	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: TISTA GARCÍA EDGAR

Asignatura: ESTRUCTURAS DE DATOS Y ALGORITMOS II

Grupo: 5

No de Práctica(s): 01

Integrante(s): GÓMEZ LUNA ALEJANDRO

No. de Equipo de cómputo empleado: 43

No. de Lista o Brigada:

Semestre: 2020-1

Fecha de entrega: 13/Agosto/2019

Observaciones:

CALIFICACIÓN: _____

- Objetivo de la práctica
  EL ESTUDIANTE IDENTIFICARÁ LA ESTRUCTURA DE LOS ALGORITMOS DE ORDENAMIENTO POR SELECCIÓN E INSERCIÓN.

- Desarrollo
  a) Ejercicios de la guía: Para este caso, se buscó la forma en cómo se implementan los algoritmos de ordenamiento Insertion-Sort y Selection-Sort en Python, pues estos no se encuentran en la guía. Empezaremos con Insertion-Sort.
  La implementación encontrada de Insertiorn-Sort en Python nos muestra el siguiente código:

[3]:
```
def instertion_sort(seq):

    for slice_end in range(len(seq)):
        pos = slice_end
        # inserting elements at correct position
        while pos > 0 and seq[pos] < seq[pos - 1]:
            (seq[pos], seq[pos - 1]) = (seq[pos - 1], seq[pos])
            pos = pos - 1


# test input
l = [54, 20, 93, 17, 70, 31, 42, 55, 21]
instertion_sort(l)
print(l)
```

```
[17, 20, 21, 31, 42, 54, 55, 70, 93]
```

Con base en este código encontrado, se presentan varias características distintas en cuanto al pseudocódigo visto en clase. Para empezar, no se utiliza una variable para almacenar el tamaño de la lista, pues el resultado de la función len() se pone directamente en el valor frontera del primer for. Además de esto, la variable slice_end tendrá un valor inicial de 0, mientras que en clase vimos que se empieza con un valor de 1. Otra característica distinta que observamos es en cuanto a la variable pos, la cual toma directamente el valor de la variable slice_end, sin decrementarle en uno su valor y luego almacenarlo, como vimos en clase. Además, no se usa una variable auxiliar para almacenar el elemento que se encuentra en el índice correspondiente al número de iteración del for. De igual forma, en la comprobación del while, se observa una especie de mejora con respecto a la vista en clase, pues, si el próximo elemento a comparar en la lista es mayor que su antecesor, entonces no tiene que seguir con los demás elementos a su izquierda, porque ya está en su respectiva posición.
Por último, se observa que, como Python permite la asignación paralela, en caso de cumplirse la condición del while, se irán intercambiando los elementos que aún no ha sido ordenado a la izquierda hasta llegar a su posición correspondiente.

La implementación encontrada de Selection-Sort en Python nos muestra el siguiente código:

[8]:
```
# Sorting function
def selection_sort(l):

    for start in range(len(l)):
        min_pos = start

        for i in range(start, len(l)):
            if l[i] < l[min_pos]:
                min_pos = i
        # swap values
        (l[start], l[min_pos]) = (l[min_pos], l[start])

# Test input
l = [52, 24, 99, 11, 70, 33, 44, 55, 13]
selection_sort(l)
print(l)
```

```
[11, 13, 24, 33, 44, 52, 55, 70, 99]
```

Con base en este código encontrado se presentan ligeras características diferentes en cuanto al pseudocódigo visto en clase. Lo primero es que, al igual que en el código anterior, no se hace uso de una variable para almacenar la longitud de la lista.
La otra característica diferente es que, como Python permite la asignación paralela, no se tiene que hacer uso de la función intercambio, pues esta acción se puede realizar en una sola línea de código,
Respecto a lo demás, el código se apega al pseudocódigo lo visto en clase, mediante el uso de dos for, uno anidado dentro del otro, se irá recorriendo la lista elemento por elemento, hasta encontrar el mínimo, intercambiarlo con el valor que se encuentra con su posición correspondiente y así, ir ordenando la lista, sin pasar por aquellos elementos que ya se encuentran ordenados.

b) Ejercicios propuestos:
   a. Ejercicio1: En la biblioteca de ordenamientos.h encontramos dos funciones: selectionSort e insertionSort. La función selectionSort nos permite implementar el algoritmo de ordenamiento Selection sort que se vió en clase. La función empieza por declarar tres variables de tipo entero; indiceMenor, i y j. La variable i servirá para ir contabilizando y controlando las iteraciones que se vayan realizando. Se tienen dos for, uno dentro del otro. El for más externo irá almacenando el número de iteración en el que nos encontramos en la variable indiceMenor, con la finalidad que en el for interno, se puedan ir comparando todos los elementos restantes a la derecha de este. Cabe resaltar que los elementos se irán ordenando de izquierda a derecha, sin pasar por aquellos que ya han sido puestos en su posición correspondiente. En cada recorrido del arreglo se escogerá el elemento más pequeño que aún no esté en su posición correcta, y se guardará el índice en el que se encuentra, para que, una vez que finalice el for interno, se pueda posicionar correctamente. Siempre que se realice este posicionamiento, se llamará a la función swap, la cual sirve para cambiar de posición a dos elementos dentro del arreglo.
      La función insertionSort nos permite implementar el algoritmo de ordenamiento Insertion sort que se vió en clase. La función empieza por declarar tres variables de tipo entero; i, j y aux. La variable i será la que lleve el número de iteración del for externo.

Es importante remarcar que la variable i empezará desde 1, pues también llevará el control de los índices del arreglo, y como el primer elemento que está en el índice cero ya se toma como “ordenado”, entonces empezará a partir de este. Posteriormente se hace uso de la variable auxiliar j, la cual almacenará el número de iteración en el que nos encontramos, y de igual forma la variable aux almacenará el elemento que estamos ordenando en esa iteración. Una vez almacenados estos valores, se presenta un ciclo de repetición while, el cual solamente se ejecutará cuando la variable j sea mayor a 0 y cuando los elementos a la izquierda del elemento que estamos ordenando sean menores a este, pues esto nos indica que el elemento en cuestión no se encuentra en su posición adecuada. Por último, posicionaremos al elemento en su posición correspondiente.

En cuanto a las funciones que vienen en la biblioteca utilidades.h, encontramos tres.

La primera es la función swap, la cual, mediante el paso por referencia de dos variables y el uso de apuntadores, nos permite intercambiar los valores de estas dos variables.

La segunda es la función printArray, la cual nos permite imprimir todos los elementos en un arreglo. Los parámetros que se le pasan es el arreglo que queramos imprimir y su tamaño.

La tercera es la función printSubArray, la cual nos permite imprimir todos los elementos que se encuentran dentro de una porción del arreglo. Nosotros le pasamos como parámetros el arreglo en cuestión y, mediante valores frontera, delimitar que porción del arreglo queremos imprimir.

b. Ejercicio 2: Para este ejercicio, se pedía crear un arreglo con 10 elementos y llenarlo con elementos de maneara aleatoria. Para llenar el arreglo de forma aleatoria se hizo uso de la función srand y rand, dentro de la biblioteca time.h y stdlib.h. Una vez llenado el arreglo con elementos aleatorios entre 1 y 30, se imprimía en pantalla los elementos del arreglo, haciendo uso de la función printArray proporcionada en la biblioteca de utilidades.h. Después, se creó un pequeño menú en donde se le pedía al usuario que seleccionara la opción correspondiente al algoritmo de ordenamiento con el que quería ordenar el arreglo.

   Con la finalidad de mostrar la manera en como se iba ordenando el arreglo con la función insertionSort y selectionSort, cada que finalizara la última instrucción del for externo, se imprimía en pantalla el arreglo, pues de esta manera se pueda visualizar como se va ordenando el arreglo cada que se recorre.

   Para finalizar, se muestra en pantalla el arreglo ya ordenado.

   Algo que sucedió cuando se quiso implementar la función de insertionSort, es que se mostró un error de compilación, pues no se estaba utilizando el identificador correcto en algunas partes de la función.

   Es importante notar que, aunque existan elementos repetidos dentro del arreglo,no fallan los algoritmos de ordenamiento.

```
El arreglo creado tiene los elementos:
9 25 24 26 29 26 9 28 17 7
    ***Bienvenido al programa de ordenamiento***
Seleccione el numero de la opcion que desee para ordenar el arreglo:
1) Insertion Sort
2) Selection sort
1
Ordenando el arreglo...
9 25 24 26 29 26 9 28 17 7
9 24 25 26 29 26 9 28 17 7
9 24 25 26 29 26 9 28 17 7
9 24 25 26 29 26 9 28 17 7
9 24 25 26 26 29 9 28 17 7
9 9 24 25 26 26 29 28 17 7
9 9 24 25 26 26 28 29 17 7
9 9 17 24 25 26 26 28 29 7
7 9 9 17 24 25 26 26 28 29
El arreglo ordenado es:
7 9 9 17 24 25 26 26 28 29
Program ended with exit code: 175
```

```
El arreglo creado tiene los elementos:
25 21 30 1 2 21 30 9 21 4
    ***Bienvenido al programa de ordenamiento***
Seleccione el numero de la opcion que desee para ordenar el arreglo:
1) Insertion Sort
2) Selection sort
2
Ordenando el arreglo...
1 21 30 25 2 21 30 9 21 4
1 2 30 25 21 21 30 9 21 4
1 2 4 25 21 21 30 9 21 30
1 2 4 9 21 21 30 25 21 30
1 2 4 9 21 21 30 25 21 30
1 2 4 9 21 21 30 25 21 30
1 2 4 9 21 21 21 25 30 30
1 2 4 9 21 21 21 25 30 30
1 2 4 9 21 21 21 25 30 30
El arreglo ordenado es:
1 2 4 9 21 21 21 25 30 30
Program ended with exit code: 175
```

c. Ejercicio 3: Para este ejercicio se puede observar que los mismos algoritmos de ordenamiento de Insertion sort y Selection sort se han implementado en el lenguaje de programación Java. Debido a esta implementación en Java, se pueden observar varias características importantes del paradigma orientado a objetos. Una de estas características importantes son las clases, las cuales nos ayudan a modelar entidades de la vida real y, para este caso, se observa que cada algoritmo de ordenamiento está siendo modelado a partir de una clase, y dicha clase tiene un solo comportamiento, el cual es la función que ordena el arreglo. Otra característica importante es la creación de objetos a partir de una clase, como podemos observar en el método main, en donde se crea una instancia de la clase selección, es decir, se crea un objeto a partir de la clase selección. De igual forma podemos observar como los modificadores de acceso nos permiten acceder a métodos de otras clases sin limitación alguna.
   En cuanto al código, no cabe mucho por resaltar, pues se tienen las mismas características básicas que las vistas en el lenguaje C. Solamente existen pequeñas variaciones, por el cambio de lenguaje, como el uso de for each, el cual es usado por la función imprimirArreglo y nos permite realizar la función de imprimir el arreglo en pantalla de manera más simple.
d. Ejercicio 4:

- Conclusiones
- Referencias
    - https://djangocentral.com/insertion-sort-in-python/. Recuperado el 12/08/2019 a las 18:47
    - https://djangocentral.com/selection-sort-in-python/ Recuperado el 12/08/2019 a las 19:05