Problema - clepsidru

Varianta 1

autor prof. Cheşcă Ciprian
Liceul Tehnologic "Costin Neniţescu" Buzău

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

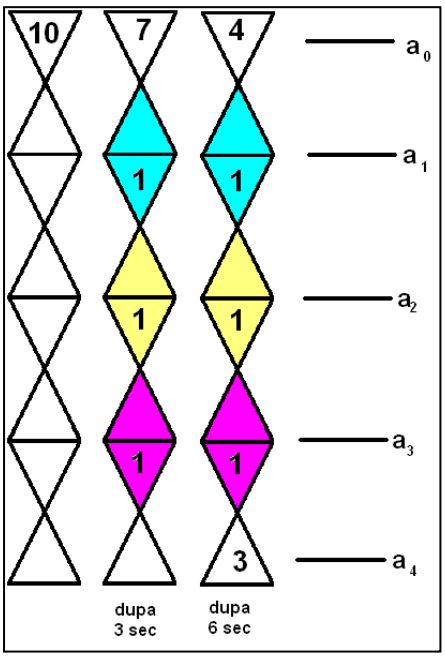
Valoarea cerută de primul experiment, adică după câte secunde ajung toate "boabele" de nisip în incinta de jos a ultimei clepsidre, se determină cu formula n + b – 1. Justificarea este următoarea : la fiecare secundă câte un "bob" de nisip cade în clepsidra următoare, aşadar ultimul "bob" de nisip din 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 cerinţei celui de-al doilea experiment, adică pentru a preciza numărul de "boabe" de nisip din incintele clepsidrelor, după cele k aşezări 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 funcţie de poziţie. La citirea unei noi poziţii am simulat, dacă era cazul, răsturnarea clepsidrului şi reaşezarea "boabelor" de nisip.

Varianta 2

autor prof. Szabo Zoltan
Inspectoratul Şcolar Judeţean Mureş

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



În starea iniţială toate boabele de nisip sunt în incinta de SUS. Ştim, 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 poziţie diferită de extremităţi 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 extremităţi, iar între extremităţi, 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 $a_0, a_1, a_2, \ldots, a_n$.

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

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

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

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

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

Dacă clepsidra este în starea 2, procedeul este identic, dar cu direcţie 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 Şincai" Baia Mare

Soluţie în O(k*n^2) – C++

Vom lucra într-un vector stare[1001], în care vom păstra numărul de boabe din fiecare clepsidră la un moment dat. Iniţializăm stare[n] = b;
Vom folosi următoarele notaţii:
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 poziţie) 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 iniţializează cu numărul de boabe aflate în recipientul de pornire b = stare[l]
Se disting următoarele situaţii:

transformareJos(int l, int k, int s)

- Dacă (l == 0) && (k == 0)

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

- Dacă(l == k)

atunci numărul de boabe rămase în recipientul iniţial 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 şi vom actualiza cu 0

- Dacă(s >= b + l - k)

atunci boabele au trecut deja prin recipient şi vom actualiza cu 0

- Altfel boaba se afla în tranziţie prin recipient şi vom actualiza cu 1

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

- Dacă (l == n) && (k == n)

atunci suntem pe nivelul superior şi boabele nu urcă, deci poziţia din vectorul stare se va actualiza cu b

- Dacă(l == k)

atunci numărul de boabe rămase în recipientul iniţial 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));

- Daca (s < (k - l))

atunci timpul este mai mic decât timpul necesar sa ajungă o boaba in recipient
şi vom actualiza cu 0

- Daca (s >= b + (k - l))

atunci boabele au trecut deja prin recipient şi vom actualiza cu 0

- Altfel boaba se afla în tranziţie prin recipient şi vom actualiza cu 1

Varianta 4

autor. prof. Popa Daniel
C.N. "Aurel Vlaicu" Orăştie

Soluţie de 100 puncte (rezolvare în O(n+k)) – Pascal

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