EJERCICIO 1

1. Ordenar por Cuenta por distribución

Datos originales:
5T, 0C, 5U, 0O, 9!, 1N, 8S, 2R, 6A, 4A, 1G, 5L, 6T, 6I, 7O, 7N
Clave es el dígito.

Paso 1: Inicializar vector de conteo
Claves van de 0 a 9.
Cuenta[0..9] inicializado a 0
Cuenta: [0,0,0,0,0,0,0,0,0,0]

Paso 2: Contar ocurrencias
5T → Cuenta[5] = 1
0C → Cuenta[0] = 1
5U → Cuenta[5] = 2
0O → Cuenta[0] = 2
9! → Cuenta[9] = 1
1N → Cuenta[1] = 1
8S → Cuenta[8] = 1
2R → Cuenta[2] = 1
6A → Cuenta[6] = 1
4A → Cuenta[4] = 1
1G → Cuenta[1] = 2
5L → Cuenta[5] = 3
6T → Cuenta[6] = 2
6I → Cuenta[6] = 3
7O → Cuenta[7] = 1
7N → Cuenta[7] = 2

Cuenta final: [2,2,1,0,1,3,3,2,1,1]

Paso 3: Transformar a acumulada
Cuenta[0] = 2
Cuenta[1] = 2+2 = 4
Cuenta[2] = 4+1 = 5
Cuenta[3] = 5+0 = 5
Cuenta[4] = 5+1 = 6
Cuenta[5] = 6+3 = 9
Cuenta[6] = 9+3 =12
Cuenta[7] =12+2 =14

Cuenta[8] =14+1 =15
Cuenta[9] =15+1 =16

Cuenta acumulada: [2,4,5,5,6,9,12,14,15,16]

Paso 4: Repartir en vector salida de derecha a izquierda para mantener estabilidad

7N → clave=7 → posición=14 → salida[14]=7N → Cuenta[7]=13
7O → clave=7 → posición=13 → salida[13]=7O → Cuenta[7]=12
6I → clave=6 → posición=12 → salida[12]=6I → Cuenta[6]=11
6T → clave=6 → posición=11 → salida[11]=6T → Cuenta[6]=10
5L → clave=5 → posición=9 → salida[9]=5L → Cuenta[5]=8
1G → clave=1 → posición=4 → salida[4]=1G → Cuenta[1]=3
4A → clave=4 → posición=6 → salida[6]=4A → Cuenta[4]=5
6A → clave=6 → posición=10 → salida[10]=6A → Cuenta[6]=9
2R → clave=2 → posición=5 → salida[5]=2R → Cuenta[2]=4
8S → clave=8 → posición=15 → salida[15]=8S → Cuenta[8]=14
1N → clave=1 → posición=3 → salida[3]=1N → Cuenta[1]=2
9! → clave=9 → posición=16 → salida[16]=9! → Cuenta[9]=15
0O → clave=0 → posición=2 → salida[2]=0O → Cuenta[0]=1
5U → clave=5 → posición=8 → salida[8]=5U → Cuenta[5]=7
0C → clave=0 → posición=1 → salida[1]=0C → Cuenta[0]=0
5T → clave=5 → posición=7 → salida[7]=5T → Cuenta[5]=6

Vector final ordenado
[0C, 0O, 1N, 1G, 2R, 4A, 5T, 5U, 5L, 6A, 6T, 6I, 7O, 7N, 8S, 9!]

2. Cuántas comparaciones y movimientos
   Cuenta por distribución no hace comparaciones directas entre elementos, solo incrementa contadores y luego coloca en posiciones finales, total es O(n) en movimientos.
   Comparado con heapsort, que hace O(n log n) comparaciones y movimientos.

3. Consideraciones sobre estabilidad
   Cuenta por distribución es estable porque procesa del final al principio y mantiene el orden relativo original de los elementos con la misma clave.
   Heapsort no es estable.

4. Orden del tiempo de ejecución
   Seudocódigo
   CuentaPorDistribución
   begin
   for i = u to v hacer Cuenta[i] = 0
   for j = 1 to N incrementar Cuenta[Kj]

```
for i = u+1 to v Cuenta[i] = Cuenta[i] + Cuenta[i-1]
for j = N downto 1 hacer
i = Cuenta[Kj]
S[i] = R[j]
Cuenta[Kj] = i-1
end
```

Análisis
Inicialización O(k)
Conteo O(n)
Acumulación O(k)
Reparto O(n)
Total O(n + k), si k pequeño frente a n, es O(n).