Tutorial: Baguete

Guilherme Ramos

A solução é simples, basta apresentar "TMJ!" caso a entrada fornecida seja "au-au". Caso contrário, a entrada necessariamente será "rrrrr" e basta apresentar "Esta repreendida!".

Tutorial: Coffee-Break Fit

Vinicius Borges

Uma estratégia por busca binária na resposta é capaz de resolver o problema. Para isso, deve-se ordenar um novo vetor contendo apenas os competidores que estão com lanche incompleto, considerando o valor excedente $a_i - b_i$ multiplicado pelo custo c_i. Suponha então que sejam K competidores com lanches incompletos.

Em seguida, devemos chutar o valor X e calcular o novo custo para cada competidor considerando as taxas impostas pela fornecedora. Para isso, faça $l = 1$ e $r = K$, e calcule $mid = (l + r)/2$. Sabendo que esse mid pode ser uma solução para o problema, verifique se o custo total de complementar o lanche fit excede o valor do orçamento M. Caso sim, guarde esse valor mid como uma resposta válida e tente aumentar o tamanho do chute X fazendo-se $l = mid + 1$. Caso contrário, deve-se diminuir o tamanho do chute para $r = mid - 1$ e tentar novamente.

A solução completa fica complexidade no tempo $O(N \log N)$ devido à operação de ordenação (a busca binária analisada separadamente possui complexidade $O(\log N)$.

Tutorial: dogsay

Guilherme Ramos

O primeiro passo é organizar a “imagem” do cachorro, copiando o formato de um dos exemplos. Apenas as 3 primeiras linhas precisam ser ajustadas com o texto fornecido (`msg`). Sendo L a quantidade de caracteres da mensagem, a primeira linha tem $2 + L$ caracteres ‘_’ (com um espaço os antecedendo). A segunda linha apresenta a própria mensagem entre espaços e os sinais ‘`<`’ e ‘`>`’. Por fim, a terceira linha se assemelha à primeira, apenas troca-se o sinal por ‘`=`’.

Tutorial: Estruturando o coffee-break

Daniel Saad Nogueira Nunes

Esse problema pode ser resolvido através do algoritmo de Duval, que obtém a fatoração **Lyndon** em $\Theta(n)$. Mas antes precisamos de duas definições:

Uma palavra **Lyndon** é aquela que é menor que todos os seus sufixos próprios. Por exemplo, `ababb` é uma palavra Lyndon, pois é menor do que:

- `babb`
- `abb`
- `bb`
- `b`

O período de uma string S, dado por X, é o menor prefixo de S tal que $S = X^k X'$, isto é, concatenações de k cópias de X com um prefixo X' (possivelmente vazio) de X. Por exemplo, se S é `abcab`, então seu período é `abc`.

Para realizar a fatoração Lyndon, que é o que se pede no problema, dividimos a string de entrada S em três partes, $S = S_1 S_2 S_3$. S_1 é o que já foi fatorizado. S_2 é o que está sendo fatorizado e S_3 é o que será fatorizado.

Gulosamente, tentamos estender S_2 o máximo possível, ao manter três índices, i, j e k. i é o início de S_2, j é o caractere que estamos avaliando se fará parte de S_2 ou não, e k sinaliza o período de S_2 de tal forma que $j - k$ é o tamanho do período.

Temos três situações:

- Se $S_j > S_k$, então podemos incluir S_j em S_2, isto é, estendemos S_2 e o período de S_2 é a própria string, portanto k passa a valer i.
- Se $S_j = S_k$, o período de S_2 aumenta de uma unidade, portanto incrementa-se k.
- Se $S_j < S_k$, não é possível estender S_2.

No terceiro caso, $S_2 = X^k X'$ para uma string X de período $j-k$. Assim, fatoramos S_2 como $|X|X|\ldots|X|$, deixando X' para a próxima iteração.

O algoritmo de Duval pode ser visualizado no seguinte link `https://cp-algorithms.com/string/lyndon_factorization.html`

Tutorial: Fenótipos

Guilherme Ramos

Para resolver este problema, basta ler as siglas dos três animais e verificar, para cada característica do filhote, se ela existe em qualquer um dos pais. Caso todas existam, o cachorrinho as herdou. Caso uma ou mais não existam, a ascendência não é certa.

Tutorial: Hurricane!

Daniel Porto

Pode-se usar uma pilha para armazenar monstros de cada jogador. Ao jogar uma carta *Hurricane!*, basta desempilhar (se possível) a carta da pilha do oponente.

Tutorial: K-ésimo Kara Kickado

Alberto Neto

Este problema é uma variação do problema de Josephus.

Escreva os números na base $(m+1)$. Ao passar uma vez pela lista, vamos estar excluindo todos os números com dígito menos significativo diferente de m. Note que, ao passar pela segunda vez na lista, o primeiro ciclo de exclusão possivelmente será menor por termos excluido no final da lista. Seja q o número de exclusões deste ciclo antes de dar a volta. Agora, vamos excluir todos os números com segundo dígito menos significativo diferente de $(m-q)$.

Este algoritmo pode ser simulado com uma função recursiva, que recebe como argumento a quantidade n de pessoas ainda na lista, o parâmetro m, quantos ainda devem ser excluidos e o parâmetro q. É possível calcular em O(1) quantas pessoas serão excluidas nesta iteração da lista, e os próximos parâmetros. A complexidade final fica $O(log(n))$, pois sempre dividimos o n por $(m+1)$ e, quando $n=1$, a recursão para.

Tutorial: Letra aleatória

Daniel Saad Nogueira Nunes

Uma forma de resolver esse problema é contar o número de ocorrências de cada símbolo de S_1 com uma tabela T.

Em seguida, para cada símbolo c em S_2, o número de ocorrências de $T[c]$ é diminuído de 1. A resposta é o caractere c tal que $T[c] = -1$.

A complexidade desta solução é $\Theta(|S_2|)$.

Outra forma de resolver é ordenando as duas strings em ordem crescente. Em seguida, comparamos cada caractere $S_1[i]$ com $S_2[i]$ e, caso sejam diferentes, a resposta é $S_2[i]$. Se chegarmos ao final de S_1 ,então a resposta é o último caractere de S_2. Esta solução tem complexidade $\Theta(|S_2| \lg(|S_2|))$.

Tutorial: Nürburgring

Daniel Porto

Para resolver o problema, deve-se criar uma função recursiva que calculará o tempo de cada volta n. Para não estourar o tempo de execução, deve-se registrar em uma hash os tempos já calculados para evitar repetir contas já realizadas.