# Fontes principais

1. Lima, A. C., Soluções para os problemas de soma máxima e do k-ésimo menor elemento de uma sequência usando modelo BSP/CGM, tese de doutorado em ciência da computação, Universidade Federal de Mato Grosso do Sul, 2015, Campo Grande - MS.

# Subsequência de soma máxima

## Subsequência de soma máxima

Problema: Dada uma sequência $x$ de números reais, encontrar uma subsequência contígua com a maior soma de seus elementos.

# Subsequência de soma máxima (em outras palavras)

Dada uma sequência de $n$ números reais $x = (x_1, x_2, \cdots, x_n)$. Uma subsequência contígua é qualquer intervalo $(x_i, \cdots, x_j)$ de $x$, tal que $0 \leq i \leq j \leq n$.

Por simplicidade consideraremos subsequência como subsequência contígua.

**Objetivo:** Determinar a subsequência $(x_i, \cdots, x_j)$ que possua o maior somatório $T = \sum_{k=i}^{j} x_k$

# Subsequência de soma máxima (modelo PRAM)

## Algoritmo de Perumalla e Deo

# Subsequência de soma máxima (modelo PRAM)

Dada uma sequência $Q = [q_1, q_2, \cdots, q_n]$, considere $Q_{ij}$ como sendo a subsequência $[q_i, q_{i+1}, \cdots, q_j]$, defina:

▷ $Range(q_k)$: o conjunto de todas as subsequências de soma máxima de $Q$ que incluem $q_k$.

Se todos os valores de $q_i$ são negativos, então a subsequência de soma máxima de $Q$ é definida como o menor número negativo encontrado. Pode-se redefinir este valor para zero, caso desejado.

# Subsequência de soma máxima (modelo PRAM)

Considere um elemento $q_k \in Q$, defina:

$\triangleright$ $l(q_k) = [q_1, q_2, \cdots, q_{k-1}, q_k]$ a subsequência composta pelos elementos a esquerda de $q_k$ incluindo $q_k$.

$\triangleright$ $r(q_k) = [q_k, q_{k+1}, \cdots, q_n]$ a subsequência composta pelos elementos a direita de $q_k$ incluindo $q_k$

## Subsequência de soma máxima (modelo PRAM)

Sejam

- ▷ $S_l(q_k) = [s_1, s_2, \cdots, s_k]$ a soma de sufixos de $l(q_k)$
- ▷ $P_r(q_k) = [p_k, p_{k+1}, \cdots, p_n]$ a soma de prefixos de $r(q_k)$
- ▷ $M_s^k$ o valor máximo da soma de sufixos de $S_l(q_k)$
- ▷ $M_p^k$ o valor máximo da soma de prefixos de $P_r(q_k)$

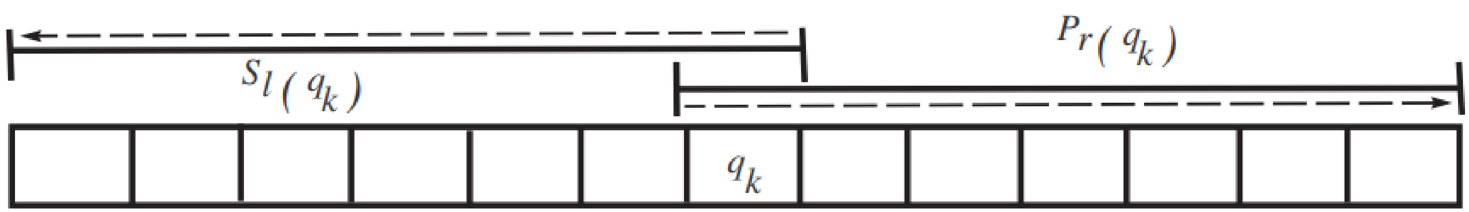
# Subsequência de soma máxima (modelo PRAM)

**Lemma 1.** *O valor máximo entre as somas de todas as subsequências que incluem $q_k$ é dado por $Max(q_k) = M_s^k + M_p^k - q_k$.*

Demonstração: Considere $SQ_{ij}$ com a soma dos elementos $Q_{ij}$.
◦ Agora considere a subsequência definida por $Q_{ab}^k$, desta forma $M_s^k = SQ_{ak}$ e $M_p^k = SQ_{kb}$, para $1 \leq a \leq k \leq b \leq n$.
◦ Considere agora que existe uma outra subsequência $Q_{a'b'}^k$ que inclui $q_k$, tal que a soma $SQ_{a'b'}^k$ é maior que a soma dada por $SQ_{ab}^k$.

# Subsequência de soma máxima (modelo PRAM)

A subsequência $Q^k_{a'b'}$ pode ser vista como duas subsequências $Q_{a'k}$ e $Q_{kb'}$, ambas incluíndo $q_k$, o valor da soma $SQ^k_{a'b'}$ pode ser escrito como $SQ^k_{a'b'} = Q_{a'k} + Q_{kb'} - q_k$.




$$\boxed{Max_{(q_k)} = M_s^k + M_p^k - q_k}$$

## Subsequência de soma máxima (modelo PRAM)

Entretanto, por definição $M_s^k = SQ_{ak} \geq SQ_{ik}$ para todo $1 \leq i \leq k$, segue assim que $SQ_{ab'} \geq SQ_{a'b'}$.

Um argumento similar pode ser aplicado para $b$ e $b'$, dado que $SQ_{ab} \geq SQ_{a'b'}$, então $SQ^k_{ab} \geq SQ^k_{a'b'}$ para todo $SQ^k_{a'b'} \in Range(q_k)$.
$\square$

# Subsequência de soma máxima (Exemplo)

| | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $Q=$ | 3 | 2 | -7 | 11 | 10 | -6 | 4 | 9 | -6 | 1 | -2 | -3 | 4 | -3 | 0 | 2 |
| $l(q_7)=$ | 3 | 2 | -7 | 11 | 10 | -6 | 4 | | | | | | | | | |
| $S_l(q_7)=$ | 17 | 14 | 12 | 19 | 8 | -2 | 4 | | | | | | | | | |
| $r(q_7)=$ | | | | | | | 4 | 9 | -6 | 1 | -2 | -3 | 4 | -3 | 0 | 2 |
| $P_r(q_7)=$ | | | | | | | 4 | 13 | 7 | 8 | 6 | 3 | 7 | 4 | 4 | 6 |

# Subsequência de soma máxima (Exemplo)

Para $M_s^7$ temos os seguintes valores:

- $M_s^7 =$ Máximo$(S_l) = 19$
- $M_p^7 =$ Máximo$(P_r) = 13$

- $M_s^7 + M_p^7 - q_7 = 19 + 13 - 4 = 28 = Max(q_7)$

# Subsequência de soma máxima (Exemplo)

Suponha que o valor $Max(q_k)$ é calculado para todo valor $q_k$ da sequência $Q$, a soma máxima entre todas as subsequências que incluem $q_k$ é computada.

Então claramente, a subsequência de soma máxima de $Q$ é composta pelos máximos destes máximos, isto é:

$MaxSeqSum =$Máximo$(Max(q_k), 1 \leq k \leq n)$.

## Subsequência de soma máxima (Algoritmo PRAM)

- $\triangleright$ Entrada: Sequência $Q[1 \cdots n]$ de números inteiros.
- $\triangleright$ Saída: Subsequência de soma máxima $Q$.

# Subsequência de soma máxima (Algoritmo PRAM)

1 Compute em paralelo as somas de prefixos de $Q$ no vetor $PSUM$.

2 Compute em paralelo as somas de sufixos de $Q$ no vetor $SSUM$.

3 Compute em paralelo o sufixo máx. de $PSUM$ no vetor $SMAX$.

4 Compute em paralelo o prefixo máx. de $SSUM$ no vetor $PMAX$.

5 **para** $i = 1$ **até** $n$ **faça em paralelo**

  (a) $M_s[i] := PMAX - SSUM[i] + Q[i]$

  (b) $M_p[i] := SMAX - PSUM[i] + Q[i]$

  (c) $M[i] := M_s[i] + M_p[i] - Q[i]$

6 Localize a sequência contígua de máximos de $M$ e armazene os valores correspondentes de $Q$ em $MSQ$

7 Imprima a subsequência de soma máxima $MSQ$.

# Subsequência de soma máxima (Algoritmo PRAM)

Entrada: $Q = \{3, 2, -7, 11, 10, -6, 4, 9, -6, 1, -2, -3, 4, -3, 0, 2\}$

| $Q =$ | 3 | 2 | -7 | 11 | 10 | -6 | 4 | 9 | -6 | 1 | -2 | -3 | 4 | -3 | 0 | 2 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $PSUM =$ | 3 | 5 | -2 | 9 | 19 | 13 | 17 | 26 | 20 | 21 | 19 | 16 | 20 | 17 | 17 | 19 |
| $SSUM =$ | 19 | 16 | 14 | 21 | 10 | 0 | 6 | 2 | -7 | -1 | -2 | 0, | 3 | -1 | 2 | 2 |
| $SMAX =$ | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 26 | 21 | 21 | 20 | 20 | 20 | 19 | 19 | 19 |
| $PMAX =$ | 19 | 19 | 19 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 | 21 |
| $M =$ | 26 | 26 | 26 | **28** | **28** | **28** | **28** | **28** | 23 | 23 | 22 | 22 | 22 | 21 | 21 | 21 |

- Soma máxima de uma subsequência: 28
- Subsequência de soma máxima: elementos do intervalo $[4, 8]$

# Subsequência de soma máxima (Algoritmo PRAM)

Custo do algoritmo

- passos 1 e 2: $O(\lg n)$ usando algoritmos de soma de prefixos e soma de sufixos.

- passos 3 e 4: podem ser resolvidos por $O(n/\lg n)$ processadores em uma máquina EREW em tempo $O(\lg n)$ utilizando variações de algoritmos de soma de prefixos e soma de sufixos.

- passo 5: $O(1)$

- passo 6: $O(\lg n)$

Tempo total: $O(\lg n)$

# Subsequência de soma máxima (BSP/CGM)

# Subsequência de soma máxima (BSP/CGM)

Utilizando as ideias de Perumalla e Deo é possível desenvolver um algoritmo paralelo BSP/CGM para o problema.

# Subsequência de soma máxima (BSP/CGM)

Entrada:

1. Um conjunto de $P$ processadores;
2. O número $i$ que rotula cada processador $p_i \in P$, onde $1 \leq i \leq P$;
3. Uma sequência $Q$ de inteiros.

Saída:

1. O vetor $M[1 \cdots n]$ de inteiros com todas as subsequências disjuntas de soma máxima.
2. O valor da soma máxima de $M$.

1 Utilize o conjunto de processadores $P$ e o vetor $Q$ para obter as somas de prefixos de $Q$ no vetor $PSUM$.
2 Utilize o conjunto de processadores $P$ e o vetor $Q$ para obter as somas de sufixos de $Q$ no vetor $SSUM$.
3 $SMAX := \text{sufixos\_maximos}(PSUM)$.
4 $PMAX := \text{prefixos\_maximos}(SSUM)$.
5 Processador $p_1$ envia $n/p$ elementos de cada vetor $Q$, $PSUM$, $SSUM$, $SMAX$, $PMAX$ para cada processador $p_i \in P$.
6 Cada processador $p_i$ obtém os vetores locais
$LocalM_s := PMAX(n/p) - SSUM(n/p) + Q(n/p)$
$LocalM_p := SMAX(n/p) - PSUM(n/p) + Q(n/p)$
$LocalM := LocalM_s(n/p) + LocalM_s(n/p) - Q(n/p)$
7 Cada processador $p_i$ envia o vetor $LocalM$ para o processador $p_1$, que computa o array:
$M = [LocalM_{p_1,1} \cdots LocalM_{p_1,n/p} \cdots LocalM_{p_p,1} \cdots LocalM_{p_p,n/p}]$
8 Encontre o maior valor númerico em $M$ (soma máxima)

## Algoritmo de sufixos máximos (BSP/CGM)

Entrada: Vetor $PSUM[1 \cdots n]$ de inteiros;

Saída: Vetor $SMAX[1 \cdots n]$ de inteiros;

## Algoritmo sufixos_maximos

1 Processador $p_1$ envia $n/p$ elementos de $PSUM$ para cada processador $p_i \in P$.

2 Em cada processador $p_i$, uma operação de propagação de máximos deve ser executada em cada vetor $PSUM(n/p)$. A operação é iniciada no elemento de índice $n/p$ e executa até o elemento de índice 1. Ao final o maior elemento está na primeira posição.

3 Por fim, uma operação de propagação de máximos também é executada entre os processadores, sendo que cada processador $p_i$ irá trabalhar com os processadores $p_k$ onde $k > i$.

## Algoritmo de prefixos máximos (BSP/CGM)

Entrada: Vetor $SSUM[1 \cdots n]$ de inteiros;

Saída: Vetor $PMAX[1 \cdots n]$ de inteiros;

Algoritmo prefixos_maximos

1 Processador $p_1$ envia $n/p$ elementos de $SSUM$ para cada processador $p_i \in P$.

2 Em cada processador $p_i$, uma operação de propagação de máximos deve ser executada em cada vetor $SSUM(n/p)$. A operação é iniciada no elemento de índice 1 e executa até o elemento de índice $n/p$. Ao final o maior elemento está na posição $n/p$.

3 Por fim, uma operação de propagação de máximos também é executada entre os processadores, sendo que cada processador $p_i$ irá trabalhar com os processadores $p_k$ onde $k < i$.

# Algoritmo de prefixos máximos (BSP/CGM)

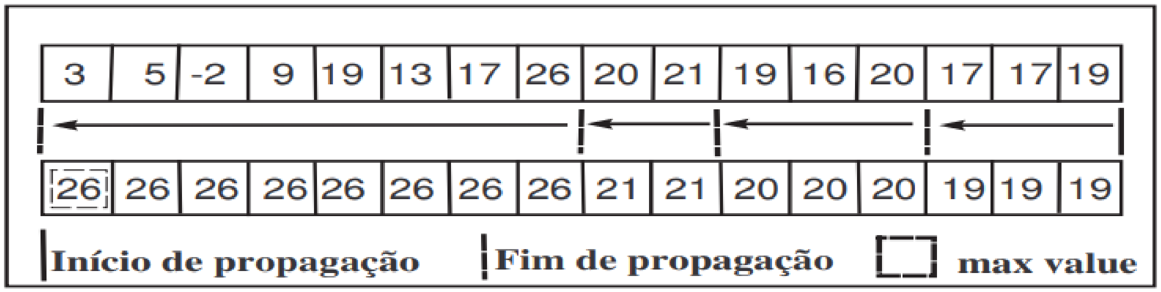
Nestes algoritmos duas operações são executadas, entretanto ambas consistem basicamente de propagação de valores máximos.

- A primeira ocorre nos vetores locais, em que os valores maiores substituem os valores menores.

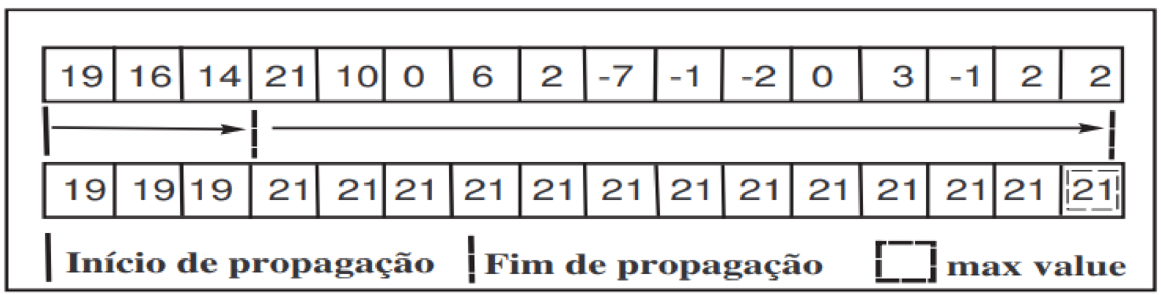
- A segunda ocorre entre os processadores, o valor máximo de cada processador é enviado para o processador subsequente ou precedente.

# Subsequência de soma máxima (BSP/CGM)

## Propagação de valores máximos locais



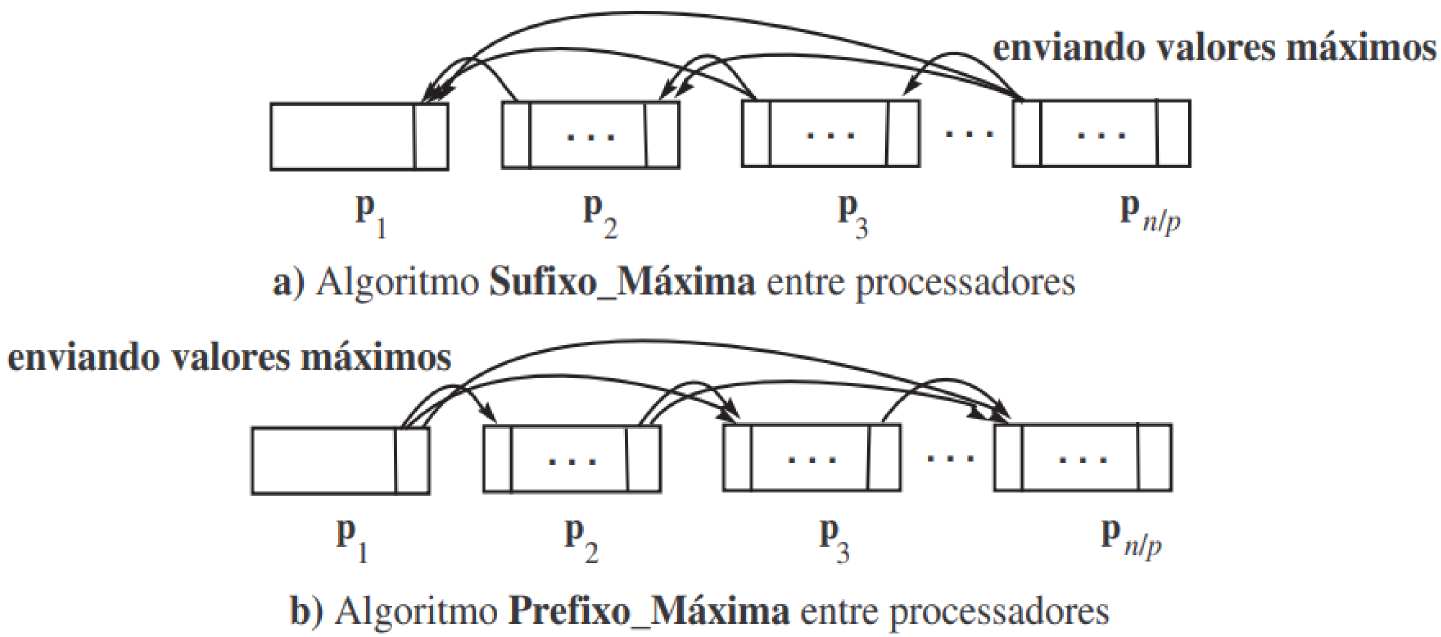

a) Algoritmo **Sufixo_Máxima** em cada processador




b) Algoritmo **Prefixo_Máxima** em cada processador

# Subsequência de soma máxima (BSP/CGM)

Propagação de valores máximos entre processadores




(b) Operações globais.

# Subsequência de soma máxima (BSP/CGM)

Complexidade do algoritmo de subsquência de soma máxima:

◦ passos 1 e 2: pode ser computado utilizando $p$ processadores em tempo $O(n/p)$ e com número constante de rodadas de comunicação.

◦ passos 3 e 4: a invocação dos algoritmos sufixos_maximos e prefixos_maximos, utiliza, respectivamente, $p$ processadores e consome tempo $O(n/p)$ com número constante de rodada de comunicação.

◦ passos 5 a 7: podem ser computados utilizando $p$ processadores em tempo $O(n/p)$ com número constante de rodada de comunicação.

- passo 8: algoritmo de redução para o máximo utiliza $p$ processadores em tempo $O(n/p)$ com número constante de rodada de comunicação.

Tempo total: $O(n/p)$, com $p$ processadores e número constante de rodadas de comunicação.

# Subsequência de soma máxima - MPI

# Subsequência de soma máxima - MPI

```
#include<mpi.h>
#include<stdio.h>
#include<stdlib.h>
#include<limits.h>

#define MASTER 0
#define n 16

#define max(a, b) ((a) > (b)) ? (a) : (b)
#define min(a, b) ((a) < (b)) ? (a) : (b)
int main(int argc, char *argv[]) {

   int id, p;
```

```
MPI_Init(&argc,&argv);
MPI_Comm_size(MPI_COMM_WORLD,&p);
MPI_Comm_rank(MPI_COMM_WORLD,&id);


// id de processos
int *pids = (int*) malloc(p*sizeof(int));

// guarda os ids de processo na ordem reversa
for(int i = 0; i < p; i++){
   pids[i] = p - i - 1;
}

MPI_Group group_world;
MPI_Group reverse_group;
```

```
MPI_Comm reverse_comm;

MPI_Comm_group(MPI_COMM_WORLD, &group_world);
MPI_Group_incl(group_world, p, pids, &reverse_group);
MPI_Comm_create(MPI_COMM_WORLD, reverse_group, &reverse_comm);

int input[n] = {3, 2, -7, 11, 10, -6, 4, 9, -6,
                1, -2, -3, 4, -3, 0, 2};
int q[n];
int psum[n] = {0};
int ssum[n] = {0};
int pmax[n] = {0};
int smax[n] = {0};

int r = (n/p);
```

```
// Distritua a entrada
MPI_Scatter(input, r, MPI_INT, q, r, MPI_INT, MASTER,
                                              MPI_COMM_WORLD);
MPI_Barrier(MPI_COMM_WORLD);

// calcular a soma de prefixos de Q no array PSUM
psum[0] = q[0];
for (int i = 1; i < r; i++){
   psum[i] = q[i] + psum[i-1];
}
```

```c
// calcular a soma de sufixo de Q no array SSUM
ssum[r - 1] = q[r - 1];
for (int i = r - 2; i >= 0; i--){
  ssum[i] = q[i] + ssum[i + 1];
}

int psum_reduce = psum[r - 1];
int ssum_reduce = ssum[0];
int psum_recv = 0, ssum_recv = 0;


MPI_Exscan(&psum_reduce, &psum_recv, 1, MPI_INT, MPI_SUM,
                                             MPI_COMM_WORLD);
MPI_Exscan(&ssum_reduce, &ssum_recv, 1, MPI_INT, MPI_SUM,
                                               reverse_comm);
```

```
if (id > MASTER){
   for (int i = 0; i < r; i++) {
      psum[i] += psum_recv;
   }
}


if (id < p - 1){
   for (int i = r - 1; i >= 0; i--){
      ssum[i] += ssum_recv;
   }
}
```

```
// Compute em paralelo o sufixo máximo de PSUM no vetor SMAX
smax[r - 1] = psum[r - 1];
for (int i = r - 2; i >= 0; i--){
   smax[i]  = max(psum[i], smax[i + 1]);
}

// Compute em paralelo o prefixo máximo de SSUM no vetor PMAX
pmax[0] = ssum[0];
for (int i = 1; i < r; i++){
   pmax[i] = max(ssum[i], pmax[i - 1]);
}
int pmax_reduce = pmax[r - 1];
int smax_reduce = smax[0];
int pmax_recv, smax_recv;
```

```
MPI_Exscan(&pmax_reduce, &pmax_recv, 1, MPI_INT, MPI_MAX,
                                                MPI_COMM_WORLD);
MPI_Exscan(&smax_reduce, &smax_recv, 1, MPI_INT, MPI_MAX,
                                                reverse_comm);
if (id > MASTER) {
   for (int i = 0; i < r; i++) {
      pmax[i] = max(pmax_recv, pmax[i]);
   }
}

if (id < p - 1) {
   for (int i = r - 1; i >= 0; i--) {
      smax[i] = max(smax_recv, smax[i]);
   }
}
```

```c
int m[n] = {0};
int ms[n] = {0};
int mp[n] = {0};
int maior_soma = INT_MIN;

for (int i = 0; i < r; i++){
   ms[i] = pmax[i] - ssum[i] + q[i];
   mp[i] = smax[i] - psum[i] + q[i];
   m[i] = ms[i] + mp[i] - q[i];
   maior_soma = max(maior_soma, m[i]);
   printf("m[%d] = %d \n", id * r + i, m[i]);
}
```

```
    int maior_global;
    MPI_Allreduce(&maior_soma, &maior_global, 1, MPI_INT, MPI_MAX,
                                                        MPI_COMM_WORLD);
    printf("maior global: %d\n", maior_global);
    MPI_Finalize();
    return 0;
}
```

Fim