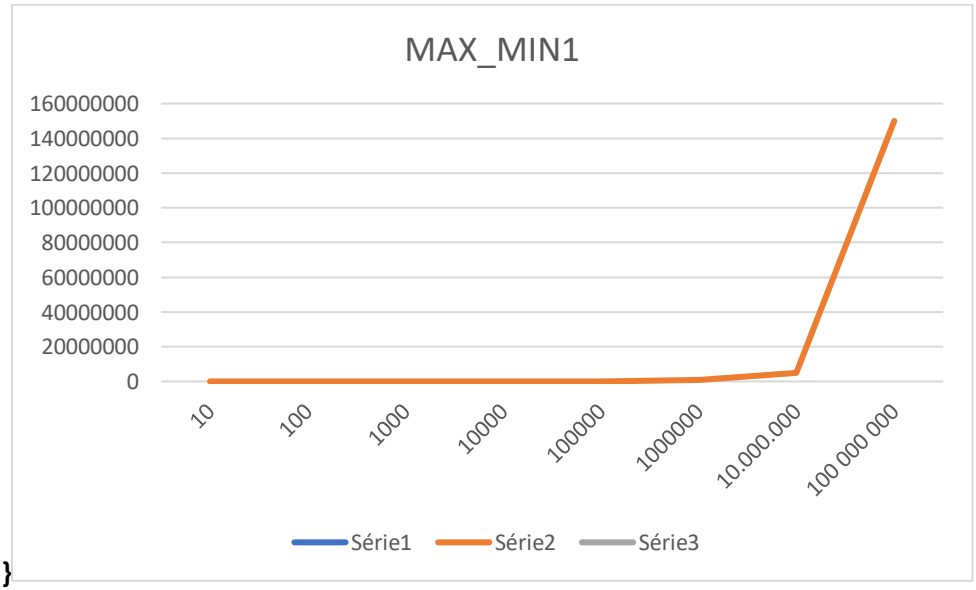
O processo de testes foi feito em 3 algoritmos (max_min1, max_min2, max_min3) sendo que todos possuem a função main() iguais para o funcionamento de contagem de tempo seja igual. Também o número de posições do vetor foram iguais para observar o tempo em todos os algoritmos crescendo o tamanho em 10^n até o valor de 100000000.

Em relação aos testes feitos percebe-se que dependendo da entrada não haverá muita diferença entre o tempo de execução. Mas, a partir do momento em que as entradas começam a ter números maiores a lógica do algoritmo max_min2 começa ser mais eficaz que os outros como podemos ver nos gráficos. Além disso, o algoritmo max_min2 dependendo da entrada ele possui o melhor caso pois possuir menos laços de repetição. O algoritmo max_min3 possui o médio caso se as entradas forem maiores. E o algoritmo max_min1 possui o pior caso pois repete funções para um mesmo valor. Códigos abaixo:

MAX_MIN1:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define tam 100000000

void max_min(int a[])
{
   int i ,max=a[0], min=a[0];

   for (i=1; i<10; i++)
   {
      if ((a[i])>max)
      {
         max=a[i];
         //printf("\nmax:%i",max);
      }

      if((a[i])<min)
      {
         min=a[i];
         //printf("\nmin:%i",min);
```

```
        }
    }
}

int main()
{
    int *vet= (int*)malloc(sizeof(int)*tam);
    int i,j;
    clock_t t;
    srand(time(NULL));
    //geração aleatório dos valores do vetor
    for(i=0; i<tam; i++)
    {
        vet[i]=rand()%1000000;
    }
    t=clock();
    max_min(vet);
    t= clock()-t;
    printf("\nTempo de execucao: %lf",((double)t)/((CLOCKS_PER_SEC/1000))); //conversão para double
    return 0;
```

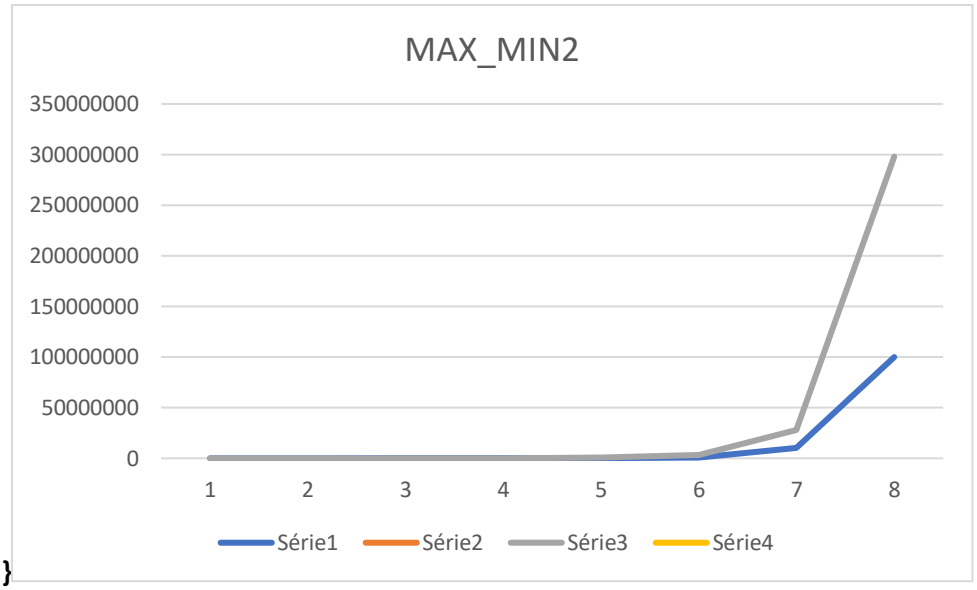



```
}
```

MAX_MIN2:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define tam 100000000

/*int randon_integer(int low, int high)
{
   int k;
   srand((unsigned)time(NULL));
   k=(rand()% high)+low;
   return k;
}
*/
void max_min(int a[])
{
   int i ,max=a[0], min=a[0];

   for (i=1; i<tam; i++)
   {
      if(a[i]>max)
      {
         max=a[i];
       //  printf("\nMax:%i",max);
      }
      else
      {
         min=a[i];
        // printf("\n\nMin:%i");
```

```
        }


    }
}


int main()
{
    int *vet= (int*)malloc(sizeof(int)*tam);
    int i,j;
    clock_t t;
    srand(time(NULL));
    //geração aleatório dos valores do vetor
    for(i=0; i<tam; i++)
    {
        vet[i]=rand()%1000000;
    }
    t=clock();
    max_min(vet);
    t= clock()-t;
    printf("\nTempo de execucao: %lf",((double)t)/((CLOCKS_PER_SEC/1000))); //conversão
para double
    return 0;
```



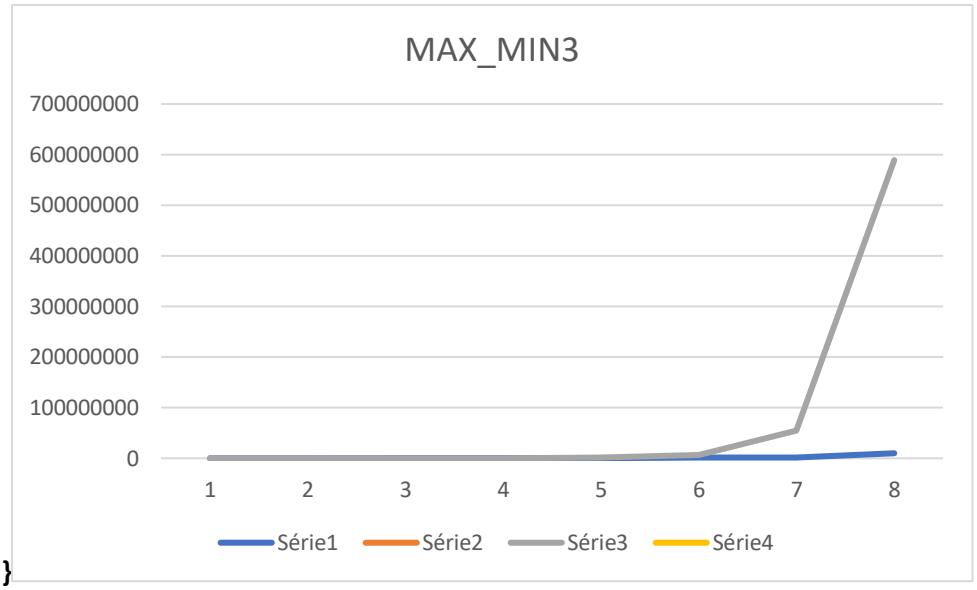
}

MAX_MIN3:

```
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#define n 100000000



void max_min(int a[])
{
   int i ,max, min,fimdoanel;

   if((n&1)>0)
   {
      a[n]=a[n-1];
      fimdoanel =n;
   }

   else fimdoanel= n-1;
```

```
if(a[0]>a[1])
{
   max=a[0];
   min=a[1];
}

else {
   max =a[1];
   min=a[0];
}
i=3;

while (i<=fimdoanel)
{
   if (a[i-1]>a[i])
   {
      if(a[i-1]>max)
      {
         max=a[i-1];
      }
   }
   if(a[i]<min)
   {
      min =a[i];
   }
   else
   {
      if(a[i-1]<min)
      {
         min=a[i-1];
```

```
        }
        if(a[i]>max)
        {
           max=a[i];
        }

      }
      i+=2;
   }
   printf("\nMax:%i",max);
   printf("\nMin:%i",min);
}

int main()
{
   int *vet= (int*)malloc(sizeof(int)*n);
   int i,j;
   clock_t t;
   srand(time(NULL));
   //geração aleatório dos valores do vetor
   for(i=0; i<n; i++)
   {
      vet[i]=rand()%1000000;
   }
   t=clock();
   max_min(vet);
   t= clock()-t;
   printf("\nTempo de execucao: %lf",((double)t)/((CLOCKS_PER_SEC/1000))); //conversão
para double
   return 0;
```


MAX_MIN3
700000000
600000000
500000000
400000000
300000000
200000000
100000000
0
1
2
3
4
5
6
7
8
Série1
Série2
Série3
Série4

}