Solutie **PermPower**
Stud. Sebastian Nechita - **Universitatea Babeş-Bolyai**

Solutie puncte bonus +10 puncte la solutiile partiale:

Pentru cazul de 10 de puncte pentru care q = e, putem sa descompunem permutarea p in ciclii si apoi sa calculam rezultatul ca fiind

k = cmmmc(len(ciclu1), len(ciclu2), …, len(ciclux))

Solutie O(n * k) - 20 puncte(10 p. oficiu + 10):

Efectiv compunem permutarea p cu ea insasi de k ori pana cand dam de permutarea q.

Solutie O(n * n + n * k) - ideea solutiei corecte:

Pentru fiecare pozitie i din {1,...n} , putem afla care este r_i minim a. i. p^ r_i (i) = q(i) (elementul sa fie pe pozitia corecta). Vom nota cu c_i , lungimea ciclului din care face parte elementul i in permutarea p. - Acest pas are complexitatea O(n*n) daca se genereaza efectiv primele n permutari.

Deoarece dupa c_i mutari, elementul i va ajunge pe pozitia i(este un ciclu), numarul cautat k este fie r_i , fie r_i + c_i fie r_i + 2 * c_i etc. Deci k e de forma r_i + a_i * c_i .

Un alt mod de a formula asta este urmatorul: pentru ca elementul de pe pozitia i in permutarea p^k sa fie cel corect, k % c_i = r_i .

Avand o astfel de ecuatie pentru fiecare pozitie i, vom avea n astfel de ecuatii.

Am gasit sistemul de n ecuatii cu o metoda in O(n * n). Daca nu stim sa rezolvam sistemul in niciun mod, putem efectiv sa incercam toate valorile pentru k pana cand sistemul este satisfacut. Desi complexitatea teoretica creste, in practica, daca pentru un k nu am gasit raspunsul corect la primele i ecuatii ne putem opri si sa incercam pentru urmatorul k de la inceput.

Solutie O(n * n + $\sqrt{n}$ * k) - 40 puncte(10 + 30):

O imbunatatire evidenta a algoritmului precedent este eliminarea ecuatiilor “duplicat”, si anume a acelora care au acelasi c_i . Pentru 2 ecuatii cu acelasi c_i , este necesar ca ele sa aiba si acelasi r_i pentru a avea o solutie(lucru garantat de cerinta problemei). Astfel vom avea doar maxim $\sqrt{n}$ ecuatii distincte deoarece $\Sigma c_i = n$.

Solutie O(n * n) - 60 puncte(10 + 50):

Odata gasit sistemul de n ecuatii, putem sa il rezolvam cu un algoritm similar cu cel al teoremei chineze a resturilor(dar putin simplificat pentru a evita unele lucruri matematice).

Cititorul familiarizat poate aplica si direct teorema chineza a resturilor.

Solutie **PermPower**
Stud. Sebastian Nechita - **Universitatea Babeş-Bolyai**

Astfel ne tinem minte 2 numere a si b, a. i. k e de forma a + x * b(x natural >= 0) dupa ce am parcurs primele i ecuatii(k satisface primele i ecuatii si este de acea forma pentru a le putea satisface).

Initial k este clar de forma 1 + x * 1.

Daca la un moment dat, k este de forma a + x * b, si urmeaza sa procesam ecuatia k % c_i = r_i, putem observa urmatoarele(pentru a-l gasi pe k de forma a' + x * b' care satisface atat ecuatiile precedente cat si noua ecuatie):

1. a' = a + q * b, pentru un anumit q.
2. (k - a) divizibil cu b ⇒ (k - a - q * b) divizibil cu b ⇒ (k - a') divizibil cu b ⇒ (x * b') divizibil cu b pentru orice x natural deci si pentru x = 1 ⇒ b' divizibil cu b.
3. a' % c_i = r_i, (a'+ b') % c_i = r_i … -> deci b' este multiplu de c_i.
4. b' este atat divizibil cu b cat si cu c_i, conditii suficiente pentru a garanta corectitudinea lui b'. b'-ul minim care satisface cele 2 conditii este b'=cmmmc(b, c_i)

Tot ce mai ramane de facut este sa il cautam pe q minim din ecuatia a' = a + q * b, a. i. a' % c_i = r_i ⇒ (q * b) % c_i = $r_i - a$. Daca exista un q minim, acesta va fi < c_i (altfel am putea scadea c_i din el pentru a obtine un q mai mic care sa satisfaca ecuatia).

Deoarece stim ca $\Sigma c_i = n$, putem pur si simplu sa incercam pe rand fiecare q fara a pune in pericol complexitatea O(n) pentru rezolvarea sistemului.

Solutie O(n) - 100 puncte:

Odata ce stim sa rezolvam sistemul in O(n), mai este necesar doar sa creem sistemul de ecuatii in complexitate O(n).

Pentru a afla r_i pentru fiecare i intr-o complexitate mai buna, putem sa descompunem permutarea in cicluri si sa tratam fiecare ciclu separat. Pornind cu o pozitie din ciclu(i), vom gasi in O(c_i) care este r_i pentru care p^(r_i)(i) = q(i) (adica cel mai mic numar pentru care elementul i ajunge pe pozitia corecta). Dupa ce stim r_i, facem observatia ca pentru toate elementele j din acest ciclu ar trebui sa avem r_i = r_j. Elementele din ciclul lui i sunt {i, p(i), p^2(i)..., p^(c_i -1)(i)}. Putem trece in O(1) de la verificarea lui i la verificarea lui p(i).

Stim ca p^(r_i)(i) = q(i). Verificam ca p^(r_i)(p(i)) = q(p(i)) <==> p(p^(r_i)(i)) = q(p(i)).

Fie astfel a = p^(r_i)(i) si b = i. Trebuie sa verificam ca a = q(b). Pentru a trece la urmatorul numar din ciclu setam a = p(a), b = p(b). Facem acest lucru de c_i ori (complexitate O(c_i)).

Procedam la fel pentru fiecare ciclu si obtinem o complexitate O(suma(lungimi ciclurilor)) = O(n)

Deoarece aceste verificari nu sunt complet necesare (ar fi necesare doar in cazul in care e posibil sa nu avem solutie pentru problema noastra), putem sa calculam r_i

doar pentru un element din fiecare ciclu si sa presupunem ca si pentru celelalte elemente este la fel.