Trucuri de combinatorică

Alexandru Luchianov

Septembrie 2022

1 Introducere

Unul din lucrurile care face combinatorica una din cele mai frumoase ramuri ale matematicii este faptul că soluția unei probleme poate fi exprimată, de cele mai multe ori, folosind un limbaj natural și un raționament ușor de înțeles. Unele persoane, însă, tratează combinatorica într-un mod mecanic si abordează lucrurile dintr-o perspectivă pur algebrică, ajungând să memoreze formule fără să înțeleagă raționamentele naturale din spatele acestora.

2 Demonstrarea unor identități populare

2.1 Câteva identități simple

Voi prezenta în continuare câteva formule arhicunoscute și usor de demonstrat. Este recomandată citirea identităților de mai jos și încercarea demonstrării acestora pe cont propriu înainte de a continua.

1. $\binom{n}{k} = \binom{n}{n-k}$
2. $\binom{n-1}{k-1} + \binom{n-1}{k} = \binom{n}{k}$
3. $\binom{n}{k} k = n\binom{n-1}{k-1}$

Acestea se pot demonstra *ușor* într-o manieră pur algebrică folosind formula combinărilor și distribuind diferit termenii, dar ca exercițiu vom încerca să evităm acest lucru și să le demonstrăm doar prin reformularea obiectului numărat.

1. Alegerea a k elemente dintr-un set de n elemente este echivalentă cu selectarea celor $n-k$ elemente pe care nu le vom alege.
2. Pentru a alege un set de k din n elemente putem începe prin a fixa apartenența la set a primului element. Dacă alegem ca acesta să nu aparțină setului, atunci mai avem de ales k elemente din cele $n-1$ rămase, altfel mai avem de ales doar $k-1$ elemente pentru a completa setul.

3. Putem reformula ambele părți drept: "în câte moduri putem alege un set de k elemente, din care să alegem un element drept reprezentant". Prima parte poate fi interpretată drept a alege mai întâi setul, apoi reprezentantul. A doua parte poate fi interpretată drept a alege mai întâi reprezentantul din cele n elemente, apoi restul setului. Aceste două interpretări duc la același rezultat, chiar dacă au formule diferite.

Identitățile de mai sus sunt doar niște exemple elementare ale ideii de numărare în moduri diferite. Această metodă poate fi generalizată mai departe pentru a demonstra identități mai avansate:

1. $\sum_{i=0}^{n} \binom{n}{i} = 2^n$
2. $\sum_{i=0}^{n} \binom{n}{i}^2 = \binom{2n}{n}$
3. $\sum_{i=r}^{n} \binom{n}{i} = \binom{n+1}{r+1}$ (Teorema crosei de hockey)

Și demonstrațiile acestora:

1. E ușor de văzut faptul că suma aceasta numără de fapt pe rând submulțimile cu i elemente pentru a obține numărul total al submulțimilor. Numărul total al submulțimilor este 2^n deoarece fiecare element poate să aparțină sau nu submulțimii alese.

2. Putem interpreta aceasta identitate drept numărul de metode de a selecta un set de n dintr-un total de $2n$ elemente. Împărțim cele $2n$ elemente în 2 grupe de câte n, apoi alegem i elemente din prima grupă pe care să le includem în set și i elemente din a doua grupă pe care să *nu* le includem. În total, obținem n elemente selectate.

3. Pentru a alege o submulțime de $r+1$ elemente din $n+1$ putem acorda elementelor id-uri de la 1 la $n+1$ și să fixăm id-ul maxim al elementelor din set. Fie acest maxim i, astfel mai avem de ales doar r elemente din cele $i-1$ cu id mai mic decât maximul nostru fixat.

Chiar dacă aceste identități în sine nu o să ajute decât foarte rar în probleme, tehnica de a reordona și reformula modul de a număra lucruri este etern folositoare. Un exemplu de problemă în care este necesară o astfel de gândire "out of the box" este următoarea:

2.2 BBQ Hard

Enunț: Snuke are N pachete de frigărui ($1 \leq N \leq 2 \cdot 10^5$). Al i-lea pachet de frigărui conține un băț de frigăruie, A_i bucăți de carne și B_i bucăți de ardei ($1 \leq A_i, B_i \leq 2 \cdot 10^3$). Bețele de frigărui nu se pot distinge unele de altele. Nici bucățile de carne nu se pot distinge unele de altele și nici bucățile de ardei. Snuke alege 2 pachete arbitrare (pachetele rămase nu vor fi utilizate), scoate toate bucățile de carne sau ardei de pe cele 2 bețe de frigărui, permută arbitrar

bucățile respective, iar apoi înfige ambele bețe prin ele pentru a forma o frigăruie completă.

În câte feluri poate forma Snuke o frigăruie completă modulo $10^9 + 7$? Două frigărui complete se consideră distincte dacă setul pachetelor de frigărui utilizate este distinct sau ordinea bucăților de mâncare este diferită.

Problema originală se poate găsi la linkul: `atcoder.jp/contests/agc001/tasks/agc001_e`.

Soluție: Dacă alegem să combinăm pachetele i și j de frigărui, atunci vom obține $A_i + A_j$ bucăți de carne și $B_i + B_j$ bucăți de ardei. Aceste bucăți pot fi rearanjate în $f(A_i + A_j, B_i + B_j) = \binom{A_i+A_j+B_i+B_j}{B_i+B_j}$ moduri.

Observația de mai sus ne permite să găsim o soluție foarte ușoară, însă mult prea lentă în $\mathcal{O}(N^2)$. Mai există o soluție alternativă bazată pe FFT care obține complexitatea de $\mathcal{O}(max_A \cdot max_B \cdot log)$ unde max_A reprezintă maximul dintre A_i, iar B reprezintă maximul dintre B_i, însă nici aceasta nu este suficient de rapidă.

Trebuie cumva să procesăm multiple perechi în același timp, însă formula de combinare a pachetelor este extrem de complicată. Un lucru pe care îl putem face e să reformulăm definiția lui $f(A_i + A_j, B_i + B_j)$ folosind o interpretare combinatorială. Putem interpreta $f(A_i+A_j, B_i+B_j)$ drept numărul de drumuri de la $(0, 0)$ la $(A_i + A_j, B_i + B_j)$ pe o matrice mergând doar în sus și în dreapta.

Este necesar să considerăm fiecare pereche (A_i+B_i, A_j+B_j) în mod explicit, așa că dorim să găsim o metodă de a separa pe i și j. O observație care ne permite să avansăm este faptul că drumul de la $(0, 0)$ la $(A_i + B_i, A_j + B_j)$ este echivalent cu drumul de la $(-A_i, -B_i)$ la (A_j, B_j).

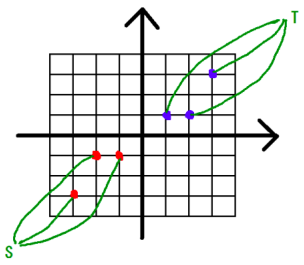

Figure 1: Ilustrație seturile S și T

Pentru fiecare pachet de frigărui (A_i, B_i), fie $(-A_i, -B_i)$ parte din setul S și (A_i, B_i) parte din setul T. Răspunsul total este egal cu numărul de drumuri de la un element din S la un element din T. Putem număra acest lucru folosind programare dinamică. Metoda exactă de a folosi programarea dinamică este lăsată ca exercițiu pentru cititor.

3 Numerele Catalan

3.1 Introducere cu privire la numerele Catalan

Aceste numere sunt des întâlnite sub diverse forme precum numărarea parantezărilor corecte de o anumită lungime, numărarea arborilor binari sau chiar numărul de triangulări al unui poligon convex. Pentru a demonstra faptul că o problemă este echivalentă cu "numerele Catalan", este necesar doar să ajungem la aceeași recurență.

Să luăm drept exemplu numărul de parantezări corecte. Notăm numărul de parantezări corecte care constau în n paranteze deschise și n paranteze închise drept $C(n)$.

O parantezare corectă ori este un șir "null", ori respectă următoarea formă: $'('+P1+')'+P2$ unde $P1$ și $P2$ reprezintă parantezări. Acest lucru ne permite să descoperim următoarea recurență dacă fixăm câte perechi de paranteze conține $P1$ și implicit câte perechi conține $P2$:

$$C(n) = \sum_{i=0}^{n-1} C(i)C(n-1-i) \tag{1}$$

Recurența de mai sus este exact recurența numerelor Catalan, însă este relativ ineficientă. Am putea folosi o formulă mai eficientă deja descoperită, însă, în cazul nostru, demonstrația din spate este mai valoroasă decât formula în sine. Pentru a obține o metodă mai eficientă este necesară o schimbare de perspectivă. Putem asocia unei parantezări un grafic construit prin transformarea fiecărei paranteze '(' în '/' și ')' în '\'.

Condițiile de corectitudine ale unui șir de paranteze pot fi interpretate în mod grafic:

- Numărul de paranteze '(' și paranteze ')' este egal $\iff$ graficul începe și se termina la aceeași înălțime.
- Pentru orice prefix din parantezare, numărul de paranteze '(' este mereu mai mare sau egal decât numărul de paranteze ')' $\iff$ graficul nu coboară niciodată sub înălțimea inițială.

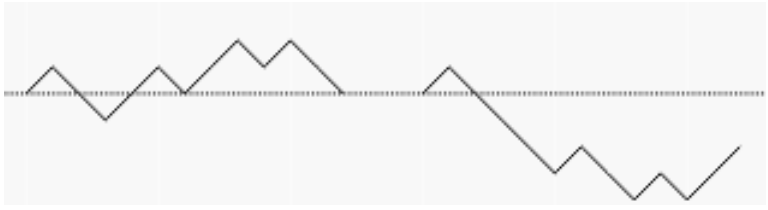

Figure 2: Graficul parantezării incorecte "())(()(()())" și cel al corespondentei sale

Parantezările care respectă prima condiție sunt ușor de numărat deoarece este necesar doar să permutăm parantezele. Din cele $\binom{2N}{N}$ care respectă prima condiție trebuie să le eliminăm pe cele care taie linia inferioară a graficului. Putem asocia fiecărei parantezări care respectă prima condiție, dar taie graficul, un corespondent unic mai ușor de numărat. Căutăm primul segment care coboară sub grafic și schimbăm direcția tuturor segmentelor care urmează, astfel obținem un grafic unic care conține $n-1$ segmente '/' și $n+1$ segmente '\'. Numărul de astfel de grafice este $\binom{2N}{N+1}$.

Astfel obținem cunoscuta formulă finală: $C(N) = \binom{2N}{N} - \binom{2N}{N+1}$, însă adevăratul câștig este învățarea metodei de rezolvare. Această metodă poate fi generalizată în diverse feluri. Putem permite graficului să se termine la alte înălțimi sau putem schimba limita inferioară a graficului (Generalizarea acestei probleme este lasata drept exercitiu). Un exemplu de aplicație este chiar următoarea problemă.

3.2 Ball Eat Chameleons

Enunț: Ringo are N cameleoni într-o cușcă ($1 \leq N \leq 5 \cdot 10^5$). Un cameleon care nu a mâncat nimic este albastru. Un cameleon își schimbă culoarea după următoarele reguli:

- Un cameleon își va schimba culoarea în albastru atunci când a mâncat mai multe bile albastre decât bile roșii.
- Un cameleon își va schimba culoarea în roșu atunci când a mâncat mai multe bile roșii decât albastre.

Inițial, niciun cameleon nu a mâncat nimic. Ringo i-a hrănit repetând următorul proces de K ori ($1 \leq K \leq 5 \cdot 10^5$) :

- Apucă o bila roșie sau albastră;
- Aruncă bila selectată în cușca cameleonilor. Apoi, unul din cameleoni o mănâncă.

După ce Ringo a aruncat K bile în cușcă, toți cameleonii erau roșii. În câte moduri modulo 998244353 ar fi putut Ringo să aleagă culorile bilelor aruncate? Două metode sunt considerate diferite dacă există un i astfel încât culorile bilelor aruncate la a i- a aruncare sunt diferite.

Problema originală se poate găsi la `atcoder.jp/contests/agc021/tasks/agc021_e` .

Soluție: În primul rând este necesar să analizăm strategia optimă a lui Snuke.

Dacă pentru fiecare cameleon bilele roșii se află în majoritate înseamnă că avem o soluție. Însă, un cameleon poate să fie roșu fără ca bilele roșii să se afle în majoritate. Cu alte cuvinte, atunci când un cameleon devine roșu, noi practic câștigăm un *"loc gratuit"* unde putem arunca o bilă albastră. Această observație ne duce la următoarea strategie:

- Înainte de începerea procesului alegem să izolăm un cameleon.
- Dacă bila curentă este albastră, alegem să prioritizăm hrănirea cameleonilor roșii cărora nu li se va schimba culoarea în albastru după mâncarea bilei, altfel hrănim cameleonul izolat la început.
- Dacă bila curentă este roșie, atunci prioritizăm cameleonii albaștri neizolați. Dacă toți cameleonii neizolați sunt roșii, atunci putem hrăni cameleonul izolat.

În urma unei analize a strategiei optime, putem observa importanța acestor "locuri gratuite" menționate anterior. Prin acest "loc gratuit", fiecare cameleon ne permite să cuplăm o bilă roșie cu una albastră, iar apoi să uităm de ele. Acest lucru este însă dependent de ordinea bilelor, fiind posibil să rămânem cu bile albastre care nu pot fi cuplate cu o bilă roșie aruncată anterior.

Acest număr de bile albastre fără pereche este egal cu diferența maximă dintre numerele de bile albastre și de bile roșii, întâlnită pe parcursul procesului. Fie acest număr notat cu P. Putem observa faptul că pentru a determina dacă o configurație este sau nu validă este suficient să cunoaștem variabilele R (numărul de bile roșii), B (numărul de bile albastre) și P.

Acest proces poate fi reprezentat sub forma de grafic într-un mod similar cu exemplul anterior al parantezărilor. Asociem bilelor roșii '/' și bilelor albastre '\'. Tot în acest grafic, putem observa că P reprezintă adâncimea celui mai de jos punct.

Considerăm următoarele cazuri:

1. Fie $R < B$: Clar imposibil ca toti cameleonii să ajungă roșii.
2. Fie $R = B$: Este clar faptul că ultima bilă trebuie să fie albastră. Cameleonul izolat va ajunge să dețină $R - (N - 1)$ bile roșii. Acest lucru implică faptul că putem lăsa doar $R - (N - 1)$ bile albastre fără pereche și să le trimitem la cameleonul izolat $\iff P \leq R - (N - 1)$.
3. Fie $R > B$: Aproape la fel ca cazul anterior, cu excepția faptului că nu mai este necesar ca ultima bilă să fie albastră. În cazul în care este albastră atunci putem să îl tratăm exact la fel. Altfel, știm că ultima bilă mereu ajunge la cameleonul izolat. Pentru ca acest cameleon să devină roșu este necesar ca bilele roșii să dețină majoritatea în el sau să o câștige cu această bilă. Ca o consecință a acestui lucru, condiția noastră asupra lui P devine puțin mai strictă: $P < R - (N - 1)$.

Pentru a număra graficele care respectă condiția noastră, putem folosi generalizările prezentate la finalul secțiunii anterioare.

4 Tehnica produsului

Să considerăm următoarea problemă clasică:

Enunț: *Fie N obiecte. Obiectul i are greutate w_i. Câte seturi de obiecte cu suma greutăților X există?*

Soluție: Problema de mai sus este o aplicație directă a programării dinamice. Definim $dp_{i,j}$ drept numărul de seturi formate din primele i obiecte cu suma greutăților egală cu j. Apoi, recurența este elementară:

$$dp_{i,j} = dp_{i-1,j} + dp_{i-1,j-w_i}$$

Putem încerca să facem problema puțin mai complicată:

Varianta grea: *Fie N obiecte. Obiectul i are greutate v_i. Care este suma dimensiunilor seturilor cu suma greutăților X ?*

Soluție: Dificultatea în această problemă constă în faptul că dimensiunea setului este coeficientul său. Dorim să reformulăm problema astfel încât toate soluțiile să aibă același coeficient. În acest scop, putem număra în câte moduri alegem un set, iar apoi alegem un reprezentant din el.

Este ușor de văzut faptul că aceste două probleme sunt echivalente. Însă acum este mult mai ușor de formulat o soluție folosind programare dinamică. Definim $dp_{i,j,h}$ drept numărul de moduri de a alege un set cu suma greutăților j, format din primele i elemente din care am ales h reprezentanți. Recurența este similară cu cea anterioară:

$$dp_{i,j,0} = dp_{i-1,j,0} + dp_{i-1,j-w_i,0}$$

$$dp_{i,j,1} = dp_{i-1,j,1} + dp_{i-1,j-w_i,0} + dp_{i-1,j-w_i,1}$$

Problema încă poate fi generalizată mai departe:

Generalizare: *Fie N obiecte. Obiectul i are greutatea v_i Definim **valoarea** unui set drept dimensiunea sa ridicată la puterea K. Care este suma valorilor seturilor cu suma greutăților X ?*

Se poate observa faptul că problema precedentă este cazul $K = 1$ al problemei curente. Putem aplica același truc, cu diferența că de data aceasta după alegerea setului fixăm un șir de K reprezentanți (nu neapărat distincți) . De exemplu, dacă am avea de ales 2 reprezentanți dintr-un set de 3 elemente am putea alege oricare dintre următoarele variante: {1, 1}, {1, 2}, {1, 3}, {2, 1}, {2, 2}, {2, 3}, {3, 1}, {3, 2}, {3, 3}.

Definiția dinamicii rămane aproape la fel, însă recurența trebuie schimbată puțin pentru a generaliza și ultima dimensiune:

$$dp_{i,j,h} = dp_{i-1,j,h} + \sum_{h_2=0}^{h} dp_{i-1,j-w_i,h_2} \cdot \binom{h}{h_2}$$

Raționamentul din spatele formulei este că, la fiecare pas, putem alege să includem elementul curent sau nu. În cazul în care îl includem, putem alege de câte ori să îl includem și pe ce poziții ale șirului de reprezentanți să fie inclus elementul nou.

Această tehnică se poate regăsi și în următoarele probleme:

- `infoarena.ro/problema/arbsumpow`
- `kilonova.ro/problems/246` (Tennis, Info1Cup 2022)

5 Two Snuke

Multe trucuri de combinatorică nu constituie principala idee a unei probleme, ci doar mici pași care conduc către rezolvare. Un exemplu de problemă care combină multiple astfel de trucuri este următoarea:

Enunț: Se dă un număr întreg N ($1 \leq N \leq 10^9$). Snuke va alege numerele întregi s_1, s_2, n_1, n_2, u_1, u_2, k_1, k_2, e_1, e_2, astfel încât următoarele 6 condiții să fie satisfăcute:

- $0 \leq s_1 < s_2$
- $0 \leq n_1 < n_2$
- $0 \leq u_1 < u_2$
- $0 \leq k_1 < k_2$
- $0 \leq e_1 < e_2$
- $s_1 + s_2 + n_1 + n_2 + u_1 + u_2 + k_1 + k_2 + e_1 + e_2 \leq N$

Pentru fiecare alegere posibilă $(s_1, s_2, n_1, n_2, u_1, u_2, k_1, k_2, e_1, e_2)$, calculează valoarea $(s_2-s_1)(n_2-n_1)(u_2-u1)(k_2-k_1)(e_2-e_1)$ și găsește suma acestor produse modulo $10^9 + 7$.

Problema originală se poate găsi la linkul: `atcoder.jp/contests/aising2020/tasks/aising2020_f`.

Soluție: În această formă originală problema poate părea intimidantă, însă aceasta poate fi redusă prin multiple trucuri.

Pentru calcularea valorii fiecărei soluții vom introduce variabila s_3 care reprezintă $s_2 - s_1$ și în mod analog variabilele n_3,u_3,k_3,e_3. Astfel contribuția unei soluții devine $s_3n_3u_3k_3e_3$ și ultima condiție poate fi rescrisă astfel:

$$2(s_1 + n_1 + u_1 + k_1 + e_1) + s_3 + n_3 + u_3 + k_3 + e_3 \leq N$$

Un lucru care face rezolvarea acestei probleme dificilă este faptul că fiecare soluție are o contribuție diferită ($s_3n_3u_3k_3e_3$). Putem evita aceasta problemă

prin introducerea unor variabile auxiliare care să simuleze această contribuție. În acest scop, definim s_4 cu condiția ca $0 \leq s_4 < s_3$ și analog variabilele n_4,u_4,k_4,e_4. Aceste variabile ne permit doar să numărăm câte soluții satisfac un set de ecuații, în loc de a însuma coeficienții unor soluții.

Următorul pas este de a aduce limita inferioară a tuturor variabilelor la 0 pentru simplitate. În acest scop înlocuim în ecuații variabila s_3 cu variabila s_5, astfel încât $s_4 + 1 + s_5 = s_3$. Astfel, putem rescrie foarte frumos ecuațiile în felul următor:

$$0 \leq s_1, n_1, u_1, k_1, e_1, s_4, n_4, u_4, k_4, e_4, s_5, n_5, u_5, k_5, e_5$$

$$2(s_1+n_1+u_1+k_1+e_1)+s_4+s_5+1+n_4+n_5+1+u_4+u_5+1+k_4+k_5+1+e_4+e_5+1 \leq N$$

De aici problema este aproape rezolvată, singura dificultate este faptul că în ultima ecuație variabilele au coeficienți diferiți. Un lucru pe care l-am putea face dacă toate variabilele $s_4, n_4, u_4, k_4, e_4, s_5, n_5, u_5, k_5, e_5$ ar fi pare, ar fi să dăm factor comun forțat pe 2. Pentru a rezolva această problemă fixăm paritatea fiecărei variabile și dăm factor comun pe 2 oricum.

Astfel, am ajuns la o formă a problemei clasice:

$$0 \leq v_1, v_2, v_3 \dots v_K$$

$$0 \leq v_1 + v_2 + v_3 + \dots v_K \leq N$$

Pasul final este cel de a introduce variabila de egalizare *slack* cu scopul de a transforma a doua condiție într-o relație de egalitate.

$$0 \leq v_1, v_2, v_3 \dots v_K, slack$$

$$v_1 + v_2 + v_3 + \dots v_K + slack = N$$

Astfel, am ajuns la forma standard a problemei combinatorice "Stars si bars" doar prin aplicarea succesivă a mici trucuri:

- Forțarea unor coeficienți prin introducerea de variabile noi;
- Reducerea limitelor inferioare ale variabilelor la 0;
- Forțarea unui factor comun prin selectarea parității;
- Introducerea unei variabile de egalizare pentru a forța o egalitate.