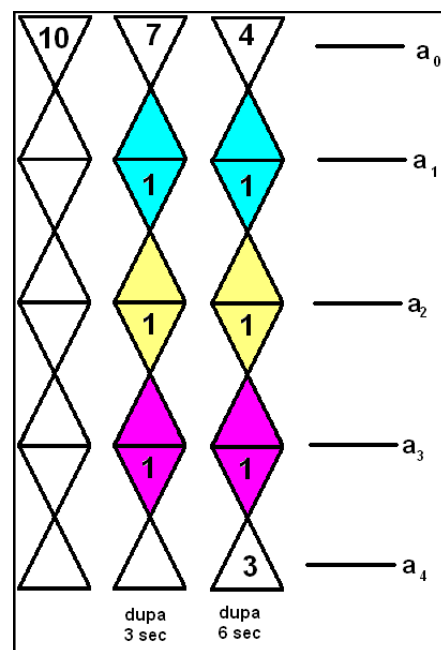
Vom observa:

-Nisipul în interiorul clepsidrului curge liniar, în orice poziție diferită de extremități avem un singur bob de nisip (sau 0, dacă încă nu a curs).

-Cantitate mai mare de nisip se poate acumula doar la cele două extremități, iar între extremități, în fiecare incintă avem câte un bob de nisip.

Fie  $n$  numărul de clepsidre,  $b$  numărul de boabe de nisip.

Asociind un vector pentru memorarea clepsidrului, acest vector  $a$  va avea  $n+1$  elemente numerotate  $a_0, a_1, a_2, \dots, a_n$ .



Incinta de sus este  $a_0$ , iar incinta de jos este  $a_n$ . Incintele intermediare sunt două incinte lipite (hașurate în imagine).

Noi deci trebuie să memorăm doar poziția SUS (inițial 0), poziția JOS (inițial 0), numărul boabelor de nisip sus BSUS (inițial b) și numărul boabelor de nisip jos BJOS (inițial nu contează, dar pentru prelucrarea problemei este indicat să fie 1).

Ne vom folosi de faptul că în T unități de timp din incinta de SUS vor curge T boabe de nisip la o distanță SUS+T.

Astfel după T unități de timp noile valori vor fi SUS, BSUS-T, JOS+T și BJOS=1.

Dacă observăm, că JOS+T a depășit dimensiunea clepsidrului, atunci aplică corecții de cumul pentru valorile lui JOS și BJOS. Iar dacă observăm, că SUS numărul BSUS de boabe este număr negativ, vom aplica de asemenea corecții pentru valorile lui SUS și BSUS.

Dacă clepsidra este în starea 2, procedeul este identic, dar cu direcție inversă.

La tipărire vom avea grijă să vedem ultima stare a clepsidrului, fiindcă în incintele intermediare vom avea 0-1 sau 1-0 boabe de nisip.

### Varianta nr.3

*autor prof. Rodica Moldovan  
C.N. "Gheorghe Șincai" Baia Mare*

### Soluție în $O(k \cdot n^2)$ – C++

Vom lucra într-un vector stare[1001], în care vom păstra numărul de boabe din fiecare clepsidră la un moment dat. Inițializăm stare[n] = b;

Vom folosi următoarele notații:

l - nivelul de pe care pornesc boabele,

k - nivelul pe care ajung boabele

s - număr de secunde într-o anumită stare

Pentru fiecare pereche citită (secunde poziție) vom actualiza (vectorul stare) numărul de boabe din clepsidre (nu la nivel de secundă ci funcție de l, k, s)

La fiecare actualizare, numărul de boabe b se inițializează cu numărul de boabe aflate în recipientul de pornire  $b = \text{stare}[l]$

Se disting următoarele situații:

**transformareJos**(int l, int k, int s)

- Dacă  $(l == 0) \ \&\& \ (k == 0)$

atunci suntem pe nivelul inferior și boabele nu coboară, deci poziția din vectorul stare se va actualiza cu b

- Dacă  $(l == k)$

atunci numărul de boabe rămase în recipientul inițial va fi  $\max(0, b - s)$ ;

- Dacă  $(k == 0)$   
atunci numărul de boabe adunate în recipientul inferior va fi  $\max(0, \min(b, s - (l - k) + 1))$ ;
- Dacă  $(s < l - k)$   
atunci timpul este mai mic decât timpul necesar să ajungă o boabă în recipient și vom actualiza cu 0
- Dacă  $(s \geq b + l - k)$   
atunci boabele au trecut deja prin recipient și vom actualiza cu 0
- Altfel boaba se afla în tranziție prin recipient și vom actualiza cu 1

analog pentru **transformareSus**(int l, int k, int s)

- Dacă  $(l == n) \ \&\& \ (k == n)$   
atunci suntem pe nivelul superior și boabele nu urcă, deci poziția din vectorul stare se va actualiza cu b
- Dacă  $(l == k)$   
atunci numărul de boabe rămase în recipientul inițial va fi  $\max(0, b - s)$ ;
- Dacă  $(k == n)$   
atunci numărul de boabe adunate în recipientul superior va fi  $\max(0, \min(b, s - (k - l) + 1))$ ;
- Dacă  $(s < (k - l))$   
atunci timpul este mai mic decât timpul necesar să ajungă o boabă în recipient  
și vom actualiza cu 0
- Dacă  $(s \geq b + (k - l))$   
atunci boabele au trecut deja prin recipient și vom actualiza cu 0
- Altfel boaba se afla în tranziție prin recipient și vom actualiza cu 1

#### Varianta 4

*autor. prof. Popa Daniel  
C.N. "Aurel Vlaicu" Orăștie*

#### Soluție de 100 puncte (rezolvare în $O(n+k)$ ) – Pascal

Ideea de rezolvare: Boabele stau în șir "indian" tot timpul, pozițiile ambelor capete fiind singurele date ce merită memorate. La deplasarea șirului trebuie să avem grijă ca limita superioară( $ls$ ) să nu treacă de  $n$  în jos și limita de jos( $lj$ ) să nu treacă de 0 în sus.