



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática



Computação Paralela

Laboratório de Busca e Ordenação

Nome: Giulia Barbieri Quagliano

TIA: 42013070

Nome: Jenifer Mathias dos Santos

TIA: 32092946

Merge Sort

Para implementar o MergeSort de forma paralela, foi criada uma área paralela utilizando "#pragma omp parallel". O código dentro dessa área será executado pelos threads, criadas conforme os núcleos disponíveis ou alguma configuração de ambiente. A variável "meio" será compartilhada entre os threads, o que é permitido através de "shared(int meio)". Dentro da área paralela também temos a área de código que terá conjuntos de seções independentes, definida por "#pragma omp sections". Dentro dela definimos cada conjunto com "#pragma omp section". Para resolver a condição de corrida, utilizamos "#pragma omp barrier" para definir um ponto de sincronização entre os threads, onde a área paralela aguarda a finalização dos threads antes de continuar.



```
1 2 3 4 5 6 7 8 9 10 11 12
jenifer.santos@MacBook-Pro-de-Zupper mergesort % ./mergesort
Digite o tamanho do vetor: 11

Digite um número: 2

Digite um número: 1

Digite um número: 4

Digite um número: 5

Digite um número: 7

Digite um número: 2

Digite um número: 8

Digite um número: 1

Digite um número: 9

Digite um número: 22

Digite um número: 50

Sorted array:
1 1 2 2 4 5 7 8 9 22 50 %
```



```
zsh: no such file or directory: ./mergesort
jenifer.santos@MacBook-Pro-de-Zupper mergesort % gcc -o mergesort MergeSort.c
jenifer.santos@MacBook-Pro-de-Zupper mergesort % ./mergesort
Digite o tamanho do vetor: 10

Digite um número: 3

Digite um número: 2

Digite um número: 7

Digite um número: 3

Digite um número: 4

Digite um número: 2

Digite um número: 5

Digite um número: 1

■ Structure Digite um número: 6
Favorites Digite um número: 7

Sorted array:
1 2 2 3 3 4 5 6 7 7 10
```



```
jenifer.santos@MacBook-Pro-de-Zupper mergesort % ./mergesort
Digite o tamanho do vetor: 12

Digite um número: 3
Digite um número: 2
Digite um número: 1
Digite um número: 4
Digite um número: 6
Digite um número: 2
Digite um número: 88
Digite um número: 9
Digite um número: 57
Digite um número: 50
Digite um número: 26
Digite um número: 21

Sorted array:
1 2 2 3 4 6 9 21 26 50 57 88 %
```



```
jenifer.santos@MacBook-Pro-de-Zupper mergesort % ./mergesort
Digite o tamanho do vetor: 9

Digite um número: 3

Digite um número: 4

Digite um número: 2

Digite um número: 7

Digite um número: 78

Digite um número: 104

Digite um número: 650

Digite um número: 986

Digite um número: 5

Sorted array:
2 3 4 5 7 78 104 650 986 %
```



```
jennifer.santos@macbook-pro-de-zuppi:mergesort % ./mergesort.c
Digite o tamanho do vetor: 13

Digite um número: 45
Digite um número: 66
Digite um número: 73
Digite um número: 234
Digite um número: 567
Digite um número: 1056
Digite um número: 789
Digite um número: 2456
Digite um número: 432
Digite um número: 563
Digite um número: 789
Digite um número: 15
Digite um número: 1

Sorted array:
1 15 45 66 73 234 432 563 567 789 789 1056 2456 24
```

Selection Sort

Para implementar o SelectionSort de modo paralelo, foi criada uma Struct para armazenar o índice do vetor e o menor elemento. Depois foi usado o reduction customizado para manter uma cópia privada do resultado do menor valor para cada execução. A variável privada que faz com que isso seja possível foi declarada na função 'selectionSort'. Dessa forma, foi evitado que o código funcionasse de modo sequencial.



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática



```
jenifer.santos@MacBook-Pro-de-Zupper selectionsort % gcc -o selectionsort SelectionSort.c
jenifer.santos@MacBook-Pro-de-Zupper selectionsort % ./selectionsort

Digite o tamanho do vetor: 12

Digite o número: 4

Digite o número: 3

Digite o número: 76

Digite o número: 54

Digite o número: 33

Digite o número: 2

Digite o número: 8

Digite o número: 12

Digite o número: 56

Digite o número: 789

Digite o número: 0

Digite o número: 24

Sorted array :
0 2 3 4 8 12 24 33 54 56 76 789 0
```



UNIVERSIDADE PRESBITERIANA MACKENZIE

Faculdade de Computação e Informática



```
jenifer.santos@MacBook-Pro-de-Zupper selectionsort % ./selectionsort

Digite o tamanho do vetor: 10

Digite o número: 45

Digite o número: 65

Digite o número: 66

Digite o número: 78

Digite o número: 22

Digite o número: 34

Digite o número: 5

Digite o número: 61

Digite o número: 92

Digite o número: 89

Sorted array :
5 22 34 45 61 65 66 78 89 92 %
```




```
Commit
Digite o tamanho do vetor: 13
Digite o número: 2
Digite o número: 11
Digite o número: 3
Digite o número: 4
Digite o número: 55
Digite o número: 67
Digite o número: 54
Digite o número: 98
Digite o número: 1
Digite o número: 45
Digite o número: 67
Structure
Digite o número: 88
Digite o número: 98
Favorites
Sorted array :
1 2 3 4 11 45 54 55 67 67 88 98 98
```



```
Digite o tamanho do vetor: 9
```

```
Digite o número: 35
```

```
Digite o número: 62
```

```
Digite o número: 12
```

```
Digite o número: 14
```

```
Digite o número: 45
```

```
Digite o número: 65
```

```
Digite o número: 87
```

```
Digite o número: 79
```

```
Digite o número: 90
```

```
Sorted array :
```

```
12 14 35 45 62 65 79 87 90 %
```



UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática



```
terminal: Local x Local (2) x 4 v
Digite o tamanho do vetor: 15

Digite o número: 32
Digite o número: 44
Digite o número: 65
Digite o número: 78
Digite o número: 93567
Digite o número: 8123
Digite o número: 99
Digite o número: 543
Digite o número: 7183
Digite o número: 478
Digite o número: 34
Digite o número: 657
Digite o número: 567
Digite o número: 830
```

Continuação 5º vetor

```
Digite o número: 478
Digite o número: 34
Digite o número: 657
Digite o número: 567
Digite o número: 830
Digite o número: 0

Sorted array :
0 32 34 44 65 78 99 478 543 567 657 830 7183 8123 93567 %
```



Linear Search

Para implementar o Linear Search de modo paralelo, usado o openmp, dessa vez para encontrar o valor requerido, percorrendo cada item do vetor. Foi usado o for de forma paralela # pragma omp parallel for, para que o processo de busca ocorresse em um tempo mais rápido do que fazendo uma busca linear sem usar paralelismo.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 int linearSearch(int* A, int n, int tos);
6
7 int main(){
8     int number, iter = 0, find;
9     int* Arr;
10
11     scanf("%d", &number); Arr = (int *)malloc( number *
12     sizeof(int));
13
14     for(; iter<number; iter++){
15         scanf("%d", &Arr[iter]);
16     }
17
18     scanf("%d", &find);
19     printf("\nTo find: %d\n", find);
20 }
21
```

```
> cc -fopenmp main.c
main.c: In function 'main':
main.c:12:2: warning: ignoring return value of 'scanf'
' declared with attribute 'warn_unused_result' [-Wunused-
sed-result]
12 | scanf("%d", &number); Arr = (int *)malloc( nu
mber * sizeof(int));
    | ~~~~~^~~~~~
main.c:15:3: warning: ignoring return value of 'scanf'
' declared with attribute 'warn_unused_result' [-Wunused-
sed-result]
15 |     scanf("%d", &Arr[iter]);
    |     ~~~~~^~~~~~
main.c:18:2: warning: ignoring return value of 'scanf'
' declared with attribute 'warn_unused_result' [-Wunused-
sed-result]
18 |     scanf("%d", &find);
    |     ~~~~~^~~~~~
> ./a.out
4
5
3
1
2
3
To find: 3
Found at 1
>
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 int linearSearch(int* A, int n, int tos);
6
7 int main(){
8     int number, iter = 0, find;
9     int* Arr;
10
11     scanf("%d", &number); Arr = (int *)malloc( number *
12     sizeof(int));
13
14     for(; iter<number; iter++){
15         scanf("%d", &Arr[iter]);
16     }
17
18     scanf("%d", &find);
19     printf("\nTo find: %d\n", find);
20 }
21
```

```
> ./a.out
5
1
2
3
4
5
1
To find: 1
Found at 0
>
```



```
Files | main.c | a.out | Console | Shell | + | : |
Find a file |
main.c |
a.out |
> f main
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 int linearSearch(int* A, int n, int tos);
6
7 int main(){
8
9     int number, iter =0, find;
10    int* Arr;
11
12    scanf("%d", &number);Arr = (int *)malloc( number *
    sizeof(int));
13
14    for(; iter<number; iter++){
15        scanf("%d", &Arr[iter]);
16    }
17
18    scanf("%d", &find);
19    printf("\nTo find: %d\n", find);
20
21    > ./a.out
22    6
23    2
24    4
25    8
26    1
27    3
28    8
29
30    To find: 8
31    Found at 3
32    >
Line 14: Col 29 | History
```

```
Files | main.c | a.out | Console | Shell | + | : |
Find a file |
main.c |
a.out |
> ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 int linearSearch(int* A, int n, int tos);
6
7 int main(){
8
9     int number, iter =0, find;
10    int* Arr;
11
12    scanf("%d", &number);Arr = (int *)malloc( number *
    sizeof(int));
13
14    for(; iter<number; iter++){
15        scanf("%d", &Arr[iter]);
16    }
17
18    scanf("%d", &find);
19    printf("\nTo find: %d\n", find);
20
21    > ./a.out
22    6
23    7
24    9
25    8
26    2
27    4
28    1
29    4
30
31    To find: 4
32    Found at 4
33    >
Line 1: Col 1 | History
```

```
Files | main.c | a.out | Console | Shell | + | : |
Find a file |
main.c |
a.out |
> ...
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <omp.h>
4
5 int linearSearch(int* A, int n, int tos);
6
7 int main(){
8
9     int number, iter =0, find;
10    int* Arr;
11
12    scanf("%d", &number);Arr = (int *)malloc( number *
    sizeof(int));
13
14    for(; iter<number; iter++){
15        scanf("%d", &Arr[iter]);
16    }
17
18    scanf("%d", &find);
19    printf("\nTo find: %d\n", find);
20
21    > ./a.out
22    8
23    7
24    9
25    4
26    5
27    6
28    2
29    1
30    3
31    2
32
33    To find: 2
34    Found at 5
35    >
Line 1: Col 1 | History
```