Enunciado del Proyecto 3

Laboratorio de Algoritmos 2 CI-2692, Universidad Simón Bolívar 28/10/2014

Entrega

Martes 25 de Noviembre del 2014 (Martes de la Semana 12) antes de las 11:59 pm en el espacio de su grupo en Aula Virtual.

1. Descripción General

Para este proyecto, se debe entregar un programa que realice ciertas operaciones sobre un árbol genealógico, entre ellas tendremos crear árbol, modificar árbol, y ordenar iniciales. Los detalles serán explicados en las siguientes secciones.

Siguiendo los principios de Encapsulamiento y Modularidad se debe implementar el proyecto.

Su programa se ejecutará mediante el comando especificado a continuación. Su módulo principal debe tener el mismo nombre.

```
1 python3 familyTree.py <entrada1> <entrada2> <salida>
```

Siendo <Entrada1>y <Entrada2>los archivos que contienen los conocimientos que debe utilizar el programa, y salida es el archivo en donde se escribirá la salida.

2. Modelo Abstracto

Siendo un árbol genealógico una estructura que representa a los antepasados y descendientes de un grupo de individuos, a continuación se muestra una especificación incompleta del TAD ArbolGenealógico. No es necesario completar el modelo abstracto ni el concreto para la entrega. Solo se debe utilizar como guía en la elaboración del proyecto.

2.1. Especificación de TAD ArbolGenealogico

```
Modelo de Representacion
  name # nombre del individuo.
  firstChild # Primer hijo
  secondChild # Segundo hijo
  posicionInicial
  # Representa la posicion que tuvo al momento
  # de definirse en el archivo de entrada.

  ...
```

3. Formato de los archivos de entrada

3.1. Archivo de entrada 1

Este archivo es el primero que debe procesarse, y contiene los conocimientos iniciales sobre quién es padre de quién. La idea es que con la lectura de este archivo, el programa pueda crear la estructura árbol definida anteriormente.

El formato del archivo es simplemente una serie de líneas en donde cada una contiene parte del conocimiento de una generación. Cada línea será un conjunto de nombres separados por uno o mas espacios, en donde el primero es el padre y los siguientes son los hijos. Por ejemplo:

```
Shubaric Pawel
Pawel Zaklina Jan
Shubaric Sergan
```

En el ejemplo anterior, Pawel es padre de Zaklina y Jan. Shubaric es el padre de Pawel y Sergan. En cuanto a la *posicionInicial*, en este caso, Shubaric tiene el 0, Pawel el 1, Zaklina el 2, Jan 3, etc. Es decir, se le asigna un número ascendente a cada nuevo nombre definido en el archivo de entrada.

Hay garantía de que cada padre tendrá a los sumo 2 hijos.

3.2. Archivo de entrada 2

Como es usual en el mundo real, los datos pueden tener errores. Este archivo contiene las posibles correcciones de los datos dados en el archivo entrada 1. *Estas correcciones deben hacerse LUEGO de haber creado la estructura árbol mencionada*. El archivo indicará quien era el verdadero

padre de un nombre dado, siendo el primero el nombre del padre, y el siguiente el nombre del hijo. Acá tendrá unicamente dos nombres por línea. Siguiendo el ejemplo anterior:

```
Jan Sergan
Pawel Zaklina
```

Acá indica que Pawel era el verdadero padre de Sergan, y Pawel el padre de Zaklina. Como puede verse, la relación padre Pawel-Zaklina estuvo correcta originalmente, por lo tanto, no debe hacerse alguna modificación en el árbol en estos casos.

4. Ordenamiento

Una vez que tengamos el árbol con las modificaciones solicitadas en el archivo entrada2, el programa deberá ser capaz de crear una lista en pre-orden, in-orden o post-orden, y LUEGO ordenar esta lista alfabéticamente. Haga implementaciones con los siguientes algoritmos de ordenamiento: quicksort, mergesort, bubblesort, insertionsort y heapsort.

5. Comparación de velocidad de los algoritmos de ordenamiento

Utilizando la función *time* de Python, calcule el tiempo que demoran los distintos algoritmos para ordenar los elementos en el árbol. Haga la comparación para árboles de tamaño 100, 500, 1000 y 10000. Luego haga un informe de máximo 2 páginas en donde se haga un análisis breve de la razón de sus velocidades. Los casos de pruebas deben ser generados por los estudiantes.

6. Formato de Salida

Una vez realizado el paso de ordenamiento, se debe imprimir en el archivo de salida cómo quedó finalmente el árbol genealógico. Debemos imprimir desde la raíz y de izquierda a derecha, todos los hijos de un determinado padre en una línea, teniendo al nombre del padre como primera palabra seguido de dos puntos. Luego los nombres de todos los hijos separados por un espacio simple. Cada nombre que se imprime debe estar acompañado por su correspondiente número asignado entre paréntesis. Continuando con el ejemplo anterior, la respuesta sería:

```
Shubaric (0): Pawel (1) Sergan (4)
Pawel (1): Zaklina (2) Jan (3)
```

A continuación se coloca un pseudocódigo:

```
1 def imprimir(self):
2   # imprimir mi nombre, luego dos puntos
```

```
3   # imprimir el nombre de todos mis hijos.
4   # para cada hijo:
5     hijo.imprimir()
```

Luego de esto, se debe dejar una linea en blanco, e imprimir en cada linea todos los nombres del árbol en pre-orden, in-orden y post-orden (Cada nombre separado por un espacio).

7. Requerimientos de Entrega

1. Se evaluará positivamente la limpieza y buena estructura del código. En particular procure separar las operaciones del TAD de la lógica de entrada-salida de su programa.
2. El código debe estar adecuadamente documentado, lo cual incluye nombre de variables y métodos auto-explicativos en la medida de lo posible. Recuerde que el propósito de la documentación no es repetir información ya expresa en el código.
3. La entrega del proyecto será directamente en el espacio de su grupo en el Aula Virtual. Deposite un archivo comprimido que contenga los archivos de código fuente, así como otros que sean necesarios para la ejecución de la aplicación. El nombre del archivo debe ser `Proy3-XX.tar.gz`, donde XX es el número de su grupo.
4. Todos sus archivos de código deben tener un encabezado con los nombres y números de carnet de ambos miembros del equipo.
5. La entrega del proyecto debe realizarse ANTES de la fecha y hora indicadas al inicio del enunciado. Se recomienda realizar la entrega con antelación y considerar imprevistos como fallas de conectividad.

Se proveerá una muestra de archivos de entrada y la salida esperada junto con este enunciado. Igualmente se adjuntará código con un ejemplo básico de lectura y escritura de archivos.

El no cumplimento de los requerimientos podrá resultar en el rechazo de su entrega.

8. Preguntas y consultas

Se recomienda hacer preguntas con la mayor antelación posible. A fin de realizar las consultas de manera pública y persistente, coloque sus preguntas en el foro del Aula Virtual.