

Cifru

- descrierea soluției -

I) Problema se poate reformula astfel: fie mulțimea $\{1, 2, \dots, n\}$ și fie σ o permutare a acestei mulțimi care îndeplinește condiția:

$$\underbrace{\sigma(\sigma(\sigma(\dots\sigma(x)\dots)))}_{k \text{ ori}} = x \quad \text{pentru orice } x \in \{1, 2, \dots, n\}$$

- Pentru început, vom da soluția pentru un caz particular al problemei și anume pentru $K=3$: să notăm cu $f(n)$ numărul acestor permutări, și calculăm în câte moduri 1 (primul element al mulțimii) poate să revină pe prima poziție după K permutări. Există două posibilități și anume:

- 1) $\sigma(1) = 1$ și atunci evident că și după K permutări successive 1 va rămâne pe prima poziție; în acest caz, pentru celelalte $n-1$ elemente ale mulțimii vor exista $f(n-1)$ permutări care îndeplinesc condiția cerută;
- 2) elementul 1 formează un ciclu de lungime trei (evident cu încă alte două elemente din mulțime, fie acestea a și b . Astfel $\sigma(1) = a$, $\sigma(a) = b$, $\sigma(b) = 1$ și atunci aceste trei elemente revin pe pozițiile lor după trei permutări successive. În acest caz, cele două elemente a și b putem să le alegem în $(n-1)(n-2) = A_{n-1}^2$ moduri, iar pentru restul de $n-3$ elemente ale mulțimii vom avea $f(n-3)$ permutări care corespund cerinței din enunț.

Având în vedere aceste două posibilități putem scrie următoarea relație de recurență:

$$f(n) = f(n-1) + (n-1)(n-2)f(n-3) \quad \text{și} \\ f(n)=1 \quad \text{pentru } n \leq 1$$

- Dacă K este număr prim, un raționament asemănător ne conduce la următoarea relație de recurență:

$$f(n) = f(n-1) + (n-1)(n-2)\dots(n-k+1)f(n-k) \quad \text{și} \\ f(n)=1 \quad \text{pentru } n \leq 1$$

- Dacă K nu este număr prim, atunci raționamentul anterior trebuie efectuat pentru fiecare divizor a lui K .

De exemplu, dacă $K=6$, atunci elementul 1 poate să revină pe prima poziție după K permutări succesive dacă

- $\sigma(1) = 1$;
- elementul 1 formează un ciclu de lungime 2 (atunci revine pe prima poziție după două permutări, după patru permutări și în final după a șase permutări); celălalt element al acestui ciclu poate fi ales în A_{n-1}^1 iar pentru mulțimea formată din celelalte $n-2$ elemente avem $f(n-2)$ permutări;

- elementul 1 formează un ciclu de lungime 3 (atunci revine pe prima poziție după trei permutări și după a șase permutări);
- elementul 1 formează un ciclu de lungime 6 (atunci revine pe prima poziție după șase permutări);

Obținem astfel pentru cazul $n=6$, relația:

$$f(n) = f(n-1) + (n-1)f(n-2) + (n-1)(n-2)f(n-3) + (n-1)(n-2)\dots(n-5)f(n-6)$$

sau

$$f(n) = f(n-1) + A_{n-1}^1 f(n-2) + A_{n-1}^2 f(n-3) + A_{n-1}^5 f(n-6)$$

și

$$f(n)=1 \text{ pentru } n \leq 1$$

Generalizând, putem scrie recurența

$$f(n) = f(n-1) + \sum_{d=2}^k A_{n-1}^{d-1} f(n-d) \quad \text{pentru toți divizorii } d \text{ ai lui } K \text{ care sunt}$$

mai mici sau egali cu n .

Programul constă în implementarea acestei formule. Pentru a obține punctaj maxim, trebuie făcute câteva optimizări cum ar fi:

- calculul $A_{n-1}^{d_2}$ unde d_2 este un divizor a lui K trebuie făcut plecând de la valoarea $A_{n-1}^{d_1}$ calculată în prealabil, unde $d_1 < d_2$ și d_1 divizor a lui K
- reținerea valorilor calculate pentru funcția f într-un vector, pentru a evita calcularea în mod repetat a acestora.

Fără aceste optimizări, un program care implementează soluția descrisă primește 50 puncte.

II) O soluție bazată pe backtracking, care generează toate permutările și verifică pentru fiecare dacă îndeplinește condiția din enunț este corectă, dar se încadrează în timpul maxim de rulare/test doar pentru valori mici ale lui n și k . Pentru datele de test propuse, o astfel de soluție primește 20 puncte;

III) Se pot lua 30 puncte cu o astfel de soluție dacă se rulează programul în timpul concursului pentru toate combinațiile posibile sugerate de observația că

- pentru 30% din teste $N, K < 13$

și se rețin rezultatele într-o matrice de constante.

În subdirectorul soluții se află două surse de 100 puncte:

- stelian.cpp, care folosește o implementare recursivă a formulei;
- cosmin.pas, care folosește un algoritm iterativ și are, în consecință, o viteză de rulare sensibil mai bună.