ICESI
2022-01

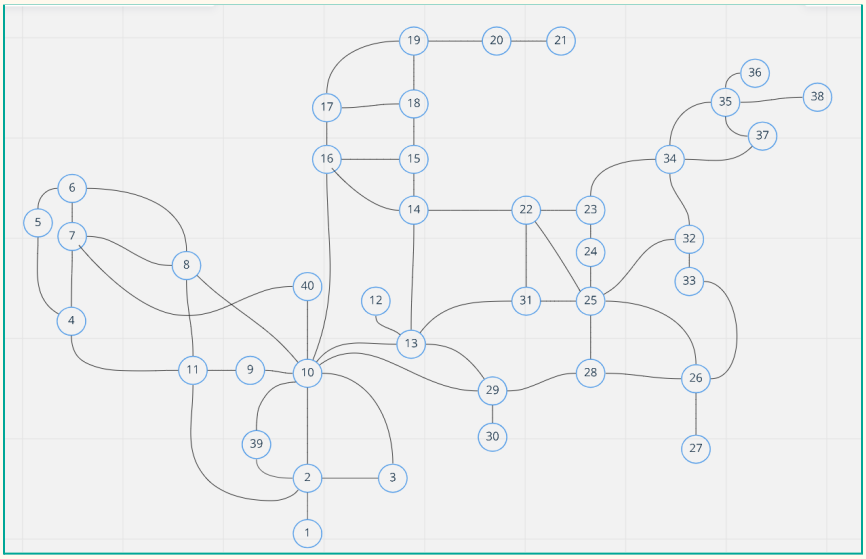
Profesor: Anibal Sosa
Participantes: Valderruten Soler Fabiana & Valencia Valencia Andy & Pino Andrés

Tarea integradora III

Computación y estructuras discretas |

Enunciado

- La universidad se prepara para recibir a los nuevos estudiantes de posgrado de la facultad de Ingeniería y diseño. Para esto se planea hacer una serie de actividades que le brinden una buena bienvenida a los nuevos. Se debe tener en cuenta que algunos de estos estudiantes son egresados de la universidad y otros la conocen por primera vez. Como toda inducción se hará un recorrido por el campus, mostrando las áreas más usadas, los salones y los espacios que ofrece la universidad para el desarrollo y comodidad de sus estudiantes. La universidad requiere de una idea creativa para que los nuevos puedan tener la mejor experiencia en su inducción.
- Los directores de la facultad planean aprovechar una de las competencias que desarrolla la universidad en sus estudiantes que diseñan y desarrollan software, por esto se desea crear un programa que le indique al nuevo estudiante de inducción cuál podría ser la ruta más corta para llegar a algún punto más recurrente de la universidad. Para así facilitar el recorrido por el campus y una mejor logística para el flujo de los estudiantes.



Método de ingeniería

Los ingenieros tienden a tratar los problemas de manera única, éste método de ingeniería difiere de muchos otros métodos usados por el resto de profesionales :

1. Identificación del problema.
2. Recopilación de la información necesaria.
3. Búsqueda de soluciones creativas.
4. Paso de las ideas a los diseños preliminares.
5. Evaluación y selección de la solución.
6. Preparación de informes y especificaciones.
7. Implementación del diseño.

Contexto problemático : La universidad ICESI desea darles una agradable inducción a los estudiantes de Posgrado de la facultad de ingeniería y diseño. Para esto, se tiene cómo idea principal, hacer un recorrido por la universidad mientras se integran a los estudiantes, de una forma diferente a la comúnmente utilizada en inducciones pasadas. Este se quiere hacer por puntos de llegada distribuidos a lo largo del campus, a los cuales, a los estudiantes se les brindarán una serie de pistas para llegar al punto exacto correspondiente, donde se encontrarán con otro grupo de estudiantes, y partirán hacía otro punto de encuentro en el cual habrá otro grupo de estudiantes esperándolos, así hasta llegar al punto final, el cuál es la tarima de Bienestar Universitario.

Desarrollo de la solución: Para dar una solución al problema el equipo de trabajo usa el método de la ingeniería siguiendo un enfoque sistemático y acorde con la situación de la problemática planteada.

Necesidades:

- ☐ Aún no existen las rutas más rápidas, que permitan el buen flujo de personas por el campus para realizar los recorridos
- ☐ La solución debe mostrar los datos de las ubicaciones iniciales y finales de cada usuario.
- ☐ El programa debe ser eficiente para cada ruta planteada.
- ☐ Distribuir correctamente a cada pareja de estudiantes, tal que no cuenten con la misma ruta que otra pareja.

1. ***Identificación del problema*** : Icesi requiere de un módulo de software que permita mostrar el comportamiento o traslaciones que hagan los usuarios al descubrir o pasar cada punto, hasta llegar al punto de encuentro final.

2. ***Recopilación de la información necesaria :*** Con el fin de tener más claridad en los conceptos involucrados se hace una búsqueda de información y experiencias del comportamiento de un recorrido habitual por el campus. Para realizar una solución más óptima y que cumpla con las

competencias del curso, se ha estudiado y trabajado el uso de grafos, listas de adyacencia, algoritmo de búsqueda BFS, algoritmo de Dijkstra. Incluso es importante crear casos de prueba que simulan las entradas comunes del programa y así poder comprobar el buen comportamiento del software en solución.

Graph: Es una composición de un conjunto de nodos, que se relacionan entre sí a través de conexiones conocidas como aristas. Estos permiten analizar las relaciones existentes que interactúan con otras.

Lista de adyacencia: Es una forma de implementar grafos computacionalmente, ya que contiene un arreglo del tamaño igual de la cantidad de los vértices del grafo, donde cada posición del arreglo corresponde un vértice, dónde cada uno tiene un apuntador a listas donde están los vértices que son adyacentes a ese vértice.

Breadth-First Search (BFS): Este algoritmo de búsqueda conocido como: Búsqueda en anchura, consiste en recorrer los nodos de un grafo, comenzando por una raíz (se elige un nodo como raíz), para explorar todos los vecinos de este nodo, luego, para cada vecino se explora su respectivo vecino adyacente, así hasta terminar de recorrer todo el grafo.

Queue: Es una estructura de datos, caracterizada por ser una secuencia de elementos en la que la operación de inserción push se realiza por un extremo y la operación de extracción pull por el otro. También se llama estructura FIFO, debido a que el primer elemento en entrar, también será el primero en salir.

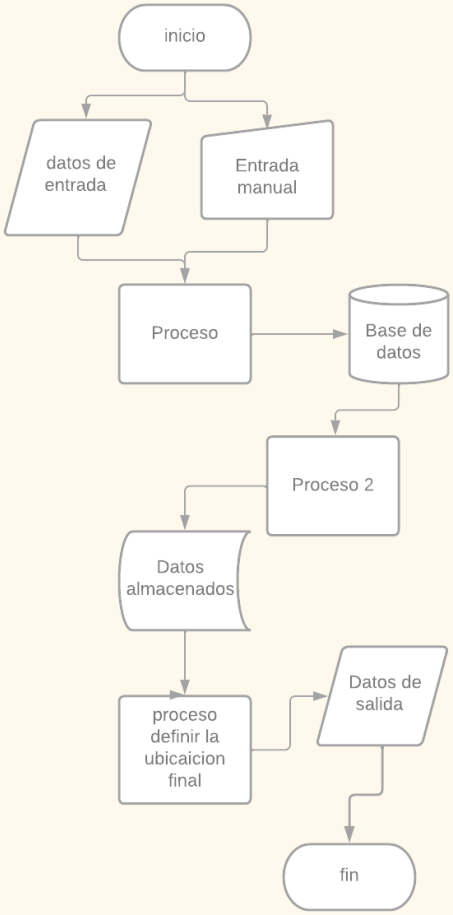
Dijkstra: Es un algoritmo para determinación del camino más corto, dado un vértice origen al resto de vértices en un grafo con pesos en cada arista.

3. ***Búsqueda de soluciones creativas :*** El equipo de trabajo ha ideado un programa que pida como entrada la cantidad de parejas en el momento de la inducción, en la segunda línea la cantidad de rutas antes de llegar a cada punto de encuentro, por ejemplo: Hay 8 parejas de estudiantes, para llegar a cada punto de encuentro, existen rutas, hay que destacar que todas las rutas parten desde el edificio E, la primera pareja sigue a Árbol Samán, va un punto, sigue a mesas de biblioteca, van dos puntos, luego sigue a cafetería central, que es el punto de encuentro, ahora están esperando la segunda pareja. La segunda pareja parte del E, sigue a Edificio de BU, luego a Cafetería Wonka, y finalmente, cafetería central para encontrarse con la otra pareja. Luego siguen a otros tres puntos para encontrarse con otro cuarteto, luego se forma un grupo de 8 personas, y siguen a encontrarse con otro grupo de 8, formando un grupo de 16. Una de las ideas que ha tenido el equipo de trabajo, ha sido usar los recursos de java como Queues, tablas hash y Stacks para almacenar los datos de las rutas (pistas, dirección, distancia en metros). Finalmente se planea que el programa arroje un listado de los datos en los que se ha movilizado cada pareja en cada ruta a través del campus de la Universidad. Además se planea hacer y entregar el test para cada estructura, así se comprueba el buen funcionamiento.

4. ***Transformación de las ideas a diseños preliminares :*** Lo primero que se hace es descartar las ideas que no son factibles. En este sentido no se descarta ninguna, ya que el equipo de trabajo es pequeño y cada idea ha sido bien planteada desde el inicio, hablando del comportamiento del programa. Si se tratase del diseño, podrían existir algunas modificaciones, entre la creación de menús para el ingreso de datos o en la presentación de los outputs.

5. ***Evaluación y selección de la solución:*** A medida que evoluciona el proceso de ingeniería y el equipo ha trabajado en la planeación y pruebas de las ideas del software. Se ha llegado a la conclusión que se eligieron las ideas ya planteadas.

6. ***Preparación de informes y especificaciones:***
   Diagrama de flujo basándose en el diseño.



7. ***Implementación del diseño:*** Implementación en un lenguaje de programación. lista de tareas para implementar:
    1. Validar los datos ingresados
    2. Crear las rutas
    3. Crear los objetos
    4. Crear las estructuras de datos: Stacks, queues.
    5. Identificar el recorrido del usuario (el punto de llegada)
    6. Mostrar dónde se encuentran los usuarios al finalizar el programa.

Código para crear una estructura, colas:

```
package model.data_structures;
import model.interfaces.Intercolas;
import java.util.NoSuchElementException;
public class Colas<T> implements Intercolas<T> {
   private Nodos<T> front;
   private Nodos<T> rear;
   private int size;

      public Colas() {
      front = null;
      rear = null;
      size = 0;
   }
   public void enqueue(T data) {
      Nodos<T> element = new Nodos<>(data);
      if (front == null) {
         rear = element;
         front = element;
      } else {
         rear.setNext(element);
         rear = element;
      }
      size++;
   }
   public T dequeue() {
      Nodos<T> trash = front;
      if (front == null) {
         throw new NoSuchElementException("Can't dequeue
from an empty queue");
      } else if (front == rear) {
         front = rear = null;
      } else {
         front = front.next();
      }
      size--;
      return trash.data();
   }
   public T front() {
      return front.data();
   }

    public boolean isEmpty() {
      return size == 0;
   }

     public int size() {
      return size;
   }


   public String toString() {
      StringBuilder sb = new StringBuilder();
      Nodos<T> head = front;
      sb.append("(");
      String prefix = "";
      while (head != null) {
         sb.append(prefix).append(head.data());
         prefix = ", ";
         head = head.next();
      }
      sb.append(")");
      return sb.toString();
   }
   public Colas<T> reverse() {
      Stack<T> aux = new Stack<>();
      Colas<T> reversed = new Colas<>();
      while (!isEmpty()) {
         aux.push(dequeue());
      }
      while (!aux.isEmpty()) {
         reversed.enqueue(aux.pop());
```

```
        }
        return reversed;
    }

    public void toQueue(T[] array) {
        front = null;
        rear = null;
        for (T var : array) {
            enqueue(var);
        }
    }
}
```

Creación de los objetos, Parejas :

```
package model.objects;

public class Parejas {
    private String nombrePareja;
    private int posicion;
    private double Ruta;

    public Parejas(String nombrePareja, int posicion, double ruta) {
        this.nombrePareja = nombrePareja;
        this.posicion = posicion;
        this.ruta  = ruta;
    }

    public String getNombrePareja() {
        return nombre;
    }

    public int getPosicion() {
        return posicion;
    }

    public int getRuta() {
        return ruta;
    }
}
```

Creación de grafo, mediante matriz de adyacencia:

```
package model.graph;
class Grafo {
    private int[][] Adyacentes;
    private Object!] Informacion;
    private int nodos:
    private boolean vacio = true;
    public Grafo(int numeroNodos) {
                Adyacentes = new int[numeroNodos][numeroNodos];
                Informacion = new Object[numeroNodos]:
                for (int i=0; i‹numeroNodos; i++)
                for (int j=0; j<numeroNodos; j++)
                Adyacentes[i]0]=0;
                nodos= numeroNodos:
                }
```

```
public boolean EsVacio() { return vacio; }}
```