Prof. Marius Nicoli

Modalităţi de calcul pentru valoarea C_N^K.

Valoarea C_N^K reprezintă numărul de modalităţi de a construi submulţimi cu K elemente alese dintr-o mulţime cu N elemente. Nu este importantă ordinea elementelor într-o submulţime. De exemplu, pentru N = 5 şi K = 3, putem considera că mulţimea din care alegem este formată din elementele {1,2,3,4,5} şi submulţimile care se pot forma sunt {1,2,3}, {1,2,4}, {1,2,5}, {1,3,4}, {1,3,5}, {1,4,5}, {2,3,4}, {2,3,5}, {2,4,5}, {3,4,5}. Deci, $C_5^3 = 10.$ Observăm că, fără a fi piedut din soluţii, le-am scris pe toate cu elemente în ordine crescătoare. Aceasta este şi o strategie pe care o folosim în algoritmii care generează efectiv aceste submulţimi.
În acest articol nu ne propunem să generăm soluţiile ci să le numărăm.
Formula de calcul este următoarea:

$$C_N^K = \frac{N!}{K!\,(N-K)!}$$

Ea poate fi implementată folosind operaţii pe numere mari: înmulţire între un număr mare şi un număr mic, împărţire a două numere mari.

Iată şi câteva **recurenţe**:

$$C_N^K = \frac{N}{K} C_{N-1}^{K-1}$$

Recurenţă este liniară. Aşadar putem spune că avem complexitate O(N) pentru calcula această valoare. Practic, întrucât acest numere cresc foarte repede sunt necesare operaţii cu numere mari (în acest caz, înmulţiri şi împărţiri între un număr mare şi un număr mic).
Dacă avem de calculat valoarea combinărilor pentru mai multe perechi N, K putem folosi şi următoarea recurenţă

$$C_N^K = C_{N-1}^{K-1} + C_{N-1}^K$$

Aceasta este cea care stă la baza “triunghiului lui Pascal”. Considerând C ca fiind o matrice cu N linii şi K coloane, un element al său se construieşte însumând două elemente de pe linia anterioară, cel de deasupra şi cel aflat deasupra şi o poziţie mai în stânga.

	0	***1***	***2***	***3***	***4***	***5***
0	1					
1	1	1				
2	1	2	1			
3	1	3	3	1		
4	1	4	6	4	1	
5	1	5	10	10	5	1

Sunt o mulţime de relaţii între elementele triunghiului lui Pascal. Iată câteva:

- elementul de pe linia i şi coloana j reprezintă valoarea C_i^j
- elementele de pe linia N sunt coeficienţii din dezvoltarea binomului $(a+b)^N$
- suma elementelor de pe linia N este 2^N.

Folosind acest mod de calcul al combinărilor complexitatea în timp este de ordinul N^2. Este necesară doar operaţia de adunare a două numere mari.
De regulă nu este nevoie de o matrice de memorie ci de 2 vectori (elementele de pe o linie se calculează doar în funcţie de cele de pe linia anterioară). Memoria se poate reduce la un singur vector observând că putem calcula elementele de pe o linie de la dreapta la stânga, adunând la elementul curent elementul anterior.

Prof. Marius Nicoli

Calculul combinărilor “modulo P”

În multe situaţii, pentru evitarea lucrului cu numere mari, nu se cere valoarea finală ci doar restul împărţirii rezultatului la un număr dat P. Spunem că avem de calculat valoarea respectivă “modulo P”.
Aritmetica “modulo P” este relativ simplă dacă este sunt necesare operaţii de adunare şi înmulţire modulo P. Astfel avem:
(A mod P+ B mod P) mod P = (A + B) mod P
(A mod P* B mod P) mod P = (A * B) mod P
Nu este valabilă o relaţie similară şi pentru împărţire (la calculul combinărilor modulo P, aplicând formula, putem calcula modulo P factorialul de la numărător şi pe cele de la numitor dar apoi este nevoie şi de împărţire).
Înainte de a arăta câteva rezultate matematice care permit calcularea în mod optim a combinărilor, iată o modalitate alternativă ce solicită noţiuni elementare de algoritmică:
Deci, trebuie calculat C_N^K mod P. Cunoaştem că valoarea combinărilor este un număr întreg. Nu putem efectua calculele întrucât valoarea facorialelor creşte foarte rapid şi ar apărea depăşiri. Vom proceda în felul următor:
Realizăm **descpmpunerea în factori primi pentru N!**. Pentru aceasta ne folosim de următorul rezultat:
Numărul prim p apare în descomputerea în factori primi a lui N factorial de e ori unde
e = $\left[\frac{N}{p}\right] + \left[\frac{N}{p^2}\right] + \left[\frac{N}{p^3}\right] + \ldots$
Astfel, vom constui un vector P cu numerele primemai mici decât N. Construim apoi un vector E astfel: E[i] = puterea la care apare P[i] în descompunerea în factpri primi a lui N. Valorile lui E le calculăm aplicând rezultatul anterior pentru fiecare valoare din P.
De exemplu, pentru N = 10, obţinem cei 2 vectori asociaţi descompunerii în factori primi ai lui N! astfel:

P	2	3	5	7
E	8	4	2	1

Putem aplica procedeul anterior pentru K! şi (N-K)! dar valorile calculate le scădem din elementele lui E.
Pentru a obţine rezultatul rămâne doar să construim valoarea ce are descompunerea în factori primi memorată în elementele lui E. Sunt necesare doar operaţii de înmulţire modulară care se realizează uşor.
Costurile utilizării acestei metode sunt:

- Calculul numerelor prime mai mici decât N, complexitate O($Nlog_2N$).
- Calculul vectorului E, complexitate aproximativ O(N).
- Memorie necesară, de ordinul N

Metodele mai eficiente se bazează însă pe realizarea împărţirii modulare.
Se pune astfel următoarea problemă: Avem A şi B două valori, ambele modulo P (adică cuprinse între 0 şi P-1). Noi trebuie să calculăm valoarea (A/B) modulo P. Cunoaştem de la matematică faptul că o împărţire este o înmulţire cu inversul numitorului. În arirmetica modulo P inversul unei valori A “modulo P” se numeşte “invers modularul lui A modulo P”. Deci, pentru P dat, invers modularul unei valori A, modulo P, este o valoare notată A^{-1} aşa încât:
$A * A^{-1} = 1$ (modulo P)
Iată câteva exemple. Pentru P = 7

A	A^{-1}	Explicaţie
0	Nu există	Nu putem vorbi despre 1/0
1	1	(1 * 1) mod 7 = 1
2	4	(2 * 4) mod 7 = 8 mod 7 = 1
3	5	(3 * 5) mod 7 = 15 mod 7 = 1

4	2	(4 * 2) mod 7 = 8 mod 7 = 1
5	3	(5 * 3) mod 7 = 15 mod 7 = 1
6	6	(6 * 6) mod 7 = 36 mod 7 = 1

Astfel: (3 / 2) modulo 7 = (3 * 4) modulo 7 = 5

Există câteva rezultate matematice foarte utile în algăritmică atunci când este necesară calcularea invers modularului.

1. Teorema lui Euler.

Dacă A şi N sunt numere prime între ele atunci: $A^{\varphi(N)} = 1$ (modulo N), unde $\varphi(N)$ se numeşte indicatorul lui Euler şi reprezintă numărul de valori mai mici decât N şi prime între ele cu N. De exemplu, $\varphi(4) = 2$, $\varphi(5) = 4$.

Pentru a calcula $\varphi(N)$ se utilizează următorul rezultat:

$\varphi(N) = N\left(\frac{p_1-1}{p_1}\right)\left(\frac{p_2-1}{p_2}\right)\dots\left(\frac{p_k-1}{p_k}\right)$, unde $p_1, p_2, \dots, p_k$ sunt factorii primi din descompunerea în factori primi a lui N.

Se observă că pentru a calcula această valoare modulo, putem mai întâi să împărţim pe N la factorii primi ai săi şi abia apoi să efectuăm operaţiile modulo care sunt doar înmulţiri (scăderile cu 1 se tratează uşăr).

Aşadar, revenind la teorema lui Euler obţinem: $A \cdot A^{\varphi(n)-1} = 1$ modulo N, adică $A^{\varphi(N)-1}$ este invers modularul lui A, modulo N. Pentru a calcula puterea sunt necesare doar operaţii de înmulţire modulară. Se foloseşte exponenţierea logaritmică.

Costurile utilizării acestei metode sunt:

- Calcul indicatorul lui Euler, complexitate O($\sqrt{N}$)
- Calcul $A^{\varphi(N)}$, complexitate O(log N).
- Nu este necesară memorie suplimentară (calculăm indicatorul lui Euler odată cu descompunerea numărului în factori primi).

În multe cazuri, N este prim. Pentru a calcula invers modularul lui A în raport cu N ne folosim de rezultatele anterioare astfel (în primul rând, dacă A este un număr modulo N, se garantează ca A şi N sunt prime între ele):

$\varphi(N) = N - 1$ (toate numerele mai mici decât N sunt prime între ele cu N).

Astfel, invers modularul lui A în raport cu N este A^{N-2}. Nu mai este necesară calcularea divizorilor ci doar exponenţierea logaritmică.

2. Algoritmul lui Euclid extins.

Modalitatea cea mai eficientă de găsire a invers modularului necesită memorie logaritmică (pe stivă) şi timp de rulare logaritmic. Se foloseşte Algoritmul lui Euclid extins.

Să arătăm modul în care realizăm acest lucru:

- Pornim de la algoritmul lui Euclid, care calculează cel mai mare divizor comun pentru două numere naturale A şi B. Ne interesează varianta recursivă:

```
int cmmdc (int a, int b) {
    if (b == 0)
        return a;
    else
        return cmmdc(b, a%b);
}
```

Cunoaştem că acest algoritm are ordinul de complexitate logaritmic, în funcţie de valorile a şi b.

Vom prezenta acum algoritmul lui Euclid extins. Acesta rezolvă ecuaţii de forma ax + by = cmmdc(a,b)
Se poate demonstra că această ecuaţie admite o infinitate de soluţii. Demonstraţia este constructivă şi rezultă din algoritmul lui Euclid extins. Să notăm d = cmmdc (a,b).
Întrucât cmmdc(a,b) = cmmdc (b,a%b), avem:
Există x, y, x0, y0 astfel încât:
$a \cdot x + b \cdot y = d$
$b \cdot x0 + (a\%b) \cdot y0 = d$
Adică
$a \cdot x + b \cdot y = b \cdot x0 + (a\%b) \cdot y0$
dar $a\%b = a - a/b*b$
Deci, $a \cdot x + b \cdot y = b \cdot x0 + (a - a/b*b) \cdot y0$
Notând c = a/b, ecuaţia devine:
$a \cdot (x-y0) = b \cdot (x0-c \cdot y0-y)$
adic ă x = y0 şi y = x0- c·y0
Am exprimat astfel pe x şi y în funcţie de x0 şi y0, soluţiile ecuaţiei de la pasul anterior al recurenţei.
În algritmul lui Euclid, de la un autoapel la altul trecem de la o pereche a,b (cu soluţiile x,y) la perechea b,a%b (cu soluţia x0,y0). Întrucât am arătat că putem exprima pe x şi y în funcţie de x0 şi y0, vom proceda astfel:
Transmitem funcţiei recursive, alături de parametrii a şi b şi pe x şi y (prin referinţă), soluţiile ecuaţiei ax+by = cmmdc(a,b)
Procedăm astfel:

- Pe ramura fără autoapel (când b = 0, cmmdc este a) ecuaţia este ax = a. Aşadar o soluţie este x = 1 şi y=0 (y poate fi oricât). Deci pe această ramură, calculăm astfel pe x şi y.
- Pe ramura cu autoapelul, pe revenire, avem în x0 şi y0 valorile calculate pentru ecuaţia din autoapel şi folosim formulele deduse pentru a obţine x şi y.

La terminarea funcţiei, x şi y sunt soluţiile ecuaţiei iniţiale.

```
void cmmdc (int a, int b, int &x, int &y) {
    if (b == 0){
        x = 1;
        y = 0;
    }else {
        int x0, y0;
        return cmmdc(b, a%b, x0, y0);
        x = y0;
        y = x0 – a/b*y0;
    }
}
```

Folosind Algoritmul lui Euclid extins vom arăta cum calculăm invers modularul unui număr A, modulo N, cu A şi N prime între ele.
Deoarece cmmdc(A,N) = 1, există x şi y astfel încât $A \cdot x + N \cdot y = 1$, modulo N
Întrucât $N \cdot y$ este divizibil cu N, avem $N \cdot y$ este egal cu 0 modulo N. Ecuaţia devine $A \cdot x = 1$, modulo N, adică x este invers modularul lui A în raport cu N. Rezolvând deci ecuaţia $A \cdot x + N \cdot y = 1$ cu algoritmul lui Euclid extins, găsim în x invers modularul lui A în raport cu N.