Tutorial: Azulejos

Vinicius Borges

Este problema pode ser resolvido ao computarmos o número de linhas que começam com azulejos verdes e brancos (`linha_verde` e `linha_branca`) e o número de colunas que começam com azulejos verdes e brancos (`coluna_verde` e `coluna_branca`). A resposta final é `linha_verde * coluna_verde + linha_branca * coluna_branca`. Inteiros de 64-bits devem ser utilizados para armazenar a resposta final e evitar overflow.

O número de linhas e colunas, verdes ou brancas, pode ser calculado em tempo $O(1)$ devido à disposição dos azulejos.

Tutorial: Bolo de FooBar

Guilherme Ramos

A solução é simples, basta ler os números `foo` e `bar` e, enquanto houver números a serem lidos, ler e processar o inteiro n. Se for divisível por `foo`, `bar` ou ambos, apresentar "foo", "bar" ou "foobar", respectivamente. Caso não seja múltiplo, nada se faz.

Tutorial: Ceborinha

Guilherme Ramos

A solução é simples, basta verificar a existência da letra 'R' (ou 'r') na linha de entrada. Se houver, a mensagem é falsa!

Tutorial: Discussão de Natal

Jeremias Gomes

A solução utiliza a ideia principal de travessia em árvores. Dessa forma uma travessia pre-order com a modificação de registrar o primeiro enfeite visitado em cada um dos níveis da árvore irá indicar quais destes podem ser vistos.

A solução para a visão da família fica:

```
visao_da_familia(arvore, nivel):
    if nivel > nivel_global:
        solucao_familia.insere(arvore.indice)
        nivel_global += 1
    if arvore.esquerda.indice != -1:
        visao_da_familia(arvore.esquerda, nivel + 1)
    if arvore.direita.indice != -1:
        visao_da_familia(arvore.direita, nivel + 1)

solucao_familia = []
nivel_global = -1
visao_da_familia(arvore, 0)
```

E a solução para a visão de Djerry fica:

```
visao_de_djerry(arvore, nivel):
    if nivel > nivel_global:
        solucao_djerry.insere(arvore.indice)
        nivel_global += 1
    if arvore.direita.indice != -1:
        visao_de_djerry(arvore.direita, nivel + 1)
    if arvore.esquerda.indice != -1:
        visao_de_djerry(arvore.esquerda, nivel + 1)

solucao_djerry = []
nivel_global = -1
visao_de_djerry(arvore, 0)
```

Tutorial: Empreitada Perigosa

Vinicius Borges

Para resolver o problema, pode-se fazer uma busca completa por meio de uma abordagem profunda (depth first search - DFS). A ideia é testar todas as possibilidades de alimentação ou não em cada dia da jornada de Blayton. Vale lembrar que os limites do problema são $D \leq 30$ e $N \leq 30$, então DFS pode ser suficiente.

A complexidade da DFS para resolução desse problema $O(2^N)$. Uma abordagem por programação dinâmica pode ser ainda mais eficiente ao resolver o problema em $O(N \times M)$ no tempo e na memória.

Tutorial: Fantástica Fábrica de Fibonaccis

Daniel Saad

Este problema é uma aplicação direta do teorema de Zeckendorf, que diz que "todo número de natural pode ser expresso como uma soma de números de Fibonacci não consecutivos".

Uma forma de resolver este problema é utilizar um algoritmo guloso. Primeiramente, processam-se todos os números de Fibonacci menores que 10^{18}. Em seguida varre-se esta lista, em ordem decrescente, e para cada número, incluímos ele em uma variável acumuladora até atingir o valor dado pela consulta. O único cuidado é não tomar dois números consecutivos.

A complexidade da solução é $\Theta(n \log_\phi q_i)$.

Tutorial: Gato na Caixa

Alberto Tavares

Tutorial: Hahtlehtihcoh

Daniel Saad

Através de uma busca completa é possível enumerar todas as possibilidades. Ao utilizar um vetor de bits, de tamanho igual ao número de vogais, é possível enumerar todos os subconjuntos de posições em que as vogais estão presentes e inserir um "h" após cada posição ligada deste vetor. A complexidade da solução é $\Theta(2^n)$, sendo n o número de vogais.

Tutorial: Iluminação Pública

Edson Alves

Tutorial: Jardinagem

Alberto Tavares

Tutorial: Kaindo Raio

Alberto Tavares

Tutorial: Lógica do Talvez

Edson Alves

Tutorial: Make

Daniel Saad

Este problema pode ser resolvido através de um procedimento de ordenação topológica em grafos, que também detecta se há ciclos. Caso haja ciclos, a mensagem "`impossível`" deve ser impressa. Caso contrário, a ordem dada pela ordenação topológica deve ser impressa. A ordenação topológica pode ser executada em tempo $\Theta(|V| + |E|)$.