

?Adreuqse arap atierid

Este problema foi inspirado na definição real de algoritmos bilaterais do Unicode. Mais especificamente pela palestra Wait, ?tahW: The Twisted Road to Right-to-Left Language Support de Moriel Schottlender (<https://www.youtube.com/watch?v=OCQd02h0RJQ>).

A solução do juiz utiliza de três passadas no texto e alguns vetores auxiliares:

- Um vetor é criado que indica qual a direção do caractere naquela posição, podendo ser $+1$ ou -1 . Visto que o primeiro caractere sempre tem direção definida, é só questão de avaliar se o caractere atual é uma letra latina minúscula e colocar $+1$, ou se for uma letra latina maiúscula, -1 , ou senão, copiar a direção do caractere anterior.
- Em seguida, é caminhado no texto criando um intervalo $[l, r)$. l é inicialmente 0, e o r é iterado. O último caractere neutro é guardado em uma variável chamada nnr . Assim que a direção do texto for alterada, o l é colocado em uma nova posição. Se a troca da direção é feita de -1 para $+1$, invertemos aquela parte do texto, desde l até nnr . A inversão leva em conta os caracteres invertidos como ‘(’ para ‘)’.
- No final, é feito um processo parecido em busca dos conjuntos de caracteres fracos que foram invertidos e devem ser agora desinvertidos. Usando de um vetor que diz se uma seção foi invertida, é procurado por um segmento $[l, r)$ de caracteres fracos invertidos, que são invertidos novamente. É necessário prestar atenção para que não seja desinvertido uma parte do segmento que anteriormente não havia sido invertido.

A complexidade desta solução é efetivamente $\mathcal{O}(n)$, usando técnicas de *two pointers*. Isto porque cada segmento $[l, r)$ em cada passo invertido é disjunto de todos os outros segmentos, e os laços iteram apenas em uma direção até o fim do texto.

Soluções usando apenas um laço com vários laços aninhados fazendo as diversas inversões de forma mais direta são possíveis.