Desarrollo Tarea 2

Estudiante: Omar David Toledo Leguizamón

Parte 1

El primer paso a realizar, será generar los archivos con las diferentes cantidades de numeros aleatorios usadas para comparar los algoritmos

In [16]:
```
import random

def generate_random_integers_file(filename, num_integers, min_value=0, max_value=10
    with open(filename, 'w') as file:
        for _ in range(num_integers):
            random_integer = random.randint(min_value, max_value)
            file.write(f"{random_integer}\n")
    print(f'{num_integers} numbers file generaterd')
```

In [17]:
```
generate_random_integers_file('TallerRecursion/data/numbers100.txt', 100)
generate_random_integers_file('TallerRecursion/data/numbers1k.txt', 1000)
generate_random_integers_file('TallerRecursion/data/numbers10k.txt', 10000)
generate_random_integers_file('TallerRecursion/data/numbers100k.txt', 100000)
generate_random_integers_file('TallerRecursion/data/numbers1M.txt', 1000000)
```

```
100 numbers file generaterd
1000 numbers file generaterd
10000 numbers file generaterd
100000 numbers file generaterd
1000000 numbers file generaterd
```

A partir de estos, se realizará la carga de datos para los algoritmos de QuickSort, MergeSort y BubbleSort para todos los conjuntos de datos y se compararán las seis salidas

1. QuickSort

Resultado con 100 datos:

```
ArrayList size: 100 recursive list size: 100
First location of number 2370. In ArrayList: 11 in recursive list:
11
Last location of number 2370. In ArrayList: 11 in recursive list:
11
Element in position 11. ArrayList: 2370 recursive list 2370
The list is not sorted
Number of even elements. Divide and conquer: 50. recursive list 50
Maximum. Divide and conquer: 9922. recursive list 9922
Position of value 2370. Divide and conquer: 11. Value in numbers
2370
```

```
Sorting algorithm: Quick
Numbers sorted. Total time(milliseconds): 1
Position of value 2370. Divide and conquer: 26. Value in numbers
2370
```

Resultado con 1000 datos:

```
ArrayList size: 1000 recursive list size: 1000
First location of number 492. In ArrayList: 779 in recursive list:
779
Last location of number 492. In ArrayList: 779 in recursive list:
779
Element in position 779. ArrayList: 492 recursive list 492
The list is not sorted
Number of even elements. Divide and conquer: 496. recursive list
496
Maximum. Divide and conquer: 9994. recursive list 9994
Position of value 492. Divide and conquer: 779. Value in numbers
492
Sorting algorithm: Quick
Numbers sorted. Total time(milliseconds): 1
Position of value 492. Divide and conquer: 52. Value in numbers 492
```

Resultado con 10000 datos:

```
ArrayList size: 10000 recursive list size: 10000
First location of number 1794. In ArrayList: 9721 in recursive
list: 9721
Last location of number 1794. In ArrayList: 9721 in recursive list:
9721
Element in position 9721. ArrayList: 1794 recursive list 1794
The list is not sorted
Number of even elements. Divide and conquer: 4985. recursive list
4985
Maximum. Divide and conquer: 9998. recursive list 9998
Position of value 1794. Divide and conquer: 9721. Value in numbers
1794
Sorting algorithm: Quick
Numbers sorted. Total time(milliseconds): 9
Position of value 1794. Divide and conquer: 1784. Value in numbers
1794
```

Resultado con 100000 datos:

```
ArrayList size: 100000 recursive list size: 100000
First location of number 312. In ArrayList: 3360 in recursive list:
3360
Last location of number 312. In ArrayList: 74670 in recursive list:
74670
```

```
Element in position 74670. ArrayList: 312 recursive list 312
The list is not sorted
Number of even elements. Divide and conquer: 50175. recursive list
50175
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 312. Divide and conquer: 3360. Value in numbers
312
Sorting algorithm: Quick
Numbers sorted. Total time(milliseconds): 49
Position of value 312. Divide and conquer: 3050. Value in numbers
312
```

Resultado con 100000 datos:

```
ArrayList size: 1000000 recursive list size: 1000000
First location of number 4832. In ArrayList: 7641 in recursive
list: 7641
Last location of number 4832. In ArrayList: 998803 in recursive
list: 998803
Element in position 146651. ArrayList: 4832 recursive list 4832
The list is not sorted
Number of even elements. Divide and conquer: 500119. recursive list
500119
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 4832. Divide and conquer: 7641. Value in numbers
4832
Sorting algorithm: Quick
Numbers sorted. Total time(milliseconds): 1125
Position of value 4832. Divide and conquer: 482969. Value in
numbers 4832
```

2. BubbleSort

Resultado con 100 datos:

```
ArrayList size: 100 recursive list size: 100
First location of number 9016. In ArrayList: 15 in recursive list:
15
Last location of number 9016. In ArrayList: 15 in recursive list:
15
Element in position 15. ArrayList: 9016 recursive list 9016
The list is not sorted
Number of even elements. Divide and conquer: 50. recursive list 50
Maximum. Divide and conquer: 9922. recursive list 9922
Position of value 9016. Divide and conquer: 15. Value in numbers
9016
Sorting algorithm: Bubble
Numbers sorted. Total time(milliseconds): 2
Position of value 9016. Divide and conquer: 86. Value in numbers
9016
```

Resultado con 1000 datos:

```
ArrayList size: 1000 recursive list size: 1000
First location of number 6462. In ArrayList: 638 in recursive list:
638
Last location of number 6462. In ArrayList: 638 in recursive list:
638
Element in position 638. ArrayList: 6462 recursive list 6462
The list is not sorted
Number of even elements. Divide and conquer: 496. recursive list
496
Maximum. Divide and conquer: 9994. recursive list 9994
Position of value 6462. Divide and conquer: 638. Value in numbers
6462
Sorting algorithm: Bubble
Numbers sorted. Total time(milliseconds): 28
Position of value 6462. Divide and conquer: 638. Value in numbers
6462
```

Resultado con 10000 datos:

```
ArrayList size: 10000 recursive list size: 10000
First location of number 8711. In ArrayList: 6659 in recursive
list: 6659
Last location of number 8711. In ArrayList: 8428 in recursive list:
8428
Element in position 8428. ArrayList: 8711 recursive list 8711
The list is not sorted
Number of even elements. Divide and conquer: 4985. recursive list
4985
Maximum. Divide and conquer: 9998. recursive list 9998
Position of value 8711. Divide and conquer: 6659. Value in numbers
8711
Sorting algorithm: Bubble
Numbers sorted. Total time(milliseconds): 433
Position of value 8711. Divide and conquer: 8727. Value in numbers
8711
```

Resultado con 100000 datos:

```
ArrayList size: 100000 recursive list size: 100000
First location of number 7701. In ArrayList: 10054 in recursive
list: 10054
Last location of number 7701. In ArrayList: 99820 in recursive
list: 99820
Element in position 85224. ArrayList: 7701 recursive list 7701
The list is not sorted
Number of even elements. Divide and conquer: 50175. recursive list
50175
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 7701. Divide and conquer: 10054. Value in numbers
```

```
7701
Sorting algorithm: Bubble
Numbers sorted. Total time(milliseconds): 78663
Position of value 7701. Divide and conquer: 76840. Value in numbers
7701
```

Resultado con 100000 datos:

```
ArrayList size: 1000000 recursive list size: 1000000
First location of number 6724. In ArrayList: 19323 in recursive
list: 19323
Last location of number 6724. In ArrayList: 999200 in recursive
list: 999200
Element in position 747411. ArrayList: 6724 recursive list 6724
The list is not sorted
Number of even elements. Divide and conquer: 500119. recursive list
500119
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 6724. Divide and conquer: 19323. Value in numbers
6724
Sorting algorithm: Bubble
Numbers sorted. Total time(milliseconds): 6708568
Position of value 6724. Divide and conquer: 671507. Value in
numbers 6724
```

3. MergeSort

Resultado con 100 datos:

```
ArrayList size: 100 recursive list size: 100
First location of number 159. In ArrayList: 40 in recursive list:
40
Last location of number 159. In ArrayList: 40 in recursive list: 40
Element in position 40. ArrayList: 159 recursive list 159
The list is not sorted
Number of even elements. Divide and conquer: 50. recursive list 50
Maximum. Divide and conquer: 9922. recursive list 9922
Position of value 159. Divide and conquer: 40. Value in numbers 159
Sorting algorithm: Merge
Numbers sorted. Total time(milliseconds): 1
Position of value 159. Divide and conquer: 2. Value in numbers 159
```

Resultado con 1000 datos:

```
ArrayList size: 1000 recursive list size: 1000
First location of number 4414. In ArrayList: 918 in recursive list:
918
Last location of number 4414. In ArrayList: 918 in recursive list:
```

```
918
Element in position 918. ArrayList: 4414 recursive list 4414
The list is not sorted
Number of even elements. Divide and conquer: 496. recursive list
496
Maximum. Divide and conquer: 9994. recursive list 9994
Position of value 4414. Divide and conquer: 918. Value in numbers
4414
Sorting algorithm: Merge
Numbers sorted. Total time(milliseconds): 2
Position of value 4414. Divide and conquer: 441. Value in numbers
4414
```

Resultado con 10000 datos:

```
ArrayList size: 10000 recursive list size: 10000
First location of number 8938. In ArrayList: 2764 in recursive
list: 2764
Last location of number 8938. In ArrayList: 3867 in recursive list:
3867
Element in position 3867. ArrayList: 8938 recursive list 8938
The list is not sorted
Number of even elements. Divide and conquer: 4985. recursive list
4985
Maximum. Divide and conquer: 9998. recursive list 9998
Position of value 8938. Divide and conquer: 2764. Value in numbers
8938
Sorting algorithm: Merge
Numbers sorted. Total time(milliseconds): 13
Position of value 8938. Divide and conquer: 8973. Value in numbers
8938
```

Resultado con 100000 datos:

```
ArrayList size: 100000 recursive list size: 100000
First location of number 7885. In ArrayList: 7555 in recursive
list: 7555
Last location of number 7885. In ArrayList: 96377 in recursive
list: 96377
Element in position 27666. ArrayList: 7885 recursive list 7885
The list is not sorted
Number of even elements. Divide and conquer: 50175. recursive list
50175
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 7885. Divide and conquer: 7555. Value in numbers
7885
Sorting algorithm: Merge
Numbers sorted. Total time(milliseconds): 151
Position of value 7885. Divide and conquer: 78696. Value in numbers
7885
```

Resultado con 100000 datos:

```
ArrayList size: 1000000 recursive list size: 1000000
First location of number 4012. In ArrayList: 1086 in recursive
list: 1086
Last location of number 4012. In ArrayList: 994811 in recursive
list: 994811
Element in position 754714. ArrayList: 4012 recursive list 4012
The list is not sorted
Number of even elements. Divide and conquer: 500119. recursive list
500119
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 4012. Divide and conquer: 1086. Value in numbers
4012
Sorting algorithm: Merge
Numbers sorted. Total time(milliseconds): 966
Position of value 4012. Divide and conquer: 400755. Value in
numbers 4012
```

Resumen

Finalmente, diseñamos una tabla que resuma los tiempos de ejecución para cada tipo de datos en milisegundos

Algoritmo	100	1k	10k	100k	1M
BubbleSort	2	28	433	78663	6708568
QuickSort	1	1	9	49	1125
MergeSort	1	2	13	151	966

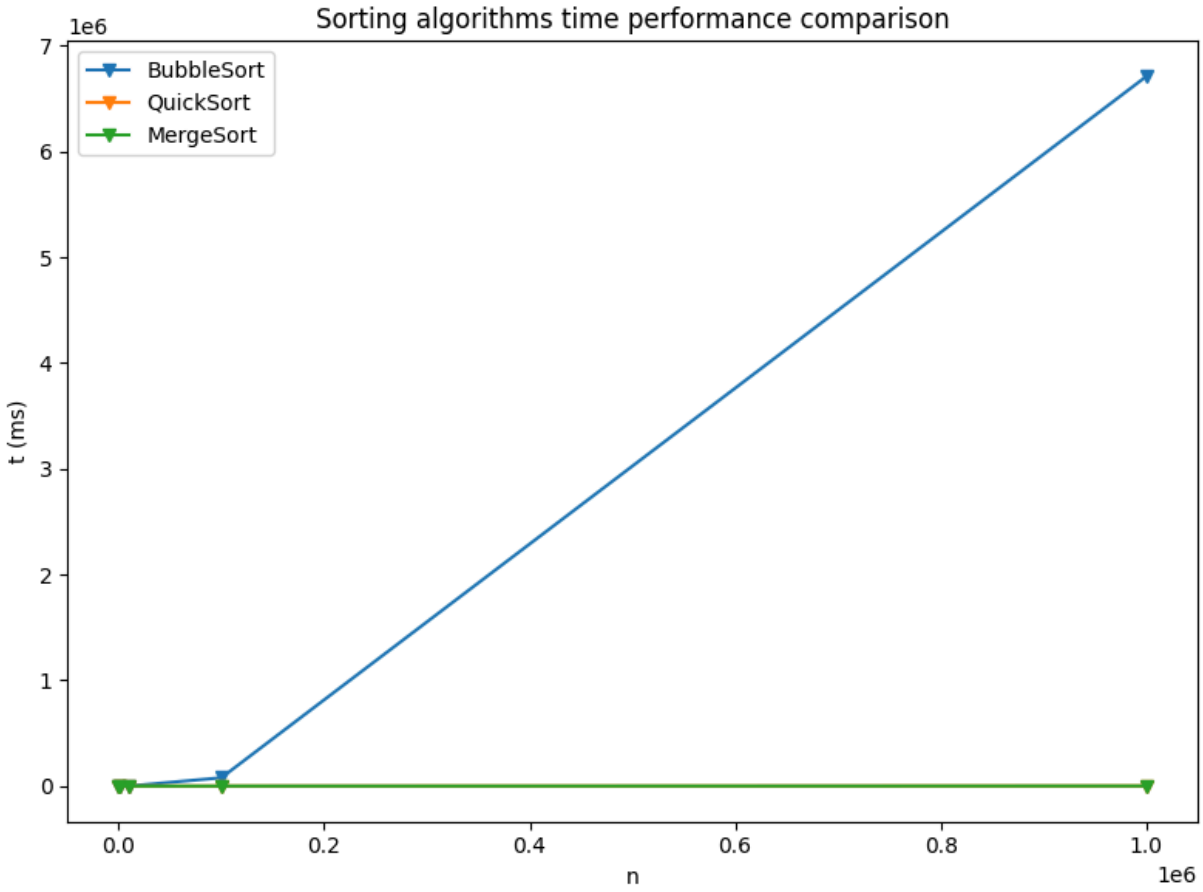
Adicionalmente, diseñamos una rutina que nos permita visualizar los resultados de manera gráfica de este conbjunto de datos obtenido.

In [18]:
```python
import matplotlib.pyplot as plt

def plot_execution_values(dictionary_info):
    methods = list(dictionary_info.keys())
    values = list(dictionary_info[methods[0]].keys())
    fig = plt.figure(figsize=(8,6))
    for method in methods:
        data = [dictionary_info[method][value] for value in values]
        plt.plot(values,data,label = method, marker = 'v')
    plt.xlabel('n')
    plt.ylabel('t (ms)')
    plt.title('Sorting algorithms time performance comparison')
    plt.legend()
    plt.tight_layout()
    plt.show()
```

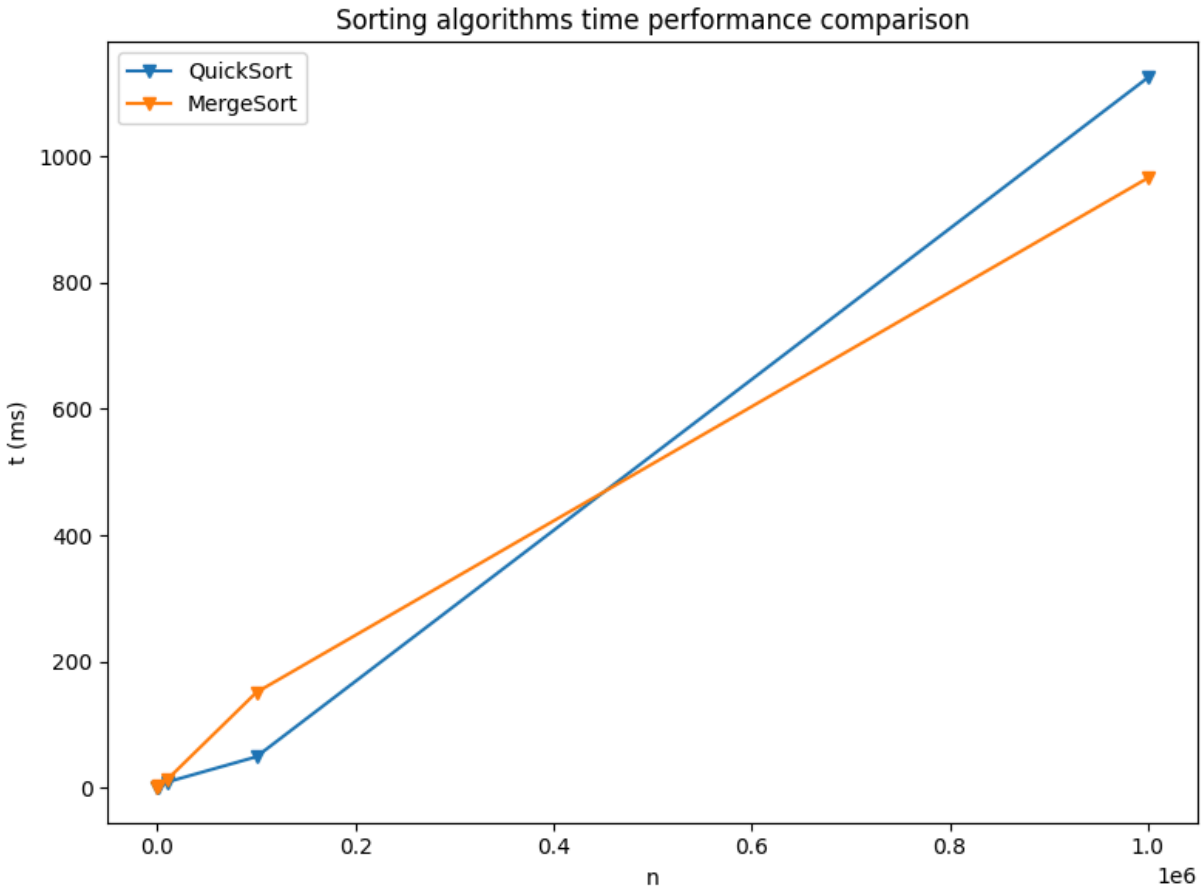
In [19]:
```
dict_ = {
    'BubbleSort' : {100:2 , 1000:28, 10000: 433, 100000: 78663, 1000000: 6708568},
    'QuickSort' : {100:1 , 1000:1, 10000: 9, 100000: 49, 1000000: 1125},
    'MergeSort' : {100:1 , 1000:2, 10000: 13, 100000: 151, 1000000: 966}
}

plot_execution_values(dict_)
```



In [20]:
```
dict_ = {
    'QuickSort' : {100:1 , 1000:1, 10000: 9, 100000: 49, 1000000: 1125},
    'MergeSort' : {100:1 , 1000:2, 10000: 13, 100000: 151, 1000000: 966}
}

plot_execution_values(dict_)
```



Parte 2

El desarrollo de esta segunda parte se encuentra en el proyecto de Java adjunto a este documento. Dado esto, no se hará enfasis en el proceso en este documento y para su revisión se recomienda revisar las clases DivideAndConquerIntegerArrays y RecursiveIntegerLinkedList.

Parte 3

Encontrar una fórmula cerrada para las siguientes ecuaciones de recurrencia:

$$f(n) = 4f(n-2) + 3^n \qquad f(0) = 1 \quad f(1) = 2$$

Podemos denotar que la ecuación descrita es una recurrencia lineal no homogenea. Dado esto, el proceso a seguir será hallar la solución homogenea y la solución particular de la ecuación, para luego relacionar las condiciones iniciales

Solución Homogénea

Vamos a solucionar la recurrencia homogénea:

$$h(n) - 4h(n-2) = 0$$

Para ello, vamos a asociar su polinomio característico y sus raíces:

$$\lambda^2 - 4 = 0 \quad \Longrightarrow \quad \lambda = 2 \ \wedge \ \lambda = -2$$

Dado esto, la solución homogénea de la recurrencia está dada por:

$$h(n) = c_1(2)^n + c_2(-2)^n$$

Solución Particular

Para la solución de la parte no homogénea, vamos a suponer una solución $p(n)$ que cumple lo siguiente:

$$p(n) - 4p(n-2) = 3^n$$

Utiizando la heurística, supondremos que la solución es de la forma $p(n) = c_3 3^n$. Dado esto, tratamos de hallar el valor de c_3:

$$\begin{aligned}
p(n) - 4p(n-2) &= 3^n \\
c_3 3^n - 4c_3 3^{n-2} &= 3^n \\
c_3 3^n - \frac{4}{9} c_3 3^n &= 3^n \\
\frac{5}{9} c_3 3^n &= 3^n \\
\frac{5}{9} c_3 3^n - 3^n &= 0 \\
3^n \left(\frac{5}{9} c_3 - 1 \right) &= 0 \\
&\Downarrow \\
\frac{5}{9} c_3 - 1 &= 0 \\
c_3 &= \frac{9}{5}
\end{aligned}$$

Dado esto, nuestra solución particular esta dada por:

$$p(n) = \frac{9}{5} 3^n = \frac{3^{n+2}}{5}$$

Aplicación de Condiciones iniciales

La solución de nuestra ecuación esta descrita de la forma:

$$f(n) = h(n) + p(n) = c_1(2)^n + c_2(-2)^n + \frac{3^{n+2}}{5}$$

El siguiente proceso sera usar las condiciones $f(0) = 1$ y $f(1) = 2$ para determinar los valores de c_1 y c_2

$$f(0) = 1 \quad = \quad c_1(2)^0 + c_2(-2)^0 + \frac{3^{0+2}}{5} \quad = \quad c_1 + c_2 + \frac{9}{5} \quad \Longrightarrow \quad c_1 + c_2 = -\frac{4}{5}$$

$$f(1) = 2 \quad = \quad c_1(2)^1 + c_2(-2)^1 + \frac{3^{1+2}}{5} \quad = \quad 2c_1 - 2c_2 + \frac{27}{5} \quad \Longrightarrow \quad 2c_1 - 2c_2 = -\frac{17}{5}$$

Resolvemos el sistema de ecuaciones lineales 2×2

$$\begin{pmatrix} 1 & 1 & -\frac{4}{5} \\ 2 & -2 & -\frac{17}{5} \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 1 & -\frac{4}{5} \\ 0 & -4 & -\frac{9}{5} \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 1 & -\frac{4}{5} \\ 0 & 1 & \frac{9}{20} \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 0 & -\frac{5}{4} \\ 0 & 1 & \frac{9}{20} \end{pmatrix}$$

En ese orden de ideas, la solución cerrada de la ecuación de recurrencia es:

$$f(n) = -\frac{5}{4}(2)^n + \frac{9}{20}(-2)^n + \frac{3^{n+2}}{5}$$

$$f(n) = 4f(n-2) + 2^n \qquad f(0) = 1 \quad f(1) = 3$$

Podemos denotar que la ecuación descrita es una recurrencia lineal no homogenea. Dado esto, el proceso a seguir será hallar la solución homogenea y la solución particular de la ecuación, para luego relacionar las condiciones iniciales

Solución Homogénea

Vamos a solucionar la recurrencia homogénea:

$$h(n) - 4h(n-2) = 0$$

Para ello, vamos a asociar su polinomio característico y sus raíces:

$$\lambda^2 - 4 = 0 \quad \Longrightarrow \quad \lambda = 2 \ \wedge \ \lambda = -2$$

Dado esto, la solución homogénea de la recurrencia está dada por:

$$h(n) = c_1(2)^n + c_2(-2)^n$$

Solución Particular

Para la solución de la parte no homogénea, vamos a suponer una solución $p(n)$ que cumple lo siguiente:

$$p(n) - 4p(n-2) = 2^n$$

Utiizando la heurística, supondremos que la solución es de la forma $p(n) = c_3\, n\, 2^n$. Dado esto, tratamos de hallar el valor de c_3:

$$\begin{aligned}
p(n) - 4p(n-2) &= 2^n \\
c_3\, n\, 2^n - 4c_3\,(n-2)\, 2^{n-2} &= 2^n \\
c_3\, n\, 2^n - c_3\,(n-2)\, 2^n &= 2^n \\
2c_3 2^n &= 2^n \\
2c_3 2^n - 2^n &= 0 \\
2^n\Big(2c_3 - 1\Big) &= 0 \\
&\Downarrow \\
2c_3 - 1 &= 0 \\
c_3 &= \frac{1}{2}
\end{aligned}$$

Dado esto, nuestra solución particular esta dada por:

$$p(n) = \frac{1}{2}\, n\, 2^n = n\, 2^{n-1}$$

Aplicación de Condiciones iniciales

La solución de nuestra ecuación esta descrita de la forma:

$$f(n) = h(n) + p(n) = c_1(2)^n + c_2(-2)^n + n\, 2^{n-1}$$

El siguiente proceso sera usar las condiciones $f(0) = 1$ y $f(1) = 3$ para determinar los valores de c_1 y c_2

$$f(0) = 1 \quad = \quad c_1(2)^0 + c_2(-2)^0 + (0)2^{0-1} \quad = \quad c_1 + c_2 \quad \Longrightarrow \quad c_1 + c_2 = 1$$

$$f(1) = 3 \quad = \quad c_1(2)^1 + c_2(-2)^1 + (1)\, 2^{1-1} \quad = \quad 2c_1 - 2c_2 + 1 \quad \Longrightarrow \quad 2c_1 - 2c_2 = 2$$

Resolvemos el sistema de ecuaciones lineales 2×2

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & -2 & 2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & -4 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

En ese orden de ideas, la solución cerrada de la ecuación de recurrencia es:

$$f(n) = 2^n + n\,2^{n-1}$$

$$f(n) = 5f(n-1) - 6f(n-2) + n \qquad f(0) = 0 \quad f(1) = 2$$

Podemos denotar que la ecuación descrita es una recurrencia lineal no homogenea. Dado esto, el proceso a seguir será hallar la solución homogenea y la solución particular de la ecuación, para luego relacionar las condiciones iniciales

Solución Homogénea

Vamos a solucionar la recurrencia homogénea:

$$h(n) - 5h(n-1) + 6h(n-2) = 0$$

Para ello, vamos a asociar su polinomio característico y sus raíces:

$$\lambda^2 - 5\lambda + 6 = 0 \quad \Longrightarrow \quad (\lambda - 2)(\lambda - 3) = 0 \quad \Longrightarrow \quad \lambda = 2 \ \wedge \ \lambda = 3$$

Dado esto, la solución homogénea de la recurrencia está dada por:

$$h(n) = c_1(2)^n + c_2(3)^n$$

Solución Particular

Para la solución de la parte no homogénea, vamos a suponer una solución $p(n)$ que cumple lo siguiente:

$$p(n) - 5p(n-1) + 6p(n-2) = n$$

Utiizando la heurística, supondremos que la solución es de la forma $p(n) = c_3\,n + c_4$. Dado esto, tratamos de hallar el valor de c_3:

$$\begin{aligned}
p(n) - 5p(n-1) + 6p(n-2) &= n \\
c_3\,n + c_4 - 5(c_3\,(n-1) + c_4) + 6(c_3\,(n-2) + c_4) &= n \\
c_3\,n + c_4 - 5c_3\,n + 5c_3 - 5c_4 + 6c_3\,n - 12c_3 + 6c_4 &= n \\
n(c_3 - 5c_3 + 6c_3) + (c_4 + 5c_3 - 5c_4 - 12c_3 + 6c_4) &= n \\
2c_3 n + (2c_4 - 7c_3) - n &= 0 \\
n(2c_3 - 1) + (2c_4 - 7c_3) &= 0 \\
&\Downarrow \\
2c_3 - 1 = 0 \quad &\wedge \quad 2c_4 - 7c_3 = 0 \\
c_3 = \frac{1}{2} \quad &\wedge \quad c_4 = \frac{7}{4}
\end{aligned}$$

Dado esto, nuestra solución particular esta dada por:

$$p(n) = \frac{1}{2}n + \frac{7}{4}$$

Aplicación de Condiciones iniciales

La solución de nuestra ecuación esta descrita de la forma:

$$f(n) = h(n) + p(n) = c_1(2)^n + c_2(3)^n + \frac{1}{2}n + \frac{7}{4}$$

El siguiente proceso sera usar las condiciones $f(0) = 0$ y $f(1) = 2$ para determinar los valores de c_1 y c_2

$$f(0) = 0 \quad = \quad c_1(2)^0 + c_2(3)^0 + \frac{1}{2}(0) + \frac{7}{4} \quad = \quad c_1 + c_2 + \frac{7}{4} \quad \Longrightarrow \quad c_1 + c_2 = -\frac{7}{4}$$

$$f(1) = 2 \quad = \quad c_1(2)^1 + c_2(3)^1 + \frac{1}{2}(1) + \frac{7}{4} \quad = \quad 2c_1 + 3c_2 + \frac{9}{4} \quad \Longrightarrow \quad 2c_1 + 3c_2 = -\frac{1}{4}$$

Resolvemos el sistema de ecuaciones lineales 2×2

$$\begin{pmatrix} 1 & 1 & -\frac{7}{4} \\ 2 & 3 & -\frac{1}{4} \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 1 & -\frac{7}{4} \\ 0 & 1 & \frac{13}{4} \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 0 & -5 \\ 0 & 1 & \frac{13}{4} \end{pmatrix}$$

En ese orden de ideas, la solución cerrada de la ecuación de recurrencia es:

$$f(n) = -5(2)^n + \frac{13}{4}(3)^n + \frac{1}{2}n + \frac{7}{4}$$

$$f(n) = 4f(n-1) - 4f(n-2) + 5 \qquad f(0) = 0 \quad f(1) = 1$$

Podemos denotar que la ecuación descrita es una recurrencia lineal no homogenea. Dado esto, el proceso a seguir será hallar la solución homogenea y la solución particular de la ecuación, para luego relacionar las condiciones iniciales

Solución Homogénea

Vamos a solucionar la recurrencia homogénea:

$$h(n) - 4h(n-1) + 4h(n-2) = 0$$

Para ello, vamos a asociar su polinomio característico y sus raíces:

$$\lambda^2 - 4\lambda + 4 = 0 \quad \Longrightarrow \quad (\lambda - 2)^2 = 0 \quad \Longrightarrow \quad \lambda = 2$$

Dado esto, la solución homogénea de la recurrencia está dada por:

$$h(n) = c_1(2)^n + c_2\, n\,(2)^n$$

Solución Particular

Para la solución de la parte no homogénea, vamos a suponer una solución $p(n)$ que cumple lo siguiente:

$$p(n) - 4p(n-1) + 4p(n-2) = 5$$

Utiizando la heurística, supondremos que la solución es de la forma $p(n) = c_3$. Dado esto, tratamos de hallar el valor de c_3:

$$\begin{aligned} p(n) - 4p(n-1) + 4p(n-2) &= 5 \\ c_3 - 4c_3 + 4c_3 &= 5 \\ c_3 &= 5 \end{aligned}$$

Dado esto, nuestra solución particular esta dada por:

$$p(n) = 5$$

Aplicación de Condiciones iniciales

La solución de nuestra ecuación esta descrita de la forma:

$$f(n) = h(n) + p(n) = c_1(2)^n + c_2\, n\,(2)^n + 5$$

El siguiente proceso sera usar las condiciones $f(0) = 0$ y $f(1) = 1$ para determinar los valores de c_1 y c_2

$$f(0) = 0 \quad = \quad c_1(2)^0 + c_2\,(0)\,(2)^0 + 5 \quad = \quad c_1 + 5 \qquad \Longrightarrow \quad c_1 = -5$$

$$f(1) = 1 \quad = \quad c_1(2)^1 + c_2\,(1)\,(2)^1 + 5 \quad = \quad 2c_1 + 2c_2 + 5 \quad \Longrightarrow \quad 2c_1 + 2c_2 = -4$$

Resolvemos el sistema de ecuaciones lineales 2×2

$$\begin{pmatrix} 1 & 0 & -5 \\ 2 & 2 & -4 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 0 & -5 \\ 0 & 2 & 6 \end{pmatrix} \Longrightarrow \begin{pmatrix} 1 & 0 & -5 \\ 0 & 1 & 3 \end{pmatrix}$$

En ese orden de ideas, la solución cerrada de la ecuación de recurrencia es:

$$f(n) = -5(2)^n + 3\, n\,(2)^n + 5$$

$$f(n) = 4f(n-1) - 4f(n-2) + 2^n \qquad f(0) = 1 \quad f(1) = 4$$

Podemos denotar que la ecuación descrita es una recurrencia lineal no homogenea. Dado esto, el proceso a seguir será hallar la solución homogenea y la solución particular de la ecuación, para luego relacionar las condiciones iniciales

Solución Homogénea

Vamos a solucionar la recurrencia homogénea:

$$h(n) - 4h(n-1) + 4h(n-2) = 0$$

Para ello, vamos a asociar su polinomio característico y sus raíces:

$$\lambda^2 - 4\lambda + 4 = 0 \quad \Longrightarrow \quad (\lambda - 2)^2 = 0 \quad \Longrightarrow \quad \lambda = 2$$

Dado esto, la solución homogénea de la recurrencia está dada por:

$$h(n) = c_1(2)^n + c_2\, n\, (2)^n$$

Solución Particular

Para la solución de la parte no homogénea, vamos a suponer una solución $p(n)$ que cumple lo siguiente:

$$p(n) - 4p(n-1) + 4p(n-2) = 2^n$$

Utiizando la heurística, supondremos que la solución es de la forma $p(n) = c_3\, n^2\, 2^n$. Dado esto, tratamos de hallar el valor de c_3:

$$\begin{aligned}
p(n) - 4p(n-1) + 4p(n-2) &= 2^n \\
c_3\, n^2\, 2^n - 4(c_3\,(n-1)^2\, 2^{n-1}) + 4(c_3\,(n-2)^2\, 2^{n-2}) &= 2^n \\
c_3\, n^2\, 2^n - 4(c_3\,(n^2 - 2n + 1)\, 2^{n-1}) + 4(c_3\,(n^2 - 4n + 4)\, 2^{n-2}) &= 2^n \\
c_3\, n^2\, 2^n - 2c_3\,(n^2 - 2n + 1)\, 2^n + c_3\, 2^n (n^2 - 4n + 4) &= 2^n \\
2^n\, n^2\, (c_3 - 2c_3 + c_3) + 2^n\, n\, (4c_3 - 4c_3) + 2^n (-2c_3 + 4c_3) &= 2^n \\
2c_3\, 2^n &= 2^n \\
2c_3\, 2^n - 2^n &= 0 \\
2^n\, (2c_3 - 1) &= 0 \\
&\Downarrow \\
2c_3 - 1 &= 0 \\
c_3 &= \frac{1}{2}
\end{aligned}$$

Dado esto, nuestra solución particular esta dada por:

$$p(n) = \frac{1}{2}\, n^2\, 2^n = n^2\, 2^{n-1}$$

Aplicación de Condiciones iniciales

La solución de nuestra ecuación esta descrita de la forma:

$$f(n) = h(n) + p(n) = c_1(2)^n + c_2\, n\,(2)^n + n^2\, 2^{n-1}$$

El siguiente proceso sera usar las condiciones $f(0) = 1$ y $f(1) = 4$ para determinar los valores de c_1 y c_2

$$\begin{aligned} f(0) = 1 &= c_1(2)^0 + c_2\,(0)\,(2)^0 + (0)^2\, 2^{0-1} &= c_1 &\implies c_1 = 1 \\ f(1) = 4 &= c_1(2)^1 + c_2\,(1)\,(2)^1 + (1)^2\, 2^{1-1} &= 2c_1 + 2c_2 + 1 &\implies 2c_1 + 2c_2 = 3 \end{aligned}$$

Resolvemos el sistema de ecuaciones lineales 2×2

$$\begin{pmatrix} 1 & 0 & 1 \\ 2 & 2 & 3 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 1 \\ 0 & 2 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & \frac{1}{2} \end{pmatrix}$$

En ese orden de ideas, la solución cerrada de la ecuación de recurrencia es:

$$f(n) = (2)^n + \frac{1}{2}\, n\,(2)^n + n^2\, 2^{n-1} = 2(2)^{n-1} + n\,(2)^{n-1} + n^2\, 2^{n-1}$$

$$f(n) = (2)^{n-1}\,(n^2 + n + 2)$$

$$f(n) = 2f(\frac{n}{4}) + 10 \qquad f(1) = 1$$

Podemos denotar que la ecuación descrita es una recurrencia no lineal no homogenea. Dado esto, el proceso a seguir será aplicar el cambio de dominio para obtener una recurrencia lineal, hallar la solución homogenea y la solución particular de la nueva ecuación, para luego relacionar las condiciones iniciales.

Cambio de Dominio

Dada la natualeza de la recurrencia, aplicaremos el cambio de dominio dado por $g(m) = n = 4^m$ y definimos a función $\gamma(m) = f(g(m)) = f(n)$

$$\begin{aligned} f(n) = f(g(m)) &= 2f(\frac{4^m}{4}) + 10 \\ &= 2f(4^{m-1}) + 10 \\ \gamma(m) &= 2\gamma(m-1) + 10 \end{aligned}$$

Dada la recurrencia lineal no homogenea en térmimos de gamma, procedemos a encontrar su solución.

Solución Homogénea

Vamos a solucionar la recurrencia homogénea:

$$h(m) - 2h(m-1) = 0$$

Para ello, vamos a asociar su polinomio característico y sus raíces:

$$\lambda - 2 = 0 \quad \Longrightarrow \quad \lambda = 2$$

Dado esto, la solución homogénea de la recurrencia está dada por:

$$h(m) = c_1(2)^m$$

Solución Particular

Para la solución de la parte no homogénea, vamos a suponer una solución $p(m)$ que cumple lo siguiente:

$$p(m) - 2p(m-1) = 10$$

Utiizando la heurística, supondremos que la solución es de la forma $p(m) = c_2$. Dado esto, tratamos de hallar el valor de c_2:

$$\begin{aligned} p(m) - 2p(m-1) &= 10 \\ c_2 - 2c_2 &= 10 \\ c_2 &= -10 \end{aligned}$$

Dado esto, nuestra solución particular esta dada por:

$$p(m) = -10$$

Aplicación de Condiciones iniciales

Para aplicar las condiciones iniciales, tomaremos la solución general de la reccurrencia lineal y aplicaremos el cambio de dominio para regresar a n mediante $m = \log_4 n = \dfrac{\log_2 n}{\log_2 4} = \dfrac{\log_2 n}{2}$.

$$\gamma(m) = c_1(2)^m - 10 \qquad \Longrightarrow \qquad f(n) = c_1(2)^{\frac{\log_2 n}{2}} - 10 = c_1\left((2)^{\log_2 n}\right)^{\frac{1}{2}} - 10 = c_1\sqrt{n} - 10$$

Aplicamos la condición $f(1) = 1$ para determinar el valor de c_1

$$f(1) = 1 = c_1\sqrt{1} - 10 \quad \Longrightarrow \quad c = 11$$

En ese orden de ideas, la solución cerrada de la ecuación de recurrencia es:

$$f(n) = 11\sqrt{n} - 10$$

Bono

Realizamos el mismo proceso de la primera parte para comparar RadixSort y CountSort

4. RadixSort

Resultado con 100 datos:

```
ArrayList size: 100 recursive list size: 100
First location of number 2876. In ArrayList: 65 in recursive list:
65
Last location of number 2876. In ArrayList: 65 in recursive list:
65
Element in position 65. ArrayList: 2876 recursive list 2876
The list is not sorted
Number of even elements. Divide and conquer: 50. recursive list 50
Maximum. Divide and conquer: 9922. recursive list 9922
Position of value 2876. Divide and conquer: 65. Value in numbers
2876
Sorting algorithm: Radix
Numbers sorted. Total time(milliseconds): 1
Position of value 2876. Divide and conquer: 30. Value in numbers
2876
```

Resultado con 1000 datos:

```
ArrayList size: 1000 recursive list size: 1000
First location of number 1465. In ArrayList: 690 in recursive list:
690
Last location of number 1465. In ArrayList: 690 in recursive list:
690
Element in position 690. ArrayList: 1465 recursive list 1465
The list is not sorted
Number of even elements. Divide and conquer: 496. recursive list
496
Maximum. Divide and conquer: 9994. recursive list 9994
Position of value 1465. Divide and conquer: 690. Value in numbers
1465
Sorting algorithm: Radix
Numbers sorted. Total time(milliseconds): 1
Position of value 1465. Divide and conquer: 149. Value in numbers
1465
```

Resultado con 10000 datos:

```
ArrayList size: 10000 recursive list size: 10000
First location of number 2266. In ArrayList: 1472 in recursive
list: 1472
Last location of number 2266. In ArrayList: 8581 in recursive list:
8581
Element in position 8581. ArrayList: 2266 recursive list 2266
The list is not sorted
Number of even elements. Divide and conquer: 4985. recursive list
4985
```

```
Maximum. Divide and conquer: 9998. recursive list 9998
Position of value 2266. Divide and conquer: 1472. Value in numbers
2266
Sorting algorithm: Radix
Numbers sorted. Total time(milliseconds): 8
Position of value 2266. Divide and conquer: 2225. Value in numbers
2266
```

Resultado con 100000 datos:

```
ArrayList size: 100000 recursive list size: 100000
First location of number 120. In ArrayList: 17730 in recursive
list: 17730
Last location of number 120. In ArrayList: 83235 in recursive list:
83235
Element in position 26708. ArrayList: 120 recursive list 120
The list is not sorted
Number of even elements. Divide and conquer: 50175. recursive list
50175
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 120. Divide and conquer: 17730. Value in numbers
120
Sorting algorithm: Radix
Numbers sorted. Total time(milliseconds): 39
Position of value 120. Divide and conquer: 1194. Value in numbers
120
```

Resultado con 100000 datos:

```
ArrayList size: 1000000 recursive list size: 1000000
First location of number 8953. In ArrayList: 12919 in recursive
list: 12919
Last location of number 8953. In ArrayList: 996065 in recursive
list: 996065
Element in position 455830. ArrayList: 8953 recursive list 8953
The list is not sorted
Number of even elements. Divide and conquer: 500119. recursive list
500119
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 8953. Divide and conquer: 12919. Value in numbers
8953
Sorting algorithm: Radix
Numbers sorted. Total time(milliseconds): 153
Position of value 8953. Divide and conquer: 895018. Value in
numbers 8953
```

5. CountSort

Resultado con 100 datos:

```
ArrayList size: 100 recursive list size: 100
First location of number 4046. In ArrayList: 70 in recursive list:
70
Last location of number 4046. In ArrayList: 70 in recursive list:
70
Element in position 70. ArrayList: 4046 recursive list 4046
The list is not sorted
Number of even elements. Divide and conquer: 50. recursive list 50
Maximum. Divide and conquer: 9922. recursive list 9922
Position of value 4046. Divide and conquer: 70. Value in numbers
4046
Sorting algorithm: Count
Numbers sorted. Total time(milliseconds): 0
Position of value 4046. Divide and conquer: 35. Value in numbers
4046
```

Resultado con 1000 datos:

```
ArrayList size: 1000 recursive list size: 1000
First location of number 8856. In ArrayList: 949 in recursive list:
949
Last location of number 8856. In ArrayList: 949 in recursive list:
949
Element in position 949. ArrayList: 8856 recursive list 8856
The list is not sorted
Number of even elements. Divide and conquer: 496. recursive list
496
Maximum. Divide and conquer: 9994. recursive list 9994
Position of value 8856. Divide and conquer: 949. Value in numbers
8856
Sorting algorithm: Count
Numbers sorted. Total time(milliseconds): 1
Position of value 8856. Divide and conquer: 862. Value in numbers
8856
```

Resultado con 10000 datos:

```
ArrayList size: 10000 recursive list size: 10000
First location of number 3941. In ArrayList: 733 in recursive list:
733
Last location of number 3941. In ArrayList: 3742 in recursive list:
3742
Element in position 3742. ArrayList: 3941 recursive list 3941
The list is not sorted
Number of even elements. Divide and conquer: 4985. recursive list
4985
Maximum. Divide and conquer: 9998. recursive list 9998
Position of value 3941. Divide and conquer: 733. Value in numbers
3941
Sorting algorithm: Count
Numbers sorted. Total time(milliseconds): 4
```

```
Position of value 3941. Divide and conquer: 3856. Value in numbers
3941
```

Resultado con 100000 datos:

```
ArrayList size: 100000 recursive list size: 100000
First location of number 2970. In ArrayList: 27314 in recursive
list: 27314
Last location of number 2970. In ArrayList: 90834 in recursive
list: 90834
Element in position 79282. ArrayList: 2970 recursive list 2970
The list is not sorted
Number of even elements. Divide and conquer: 50175. recursive list
50175
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 2970. Divide and conquer: 27314. Value in numbers
2970
Sorting algorithm: Count
Numbers sorted. Total time(milliseconds): 19
Position of value 2970. Divide and conquer: 29569. Value in numbers
2970
```

Resultado con 100000 datos:

```
ArrayList size: 1000000 recursive list size: 1000000
First location of number 3992. In ArrayList: 30015 in recursive
list: 30015
Last location of number 3992. In ArrayList: 980456 in recursive
list: 980456
Element in position 788948. ArrayList: 3992 recursive list 3992
The list is not sorted
Number of even elements. Divide and conquer: 500119. recursive list
500119
Maximum. Divide and conquer: 10000. recursive list 10000
Position of value 3992. Divide and conquer: 30015. Value in numbers
3992
Sorting algorithm: Count
Numbers sorted. Total time(milliseconds): 43
Position of value 3992. Divide and conquer: 398741. Value in
numbers 3992
```

Finalmente, diseñamos una tabla que resuma los tiempos de ejecución para cada tipo de datos en milisegundos

Algoritmo	**100**	**1k**	**10k**	**100k**	**1M**
QuickSort	1	1	9	49	1125
MergeSort	1	2	13	151	966
RadixSort	1	1	8	39	153

Algoritmo	100	1k	10k	100k	1M
CountSort	0	1	4	19	43

In [21]:
```
dict_ = {
    'QuickSort' : {100:1 , 1000:1, 10000: 9, 100000: 49, 1000000: 1125},
    'MergeSort' : {100:1 , 1000:2, 10000: 13, 100000: 151, 1000000: 966},
    'RadixSort' : {100:1 , 1000:1, 10000: 8, 100000:39, 1000000: 153},
    'CountSort' : {100:0 , 1000:1, 10000: 4, 100000: 19, 1000000: 43}
}

plot_execution_values(dict_)
```