TPO Programación 3

Integrantes:

Timoteo Güerini	**LU:1106625**
Murad Nazareno	**LU:1100795**

Grupo 24. MIM verano 2021

Turno Noche

Profesor: Rodriguez, Guillermo Horacio

Algoritmos tratados:

- DFS
- BFS
- Dijkstra

Índice

Implementación

Las implementaciones usadas se encuentran en la carpeta implementaciones, las mismas son:

- Cola estática: utilizando un arreglo para realizar la implementación
- Conjunto estático: utilizando un arreglo para realizar la implementación
- Grafo estático: utilizando matriz de adyacencia, se eligió este tipo de implementación ya que es más óptimo en espacio, en caso de que el grafo utilizado sea denso

DFS(depth-first search)

El algoritmo comienza la búsqueda por un nodo actual, toma luego sus adyacentes y a partir de uno de ellos realiza la búsqueda en profundidad. Cuando termina la búsqueda para ese nodo adyacente continúa con el siguiente, mediante un algoritmo recursivo

Elementos que utiliza el algoritmo:

- Nodo actual: es el nodo con el que se trabaja en el momento
- Adyacentes: es un conjunto que indica los nodos que tienen relación directa con el nodo actual
- Visitados: es un conjunto que muestra los nodos que visitó el algoritmo, este nodo devuelve el resultado final.

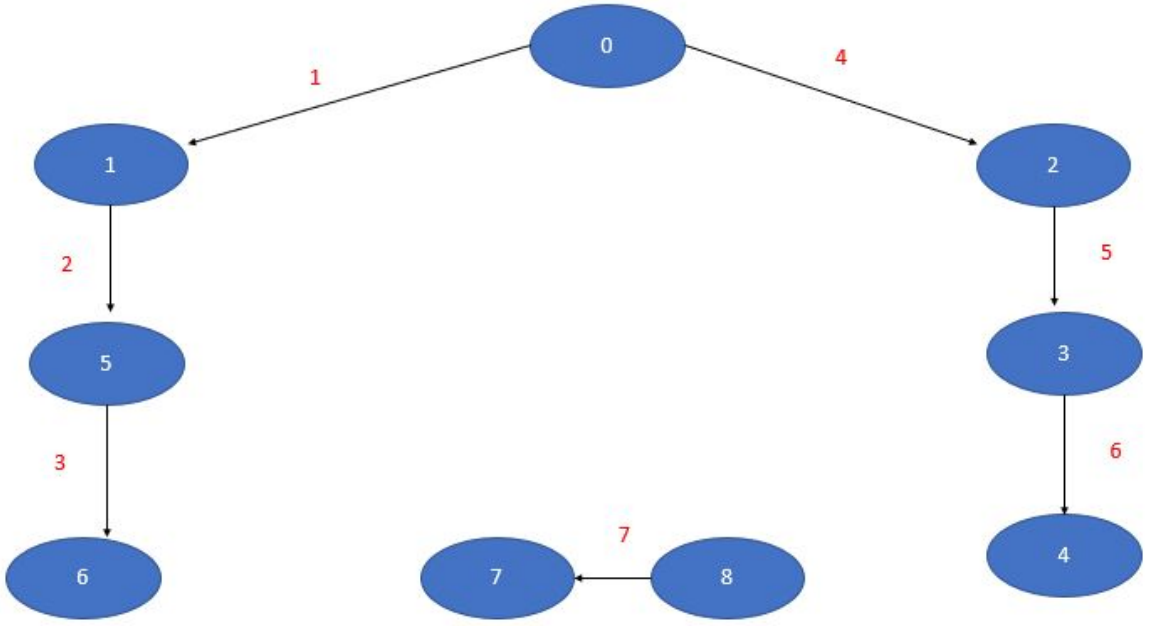
Algoritmo

El algoritmo se encuentra en la carpeta “algoritmos” en el archivo “DFS”. En esa clase se encuentran tres métodos, uno es “DFS”, otro es un método que devuelve los vértices adyacentes de un nodo, y un tercer método para poder realizar la búsqueda de subgrafos aislados, en caso de que el grafo no sea conexo
En el método DFS se recibe el grafo con el que se va a trabajar, como también el nodo con el que se trabaja en el momento, llamado “Nodo actual”, y por último se recibe el conjunto que representa los nodos visitados, en el cual se almacena el resultado final. Este método siempre agrega el nodo actual al conjunto visitados. Luego se le busca sus adyacentes, para agregarlo al conjunto visitados, pero al realizarse de manera recursiva, se hace una búsqueda en profundidad.
Para obtener los nodos adyacentes al nodo actual, se debió crear un método que devuelva los mismos. Este método recibe el grafo, y el nodo actual, el cual retorna un conjunto con los nodos adyacentes al nodo actual.
Al final del método, se encuentra un llamado a un método “forest” el cual busca nodos que no se hayan visitados en el grafo, en caso de que existan los mismos, se realiza una llamada al método DFS, utilizando como nodo actual al encontrado. Esto suele suceder, si el grafo tiene subgrafos aislados de la mayoría de nodos, es decir si el grafo no es conexo

Prueba

En la carpeta “resources” se encuentra un archivo “PruebaDFS” el cual ya posee un caso cargado.

Un caso de prueba es el siguiente



En donde se ve el orden de ejecución en los números marcados en rojo

La salida es la siguiente:

Salida algoritmo DFS: [0,1,5,6,2,3,4]

Salida forest: [8,7]

BFS

El algoritmo comienza la búsqueda por un nodo actual, toma luego sus adyacentes y a partir de uno de ellos realiza la búsqueda en forma horizontal.

Elementos

- Nodo actual: es el nodo con el que se trabaja en el momento
- Adyacentes: es un conjunto que indica los nodos que tienen relación directa con el nodo actual
- Visitados: es un conjunto que muestra los nodos que visitó el algoritmo, este nodo devuelve el resultado final.

- Cola: posee los elementos adyacentes al nodo actual, este mismo sirve para saber cuál va a ser el nuevo nodo actual en la próxima iteración

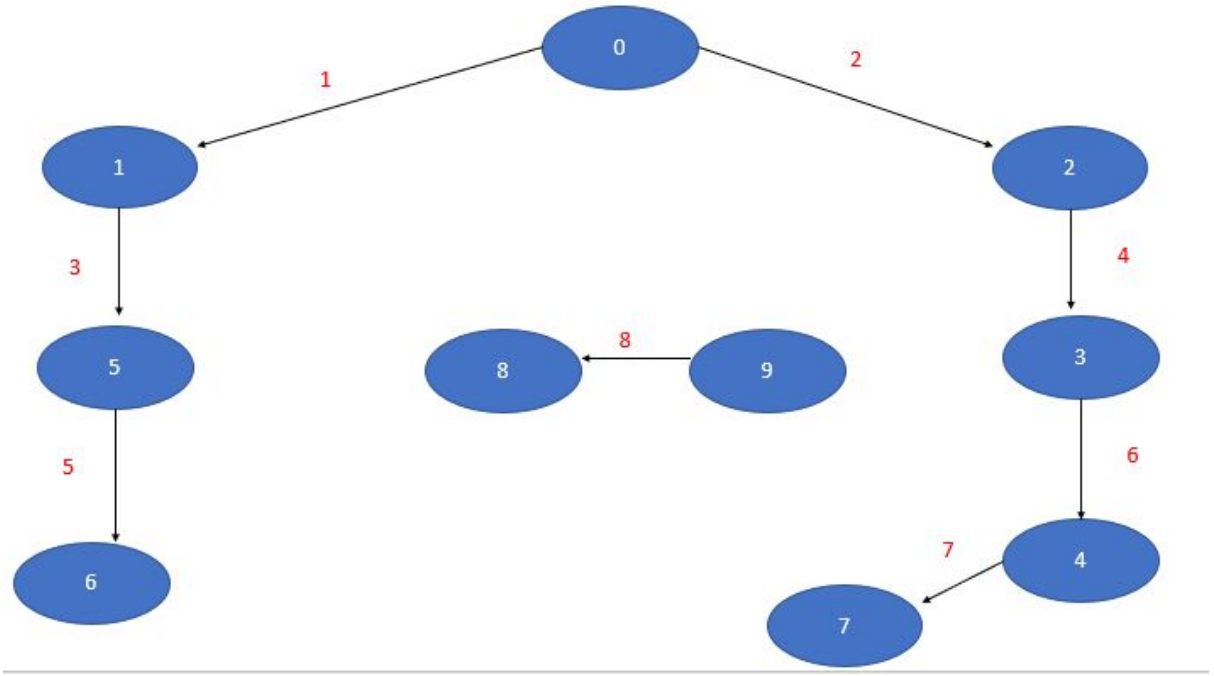
Algoritmo

El algoritmo se encuentra en la carpeta algoritmos, en el archivo “BFS”. En este mismo archivo hay 3 métodos. En esa clase se encuentran tres métodos, uno es “BFS”, otro es un método que devuelve los vértices adyacentes de un nodo, y un tercer método para poder realizar la búsqueda de subgrafos aislados, en caso de que el grafo no sea conexo
Para obtener los nodos adyacentes al nodo actual, se debió crear un método que devuelva los mismos. Este método recibe el grafo, y el nodo actual, el cual retorna un conjunto con los nodos adyacentes al nodo actual.
El método BFS recibe un grafo, el nodo actual , el conjunto visitados, y una cola vacía. En primer lugar se agrega a la cola y al conjunto visitados el nodo actual, luego se ingresa a un while, en donde se itera siempre y cuando la cola no esté vacía. Dentro de ese while se selecciona el primer valor, y se lo des-acola. Una vez encontrado ese valor, se le buscan sus nodos adyacentes, para poderlos meter en el conjunto visitados, y en la cola.
Al final del método, se encuentra un llamado a un método “forest” el cual busca nodos que no se hayan visitados en el grafo, en caso de que existan los mismos, se realiza una llamada al método BFS, utilizando como nodo actual al encontrado. Esto suele suceder, si el grafo tiene subgrafos aislados de la mayoría de nodos, es decir si el grafo no es conexo

Prueba

En la carpeta “resources” se encuentra un archivo “PruebaBFS” el cual ya posee un caso cargado.

Caso de prueba



En los números en rojo se puede ver el orden de ejecución del algoritmo BFS

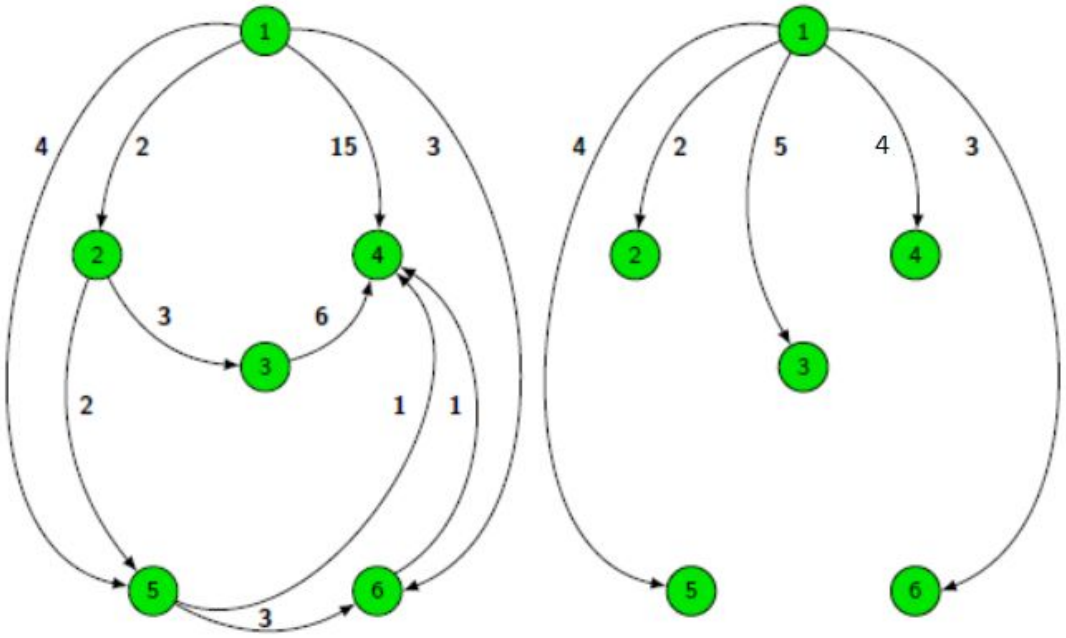
La salida es:

Salida algoritmo BFS: [0,1,2,5,3,6,4,7]

Salida forest: [9,8]

Dijkstra

Este algoritmo se basa en encontrar los caminos conformados por las aristas de menor peso (o costo) desde un **vértice x** dado hacia todos los demás vértices del grafo. El resultado del proceso es un grafo con los mismos vértices pero que sólo tiene aristas desde el **vértice x** hacia los demás con el costo encontrado.



Elementos que utiliza el algoritmo:

- Grafo: es el grafo a utilizar en el método y se recibe como parámetro
- Vertice: es el vértice recibido como parametro parametro
- VerticesGrafo: es un conjunto con los vértices del grafo
- Pendientes: son los vértices que quedan por visitar en
- Visitados: es un conjunto que guarda los vértices ya visitados en la iteración de “pendientes”

Algoritmo

El algoritmo se encuentra en la carpeta “algoritmos” en el archivo “Dijkstra”. En esa clase se encuentran tres métodos, uno es “dijkstra”, otro es un método que devuelve los vértices adyacentes de un nodo, y un tercer método para poder copiar conjuntos (ya que las instancias de los conjuntos deben ser diferentes para realizar una copia)
En el método Dijkstra se reciben el grafo y el **vértice_parametro** con los que se van a trabajar. La técnica se basa en lo siguiente: se crea primero un grafo auxiliar solamente conectando el vértice dado con sus adyacentes. Posteriormente, se crea un conjunto de **vértices pendientes** con todos los vértices del grafo menos el **vértice_parámetro.**
Se comienza a iterar el conjunto **pendientes** y, utilizando el **vértice con la arista de menor peso conectada al vertice_parametro,** comienzo a preguntar si existe un camino de menor costo hacia otro **vertice_p** que pasa por el **vertice_menor.** De ser asi, se agrega un camino hacia ese vertice_p, sumando los costos que existen entre **vertice_parametro---> vertice_menor----> vertice_p.** Si no, se dejará intacta la arista entre **vertice_parametro y vertice_p**

Prueba

En la carpeta “resources” se encuentra un archivo “PruebaDijkstra” el cual ya posee un caso cargado.