Problema 1 - Ciuruleț

Stud. Adrian Budău, Lucian Bicsi
Universitatea din București

Soluția 1 – 10 puncte

Se pot genera prin metoda backtracking toate șirurile inițiale posibile, urmând să se simuleze algoritmul pe fiecare șir și să se compare rezultatele. Dintre configurațiile inițiale care verifică condițiile din input, se va alege cea cu număr maxim de valori de `1`.

Complexitate: `O(2^N * N * log(N))`

Soluția 2 – 30 de puncte

Pentru cazul în care nu există caractere „`?`”, se poate aplica următoarea strategie: presupunem inițial că soluția conține numai valori de `0`; parcurgem valorile crescător și, dacă pe poziția `i` valoarea din șirul final este `1`, atunci pe poziția `i`, valoarea din șirul inițial trebuie să fie tot 1, iar valorile de pe pozițiile mai mari decât `i` multiple de `i` din șirul inițial nu vor modifica structura șirului final, deci putem să le considerăm inițial egale cu `1` (pentru a maximiza numărul de valori de `1` din șirul inițial). Bineînțeles, în același timp, valorile de pe pozițiile divizor de-al lui `i` trebuie să fie egale cu `0` în șirul inițial, însă observația nu este esențială în rezolvarea problemei în acest caz *(în esență, deoarece se garantează că șirul final este unul valid)*.

Complexitate: `O(N * log(N))`

Soluția 3 – 50 de puncte - 100 de puncte

Vom adapta strategia de mai sus pentru a rezolva Greedy și cazul mai general. Pentru aceasta, avem nevoie de următoarea observație:

Dacă pe poziția `i` din șirul final avem caracterul `1`, vom marca toți multiplii săi cu `1` și toți divizorii săi cu `0`. Dacă pe poziția `i` din șirul final avem caracterul `?`, atunci vom parcurge toți multiplii lui `i` și vom alege să punem `1` pe poziția `i` dacă și numai dacă nu regăsim o valoare de `1` printre aceștia. Explicația constă în următorul fapt: să presupunem că punem `1` pe poziția i. Atunci, aplicând algoritmul din enunț, vom obține valoarea `0` pe toate pozițiile multiplu de `i`. Dacă în șirul final avem un `1` pe o poziție multiplu de i, atunci alegerea noastră nu este una validă.

Se observă că, în acest caz, marcarea divizorilor unei poziții de `1` cu `0` nu mai e necesară, iar algoritmul se poate aplica într-o manieră foarte asemănătoare ciurului lui Eratostene.

Se poate observa că soluția este mereu unică, demonstrația urmând să fie lăsată pentru cititori.

Complexitate: `O(N * log(N))` sau `O(N sqrt(N))` (în funcție de implementare)