Soluții pentru problema *Fuzetea*

Lotul Național de Informatică, Craiova 2025

Subtask 1 ($1 \leq N \leq 7$)

O soluție directă folosește *backtracking* cu simulare completă. Pentru fiecare intrare (oltean) încercăm toate locurile libere disponibile conform regulilor date și numărăm configurațiile care conduc la plasarea tuturor celor N olteni. Complexitatea este $\mathcal{O}(N!)$, dar pentru $N \leq 7$ este suficient de rapid.

Subtask 2 ($1 \leq N \leq 20$)

Generalizăm backtracking-ul folosind *programare dinamică cu memorare* pe stări. Reprezentăm configurația curentă printr-o mască de biți de lungime N, unde un bit indică dacă locul este ocupat. Starea DP este dată de mască și de pasul curent i, iar tranziția încearcă pozițiile permise pentru al i-lea sosit, aplicând regulile introvertiților/extrovertiților. Numărul stărilor este $\mathcal{O}(N2^N)$, iar fiecare tranziție în $\mathcal{O}(N)$, deci în total $\mathcal{O}(N^2 2^N)$. În practică, numărul de stări relevante și construibile pentru un șir de introverți / extroverți dat este semnificativ mai mic.

Subtask 3 (toți oltenii sunt extrovertiți)

Dacă toți cei N olteni sunt extrovertiți, observăm că în momentul în care intră al k-lea, el vrea să stea lângă un grup continuu de oameni deja așezați. Astfel:

- Primul extrovertit poate alege oricare dintre cele N locuri.
- Fiecare dintre următorii $N-2$ extrovertiți se poate poziționa fie în stânga, fie în dreapta grupului deja format (întotdeauna există două capete libere‘
- Excepție o face ultimul oltean, care are un singur loc unde se poate așeza.

Prin urmare, numărul total de moduri este:

$$N \times 2^{N-2} \pmod{10^9+7}.$$

Subtask 4 (toți oltenii sunt introvertiți)

Când toți N oltenii sunt introvertiți, fiecare încearcă să se așeze cu cât mai mulți vecini liberi.

Observații:

- La început, sunt destul de multe scaune încât fiecare oltean se poate așeza fără să aibă vreun vecin. Considerăm că, pentru o anumită configurație, primele k persoane s-au așezat fără vecini, iar a $k+1$-a persoană este nevoită să aibă cel puțin un vecin.
- Fiecare configurație validă poate fi descrisă prin succesiunea de *găuri* (secvențe de scaune libere) dintre k persoane deja așezate.
- Numărul de astfel de configurații este dat de răspunsul combinatoric la întrebarea "Dacă avem $N-k$ locuri libere și k seturi de mărime 1 sau 2, în cate moduri putem împărți locurile în aceste seturi?", care se poate rezolva prin combinări / "stars and bars".
- Notăm cu a numărul găurilor de lungime 1 și cu b numărul găurilor de lungime 2 astfel încât

$$a + 2b = N - k, a + b = k$$

- Mai întâi, se umplu găurile de lungime 2, în orice ordine ($b!$) iar pentru fiecare alegem dacă mai întâi punem o persoana pe prima sau pe a doua poziție din gaură (2^b). Pentru fiecare gaură de lungime 2, rămâne o gaură de lungime 1.
- Apoi, găurile de lungime 1 (în total acum $a+b$) se umplu în orice ordine ($(a+b)!$).
- În total, numărul de moduri de a umple găurile este:

$$2^b \times b! \times (a+b)!.$$

- Prin parcurgerea tuturor posibilităților pentru k, a, b și adunarea rezultatelor, putem obține soluția în $\mathcal{O}(N^2)$ timp, sau chiar $\mathcal{O}(N)$ în urma unor observații matematice.

Subtask 5 și 6 $N \leq 100$ și $N \leq 250$

Există multiple soluții folosind dinamici în 3 dimensiuni, dintre care funcționează echivalent cele ale căror stări rețin destulă informație pentru a deduce:

- numărul de persoane deja aflate la masă

- numărul de *găuri* totale (așa cum au fost definite în Subtask 4)
- numărul de *găuri* de lungime exact 1.

1. Reducerea mesei circulare la un segment

Deoarece rotațiile mesei circulare sunt considerate distincte, putem să:

1. Fixăm primul oltean într-unul din cele N scaune (în N moduri).
2. Rezolvăm apoi plasarea celorlalți $N-1$ olteni pe un șir liniar de $N-1$ scaune, al cărui capăt stâng și capăt drept se învecinează ambele cu primul oltean.

2. Găurile de scaune

După ce unii olteni s-au așezat, scaunele rămase neocupate se împart în mai multe *găuri* (segmente consecutive de scaune libere). Pentru o gaură de lungime L avem:

- Dacă $L \geq 3$: două scaune „la margine" (fiecare cu un vecin ocupat) și $L-2$ scaune „interioare" (fără vecini ocupați).
- Dacă $L = 2$: două scaune "la margine", ambele cu un vecin ocupat.
- Dacă $L = 1$: un singur scaun, cu doi vecini ocupați.

3. Programare dinamică în două faze

Procesăm sosirile în ordine și menținem două faze:

Faza 1 (toate găurile au lungime ≥ 3). Definim:

$$U = \text{numărul de găuri de lungime } \geq 3$$

$$A = \text{numărul de găuri de lungime } \geq 2$$

Cât timp fiecare introvertit găsește un scaun fără vecini sau fiecare extrovertit găsește unul cu doi vecini, continuăm în Faza 1. Așezarea într-o gaură de lungime ≥ 3 împarte acea gaură în două, actualizând U și A.

Punctul de trecere spre Faza 2. În momentul în care cineva nu mai poate lua locul ideal (nu există găuri suficient de mari), toate găurile rămase sunt de lungime 1 sau 2. Calculăm:

$$g = U + A, \quad r = \text{numărul de scaune rămase}, \quad h_2 = r - g, \quad h_1 = g - h_2.$$

Înmulțim cu $\binom{U}{h_2 - A}$ pentru a alege care dintre găurile mari au devenit găuri de lungime 2, și ne folosim de rezultatele din Faza 2.

Faza 2 (doar găuri de lungime 1 sau 2). Definim:

$$h_1 = \#\text{ găuri de lungime 1}, \quad h_2 = \#\text{ găuri de lungime 2}.$$

- **Introvertit:**
  - Dacă există $h_2 > 0$, se așază la unul dintre cele $2h_2$ scaune cu un vecin, transformând o gaură de 2 în una de 1.
  - Altfel, ocupă unul dintre cele h_1 scaune cu doi vecini, eliminând o gaură de 1.
- **Extrovertit:**
  - Dacă există $h_1 > 0$, preferă unul dintre cele h_1 scaune cu doi vecini, eliminând o gaură de 1.
  - Altfel, ocupă unul dintre cele $2h_2$ scaune cu un vecin, transformând o gaură de 2 în una de 1.

Din această definiție recursivă putem obține o formula pentru

`DP[`h_1`][`h_2`]` = numărul de moduri de a umple h_1 găuri de lungime 1 și h_2 găuri de lungime 2

4. Complexitate și modulo

Ambele faze se pot implementa folosind programare dinamică / memoizare, iar numărul de stări totale este $\mathcal{O}(N^3)$. Fiecare tranziție se realizează în timp constant, deci soluția folosește $\mathcal{O}(N^3)$ timp și $\mathcal{O}(N^3)$ memorie. De asemenea, pentru a folosi doar $\mathcal{O}(N^2)$ memorie observăm că Faza 2 are doar N^2 stări iar în Faza 1 putem sorta stările după numărul de persoane așezate la masă k, observând ca acesta crește doar cu câte 1. Astfel, putem aplica optimizarea în care reținem doar 2 matrici, pentru k si $k+1$.