

Explicação dos Exercícios

Exercícios D, F, G, H

Larger Score

- É dado um vetor não necessariamente ordenado
- Observa-se que o vetor é dividido em duas partes:

1ª Parte -> $(v[0], v[1], \dots, v[k-1])$

2ª Parte -> $(v[k], v[k+1], \dots, v[n-1])$

Objetivo -> Seja $S = v[0] + v[1] + \dots + v[k-1]$, trocar um valor dessa soma S , por um valor da segunda parte, de forma que se tenha uma nova soma $S' \geq S+1$.

Larger Score

Como eu posso fazer isso?

As trocas podem ser feitas entre elementos adjacentes.

Ex:

$k = 2$

8 4 2 3 6 7 5

8 4 2 6 3 7 5

8 4 6 2 3 7 5

8 6 4 2 3 7 5

Nesse exemplo o resultado é 3, isto é, houveram três trocas.

Larger Score

Como resolver?

O exercício quer que sejam realizadas o mínimo de trocas possíveis.

8 9 2 5 2 6 12 2 3 6

Minha Solução:

Talvez seja overkill, mas... vamos lá.

1º Passo: Ordenar o vetor, guardando seus índices.

2 2 2 3 5 6 6 8 9 12

índices -> 5 8 3 9 4 10 6 1 2 7

2º Passo: Passar todos os elementos do vetor ordenado

Larger Score

2 2 2 3 5 6 6 8 9 12

índices -> 5 8 3 9 4 10 6 1 2 7

Nesse passo se o número pertencer à primeira parte do vetor (número vermelho) adiciona o índice em uma priority queue.

Ex: Ao passarmos pelo 2, por exemplo adicionamos o índice 3 na priority, ao passarmos pelo elemento 5 adicionamos o índice 4 na priority e assim sucessivamente.

Se o número pertencer a segunda parte do vetor (número azul), a gente verifica se tem algum item na priority, se tiver, a gente encontrou um par que pode ser trocado.

Dessa forma, pode-se verificar qual é a mínima distância entre todos esses pares, e essa é nossa resposta final.

Larger Score

E porquê usar a priority?

Porque ela vai guardar os elementos de índice mais alto, ou seja, os mais próximos à sua posição atual.

Observe que como os elementos estão ordenados, eu vou achar somente elementos maiores do que aqueles que eu já coloquei na priority, e isso traz a solução ótima do exercício.

:)

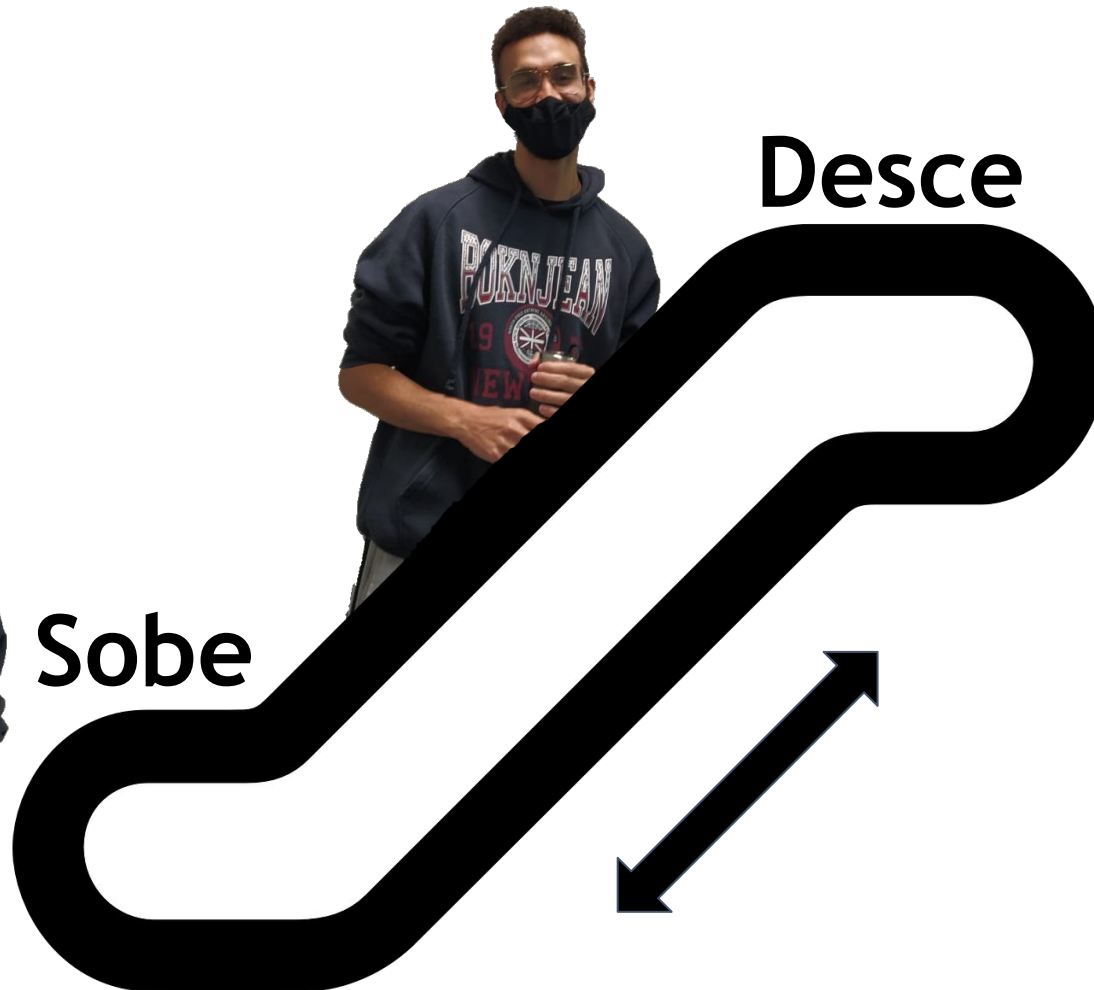
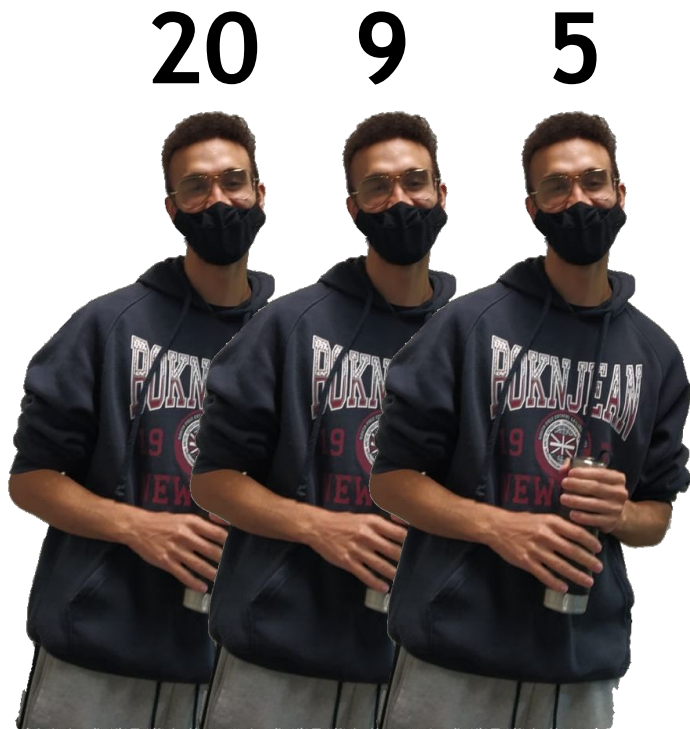
Escalator

- **Descrição:** uma escada rolante permite o deslocamento em ambas suas direções. O percurso para ir de uma ponta da escada até a outra leva 10 segundos, isto é, se uma pessoa entrar no tempo T , ela deixa a escada no tempo $T + 10$. As seguintes regras se aplicam:
 - a escada está inicialmente parada;
 - quando uma pessoa chega em uma de suas extremidades, a escada inicia o movimento na direção desejada;
 - se a escada já estiver em movimento, a pessoa pode entrar imediatamente nela;
 - caso contrário, é preciso esperar ela parar e iniciar o movimento contrário.

Escalator

- **Problema:** simular o tempo de funcionamento da esteira, dadas informações sobre N pessoas, incluindo o tempo de chegada de cada um e a direção de sua locomoção.

Escalator



Escalator

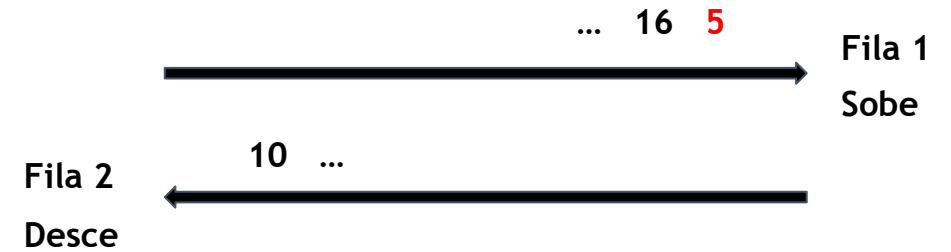
- **Solução:** primeiramente, precisamos pensar em uma forma de guardar as informações dos N papis de maneira a manter a ordem de chegada de cada um e segui-la, para cada uma das direções possíveis.
- Mas, como podemos fazer isso?
- Um vetor seria uma possibilidade, porém, dado o problema, podemos pensar que o problema pede para que o primeiro papi que entrar em qualquer um dos lados também deve ser o primeiro a sair, portanto...

Escalator



Escalator

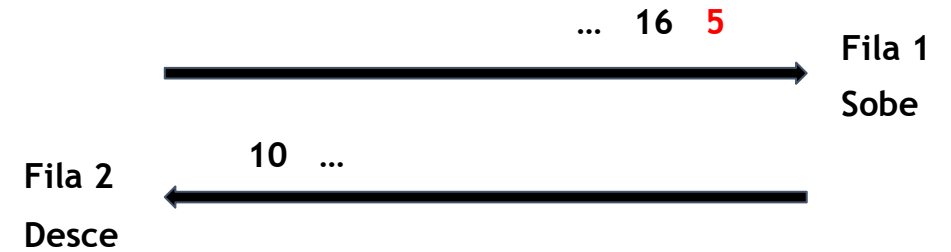
1. Escolher o menor tempo T como tempo de início do movimento, definindo a direção;



$T = 5$
 Direção = sobe

Escalator

2. Verificar se $T + 10$, o tempo que demora para um papi chegar de uma extremidade a outra, é menor ou igual ao próximo papi esperando na fila atual;



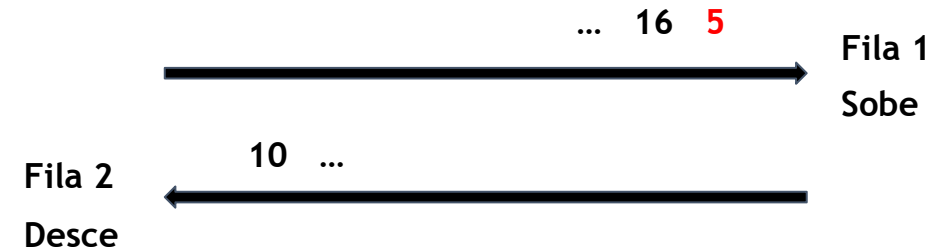
$$T = 5$$

Direção = sobe

$$T + 10 = 15 \leq 16 ?$$

Escalator

3. Caso seja verdadeiro, o próximo papi da fila entra na escada rolante e retornamos ao passo 1;
4. Caso contrário, verificamos quem chegou primeiro na escada rolante, o papi da Fila 1 ou o papi da Fila 2;



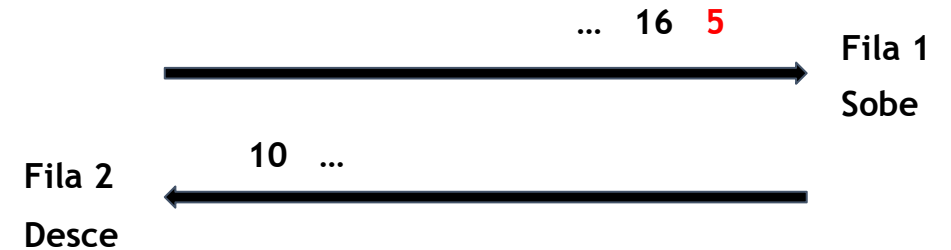
$$T = 5$$

Direção = sobe

$$T + 10 = 15 \leq 16 ? \text{ X}$$

Escalator

5. Se o papi da fila na direção que estamos atualmente chegou primeiro, trocamos o valor do tempo para o do papi atual e continuamos na mesma direção;
6. Se o papi da fila na direção oposta chegou antes, então somamos 10 ao T, referente a trajetória do último papi da fila atual e trocamos a direção;



$16 < 10 ? \text{X}$
 Portanto, $T = T + 10$
 Direção = desce

Escalator

- O algoritmo se repete até que ambas as filas estejam vazias e tenhamos simulado todas as viagens de cada papi;
- Ao final das trajetórias, basta somar 10 ao T obtido, contabilizando a última viagem feita pelo último papi.

Brackets balanceados

- Caracteres delimitadores como (,), {, }, [e] podem ser chamados de *brackets*.
- Dois *brackets* são considerados um par se um *bracket* de abertura ocorre a esquerda de um de fechamento e se eles são exatamente do mesmo tipo: (), [], {}.

Brackets balanceados

- Uma certa expressão é bem definida ou balanceada se atende uma das seguintes propriedades:
 - Ela é uma cadeia de caracteres vazia
 - Ela é formada por uma cadeia bem definida envolvida por *brackets*. Ou seja, se S é balanceada, então as cadeias (S) , $[S]$ e $\{S\}$ também são.
 - Ela é formada pela concatenação de duas cadeias balanceadas. Portanto, se X e Y são balanceadas, então XY também é.

Exemplos

- Expressões balanceadas

()

{}

[]

{[](){}[]}

- Expressões não balanceadas

Problema

- Para uma expressão qualquer formada apenas por *brackets*, como determinar se ela é balanceada ou não?
- Uma forma de resolver isto é utilizando uma **pilha** de apoio.
 - Percorreremos a expressão da esquerda para direita:
 - Se o caractere atual é um *bracket* de abertura: empilhamos o *bracket*
 - Se o caractere atual é um *bracket* de fechamento
 - Se a pilha está vazia, a expressão não está balanceada
 - Se o caractere corresponde ao topo da pilha, desempilhamos
 - Caso contrário, a expressão não está balanceada
 - Terminando de percorrer a expressão, se a pilha ainda contiver algum elemento, então algum *bracket* não foi fechado.

Exemplos

{ [()] () }



Exemplos

{ [()] () }



{

Exemplos

{ [()] () }

↑

[
{

Exemplos

{ [()] () }

↑

(
[
{

Exemplos

{ [()] () }

↑

[
{

Exemplos

{ [()] () }

 ↑

{

Exemplos

{ [()] () }

↑

(
{

Exemplos

{ [()] () }

 ↑

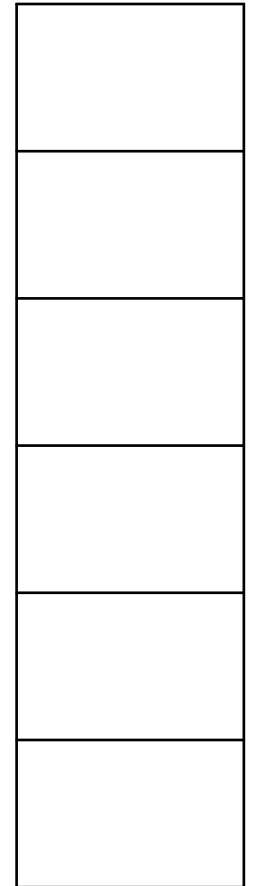
{

Exemplos

{ [()] () }

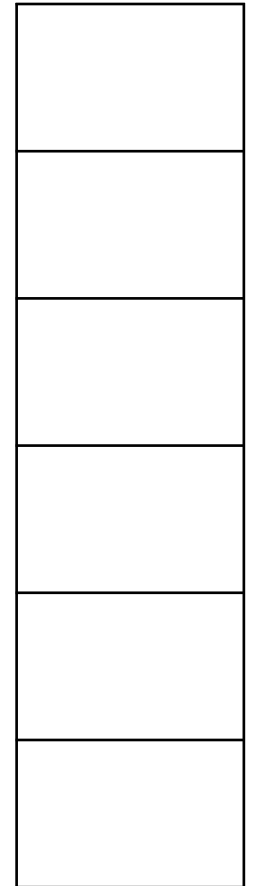


Fim da cadeia e pilha vazia: OK



Exemplos

{ [}]



Exemplos

{ [}]

↑

{

Exemplos

{ [}]

↑

[
{

Exemplos

{ [}]


↑

Brackets não correspondem!

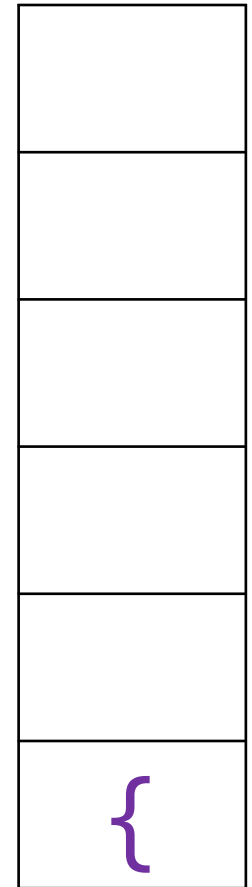
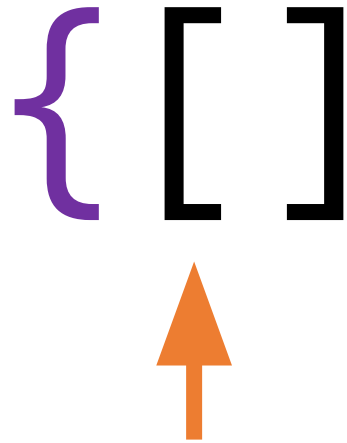
[
{

Exemplos

{ []



Exemplos



Exemplos

{ []

 ↑

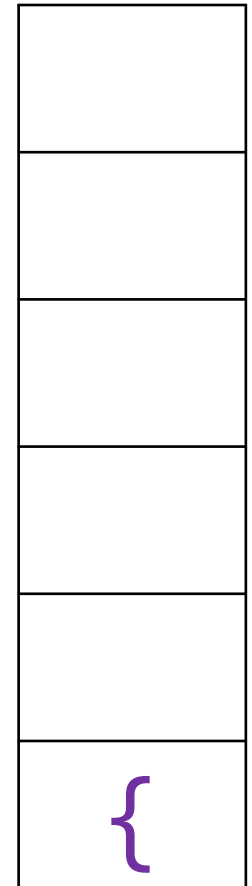
[
{

Exemplos

{ []




A cadeia terminou sem problemas, mas ainda há elementos na pilha!

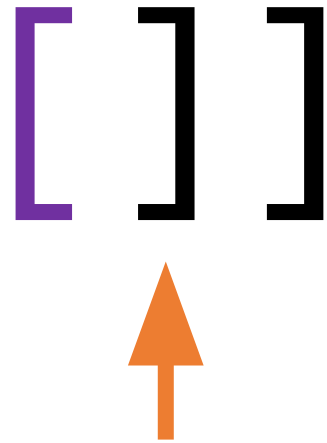


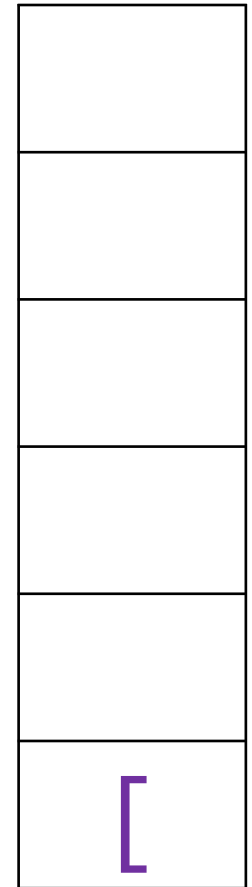
Exemplos

[]]



Exemplos

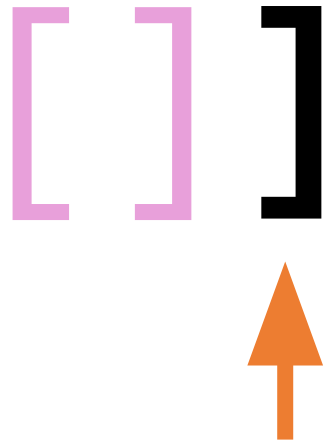




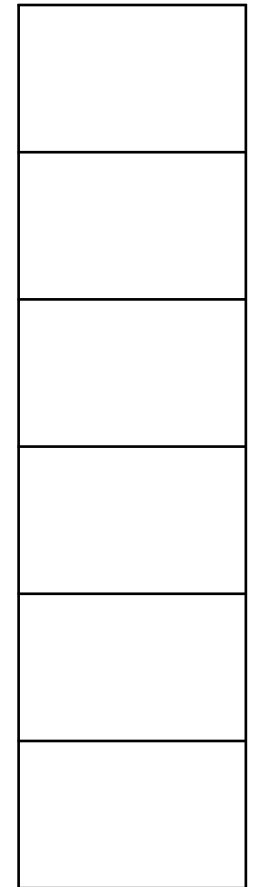
Exemplos

[]]
↑

Exemplos



Pilha vazia!



Sugestões de exercícios

- [EXPRESS11 - Expressões \(SPOJ BR\)](#)
- [Balanced Brackets \(HackerRank\)](#)
- Desafio: [224C - Bracket Sequence \(CodeForces\)](#)