**Practica de Laboratorio de Objetos y Abstracción de Datos**

**Ciencia de la Computación - UNSA**

**Pregunta 1.** Modificar el algoritmo de ordenación por inserción para que, en lugar de ordenar todos los números, apenas sean ordenados k los primeros dígitos.

El objetivo es conseguir que apenas los k menores valores.

Responda:

* ¿Cuál es el número de comparaciones en el mejor caso?
* ¿Cuál es el número de comparaciones en el peor caso?

|  |
| --- |
| void insertionSort(vector<int>& v) {  for (int i = 0; i < v.size(); i++) {  for (int j = i – 1; j >= 0; j--) {  if (v[j] <= v[j + 1]) break;  swap(v[j], v[j + 1]);  }  }  } |

**Pregunta 2.** ¡Cual es la complejidad Big O de los siguientes fragmentos de código?

|  |  |  |
| --- | --- | --- |
| a)  for (int i = 0; i < 500; i++) {  cout << i << endl;  } | b)  int sum = 0; for (int i = 0; i < n; i++) {  for (int j = 0; j < 100; j++) {  for (int k = 0; k < j\*j; k++) {  sum++;  }  } } | c)  for (int i = 0; i < n; i++) {  for (int j = 0; j < n\*2; j++) {  cout << j << endl;  } } |

**Pregunta 3**.

|  |
| --- |
| int raiseToPower(int m, int n) {  if (n == 0) return 1;  return m \* raiseToPower(m, n – 1);  } |

* ¿Cuál es la complejidad Big-O de la función anterior, escrita en términos de m y n? Puede suponer que toma tiempo O(1) multiplicar dos números.
* Si tarda 1μs en computar raiseToPower(100, 100), aproximadamente ¿cuánto tiempo se tarda en computar raiseToPower(200, 10000)?

**Pregunta 4.**

|  |
| --- |
| int raiseToPower(int m, int n) {  if (n == 0) {  return 1;  } else if (n % 2 == 0) {  int halfPower = raiseToPower(m, n / 2);  return halfPower \* halfPower;  } else {  int halfPower = raiseToPower(m, n / 2);  return m \* halfPower \* halfPower;  }  } |

* ¿Cuál es la complejidad big-O de la función anterior, escrita en términos de m y n? Puede suponer que toma tiempo O(1) multiplicar dos números.

**Pregunta 5.** Escribir una función denominada particionable que toma un vector de ints y devuelve true si es posible dividir los ints en dos grupos de manera que cada grupo tenga la misma suma.

Por ejemplo, el vector {1,1,2,3,5} se puede dividir en {1,5} y {1,2,3}. Sin embargo, el vector {1,4,5,6} no se puede dividir en dos.

bool particionable(vector<int>& nums) { ...