Tutorial: Iluminação Pública

Edson Alves da Costa Júnior

Este problema pode ser resolvido por meio de um algoritmo guloso. Basta ordenar os intervalos $[L_i, R_i]$ que representam os alcances dos postes. Agora, começando na posição 1, basta escolher o intervalo com $L_i \leq 1$ com maior valor R_i, o que garantirá a iluminação da posição 1 e do maior número possível de casas à esquerda desta posição. Daí basta prosseguir com a mesma estratégia gulosa a partir da posição $R_i + 1$.

A complexidade desta solução é $O(N \log N)$, por conta da ordenação dos intervalos.

Tutorial: Colonização

Vinicius Ruela Pereira Borges

Uma maneira de resolver esse problema é por meio de uma fila de prioridade mínima, utilizando-a para simular a chegada das expedições na Quadradonia e como as reservas de ouro são exploradas pelas expedições e recuperadas pela natureza.

Como devemos tratar os casos de empate na chegada dos navios à Quadradonia, podemos definir uma estrutura do tipo pair ¡ano da expedição, navio¿ para a fila de prioridade e assim poder simular o problema. Os limites do problema indicam que o ritmo de exploração sempre é maior do que a recuperação de ouro realizada pela natureza.

Inicialmente, deve-se inserir na fila de prioridade os pares ¡0,1¿, ¡0,2¿, ¡0,3¿ ... ¡0,N¿. A simulação do processo ocorre como se segue: desenfileira o par ¡anos,id¿ (representa o navio chegando à Quadradonia), retira M toneladas de ouro e verifica se as reservas de ouro foram zeradas. Caso existam reservas de ouro, deve-se enfileirar o par ¡anos+t_1,id¿ na fila de prioridade. Senão, deve-se encerrar a simulação e imprimir como resposta a variável `anos`.

Tutorial: Sombra

Vinicius Ruela Pereira Borges

Para resolver o problema, deve-se considerar apenas os momentos em que Danilo Saad sai de um quilômetro arborizado para um quilômetro não-arborizado da ciclovia, pois fica mais fácil computar a quantidade de trechos não-arborizados considerando essa mudança. Para isso, podemos utilizar uma variável booleana `ehSombra` que indique se Saad está ou não em uma região não-arborizada.

Como pode existir mais de uma árvore por quilômetro, pode-se gerar um vetor que represente a ciclovia, em que cada posição i está associada a um quilômetro da ciclovia que indica se o quilômetro i existe sombra ou não.

Seja `trechos` a variável que descreve a quantidade de trechos não-arborizados da ciclovia. Pedalando quilômetro por quilômetro da ciclovia, se Danilo Saad passa de um quilômetro arborizado para um quilômetro não-arborizado, marca-se `ehSombra = false` e incrementa-se a variável `trechos`. Saad continua sua pedalada e caso o próximo quilômetro seja não-arborizado, não se deve fazer nada (vale ressaltar que ainda o trecho não-arborizado é o mesmo). Caso contrário, deve-se alterar a variável `ehSombra = true`, pois o trecho não-arborizado terminou. Deve-se fazer esse processo por toda a extensão da ciclovia.

Ao final deve-se imprimir o número armazenado em `trechos`.

Tutorial: Índice-h

Daniel Saad Nogueira Nunes

Este é um problema que envolve ordenação e pode ser resolvido de maneira eficiente com uma variação do *CountingSort*.

Primeiramente, deve-se calcular o índice-h de cada autor. Para isto, basta verificar quantas publicações com i citações ele tem. Isso pode ser feito com um vetor de tamanho $P + 1$ em que P é o número de publicações, visto que o índice-h de um autor não pode ser maior que P, da seguinte forma:

```
for(int i=0;i<=P;i++)
    count[min(P, citacoes[i])]++;
}
```

Obviamente, este processo leva tempo $\Theta(P)$.

Dado que `count[i]` te dá o número de publicações com i citações, o índice-h pode ser calculado inicializando um acumulador com `count[P]` e varrendo este vetor da direita para esquerda. Sempre que o acumulador for maior ou igual ao índice i do vetor, então você tem que o índice-h é i, caso contrário, adiciona-se ao acumulador o valor `count[i]` e o índice i é decrementado, conforme pseudocódigo abaixo.

```
for (int i = P; i >= 0; i--) {
    sum += count[i];
    if (sum >= i) {
        return i;
    }
}
```

Com o índice-h calculado para cada autor, basta agora ordená-los levando em consideração, primeiramente o índice-h, e depois o nome do autor em ordem lexicográfica.

Isso nos dá uma complexidade total de $\Theta(NP + N \lg N)$. O primeiro terno se refere ao cálculo do índice-h para cada autor e o segundo termo se refere à ordenação dos N autores.

Tutorial: Processos

Vinicius Ruela Pereira Borges e Edson Alves da Costa Junior

O problema pode ser interpretado como cada mesa sendo uma pilha, pois os processos são colocados um sobre o outro, logo, a retirada dos processos de ambas as mesas deve obedecer a política: o último processo que entra na mesa é o primeiro a sair. Nesse sentido, queremos basicamente ordenar a pilha (mesa A) com o auxílio de outra pilha (mesa B), em que os processos ordenados fiquem na mesa A.

A solução consiste em: na leitura da entrada, empilhe todos os N processos na mesa A. O procedimento de ordenação dos processos de seu Anacleto pode durar no máximo N ciclos. Portanto, considere $k = 0, 1, 2, ..., N-1$. De maneira geral, no k-ésimo ciclo, desempilhe o processo p_i da pilha A e assuma-o como sendo o maior. Em seguida, desempilhe o processo p_k e verifique se $p_k > p_i$. Se for verdadeiro, empilhe p_i na pilha B e mantenha p_k como o maior. Caso contrário, empilhe p_k diretamente na mesa B e mantenha p_i como maior. Faça isso até que a mesa A fique com $N - k$ processos. Por fim, empilhe o processo de maior identificador encontrado na pilha A e retorne todos os outros processos da pilha B para a pilha A (desempilhe um processo da pilha B e empilhe na pilha A). O ciclo termina e por isso, incremente k. Se a pilha A estiver ordenada, deve-se finalizar o procedimento, pois o problema pede que seja determinada a quantidade **mínima** de ciclos do procedimento de Seu Anacleto.

Como verificar se os processos estarão ordenados na pilha A? Em cada ciclo, no momento em que os processos forem retornados da pilha B para a pilha A, basta verificar se cada processo i recém-tirado da pilha B possui p_i maior do que o p_j referente ao processo j que está no topo da pilha B. Se todas essas movimentações forem respeitadas, a pilha estará ordenada.

Tutorial: Mineirês

Daniel Saad Nogueira Nunes

Uma maneira fácil de resolver este problema é construir um dicionário utilizando uma estrutura de mapeamento que converte uma string em outra string (com espaços, possivelmente). Em `C++`, a estrutura `map<string,string>` poderia ser utilizada como dicionário.

O dicionário pode ser montado da seguinte forma: cada string do mineirês lida teria como tradução seria mapeada em uma sequência de strings do português.

Com o dicionário montado, basta ler cada string do texto original e, se ela corresponder a uma string do mineirês, você imprime a tradução desta string.

Em pseudocódigo teríamos algo mais ou menos assim:

1. Construa o dicionário utilizando a descrição acima.
2. Para cada palavra do texto,verifique se ela ocorra no dicionário.
   (a) Caso ela ocorra, substitua pela tradução em português.
   (b) Caso contrário, deixe ela como está.

Tutorial: Sem Número

Guilherme Novaes Ramos

Uma solução simples é armazenar os N valores, ordená-los e verificar qual o índice que não corresponde ao valor armazenado. Supondo um algoritmo eficiente de ordenação, esta abordagem tem complexidade $O(n \log n)$.

Uma solução mais elegante é considerar que todos os números são distintos e na sequência de 0 a N, formando uma Progressão Aritmética de razão 1 cuja soma dos elementos é facilmente calculada. A diferença deste valor para a soma dos inteiros apresentados na entrada será o número que Polly pulou. Esta solução tem complexidade $O(1)$.

Tutorial: Ao Resgate!

Daniel Saad Nogueira Nunes

Para resolver este problema, é possível efetuar uma simulação baseando-se nas regras do enunciado.

Suponha que tenhamos uma variável chamada *total* que dê o tempo decorrido desde o início dos atendimentos.

1. Partindo do quilômetro referente ao último atendimento, verifique se existem solicitações pendentes (aquelas com tempo menor que *total*) e vá para aquela mais próxima em termos de distância. Em caso de empate, Vá para aquela com maior quilômetro. Adicione k a *total* em que k é a distância em quilômetros percorrida.

2. Caso não exista solicitações pendentes (todas as solicitações tenham tempo maior que *total*), o motorista aguarda um tempo t equivalente à diferença de *total* até a(s) próxima(s) solicitação(ões) com menor tempo. Então utiliza-se as mesmas comparações efetuadas pra o primeiro caso e adiciona-se a total o tempo $t + k$.

3. Repita os passos anteriores até que Guilherme seja atendido.

4. O tempo total de espera de Guilherme está em *total*.

O procedimento para após Guilherme ter sido atendido. O custo total desta abordagem simples é $\Theta(N^2)$. Utilizando-se estruturas mais sofisticadas é possível atingir uma complexidade menor.

Desejamos mais sorte ao nosso amigo da próxima vez e que parem de fazer problemas com o nome dele.

Tutorial: Queimada Brasileira

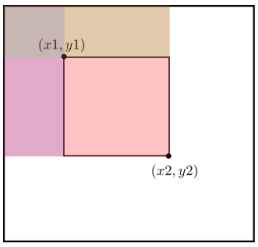
Daniel Saad Nogueira Nunes

Este problema pode ser resolvido através de uma técnica chamada soma de prefixos $2D$.

Suponha que tenhamos computado em uma matriz auxiliar $S[i][j]$ o número de áreas queimadas considerando o retângulo com extremidade superior esquerda em $(1, 1)$ e extremidade inferior direita em (i, j).

Dado S, a área desmatada de qualquer retângulo com extremidade superior esquerda em (x_1, y_1) e extremidade inferior direita em (x_2, y_2) pode ser calculada através de 4 operações: $S[x_2, y_2] - S[x_1 - 1, y_2] - S[x_2, y_1 - 1] + S[x_1 - 1, y_1 - 1]$.

Isso pode ser representado graficamente com a figura abaixo.



A área de interesse é a do retângulo rosa. Podemos obter a área rosa ao pegarmos a área total e descontarmos os retângulos lilás, marrom e dourado. $S[x_2, y_2]$ representa a soma dos quatro retângulos, isto é, a soma total da origem $(1, 1)$ até (x_2, y_2). $S[x_1 - 1, y_2]$ corresponde às áreas dos retângulos marrom e dourado. $S[x_2, y_1 - 1]$ corresponde às áreas dos retângulos lilás e marrom. Como a área marrom foi descartada duas vezes por $S[x_1 - 1, y_2]$ e $S[x_2, y_1 - 1]$, adicionamos a área do retângulo marrom com $S[x_1 - 1, y_1 - 1]$.

O único problema que resta é calcular a matriz S, mas isso pode ser feito de maneira similar.

1. Inicializamos a linha 0 e coluna 0 de S com zeros.
2. Cada elemento $S[i][j]$ pode ser calculado como $S[i-1][j] + S[i][j-1] - S[i-1][j-1] + M[i][j]$, em que $M[i][j]$ é a matriz original do problema.

O cálculo de cada elemento de S leva portanto tempo constante. Logo, para computar S, precisamos de $\Theta(N^2)$ passos.

A complexidade total da solução é $\Theta(N^2K)$, visto que cada consulta pode ser respondida em tempo constante com a presença de S.

Tutorial: Eleições

Edson Alves da Costa Júnior

Para N eleitores há $k = N/A$ conjuntos completos com A eleitores. Como cada grupo completo resulta em B votos para Chapa I serão, no mínimo, kB votos. Esta solução tem complexidade $O(1)$.

Tutorial: Extrato Bancário

Daniel Saad Nogueira Nunes

A técnica para resolução deste problema é simples. Basta inicializar quatro acumuladores V_a, V_d, V_g e V_p, para cada categoria (alimentação, despesas domésticas, gastos gerais, e transporte). Para cada linha da entrada com um valor X_i e uma string C_i, compare a string C_i com cada uma das categorias. Caso as duas strings sejam iguais, adicione ao acumulador correspondente à categoria C_i, o valor de X_i.

Ao final, basta imprimir cada categoria em ordem alfabética com o valor do acumulador correspondente e a porcentagem, que pode ser calculada como o valor do acumulador daquela categoria dividido pela soma dos quatro acumuladores.

Tutorial: Dardos

Edson Alves da Costa Júnior

Verificar todos os N alvos para cada um dos M disparos levaria a uma solução $O(NM)$, que resultaria num veredito TLE.

Para diminuir a complexidade da solução, é preciso pré-computar a distância máxima de cada círculo ao centro (isto é, r_i^2). Assim, para cada disparo, é preciso localizar, por meio de uma busca binária, o ponto maior ou igual à distância $d^2 = x^2 + y^2$ e totalizar a pontuação referente à resposta, quando for o caso.

Esta solução verifica cada disparo em $O(\log N)$, levando a uma solução $O(N + M \log N)$.

Tutorial: Olá Novo Mundo

Guilherme Novaes Ramos

A solução é simples, basta reconhecer a nacionalidade de um marinheiro e apresentar a saudação associada. É preciso registrar quais já foram ditas, de modo que não haja repetição se houver mais de um marinheiro do mesmo país de origem. Esta abordagem tem complexidade O(n).