Tutorial: Balão++

Guilherme Ramos

A solução é simples, basta verificar se a entrada é “balao” ou “baloes” para apresentar “Mais! Mais!”. Qualquer outro valor resulta em “Bu!”.

Tutorial: Hora do Lanche

Author name

Para resolver o prolema, calcule a distância Euclidiana entre cada estudante e a cantina. Em seguida, ordene de maneira ascendente essas distâncias, mas associando cada uma com o identificador inteiro do estudante. Trate os casos de empate durante a ordenação, isto é, ao presenciar duas distâncias iguais, deve-se dar prioridade ao maior índice.

Para evitar valores reais na solução, não é necessário passar a raiz no cálculo da distância Euclidiana.

Complexidade da solução: $O(NlogN)$ no tempo.

Tutorial: Krash Bondjump

Jeremias Gomes

Este problema pode ser resolvido utilizando uma estratégia gulosa onde, dada a posição atual de Krash, calcula-se e salva-se a máxima posição que é possível alcançar a partir dela própria. Assim, para as posições seguintes da fase, basta apenas confirmar se foi possível chegar até a aquela posição ou não, atualizando o máximo alcance onde, se houver alguma posição a qual não é possível alcançar, a fase é impossível.

A solução descrita é o seguinte:

```
alcance = 1
for i from 1 to P:
  if alcance < i:
    return "Impossivel"
  alcance = max(alcance, ponte[i] + i)
return "Possivel"
```

Tutorial: Nara

Lucas Mattioli

A primeira observação é que vale apenas buscarmos substrings de tamanho exatamente K em vez de substrings com tamanhos maiores que K. Isto é verdade porque se uma string de tamanho Y tal que $Y > K$ faz T_i pertencer a S, então essa string tem uma substring de tamanho K que também faz a mesma coisa. Assim, o problema vira "... substrings de tamanho pelo menos K" para "... substrings de tamanho igual a K", o que facilita um pouco o escopo.

Com a observação acima e com o fato de que todas as strings são binárias, podemos chegar à conclusão de que só existem 2^K strings possíveis que poderiam estar presentes em "pertencimentos". Assim, antes de processar qualquer query, passamos em todas as substrings de S de tamanho K e guardamos num array booleano de tamanho 2^{20} (20 sendo o máximo valor de K) as que estão presentes ali, interpretando cada substring como uma bitmask. Quando uma query T_i chegar, basta passar em todas as substrings de tamanho K de T_i e verificar, pra cada uma, se ela existe ou não em S usando o array que preprocessamos; se qualquer uma existir, então a resposta para aquela T_i é 1.

A complexidade final fica $O(N + \sum_{i=1}^{Q} |T_i|)$.

Tutorial: O sapo não lava o pé

Guilherme Ramos

Primeiramente, lê-se o string com o trecho da música. A seguir, enquanto houver uma vogal a ser lida, ler a vogal e substituir todas as ocorrências de vogais (A, E, I, O, U) por esta vogal lida e apresentar o resultado. Dois detalhes são importantes: manter o trecho original armazenado para fazer a troca corretamente e manter a caixa das vogais.