Análise Combinatória

Arranjos

Prof. Edson Alves

Faculdade UnB Gama

Sumário

Arranjos

Definição

Definição de arranjo

Seja A um conjunto com n elementos distintos e p um inteiro não negativo tal que $p \leq n$. Um **arranjo** destes n elementos, tomados p a p, consiste em uma escolha de p elementos distintos dentre os n possíveis, onde cada arranjo difere dos demais tanto pela qualidade quanto pela posição dos elementos.

Notação: $A(n, p)$

Por exemplo, se $A = \{1, 2, 3, 4\}$ e $p = 2$, há 12 arranjos distintos, a saber:

$$12, 13, 14, 21, 23, 24, 31, 32, 34, 41, 42, 43$$

Cálculo de $A(n, p)$

- Utilizando a mesma abordagem usada para computar $P(n)$, segue que

$$A(n, p) = n \times (n-1) \times (n-2) \times \ldots \times (n-(p-1))$$

- A lista acima contém p fatores multiplicativos e se assemelha a um fatorial
- Se o termos remanescentes do fatorial forem multiplicados e a expressão dividida por estes mesmos elementos, obtém-se

$$A(n, p) = \frac{n \times (n-1) \times \ldots \times (n-(p-1)) \times (n-p) \times \ldots \times 2 \times 1}{(n-p) \times \ldots \times 2 \times 1} = \frac{n!}{(n-p)!}$$

Implementação de $A(n, p)$ em C++

```cpp
long long A(long long n, long long p)
{
    if (n < p)
        return 0;

    long long res = 1;

    for (long long i = n; i > p; --i)
        res *= i;

    return res;
}
```

Caracterização dos arranjos

- Assim como no caso das permutações, os arranjos podem ser interpretados como a retirada de bolas distintas de uma caixa, sem reposição, onde a ordem da retira é importante
- A diferença em relação às permutações é que número p de bolas a serem removidas não é, necessariamente, igual a n
- Observe que $A(n, n) = P(n)$

Arranjos com repetições

- Nos arranjos com repetições, as bolas são repostas na caixa após cada retirada
- Deste modo, a cada retirada há n possíveis escolhas
- Assim,

$$AR(n, p) = n \times n \times \ldots \times n = n^p$$

Programação Dinâmica em problemas de contagem

- Uma variante mais complicada do arranjo com repetições seria: *Quanto são os arranjos de* n *elementos, sendo que estes elementos não necessariamente distintos?*
- Considere que existam apenas k elementos distintos, que o i-ésimo elemento distinto ocorre n_i vezes e que $n_1 + n_2 + \ldots + n_k = n$
- Este problema, como muitos outros em combinatória, pode ser resolvido por uma relação de recorrência
- Estas relações podem ser implementadas, de forma eficiente, por meio de algoritmos de programação dinâmica

Exemplo: Número de resultados distintos

- Considere o seguinte cenário: a equipes da Escola A e b equipes da escola B participaram de uma gincana escolar. Quantos são os possíveis resultados da gincana, sendo que serão divulgadas as k melhores equipes? Considere que, dentro de uma mesma escola, as equipes sejam indistinguíveis
- Assuma que $k \leq a + b$
- Por exemplo, se $a = 2, b = 3$ e $k = 3$, então os resultados possíveis seriam

AAB ABA ABB BAA BAB BBA BBB

Solução por recorrência

- Seja $\sigma(k, a, b)$ o número de arranjos distintos para as k primeiras posições levando-se em consideração a equipes da escola A e b equipes da escola B
- Observe que, a cada etapa da geração de um determinado arranjo, há duas opções: escolher uma equipe da escola A ou uma equipe de escola B
- Assim,
$$\sigma(k, a, b) = \sigma(k-1, a-1, b) + \sigma(k-1, a, b-1)$$

Solução por recorrência

- São três os casos-base:
    1. $\sigma(k, a, b) = 0$ se $a < 0$
    2. $\sigma(k, a, b) = 0$ se $b < 0$
    3. $\sigma(0, a, b) = 1$
- Considerando que há $O(KAB)$ estados distintos, onde K, A e B são os valores máximos para k, a e b, respectivamente, e que a transição é feita em $O(1)$, esta solução tem complexidade $O(KAB)$

Solução com complexidade $O(KAB)$

```
long long dp(int k, int a, int b)
{
    if (a < 0 || b < 0)
        return 0;

    if (k == 0)
        return 1;

    if (st[k][a][b] != -1)
        return st[k][a][b];

    auto res = dp(k - 1, a - 1, b) + dp(k - 1, a, b - 1);

    st[k][a][b] = res;
    return res;
}
```

Problemas propostos

1. AtCoder Beginner Contest 046B – Painting Balls with AtCoDeer
2. AtCoder Beginner Contest 159A – The Number of Even Pairs
3. 630C – Lucky Numbers
4. 11115 – Uncle Jack

Referências

1. **SANTOS**, José Plínio O., **MELLO**, Margarida P., **MURARI**, Idani T. *Introdução à Análise Combinatória*, Editora Ciência Moderna, 2007.

Soluções dos problemas propostos

AtCoder Beginner Contest 159A – The Number of Even Pairs

Versão resumida do problema: dadas $N + M$ bolas, determine o número de maneiras de se escolhar duas destas bolas de forma que a soma dos números escritos em ambas bolas seja par. A ordem da retirada das bolas deve ser desconsiderada, em N bolas os números escritos são pares e nas outras M estão escritos números ímpares.

Restrições:

- $0 \leq N, M \leq 100$
- $2 \leq N + M$

Solução com complexidade $O(1)$

- Para que a soma dos números escritos nas bolas seja par há apenas duas possibilidades:
    - escolher duas bolas com números pares, ou
    - escolher duas bolas com números ímpares
- No primeiro caso, há $A(N, 2)$ formas de se escolher duas bolas com números pares, mas como a ordem de retirada não importa no problema e o arranjo contabiliza ordens distintas, é preciso dividir este resultado por 2
- O mesmo vale para as M bolas ímpares
- Assim, a solução S será dada por

$$S = \frac{A(N, 2) + A(M, 2)}{2} = \frac{N(N-1) + M(M-1)}{2}$$

Solução com complexidade $O(1)$

```cpp
#include <bits/stdc++.h>

using namespace std;

int main()
{
    int N, M;
    cin >> N >> M;

    cout << N*(N - 1)/2 + M*(M - 1)/2 << '\n';

    return 0;
}
```

Codeforces 630C – Lucky Numbers

Versão resumida do problema: Determine quantos números distintos de até N dígitos podem ser formados utilizando apenas os dígitos 7 e 8.

Restrição: $1 \leq N \leq 55$

Solução com complexidade $O(1)$

- Para um M fixo, há 2^M números distintos que podem ser formados usando os dígitos 7 e 8, pois, para cada posição há duas escolhas: 7 ou 8
- Assim, a resposta S do problema é dada por

$$S = \sum_{i=1}^{N} 2^i = 2^1 + 2^2 + \ldots + 2^N$$

- Observe que

$$S + 1 = 1 + 2^1 + 2^2 + \ldots + 2^N = 2^{N+1} - 1$$

- Assim $S = 2^{N+1} - 2$ e esta expressão pode ser computada em $O(1)$ por meio de um deslocamento binário

Solução com complexidade $O(1)$

```cpp
#include <bits/stdc++.h>

using namespace std;

int main()
{
    int N;
    cin >> N;

    cout << (1LL << (N + 1)) - 2 << '\n';

    return 0;
}
```