**EXERCÍCIO 1**

**No algoritmo abaixo descrito, informe a quantidade de vezes que a Linha 1 será executada, em tempo de execução e em função de n.**

**public static int Func(int n) {**

**int m = 0;**

**for (int i=1; i <= n; i++)**

**for (int j = 1; j <= n; j++ ) {**

**m = m + 1; // Linha 1**

**}**

**}**

**return m;**

**}**

***Resolução: Linha 1 é executada n² vezes.***

**EXERCÍCIO 2**

**No algoritmo abaixo descrito, informe a quantidade de vezes que a Linha 1 será executada, em tempo de execução e em função de n.**

**public static int Func(int n) {**

**int m = 0;**

**for (int i=2; i < n; i++)**

**for (int j = 2; j < n; j++ ) {**

**m = m + 1; // Linha 1**

**}**

**}**

**return m;**

**}**

***Resolução: Linha 1 é executada ou vezes.***

**EXERCÍCIO 3**

**No algoritmo abaixo descrito, informe a quantidade de vezes que a Linha 1 será executada, em tempo de execução e em função de n.**

**public static int Func(int n) {**

**int i = 4;**

**int m = 0;**

**while (i <= n) {**

**m = m + 1; // Linha 1**

**i = i + 2;**

**}**

**return m;**

**}**

***Resolução: Linha 1 é executada vezes.***

**EXERCÍCIO 4**

**No algoritmo abaixo descrito, informe a quantidade de vezes que a Linha 1 será executada, em tempo de execução e em função de n.**

**public static int Func(int n){**

**int i = 1;**

**int m = 0;**

**while (i <= n) {**

**m = m + 1; // Linha 1**

**i = i \* 2;**

**}**

**return m;**

**}**

***Resolução: Linha 1 é executada* log2(n) + 1 *vezes.***

**EXERCÍCIO 5**

**No algoritmo abaixo descrito, informe a quantidade de vezes que a Linha 1 será executada, em tempo de execução e em função de n.**

**public static int Func(int n) {**

**int m = 0;**

**for (int i=1; i <= n; i++) {**

**for (int j = i; j <= n; j++ ) {**

**m = m + 1; // Linha 1**

**}**

**}**

**return m;**

**}**

***Resolução: Linha 1 é executada vezes.***

**EXERCÍCIO 6**

**Vamos supor que estamos comparando implementações de ordenação por inserção e ordenação por intercalação na mesma máquina. Para entradas de tamanho *n*, a ordenação por inserção é executada etapas, enquanto a ordenação por intercalação é executada em etapas. Para que valores de *n* a ordenação por inserção supera a ordenação por intercalação?**

***Resolução: Para valores de n maiores do que 1 e menores do que 26 o algoritmo de inserção é melhor.***

**EXERCÍCIO 7**

**Qual é o menor valor de n tal que um algoritmo cujo tempo de execução é funciona mais rápido que um algoritmo cujo tempo de execução é na mesma máquina?**

***Resolução: A partir do valor 15 de n o algoritmo funciona mais rápido do que o de complexidade .***

**EXERCÍCIO 8**

**Considere dois algoritmos A e B com complexidades respectivamente iguais a e . Qual o maior valor de n, para o qual o algoritmo B é mais eficiente que o algoritmo A?**

***Resolução: Para valores de n menores do que 32 o algoritmo B é melhor do que o algoritmo A.***

**EXERCÍCIO 9**

**Considere dois computadores C1 e C2 que executam e operações por segundo e dois algoritmos de ordenação A e B que necessitam e operações com entrada de  
tamanho n, respectivamente. Qual o tempo de execução de cada algoritmo em cada um dos computadores C1 e C2 para ordenar elementos?**

***Resolução:***

|  |  |  |
| --- | --- | --- |
|  | ***A*** | ***B*** |
| ***C1*** | ***500000000*** | ***320*** |
| ***C2*** | ***5000000*** | ***3,2*** |

**EXERCÍCIO 10**

**Um algoritmo tem complexidade . Num certo computador, num tempo t, o algoritmo resolve um problema de tamanho 25. Imagine agora que se tenha disponível um computador 100 vezes mais rápido. Qual o tamanho máximo de problema que o mesmo algoritmo resolve no mesmo tempo t no computador mais rápido?**

***Resolução: Seria possível realizar um problema de tamanho 31 no mesmo tempo.***