# Tutorial: Alazão

Eric Grochowicz

O problema pode ser visto como "Dado um grafo direcionado, existe um passeio que sai do nodo 1 e vai até o $N$ de tamanho múltiplo de $K$?"

Dica 1: Podemos observar que pode ser benéfico fazer um passeio que passa por um mesmo ciclo mais de uma vez a fim de que o tamanho do passeio fique múltiplo de $K$.

Dica 2: A partir da observação anterior, temos que nunca é necessário visitar um mesmo nodo mais de $K$ vezes, uma para cada resto da divisão do tamanho do caminho por $K$ possível.

Para resolver esse problema, podemos criar um novo grafo com $N \cdot K$ nodos e $M \cdot K$ arestas.

Nesse novo grafo, para cada nodo $u$ do grafo original serão criados $K$ nodos $\{u, x\}$ (para todo $x$ de 0 até $K-1$). Agora, o novo grafo possui $N \cdot K$ nodos $\{u, x\}$, onde $u$ é o nodo do grafo original e $x$, o resto da divisão do tamanho do passeio de 1 até o nodo atual.

Ao percorrer o grafo novo, saindo de um nodo $\{u, x\}$, é possível ir para qualquer nodo $\{v, (x+1) (\text{mod } K)\}$ tal que existe uma aresta que conecta $u$ com $v$ no grafo original.

É suficiente percorrer o grafo novo com DFS ou BFS partindo do nodo $\{1, 0\}$ e verificar se o nodo $\{N, 0\}$ foi visitado. A complexidade da solução é $O((N + M) \cdot K)$.

# Tutorial: Cidades Planejadas

Vinicius Borges

O problema deve ser resolvido verificando se o mapa fornecido é simétrico em relação à diagonal principal, logo precisamos verificar se o caractere na posição $(i, j)$ é igual ao caractere da posição $(j, i)$, para todos $i$ e $j$.

A complexidade deste código fica em $O(N^2)$, devido à necessidade de se acessar todas as posição abaixo (ou acima) da diagonal principal, e acessando-se sua posição associada simétrica.

# Tutorial: De volta para casa

Eduardo Freire

Enraíze a árvore no vértice 1. Denotaremos por $E(x)$ o valor esperado de arestas em um caminho aleatório do vértice $x$ para o vértice 1 e por $p(x)$ o vértice pai de $x$ (convencionamos que $p(1)$=1).

Para cada vértice $x$ calcularemos um par de valores $f(x) = (a_x, b_x)$. Esse par significará que $E(x) = a_x + b_x \cdot E(p(x))$. Para calcular esses pares, usamos uma dfs. Se $x$ é uma folha, temos que $E(x) = 1 + E(p(x))$, logo $f(x) = (1, 1)$. Agora assuma que $x$ não é uma folha e $x \neq 1$. Chame os filhos de $x$ de $y_1, \ldots, y_{m-1}$ (note que $x$ possui exatamente $m$ vizinhos, contando com o seu pai). Temos

$$E(x) = 1 + \frac{E(p(x))}{m} + \frac{1}{m} \sum_{i=1}^{m-1} E(y_i).$$

Agora calculamos recursivamente os valores dos termos que aparecem no somatório acima. Isso nos dará $E(y_i)$ em termos de $E(x)$, já que cada $y_i$ tem $x$ como pai. Logo, temos

$$E(x) = 1 + \frac{E(p(x))}{m} + \frac{1}{m} \sum_{i=1}^{m-1} f(y_i).\text{first} + f(y_i).\text{second } E(x).$$

Defina $a = \sum_{i=1}^{m-1} f(y_i).\text{first}$ e $b = \sum_{i=1}^{m-1} f(y_i).\text{second}$. Reescrevendo a equação encontrada para $E(x)$ em termos de $a$ e $b$ e multiplicando-a por $m$, obtemos

$$mE(x) = m + E(p(x)) + a + bE(x).$$

Isolando $E(x)$ concluímos que

$$E(x) = \frac{m + a}{m - b} + \frac{E(p(x))}{m - b}.$$

Segue que $f(x) = (\frac{m+a}{m-b}, \frac{1}{m-b})$.

Tendo os valores $f(x)$ em mão e o fato que $E(1) = 0$, podemos facilmente calcular os valores de $E(x)$ de cima para baixo com uma segunda dfs.

# Tutorial: Estrutural

Daniel Saad Nogueira Nunes

Primeiramente, devemos construir um grafo $G = (V, E)$ não-direcionado baseado na distância de Hamming. Isto é, para quaisquer duas strings $S_u$ e $S_v$, criamos a aresta $(u, v)$, desde que o número de caracteres que sejam diferentes de $S_u$ para $S_v$ seja menor ou igual a $X$. Isso pode ser feito em tempo $\Theta(N^2)$.

Em seguida, precisamos detectar qualquer ciclo hamiltoniano em $G$ e imprimi-lo. Para isso, utilizamos a abordagem de programação dinâmica.

Suponha que exista um caminho hamiltoniano do vértice 0 até o vértice $i$ considerando apenas o grafo induzido pelo subconjunto $S \subseteq V$ de vértices. Obviamente, $0 \in S$ e $i \in V$. Tome um vértice $j$ qualquer de forma que $j \notin S$. Assim, podemos concluir que existe um caminho hamiltoniano que inicia no vértice 0 e termina no vértice $i$, considerando o conjunto $S \cup \{j\}$ se e somente se existe uma aresta $(i, j)$.

Os conjuntos de $S$ podem ser representados por um vetor de bits de tamanho $N$, logo, podem ser armazenados em variáveis inteiras. Como são $2^N$ subconjuntos possíveis, temos que criar uma tabela de programação dinâmica de tamanho $T[0, n-1][0, 2^N - 1]$, de modo que $T[i][S]$ que existe um caminho hamiltoniano que inicia em 0 e termina $i$ considerando o conjunto $S \subseteq V$. Além disso, $T[i][S]$ diz qual o nó que precede o vértice $i$ neste caminho, para conseguirmos recuperar a resposta. A montagem da tabela de programação dinâmica segue abaixo.

```
/** matrix[i][S] stores the befor the last vertex in the path
    * iff there is a hamiltonian path
    * starting at vertex 0 and ending in vertex i considering all
    * vertex in S.
    */
vector<vector<int>> matrix(n, vector<int>(1 << n, -2));
matrix[0][1 << 0] = -1;
for (int mask = 0; mask < 1 << n; mask++) {
    for (int j = 0; j < n; j++) {
        // We pick subsets that contain vertex j
        if ((mask & (1 << j)) == 0)
            continue;
        // For every vertex  i != j
        for (int i = 0; i < n; i++) {
            // If there is a hamilton path regarding subset S\{j}
            // ending in i and there is an edge i->j
            if (mask & 1 << i and i != j and
                matrix[i][mask ^ (1 << j)] != -2 and graph[i][j]) {
                matrix[j][mask] = i;
            }
        }
    }
}
return matrix;
```

Existirá o ciclo hamiltoniano apenas se $T[i][V]$ está definido e se existe uma aresta de $(i, 0)$. Ao consultar a tabela $T$ é possível reconstruir esse caminho.

A solução possui complexidade $\Theta(N^2 \cdot 2^N)$.

# Tutorial: Fila da Padaria

Caleb Martim

Processe cada elemento do vetor $A$ do maior valor para o menor valor, dessa forma, quando estivermos processando o elemento $A_i$, a resposta do problema no índice $i$ é a mesma resposta para a pergunta "Quantos elementos com índices menores do que $i$ no vetor $A$ já foram processados e possuem valores maiores que $A_i$?". Para respondermos isso podemos definir previamente um vetor $P$ de tamanho $N$ tal que quando estivermos computando a resposta para o elemento $A_i$, $P_j = 1$ se o elemento no índice $j$ já foi processado e $A_j > A_i$, ou $P_j = 0$ caso contrário. Dessa forma, a resposta então será o valor da soma dos elementos com índices no intervalo $[1, i]$ em $P$. Se construirmos uma estrutura como uma SegTree ou uma Fenwick Tree sobre $P$, a resposta para cada consulta pode ser computada em $O(\log_2(N))$.

Inicialmente, todo valor em $P$ é 0. Para sabermos quando um valor em $P$ deve ser atualizado para 1, podemos manter uma pilha onde iremos inserir nossos elementos após computarmos suas respostas e iremos os retirar da pilha para atualizarmos $P$ apenas quando estivermos tentando computar a resposta para um elemento com valor menor a destes elementos que estão na pilha. Note que esta pilha é monotônica e, portanto, a complexidade de seu uso é $O(N)$.

Assim finalizamos a solução.

Complexidade de tempo: $O(N \cdot \log_2(N))$

Complexidade de memória: $O(N \cdot \log_2(N))$

# Tutorial: Gama, sempre Gama!

Edson Alves

Seja $dp(i, t)$ a maior soma de fatores engajamento possível escolhendo zero ou mais elementos dentre os $i$ primeiros clipes, cuja soma das durações é menor ou igual a $t$ segundos. O caso base é dado por $dp(0, 0) = 0$, ou seja, sem nenhuma opção de escolha, a soma máxima é zero.

Para $i > 0$ há duas transições possíveis. A primeira delas é

$$dp(i, t) = dp(i - 1, t),$$

que significa que o $i$-ésimo clipe não foi selecionado. Caso $t_i \leq t$, há uma segunda transição possível, dada por

$$dp(i, t) = dp(i - 1, t - t_i) + e_i,$$

correspondente à escolha do $i$-ésimo clipe. Para cada clipe, deve ser escolhida, dentre as transições possíveis, a que maximiza o valor de $dp(i, t)$.

A resposta do problema é dada por $dp(N, 60 \times M)$ (pois $M$ é dado em minutos). Como há $O(NM)$ estados e as transições são feitas em $O(1)$, esta solução tem complexidade $O(NM)$.

# Tutorial: Hortaliças

Edson Alves

O problema consiste em determinar o segmento de reta $r$, paralela a $AB$ e $CD$, entre ambas, que minimiza a função

$$d(r, N) = \sum_{i=1}^{N} \text{dist}(r, P_i)$$

Primeiramente, lembre que a distância entre a reta $r$ e o ponto $P = (x, y)$ é dada por

$$\text{dist}(r, P) = \frac{|ax + by + c|}{\sqrt{a^2 + b^2}}$$

Seja $M = D_y - A_y$. Uma solução que avalia todas as $M - 1$ retas possíveis tem complexidade $O(MN)$, e leva a um veredito TLE.

Para reduzir a complexidade da solução, observe que, como todas as retas são paralelas, os coeficientes $a$ e $b$ serão idênticos em todas as possíveis retas. Assim, a função a ser minimizada pode ser simplificada para

$$d(r, N) = \sum_{i=1}^{N} |ax_i + by_i + c_r|,$$

dispensando o cálculo da raiz quadrada, o uso de frações ou de aritmética de ponto flutuante.

Agora, considere um segmento de reta $r$ fixo, com coeficiente $c_r$. Para $c_s = c_r + 1$, há dois cenários possíveis que relacionam $d(s, N)$ com $d(r, N)$:

1. se $c_r \geq -(ax_i + by_i)$, então $|ax_i + by_i + c_s| = |ax_i + by_i + c_r| + 1$;
2. se $c_r < -(ax_i + by_i)$, então $|ax_i + by_i + c_s| = |ax_i + by_i + c_r| - 1$;

Seja $\sigma_r(P_i)$ a função sigma de Kronecker tal que $\sigma_r(P_i) = 1$ se $c_r > -(ax_i + by_i)$, ou zero, caso contrário; e $\mu_r(P_i)$ tal que $\mu_r(P_i) = 1$ se $c_r < -(ax_i + by_i)$, e zero, caso contrário. Assim,

$$\delta(r, N) = \sum_{i=1}^{N} \sigma_r(P_i) - \mu_r(P_i)$$

computa a variação $d(s, N) - d(r, N)$. Como $\delta(r, N)$ é monótona não-decrescente em relação ao coeficiente $c_r$, a função $d(r, N)$ é unimodal. Assim, a reta que minimiza $d(r, N)$ pode ser determinada por meio de uma busca ternária nos valores de $c$ das retas candidatas a solução.

Esta solução tem complexidade $O(N \log M)$.

# Tutorial: Jogo da Moeda

Eduardo Moreira

Todo esse tutorial irá usar a indexação do círculo em zero.

Dado uma posição $i$ que pode andar em múltiplos de $a_i$ pelo círculo, vamos provar que $i$ pode andar em múltiplos de $gcd(a_i, N)$ pelo círculo.

Dizer que $i$ pode "andar" em múltiplos de $a_i$ pelo círculo nos da a seguinte equação modular:

$$i + k \cdot a_i \equiv j \ (mod\ N)$$

Que pode ser reescrita da seguinte forma:

$$i + k \cdot a_i - j = c \cdot N$$

Rearranjando os termos, temos:

$$k \cdot a_i - c \cdot N = j - i$$

A equação acima é uma equação diofantina da forma $(Ax + By = C)$ com constantes $A = a_i$ e $B = N$, e é provado que para quaisquer par $(x, y)$ de inteiros, $C = (j - i)$ pode assumir apenas múltiplos de $gcd(A, B)$.

Com isso, sabemos que dado uma posição $i$, essa posição pode ir, em um movimento, para qualquer outra posição $j$ se e somente se:

$$i \equiv \mathrm{j}\ (mod\ gcd(a_i, N))$$

Com essa equação, conseguimos construir um grafo direcionado, tal que $i$ tem uma aresta para $j$ se $i \equiv \mathrm{j}\ (mod\ gcd(a_i, N))$.

Infelizmente, esse grafo possui uma quantidade quadrática de arestas, mais especificamente, o grafo possui essa quantidade de arestas:

$$\sum_{i|N} \left( \varphi\left(\frac{N}{i}\right) \cdot \frac{N}{\mathrm{gcd}(i,N)} \right)$$

Onde $\varphi(x)$ é a quantidade de números menores ou iguais a $x$ e coprimos com $x$.

Provar que essa soma é quadrática não vem muito ao caso aqui, mas você pode pensar no caso que $N$ é primo, nesse caso, a quantidade de arestas do grafo é igual a $N^2$.

Porém, podemos perceber que o conjunto de arestas de um nodo $i$ é definido por duas coisas

- $gcd(a_i, N)$.
- $i\ mod\ gcd(a_i, N)$

Com essa observação, podemos diminuir o número de arestas do grafo. Pois sempre que visitamos um nodo que já teve o seu conjunto de arestas visitadas, não precisamos percorrer as arestas dele novamente.

Vamos provar que o número de arestas desse grafo é $O(N \cdot D)$ onde $D$ é o número de divisores de $N$.

Vamos contar a contribuição de arestas para cada divisor de $N$, vou chamar esse divisor de $d$. Sabemos que só existem $d$ nodos que importam para esse divisor, pois só ligamos para o $mod$ da posição no círculo por $d$, e o número de arestas que cada uma dessas posições contribui para o grafo é $\frac{N}{d}$. Com isso, temos que $d$ contribui $d \cdot \frac{N}{d} = N$ arestas.

Então podemos fazer uma bfs normal nesse grafo, e só botamos um nodo na fila se nenhum dos nodos visitados até agora tem aquelas duas propriedades mencionadas anteriormente iguais as dele.

# Tutorial: Kart Indoor

Vinicius Borges

Observe que as numerações dos karts segue a ordem de largada, isto é, $1, 2, \ldots, N$. A partir da ordem de chegada dos karts, precisamos determinar a quantidade de trocas (*swaps*) que serão efetuadas para recuperarmos a ordem de largada, que está naturalmente ordenada de maneira crescente.

Para isso, um algoritmo de ordenação que possui tempo de execução $O(N^2)$ já serve para esse propósito uma vez que $N$ é $5 \cdot 10^3$ no pior caso. Sempre que uma troca de posição for efetuada, contabilize-a e ao final, apresente este valor na tela.

Abaixo segue a solução por meio do algoritmo Bubble Sort:

```
typedef long long ll;

int main(){
    int n,aux;
    ll ans;
    int karts[5001];

    scanf("%d",&n);

    for(int i = 0; i < n; i++){
        scanf("%d",&karts[i]);
    }

    ans = 0;

    for(int i = 0; i < n; i++){
        for(int j = i+1; j < n; j++){
            if(karts[i] > karts[j]){
                aux = karts[i];
                karts[i] = karts[j];
                karts[j] = aux;
                ans++;
            }
        }
    }

    printf("%lld\n",ans);

    return 0;
}
```

# Tutorial: Luka, Miku e Chocolate

Caleb Martim

O problema pode ser modelado como um grafo direcionado onde o número que identifica uma pessoa é um vértice e uma aresta é ligada entre vértices $a$ e $b$ se a pessoa $a$ gosta da pessoa $b$. É possível observar que a resposta de cada consulta é equivalente à resposta da pergunta "Existe um caminho do vértice $a$ para o vértice $b$ no grafo direcionado?"

Note, primeiramente, que devido às restrições da entrada, o grafo é composto por um conjunto de componentes conexos, de modo que cada componente conexo possui no máximo um ciclo simples. Para os vértices que não compõe ciclos, se a direção de suas arestas for invertida, esses vértices irão compor árvores.

Usando o algoritmo de Kahn para ordenação topológica, encontraremos os vértices que compõe ciclos e os que compõe árvores. Caso um vértice seja visitado na execução do algoritmo, ele faz parte de uma árvore, caso contrário, ele faz parte de um ciclo.

Agora, sejam $a$ e $b$ dois vértices pertencentes ao grafo. Caso eles não façam parte do mesmo componente conexo, não há caminho de $a$ para $b$. Uma DFS pode ser executada para identificar os vértices de cada componente conexo. A complexidade de executar a DFS é $O(N + M)$. Após esse cálculo, será possível verificar se dois vértices fazem parte do mesmo componente conexo em $O(1)$.

Caso $a$ e $b$ façam parte do mesmo componente conexo, temos quatro casos que podemos analisar:

- $a$ **e** $b$ **fazem parte de um ciclo:** como cada componente conexo possui no máximo um ciclo, $a$ e $b$ necessariamente fazem parte do mesmo ciclo. Logo, com certeza há um caminho de $a$ para $b$ neste caso.

- $a$ **faz parte de um ciclo, mas** $b$ **não:** observe que se $a$ faz parte de um ciclo, os caminhos possíveis começando de $a$ só terminam em vértices que fazem parte do mesmo ciclo, logo não há caminho de $a$ para $b$ neste caso.

- $a$ **não faz parte de um ciclo, mas** $b$ **faz:** é possível verificar que, aqui, se um componente conexo possui um ciclo simples, todo vértice do componente possui um caminho para pelo menos um vértice que pertence ao ciclo, mas como um vértice pertencente a um ciclo sempre possui caminho para outro vértice que faz parte do mesmo ciclo, então todo vértice no componente conexo possui, na verdade, caminho para todo vértice que pertence ao ciclo do componente. Portanto, com certeza há um caminho de $a$ para $b$ neste caso.

- $a$ **não faz parte de um ciclo, nem** $b$ **faz:** como mencionado anteriormente, quando a direção das arestas são invertidas, esses vértices passam a fazer parte de uma árvore. Como as arestas foram invertidas, queremos verificar agora se há um caminho do vértice $b$ para o vértice $a$ na árvore. Note que esse caminho existe se, e somente se, $b$ é ancestral de $a$ nessa árvore.

  Para cada uma das árvores criadas, encontre a raíz da árvore e inicie uma DFS a partir dela. Com essa DFS calcule o tempo de entrada e o tempo de saída de cada vértice na execução da DFS, guarde essas informações respectivamente em vetores $tempoDeEntrada$ e $tempoDeSaida$. Sabemos que $b$ é ancestral de $a$ numa árvore quando

  $$tempoDeEntrada[b] \leq tempoDeEntrada[a] \text{ e } tempoDeSaida[a] \leq tempoDeSaida[b]$$

  Esses vetores podem ser computados em $O(N + M)$ e, assim, verificar se $b$ é ancestral de $a$ pode ser realizado em $O(1)$.

Assim, cobrimos todos os casos para respondermos às consultas e finalizamos nossa solução.
Complexidade de tempo: $O(N + M + Q)$
Complexidade de memória: $O(N + M)$