**Trabajo Cooperativo en sesión sincrónica**

1. Publicar Hoja de Trabajo con los problemas a resolver en el Aula Sicua+
2. Usar Collaborate / Zoom para conformación de grupos de trabajo (parejas conformadas aleatoriamente). Trabajo en grupos durante 5 minutos por problema.
3. Usar Padlet para la publicación de las respuestas trabajadas por los grupos.

Cada grupo publica sus respuestas propuestas en el Padlet de la sesión.

1. Regreso a Sala General (Re-agrupamiento del curso).
2. Discusión de resultados de los problemas publicados en el Padlet entre el profesor y losestudiantes.

Código de Referencia: <https://...>

Algunos algoritmos de ordenamiento iterativos son: Selection, Insertion y Shell

**1. Algoritmo SelectionSort**

1 /\*\*

2 \* Selection sort para un arreglo generico aplicando la comparación less

3 \*/

4 public static void sort(Comparable[] a)

5 {

6 int N = a.length;

7 **for** (int i = 0; i < N; i++)

8 {

9 int min = i;

10 **for** (int j = i+1; j < N; j++)

11 **if** (less(a[j], a[min]))

12 min = j;

13 exch(a, i, min);

14 }

15 }

**Problema 1.1 Propiedades**

|  |  |
| --- | --- |
| **Propiedad** | **Solución** |
| Complejidad temporal del algoritmo en notación O( …) en su peor caso |  |
| Complejidad temporal del algoritmo en notación O( …) en su mejor caso |  |
| ¿El algoritmo es *In-place*? Si o No y justificar |  |
| ¿El algoritmo es *Estable*? Si o No y justificar |  |

**Problema 1.2 Aplicación del algoritmo**

Conjunto de datos a ordenar (representación en Lista)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem | J | W | F | Z | L | S | C | X | G | N | A | U | D | P |

Ilustre como queda el conjunto de datos después de que el primer elemento queda ordenado aplicando el algoritmo selectionSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Ilustre como queda el conjunto de datos después de que el segundo elemento queda ordenado aplicando el algoritmo selectionSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Ilustre como queda el conjunto de datos después de que el tercer elemento queda ordenado aplicando el algoritmo selectionSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Ilustre como queda el conjunto de datos después de que el cuarto elemento queda ordenado aplicando el algoritmo selectionSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Problema 1.3 Tratamiento de elementos repetidos**

Considere el caso cuando hay elementos repetidos en la lista. ¿Qué pasa con el algoritmo en este caso?

**2. Algoritmo InsertionSort**

1 / \*\*

2 \* Insertion sort para un arreglo generico aplicando la comparación less

3 \*/

4 public static void sort(Comparable[] a)

5 {

6 int N = a.length;

7 **for** (int i = 0; i < N; i++)

8 **for** (int j = i; j > 0; j--)

9 **if** (less(a[j], a[j-1]))

10 exch(a, j, j-1);

11 **else** break;

12 }

**Problema 2.1 Propiedades**

|  |  |
| --- | --- |
| **Propiedad** | **Solución** |
| Complejidad temporal del algoritmo en notación O( …) en su peor caso |  |
| Complejidad temporal del algoritmo en notación O( …) en su mejor caso |  |
| ¿El algoritmo es *In-place*? Si o No y justificar |  |
| ¿El algoritmo es *Estable*? Si o No y justificar |  |

**Problema 2.2 Aplicación del algoritmo**

Conjunto de datos a ordenar (representación en Lista)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem | J | W | F | Z | L | S | C | X | G | N | A | U | D | P |

Ilustre como queda el conjunto de datos después de que el primer elemento queda ordenado aplicando el algoritmo insertionSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Ilustre como queda el conjunto de datos después de que el segundo elemento queda ordenado aplicando el algoritmo insertionSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Ilustre como queda el conjunto de datos después de que el tercer elemento queda ordenado aplicando el algoritmo insertionSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

Ilustre como queda el conjunto de datos después de que el cuarto elemento queda ordenado aplicando el algoritmo insertionSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Problema 2.3 Tratamiento de elementos repetidos**

Considere el caso cuando hay elementos repetidos en la lista. ¿Qué pasa con el algoritmo en este caso?

**3. Algoritmo ShellSort**

1 /\*\*

2 \* shellSort sort para un arreglo generico aplicando la comparación less

3 \*/

4 public static void sort(Comparable[] a)

5 {

6 int N = a.length;

7 int h = 1;

8 **while** (h < N/3) // Se calcula el tamaño del primer gap. La lista se h-ordena con este tamaño

9 h = 3\*h + 1; // 1, 4, 13, 40, 121, 364, ...

10 **while** (h >= 1)

11 { // h-sort the array.

12 **for** (int i = h; i < N; i++)

13 {

14 **for** (int j = i; j >= h && less(a[j], a[j-h]); j -= h)

15 exch(a, j, j-h);

16 }

17 h = h/3;

18 }

19 }

**Problema 3.1 Propiedades**

|  |  |
| --- | --- |
| **Propiedad** | **Solución** |
| Complejidad temporal del algoritmo en notación O( …) en su peor caso |  |
| Complejidad temporal del algoritmo en notación O( …) en su mejor caso |  |
| ¿El algoritmo es *In-place*? Si o No y justificar |  |
| ¿El algoritmo es *Estable*? Si o No y justificar |  |

**Problema 3.2 Aplicación del algoritmo**

Conjunto de datos a ordenar (representación en Lista)

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem | J | W | F | Z | L | S | C | X | G | N | A | U | D | P |

3.5.1 ¿Cuál es el valor más grande de h que encuentra el algoritmo (resultado del primer *while*)? \_\_\_\_\_  
Con este valor de h se aplica el primer h-sort del conjunto de datos.

3.5.2 Ilustre como queda el conjunto de datos después de terminar el h-sort con el valor h dado en el punto anterior aplicando el algoritmo shellSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

3.5.3 ¿Cuál es el segundo valor de h para iniciar el segundo h-sort del conjunto de datos? \_\_\_\_\_

3.5.4 Ilustre como queda el conjunto de datos después de terminar el h-sort con el valor h dado en el punto anterior aplicando el algoritmo shellSort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| posic | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
| elem |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

**Problema 3.3 Tratamiento de elementos repetidos**

Considere el caso cuando hay elementos repetidos en la lista. ¿Qué pasa con el algoritmo en este caso?