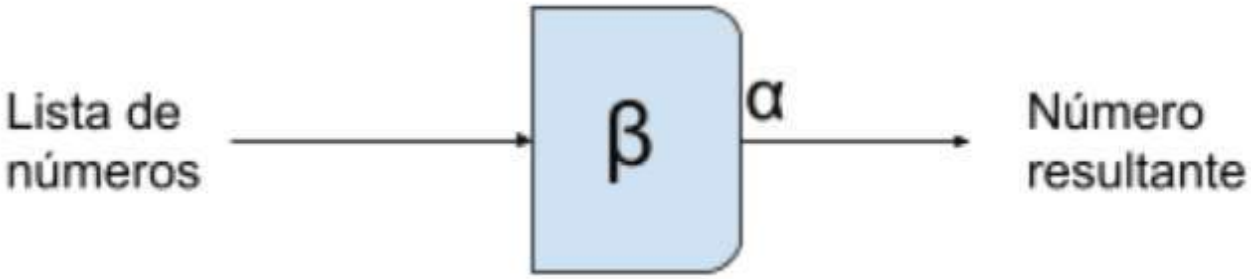
Por último, um desafio.

Imagine o seguinte elemento, que recebe uma lista de números, aplica uma função beta na lista e retorna o resultado multiplicado por um fator alfa:



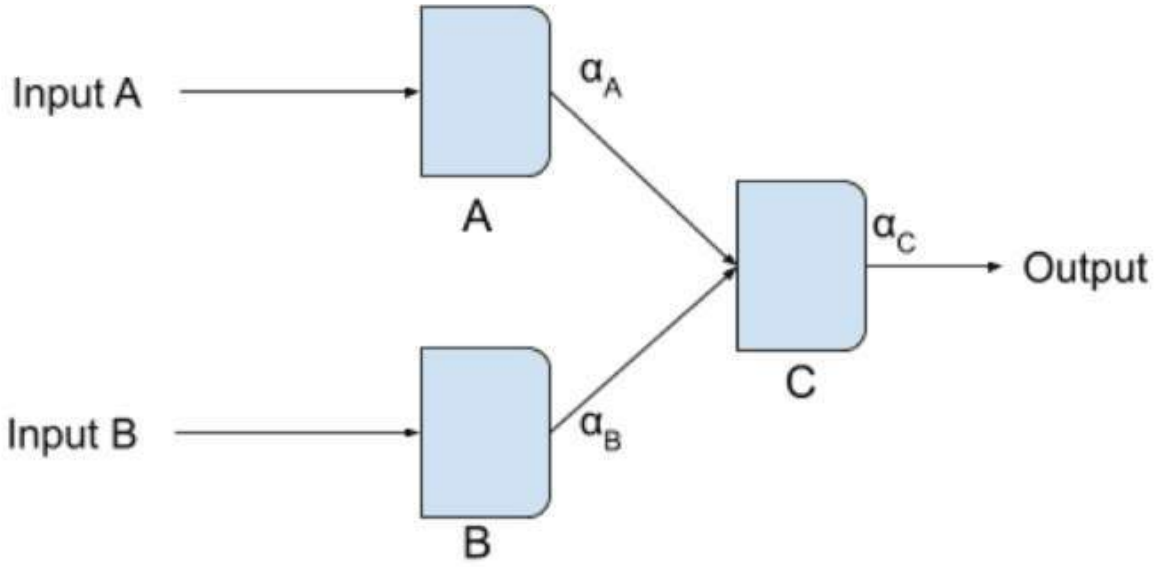
A função beta é definida pela equação:

$$\beta(inputs) = floor\left(tanh\left(\frac{\Sigma\, inputs}{1000}\right) \times 10\,\alpha\right)$$

Considerando que exista uma função "sum" que soma todos os elementos de um array, é possível implementar a fórmula com a seguinte linha de código em Javascript

```
beta = Math.floor(Math.tanh(sum(inputs)/1000) * 10 * alpha);
```

Uma vez definido o nosso elemento unitário, é possível encadeá-lo para criar uma rede. Assim, as saídas dos primeiros elementos tornam-se as entradas para os próximos elementos, vide imagem abaixo de exemplo:



Mesmo que os elementos pertençam a uma mesma rede, eles são independentes entre si, de modo que cada um possui um fator alfa diferente.

Definição dos fatores alfa

Inicialmente, todos os elementos de uma rede se encontram com o fator alfa = 5. Para esta situação inicial, damos o nome de rede indefinida. É dada uma lista de números cujo output da rede é usado para definir o fator alfa de cada elemento.

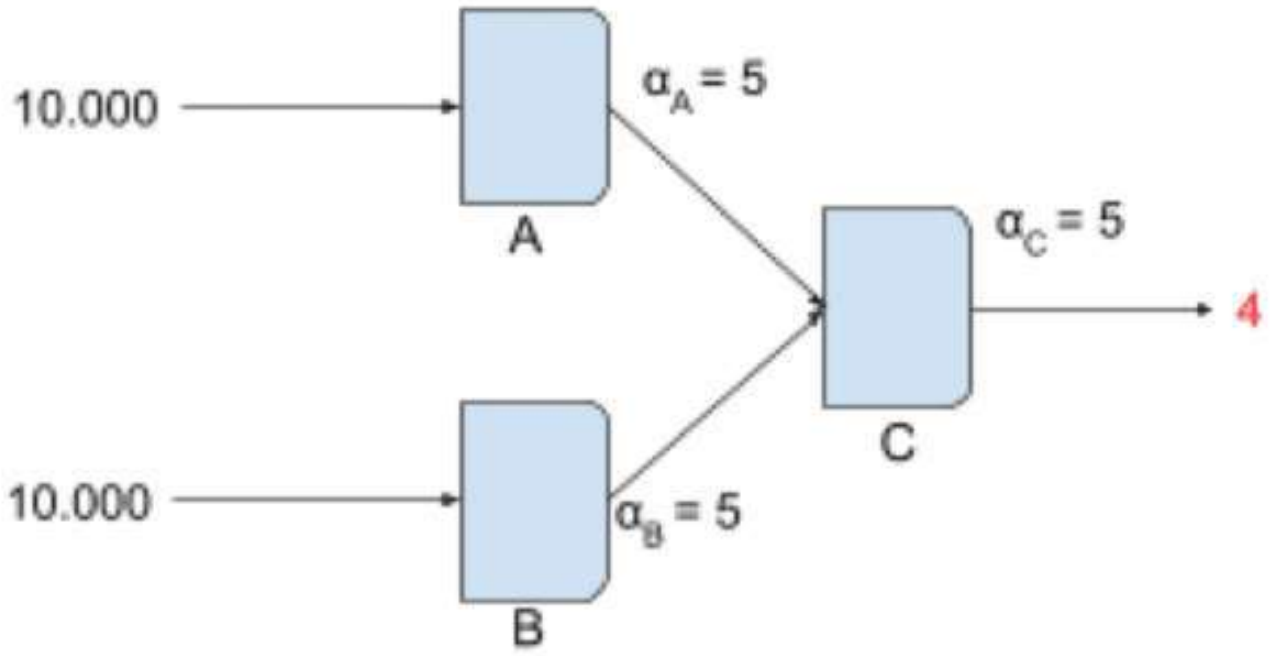
Define-se um elemento de cada vez da camada, antes de passar para a próxima. Na imagem acima, seria o equivalente a uma ordem alfabética.

A seguir são ilustrados os passos de definição da rede, de acordo com a lista de exemplo:

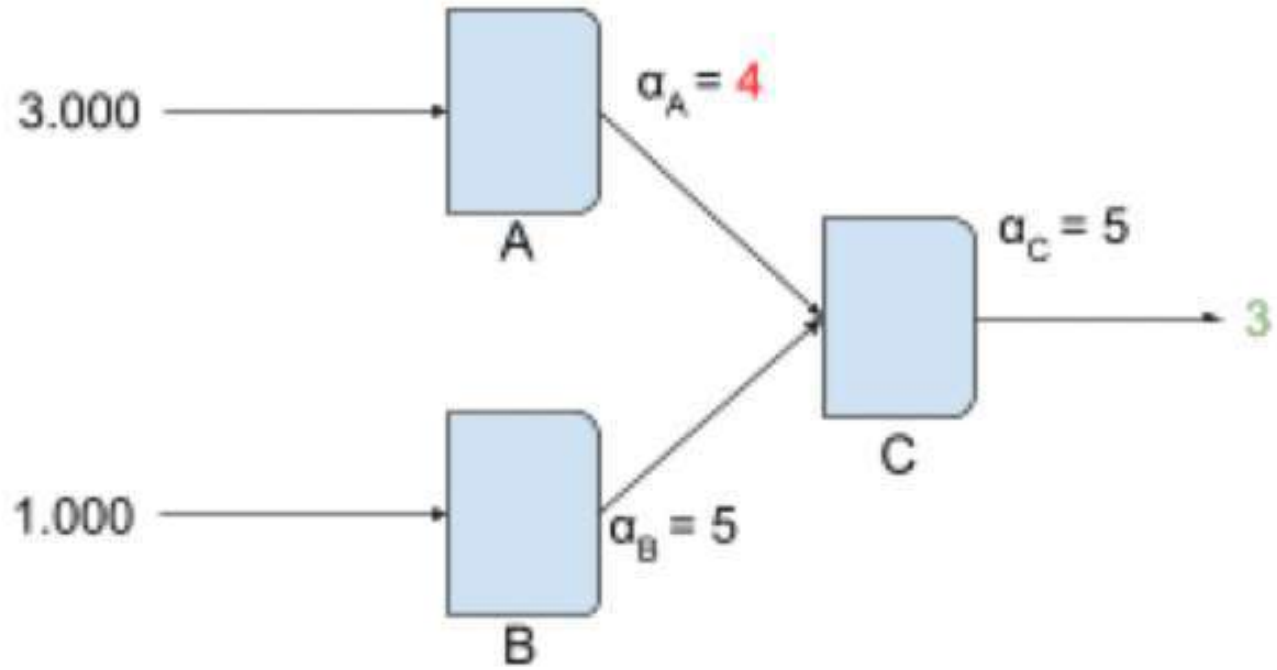
Input A	Input B	Input C (calculado)	Output rede	Alfa definido
[10.000]	[10.000]	[49, 49]	4	Elemento A
[3.000]	[1.000]	[39, 38]	3	Elemento B
[-123.000]	[-456.000]	[-40, -30]	-4	Elemento C

É importante ressaltar que os alfas calculados já são utilizados na próxima iteração. Após o cálculo da primeira iteração, o alfa de A é atualizado e já é considerado no cálculo de alfa B.

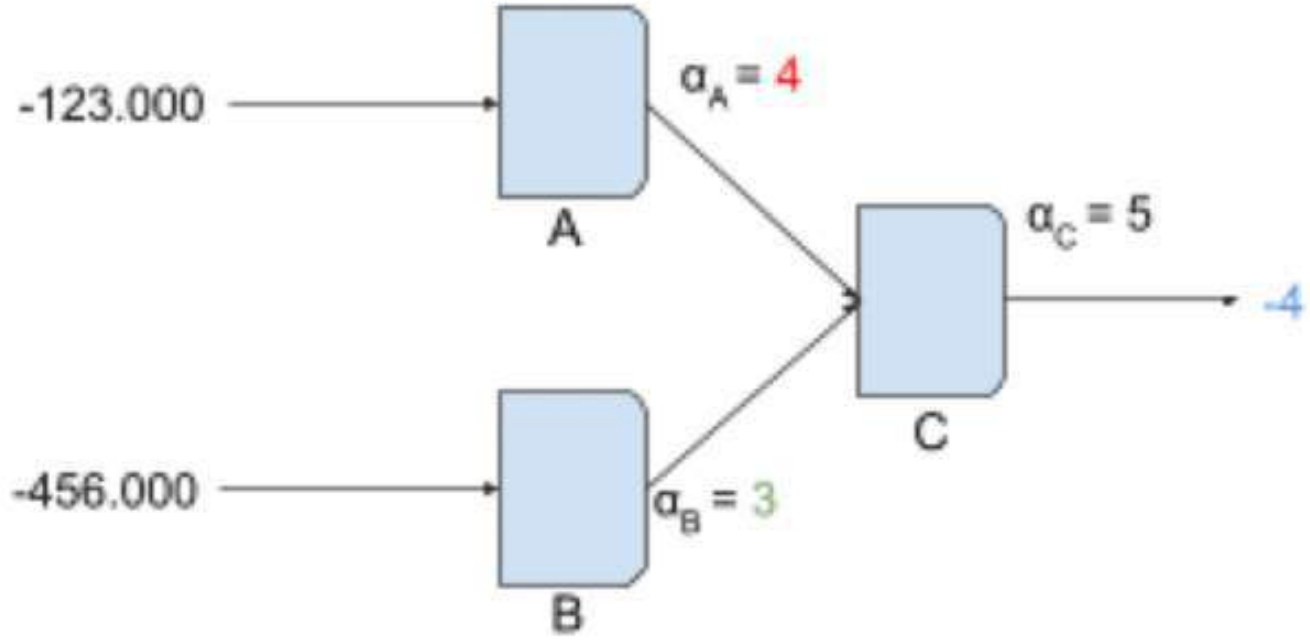
Passo 1:



Passo 2:

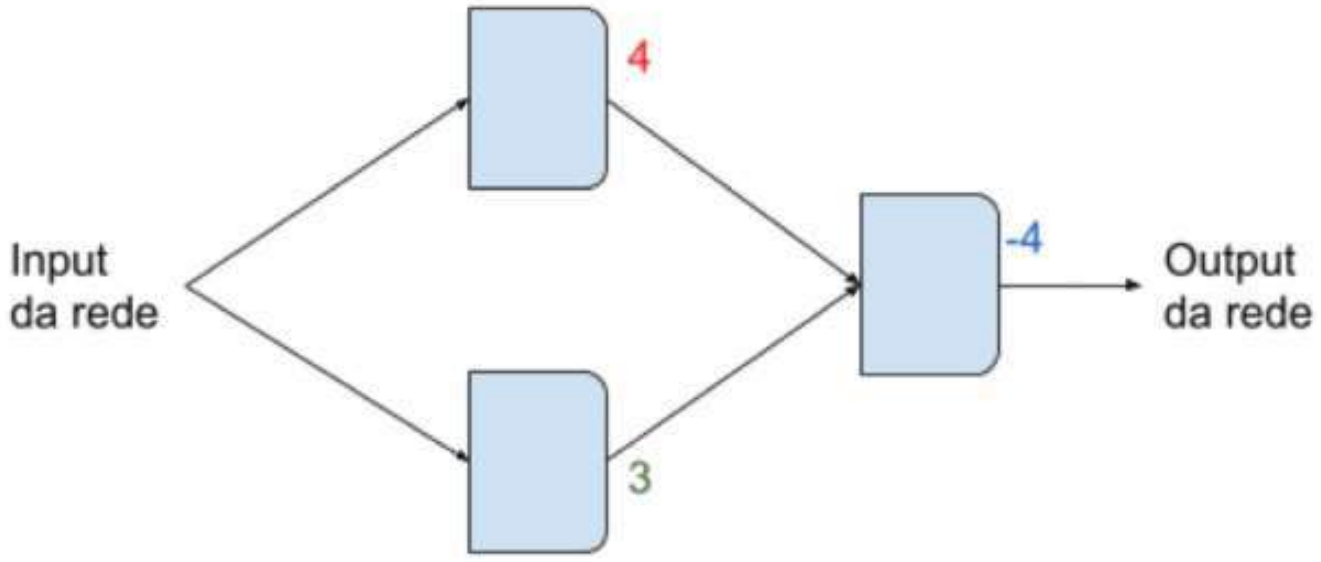


Passo 3:



Processando dados

Uma vez calculados todos os fatores alfa, denominamos a rede como definida. Nesta situação, uniformizamos a entrada de todos os elementos da primeira camada para processar dados, ou seja, todos os elementos da primeira camada possuem a mesma lista como input. A seguir há uma tabela com os inputs e respectivos outputs de exemplo

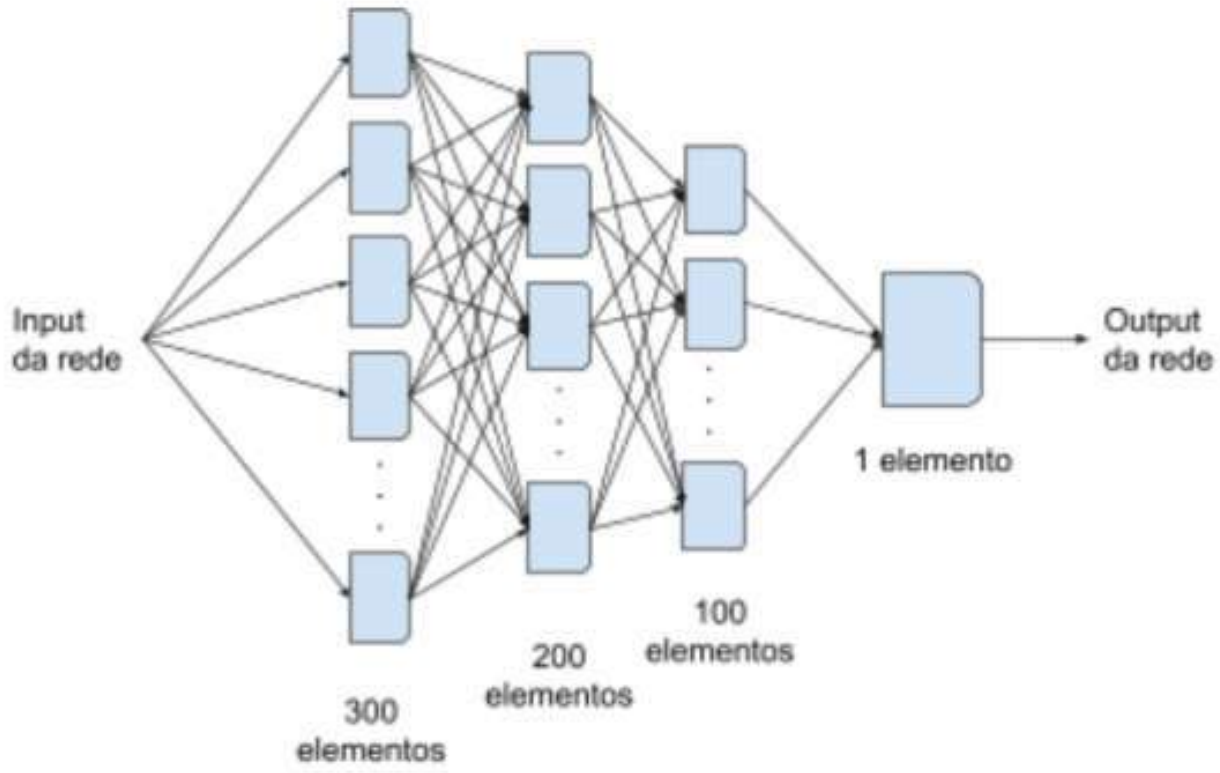


Input da rede	Output da rede
[100]	-1
[200, 300]	-2
[400, 500, 600]	-3
[700, 800]	-3
[900]	-2

Desafio

ATENÇÃO: Não publique a solução no Github ou nenhuma outra plataforma.

Agora é a sua vez de criar uma rede! Baseado nos conceitos explicados até aqui, construa em Typescript uma rede de 4 camadas com 300, 200, 100 e 1 elementos, respectivamente. Vale notar que a saída de cada elemento é usada em todos os elementos da camada seguinte, conforme é mostrado na figura



Enviamos junto a este documento dois arquivos:
- network-definition.json: utilize-o para definir os fatores alfa da rede;
- network-processing.json: após definição da rede, processe esta lista de números.

Para confirmar que você acertou o desafio, pedimos dois números como resposta:
- A soma dos fatores alfa de todos elementos da rede após definição da rede (no nosso exemplo inicial de 3 elementos, a resposta seria 4 - 4 + 3 = 3)
- A soma de todos os outputs da rede após processamento do arquivo network-processing.json (na tabela de exemplo acima, a resposta seria -11)

Dica: ambas respostas são números entre -2000 e 2000.

Observações:
- Não publique o enunciado ou a solução do desafio em nenhum lugar
- Sugerimos confirmar seu resultado de alguma forma antes de enviar
- Faça o código em um único arquivo .ts

Resposta

Soma dos fatores alfa: *

Soma dos outputs: *

Faça o upload do código typescript abaixo: *

Clique para escolher um arquivo ou arraste ele para cá

Tamanho máximo: 10 MB

Seguir →