Algoritmo de Ordenamiento por Inserción

Computer Science

CS1100 - Introducción a Ciencia de la Computación

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Logro de la Sesión

Al finalizar esta sesión, estarás en la capacidad de:

- Comprender el algoritmo de ordenamiento por inserción.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Logro de la Sesión

Al finalizar esta sesión, estarás en la capacidad de:

- Comprender el algoritmo de ordenamiento por inserción.
- Implementar el algoritmo de ordenamiento por inserción en Python.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Insertion Sort

Revisión del algoritmo...

- £Cúal es el algoritmo del Insertion Sort?

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Insertion Sort

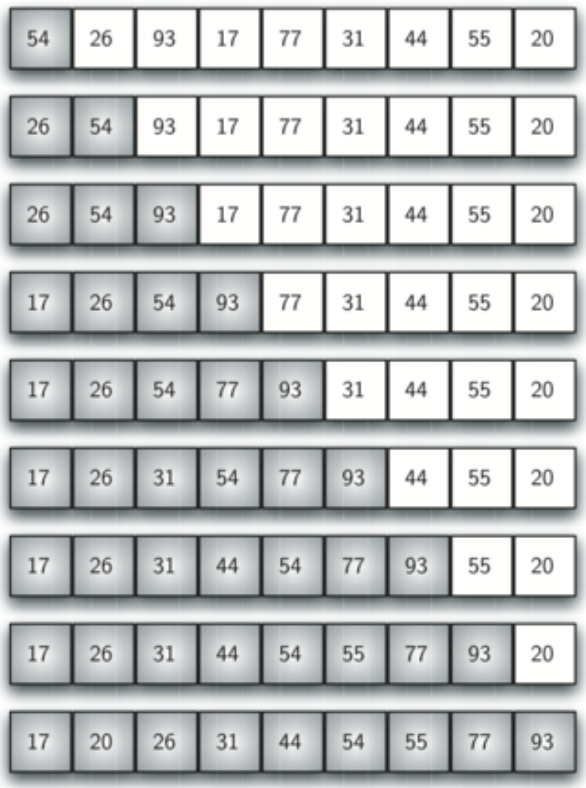
Revisión del algoritmo...

- £Cúal es el algoritmo del Insertion Sort?
- Tal como observamos en el vídeo, el proceso consiste en revisar cada elemento del conjunto, y, de ser este elemento menor al que está a su izquierda, deberá cambiar de posición con el de su izquierda. Esta comparación con el de su izquierda se repite hasta que encuentre su posición adecuada. Finalmente, se pasa al siguiente elemento, y se vuelve a realizar el proceso.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Insertion Sort

£Qué ocurre luego de cada pasada.



UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Ordenar los elementos

£En qué nos fijamos primero cuando queremos ordenar una lista de menor a mayor?

```
[23,29, 13, 34, 17, 25]
```

- Buscamos inmediatamente el menor elemento.
- Un procedimiento natural al ordenar es pensar primero en que el primer número va ala principio.
- £Cómo puede hacer esto el computador?

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Escribe una función que reciba una lista y encuentre el menor elemento de la lista.

```
1  def indice_menor_elemento(lista):
2      menor = lista[0]
3      indice = 0
4      for i in range(len(lista)):
5          if lista[i] < menor:
6              menor = lista[i]
7              indice = i
8      return indice
9
10 lista = [23,29,13,34,17,25]
11 print(indice_menor_elemento(lista))
```

```
1  2
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Question:

£Qué imprime el siguiente código?

```
1 lista = [3,2,1]
2 i = indice_menor_elemento(lista)
3 aux = lista[i]
4 lista[i] = lista[0]
5 lista[0] = aux
6 print(lista)
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Question:

£Qué imprime el siguiente código?

```
1 lista = [3,2,1]
2 i = indice_menor_elemento(lista)
3 aux = lista[i]
4 lista[i] = lista[0]
5 lista[0] = aux
6 print(lista)
```

```
1 [1, 2, 3]
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Ordenar los elementos

Una vez que pusimos el primer elemento en la primera posición, £Qué buscamos?

```
[ 13, 29, 23, 34, 17, 25]
```

- Buscamos inmediatamente el menor elemento.
- Continuamos buscando el segundo elemento para situarlo en la segunda posición.
- Cuando queremos situar el segundo elemento en la seguna posición, ya sabemos que el menor elemento está en la primera posición.
- En este caso, buscar el segundo menor elemento es lo mismo que buscar el menor elemento en la sublista que parte de la segunda posición.

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Ordenar los elementos

Una vez que pusimos el primer elemento en la primera posición, £Qué buscamos?

```
[ 13, 29, 23, 34, 17, 25]
```

- En general, al buscar el `n`-ésimo menor elemento sabremos que los `n-1` menores elementos ya están donde corresponde

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Question:

£Cómo generalizamos este código?

```
1 lista = [3,4,2,1]
2 i = indice_menor_elemento(lista)
3 aux = lista[i]
4 lista[i] = lista[0]
5 lista[0] = aux
6 print(lista)
```

```
1 [1, 4, 2, 3]
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Question:

£Cómo generalizamos este código?

```
1  def ordenar(lista):
2      for i in range(len(lista)):
3          indice = indice_menor_elemento(lista[i:])
4          aux = lista[indice + i]
5          lista[indice + i] = lista[i]
6          lista[i] = aux
7
8  lista = [3,4,2,1]
9  ordenar(lista)
10 print(lista)
```

```
1  [1, 2, 3, 4]
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Question:

£Cuántas operaciones realizar el algoritmo?

```
1  def ordenar(lista):
2      for i in range(len(lista)):
3          indice = indice_menor_elemento(lista[i:])
4          aux = lista[indice + i]
5          lista[indice + i] = lista[i]
6          lista[i] = aux
7
8  lista = [3,4,2,1]
9  ordenar(lista)
10 print(lista)
```

- Para situar el menor elemento, realiza *n-1* operaciones.
- Para situar el segundo menor, *n-2* operaciones.
- Total de operaciones: $\frac{n(n-1)}{2}$

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

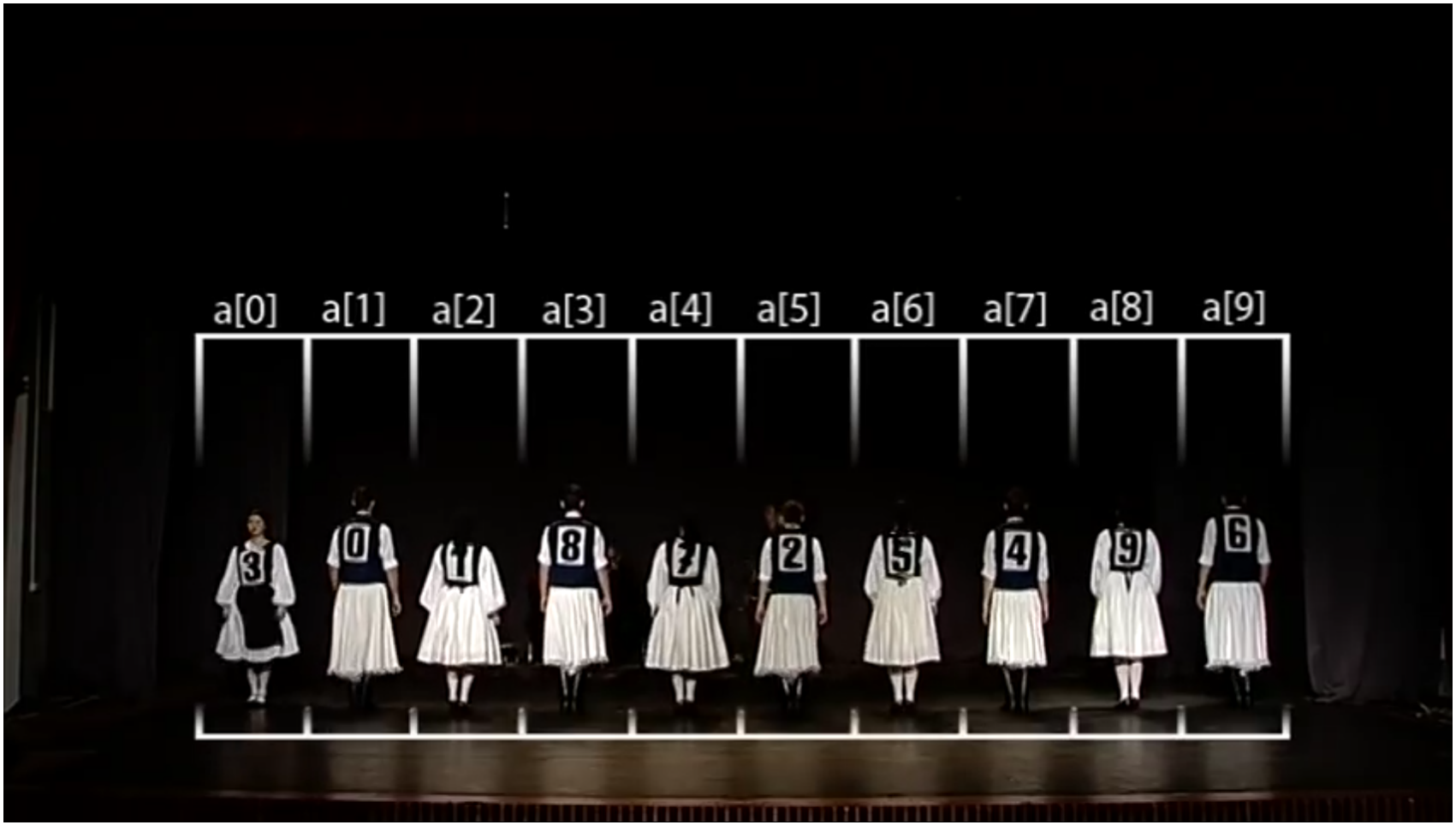
Question:

£Cómo mejoramos el algoritmo?

```
1  def ordenar(lista):
2      for i in range(len(lista)):
3          indice = indice_menor_elemento(lista[i:])
4          aux = lista[indice + i]
5          lista[indice + i] = lista[i]
6          lista[i] = aux
7
8  lista = [3,4,2,1]
9  ordenar(lista)
10 print(lista)
```

- Cambiaremos el enfoque de nuestro algoritmo.
- En vez de preocuparnos de situar el `i`-ésimo elemento en la posición `i`, nos preocuparemos de que los primeros `i` elementos estén ordenados.

Insert Sort



- Ver este video

Insert Sort

- £Cómo lo programamos?
- Debemos mover cada elemento (de izquierda a derecha), tan a la izquierda como sea necesario, intercambiándolo en cada paso con su vecino.

```
1  def ordenar(lista):
2      for i in range(1, len(lista)):
3          h=i
4          while h > 0 and lista[h] < lista[h - 1]:
5              aux = lista[h]
6              lista[h] = lista[h - 1]
7              lista[h - 1] = aux
8              h=h-1
9
10 lista = [3,4,2,1]
11 ordenar(lista)
12 print(lista)
```

```
1  [1, 2, 3, 4]
```

Insertion Sort - Tiempo de ejecución

Tiempo de ejecución en diversos casos

- El peor caso: O(n x n).
- El mejor caso: O(n).
- El caso promedio para un arreglo aleatorio: O(n x n).
- El caso "casi ordenado": O(n).

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Ejemplo 1

Enunciado

Suponga que usted tiene que ordenar la siguiente lista de números: [15, 5, 4, 18, 12, 19, 14, 10, 8, 20] £Cuál de las siguientes listas representa la lista parcialmente ordenada tras tres pasadas completas del ordenamiento por inserción?

```
1 [4, 5, 12, 15, 14, 10, 8, 18, 19, 20]
2 [15, 5, 4, 10, 12, 8, 14, 18, 19, 20]
3 [4, 5, 15, 18, 12, 19, 14, 10, 8, 20]
4 [15, 5, 4, 18, 12, 19, 14, 8, 10, 20]
```

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Cierre

Conclusiones

- £Qué podemos decir sobre el tiempo de ejecución de este algoritmo?

UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA