

\* Divisão e Conquista

\* Par mais próximo

Não faz o código

\* Subvetor de Soma Máxima

8	-10	3	-1	4	3	-6
---	-----	---	----	---	---	----

pref, suf, ssm, soma



sufix pref.

SSM(v, l, r)

if ( $l = r$ )

resp.soma = v[l]

resp.ssm = resp.pref = resp.suffix = max(v[0], 0)

return resp

mid = (l+r)/2

left = SSM(v, l, mid)



left right

right = SSM(v, mid+1, r)

resp.ssm = max(left.ssm, right.ssm,  
left.suffix + right.prefix)

resp.soma = left.soma + right.soma

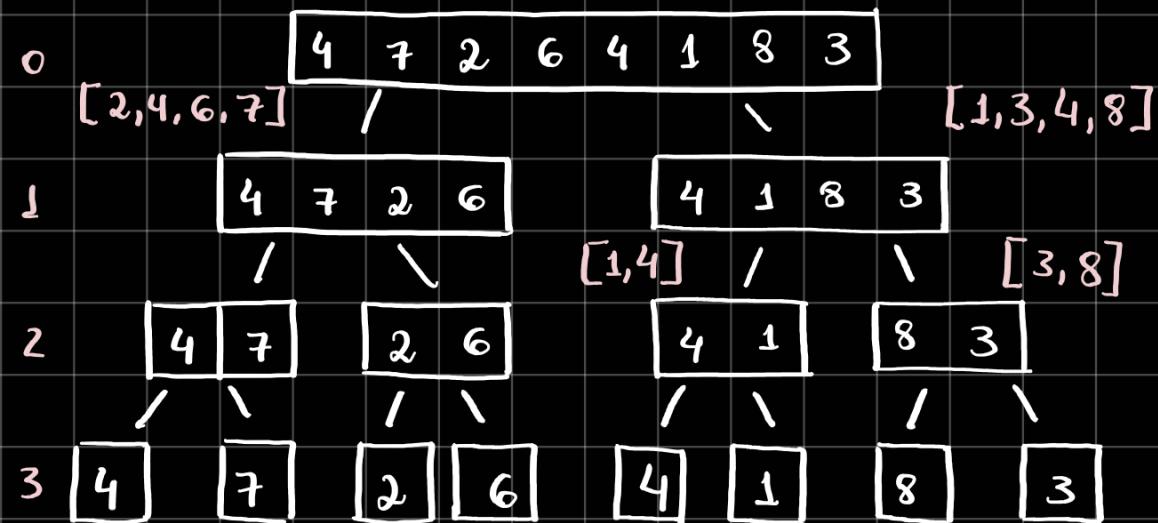
resp.pref = max(left.pref, left.soma, right.pref)

resp.suffix = max(right.suffix, right.soma, left.suffix)

return resp

$$\text{Complexidade: } T(n) = 2T\left(\frac{n}{2}\right) + C = O(n)$$

## \* Merge Sort



Na linha 2 vamos imaginar uma pilha de cartas, comparo [4, 7] e [2, 6], então 2 é menor que 4, coloco ele na pilha ordenada. Agora temos 4 na pilha esq e 6 na direita, tira ele e coloca na de cima ordenado e assim por diante

mergesort (lista, inicio=0, fim=list.length)

if (fim - inicio > 1)

meio = (fim + inicio) / 2

mergesort (lista, inicio, meio)

mergesort (lista, meio, fim)

merge (lista, inicio, meio, fim)  $\Rightarrow$  junta e faz a ordenação da lista

merge (lista, inicio, meio, fim)

left = lista[inicio:meio]

right = lista[meio:fim]

top-left = 0, top-right = 0

for k = inicio até fim

if top-left >= left.length

} Verifica se extrapoleou o limite de cartas

```

lista[k] = right[top-right]
top-right = top-right + 1
else if top-right >= right.length
    lista[k] = left[top-left]
    top-left = top-left + 1
else if left[top-left] < right[top-right]
    lista[k] = left[top-left]
    top-left = top-left + 1
else
    lista[k] = right[top-right]
    top-right = top-right + 1

```

de um lado ou seja  
 se top é maior ou  
 igual a lista do seu  
 lado, se sim vai usar  
 quem está do outro lado

(confiro se o  
 topo de cartas  
 da esq é maior  
 que o da direita)

se for avança  
 depois repito  
 para a direita