|  |  |  |
| --- | --- | --- |
| **CARRERA:**  Ingeniería de Software | **GUÍA**  No. 03 | **TIEMPO ESTIMADO:**  8 horas. |
| **ASIGNATURA:**  Estructura de Datos  NRC: 3251 | **FECHA DE ELABORACIÓN:** 31 de marzo del 2021 **SEMESTRE**: Nov 2020 – Abr 2021 | |
| **TÍTULO:**  Proyecto Tercer Parcial | **DOCENTE:**  Ing. Fernando Solis | |

|  |
| --- |
| 1. **INTRODUCCIÓN** |

1. **Grafos**

Un grafo es una estructura de datos compuesta de vértices (nodos) y aristas (conexiones entre nodos), estos representan la relación que existe entre los vértices conectados por aristas.

En otra definición más formal tenemos que, un grafo es una abstracción matemática que se representará por G = (V, A) donde V es un conjunto de puntos y A es un conjunto de líneas que unen dos puntos de V; A puede ser vacío, llamado conjunto de aristas (Álvarez & Parra Muñoz, 2013).

Como conceptos generales, una arista o arco representa una relación entre dos nodos. Esta relación, al estar formada por dos nodos, se representa pro (U, V) siendo U, V el par de nodos (Joyanes Aguilar, Sanchez García, & Zahonero Martínez, 2007).

1. **Representación de un grafo**

La representación de los grafos se la puede hacer mediante distintos elementos, a continuación tenemos algunas de sus representaciones:

* 1. **Matriz de adyacencia**

Es una matriz booleana que representa las conexiones entre pares de vértices. La matriz de adyacencia de un grafo es simétrica. Si un vértice es aislado entonces la correspondiente fila (columna) está compuesta sólo por ceros (García, s.f.).

* 1. **Matriz de incidencia**

La matriz de incidencia sólo contiene ceros y unos (matriz binaria). Como cada arista incide exactamente en dos vértices, cada columna tiene exactamente dos unos (García, s.f.). El número de unos que aparece en cada fila es igual al grado del vértice correspondiente.

* 1. **Lista de adyacencia**

Representa una lista de todos los vértices, cada vértice guarda una lista de adyacencia con un objeto arista para cada vértice alcanzable desde él. Es una lista de listas.

Para grafos dispersos la matriz de adyacencia ocupa el mismo espacio que si el grafo tuviera muchos arcos (grafo denso). Cuando esto ocurre, se elige la representación del grafo mediante listas enlazadas, denominadas listas de adyacencia (Joyanes Aguilar, Sanchez García, & Zahonero Martínez, 2007).

1. **Búsqueda Primero en Amplitud**

Su funcionamiento consiste en ir expandiendo cada uno de los nodos que va localizando, de forma recurrente (desde el nodo padre hacia el nodo hijo). Cuando ya no quedan más nodos que visitar en dicho camino, regresa al nodo predecesor, de modo que repite el mismo proceso con cada uno de los vecinos del nodo (López Mamani, 2020).

1. **Búsqueda Primero en Profundidad**

Una búsqueda en anchura (BFS) es un algoritmo de búsqueda para lo cual recorre los nodos de un grafo, comenzando en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo), para luego explorar todos los vecinos de este nodo, si se encuentra el nodo antes de recorrer todos los nodos, concluye la búsqueda (López Mamani, 2020).

|  |
| --- |
| 1. **OBJETIVO** |

Implementar los recorridos en Amplitud y Profundidad para Grafos en lenguaje c++, a través del uso de la teoría revisada en clases y en fuentes digitales de información, para generar el recorrido de un grafo a partir de uno de sus vértices.

|  |
| --- |
| 1. **DESARROLLO** |

1. **Recursos**
2. Un estudiante por computador.
3. Repositorios de bases digitales, bibliotecas virtuales, y revistas científicas.
4. Material guía que se encuentra compartido en el aula virtual
5. **Desarrollo del taller práctico**
6. Implementar las funciones buscarAmplitud() y buscarProfundidad() para realizar los recorridos respectivos para un grafo dirigido, ingresando el vértice inicial y mostrando su matriz y lista de adyacencia correspondiente.

|  |
| --- |
| 1. **CONCLUSIONES** |

1. Se realizaron las acciones previstas de modo que se representan los recorridos tanto en amplitud como en profundidad en una matriz de adyacencia y su lista de vértices visitados y sus parientes.
2. Se logró un aprendizaje significativo de grafos y sus respectivos recorridos implementados en este proyecto

|  |
| --- |
| 1. **BIBLIOGRAFÍA** |

Álvarez, M. F., & Parra Muñoz, J. A. (2013). *Teoría de Grafos*. Obtenido de Universidad del Bio-Bio: http://repobib.ubiobio.cl/jspui/bitstream/123456789/1953/3/Alvarez\_Nunez\_Marcelino.pdf

García, E. M. (s.f.). *Universidad tecnológica de Mixteca*. Obtenido de Estrutura de datos - Grafos: http://www.utm.mx/~mgarcia/ED4(Grafos).pdf

Guardati, S. (2007). *Estructura de Datos Orientada a Objetos.* México: PEARSON Prentice Hall.

Joyanes Aguilar, L., Sanchez García, L., & Zahonero Martínez, I. (2007). *Estructura de datos en C++.* Madrid: McGRAW-HILL/INTERAMERICANA DE ESPAÑA.

López Mamani, M. (2020). *Avantica*. Obtenido de DFS vs BFS: https://www.avantica.com/es/blog/dfs-vs-bfs

|  |
| --- |
| 1. **ANEXOS** |

1. buscarAmplitud(T, int\*\*, int\* ,int)

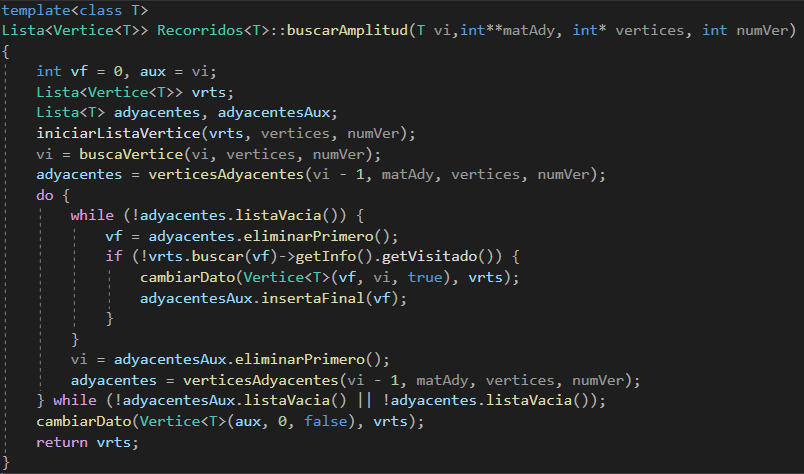


Figura : Función para búsqueda en amplitud

1. buscarProfundidad(int, int\*\*, int\* , int);

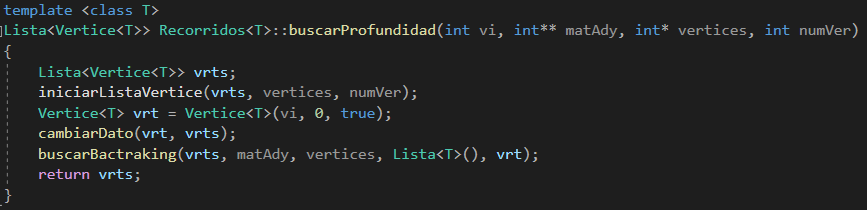


Figura : Función para búsqueda en profundidad

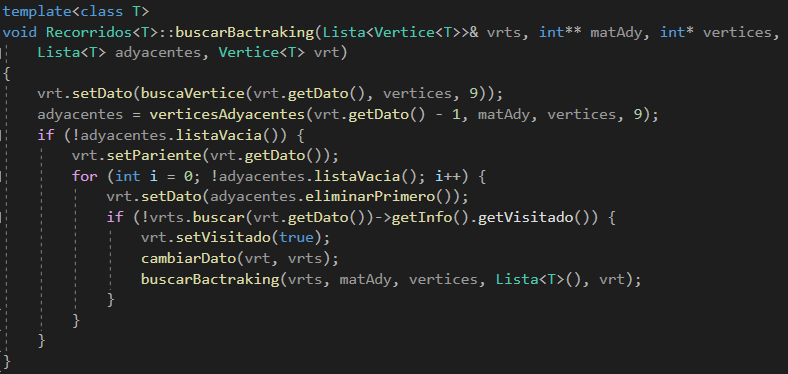


Figura : Función para búsqueda aplicando backtraking

1. Interfaz gráfica

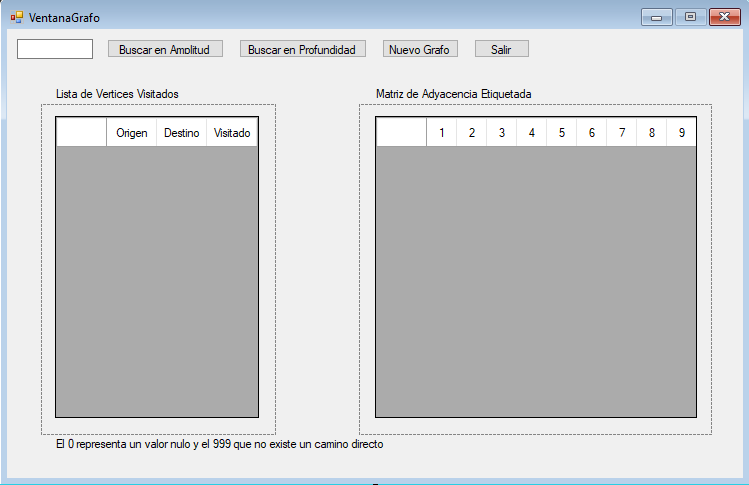


Figura : Interfaz gráfica para observar matriz

|  |
| --- |
| 1. **NOMBRES, APELLIDOS Y FIRMA DE LOS ESTUDIANTES PARTICIPANTES** |

Estudiante(s): Chuquimarca Llulluna Kevin Santiago

Iza Quinatoa Marco Alexander

|  |
| --- |
| 1. **FECHA DE ENTREGA** |

Sangolquí, a 06 de abril de 2021