TASK3 – Seminarios 2

Ana Flávia Dias Rodrigues- 561288

Os testes foram feitos ultilizando o código com os valores de 1000, 10000 e 100000.

Questão 3) Com os threads quanto maior o valor de um for maior será significância do trabalho.

Questão4) Houve um melhoramento de desempenho e ospeedupaumentou de acordo com o aumento dos threads.

Questão 5) Oalgoritmoselectionsortordenou de forma mais eficiente e dessa forma o tempo de execução foi menor e com issouma melhoria de desempenhodamáquina.

```
#include <stdio.h>
#include <omp.h>
#include <time.h>
#include <stdlib.h>
#include <math.h>

struct Task3{
   int val;
   int index;
}
void selectionSort(int array[],int n) {
   for (int i = 0; i < (n - 1); i++) {
      int indice = i;
      for (int j = (i + 1); j < n; j++){
         if (array[indice] > array[j]){
            indice = j;
         }//fimif
      }//fim for j

      int auxiliar = array[indice];
      array[indice] = array[i];
      array[i] = auxiliar;
   }//fim for i
```

```
}//fimselectionSort


#pragma omp declare reduction(maximum : struct Task3 : omp_out = omp_in.val > omp_out.val ? omp_in : omp_out)


void selectionsort(int arr[], int size,int threds) {
    //#pragma omp parallel
    omp_set_num_threads(threds);

    for (int i = size - 1; i > 0; --i) {
        struct Compare max;
        max.val = arr[i];
        max.index = i;

        #pragma omp parallel for //reduction(maximum:max)

        for (int j = i - 1; j >= 0; j--) {
            #pragma omp critical
        }//fim for

        if (arr[j] > max.val) {
            max.val = arr[j];
            max.index = j;
        }//fimif
    }//fim for i
    int tmp = arr[i];
    arr[i] = max.val;
    arr[max.index] = tmp;
}//fimselectionsort

int main() {
    int MAX;
    printf("TAMANHO\tSerial\t1\t2\t4\t8\t16\t32\64\128\n");
```

```
for(int max=1000;max<100000;max*=10){
    double s1,s2,s4,s8,s16,s32,e1,e2,e4,e8,e16,e32,e64,e128;

    MAX=max;
    int x[MAX];
    printf("%d\t",max);

    fflush(stdout);

    for (int i = 0; i < MAX; i++) {
        x[i] = rand() % 1000;
    } //fim for

    double tempiS=clock();
    selectionSort(x,MAX);
    double tempis=clock();
    double proctime = ((double)tempis - tempiS) / CLOCKS_PER_SEC;

    printf("%6.4f\t",proctime);

    for(int n=1; n<=32; n*=2) {

        for (int i = 0; i < MAX; i++) {
            x[i] = rand() % 1000;
        } //fim for i

        double tempoI = clock();
        selectionsort(x, MAX,n);
        double tempoF = clock();
        double procTime = ((double)tempoF - tempoI) / CLOCKS_PER_SEC;

        if(n==1) {
            s1= (proctime/procTime);
            e1= s1/n;
```

```
    }else if(n==2) {
        s2=(proctime/procTime);
        e2=s2/n;
    }else if(n==4){
        s4=(proctime/procTime);
        e4=s4/n;
    }else if(n==8){
        s8=(proctime/procTime);
        e8=s8/n;
    }else if(n==16){
        s16=(proctime/procTime);
        e16=s16/n;
    }else if(n==32){
        s32=(proctime/procTime);
        e32=s32/n;
    }else if(n==64){
        s64=(proctime/procTime);
        e64=s64/n;
    }else if(n==128){
        s128=(proctime/procTime);
        e128=s128/n;
    }

    printf("%6.4f\t", procTime);
}
printf("\n\t\t");

printf("%6.4f\t",s1);
printf("%6.4f\t",s2);
printf("%6.4f\t",s4);
printf("%6.4f\t",s8);
printf("%6.4f\t",s16);
printf("%6.4f\t\n",s32);
printf("%6.4f\t\n",s64);
```

```
        printf("%6.4f\t\n",s128);
        printf("\t\t%6.4f\t",e1);
        printf("%6.4f\t",e2);
        printf("%6.4f\t",e4);
        printf("%6.4f\t",e8);
        printf("%6.4f\t",e16);
        printf("%6.4f\t\n",e32);
        printf("%6.4f\t\n",e64);
        printf("%6.4f\t\n",e128);
    }
    return 0;
}//fimmain
```

Tamanho	Seq	1	2	4	8	16	32	64	128
1000	0.00	0.00	0.00	0.02	0.06	0.12	0.24	0.48	0.96
		0.96	1.00	0.14	0.04	0.02	0.01	0.01	0.00
10000	0.20	0.20	0.14	1.09	0.66	1.28	3.05	2.56	4.10
		1.04	1.43	0.19	0.31	0.16	0.07	0.05	0.01
100000	25.52	23.29	15.05	59.32	27.60	34.65	45.98	52.69	63.56
		1.10	1.70	0.43	0.92	0.74	0.55	0.65	0.35