UT9 - PD2

Ejercicio 1

1.
- Propuesta por T.N. Hibbard en 1963. Los incrementos son 1, 3, 7, 15, 31.
- Robert Sedgewick propuso varias secuencias en 1986, siendo una de las más conocidas. Los incrementos son 1, 5, 19, 41, 109.
- Propuesta por H. Tokuda en 1992, en esta secuencia los incrementos son 1, 4, 9, 20, 46, 103.

2.

Análisis del tiempo de ejecución: si consideramos en lo visto en la clase el orden del tiempo de ejecución de n^1.26 como el caso más adverso de este.

La duración del Shell Sort se ajusta a la secuencia de incrementos utilizada. En el menor caso, el tiempo de ejecución del Shell Sort utilizando la secuencia de Sedgewick es aproximadamente O(n^(3/2)) y para la secuencia de Tokuda es aproximadamente O(n^(5/4))

3. **Cada uno de los métodos aplicado a tablas**

T.N. Hibbard(1, 3, 7, 15, 31) → 256 - 458 - 655 - 298 - 043 - 648 - 778 - 621 - 655 - 019 - 124 - 847

256	458	655	298	043	648	778	621	655	019	124	847
256	458	019	124	043	648	778	621	655	655	298	847
256	458	019	124	043	648	778	621	655	655	298	847
124	043	019	256	298	648	655	458	655	778	621	847
124	043	019	256	298	648	655	458	655	778	621	847
019	043	124	256	298	458	621	648	655	655	778	847

Robert Sedgewick (1,5) → 256 - 458 - 655 - 298 - 043 - 648 - 778 - 621 - 655 - 019 - 124 - 847

256	458	655	298	043	648	778	621	655	019	124	847
124	458	621	298	019	256	778	655	655	043	648	847
124	458	621	298	019	256	778	655	655	043	648	847
019	043	124	256	298	458	621	648	655	655	778	847

H. Tokuda (1, 4, 9) → 256 - 458 - 655 - 298 - 043 - 648 - 778 - 621 - 655 - 019 - 124 - 847

256	458	655	298	043	648	778	621	655	019	124	847
019	124	655	298	043	648	778	621	655	256	458	847
019	124	655	298	043	648	778	621	655	256	458	847
019	124	458	298	043	256	655	621	655	468	778	847
019	124	458	298	043	256	655	621	655	468	778	847
019	043	124	256	298	458	621	648	655	655	778	847

Ejercicio 2

1. Este algoritmo ya queda ordenado antes de la última iteración.

44	**55**	**12**	**42**	**94**	**18**	**6**	**67**	**I**
6	44	55	12	42	94	18	67	**1**
6	12	44	55	42	94	18	67	**2**
6	12	18	44	55	42	94	67	**3**
6	12	18	42	44	55	94	67	**4**
6	12	18	42	44	55	94	67	**5**
6	12	18	42	44	55	94	67	**6**
6	12	18	42	44	55	67	94	**7**
6	12	18	42	44	55	67	94	**8**

2.

Burbuja Mejorado con Bandera

Para esta solución utilice una bandera para determinar si se llevaron a cabo intercambios en una ocasión anterior.

Pseudocódigo:

```
algoritmo BurbujaMejoradoConBandera(arr)
    n ← longitud(arr)
    para i desde 1 hasta n - 1 hacer
        intercambiado ← falso
```

```
        para j desde 1 hasta n - i hacer
            si arr[j - 1] > arr[j] entonces
                intercambiar(arr[j - 1], arr[j])
                intercambiado ← verdadero
            fin si
        fin para
        si no intercambiado entonces
            romper
        fin si
    fin para
fin algoritmo
```

Burbuja Bidireccional (Cocktail Shaker Sort)

Esta variante del algoritmo de burbuja realiza pasadas en ambas direcciones (hacia adelante y hacia atrás) en cada iteración, lo que puede ayudar a mover los elementos más grandes y más pequeños a sus posiciones correctas más rápidamente.

Pseudocódigo:

```
algoritmo BurbujaBidireccional(arr)
    inicio ← 0
    fin ← longitud(arr) - 1
    intercambiado ← verdadero
    mientras intercambiado hacer
        intercambiado ← falso
        para i desde inicio hasta fin - 1 hacer
            si arr[i] > arr[i + 1] entonces
                intercambiar(arr[i], arr[i + 1])
                intercambiado ← verdadero
            fin si
```

```
        fin para
        si no intercambiado entonces
            romper
        fin si
        fin ← fin - 1
        para i desde fin - 1 hasta inicio hacer
            si arr[i] > arr[i + 1] entonces
                intercambiar(arr[i], arr[i + 1])
                intercambiado ← verdadero
            fin si
        fin para
        inicio ← inicio + 1
    fin mientras
fin algoritmo
```

3.

Tabla Método 1

44	55	12	42	94	18	6	67	I
6	44	55	12	42	94	18	67	**1**
6	12	44	55	42	94	18	67	**2**
6	12	18	44	55	42	94	67	**3**
6	12	18	42	44	55	94	67	**4**
6	12	18	42	44	55	94	67	**5**
6	12	18	42	44	55	94	67	**6**
6	12	18	42	44	55	67	94	**7**

Tabla Método 2

44	55	12	42	94	18	6	67	I

6	44	55	12	42	94	18	67	**1**
6	12	44	55	42	94	18	67	**2**
6	12	18	44	55	42	94	67	**3**
6	12	18	42	44	55	94	67	**4**
6	12	18	42	44	55	94	67	**5**
6	12	18	42	44	55	94	67	**6**
6	12	18	42	44	55	67	94	**7**

Algoritmo de “shakersort”:

44	55	12	42	94	18	6	67	I
6	44	55	12	42	18	6	94	**1**
6	12	44	55	42	18	67	94	**2**
6	12	18	44	42	55	67	94	**3**
6	12	18	42	44	55	67	94	**4**