Tutorial: Alice, Bob e Charlie

Edson Alves

Tutorial: Bozo Boboca

Guilherme Ramos

Há diversas soluções possíveis. Após ler a dica, cada iteração pode ser feita em dois passos. No primeiro, leia a fala, identifique as palavras distintas, e junte as letras iniciais de cada dessas palavras. No segundo, crie um índice d para o primeiro caractere da dica e avance sobre cada caractere das iniciais com um índice i - quando o d-ésimo caractere da dica for igual ao i-ésimo caractere das iniciais, incremente d. Ao final do processo, se d for maior ou igual ao tamanho da dica, todos os caracteres foram encontrados na ordem correta e, portanto a fala é uma lorota. Caso contrário, não há falsidade nela.

Para agilizar, a sequência de iniciais deve conter apenas os caracteres existentes na dica.

Tutorial: Chega de Palíndromos!

Edson Alves

Tutorial: Decifrando Relatórios

Daniel Saad

Para resolver este problema, basta realizar busca exaustiva através de um backtracking. Para cada letra, assumimos um valor de 0 a 9. Ao final do processo, verificamos se os números mapeados satisfazem a soma além de guardar a melhor solução possível.

Tutorial: É Melhor Chamar o Celso

Jeremias Gomes

Dada uma determinada redução, é possível verificar se esta passa nas inspeções, da seguinte forma:

```
preco_ajustado = preco_anterior - reducao
if preco_ajustado == 0:
    return False

while preco_ajustado <= limite:
    preco_ajustado = preco_ajustado * 2
    inspecoes_restantes = inspecoes_restantes - 1

if inspecoes < 1:
    return True
return False
```

Assim, dado o conjunto de possíveis reajustes no preço $[0..P-1]$, a verificação citada utilizando esse conjunto irá resultar em algo que pode ser visualizado da seguinte forma (utilizando o caso de teste único do exemplo 1):

0	1	2	3	4	5	6	**7**	8	9
False	False	False	False	False	False	False	**True**	True	True

Essa estrutura permite a realização de uma busca binária, utilizando a verificação mostrada anteriormente, para encontrar o mínimo reajuste que possibilite passar em todas as inspeções. A complexidade dessa solução é $O(n * lg(n))$.

Tutorial: Faraó e a Pirâmide

Edson Alves

O número total de tijolos $T(N)$ é o N-ésimo número triangular, isto é,

$$T(N) = \frac{N(N+1)}{2}$$

O número de estratégias corresponde ao número de divisores $\sigma(n)$ de $T(N)$. Como $T(N) = O(N^2)$, tentar computar os divisores de $T(N)$, no pior caso, excede o limite de tempo do problema. Observe, contudo, que N e $N+1$ são primos entre si, isto é, não possuem divisores em comum. Como a função $\sigma(n)$ é multiplicativa, o número de estratégias será dado por

$$\sigma(T(N)) = \sigma(N)\sigma\left(\frac{N+1}{2}\right),$$

se N é ímpar, ou

$$\sigma(T(N)) = \sigma(N+1)\sigma\left(\frac{N}{2}\right),$$

se N é par. Se $\sigma(n)$ for computado em $O(\sqrt{n})$, a solução tem complexidade $O(\sqrt{N})$.

Tutorial: Gastar Dinheiro? Não Quero!

Tiago Fernandes

Tutorial: Hound 6

Jeremias Gomes

A solução utiliza a ideia principal de calcular o coeficiente angular de todas as coordenadas dadas. Cada coeficiente sendo utilizado como a chave para uma tabela de hash que irá conter as coordenadas dos pontos que constituíram aquele declive. Dessa forma, sempre que dois pontos (coordenadas) resultarem no mesmo declive, estes serão adicionados aos valores daquela chave.

Para evitar que os valores dos coefientes estejam em ponto flutuante, pode-se utilizar os coeficientes da equação da reta como uma tupla (a, b, c), a partir da seguinte equação:

$$a * x + by + c = 0$$

Assim, dados dois pontos P_1 e P_2, se os pontos estiverem na horizontal, os valores ficam: $a = 1$, $b = 0$ e $c = -x_1$. Caso contrário, estes podem ser deduzidos a partir do seguinte:

$$a * x_1 + by_1 + c = 0$$

$$a * x_1 + by_1 + c = 0$$

que resultam em:

$$a = y_2 - y_1$$

$$b = x_1 - x_2$$

$$c = x_2 * y_1 - x_1 * y_2$$

Dados esses valores, deve-se levar em consideração, ainda, duas possibilidades: (1) o valor de a ser sempre positivo, pois as retas $-a * x + -by - c = 0$ e $a * x + by + c = 0$ tem a mesma inclinação; e (2) o caso mais geral em que $-ka * x + -kby - kc = 0$ também é o mesmo que $a * x + by + c = 0$. Para o segundo caso, pode-se dividir a tupla pelo Máximo Divisor Comum dos mesmos.

Uma última otimização envolvendo o uso de memória é que, ao invés de inserir todas as inclinações de todos os pares incluindo todos eles de uma única vez, esse processo pode ser realizado para cada coordenada em relação as demais, onde é salvo, ao término da iteração, somente aquele declive que possui o maior número de elementos. A solução fica o seguinte:

```
max_quantidade = 0
for i from 1 to n:
    linhas.remove_todo_conteudo()   # Tabela de hash
    for j from i + 1 to n:
        x1, y1 = pontos[i]
        x2, y2 = pontos[j]
        if x1 == x2:
            a, b, c = 1, 0, -x1
        else:
            a, b, c = y2 - y1, x1 - x2, x2 * y1 - x1 * y2
            if a < 0:
                a, b, c = -a, -b, -c
            G = gcd(a, gcd(b, c))
            a, b, c = a // G, b // G, c // G
```

```
        declive = tupla(a, b, c)
        linhas[declive].adiciona_sem_repetir(i)
        linhas[declive].adiciona_sem_repetir(j)

    foreach pontos_mesma_reta in linhas.valores:
        quantidade = len(pontos_mesma_reta)
        if quantidade > max_quantidade:
            max_quantidade = quantidade
            indices = pontos_mesma_reta
```

A complexidade da solução é $O(n^2)$.

Tutorial: Invencíveis

Vinícius Borges

A resolução do problema comparando-se as habilidades de cada par de discentes leva ao veredicto "Time Limit Exceeded" devido ao custo computacional $O(N^2)$

O problema pode ser resolvido em tempo $O(N \log N)$ utilizando-se a técnica **Dois Ponteiros**. Seja h o vetor $h = [h_1, \ldots, h_N]$ contendo as habilidades dos competidores. Ordene esse vetor de maneira decrescente e atribua duas variáveis $l = 1$ e $r = 2$. Faça a seguinte operação enquanto $r <= N$:

- Se $h[l] - h[r] > S$, significa que a diferença está maior do que o valor buscado. Considerando a ordenação decrescente realizada, faça $l \leftarrow l + 1$ para buscar um valor $h[l] - h[r]$ menor.
- Caso contrário, se $h[l] - h[r] < S$, faça $r \leftarrow r + 1$ para buscar um valor $h[l] - h[r]$ menor no vetor.
- Caso contrário, se $l \neq r$, encontramos a solução!

Tutorial: João dos Venenos

Daniel Saad

Este problema pode ser resolvido por meio da Programação Dinâmica do Problema da Mochila com repetições. Para ser mais preciso, a relação de recorrência que define a solução recursiva para o problema considerando os i primeiros itens e capacidade j de mochila é a seguinte:

$$T(i,j) = \begin{cases} 0, & i = 0 \\ \max\{T(i-1,j), \max_{1 \le k \le q_i} T(i, j - k \cdot w_i) + k \cdot v_i\}, & w_i \le j \wedge Q(i, j - w_i) < q_i \\ T(i-1,j), & \text{caso contrário} \end{cases}$$

Uma implementação direta desta relação de recorrência nos traz uma solução ineficiente, pois, para cada célula $T[i][j]$ da matriz de programação dinâmica, precisamos gastar q_i comparações para determinar o valor máximo, o que nos dá uma solução $O(NEQ)$, em que Q é a maior quantidade de repetições dos exercícios.

É possível reduzir este custo para $O(NE)$ ao manter uma janela deslizante (sliding window) que responda sempre qual é o maior elemento dentre $T(i, j - k \cdot w_i) + k \cdot v_i$, $1 \le k \le j_i$ em tempo constante (ou pelo menos constante amortizado). Ao todo são necessários w_i janelas deslizantes, um para cada valor de resto de j por w_i.

Tutorial: Karate

Alberto Tavares

Tutorial: Libertadores da América

Daniel Saad

Para resolver este problema, mantemos uma matriz T com a seguinte definição recursiva:

$$T(k,i) = \begin{cases} 0, k \leq N \wedge i \neq k \\ 1, k \leq N \wedge i = k \\ \sum_{j=1}^{N}(T(a,i) \cdot T(b,j) \cdot M[i][j]) + \sum_{j=1}^{N}(T(b,i) \cdot T(a,j) \cdot M[i][j]), \quad \text{c.c} \end{cases}$$

Em que a e b representam os confrontos (ou equipes) utilizados para organizar o confronto k
Em outras, palavras, $T(k,i)$ nos dá a probabilidade da i-ésima equipe chegar e vencer o k-ésimo confronto.
Suponha que r é o último confronto, isto é, a final da copa, então o resultado é simplesmente:

$$\sum_{i=1}^{N} T(r,i) \cdot b_i$$

Ou seja, a soma das probabilidades $T(r,i)$ para todos as equipes em que $b_i = 1$ (as equipes brasileiras).
Para calcular $T(k,i)$, podemos usar uma busca em profundidade na árvore de confronts com auxílio de uma tabela (memoization).
A complexidade da solução é $\Theta(N^3)$.

Tutorial: Maratona no Linf

Vinícius Borges

O problema é resolvido somando-se a quantidade total de competidores e verificando-se a quantidade deles que cabem no laboratório:

$$total = \sum_{i=1}^{N} a_i$$

e considerando que o LINF poderá receber $\lfloor \frac{C}{3} \rfloor$ equipes, a resposta é dada como:

$$ans = min(\lfloor \frac{C}{3} \rfloor, \lfloor \frac{total}{3} \rfloor)$$

Tutorial: Nefasta Publicidade

Vinicius Borges

Na maneira mais simples de resolver, não é necessário se preocupar com o valor N da entrada já que o importante é lidar com a duração M do vídeo.

Sabemos que se $M < 4$ com $M/neq2$, não será possível obter uma solução, devendo-se imprimir "-1". Tirando esse caso, podemos definir o problema da seguinte forma:

$M = 2 \times X + 5 \times Y$

em que existem restrições sobre os valores de X e Y, que dependem da paridade de M. Em relação a X, sabemos que apertar o botão "+2" várias vezes sempre leva a um valor par. Por isso, se M for par, Y tem que ser par também, ao passo que se M for ímpar, Y precisa ser ímpar.

Tutorial: Observe o Equilíbrio

Edson Alves