

Tutorial: Alpinismo

Daniel Saad Nogueira Nunes

Uma forma de resolver esse problema é adicionar duas montanhas fictícias com valor -1 ao início e ao final. Então a entrada modificada seria $V' = \langle -1, v_1, \dots, v_n, -1 \rangle$. Em seguida, basta comparar, para cada $1 \leq i \leq n$, se $V'[i] > V'[i-1]$ e $V'[i] < V'[i+1]$ para saber se $V'[i]$ é um pico ou não.

Tutorial: Bodybuilder

Caio Lauro de Lima

Para resolver esse problema, basta realizar a leitura dos dois números, multiplicá-los e dar o resultado da multiplicação como resposta.

Tutorial: Criptografia

Daniel Saad Nogueira Nunes

Uma forma de resolver esse problema é computar o crivo de Erastóstenes para todos os números até 10^7 , mas de uma maneira diferente. O valor $F[k]$ conterá o número de divisores naturais de k . Assim, o número será semiprimo se e somente se $F[k] = 2$.

```
1 for  $i \leftarrow 0$  to  $10^7$  do  $F[i] \leftarrow 0$ 
2 for  $i \leftarrow 2$  to  $10^7$  do
3   if  $F[i] = 0$  then
4      $j \leftarrow i$ 
5     while  $j \leq 10^7$  do
6        $x \leftarrow j$ 
7       while  $x \bmod i = 0$  do
8          $x \leftarrow x/i$ 
9          $F[j] \leftarrow F[j] + 1$ 
10       $j \leftarrow j + i$ 
```

Em seguida, computamos um vetor $P[i] = 1$ se e somente se $F[i] = 2$.

```
1 for  $i \leftarrow 0$  to  $10^7$  do
2   if  $F[i] = 2$  then
3      $P[i] \leftarrow 1$ 
4   else
5      $P[i] \leftarrow 0$ 
```

E agora transformamos P em um vetor de soma de prefixos. Em outras palavras: $P[i]$ conterá o total de semiprimos no intervalo $[0, i]$.

```
1 for  $i \leftarrow 2$  to  $10^7$  do
2    $P[i] \leftarrow P[i] + P[i - 1]$ 
```

Todo esse processo é realizado apenas uma única vez. Para responder uma consulta sobre um intervalo $[l, r]$, basta fazer $P[r] - P[l - 1]$, que é o número de semiprimos no intervalo $[0, r]$ menos o número de semiprimos no intervalo $[0, l - 1]$, ou, em outras palavras, o número de semiprimos no intervalo $[l, r]$.

Tutorial: Dança das Cadeiras

Igor de Oliveira Moura

Uma das formas de resolver este problema é por meio de uma simulação com dois laços aninhados: o primeiro percorre as K batidas e o segundo percorre os N alunos.

A cada batida, todos os alunos se movem uma posição para a direita, e o aluno da última posição volta para a primeira, pois as cadeiras estão em círculo.

Para fazer isso, no início de cada batida guardamos em uma variável o aluno da última posição, já que ele seria sobrescrito no deslocamento. Em seguida, percorremos o vetor de trás para frente, da posição $N - 1$ até 1, copiando o valor da posição anterior para a atual, ou seja, $v[j] \leftarrow v[j - 1]$. Esse sentido é necessário: se percorrêssemos da frente para trás, o valor de $v[0]$ seria copiado para todas as posições. Por fim, colocamos em $v[0]$ o aluno que havia sido guardado.

Depois das K batidas, imprimimos o vetor resultante.

Tutorial: Estratégia de Pneus

Igor de Oliveira Moura

A solução utiliza uma abordagem gulosa, escolhendo os pneus de acordo com a prioridade H , M e S . Enquanto restarem voltas ($n > 0$), se o número de voltas restantes for suficiente para utilizar o pneu duro ($n \geq h$), escolhemos esse pneu. Caso contrário ($n < h$), verificamos se podemos utilizar o pneu médio (M), ou seja, se $n \geq m$ e $n \geq s$. Se nenhuma dessas condições for satisfeita, utilizamos o pneu macio (S), que pode ser usado apenas parcialmente na última escolha. A cada escolha, subtraímos de n a durabilidade do pneu correspondente e armazenamos o tipo de pneu em um vetor. O processo termina quando $n \leq 0$, e imprimimos a quantidade de pneus utilizados e a sequência escolhida. No pior caso, quando a durabilidade do pneu macio é 1, o laço executa $O(n)$ vezes, portanto a complexidade de tempo e de espaço é $O(n)$.

Tutorial: Foos Contra Fighters

Caio Lauro de Lima

Cada *Foo* corresponde a uma ocorrência da sequência `foo` em S , isto é, a três caracteres consecutivos iguais a `f`, `o` e `o`, nessa ordem. Então, o problema se resume a contar em quantas posições da string isso acontece.

Para isso, percorremos as posições i de S em que ainda cabem três caracteres (ou seja, de 0 até $n - 3$, incluso). Em cada posição, verificamos se $S[i] = \text{f}$, $S[i + 1] = \text{o}$ e $S[i + 2] = \text{o}$, e, quando isso acontece, somamos um ao contador. Ao final, imprimimos o valor do contador, que é o número de potes necessários.

Tutorial: Guerra nas Estrelas

Arthur Andrade D'Oliveira

Como a resposta é sempre a mesma, independentemente da palavra recebida, basta imprimir **Que a Força esteja com você!** seguida de uma quebra de linha. A palavra da entrada pode ser ignorada.

A complexidade de tempo e o uso de memória são $O(1)$, pois o tamanho da mensagem é constante.

Tutorial: Hertz pra todos

Igor de Oliveira Moura

Solução

Primeiro, precisamos descobrir quais residências já recebem sinal das torres existentes. Para isso, percorremos cada residência e verificamos se ela está dentro do raio de cobertura de alguma torre. Uma residência na posição (x_i, y_i) é atendida por uma torre na posição (a_j, b_j) , de raio r_j , quando

$$(x_i - a_j)^2 + (y_i - b_j)^2 \leq r_j^2.$$

Guardamos a quantidade de residências já atendidas e armazenamos as posições daquelas que ainda estão sem sinal. Se todas as residências já possuem cobertura, basta imprimir -1 , pois não é necessário instalar uma nova torre.

Agora, precisamos encontrar a melhor posição para a nova torre. Uma primeira ideia seria testar todas as posições (x, y) possíveis e contar quantas residências sem sinal seriam atendidas. Porém, como existem 1001^2 posições possíveis, verificar todas as residências em cada uma delas seria muito custoso.

Para resolver isso de forma mais eficiente, podemos utilizar um **vetor de diferenças**.

Encontrando os intervalos de cobertura

A ideia é fixar uma coordenada x para a nova torre e descobrir quais valores de y permitem atender cada residência.

Considere uma residência na posição (c_x, c_y) . Para que ela seja atendida pela nova torre, de raio R , precisamos que

$$(x - c_x)^2 + (y - c_y)^2 \leq R^2.$$

Como a coordenada x está fixa, podemos reorganizar a expressão:

$$(y - c_y)^2 \leq R^2 - (x - c_x)^2.$$

Primeiro, calculamos a distância horizontal entre a torre e a residência:

$$d_x = |x - c_x|.$$

Se $d_x > R$, a residência não pode ser atendida por nenhuma torre nessa coordenada x , independentemente do valor de y .

Caso contrário, podemos calcular a maior distância vertical permitida:

$$d_y = \left\lfloor \sqrt{R^2 - d_x^2} \right\rfloor.$$

Assim, qualquer posição da torre cuja coordenada y esteja no intervalo

$$[c_y - d_y, c_y + d_y]$$

consegue atender essa residência.

Como a nova torre deve estar dentro dos limites do plano, ajustamos o intervalo para

$$y_{\min} = \max(0, c_y - d_y)$$

e

$$y_{\max} = \min(1000, c_y + d_y).$$

Portanto, para cada residência sem sinal, encontramos um intervalo de posições em que a nova torre consegue atendê-la, considerando a coordenada x que estamos testando.

Utilizando o vetor de diferenças

Depois de encontrar esses intervalos, precisamos descobrir quantas residências seriam atendidas em cada coordenada y .

Poderíamos percorrer cada intervalo e incrementar todas as posições dentro dele, mas isso ainda exigiria muitas operações.

Em vez disso, utilizamos dois vetores, chamados **entra** e **sai**, inicialmente preenchidos com zero.

Para cada intervalo $[y_{\min}, y_{\max}]$, fazemos:

- Incrementamos **entra** $[y_{\min}]$, indicando que uma residência começa a ser atendida nessa posição.
- Incrementamos **sai** $[y_{\max} + 1]$, indicando que essa residência deixa de ser atendida a partir dessa posição.

Após processar todas as residências sem sinal, percorremos as coordenadas y de 0 até 1000, mantendo uma variável **quantidade**:

$$\text{quantidade} += \text{entra}[y] - \text{sai}[y].$$

Dessa forma, conseguimos saber quantas residências seriam atendidas pela nova torre em cada posição (x, y) , sem precisar verificar novamente a distância até cada residência.

Sempre que encontramos uma quantidade maior que a melhor registrada, atualizamos a resposta e guardamos as coordenadas da torre.

Repetimos esse processo para todos os valores de x entre 0 e 1000. Ao final, somamos a maior quantidade de novas residências atendidas com a quantidade que já recebia sinal das torres existentes.

Otimização

Como o raio R da nova torre é sempre o mesmo, podemos calcular antecipadamente os valores

$$\text{alcance}[d_x] = \left\lfloor \sqrt{R^2 - d_x^2} \right\rfloor$$

para todo $0 \leq d_x \leq R$.

Assim, durante a varredura, basta consultar o valor já calculado para descobrir o alcance vertical de cada residência, evitando calcular a raiz quadrada repetidamente.

Tutorial: Indo a Uberlândia

Daniel Saad Nogueira Nunes

Solução

Para resolver o problema, precisamos calcular quantos pedágios serão encontrados durante o percurso e, em seguida, determinar o valor total a ser pago.

Sabemos que existe um pedágio a cada x quilômetros e que o percurso possui y quilômetros. Assim, podemos calcular inicialmente a quantidade de pedágios utilizando a divisão inteira:

$$\text{pedagios} = \left\lfloor \frac{y}{x} \right\rfloor.$$

Porém, precisamos tomar um cuidado: se y for múltiplo de x , o último pedágio estaria exatamente no destino. Nesse caso, ele não deve ser considerado.

Para verificar essa situação, utilizamos o resto da divisão. Se $y \bmod x = 0$, subtraímos um da quantidade de pedágios.

Depois disso, basta multiplicar a quantidade encontrada pelo valor de cada pedágio, que é de R\$ 7,50:

$$\text{custo} = \text{pedagios} \times 7,50.$$

Por fim, imprimimos o valor total com duas casas decimais, conforme o formato solicitado.

Tutorial: Juntando os Dígitos

Gabriel César Silvino Xavier

Uma forma de resolver o problema é substituir todos os caracteres x por 1, mas reservar o último para ajustar a soma do algoritmo de Luhn. Calculamos a contribuição dos demais dígitos, percorrendo a sequência da direita para a esquerda e aplicando as regras do algoritmo.

Se a soma for S , a contribuição necessária na posição reservada é $r = (10 - (S \bmod 10)) \bmod 10$. Caso $r = 0$, a posição reservada precisaria receber 0, mas o problema só permite dígitos de 1 a 9. Para evitar isso, basta mudar o penúltimo x de 1 para 2. Isso aumenta a soma em 2 se essa posição for dobrada, ou em 1 caso contrário. Assim, o novo valor de r passa a ser 8 ou 9, respectivamente.

Para preencher o último x , usamos r diretamente se sua posição não for dobrada. Nas posições dobradas, os dígitos de 1 a 4 contribuem com 2, 4, 6 e 8, enquanto os de 5 a 9 contribuem com 1, 3, 5, 7 e 9, respectivamente. Basta escolher o dígito cuja contribuição é r . Assim, obtemos uma sequência válida em $O(n)$ de tempo e espaço.

Tutorial: Kernel

Gabriel César Silvino Xavier

Uma forma de resolver o problema é usar quatro laços aninhados. Os dois primeiros percorrem as posições (i, j) em que o kernel cabe na imagem, de 0 a $H - K$ nas linhas e de 0 a $W - K$ nas colunas. Para cada posição, iniciamos uma soma em zero. Os outros dois laços percorrem os elementos (a, b) do kernel e acrescentam `imagem[i + a][j + b] · kernel[a][b]` à soma.

Depois, imprimimos o valor calculado e seguimos para a próxima posição. A complexidade é $O(HWK^2)$ em tempo e $O(HW)$ em espaço.

Tutorial: Leitura de Relatórios

Daniel Saad Nogueira Nunes

Uma forma de resolver esse problema é procurar a posição do primeiro caractere que não é 0 e imprimir a partir dele. Caso não seja encontrado algum caractere diferente de zero na sequência, basta imprimir um zero.

Tutorial: Mentalista

Arthur Andrade D'Oliveira

Como todos os horários pertencem ao mesmo dia, podemos converter cada horário em minutos. Para um horário com h horas e m minutos, o valor é $60h + m$. Por exemplo, 14:20 corresponde a $14 \cdot 60 + 20 = 860$ minutos.

Seja T o horário do roubo, E o horário de entrada de uma pessoa e S o horário de saída, todos convertidos em minutos. Essa pessoa estava no local se, e somente se,

$$E \leq T \leq S.$$

As duas comparações incluem igualdade, pois alguém que entrou ou saiu exatamente no horário do roubo também é suspeito.

Leia os registros e verifique essa condição para cada pessoa. Quando ela for satisfeita, imprima o nome. Se ninguém satisfizer a condição, nenhum nome será impresso e a saída ficará vazia, conforme solicitado.

A verificação é correta porque o intervalo de presença de cada pessoa contém exatamente os instantes entre sua entrada e sua saída, incluindo os extremos. A condição acima seleciona todas e somente as pessoas cujo intervalo contém o instante do roubo. Os registros são independentes, e todos são examinados.

Não é necessário ordenar os nomes: a saída aceita qualquer ordem, inclusive a ordem da entrada usada pela solução de referência. Como os nomes na entrada são únicos, cada suspeito será impresso uma única vez.

A complexidade de tempo é $O(N)$ e o espaço auxiliar é $O(1)$, pois cada registro pode ser processado e descartado após a leitura. Os nomes têm comprimento limitado a 20 letras.