**SOLUTIE CABANE**

Solutia este programare dinamica.

Permutarile se construiesc (se numara) prin inserarea rand pe rand a cate unui element incepand cu elementul 1.

Ca in orice dinamica de numarare este importanta construirea unor grupuri de permutari cu aceleasi proprietati.

In cazul nostru ne intereseaza unde se afla ultimile K elemente inserate.

Acest tip de rationament ne duce la complexitate de N^(K+1) cu memorie N^K, putin cam mult...

De fapt nu ne intereseaza pozitiile ultimilor K inserate ci modul in care sunt ele dispuse.

Notam cu K-1 ultimul element inserat, K-2 penultimul, K-3 antepenumultimul si tot asa.

Notam in continuare cu 0 elementele mai mici decat elementul ce trebuie inserat cu cel putin K.

De exemplu sa presupunem K = 2, permutarea 3 1 5 2 4 si acum trebuie sa inseram 6.

Permutarea se codifica prin 0 0 2 0 1. Mai mult decat atat grupurile de 0 se compacteaza intr-un singur 0, deci codificarea permutarii este 0 2 0 1.

In total numarul de codificari (stari) este K ! * 2 ^ (K+ 1), adica pentru orice permutare de ordin K exista K+1 pozitii in care pot exista sau nu 0-uri.

De exemplu pentru K = 2 sunt 16 codificari (numeratate de la 0 la 15) :

0. 1 2
1. 1 2 0
2. 1 0 2
3. 1 0 2 0
4. 0 1 2
5. 0 1 2 0
6. 0 1 0 2
7. 0 1 0 2 0
8. 2 1
9. 2 1 0
10. 2 0 1
11. 2 0 1 0
12. 0 2 1
13. 0 2 1 0
14. 0 2 0 1
15. 0 2 0 1 0

O astfel de codificare se transforma in alta prin inserarea unui element si degradarea celor existe. De exemplu in codificarea 1 inseram pe 3 la mijloc si degradam (scadem 1 din fiecare element, mai putin din 0-uri) in mod corespunzator :

1 2 0 =>(inserare) 1 3 2 0 =>(degradare) 0 2 1 0 (codificarea 13)

Deci 13 este o stare urmatoare lui 3.

Evident langa un 0 nu putem insera pentru ca acel element este prea mic.

Deci fiecare stare are maxim k+1 stari urmatoare (locuri in care se poate insera ultimul element).

Formam acest graf al starilor (codificarilor) dupa care aplicam dinamica :

A[i][j] = cate permutari de i numere avand codificarea j exista

Algoritmul are o complexitate de **O(N * (K !) * (2 ^ (K+1)) * K + (K !) * (2^(K+1)) * (K^2))**, adica dinamica + constructia grafului.