Soluţia la problema 2 – Numar

Rezolvare problemei se bazează pe următoarea observaţie: b_m nu poate fi mai mare decât $m*a_1$

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

$$m = [x/a_1]+ [x/a_2]+...+ [x/a_n] - [x/(a_1a_2)]-...- [x/(a_{n-1}a_n)]+ [x/(a_1a_2a_3)]+...$$

Noi îl ştim pe m şi trebuie să îl determinăm pe x. Aceasta se poate face prin căutarea binară a lui x în intervalul $[1..m*a_1]$.
Totuşi 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 submulţimile mulţimii $\{a_1, a_2, ...,a_n\}$ şi reţinând valorile corespunzătoare produselor. Nu este însă necesar să generăm toate submulţimile ci doar pe cele care reprezintă produse mai mici decât $m*a_1$!!! Se poate demonstra sau observa că şi în cazul cel mai defavorabil, când şirul 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 consecinţă, vom utiliza un backtracking optimizat.

În final, trebuie să avem în vedere că prin căutarea binară s-ar putea să determinăm un număr x1>x, care să conducă la aceeaşi valoare a expresiei la care ne conduce şi x, dar care să nu fie în şirul b!!
De exemplu, pentru n=2, $a_1=2$ şi $a_2=3$,
[11/2]+ [11/3] - [11/6] = [10/2]+ [10/3] - [10/6], dar 11 nu se află în şirul multiplilor numerelor 2 şi 3!!
Pentru a determina rezultatul problemei, vom face un **`for`** descrescător plecând de la valoarea pe care ne-o furnizează căutarea binară şi ne vom opri la primul număr care este multiplu a cel puţin unuia dintre numerele prime date prin şirul a. (o sursă care nu va face această determinare „prinde” doar o parte din teste).

Numărul de paşi în cazul cel mai nefavorabil, nu depăşeşte:
10000(backtracking) + 10000*24(căutarea binară şi calculul expresiei) + 1000*1000 ($=a_1*n$ pentru cele două for-uri de după căutarea binară: ecartul între doi termeni din şirul b nu poate fi mai mare decât a_1) .
Sursa care implementează acest algoritm se află în subdirectorul sol în fişierul numar100.cpp .

Soluţii alternative:

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

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

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

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

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