

Complexidade

1 – Para cada um dos programas seguintes:

- Efectue a análise de complexidade.
- Calcule analiticamente quanto o tempo de execução deverá aumentar caso a dimensão de N seja aumentada $4x$.

- a)

```
for(long i=0;i<n;i++)  
    for(long j=0;j<n;j++)  
        soma++;
```
- b)

```
for (long i=0;i<n; i++)  
    soma++;
```
- c)

```
for (long i=0;i<n;i+=2)  
    soma++;
```
- d)

```
for(long i=0;i<1000;i++)  
    for(long j=0;j<n;j++)  
        soma++;
```
- e)

```
for(long i=0;i<n;i++)  
    soma++;  
for(long j=0;j<n;j++)  
    soma++;
```
- f)

```
if(n>20000) n=20000;  
for(long i=0;i<n;i++)  
    for(long j=0;j<n;j++)  
        soma++;
```
- g)

```
for(long i=0;i<n;i++)  
    for(long j=0;j<n*n;j++)  
        soma++;
```
- h)

```
for(long i=0;i<n;i++)  
    for(long j=0;j<i;j++)  
        soma ++;
```

```
i) for(long i=0;i<n*n;i++)  
    for(long j=0;j<i;j++)  
        soma ++;
```

```
j)  for(long i=1;i<n;i*=2)  
      soma++;
```

2 – Relativamente a cada uma das alíneas da pergunta anterior, implemente o código e verifique o tempo de execução (use *System.nanoTime()* para obter um valor *long* com o tempo actual em *ns* ($1\text{ ns} = 10^{-9}$ segundos)). Compare as suas previsões com os resultados obtidos. Deve procurar encontrar valores de *n* que resultem em tempos de execução significativos (alguns segundos). Pode definir uma constante *long* acrescentando ao valor constante um *L*; por exemplo *long n=127343734L*. Dependendo do computador, poderá ser difícil cumprir isso para algumas alíneas.

3 – Considere uma matriz de *N* x *N*, na qual os números estão colocados por ordem ascendente, tanto nas linhas como nas colunas. Pretende-se criar e implementar um algoritmo que verifica se um determinado número está ou não na matriz. Sendo assim,

a) Crie, implemente e teste um algoritmo de complexidade $O(N^2)$

b) Crie, implemente e teste um algoritmo de complexidade $O(N)$

4 – Implemente e teste os três algoritmos de cálculo da máxima sequência contígua estudados nas aulas teóricas. Analise o seu desempenho e verifique se está alinhado com a respectiva classe de complexidade.

5 – Um elemento maioritário num *array* de dimensão *N* é um valor que aparece mais do que $N/2$ vezes. Por exemplo, o elemento maioritário de [3,3,4,2,4,4,2,4,4] é 4, enquanto que [3,3,4,2,4,4,2,4] não tem um elemento maioritário. Construa, analise e implemente um algoritmo para resolver este problema. (**Desafio:** *É possível, [mas não muito óbvio], resolver o problema em tempo linear...*)