Descrierea soluţiei Munte

Trebuie să aranjăm cele *n* elemente ale unui şir în toate modurile posibile în formă de munte.

Prima dată să rezolvăm cazurile particulare:

1. **dacă avem două sau mai multe valori maximale**, problema are 0 soluţii, fiindcă vârful trebuie să fie unic.
2. **dacă un element apare de mai mult de două ori în şir**, atunci: „o ladă la stânga, o ladă la dreapta" şi a treia ladă deja va acoperi inevitabil una dintre cele două lăzi. Deci problema iarăşi nu are soluţie.

Mai sunt încă două cazuri:

3. **Toate elementele sunt distincte.** Pentru că vârful muntelui trebuie să fie pe una dintre poziţiile 2,3,…,*n*-1 , şi aceste mulţimi de soluţii sunt disjuncte două câte două. Rezultatul final va fi suma soluţiilor parţiale mai sus amintite.

Dacă vârful se află pe poziţia i, vom avea în total C_{n-1}^{i} cazuri distincte (restul de *n*-1 elemente ne stau la dispoziţie, fiecare combinare se poate lua o singură dată în ordine strict crescătoare, celelalte elemente vor fi unic determinate în dreapta vârfului).

Deci avem în total $\boxed{C_{n-1}^{1}+C_{n-1}^{2}+\ldots+C_{n-1}^{n-2}=2^{n-1}-2}$ (formulă învăţată la combinatorică unde valoarea -2 este echivalentul termenilor lipsă C_{n-1}^{0} şi C_{n-1}^{n-1}

4. **Avem perechi de elementele egale.** În acest caz anumite elemente sunt forţate să fie atât la stânga cât şi la dreapta, deci nu vor intra în calculul formulei. Însă intervine o schimbare majoră: vârful nu mai poate ajunge la extremităţi.

Deci dacă **numărul elementelor distincte este** *p*, atunci rezultatul va fi:
$\boxed{C_{p-1}^{0}+C_{p-1}^{1}+\ldots+C_{p-1}^{p-1}=2^{p-1}}$.

Algoritmul necesită o sortare, **sortarea prin numărare în timp liniar** este ideală pentru această cerinţă.

Calculul valorii 2^p se poate rezolva în timp logaritmic sau în timp liniar, dar acest calcul în programul nostru nu influenţează timpul de execuţie.