



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: M.I Edgar Tista García

Asignatura: Estructura de Datos y Algoritmos II

Grupo: 9

No de Práctica(s): 6-7

Integrante(s): Díaz Hernández Marcos Bryan

*No. de Equipo de
cómputo empleado:* Equipo Personal

No. de Lista o Brigada: 9

Semestre: 2021-1

Fecha de entrega: 17 de noviembre de 2020

Observaciones:

CALIFICACIÓN: _____

Objetivo de la practica

El estudiante conocerá las formas de representar un grafo e identificará las características necesarias para comprender el algoritmo de búsqueda por expansión.

Introducción

Esta practica esta elaborada con la intención de poder entender los ejercicios realizados, lamentablemente no pude resolver los todos, por cuestiones de mala organización por parte mía pero los que sí, estarán analizados dentro del presente documento.

Ejercicios de la practica:

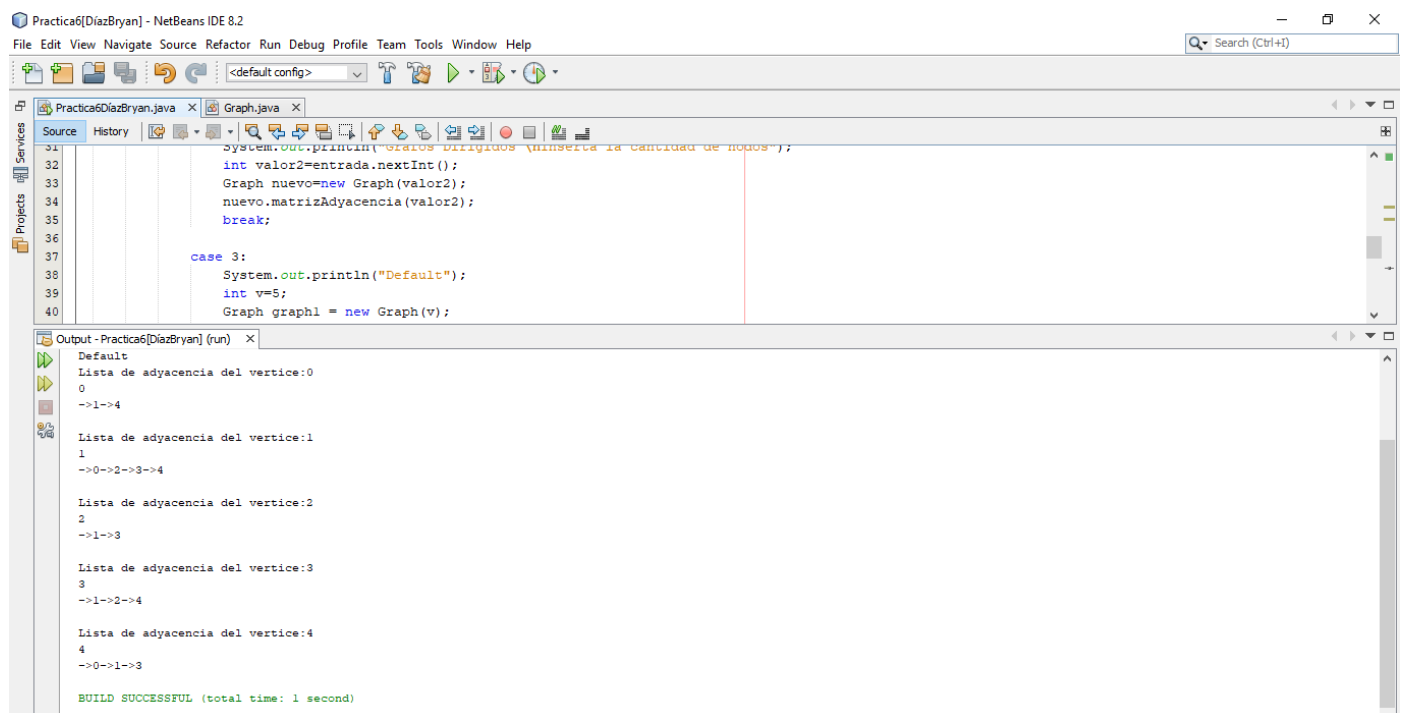
- Ejercicio 1:

El primer ejercicio consistió en copiar el código correspondiente a la representación por medio de listas adyacente de un grafo.

- Relación con la teoría:

Este ejercicio simplemente fue copiar el ejercicio, pero la representación corresponder a un grafo no dirigido, por lo que los elemento se envían a las dos listas para que se pueda representar la relación entre dos vértices.

- Evidencia de implementación



```
Practica6[DiazBryan] - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Practica6DiazBryan.java Graph.java
Source History
31 system.out.println("Ostros dirigidos (inserta la cantidad de nodos)");
32 int valor2=entrada.nextInt();
33 Graph nuevo=new Graph(valor2);
34 nuevo.matrizAdyacencia(valor2);
35 break;
36
37 case 3:
38     System.out.println("Default");
39     int v=5;
40     Graph graph1 = new Graph(v);

Output - Practica6[DiazBryan] (run)
Default
Lista de adyacencia del vertice:0
0
->1->4

Lista de adyacencia del vertice:1
1
->0->2->3->4

Lista de adyacencia del vertice:2
2
->1->3

Lista de adyacencia del vertice:3
3
->1->2->4

Lista de adyacencia del vertice:4
4
->0->1->3

BUILD SUCCESSFUL (total time: 1 second)
```

- Ejercicio 2:

El segundo ejercicio consistía en transformar la representación que se nos presentó, a una que representara nodos dirigidos, y posteriormente permitirle al usuario el poder decidir la cantidad de vértices y la cantidad de aristas que los van a conectar.

1. Compila y ejecuta el proyecto. Realiza un análisis de la implementación presentada, e indica las características que te parezcan relevantes de ésta.

Los elementos más importantes que he se encuentran dentro del código, es la creación de un arreglo de listas, y que cada una de estas guarda el valor de los nodos, es decir que la posición del arreglo indica el valor del nodo y las listas guardan los valores con los que está asociado cada uno de los nodos, por eso se envían valores entre $[0,4]$, porque estos son los valores que tomaría cada uno y por lo mismo dentro de la instrucción de addEdge se debe de colocar en ambas listas, el valor del nodo, ya que eso implica que si A se relaciona con C, C está relacionado con A, y de esta forma se puede colocar la lista más apropiada a la teoría de las listas adyacentes.

```
void addEdge(int v,int w){
    adjArray[v].add(w);
    adjArray[w].add(v);
}
```

Método que permite el representar los grafos no dirigidos.

2. Modifica la clase Graph para que ahora se trabaje con grafos dirigidos.

Por lo que entendí para el grafo dirigido, se tiene que tomar el valor de los nodos a los que llega el nodo, por lo que no tendría que haber doble secuencias de los valores o no se tendría que referencia la llegada y la salida del nodo, por lo que solo llega.

```
void addEdge(int v,int w){
    adjArray[v].add(w);
}
```

Método que permite el representar los grafos dirigidos.

3. Modifica la función principal de tal manera que le permitas al usuario crear su propio grafo indicando la cantidad de nodos y las aristas que los conectan.

```
for(int i=0;i<valor1;i++){
    System.out.println("Inserta la cantidad de aristas para el nodo de valor:"+i);
    aristas=entrada.nextInt();
    for(int j=0;j<aristas;j++){
        System.out.println("Inserta el valor del nodo a donde llega["+(j+1)+"]");
        nodoDestino=entrada.nextInt();
        graph.addEdge(i,nodoDestino);
    }
}
```

Método que permite introducir cantidad de nodos y aristas.

- Dificultades en el código

La mayor dificultad que me encontré dentro de la elaboración de este ejercicio fue el poder crear los nodos, y eso era porque estaba introduciendo valores que no estaba definidos, es decir que le pedía al usuario el valor de los nodos, sabiendo que estos no iban a poder encontrarse en caso de que el arreglo de listas no estuviera definido para ese valor.

```
for(int i=0;i<valor1;i++){
    System.out.println("Inserta la cantidad de aristas para el nodo de valor:"+i);
    aristas=entrada.nextInt();
    // ...
}
```

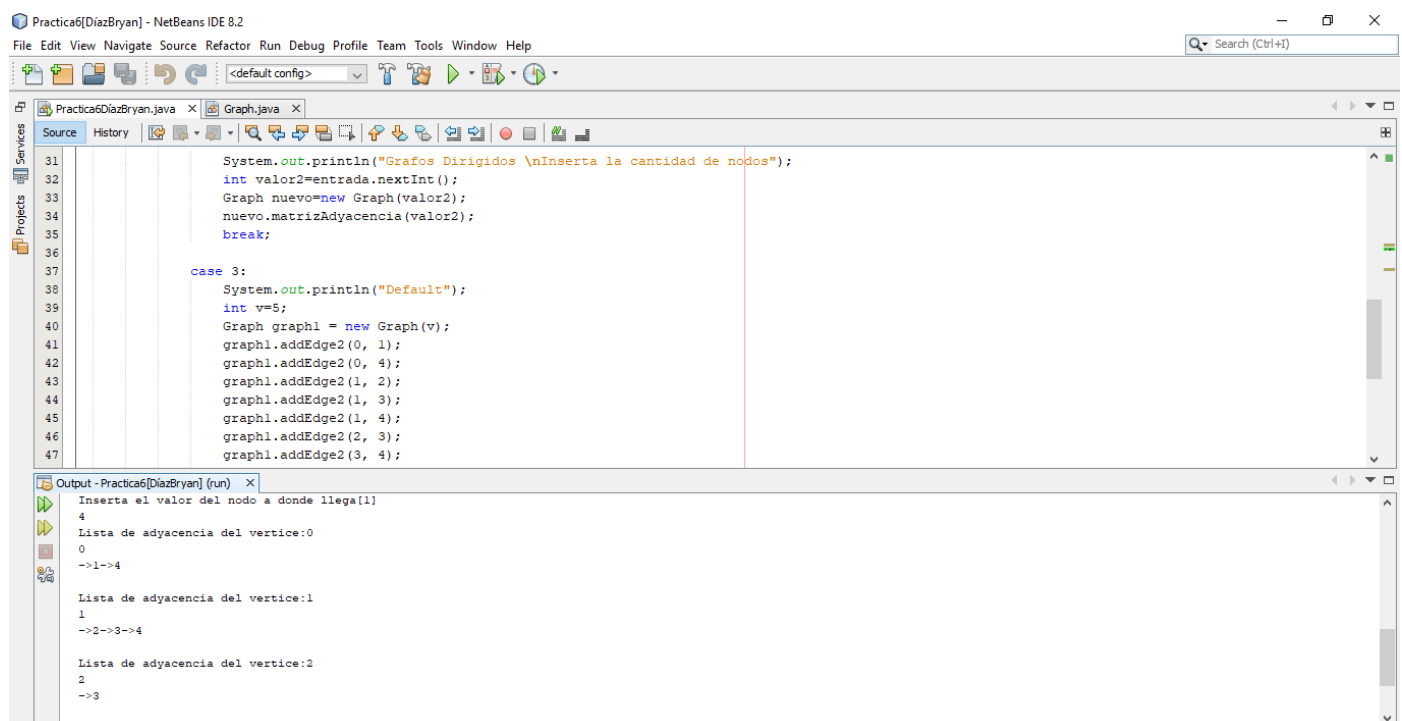
En la línea de código estaba la instrucción que tuve que eliminar.

En este caso el valor de los nodos va a corresponder con la cantidad de los nodos, siendo ese rango de $[0,n]$; donde n es la cantidad de nodos.

- Relación con la teoría:

Con respecto a la carga teórica que tiene este ejercicio, se fundamenta en la forma de representación de los grafos, que sería para un grafo que estuviera dirigido por ello es por lo que la implementación es un poco mas sencilla pero es necesario saber las diferencias entre los tipos de representación para poder saber en que momento realizar la intervención en el código que se nos presenta para poder modificarlo.

- Evidencia de implementación



- Ejercicio 3:

El tercer ejercicio consistió en la elaboración de la representación por medio de la tabla de adyacencia, donde esta es aplicada a los grafos dirigidos por ser más accesible el código para la realización de la representación.

- Dificultades en el código

La mayor dificultad que encontré dentro de la elaboración de este ejercicio el método/conjunto de líneas que buscaran donde colocar la adyacencia, porque en un principio estaba elaborando por parte todo el conjunto que debería funcionar para la elaboración de esta asignación.

Lo primero que tuve que realizar es pensar en como se podría verificar con cuantos nodos esta relacionado el primer nodo, o a cuantos llega, esa fue la primera cuestión que tuve dentro de mi cabeza para poder pensar en que tenía tener for anidados, por lo que el primer for recorre todos los nodos, de forma empírica o que va a repetir todo para cada uno de los nodos, en este caso lo que repetirá será:

1. Nodos con los que se relaciona
2. Inserción de los valores con los cuales se relaciona.
3. Recorrer las filas del arreglo
4. Recorres las columnas del arreglo.

```
int valores,i=0;
for(int z=1;z<v;z++){
    System.out.println("Inserta la cantidad de nodos a donde llega el nodo["+z+"]");
    valores=entrada.nextInt();
    i++;
    for(int j=0;j<valores;j++){
        System.out.println("Inserta el valor del nodo a donde llega["+j+"]");
        nodoDestino=entrada.nextInt();
        for(i=i;i<v;i++){
            for(int k=1;k<v;k++){
                if(arreglo[0][k]==nodoDestino){
                    arreglo[i][k]=1;
                    imprimirMatriz(v,arreglo);
                    break;
                }
            }
        }
        break;
    }
}
```

Bloque de código que resuelve el colocar las relaciones entre los nodos

Algo que me causo problemas fue el recorres fila a fila, porque en un principio el no tener en consideración que los for se van a reiniciar una y otra vez me llevo a cometer el error en pensar en que esto consideraría que cada renglón representa un nodo, entonces esto se repetía una y otra vez sobre el mismo renglón y eventualmente me leía para todos los renglones.

Esto se resolvía agregando un incremento en el for que indicaba las filas, para poder comenzar cada vez en una nueva fila, de esta manera el algoritmo ya funciona de forma adecuada.

```
i++;
for(int j=0;j<valores;j)
    System.out.println(
        nodoDestino=entrada
        for(i=i;i<v;i++){
```

El contador i, permite el salto de fila.

- Relación con la teoría:

Este ejercicio necesita de la teoría al cien, porque realmente es necesario saber el principio de como se arma, para poder colocar los vértices en columnas 0 y la fila 0, además saber dónde colocar los valores de uno, dentro del arreglo, que si no es algo complicado necesita su chiste, por lo que igual me resulto un poco confuso.

En cuanto a lo que no se relaciona con la forma de la representación de los grafos se tiene el uso de los arreglos que es prácticamente todo lo que se utiliza para poder resolver el ejercicio.

- Evidencia de implementación

The screenshot shows the NetBeans IDE interface. The main editor window displays the source code for `Practica6DiazBryan.java`. The code implements a graph structure using a 2D array `arreglo`. It includes methods for inserting nodes and values, and for checking connections between nodes. The output window shows the execution results, including the insertion of nodes and values, and the resulting graph structure.

```

45         arreglo[i+1][0]=nodo;
46     }
47     int valores,i=0;
48     for(int z=1;z<v;z++){
49         System.out.println("Inserta la cantidad de nodos a donde llega el nodo["+z+"]");
50         valores=entrada.nextInt();
51         i++;
52         for(int j=0;j<valores;j++){
53             System.out.println("Inserta el valor del nodo a donde llega["+j+1+"]");
54             nodoDestino=entrada.nextInt();
55             for(int k=1;k<v;k++){
56                 for(int l=1;l<v;l++){
57                     if(arreglo[0][k]==nodoDestino){
58                         arreglo[i][k]=1;

```

The output window shows the following text:

```

Inserta el valor del nodo a donde llega[2]
5
0 1 2 3 4 5
1 0 0 1 1 0
2 1 0 0 0 0
3 0 0 0 1 1
4 0 1 0 0 1
5 0 0 0 0 0
Inserta la cantidad de nodos a donde llega el nodo[5]
0
0 1 2 3 4 5
1 0 0 1 1 0
2 1 0 0 0 0
3 0 0 0 1 1
4 0 1 0 0 1
5 0 0 0 0 0
BUILD SUCCESSFUL (total time: 2 minutes 2 seconds)

```

Conclusiones

Como se habrá dado cuenta no realice todos los ejercicios, simplemente 3 y hasta eso solo dos, porque el primero es solo copiar, por motivos personales no los hice y estos los realice a ultima hora, porque no tenía contemplado el poder elaborarlos, pero los que elabore por lo menos me dejaron una lección y algo de conocimiento sobre las representaciones.

En cuanto al objetivo pues comprendí las formas de representación únicamente ya que no realice los demás ejercicios ni la practica 7, así que el objetivo no se cumplió por completo, únicamente una fracción de él, y posiblemente la mas fácil.