TEC | Tecnológico de Costa Rica

Tarea Corta #1 | Implementación y Análisis de Algoritmos

Análisis de Algoritmos | IC-3002

Profesor:

Ing. Joss Pecou Johnson

Grupo 60

Pertenece a:

Alice Arias Salazar | 2023104639

Deywenie Smith Gregory | 2024096722

Hyldia Thomas Hodgson | 2023395892

Centro Académico de Limón

Semestre II | 2025

Contenido

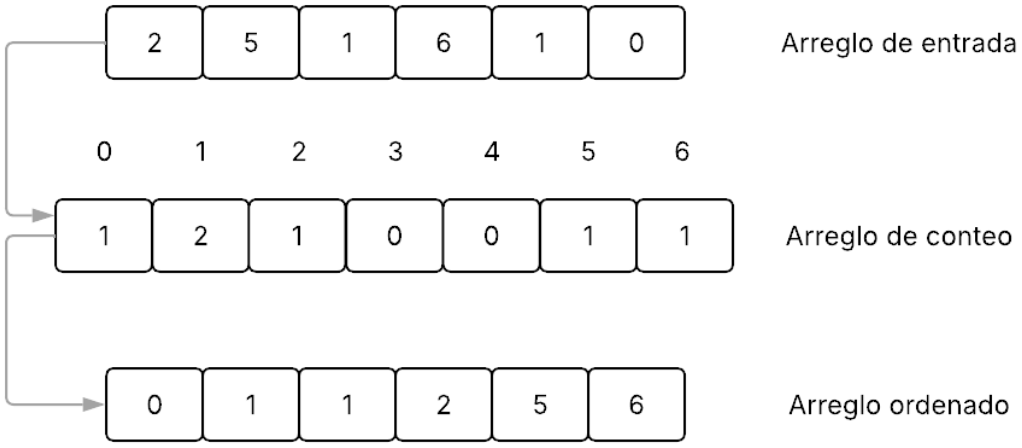
Parte 1: Implementación y análisis de ordenamiento

Descripción del algoritmo:

El algoritmo de ordenamiento implementado es Counting Sort. Este algoritmo permite ordenar números enteros sin la necesidad de realizar comparaciones. Se cuenta el número total de elementos con cada valor distinto y luego se utiliza este recuento para determinar las posiciones de cada valor clave en la salida.

Si se tiene un arreglo de n elementos desordenados, se siguen los siguientes pasos:

1) Se encuentra el valor máximo de la lista
2) Crea una lista de conteo, está tendrá el tamaño del valor máximo + 1.
3) Se cuenta cuantas veces aparece cada número en el arreglo
4) Reconstruye la lista ordenada utilizando la lista de conteo.



Análisis de complejidad

El algoritmo Counting Sort recorre todos los elementos de arreglo. Para esto utiliza un bucle en donde se cuentan las apariciones de cada valor. Luego, se realiza un segundo donde se reconstruye la lista. Esto hace que el algoritmo posea una complejidad de O(n+k), donde n representa el tamaño de la entrada y k es el rango de los valores. Esto se puede reescribir a O(n).

Complejidad Temporal:

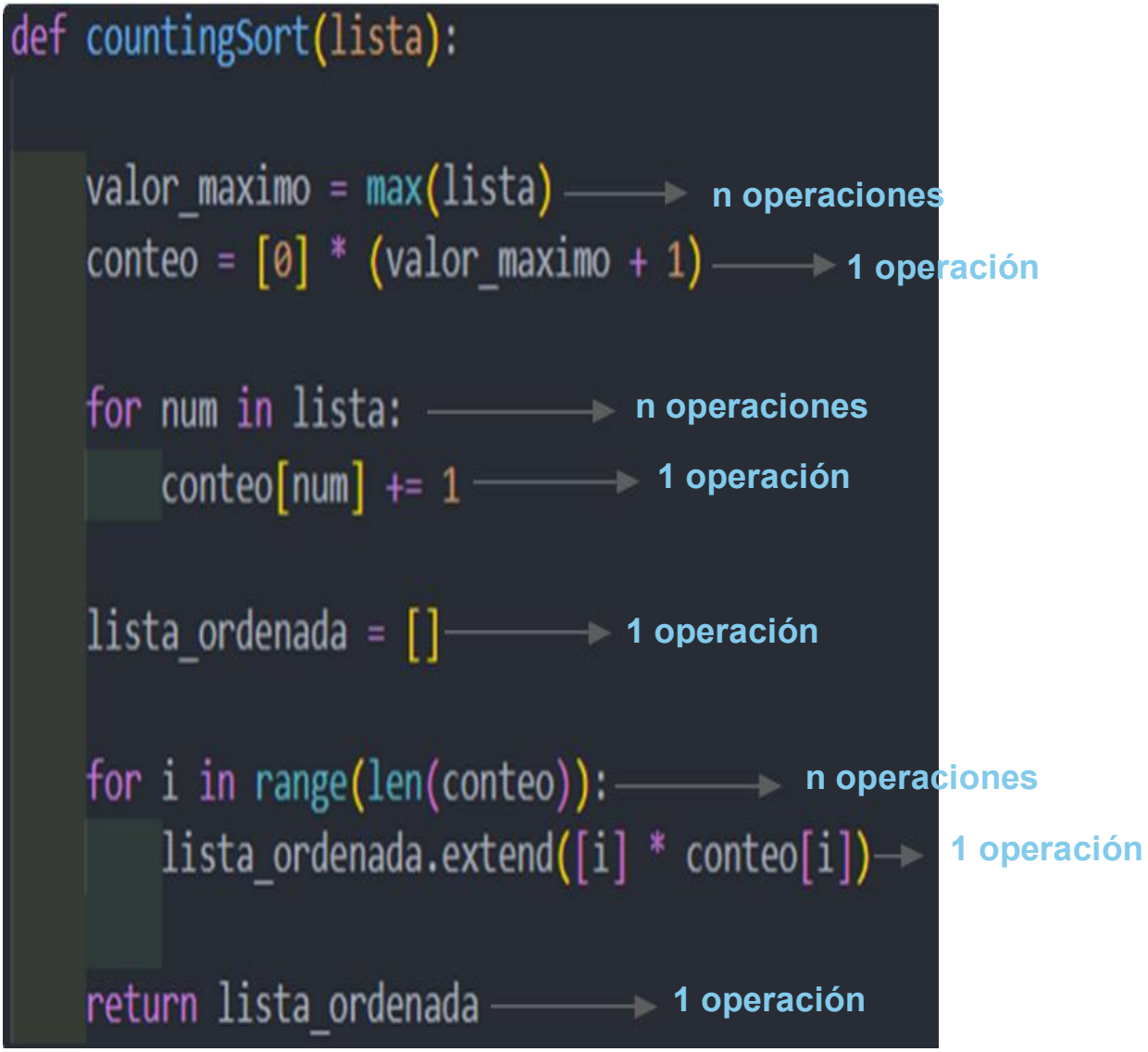
- **Peor caso:** El peor caso posee una complejidad lineal, O(n).

- **Mejor caso:** La complejidad lineal del mejor caso es igual a la del peor caso, Ω(n).
- **Caso medio:** El caso medio es θ(n).

Complejidad Espacial:

- Al necesitar un arreglo auxiliar, este algoritmo posee una complejidad espacial de O(n).

Función T(n):



T(n) = n + 1 + n + 1 + 1 + n + 1 + 1

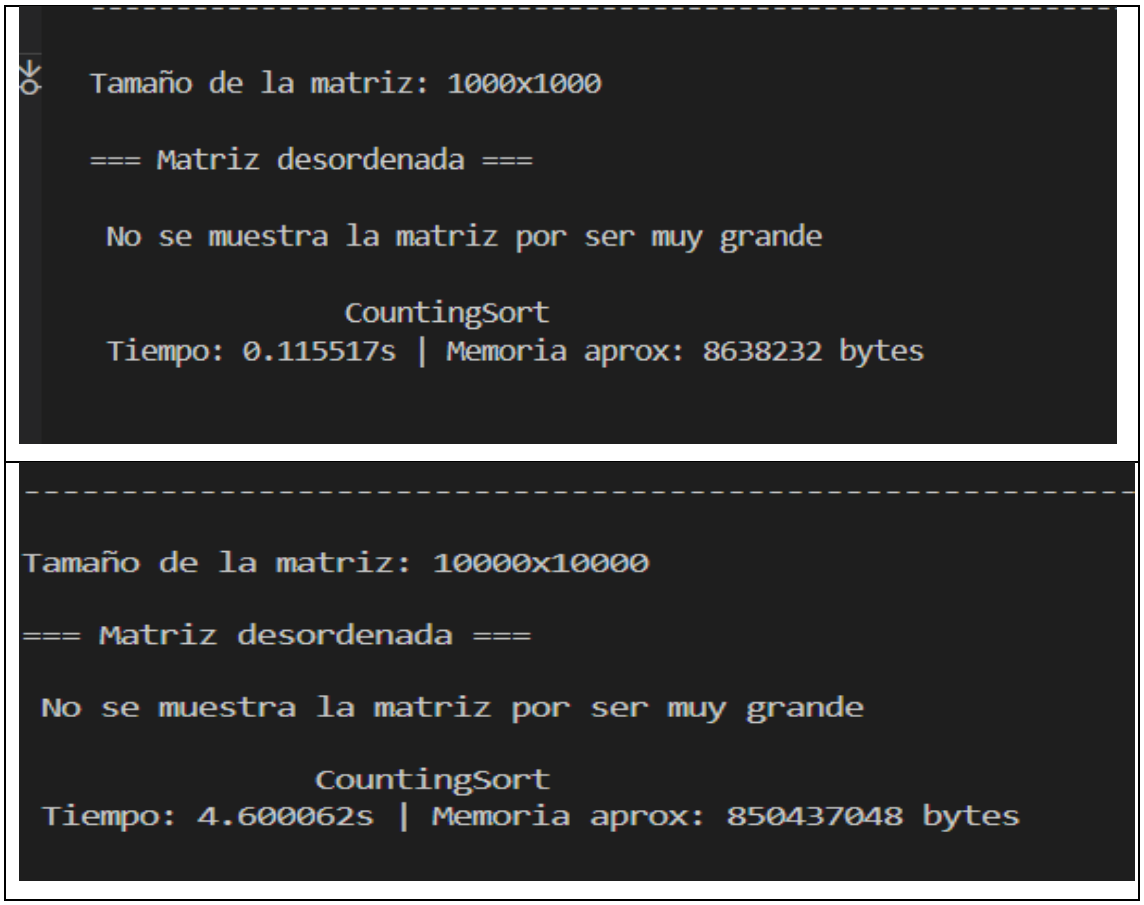
T(n) = 3n + 5

T(n) = O(n)

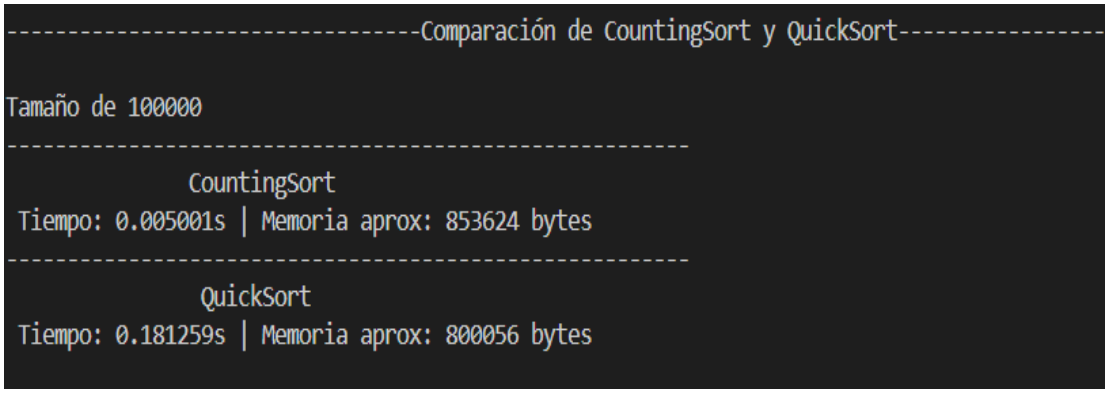
Resultados

```
Tamaño de la matriz: 10x10

=== Matriz desordenada ===
[332, 350, 725, 511, 515, 164, 212, 694, 487, 544]
[357, 784, 661, 986, 973, 469, 43, 361, 32, 896]
[1, 319, 161, 237, 258, 215, 672, 576, 630, 718]
[710, 178, 590, 339, 426, 351, 569, 666, 523, 275]
[36, 218, 351, 952, 132, 258, 204, 86, 416, 802]
[260, 232, 392, 868, 995, 391, 24, 765, 910, 440]
[58, 627, 727, 721, 891, 20, 68, 798, 824, 563]
[374, 481, 196, 474, 162, 12, 711, 560, 882, 697]
[533, 42, 574, 396, 163, 812, 222, 148, 729, 265]
[976, 719, 606, 659, 674, 372, 931, 274, 482, 218]

Matriz ordenada con CountingSort:

[164, 212, 332, 350, 487, 511, 515, 544, 694, 725]
[32, 43, 357, 361, 469, 661, 784, 896, 973, 986]
[1, 161, 215, 237, 258, 319, 576, 630, 672, 718]
[178, 275, 339, 351, 426, 523, 569, 590, 666, 710]
[36, 86, 132, 204, 218, 258, 351, 416, 802, 952]
[24, 232, 260, 391, 392, 440, 765, 868, 910, 995]
[20, 58, 68, 563, 627, 721, 727, 798, 824, 891]
[12, 162, 196, 374, 474, 481, 560, 697, 711, 882]
[42, 148, 163, 222, 265, 396, 533, 574, 729, 812]
[218, 274, 372, 482, 606, 659, 674, 719, 931, 976]

               CountingSort
 Tiempo: 0.001001s | Memoria aprox: 2024 bytes
```

```
 ------------------------------------------------------------------------

 Tamaño de la matriz: 100x100

 === Matriz desordenada ===

  No se muestra la matriz por ser muy grande

               CountingSort
  Tiempo: 0.006987s | Memoria aprox: 93272 bytes

 ------------------------------------------------------------------------
```



Comparación con otro algoritmo

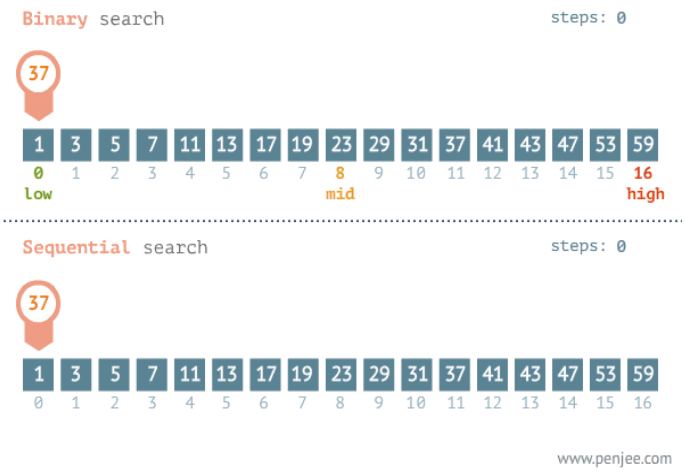


Parte 2: Implementación y análisis de búsqueda

Descripción del algoritmo:

El algoritmo de búsqueda implementado es Búsqueda Lineal. Este algoritmo comprueba secuencialmente cada elemento de un arreglo, comparando su valor con del elemento que se desea encontrar.

Se estará comparando con Búsqueda Binaria.



Análisis de complejidad:

Complejidad Temporal:

- **Peor caso:** El peor caso de la búsqueda lineal es de O(n).
- **Mejor caso:** La búsqueda lineal posee un mejor caso Ω(1).
- **Caso medio:** El caso medio de la búsqueda lineal es de θ(n).

Complejidad Espacial:

- O(1), ya que el algoritmo utiliza una cantidad fija de memoria independientemente del tamaño de la entrada.

Función T(n):

En este caso, la búsqueda se realizó con bucle for anidado para facilitar el recorrido dentro de la matriz

```
def busqueda_lineal(matriz, numero):
    for i in range(len(matriz)):          → n operaciones
        for j in range(len(matriz[i])):   → n operaciones
            if matriz[i][j] == numero:    → 1 operación
                return (i, j)             → 1 operación
    return -1                             → 1 operación
```

$$\mathbf{T(n) =} (n * n) + 1 + 1 + 1$$

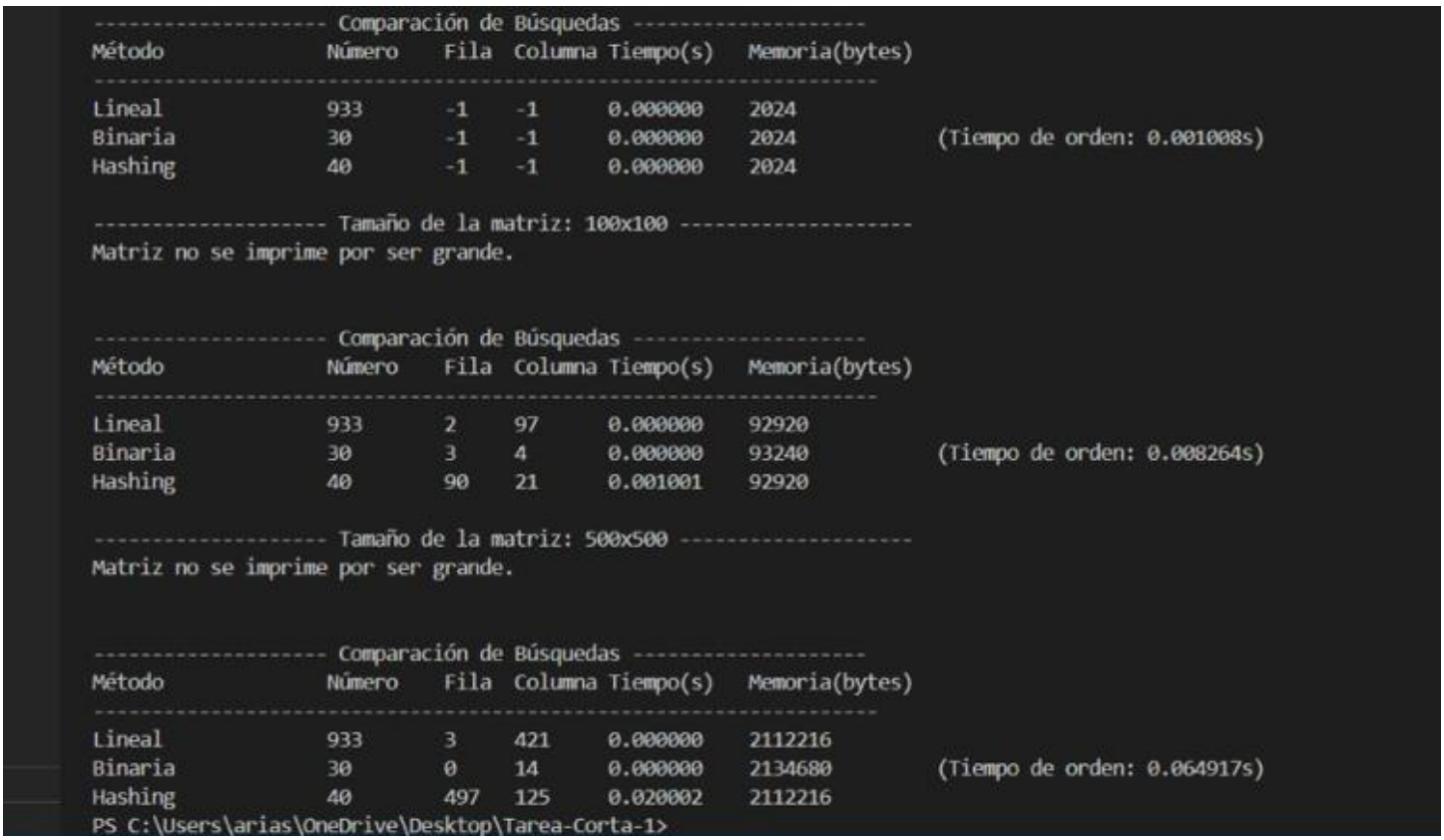
$$\mathbf{T(n) =} n^2+3$$

$$\mathbf{T(n) =} O(n^2)$$

Resultados:

```
---------------------------------------- ALGORITMO DE BÚSQUEDA ----------------------------------------


------------------- Tamaño de la matriz: 10x10 -------------------
Matriz generada:
[157, 4, 457, 572, 976, 773, 632, 546, 997, 847]
[982, 934, 791, 887, 724, 941, 862, 879, 994, 954]
[69, 626, 907, 808, 169, 351, 383, 862, 670, 127]
[472, 474, 796, 472, 91, 548, 618, 888, 945, 144]
[539, 947, 19, 354, 46, 967, 618, 722, 445, 473]
[55, 485, 477, 340, 231, 429, 316, 195, 423, 334]
[988, 857, 690, 187, 344, 152, 274, 153, 693, 76]
[90, 668, 942, 570, 436, 44, 891, 318, 665, 294]
[488, 934, 975, 988, 867, 323, 352, 382, 455, 575]
[303, 814, 84, 315, 354, 510, 825, 812, 37, 421]

---------------------------------------------------- Comparación de Búsquedas ----------------------------------
Algoritmo          Número    Fila  Columna Tiempo(segundos)Memoria(bytes)
------------------------------------------------------------------------------------------------------------------
Lineal             768       -1    -1      0.509262    2024
Binaria            1         -1    -1      0.000000    2024           (Tiempo ordenando: 0.511885segundos)

------------------- Tamaño de la matriz: 100x100 -------------------
Matriz no se imprime por ser grande.


---------------------------------------------------- Comparación de Búsquedas ----------------------------------
Algoritmo          Número    Fila  Columna Tiempo(segundos)Memoria(bytes)
------------------------------------------------------------------------------------------------------------------
Lineal             768       6     36      0.000000    92920
Binaria            1         25    0       0.000000    93368          (Tiempo ordenando: 17.990112segundos)

------------------- Tamaño de la matriz: 500x500 -------------------
Matriz no se imprime por ser grande.


---------------------------------------------------- Comparación de Búsquedas ----------------------------------
Algoritmo          Número    Fila  Columna Tiempo(segundos)Memoria(bytes)
------------------------------------------------------------------------------------------------------------------
Lineal             768       3     190     0.000000    2112216
Binaria            1         2     0       0.000000    2135544        (Tiempo ordenando: 273.881912segundos)
PS C:\Users\arias\OneDrive\Desktop\Tarea-Corta-1>
```

Comparación con otro algoritmo:



```
-------------------- Comparación de Búsquedas --------------------
Método              Número    Fila  Columna Tiempo(s)   Memoria(bytes)
------------------------------------------------------------------
Lineal              933       -1    -1      0.000000    2024
Binaria             30        -1    -1      0.000000    2024          (Tiempo de orden: 0.001008s)
Hashing             40        -1    -1      0.000000    2024

-------------------- Tamaño de la matriz: 100x100 --------------------
Matriz no se imprime por ser grande.


-------------------- Comparación de Búsquedas --------------------
Método              Número    Fila  Columna Tiempo(s)   Memoria(bytes)
------------------------------------------------------------------
Lineal              933       2     97      0.000000    92920
Binaria             30        3     4       0.000000    93240         (Tiempo de orden: 0.008264s)
Hashing             40        90    21      0.001001    92920

-------------------- Tamaño de la matriz: 500x500 --------------------
Matriz no se imprime por ser grande.


-------------------- Comparación de Búsquedas --------------------
Método              Número    Fila  Columna Tiempo(s)   Memoria(bytes)
------------------------------------------------------------------
Lineal              933       3     421     0.000000    2112216
Binaria             30        0     14      0.000000    2134680       (Tiempo de orden: 0.064917s)
Hashing             40        497   125     0.020002    2112216
PS C:\Users\arias\OneDrive\Desktop\Tarea-Corta-1>
```

Enlace del repositorio de GitHub: [Tarea Corta 1]