Estructura de Datos (2021-1)
Laboratorio 3

Francisco Ignacio Salinas Alarcón
frsalinas@udec.cl
Matrícula: 2015442662

April 20, 2021

Ejercicios:

1. Desarrollo: Cree en Python 3.9 un archivo LinkedList y ArrayList.
En LinkedList defino dos clases, la clase nodo y la clase LinkedList, donde defino el comportamiento del método de listas linkeadas. Cree las funciones de añadir al inicio, eliminar al final, dar el tamaño de la lista, buscar un elemento dado un índice e imprimir la lista usando la clase Nodo para desarrollar las funciones.
Código adjunto al informe. Capturas:

LinkedList.py > ...

```python
class Nodo():
    def __init__(self, elemento):
        self.elemento = elemento
        self.siguiente = None

class LinkedList():
    def __init__(self):
        self.cabecera = None
        self.contador=0

    def agregar_elemento_inicio(self, elemento):
        nuevo_nodo = Nodo(elemento)
        nuevo_nodo.siguiente=self.cabecera
        self.cabecera=nuevo_nodo

    def eliminar_elemento_final(self):
        if self.cabecera is None:
            print("La lista no contiene elementos para eliminar.")
            return
        elemento_eliminar = self.cabecera
        while elemento_eliminar.siguiente.siguiente is not None:
            elemento_eliminar = elemento_eliminar.siguiente
        elemento_eliminar.siguiente = None

    def tamaño_lista(self):
        if self.cabecera is None:
            return 0
        inicio_lista=self.cabecera
        while inicio_lista is not None:
            self.contador=self.contador+1
            inicio_lista=inicio_lista.siguiente
            return self.contador

    def buscar_por_indice(self, indice):
        elemento_actual=self.cabecera
        while(elemento_actual):
            if (self.contador==indice-1):
                return elemento_actual.elemento
            self.contador +=1
            elemento_actual=elemento_actual.siguiente
```

```
33
34      def buscar_por_indice(self, indice):
35          elemento_actual=self.cabecera
36          while(elemento_actual):
37              if (self.contador==indice-1):
38                  return elemento_actual.elemento
39              self.contador +=1
40              elemento_actual=elemento_actual.siguiente
41
42
43      def imprimir_lista(self):
44          if self.cabecera is None:
45              print("La lista no contiene elementos.")
46              return
47          else:
48              elemento_cabecera = self.cabecera
49              while elemento_cabecera is not None:
50                  print(elemento_cabecera.elemento , " ")
51                  elemento_cabecera = elemento_cabecera.siguiente
```

De la misma forma, para ArrayList, cree una clase que contiene las mismas funciones que LinkedList solo que en este caso se ocupan array de python y es un método mas directo.
Código adjunto al informe. Capturas:

ArrayList.py > ...

```
1   import array as arr
2
3   class ArrayList():
4       def __init__(self):
5           self.elementos=arr.array('i',[])
6
7       def agregar_elemento_inicio(self, elemento_int):
8           self.elementos.insert(0,elemento_int)
9
10      def eliminar_elemento_final(self):
11          if len(self.elementos)==0:
12              return "La lista no contiene elementos para eliminar."
13          else:
14              return self.elementos.pop()
15
16      def tamaño_lista(self):
17          if len(self.elementos)==0:
18              return "La lista no contiene elementos."
19          else:
20              self.tamaño=len(self.elementos)
21              return self.tamaño
22
23      def buscar_por_indice(self, indice):
24          if len(self.elementos)==0:
25              return "La lista no contiene elementos."
26          else:
27              self.elemento_del_indice=self.elementos[indice-1]
28              return self.elementos[indice-1]
29
30      def lista_completa(self):
31          if len(self.elementos)==0:
32              return "La lista no contiene elementos."
33          else:
34              return self.elementos
35
```

2. Desarrollo: Para usar la clase ArrayList cree un archivo python llamado mainArrayList.py, donde este archivo llama a la clase y crea una interacción con el cliente para usar las funciones de la clase. Doy como opción las funciones que contiene la clase ArrayList y poder interactuar a través de la terminal con esta. También imprime en pantalla los tiempo de ejecución.
Capturas:

mainArrayList.py > {} random

```python
import random        Find related code in 1. Estructuras_de_datos
import time
from ArrayList import ArrayList

lista=ArrayList()


def salir():
    print("Salida...!")
    exit()

def main():
    while True:
        print("ArrayList:")
        print(" ")
        print("Selecciona una opción:\n")
        print("1. Agregar elemento al inicio de la lista.")
        print("2. Eliminar un elemento al final de la lista.")
        print("3. Tamaño de la lista.")
        print("4. Buscar elemento por indice.")
        print("5. Imprimir la lista.")


        opcion=int(input("Opción: "))
        if opcion ==1:
            print("1. Agregar elemento al inicio de la lista.")
            elemento_int=int(input("Ingresar elemento para añadir al inicio de la lista:"))
            tiempo_agregar_inicio=time.perf_counter()
            lista.agregar_elemento_inicio(elemento_int)
            tiempo_agregar_final=time.perf_counter()
            tiempo_total_agregar=format(tiempo_agregar_final-tiempo_agregar_inicio,'.10f')
            print("Tiempo de ejecución de la operación: ",tiempo_total_agregar)
            print("Operación realizada!\n")

        if opcion ==2:
            print("2. Eliminar un elemento al final de la lista.")
            tiempo_eliminar_inicio=time.perf_counter()
            lista.eliminar_elemento_final()
            tiempo_eliminar_final=time.perf_counter()
            tiempo_total_eliminar=format(tiempo_eliminar_final-tiempo_eliminar_inicio,'.10f')
            print("Tiempo de ejecución de la operación: ",tiempo_total_eliminar)
            print("Operación realizada!\n")
```

```python
mainArrayList.py > {} random
                print("Tiempo de ejecución de la operación: ",tiempo_total_eliminar)
                print("Operación realizada!\n")

            if opcion == 3:
                print("3. Tamaño de la lista.")
                print("El tamaño de la lista es de:")
                tiempo_tamaño_inicio=time.perf_counter()
                print(lista.tamaño_lista(),"elementos.")
                tiempo_tamaño_final=time.perf_counter()
                tiempo_tamaño_total=format(tiempo_tamaño_final-tiempo_tamaño_inicio,'.10f')
                print("Tiempo de ejecución de la operación: ",tiempo_tamaño_total)
                print("Operación realizada!\n")

            if opcion == 4:
                print("4. Buscar elemento por indice.")
                elemento_int=int(input("Ingresa el indice, para retornar el número correspondiente:"))
                tiempo_buscar_inicio=time.perf_counter()
                print("Para el indice ingresado corresponde el elemento de la lista es:", lista.buscar_por_indice(elemento_int))
                tiempo_buscar_final=time.perf_counter()
                tiempo_buscar_total=format(tiempo_buscar_final-tiempo_buscar_inicio,'.10f')
                print("Tiempo de ejecución de la operación: ",tiempo_buscar_total)
                print("Operación realizada!\n")

            if opcion ==5:
                print("5. Imprimir la lista.")
                print("Lista:")
                tiempo_imprimir_inicio=time.perf_counter()
                print(lista.lista_completa())
                tiempo_imprimir_final=time.perf_counter()
                tiempo_imprimir_total=format(tiempo_imprimir_final-tiempo_imprimir_inicio,'.10f')
                print("Tiempo de ejecución de la operación: ",tiempo_imprimir_total)
                print("Operación realizada!\n")

            if opcion == 6:
                salir()

    if __name__ == '__main__':
        main();
```

La cual interactúa de la siguiente manera:

```
/mainArrayList.py"
ArrayList:

Selecciona una opción:

1. Agregar elemento al inicio de la lista.
2. Eliminar un elemento al final de la lista.
3. Tamaño de la lista.
4. Buscar elemento por indice.
5. Imprimir la lista.
Opción: 1
1. Agregar elemento al inicio de la lista.
Ingresar elemento para añadir al inicio de la lista:30
Tiempo de ejecución de la operación:  0.0000142990
Operación realizada!

ArrayList:

Selecciona una opción:

1. Agregar elemento al inicio de la lista.
2. Eliminar un elemento al final de la lista.
3. Tamaño de la lista.
4. Buscar elemento por indice.
5. Imprimir la lista.
Opción: 1
1. Agregar elemento al inicio de la lista.
Ingresar elemento para añadir al inicio de la lista:40
Tiempo de ejecución de la operación:  0.0000118830
Operación realizada!

ArrayList:

Selecciona una opción:

1. Agregar elemento al inicio de la lista.
2. Eliminar un elemento al final de la lista.
3. Tamaño de la lista.
4. Buscar elemento por indice.
5. Imprimir la lista.
Opción: 3
3. Tamaño de la lista.
El tamaño de la lista es de:
2 elementos.
Tiempo de ejecución de la operación:  0.0000409620
Operación realizada!
```

```
ArrayList:

Selecciona una opción:

1. Agregar elemento al inicio de la lista.
2. Eliminar un elemento al final de la lista.
3. Tamaño de la lista.
4. Buscar elemento por indice.
5. Imprimir la lista.
Opción: 5
5. Imprimir la lista.
Lista:
array('i', [40, 30])
Tiempo de ejecución de la operación:  0.0000576150
Operación realizada!

ArrayList:

Selecciona una opción:

1. Agregar elemento al inicio de la lista.
2. Eliminar un elemento al final de la lista.
3. Tamaño de la lista.
4. Buscar elemento por indice.
5. Imprimir la lista.
Opción:
```

3. Desarrollo: Para usar la clase LinkedList, de igual manera, cree un archivo python llamado mainLinkedList.py, donde este archivo llama a las clases y crea una interacción con el cliente para usar las funciones de la clase. Doy como opción las funciones que contiene la clase LinkedList y poder interactuar a través de la terminal con esta. También imprime en pantalla los tiempo de ejecución.
Capturas:

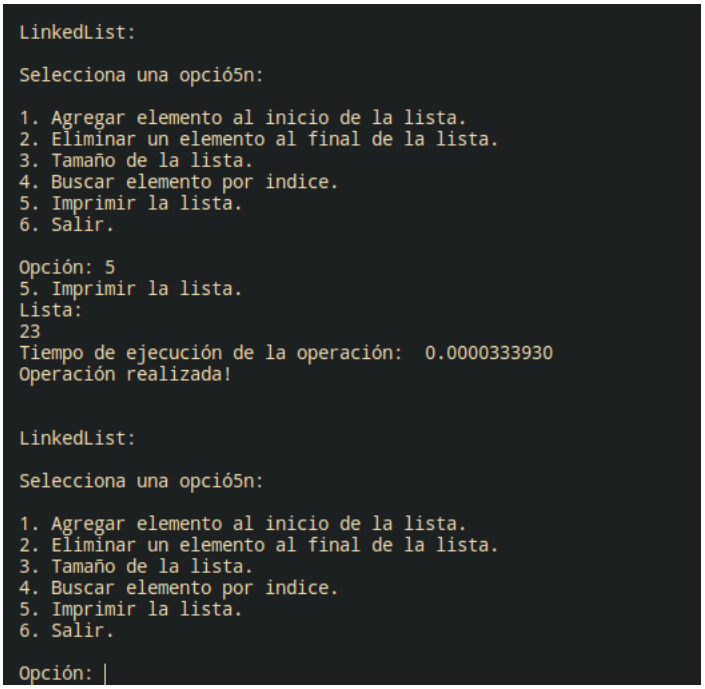
mainLinkedList.py > main

```python
import time
from LinkedList import LinkedList, Nodo

lista=LinkedList()

def salir():
    print("Salida...!")
    exit()

def main():
    while True:
        print("\nLinkedList:")
        print(" ")
        print("Selecciona una opció5n:\n")
        print("1. Agregar elemento al inicio de la lista.")
        print("2. Eliminar un elemento al final de la lista.")
        print("3. Tamaño de la lista.")
        print("4. Buscar elemento por indice.")
        print("5. Imprimir la lista.")
        print("6. Salir.\n")


        opcion=int(input("Opción: "))
        if opcion ==1:
            print("1. Agregar elemento al inicio de la lista.")
            elemento_int=int(input("Ingresar elemento para añadir al inicio de la lista:"))
            tiempo_agregar_inicio=time.perf_counter()
            lista.agregar_elemento_inicio(elemento_int)
            tiempo_agregar_final=time.perf_counter()
            tiempo_total_agregar=format(tiempo_agregar_final-tiempo_agregar_inicio,'.10f')
            print("Tiempo de ejecución de la operación: ",tiempo_total_agregar)
            print("Operación realizada!\n")

        if opcion ==2:
            print("2. Eliminar un elemento al final de la lista.")
            tiempo_eliminar_inicio=time.perf_counter()
            lista.eliminar_elemento_final()
            tiempo_eliminar_final=time.perf_counter()
            tiempo_total_eliminar=format(tiempo_eliminar_final-tiempo_eliminar_inicio,'.10f')
            print("Tiempo de ejecución de la operación: ",tiempo_total_eliminar)
            print("Operación realizada!\n")

```

```python
mainLinkedList.py > ...
                tiempo_eliminar_final=time.perf_counter()
                tiempo_total_eliminar=format(tiempo_eliminar_final-tiempo_eliminar_inicio,'.10f')
                print("Tiempo de ejecución de la operación: ",tiempo_total_eliminar)
                print("Operación realizada!\n")

            if opcion == 3:
                print("3. Tamaño de la lista.")
                print("El tamaño de la lista es de:")
                tiempo_tamaño_inicio=time.perf_counter()
                print(lista.tamaño_lista(),"elementos.")
                tiempo_tamaño_final=time.perf_counter()
                tiempo_tamaño_total=format(tiempo_tamaño_final-tiempo_tamaño_inicio,'.10f')
                print("Tiempo de ejecución de la operación: ",tiempo_tamaño_total)
                print("Operación realizada!\n")

            if opcion == 4:
                print("4. Buscar elemento por indice.")
                elemento_int=int(input("Ingresa el indice, para retornar el número correspondiente:"))
                print("Para el indice ingresado corresponde el elemento de la lista es:")
                tiempo_buscar_inicio=time.perf_counter()
                print(lista.buscar_por_indice(elemento_int))
                tiempo_buscar_final=time.perf_counter()
                tiempo_buscar_total=format(tiempo_buscar_final-tiempo_buscar_inicio,'.10f')
                print("Tiempo de ejecución de la operación: ",tiempo_buscar_total)
                print("Operación realizada!\n")

            if opcion ==5:
                print("5. Imprimir la lista.")
                print("Lista:")
                tiempo_imprimir_inicio=time.perf_counter()
                lista.imprimir_lista()
                tiempo_imprimir_final=time.perf_counter()
                tiempo_imprimir_total=format(tiempo_imprimir_final-tiempo_imprimir_inicio,'.10f')
                print("Tiempo de ejecución de la operación: ",tiempo_imprimir_total)
                print("Operación realizada!\n")

            if opcion == 6:
                salir()

    if __name__ == '__main__':
        main();
```

La cual interactúa de la siguiente manera:

```
LinkedList:

Selecciona una opció5n:

1. Agregar elemento al inicio de la lista.
2. Eliminar un elemento al final de la lista.
3. Tamaño de la lista.
4. Buscar elemento por indice.
5. Imprimir la lista.
6. Salir.

Opción: 1
1. Agregar elemento al inicio de la lista.
Ingresar elemento para añadir al inicio de la lista:23
Tiempo de ejecución de la operación:  0.0000243030
Operación realizada!


LinkedList:

Selecciona una opció5n:

1. Agregar elemento al inicio de la lista.
2. Eliminar un elemento al final de la lista.
3. Tamaño de la lista.
4. Buscar elemento por indice.
5. Imprimir la lista.
6. Salir.

Opción: 3
3. Tamaño de la lista.
El tamaño de la lista es de:
1 elementos.
Tiempo de ejecución de la operación:  0.0000331190
Operación realizada!


LinkedList:

Selecciona una opció5n:

1. Agregar elemento al inicio de la lista.
2. Eliminar un elemento al final de la lista.
3. Tamaño de la lista.
4. Buscar elemento por indice.
5. Imprimir la lista.
6. Salir.

Opción: 5
5. Imprimir la lista.
Lista:
23
Tiempo de ejecución de la operación:  0.0000333930
Operación realizada!
```



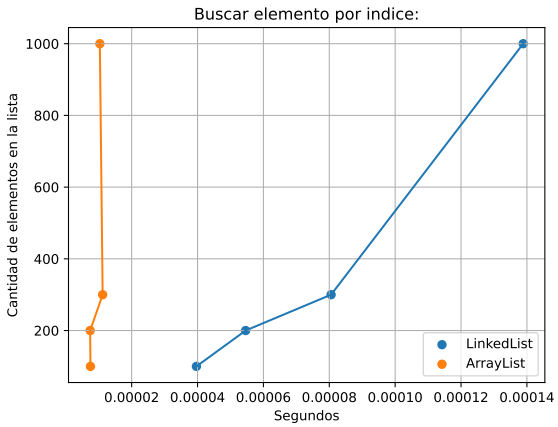
4. Desarrollo: Para buscar un elemento dado un índice tenemos:
Para el **Linked List** tenemos que: Para una lista con 100 elementos, la operación de buscar el numero correspondiente al índice 80 es de 0.0000396640 segundos. De la misma manera, para 200 datos en la lista, encontrar el numero del índice 160 tomo 0.0000546300 segundos y para 300 elementos, tomo 0.0000805510 segundos encontrar el elemento del índice 240. Para 1000 elementos en la lista, tomo 0.0001388760 segundos encontrar el numero correspondiente al índice 600.
Tenemos entonces para LinkedList:

Elementos	Segundos:
100	0.000039664
200	0.00005463
300	0.000080551
1000	0.000138876

De forma análoga para ArrayList tenemos:

Elementos	Segundos:
100	0.000007491
200	0.000007421
300	0.000011191
1000	0.000010348

Podemos representar esta información en un gráfico de la forma:



Por otro lado, la tabla de tiempos que toma agregar un elemento al inicio de la lista, dado una cantidad de elementos ya insertas en la lista es:
Para LinkedList:

Elementos	Segundos:
250	0.000002684
500	0.000002115
750	0.000003695
1000	0.000002246

Para ArrayList:

Elementos	Segundos:
250	0.00000115
500	0.000001278
750	0.00000204
1000	0.000003842

También tenemos la tabla de datos, de cuanto tiempo demora eliminar el ultimo elemento de la lista, dado una cantidad de elementos ya insertas en la lista. Tenemos: Para LinkedList:

Elementos	Segundos:
250	0.000021285
500	0.000053875
750	0.000069692
1000	0.00009062

Para ArrayList:

Elementos	Segundos:
250	0.000005346
500	0.000004107
750	0.000007652
1000	0.00000466

4. Desarrollo: Ambos métodos tienen sus ventajas y desventajas.
Las listas linkeadas nos ofrecen un tamaño dinámico, características que nos ayudan a ahorrar memoria cuando halamos de una gran cantidad de datos, y no agotar toda la memoria muy rápido, también facilita la inserción de elementos. Sus desventajas pueden ser que no nos permite acceder a elementos aleatorios, sino que esta debe recorrer de forma secuencial la lista, por lo tanto se restringe su uso para realizar, por ejemplo, busquedas binarias.
Según los datos obtenidos tenemos que Arraylist es mejor para almacenar y acceder a datos y por otro lado, linkedlist es mucho mejor para manipular los datos de la lista.
Esto lo podemos evidenciar ya que insertar o eliminar un elemento para Arrays es $O(n)$ mientras que para LinkedList es $O(1)$. Tenemos que para acceder a un elemento, para Arrays tenemos que es $O(1)$, y para linkedList es $O(n)$. Por lo tanto tenemos que cada una de las ADT List tiene sus ventajas, una sera mas útil y mejor dado el contexto en donde esta se apliquen.