

Soluția la problema 2 – Numar

Rezolvare problemei se bazează pe următoarea observație: b_m nu poate fi mai mare decât $m \cdot a_1$

Fie x un număr natural și $m =$ numărul de multiplii ai celor n numere prime din intervalul $[1; x]$.
Avem (din principiul includerii și al excluderii):

$$m = [x/a_1] + [x/a_2] + \dots + [x/a_n] - [x/(a_1 a_2)] - \dots - [x/(a_{n-1} a_n)] + [x/(a_1 a_2 a_3)] + \dots$$

Noi îl știm pe m și trebuie să îl determinăm pe x . Aceasta se poate face prin căutarea binară a lui x în intervalul $[1..m \cdot a_1]$.

Totuși ne lovim de următoarea problemă: trebuie să generăm produsele de la numitorii care apar în expresia de mai sus. E evident că acestea se pot calcula generând o singură dată submulțimile mulțimii $\{a_1, a_2, \dots, a_n\}$ și reținând valorile corespunzătoare produselor. Nu este însă necesar să generăm toate submulțimile ci doar pe cele care reprezintă produse mai mici decât $m \cdot a_1!!!$ Se poate demonstra sau observa că și în cazul cel mai defavorabil, când șirul a e format din cele mai mici numere prime iar $m=15000$, numărul de produse care îndeplinesc acest criteriu este mai mic decât 10000. În consecință, vom utiliza un backtracking optimizat.

În final, trebuie să avem în vedere că prin căutarea binară s-ar putea să determinăm un număr $x_1 > x$, care să conducă la aceeași valoare a expresiei la care ne conduce și x , dar care să nu fie în șirul b !!

De exemplu, pentru $n=2$, $a_1=2$ și $a_2=3$,

$[11/2] + [11/3] - [11/6] = [10/2] + [10/3] - [10/6]$, dar 11 nu se află în șirul multiplilor numerelor 2 și 3!!

Pentru a determina rezultatul problemei, vom face un **for** descrescător plecând de la valoarea pe care ne-o furnizează căutarea binară și ne vom opri la primul număr care este multiplu a cel puțin unuia dintre numerele prime date prin șirul a . (o sursă care nu va face această determinare „prinde” doar o parte din teste).

Numărul de pași în cazul cel mai nefavorabil, nu depășește:

$10000(\text{backtracking}) + 10000 \cdot 24(\text{căutarea binară și calculul expresiei}) + 1000 \cdot 1000 (=a_1 \cdot n \text{ pentru cele două for-uri de după căutarea binară: ecartul între doi termeni din șirul } b \text{ nu poate fi mai mare decât } a_1)$.

Sursa care implementează acest algoritm se află în subdirectorul sol în fișierul `numar100.cpp`.

Soluții alternative:

1) interclasarea șirurilor de multiplii (optimizată: începînd cu multiplii numerelor a_n și a_{n-1} etc) = 50 puncte - `numar50.cpp`

2) generare directă a multiplilor și marcarea lor într-un vector de contorizare = 30 puncte - `numar30.cpp`

(Aceste două metode „au probleme” cu spațiul de memorie disponibil și cu limita de timp la testele pe care nu le iau).

3) generare directă a multiplilor și reținerea lor într-un arbore binar echilibrat (avl) = pana la 40 puncte;

(Depășește limita de timp la testele pe care nu le ia. Pot să apară probleme din cauza alocării memoriei) - `numar40.cpp`