Problema 1 – **game** 100 puncte

Autor: Drd. **Paul Diac**, Facultatea de Informatică Iaşi

Soluţii – drd Paul Diac, student Liana-Ştefania Ţucăr, stud. Cristian Vîntur

Pentru a obtine cele `30 puncte` pentru care valorile a_i nu depasesc `20` se pot genera toate submultimile posibile asociand fiecarei submultimi starea de castig sau de pierdere.

O stare de pierdere poate duce prin toate mutarile posibile doar in stari de castig pentru adversar. Dintr-o stare de castig se poate efectua cel putin o mutare care duce la o stare de pierdere. O repetitie a unui numar este inutila in analiza jocului, deoarece doua valori a_i egale vor ramane egale pe tot parcursul jocului.

Pentru solutii ce obtin mai multe puncte, vom considera cunoscute elementele de teoria jocurilor: jocul `NIM` si `numere Sprague-Grundy`. Pentru fiecare factor prim distinct al tuturor valorilor a_i vom asocia o gramada in jocul NIM, analiza pentru fiecare factor prim fiind independenta. Vom considera separat exponentii la care apare acest numar prim in toate numele din sir. Si aici doi sau mai multi exponenti cu valori egale se modifica similar, deci putem retine doar multimea exponentilor distincti. Numarul acestora este $\log(10^6)$, maxim 20 pentru cel mai mic numar prim. Putem calcula pentru fiecare submultime de exponenti valoarea `mex` (minimal-exclusive) asociata (Sprague-Grundy). Aceasta se poate calcula recursiv pentru toate submultimile, aplicand definitia. Daca suma `XOR` (similar cu NIM), a tuturor valorilor `mex` (minimal-exclusive) asociata (Sprague-Grundy). Aceasta se poate calcula recursiv pentru toate asociate pentru submultimile exponentilor ale fiecarui numar prim este nenula, Alice are strategie de castig. Daca este nula, Bob va castiga.

Pentru a numara numarul de mutari posibile distinctepe care le poate realiza Alice la prima ei mutare (in cazul in care ea are strategie de castig), putem incerca toate valorile posibile p^k prin care poate mutaAlice, actualizand de fiecare data sumaxor cu `mex`-ul multimii eliminate si adaugate si testand egalitatea cu 0.

Complexitate **O(NlogN)** .

Pentru a numara mutarile lui Bob in cazul in care el castiga, o posibila solutie este sa consideram toate posibilitatile pentru mutarea lui, si apoi sa testam daca ar fi putut exista o mutare anterioara a lui Alice, care sa produca impreuna cu mutarea fixata pentru Bob efectul de a readuce `suma xor` la 0. Se trateaza separat doua cazuri: atunci cand Alice foloseste acelasi numar prim ca si Bob, putem itera toate combitatiile celor doi exponenti cu care muta Alice si Bob.

Complexitate **O($N\log^2$)** .

Daca Alice foloseste alt numar prim, modificarile in suma xor devin independente. Putem precalcula pentru fiecare valoare `mex` posibila (care e maxim 20), prin cate mutari initiale se poate produce aceasta diferenta in suma xor. Mutarea lui Bob fixata este optima, daca exista o astfel de mutare in vectorul precalculat excluzand numarul ales de Bob (pe cazul acesta, numerele prime sunt diferite). Impreuna, aceste mutari se anuleaza in `suma xor` si deci se revine la 0, stare de pierdere pentru Alice.

Trebuie sa actualizam frecventele din acest vector a numarului prim ales de Bob, iterand toti exponentii posibili.

Complexitatea este **O(NlogN)** pentru acest caz.

Solutia care incearca toate combinatiile pentru primele doua mutari si retine doar mutarile distincte ale lui Bob, de complexitate **O($N^2 * \log^2$)** obtine punctaje partiale.