



Tecnológico de Monterrey

Casos de prueba Grafos

Programming of data structures

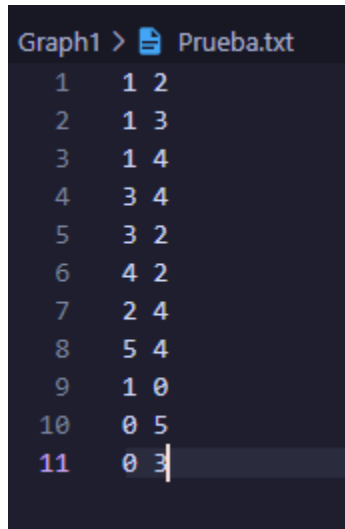
Inés Alejandro Garcia Mosqueda A00834571

Profesores: Luis Ricardo Peña Llamas
Jorge Gonzalez

07/11/2022

Caso 1

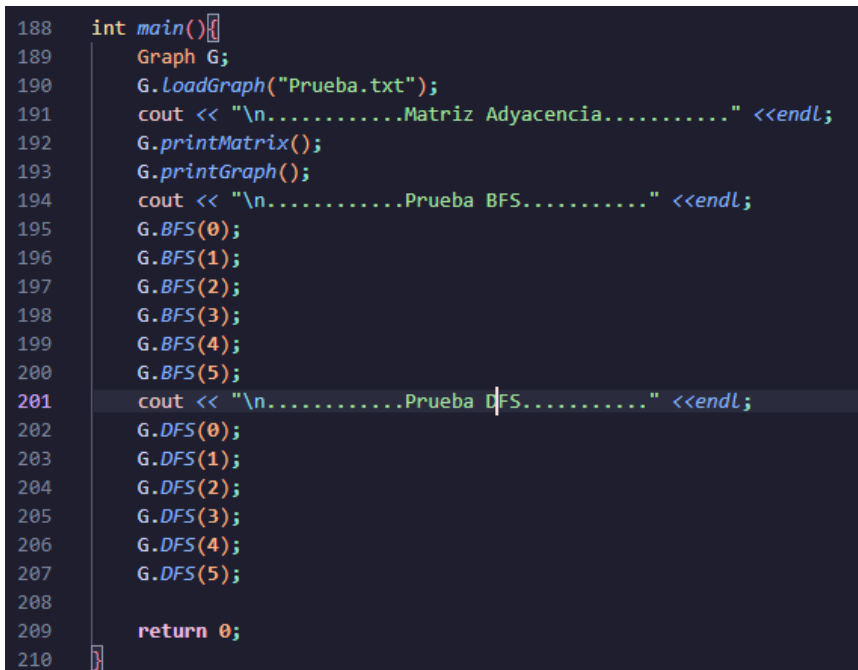
Los casos de prueba son implementando grafos dirigidos



```
Graph1 > Prueba.txt
1  1 2
2  1 3
3  1 4
4  3 4
5  3 2
6  4 2
7  2 4
8  5 4
9  1 0
10 0 5
11 0 3
```

Se cargan los grafos desde un archivo txt para ingresar los datos de una forma más rápida (los grafos ingresados son aleatorios)

El caso de prueba es el siguiente:



```
188 int main()
189     Graph G;
190     G.LoadGraph("Prueba.txt");
191     cout << "\n.....Matriz Adyacencia....." <<endl;
192     G.printMatrix();
193     G.printGraph();
194     cout << "\n.....Prueba BFS....." <<endl;
195     G.BFS(0);
196     G.BFS(1);
197     G.BFS(2);
198     G.BFS(3);
199     G.BFS(4);
200     G.BFS(5);
201     cout << "\n.....Prueba DFS....." <<endl;
202     G.DFS(0);
203     G.DFS(1);
204     G.DFS(2);
205     G.DFS(3);
206     G.DFS(4);
207     G.DFS(5);
208
209     return 0;
210 }
```

Se ingresa el nombre del archivo como parámetro de loadGraph, la cual llena una matriz de adyacencia y una lista de adyacencia, la cual es indispensable para el funcionamiento del BFS y DFS

(La matriz y la lista de adyacencia son generados dinámicamente, por lo que se implementan otras estructuras de datos internamente para la creación de la matriz y la lista)

Dentro del caso de prueba se prueban las funciones BFS y DFS por cada grafo que se ingreso, obteniendo los siguientes resultados:

```
.....Matriz Adyacencia.....
0 0 0 1 0 1
1 0 1 1 1 0
0 0 0 0 1 0
0 0 1 0 1 0
0 0 1 0 0 0
0 0 0 0 1 0

Vertex 0: -> 5 -> 3
Vertex 1: -> 2 -> 3 -> 4 -> 0
Vertex 2: -> 4
Vertex 3: -> 4 -> 2
Vertex 4: -> 2
Vertex 5: -> 4

.....Prueba BFS.....
BFS start vertex (0) :
0 5 3 4 2

BFS start vertex (1) :
1 2 3 4 0 5

BFS start vertex (2) :
2 4

BFS start vertex (3) :
3 4 2

BFS start vertex (4) :
4 2

BFS start vertex (5) :
5 4 2
```

```
.....Prueba DFS.....

DFS start vertex (0) :
0 5 4 2 3

DFS start vertex (1) :
1 2 4 3 0 5

DFS start vertex (2) :
2 4

DFS start vertex (3) :
3 4 2

DFS start vertex (4) :
4 2

DFS start vertex (5) :
5 4 2
```

Caso 2

Se analizará la siguiente entrada como caso de prueba

```
Graph1 > Prueba2.txt
1 1 2
2 3 6
3 10 6
4 2 3
5 4 5
6 0 5
7 9 6
8 6 7
9 3 2
10 8 7
11 9 1
12 1 4
13 1 9
14 1 8
15 3 4
16 6 3
17 7 2
18 3 7
19 10 8
20 5 10
21 3 10
22 7 1
```

Repitiendo el mismo código de caso de prueba que en el caso 1, se obtienen los siguientes resultados:

.....Matriz Adyacencia.....Prueba BFS.....Prueba DFS.....
0 0 0 0 0 1 0 0 0 0 0	BFS start vertex (0) :	DFS start vertex (0) :
0 0 1 0 1 0 0 0 0 1 1 0	0 5 10 6 8 7 3 2 1 4 9	0 5 10 6 7 2 3 4 1 9 8
0 0 0 1 0 0 0 0 0 0 0 0		
0 0 1 0 1 0 1 1 0 0 0 1	BFS start vertex (1) :	DFS start vertex (1) :
0 0 0 0 0 1 0 0 0 0 0 0	1 2 4 9 8 3 5 6 7 10	1 2 3 6 7 4 5 10 8 9
0 0 0 0 0 0 0 0 0 0 0 1		
0 0 0 1 0 0 0 0 1 0 0 0	BFS start vertex (2) :	DFS start vertex (2) :
0 1 1 0 0 0 0 0 0 0 0 0	2 3 6 4 7 10 5 1 8 9	2 3 6 7 1 4 5 10 8 9
0 0 0 0 0 0 0 0 1 0 0 0		
0 1 0 0 0 0 1 0 0 0 0 0	BFS start vertex (3) :	DFS start vertex (3) :
0 0 0 0 0 0 0 1 0 1 0 0	3 6 2 4 7 10 5 1 8 9	3 6 7 2 1 4 5 10 8 9
Vertex 0: -> 5	BFS start vertex (4) :	DFS start vertex (4) :
Vertex 1: -> 2 -> 4 -> 9 -> 8	4 5 10 6 8 7 3 2 1 9	4 5 10 6 7 2 3 1 9 8
Vertex 2: -> 3	BFS start vertex (5) :	DFS start vertex (5) :
Vertex 3: -> 6 -> 2 -> 4 -> 7 -> 10	5 10 6 8 7 3 2 1 4 9	5 10 6 7 2 3 4 1 9 8
Vertex 4: -> 5	BFS start vertex (6) :	DFS start vertex (6) :
	6 7 3 2 1 4 10 9 8 5	6 7 2 3 4 5 10 8 1 9
Vertex 5: -> 10	BFS start vertex (7) :	DFS start vertex (7) :
Vertex 6: -> 7 -> 3	7 2 1 3 4 9 8 6 10 5	7 2 3 6 4 5 10 8 1 9
Vertex 7: -> 2 -> 1	BFS start vertex (8) :	DFS start vertex (8) :
	8 7 2 1 3 4 9 6 10 5	8 7 2 3 6 4 5 10 1 9
Vertex 8: -> 7	BFS start vertex (9) :	DFS start vertex (9) :
Vertex 9: -> 6 -> 1	9 6 1 7 3 2 4 8 10 5	9 6 7 2 3 4 5 10 8 1
Vertex 10: -> 6 -> 8	BFS start vertex (10) :	DFS start vertex (10) :
	10 6 8 7 3 2 1 4 9 5	10 6 7 2 3 4 5 1 9 8

Conclusión de casos prueba

Si se analiza la matriz de adyacencia generada y los resultados de las búsquedas BFS y DFS, e puede concluir que el programa funciona, ya que al ser los grafos direccionados, efectivamente se generan las búsquedas impresas por las funciones implementadas