**Algoritmica Práctica 2: Divide y Vencerás**

**Nour Eddine El Alaoui - W456114**

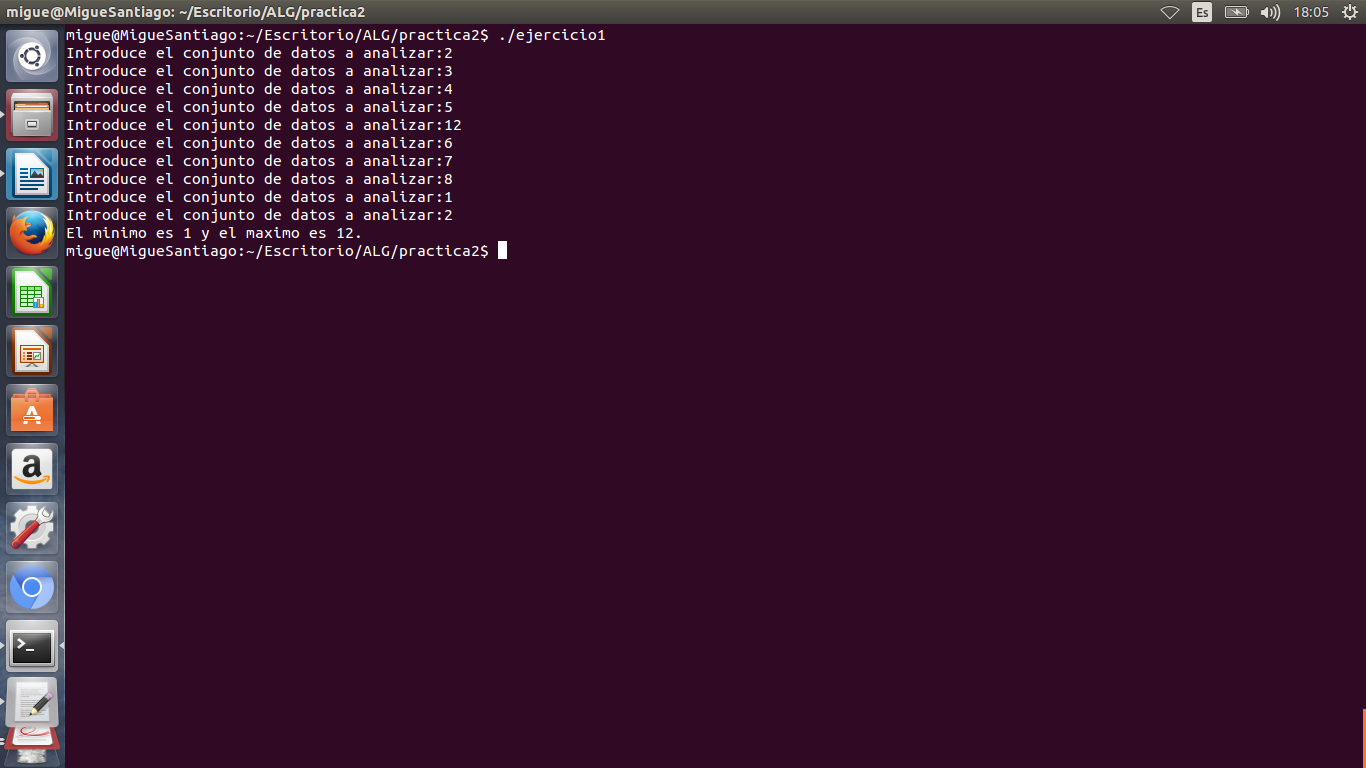
**Miguel Santiago Cervilla - 76438234B**

**Ejercicio 1 :**

**Para el ejercicio hemos usado :**

1) algortimo pair<int,int> Max\_Min(const vector<int> & v) usando recursividad.

Resultado de vector analizando máximo y mínimo:



**Eficiencia :**

Eficiencia del algoritmo usado:

El algoritmo utilizado, debe dividir el vector inicial en dos partes que a su vez se van dividiendo de manera recursiva, para llegar a al caso base.

En el mejor de los caso, el conjunto a analizar se corresponderá con el caso baso, que es el vector de un solo dato, el cual su máximo y mínimo corresponderá con él mismo.

En otro caso, el vector deberá dividir en 2 subvectores dicho vector, que a su vez se irán dividiendo en 2 hasta llegar al caso base.

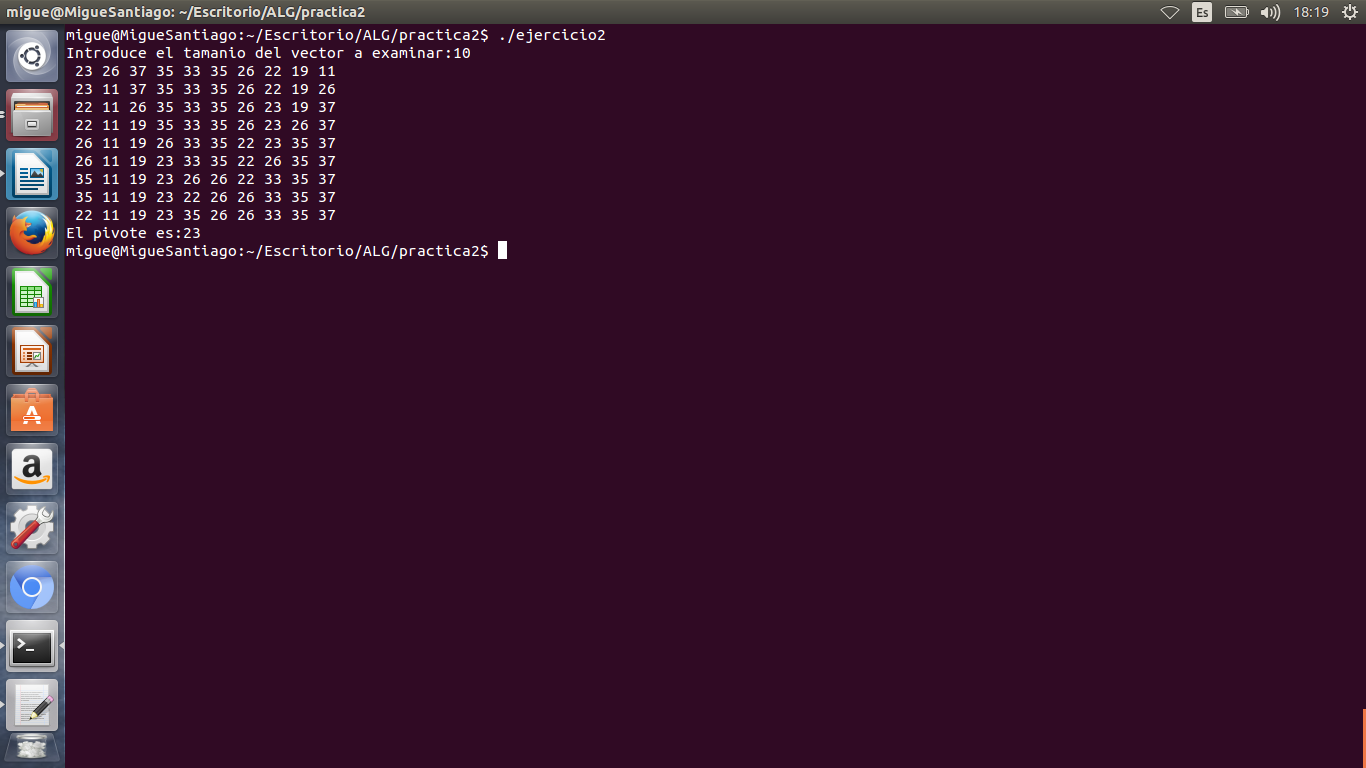
La eficiencia del algoritmo es n log (n/2).

**Ejercicio 2 : (incompleto)**

Para este ejercicio hemos usado el método de pivote proporcionado en la teoría haciendo unas cuantas modificaciones.

void pivote(vector <int> &datos,int i,int j)

Resultado al ejercutar el algoritmo pivote modificado sobre un conjunto de datos aleatorio.



**Ejercicio 4 :**

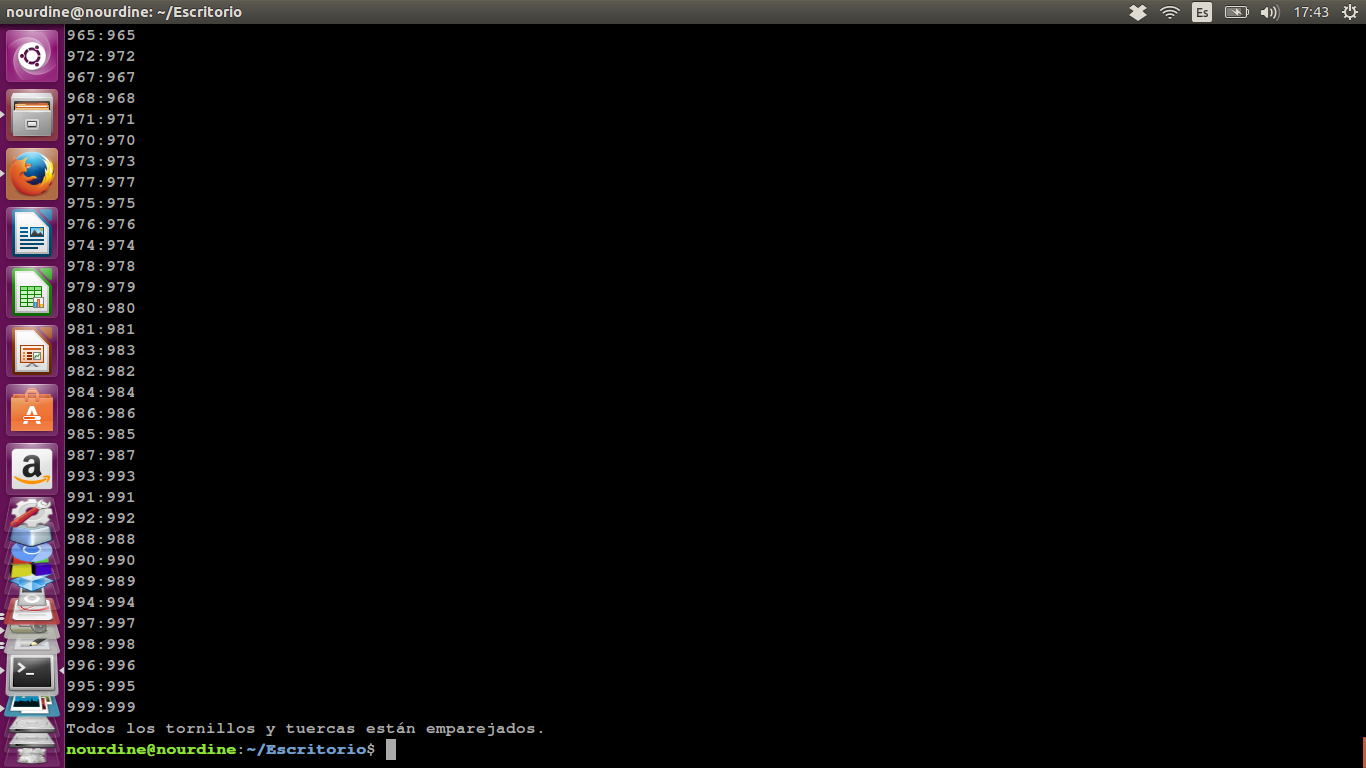
* **Para el ejercicio hemos usado :**

1. algoritmoClasicoTornilloTuerca(VectorTornillos,VectorTuercas,Inicio,Final);

2) Divide y Vencerás con QuickSort.

3)quickSortTornilloTuerca<Umbral>(VectorTornillos,VectorTuercas,Inicio,Final);

Resultado: Vectores de entrada modificados siendo emparejados.



* **Eficiencia :**
* Eficiencia del Algoritmo Clásico :

El algoritmo clásico debe conseguir formar parejas de tornillo-tuerca con las restricciones dadas. La estrategia a seguir es realizar un intento de emparejamiento seleccionando uno por uno todos los tornillos y eligiendo tuercas que no hayan sido emparejadas aún.

El número de intentos necesarios en el peor de los casos tras enfrentar a dos vectores de tornillos/tuercas del mismo tamaño, n, es:

Y cuyo orden de complejidad resultante es :

**O ()**

* Eficiencia del Algoritmo Divide y Vencerás :

En la resolución por DyV, se generan 2 subproblemas que logran dividir el tamaño del problema en 2. Además, en la descomposición/combinación existe una carga total de 3n.

**T(n) = , n**

**2T () + 3n, n>n0**

Entonces el orden De Complejidad :

**O(n)=n log2 n**