

# Análise Combinatória

## Combinações

---

Prof. Edson Alves

Faculdade UnB Gama

1. Combinações
2. Coeficientes binomiais
3. Equações lineares com coeficientes unitários
4. Soluções dos problemas propostos

# Combinações

---

# Definição

## Definição de combinação

Seja  $A$  um conjunto com  $n$  elementos distintos e  $p$  um inteiro não negativo tal que  $p \leq n$ . Uma **combinação** deste  $n$  elementos, tomados  $p$  a  $p$ , consiste em uma escolha de  $p$  elementos distintos dentre os  $n$  possíveis, onde cada combinação difere das demais pela qualidade dos elementos, mas não pela ordem.

**Notação:**  $C(n, p)$

Por exemplo, se  $A = \{1, 2, 3, 4\}$  e  $p = 2$ , há 6 combinações distintas, a saber:

12, 13, 14, 23, 24, 34

- Se  $p < 0$ , então  $C(n, p) = 0$
- Nos demais casos,  $C(n, p)$  pode ser computado a partir de  $A(n, p)$ : basta contar, como apenas um, todos os arranjos que diferem apenas pela ordem de seus elementos
- Como  $p$  elementos distintos geram  $p!$  arranjos distintos, segue que

$$C(n, p) = \frac{A(n, p)}{p!} = \frac{n!}{(n - p)!p!} = \binom{n}{p}$$

## Caracterização das combinações

- Assim como feito com as permutações e com os arranjos, as combinações também podem ser caracterizadas por meio de uma analogia com um sorteio de bolas
- Neste sentido, uma combinação  $C(n, p)$  corresponderia a retira de  $p$  bolas dentre as  $n$  bolas distintas contidas em uma caixa, sem reposição, onde a ordem das bolas não é relevante
- Assim, as retiradas 123, 321 e 213, por exemplo, seriam todas consideradas uma mesma combinação, uma vez que a qualidade das bolas é a mesma, embora tenha sido retiradas em ordens distintas

## Coeficientes binomiais

---

## Definição de coeficiente binomial

Sejam  $n$  e  $p$  dois inteiros não-negativos tais que  $n \geq p$ . O coeficiente binomial é dado por

$$\binom{n}{p} = \frac{n!}{(n-p)!p!}$$



## Implementação dos coeficientes binomiais

- Na prática, pode ser que o valor de  $\binom{n}{p}$  possa ser armazenado em uma variável inteira, mas o cálculo dos fatoriais envolvidos no numerador e no denominador pode resultar em um *overflow*
- Há duas maneiras de contornar este problema: por cancelamento ou por recorrência
- A ideia do cancelamento é que, embora seja representado na forma de fração,  $\binom{n}{p}$  é sempre um número inteiro
- Assim, é possível realizar os cancelamentos devidos antes de multiplicar os fatores restantes

## Implementação dos binomiais por cancelamento

```
41 long long binom(int n, int m, const vector<long long>& primes)
42 {
43     if (n < m)
44         return 0;
45
46     long long res = 1;
47
48     for (auto p : primes) {
49         if (p > n)
50             break;
51
52         for (int k = E(n, p) - E(m, p) - E(n - m, p); k > 0; --k)
53             res *= p;
54     }
55
56     return res;
57 }
```

# Triângulo de Pascal

- Os números binomiais surgem nas expansões do monômio  $(a + b)^n$ , para  $n$  não-negativo
- Estas expansões formam o Triângulo de Pascal:

1  
1 1  
1 2 1  
1 3 3 1  
1 4 6 4 1  
1 5 10 10 5 1  
...

- A observação cuidadosa deste triângulo permite definir os coeficientes binomiais recursivamente

## Definição recursiva dos coeficientes binomiais

- Sejam  $n$  e  $m$  inteiros não-negativos. Então os casos-base da recursão são

$$\binom{n}{0} = \binom{n}{n} = 1$$

- A transição é dada por

$$\binom{n}{m} = \binom{n-1}{m} + \binom{n-1}{m-1}$$

## Implementação dos coeficientes binomiais por DP

```
1 long long binom(int n, int m)
2 {
3     vector<vector<long long>> dp(n + 1, vector<long long>(n + 1, 0));
4
5     for (int i = 0; i <= n; ++i)
6     {
7         dp[i][0] = dp[i][i] = 1;
8
9         for (int j = 1; j < i; ++j)
10             dp[i][j] = dp[i - 1][j] + dp[i - 1][j - 1];
11     }
12
13     return dp[n][m];
14 }
```

## Redução da complexidade de memória

- A implementação acima tem complexidade  $O(n^2)$  para execução e para memória
- É possível reduzir o uso de memória para  $O(m)$  através de uma implementação cuidadosa, que se vale das propriedades da recorrência
- A ideia central é computar os coeficientes de cada linha da direita para a esquerda, uma vez que o coeficiente da próxima linha é computado a partir do coeficiente que ocupa a mesma posição na linha anterior e o coeficiente da linha anterior na posição anterior

## Implementação dos coeficientes binomiais em $O(nm)$ memória $O(m)$

```
1 long long binom(int n, int m)
2 {
3     if (m > n)
4         return 0;
5
6     vector<long long> dp(m+1, 0);
7     dp[0] = 1;
8
9     for(int i = 1; i <= n; ++i)
10         for(int j = m; j > 0; --j)
11             dp[j] = dp[j] + dp[j - 1];
12
13     return dp[m];
14 }
```

## Propriedades dos coeficientes binomiais

- Combinações complementares (permite a redução do espaço de memória em 50%):

$$\binom{n}{p} = \binom{n}{n-p}$$

- Soma de uma linha (consequência da expansão do binômio  $(1 + 1)^n$ ):

$$\binom{n}{0} + \binom{n}{1} + \dots + \binom{n}{n} = 2^n$$



## Propriedades dos coeficientes binomiais

- Soma alternada de uma linha (consequência da expansão do binômio  $(1 - 1)^n$ ):

$$\binom{n}{0} - \binom{n}{1} + \dots + (-1)^n \binom{n}{n} = 0$$

- Soma de uma coluna, com  $n \geq p$  (*Hockey-Stick Identity*):

$$\binom{p}{p} + \binom{p+1}{p} + \binom{p+2}{p} + \dots + \binom{n}{p} = \binom{n+1}{p+1}$$

- Soma de uma linha com coeficientes lineares:

$$\sum_{k=0}^n k \binom{n}{k} = n2^{n-1}$$

- Soma de uma linha com coeficientes quadráticos:

$$\sum_{k=0}^n k^2 \binom{n}{k} = (n^2 + n)2^{n-2}$$

- Soma dos quadrados dos coeficientes de uma linha:

$$\sum_{k=0}^n \binom{n}{k}^2 = \binom{2n}{n}$$

- Se  $F(n)$  é o  $n$ -ésimo número de Fibonacci, vale que

$$\sum_{k=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n-k}{k} = F(n+1)$$

## **Equações lineares com coeficientes unitários**

---

## Equações lineares com coeficientes unitários

- Considere, para  $r$  natural e  $n$  inteiro, a equação linear dada por

$$x_1 + x_2 + \dots + x_r = n$$

- Quando as variáveis  $x_i$  pertencem aos reais, racionais ou inteiros, a equação tem infinitas soluções
- O número de soluções, porém, é finito, ou mesmo pode não existir solução, caso as variáveis estejam restritas aos inteiros positivos

- De fato, se  $n < r$ , a equação não tem solução nos inteiros positivos
- Para  $n \geq r$ , o valor  $n$  pode ser escrito como

$$n = 1 + 1 + 1 + \dots + 1$$

- Cada solução pode ser construída substituindo-se  $r - 1$  dentre os símbolos '+' da soma anterior por barras verticais'
- A soma resultante à esquerda de cada uma das barras, e à direita da última, corresponde aos valores das  $r$  variáveis  $x_i$
- Esta estratégia é conhecida como barras e estrelas (*stars and bars*)

## Soluções, restritas aos positivos, das equações lineares com coeficientes unitários

- Cada uma das soluções nos inteiros positivos corresponde a um posicionamento distinto das barras
- Assim, o total de soluções é dado por

$$S = C(n - 1, r - 1) = \binom{n - 1}{r - 1}$$

- Ou seja, basta tomar  $r - 1$  dentre os  $n - 1$  símbolos '+'

## Equações lineares com coeficientes unitários restritas aos não-negativos

- Caso as variáveis  $x_i$  possam assumir também o valor zero, o novo total de soluções pode ser determinado por meio de uma mudança de variáveis
- Considere a equação abaixo, com  $r$  e  $n$  positivos e  $x_i \geq 0$ :

$$x_1 + x_2 + \dots + x_r = n$$

- Fazendo  $y_i = x_i + 1$ , isto é,  $x_i = y_i - 1$ , obtém-se a equação equivalente

$$y_1 + y_2 + \dots + y_r = n + r, \quad y_i \geq 1$$



## Soluções das equações lineares com coeficientes unitários restritas aos não-negativos

Assim, o número de soluções da equação original, restrito aos inteiros não-negativos, é dado por  $C(n + r - 1, r - 1)$ , ou sua combinação complementar,  $C(n + r - 1, n)$ .

Por exemplo,

$$x_1 + x_2 + x_3 = 10$$

tem

$$C(10 - 1, 3 - 1) = C(9, 2) = 36$$

soluções nos inteiros positivos, e

$$C(10 + 3 - 1, 3 - 1) = C(12, 2) = 66$$

soluções nos inteiros não-negativos.

# Combinações com repetição

## Definição de combinação com repetição

Uma combinação com repetição de  $n$  elementos distintos, tomados  $p$  a  $p$ , é um escolha de  $p$  objetos, dentre os  $n$  possíveis, onde cada objeto pode ser escolhido até  $p$  vezes.

**Notação:**  $CR(n, p)$

- Seja  $x_i$  a quantidade de vezes que o objeto  $i$  foi escolhido em uma combinação, com  $0 \leq x_i \leq p$
- Então o número de combinações com repetição de  $n$  elementos tomados  $p$  a  $p$  será igual ao número de soluções da equação

$$x_1 + x_2 + \dots + x_n = p$$

- Conforme visto anteriormente,

$$CR(n, p) = C(p + n - 1, n - 1) = C(p + n - 1, p)$$

## Caracterização das combinações com repetições

- A combinação com repetição é o primeiro de quatro problemas fundamentais de contagem
- Estes problemas tratam da questão de se distribuir  $n$  bolas em  $p$  caixas
- Na combinação com repetições, as  $n$  bolas são idênticas e as  $p$  caixas são distintas
- Observe que, nesta analogia, uma ou mais caixas podem ficar vazias ( $x_i \geq 0$ )

## Problemas propostos

1. [AtCoder Beginner Contest 094D](#) – Binomial Coefficients
2. [AtCoder Beginner Contest 132D](#) – Blue and Red Balls
3. [Codeforces 478B](#) – Random Teams
4. [OJ 10219](#) – Find the ways!
5. [OJ 11955](#) – Binomial Theorem

1. GeeksForGeeks. [Dinamic Programming Binomial Coeficient](#). Acesso em 14/01/2021.
2. **SANTOS**, José Plínio O., **MELLO**, Margarida P., **MURARI**, Idani T. *Introdução à Análise Combinatória*, Editora Ciência Moderna, 2007.
3. Wikipédia. [Binomial coefficient](#). Acesso em 14/01/2021.
4. Wikipédia. [Hockey-stick identity](#). Acesso em 14/01/2021.
5. Wikipédia. [Stars and bars](#). Acesso em 14/01/2021.

## **Soluções dos problemas propostos**

---

**Versão resumida do problema:** divida  $n$  participantes em  $m$  times de modo a minimizar e a maximizar o número de pares de amigos. Cada time deve ter no mínimo um participante e os membros de um mesmo time se tornarão amigos.

**Restrição:**  $1 \leq m \leq n \leq 10^9$



## Solução com complexidade $O(1)$

- Para maximizar o número de pares de amigos é preciso formar o maior grupo possível
- Isto significa que devem ser formados  $m - 1$  grupos com um único participante e um grande grupo com o  $A = n - (m - 1)$  restantes
- Assim,

$$k_{\max} = \binom{A}{2} = \frac{A(A - 1)}{2}$$

- Já para minimizar o número de amigos os participantes devem ser distribuídos da maneira mais uniforme possível

## Solução com complexidade $O(1)$

- Pela divisão de Euclides,  $n = mq + r$ , com  $0 \leq r < m$
- Logo devem ser formados  $m - r$  grupos com  $B = q$  membros, e os  $r$  grupos restantes terão um membro a mais, isto é,  $B + 1$  membros
- Deste modo,

$$k_{\min} = (m - r) \binom{B}{2} + r \binom{B + 1}{2} = (m - r) \frac{B(B - 1)}{2} + r \frac{(B + 1)B}{2}$$

- Esta solução tem complexidade  $O(1)$

## Solução com complexidade $O(1)$

```
6 pair<ll, ll> solve(ll n, ll m)
7 {
8     auto A = n - (m - 1);
9     auto kmax = A*(A - 1)/2;
10
11     auto B = n / m, r = n % m;
12     auto kmin = (m - r)*(B*(B - 1)/2) +
13                 r*((B + 1)*B/2);
14
15     return {kmin, kmax};
16 }
```

**Versão resumida do problema:** dada  $N$  inteiros não-negativos  $a_1, a_2, \dots, a_N$ , determine o par  $a_i > a_j$  que maximiza o coeficiente binomial  $\binom{a_i}{a_j}$ .

**Restrições:**

- $2 \leq N \leq 10^5$
- $0 \leq a_i \leq 10^9$

## Solução com complexidade $O(N)$

- Este problema pode ser resolvido mediante duas importantes observações
- A primeira delas é que, para um inteiro não-negativo  $m$  fixo,

$$\binom{i}{m} < \binom{j}{m}$$

para  $i < j$ ,  $m \leq i, j$

- Isto significa que, para uma coluna fixa, quanto maior a linha do Triângulo de Pascal, maior o valor do coeficiente binomial correspondente
- A segunda observação é que, para uma linha  $n$  fixa, os coeficientes formam uma sequência crescente até o coeficiente central e segue numa sequência decrescente até o último coeficiente

## Solução com complexidade $O(N)$

- Assim, o coeficiente  $\binom{n}{\lfloor n/2 \rfloor}$  é o maior dentre todos de uma mesma linha e, quanto mais próximo deste centro, maior o coeficiente
- Assim, se os valores da sequência deles forem ordenados, o maior deles será o  $a_i$  procurado
- Para determinar o  $a_j$ , é preciso avaliar os  $N - 1$  termos restantes, em relação à sua distância ao centro: o mais próximo deles é o  $a_j$  desejado
- Esta solução tem complexidade  $O(N)$

## Solução com complexidade $O(1)$

```
5 pair<int, int> solve(vector<int>& as) {  
6     sort(as.begin(), as.end());  
7  
8     auto ai = as.back(), aj = -1, dist = 2000000010;  
9     as.pop_back();  
10  
11     for (auto a : as) {  
12         auto k = min(a, ai - a);  
13  
14         if (ai/2 - k < dist) {  
15             aj = a;  
16             dist = ai/2 - k;  
17         }  
18     }  
19  
20     return { ai, aj };  
21 }
```

**Versão resumida do problema:** determine o número de dígitos da representação decimal de  $\binom{n}{k}$ .

**Restrições:**

- $1 \leq k \leq n$ , e
- a resposta é menor do que  $2^{31} - 1$



## Solução em $O(n)$

- O número de dígitos  $D$  de um inteiro  $x$  em uma base  $b > 1$  é dado por

$$D = \lfloor 1 + \log_b x \rfloor$$

- O coeficiente binomial pode ser escrito como

$$\binom{n}{k} = \frac{n!}{(n-k)!k!} = \frac{n \times (n-1) \times \dots \times (n-k+1)}{k \times (k-1) \times \dots \times 2 \times 1}$$

- Combinando ambas expressões, com  $m = n - k + 1$ , obtemos

$$D = \left\lfloor 1 + \sum_{i=m}^n \log_{10} i - \sum_{i=1}^k \log_{10} i \right\rfloor$$

## Solução em $O(n)$

```
5 int solve(int n, int k)
6 {
7     auto logn = 0.0, logk = 0.0;
8
9     for (int i = n; i > n - k; i--)
10         logn += log10(i);
11
12     for (int i = 2; i <= k; i++)
13         logk += log10(i);
14
15     int ans = (int) floor(logn - logk + 1);
16
17     return ans;
18 }
```