**Universidad de San Buenaventura**

**Facultad ingeniería de sistemas**



**Taller 4 Corte 3**

**Análisis de algoritmos**

**Presenta:**

Juan Felipe Hurtado Villani

Cristian Apraez

Samuel Martínez

**Punto 1:**

Para comenzar con este punto, primero planteare la teoría y luego hare un pequeño ejemplo antes del desarrollo del punto:

The stooge sort es un algoritmo recursivo de clasificación u ordenamiento, definido de la siguiente manera:

Paso 1: si el valor en el índice 0 es mayor que el valor en el último índice, cámbielos.

Paso 2: recursivamente,

a) Stooge clasifica las 2/3 iniciales de la matriz.

b) Stooge clasifica los últimos 2/3 de la matriz.

c) Stooge clasifica las 2/3 iniciales nuevamente para confirmar.

**Ejemplo:**

Entrada: 2 4 5 3 1

Salida: 1 2 3 4 5

Explicación:

Inicialmente, intercambie 2 y 1 siguiendo el paso 1 anterior.

1 4 5 3 2

Ahora, ordena de forma recursiva 2/3 de los elementos iniciales.

1 4 5 3 2

1 3 4 5 2

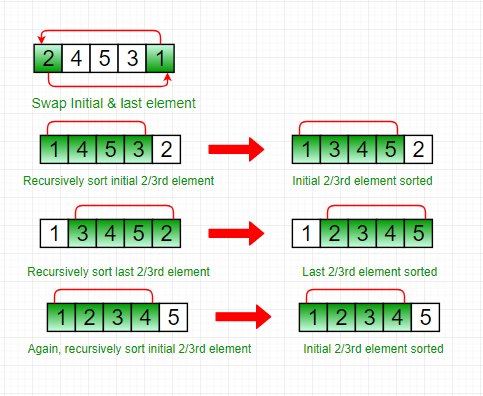
Luego, ordena de forma recursiva los últimos 2/3 de los elementos.

1 3 4 5 2

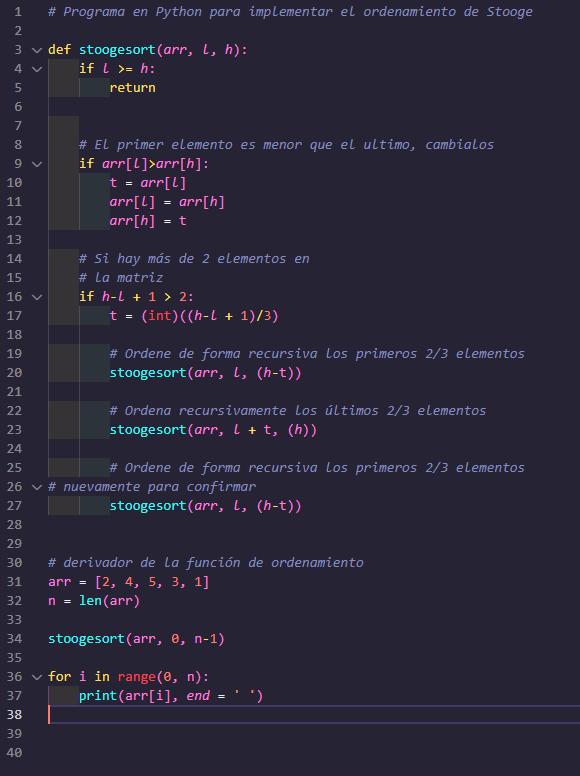
1 2 3 4 5

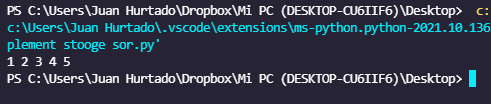
Nuevamente, ordene las 2/3 iniciales de los elementos para confirmar que los datos finales están ordenados.

1 2 3 4 5

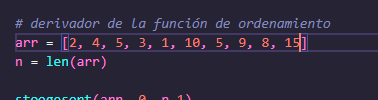


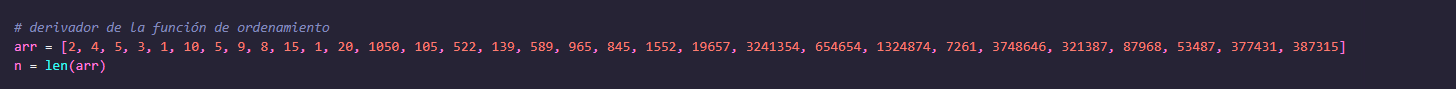
Explicado este ejemplo, pasaremos al código:

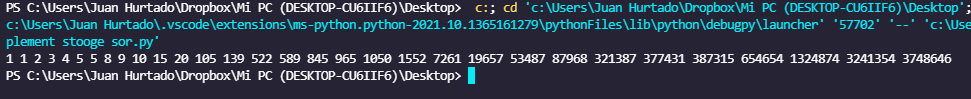




Ahora con los datos que nos piden, haremos dos ejemplos:







La complejidad del tiempo de ejecución del stooge sort se puede escribir como,

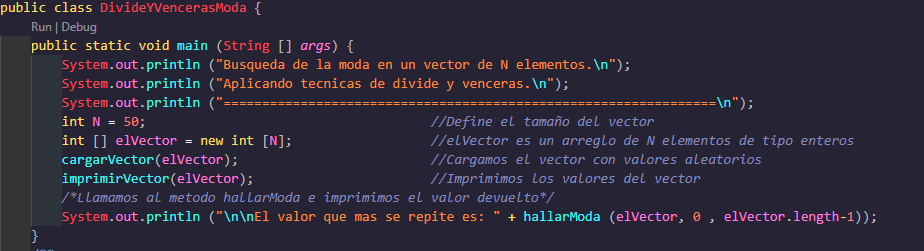
T (n) = 3T (3n / 2) +? (1)

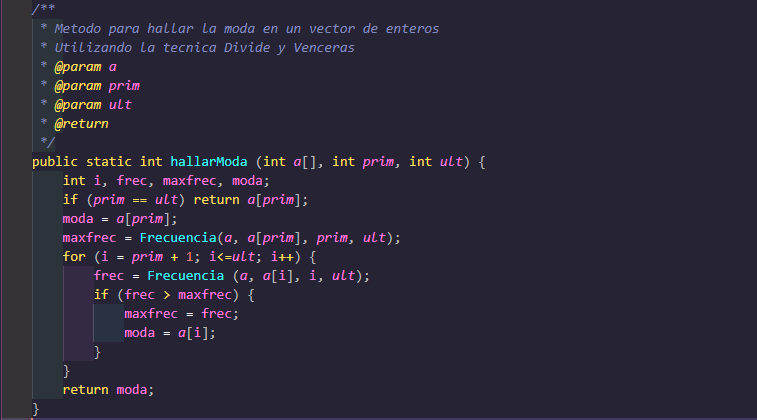
La solución de la recurrencia anterior es , por lo tanto, es más lenta que incluso el bubble sort (n ^ 2).

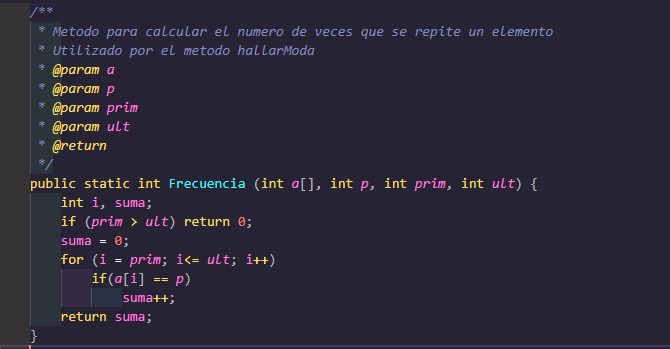
**Punto 2:**

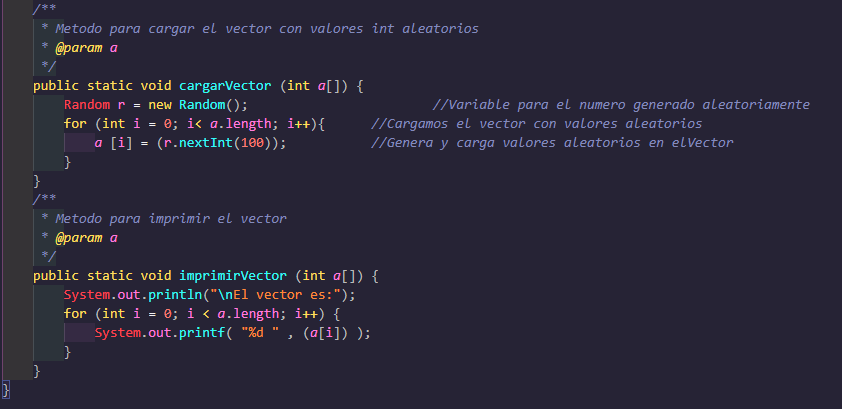
Para el divide y vencerás tenemos que:

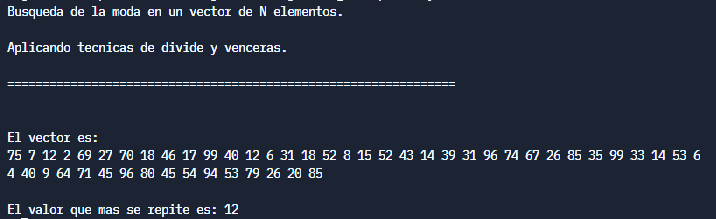
1. Definir el tamaño del vector
2. Nuestro componente “elVector” es un arreglo de N elementos de tipo entero.
3. Cargamos el vector con valores aleatorios.
4. Imprimimos los valores del vector.
5. Llamamos al método “hallarModa” e imprimimos el valor devuelto por el mismo.

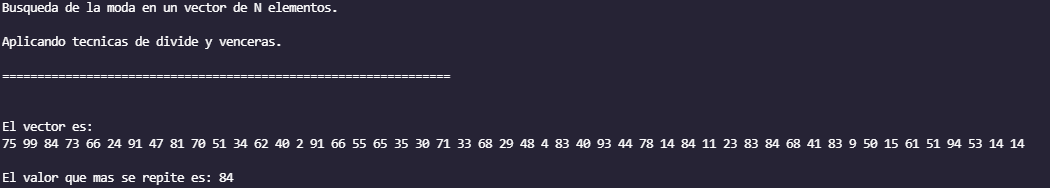












 La ecuación de recurrencia es la siguiente:

        |

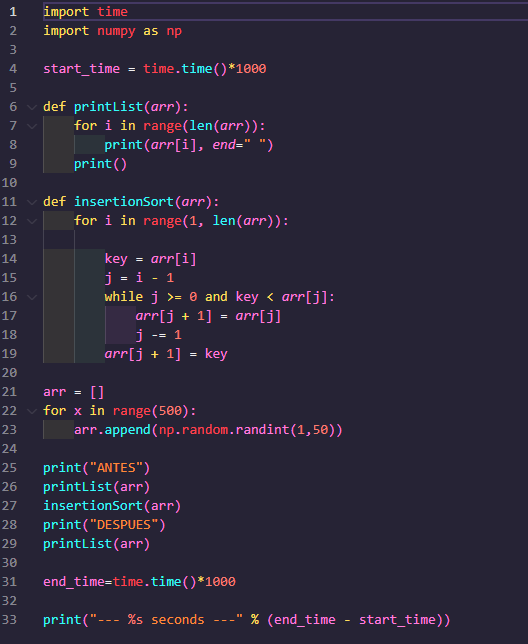
   T(n) <= | O (1) si n =1                //if (prim == ult) return a[prim];

        | T(n/2) + O(n)

        |

 La complejidad de la función Frecuencia es O(n), entonces la complejidad para calcular la moda está en orden de O(n^2)

**Punto 3:** 



MERGE-SORT

|  |  |  |  |
| --- | --- | --- | --- |
| Entrada | Tiempo Real (seg) | Complejidad(θ(nLog(n))) | Constantes |
| 10 | 0.0 | 33.21928 | 0.0 |
| 10 | 0.0 | 33.21928 | 0.0 |
| 10 | 0.0 | 33.21928 | 0.0 |
| 50 | 0.0 | 282.19281 | 0.0 |
| 50 | 0.0 | 282.19281 | 0.0 |
| 50 | 0.0 | 282.19281 | 0.0 |
| 100 | 8.001953125 | 664.38562 | 0.012044139 |
| 100 | 0.0 | 664.38562 | 0.0 |
| 100 | 0.0 | 664.38562 | 0.0 |
| 500 | 7.998046875 | 4482.89214 | 0.001784126 |
| 500 | 8.062744140625 | 4482.89214 | 0.001798558 |
| 500 | 8.1982421875 | 4482.89214 | 0,001828784 |

INSERTION-SORT

|  |  |  |  |
| --- | --- | --- | --- |
| Entrada | Tiempo Real (seg) | Complejidad(O(n^2)) | Constantes |
| 10 | 0.0 | 100 | 0.0 |
| 10 | 0.0 | 100 | 0.0 |
| 10 | 0.99951171875 | 100 | 0.00999511 |
| 50 | 0.992431640625 | 2500 | 0.000396972 |
| 50 | 1.0029296875 | 2500 | 0.000401178 |
| 50 | 0.995361328125 | 2500 | 0.000398144 |
| 100 | 1.09716796875 | 10000 | 0.000109716 |
| 100 | 2.0 | 10000 | 0.0002 |
| 100 | 0.997802734375 | 10000 | 0.0000997802 |
| 500 | 16.125244140625 | 250000 | 0.00006450097 |
| 500 | 17.06494140625 | 250000 | 0.00006825976 |
| 500 | 15.08056640625 | 250000 | 0.000060322265 |