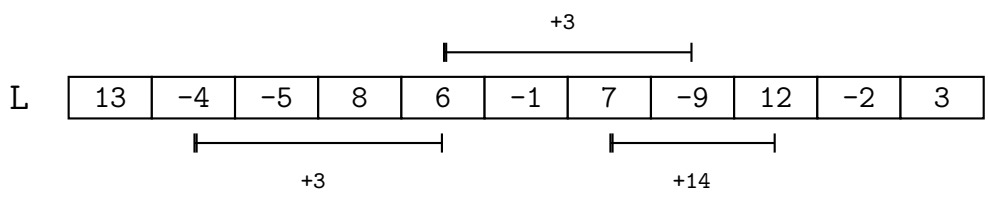
O problema da faixa com a soma máxima

Imagine que L é uma lista de tamanho n que contém números positivos e negativos

Daí, nós calculamos a soma de uma faixa de elementos consecutivos

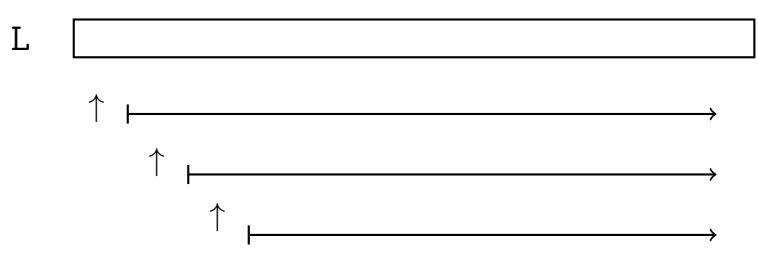


E daí, a gente define o problema assim

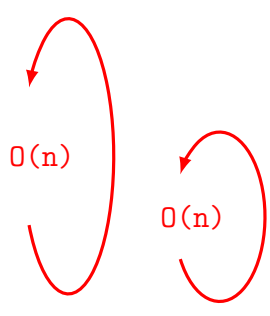
→ *Encontrar a faixa que tem a soma máxima*

Certo.

Agora, a gente considera o algoritmo porcaria para esse problema — (*que calcula a soma de todas as faixas*)



A coisa fica assim




```
func  PFSM-1.0 ( L )

    MF ← 0

    Para i ← 1 Até N-1

        soma ← 0

        Para j ← i Até N

            soma ← soma + L[j]

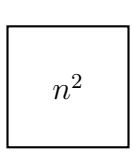
            Se ( soma > MF )

                MF ← soma

    Retorna ( MS )
```

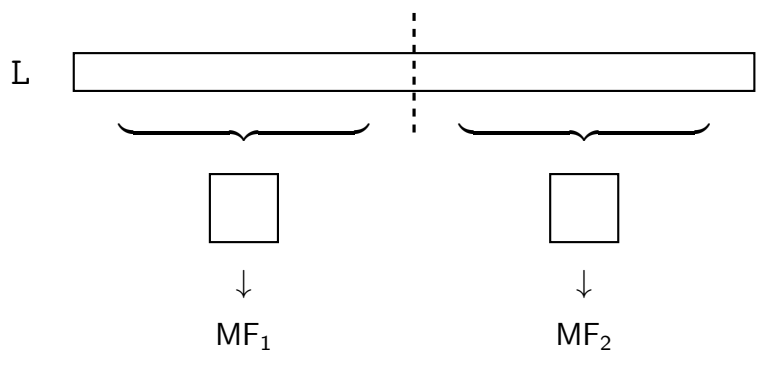
O tempo de execução desse algoritmo é $O(n^2)$ — (*o que é uma porcaria ...*)

Mas isso é um bom começo



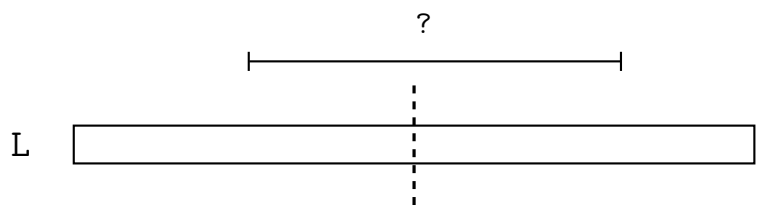
Porque agora, a gente pode aplicar a estratégia da divisão.

Quer dizer, a ideia é aplicar o algoritmo porcaria nas duas metades da lista



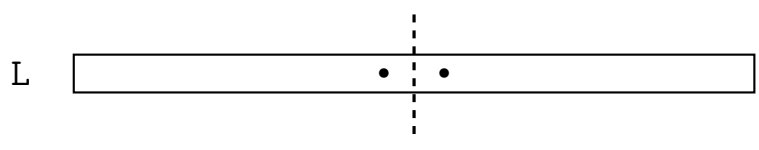
Só que isso ainda não dá a solução do problema.

Porque pode haver uma faixa com soma maior que contém elementos nas duas metades

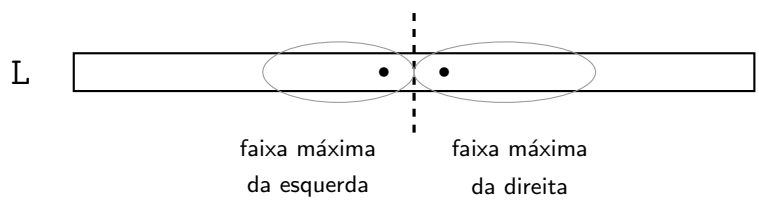


Então agora, a gente precisa encontrar a melhor faixa desse tipo.

Para isso, a gente nota que essa faixa vai conter os elementos que ficam dos dois lados da quebra

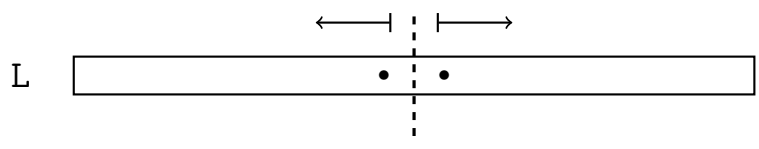


E a ideia é que dos dois lados nós vamos pegar a faixa de soma máxima que contém o elemento



Porque daí, juntando as duas, nós vamos ter a faixa máxima que contém elementos nas duas metades — (*porque?*)

E para obter a faixa máxima de cada lado, basta percorrer a lista — (*do meio para as pontas*)



A coisa fica assim

```
func faixaMeio ( L, inicio, meio, fim )

    FE ← 0;   soma ← 0

    Para i ← meio Até inicio

        soma ← soma + L[i]

        Se ( soma > FE )

            FE ← soma

    FD ← 0;   soma ← 0

    Para j ← meio+1 Até fim

        soma ← soma + L[j]

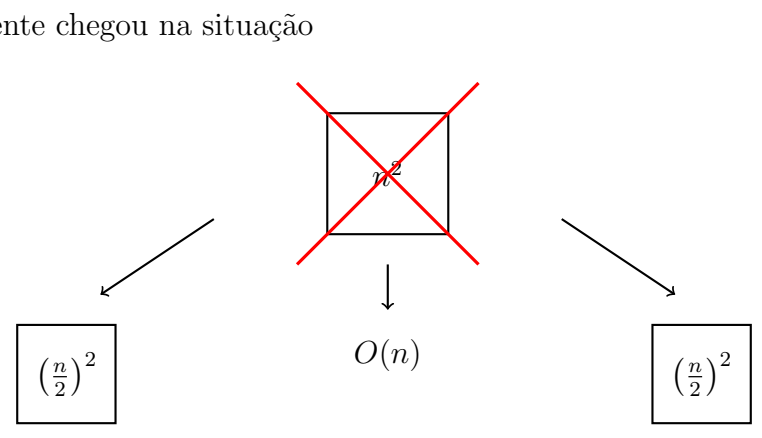
        Se ( soma > FD )

            FD ← soma

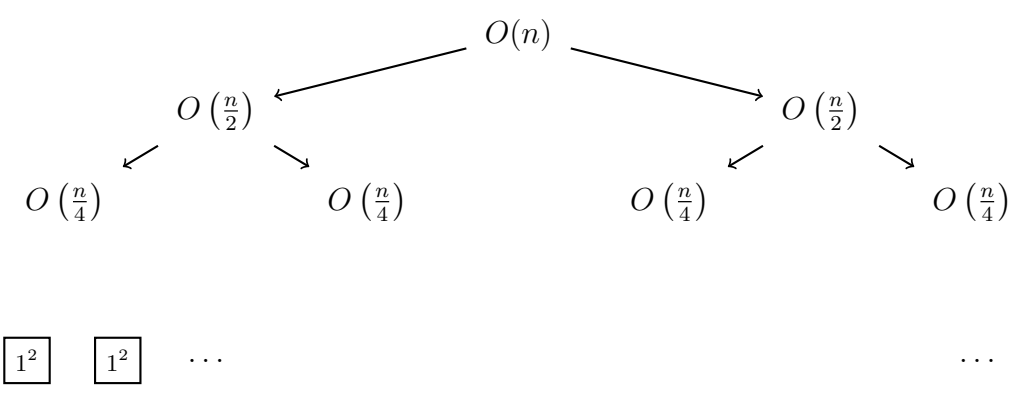
    Retorna (FE + FD)
```

A boa notícia é que isso leva tempo $O(n)$.

E mais uma vez, a gente chegou na situação



Daí, girando a manivela a gente chega outra vez nisso aqui



Quer dizer, o nosso algoritmo resolve o problema em tempo $O(n \log n)$.

Só que ainda falta escrever o pseudo-código

```
func  PFSM-2.0 ( L, inicio, fim )

    Se ( inicio = fim )

        Retorna (PFSM-1.0 (L, inicio, fim))

    meio ← (inicio + fim) / 2

    FM1 ← PMS-2.0 (L, inicio, meio)

    FM2 ← PMS-2.0 (L, meio+1, fim)

    FM3 ← faixaMeio (L, inicio, meio, fim)

    FM ← MAX { FM1, FM2, FM3 }

    Retorna (FM)
```

Legal, não é?