Tutorial: Abdução Alienígena

Vinicius Borges

A interpretação do problema mostra que se trata de um problema de caminhos mínimos em grafos. Podemos visualizar o mapa da zona rural como se fosse um grafo, em que cada abrigo é um vértice e as trilhas, que ligam dois abrigos, são as arestas. Assume-se que existem N vértices e M arestas nesse grafo.

Como o disco voador não precisa "andar/caminhar" pela trilha, percebe-se que o trajeto é feito voando e os pesos das arestas do grafo podem ser ignorados. Logo, a partir do abrigo de partida, o disco voador percorre o grafo por meio da técnica "busca em largura", podendo-se obter o menor caminho d_{DV} desde o vértice de partida até o vértice associado à cachoeira. Esse passo pode ser feito em uma complexidade computacional $O(N + M)$.

Aristênio precisa obrigatoriamente caminhar pelo terreno, então o menor caminho d_A desde a sua casa (vértice 1) até o abrigo deve ser feito utilizando o algoritmo de Dijkstra. A complexidade computacional para uma implementação utilizando uma heap mínima e representação do grafo por lista de adjacências é $O(N + M \log N)$.

Para que Aristênio consiga se salvar da abdução, basta verificar se $d_A < d_{DV}$. A complexidade computacional deste algoritmo é $O(N + M) + O(N + M \log N)$.

Tutorial: Boate Azul

Daniel Porto

Para resolver o problema, deve-se criar todos os pares de letras possíveis a partir da entrada. Depois de ordenar esses pares, ir inserindo sempre o primeiro par na lista final e removendo todos os outros pares que contém as letras que estão sendo inseridas. Após isso, deve-se verificar se todas as letras foram inseridas para garantir a listagem correta.

Tutorial: Carimbador Maluco

Guilherme Ramos

Para cada entrada, basta verificar se as palavras selado, registrado, carimbado, avaliado e rotulado existem, nesta ordem, na entrada. Uma forma é buscar cada uma delas e armazenar o índice em que ocorre na entrada. Se alguma não for encontrada, `‘‘Nao vai a lugar nenhum!’’`. Caso contrário, a resposta é `‘‘Sim, sim, sim, sim!’’` apenas se os índices encontrados estão em ordem crescente.

Tutorial: Entra no meu Time

Jeremias Moreira Gomes

O problema é focado na restrção de memória. Utilizando um vetor de 64 posições, é possível acumular a quantidade de bits por posição de todos os números da lista. Após o acúmulo, se a quantidade de bits daquela posição não for múltiplo de três, então aquele bit faz parte do número final.

Tutorial: Formiguinhas

Guilherme Ramos

Leia duas linhas, ignore os conteúdos, e apresente as mensagens como indicadas nos exemplos.

Tutorial: Hora de Ao Mossar

Alberto Neto

Pensando no problema como uma recorrência, podemos definir a função f com $f(x)$ igual ao valor esperado de almoços começando com x centavos para alcançar 00 centavos. É claro que $f(0) = 0$, e temos a recorrência

$$f(x) = 1 + \sum_{j=0}^{99} p[j] * f((x + j)\%100)$$

onde $p[j]$ é a probabilidade de Duda comprar um almoço de j centavos. Se considerarmos cada $f(x)$ como uma variável, teremos um sistema de equações lineares de 99 variáveis e 99 equações. Utilizando o algoritmo de Gauss, resolvemos o sistema e teremos o valor de todo $f(x)$.

O único problema dessa abordagem é que alguns valores podem ser inalcançáveis. Por exemplo, se tivermos almoços apenas de preços pares e começarmos com ímpar centavos, nunca será possível chegar no 00. Neste caso, o algoritmo de Gauss poderá acusar que não há solução para o sistema. Para evitarmos isso podemos fazer uma dfs a partir do 0 e encontrar todos os valores alcançáveis. Fazendo Gauss apenas para as variáveis alcançáveis e suasa equações correspondentes, teremos uma única resposta.

Tutorial: Jiraiya-ja-ya

Daniel Saad Nogueira Nunes

Esse problema pode ser resolvido através de programação dinâmica através da seguinte equação de recorrência:

$$dp[i][j] = \begin{cases} 0, \quad \text{se } i \geq j \\ dp[i+1][j-1], \quad s[i] == s[j] \\ \min\{dp[i+1][j]+1, dp[i][j-1], dp[i+1][j-1]\}+1, \quad \text{caso contrário} \end{cases}$$

A matriz de programação dinâmica deve ser preenchida diagonal à diagonal.

Complexidade: $\Theta(n^2)$, visto que cada cédula da matriz pode ser preenchida em tempo constante.

Solução alternativa: outra forma de resolver o problema é computar a distância de edição da string S com o seu reverso e dividir o resultado por dois.

Tutorial: Kaskata

Alberto Neto

Seja b_t um vetor tal que $b_t[i]$ seja a quantidade de bolas na posição i no tempo t. Queremos encontrar b_m. Podemos montar uma matriz R da seguinte forma: na linha i da matriz temos 1 nas colunas $i-1$ e $i+1$, e 0 nas outras. É fato que $R * b_t = b_{t+1}$, onde $*$ é o produto usual de matrizes. A prova dessa afirmação fica de exercício ao leitor.

A resposta é precisamente $b_0 + b_1 + \cdots + b_m$. Isso é o mesmo que calcular $(R^0 + R^1 + \cdots + R^m) * b_0$, então basta encontrarmos a soma de potências de R.

Para calcular a soma das potências de R, notemos o seguinte. Se $S = R^0 + \cdots + R^{k-1}$ então $S + R^k * S = R^0 + \ldots R^{2k-1}$. Calculando estas somas para potências de 2 podemos calcular a soma desejada para m, basta olharmos para a expansão de m na base binária (de forma análoga ao algoritmo de exponenciação rápida).

Tutorial: Labuta Diária

Daniel Saad Nogueira Nunes

Esse problema é uma versão do problema do ancestral comum mais baixo (Lowest Common Ancestor) ou, simplesmente, LCA.

Para resolvê-lo, podemos seguir os passos:

1. Faça um passeio Euleriano na árvore, armazenando o número de visitação, a altura, bem como a primeira vez que cada nó foi visitado no passeio. Compute também a distância da raiz a cada nó, o que pode ser feito simplesmente pegando a distância da raiz até o pai e somando com o custo da aresta do pai para o nó.

2. Crie uma árvore de segmentos sobre os nós visitados durante o passeio.

Para cada pergunta envolvendo dois nós u e v, basta:

1. Responda a consulta de mínimo (RMQ) usando a árvore de segmentos com base na primeira ocorrência dos nós u e v no passeio para achar o nó com menor numeração (ancestral comum mais baixo). Chame esse nó de w. Caso u seja ancestral de v ou vice-versa, basta escolher o de menor altura.

2. Dê como resposta a distância da raiz até u, somada a distância da raiz até v e descontar o resultado do dobro da distância da raiz até w.

Complexidade: para construir a árvore de segmentos, leva-se tempo $\Theta(n)$. Cada pergunta pode ser respondida em $\Theta(\lg n)$.