

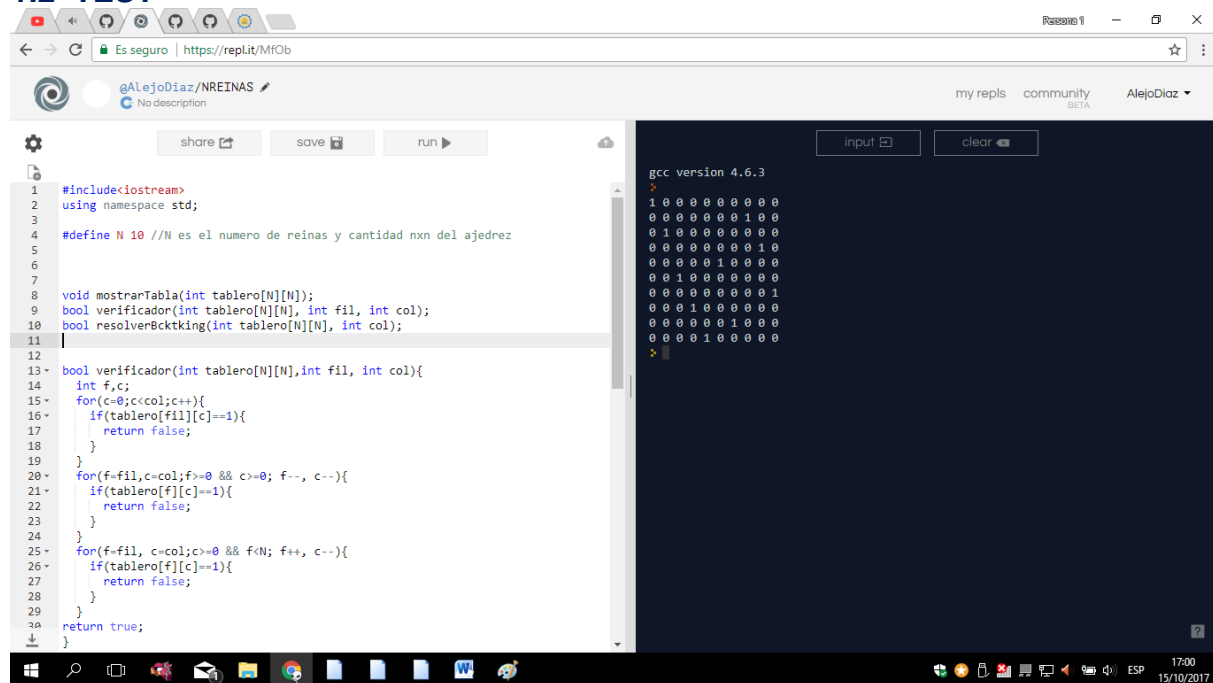
Laboratorio Nro. 3: Vuelta atrás(BackTracking)

Juan Gonzalo Quiroz Cadavid
Universidad Eafit
Medellín, Colombia
jquiro12@eafit.edu.co

Alejandro Díaz Cano
Universidad Eafit
Medellín, Colombia
adiazc@eafit.edu.co

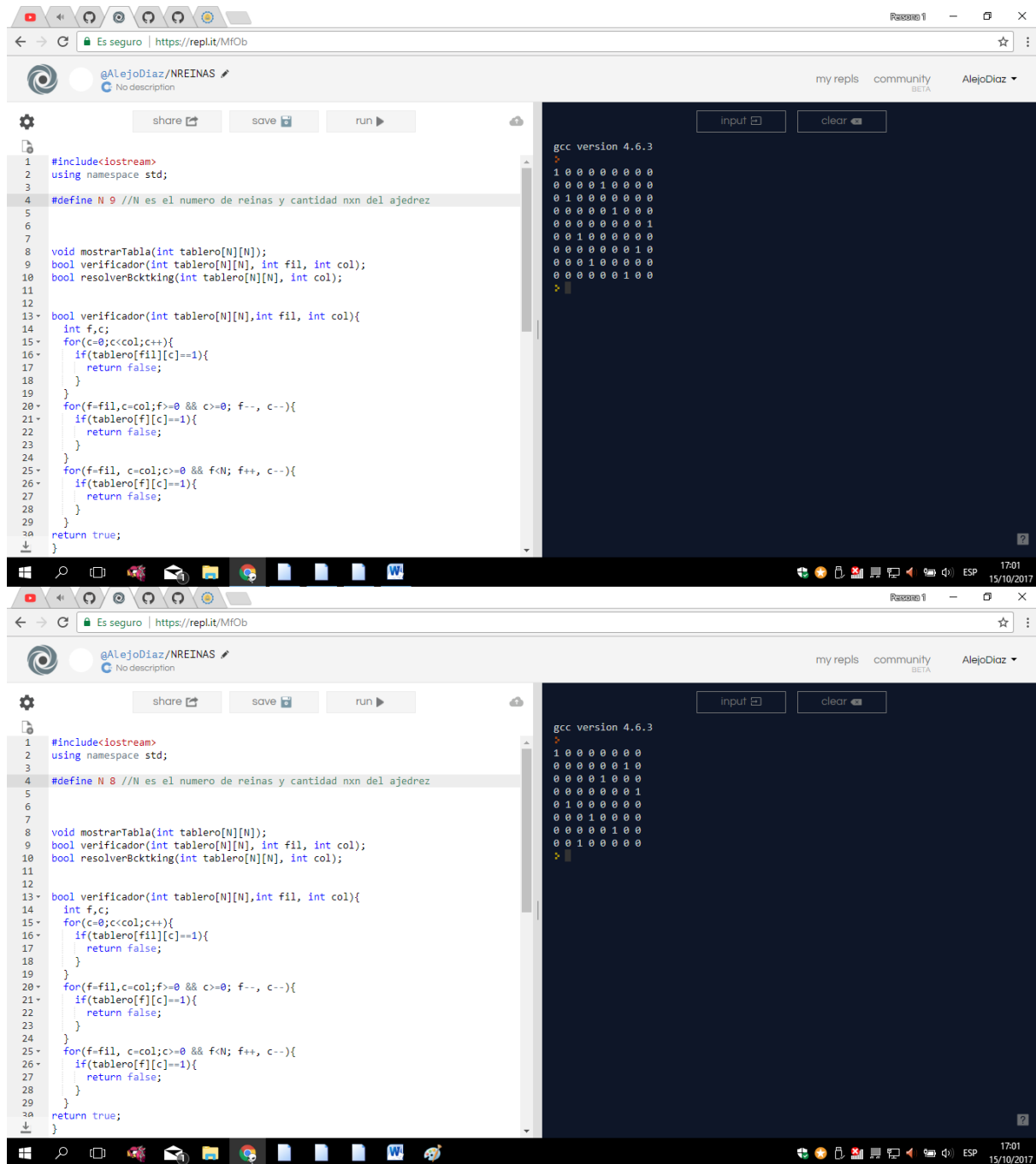
1)1. parte código-> Github

1.2 TEST



The screenshot shows a Repl.it IDE window. The left pane contains C++ code for a Backtracking algorithm to solve the N-Queens problem. The code includes headers, a constant N=10, and functions for displaying the board, verifying a position, and solving the problem. The right pane shows the output of the program, which is a 10x10 grid of 0s and 1s representing a valid solution. The output is as follows:

```
gcc version 4.6.3
1 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0
0 1 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0
0 0 0 0 0 1 0 0 0 0
0 0 1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 1 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0 0 0
```

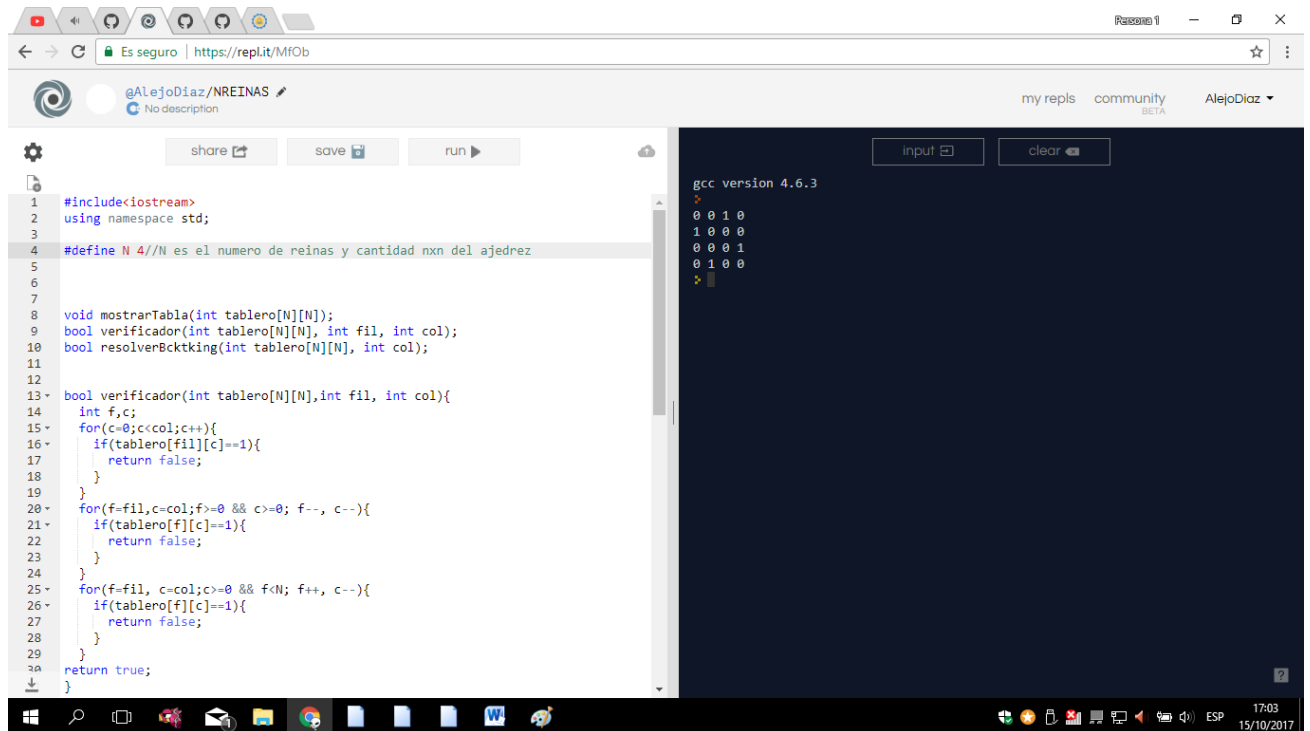


```
1 #include<iostream>
2 using namespace std;
3
4 #define N 9 //N es el numero de reinas y cantidad nxn del ajedrez
5
6
7
8 void mostrarTabla(int tablero[N][N]);
9 bool verificador(int tablero[N][N], int fil, int col);
10 bool resolverBcktking(int tablero[N][N], int col);
11
12
13 bool verificador(int tablero[N][N],int fil, int col){
14     int f,c;
15     for(c=0;c<col;c++){
16         if(tablero[fil][c]==1){
17             return false;
18         }
19     }
20     for(f=fil,c=col;f>=0 && c>=0; f--, c--){
21         if(tablero[f][c]==1){
22             return false;
23         }
24     }
25     for(f=fil, c=col;c>=0 && f<N; f++, c--){
26         if(tablero[f][c]==1){
27             return false;
28         }
29     }
30     return true;
31 }
```

```
gcc version 4.6.3
1 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 1 0 0 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 1
0 0 0 0 0 1 0 0
```

```
1 #include<iostream>
2 using namespace std;
3
4 #define N 8 //N es el numero de reinas y cantidad nxn del ajedrez
5
6
7
8 void mostrarTabla(int tablero[N][N]);
9 bool verificador(int tablero[N][N], int fil, int col);
10 bool resolverBcktking(int tablero[N][N], int col);
11
12
13 bool verificador(int tablero[N][N],int fil, int col){
14     int f,c;
15     for(c=0;c<col;c++){
16         if(tablero[fil][c]==1){
17             return false;
18         }
19     }
20     for(f=fil,c=col;f>=0 && c>=0; f--, c--){
21         if(tablero[f][c]==1){
22             return false;
23         }
24     }
25     for(f=fil, c=col;c>=0 && f<N; f++, c--){
26         if(tablero[f][c]==1){
27             return false;
28         }
29     }
30     return true;
31 }
```

```
gcc version 4.6.3
1 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 1
0 1 0 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
0 0 1 0 0 0 0 0
```



```

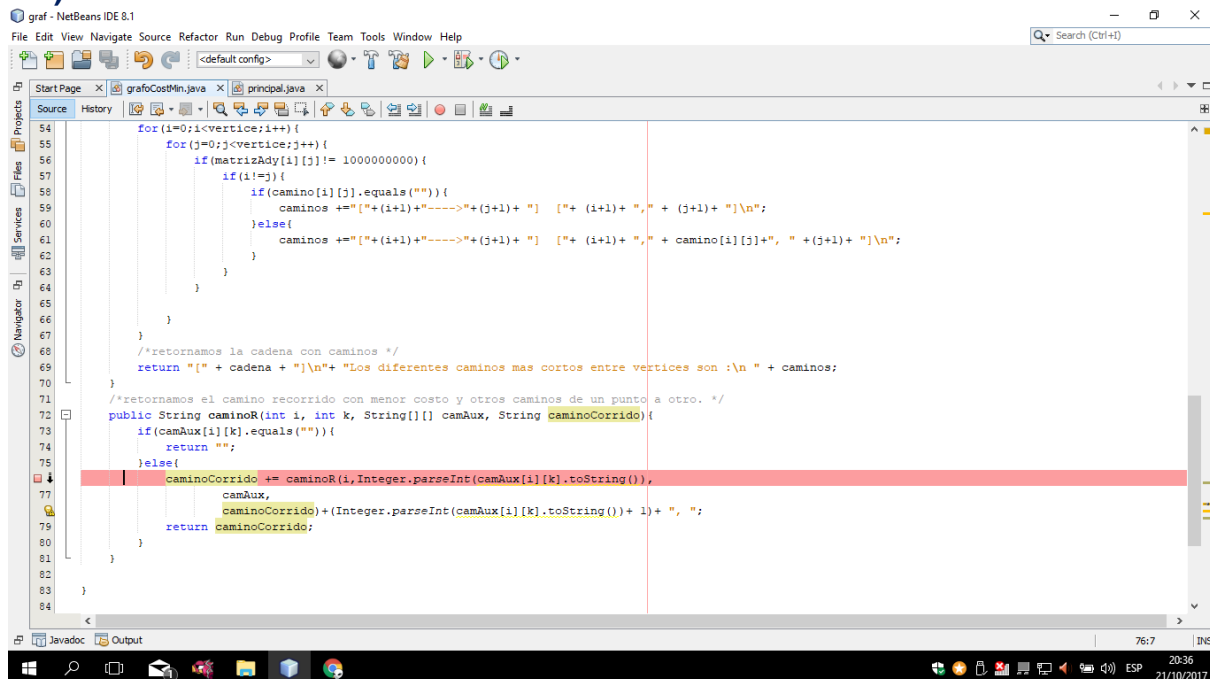
1 #include<iostream>
2 using namespace std;
3
4 #define N 4 //N es el numero de reinas y cantidad nxn del ajedrez
5
6
7
8 void mostrarTabla(int tablero[N][N]);
9 bool verificador(int tablero[N][N], int fil, int col);
10 bool resolverBcktking(int tablero[N][N], int col);
11
12
13 bool verificador(int tablero[N][N],int fil, int col){
14     int f,c;
15     for(c=0;c<col;c++){
16         if(tablero[fil][c]==1){
17             return false;
18         }
19     }
20     for(f=fil,c=col;f>=0 && c>=0; f--, c--){
21         if(tablero[f][c]==1){
22             return false;
23         }
24     }
25     for(f=fil, c=col;c>=0 && f<N; f++, c--){
26         if(tablero[f][c]==1){
27             return false;
28         }
29     }
30     return true;
31 }
  
```

```

gcc version 4.6.3
>
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0
>
  
```

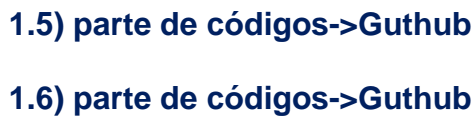
1.3)Parte de Codigos->Github

1.4)

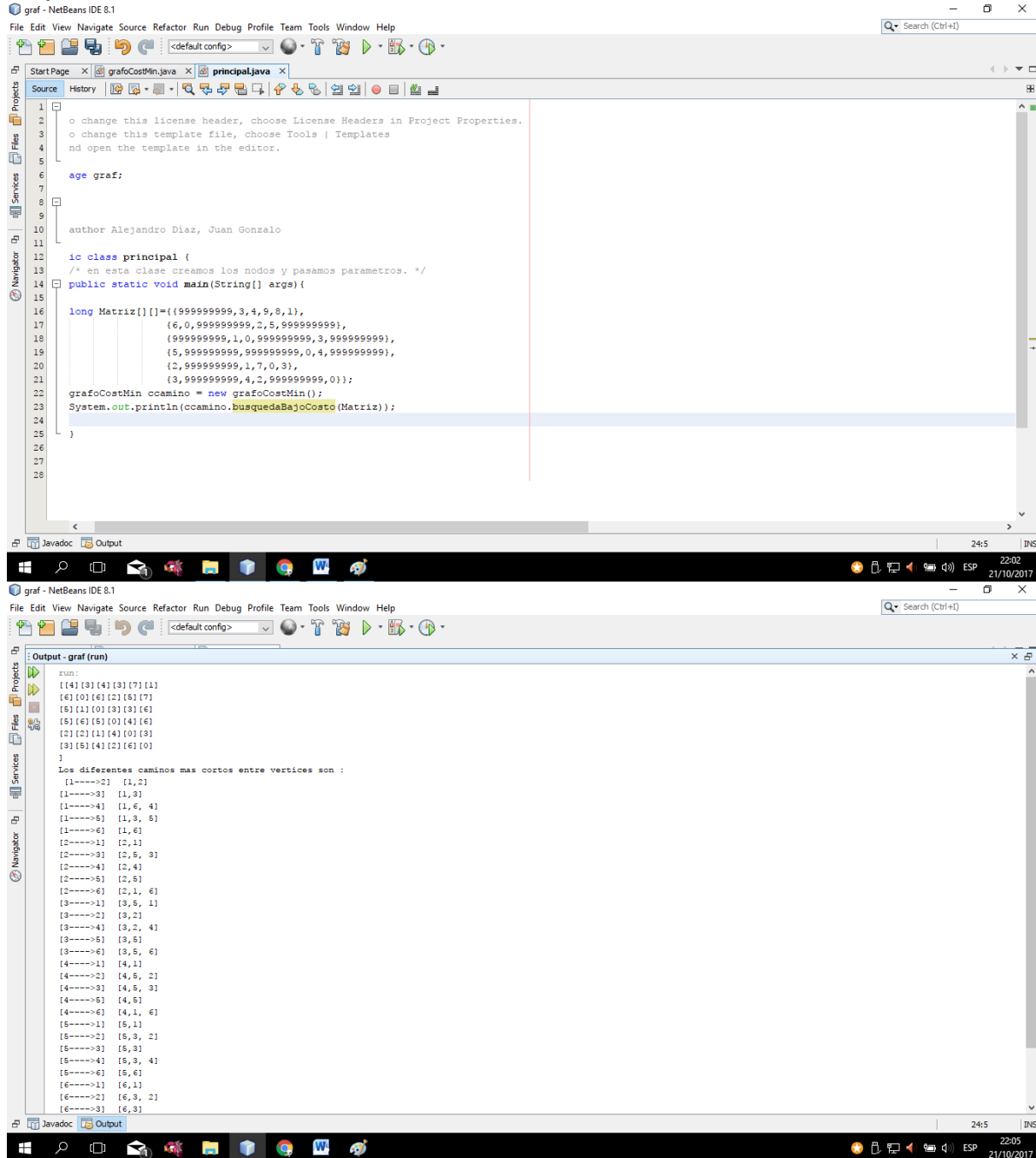


```

54 for(i=0;i<vertices;i++){
55     for(j=0;j<vertices;j++){
56         if(matrizAdj[i][j]!= 1000000000){
57             if(i!=j){
58                 if(camino[i][j].equals("")){
59                     caminos += "(i+1)+----->(j+1)+ " + "(i+1)+ ", " + "(j+1)+ " + "\n";
60                 }else{
61                     caminos += "(i+1)+----->(j+1)+ " + "(i+1)+ ", " + camino[i][j]+", " + "(j+1)+ " + "\n";
62                 }
63             }
64         }
65     }
66 }
67
68 /*retornamos la cadena con caminos */
69 return "(" + cadena + ")\n" + "Los diferentes caminos mas cortos entre vertices son :\n " + caminos;
70
71 /*retornamos el camino recorrido con menor costo y otros caminos de un punto a otro. */
72 public String caminoR(int i, int k, String[][] camAux, String caminoCorrido){
73     if(camAux[i][k].equals("")){
74         return "";
75     }else{
76         caminoCorrido += caminoR(i,Integer.parseInt(camAux[i][k].toString()),
77                                 camAux,
78                                 caminoCorrido+(Integer.parseInt(camAux[i][k].toString())+ 1)+ " ",
79                                 caminoCorrido);
80     }
81 }
82
83 }
84
  
```



1.7)



The screenshot displays the NetBeans IDE 8.1 interface. The main editor window shows the source code of a Java file named 'principal.java'. The code includes a license header, a package declaration 'age graf;', an author comment 'author Alejandro Diaz, Juan Gonzalo', and a class definition 'ic class principal {'. Inside the class, there is a comment '/* en esta clase creamos los nodos y pasamos parametros. */' and a public static void main method. The main method initializes a long matrix 'Matrix' with a 6x6 grid of values. It then creates an instance of 'grafoCostMin' and calls its 'busquedaBajoCosto' method, passing the matrix. The output window at the bottom shows the execution results, including the matrix values and a list of shortest paths between vertices.

```
1  o change this license header, choose License Headers in Project Properties.
2  o change this template file, choose Tools | Templates
3  nd open the template in the editor.
4
5
6  age graf;
7
8
9
10 author Alejandro Diaz, Juan Gonzalo
11
12
13 ic class principal {
14     /* en esta clase creamos los nodos y pasamos parametros. */
15     public static void main(String[] args){
16
17         long Matrix[][] = {{999999999, 3, 4, 9, 8, 1},
18                             {6, 0, 999999999, 2, 5, 999999999},
19                             {999999999, 1, 0, 999999999, 3, 999999999},
20                             {5, 999999999, 999999999, 0, 4, 999999999},
21                             {2, 999999999, 1, 7, 0, 3},
22                             {3, 999999999, 4, 2, 999999999, 0}};
23
24         grafoCostMin ccaminio = new grafoCostMin();
25         System.out.println(ccaminio.busquedaBajoCosto(Matrix));
26
27     }
28 }
```

run:

```
[[4] [3] [4] [3] [7] [1]
[6] [0] [6] [2] [6] [7]
[6] [1] [0] [3] [3] [6]
[8] [6] [6] [0] [4] [6]
[2] [2] [1] [4] [0] [3]
[3] [5] [4] [2] [6] [0]
]
```

Los diferentes caminos mas cortos entre vertices son :

```
[1---->2] [1, 2]
[1---->3] [1, 3]
[1---->4] [1, 6, 4]
[1---->5] [1, 3, 5]
[1---->6] [1, 6]
[2---->1] [2, 1]
[2---->3] [2, 5, 3]
[2---->4] [2, 4]
[2---->5] [2, 5]
[2---->6] [2, 1, 6]
[3---->1] [3, 5, 1]
[3---->2] [3, 2]
[3---->4] [3, 2, 4]
[3---->5] [3, 5]
[3---->6] [3, 5, 6]
[4---->1] [4, 1]
[4---->2] [4, 6, 2]
[4---->3] [4, 6, 3]
[4---->5] [4, 5]
[4---->6] [4, 1, 6]
[5---->1] [5, 1]
[5---->2] [5, 3, 2]
[5---->3] [5, 3]
[5---->4] [5, 9, 4]
[5---->6] [5, 6]
[6---->1] [6, 1]
[6---->2] [6, 3, 2]
[6---->3] [6, 3]
```

1.8). Paso un grafo como parámetro y verifico sus conexiones, almaceno las rutas o enlaces entre los nodos del grafo, después de almacenarlos en variables auxiliares comparo las rutas de menor costo entre los puntos y dejo los caminos de menor costos, y después muestro en pantalla sus conexiones.

2).1. parte de códigos->Github

3) Simulacro de preguntas de sustentación de Proyectos

Los algoritmos más importantes para resolver este problema son:

1. [“El algoritmo de Dijkstra”](#) resuelve el problema de ruta más corta de fuente única.

[El algoritmo Bellman-Ford](#) resuelve el problema de fuente única si los pesos de borde pueden ser negativos.

[Un algoritmo de búsqueda *](#) resuelve la ruta más corta de un solo par utilizando heurísticas para intentar acelerar la búsqueda.

[El algoritmo Floyd-Warshall](#) resuelve todos los caminos más cortos de los pares.

[El algoritmo de Johnson](#) resuelve todos los caminos más cortos de los pares, y puede ser más rápido que Floyd-Warshall en [gráficos dispersos](#).

[El algoritmo Viterbi](#) resuelve el problema más corto de la ruta estocástica con un peso probabilístico adicional en cada nodo.”

Tomado de: https://en.wikipedia.org/wiki/Shortest_path_problem#Algorithms

2.

Valor de N	Fuerza bruta	BackTracking
4	1 s	1 s
8	3 mns	1 s
16	Mas de 5 mns	2s
32	Mas de 5 mns	3:20 mn
N	n!	$T(n) = 2 T(n-1) + 1$

3. DFS es mejor usarlo cuando se desea buscar las conexiones con todos los nodos buscando en arboles, etc. Y BFS es mejo cuando se quiere buscar el camino mas corto aunque puede tomar mas tiempo.

4.-A*:”El problema de algunos algoritmos de búsqueda en grafos informados, como puede ser el [algoritmo voraz](#), es que se guían en exclusiva por la [función heurística](#), la cual puede no

indicar el camino de coste más bajo, o por el coste real de desplazarse de un nodo a otro (como los [algoritmos de escalada](#)), pudiéndose dar el caso de que sea necesario realizar un movimiento de coste mayor para alcanzar la solución. Es por ello bastante intuitivo el hecho de que un buen algoritmo de búsqueda informada debería tener en cuenta ambos factores, el valor heurístico de los nodos y el coste real del recorrido.

Así, el algoritmo A* utiliza una función de evaluación , donde representa el valor

heurístico del nodo a evaluar desde el actual, n , hasta el final, y , el coste real del camino recorrido para llegar a dicho nodo, n , desde el nodo inicial. A* mantiene dos estructuras de datos auxiliares, que podemos denominar *abiertos*, implementado como una cola de prioridad

(ordenada por el valor de cada nodo), y *cerrados*, donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté

primero en abiertos, y en caso de que no sea un nodo objetivo, calcula la de todos sus hijos, los inserta en abiertos, y pasa el nodo evaluado a cerrados.

El algoritmo es una combinación entre búsquedas del tipo [primero en anchura](#) con [primero en](#)

[profundidad](#): mientras que tiende a primero en profundidad, tiende a primero en anchura. De este modo, se cambia de camino de búsqueda cada vez que existen nodos más prometedores." Tomado de: https://es.wikipedia.org/wiki/Algoritmo_de_b%C3%BAsqueda_A*

-Breadth First Search: es un [algoritmo](#) de [búsqueda no informada](#) utilizado para recorrer o buscar elementos en un [grafo](#) (usado frecuentemente sobre [árboles](#)). Intuitivamente, se comienza en la raíz (eligiendo algún nodo como elemento raíz en el caso de un grafo) y se exploran todos los vecinos de este nodo. A continuación para cada uno de los vecinos se exploran sus respectivos vecinos adyacentes, y así hasta que se recorra todo el árbol. **"tomado de:**
https://es.wikipedia.org/wiki/B%C3%BAsqueda_en_anchura

-Greedy Best First Search: es un [algoritmo de búsqueda](#) que explora un [gráfico](#) expandiendo el nodo más prometedor elegido según una regla especificada.

[Judea Pearl](#) describió la mejor primera búsqueda como la estimación de la promesa del

nodo n mediante una "función de evaluación heurística" que, en general, puede depender de la descripción de n , la descripción del objetivo, la información recopilada por la búsqueda hasta ese punto y, lo más importante, de cualquier conocimiento adicional sobre el dominio del problema ". ^{[1][2]}

Algunos autores han utilizado la "mejor primera búsqueda" para referirse específicamente a una búsqueda con una [heurística](#) que intenta predecir qué tan cerca está el final de una ruta a una solución, de modo que las rutas que se juzgan más cercanas a una solución se extienden primero. Este tipo específico de búsqueda se denomina búsqueda [codiciosa - primera](#) búsqueda

"tomado de: https://en.wikipedia.org/wiki/Best-first_search

Uniform Cost Search:" "El algoritmo de Dijkstra, que tal vez sea más conocido, puede considerarse una variante de búsqueda de costo uniforme, donde no hay un estado de objetivo y el procesamiento continúa hasta que todos los nodos se hayan eliminado de la cola de prioridad, es decir, hasta las rutas más cortas a todos los nodos (no solo un nodo objetivo) se han determinado ". - Esto no es una diferencia. Esto es exactamente cómo el algoritmo de Dijkstra se utiliza normalmente para problemas de un solo objetivo de fuente única. " **tomado de:**


https://en.wikipedia.org/wiki/Talk%3AUniform-cost_search

Búsqueda binaria:" Un algoritmo de búsqueda es aquel que está diseñado para localizar un elemento con ciertas propiedades dentro de una estructura de datos; por ejemplo, ubicar el registro correspondiente a cierta persona en una base de datos, o el mejor movimiento en una partida de ajedrez. ...

"tomado de: https://es.wikipedia.org/wiki/B%C3%BAsqueda_binaria

4) Simulacro de Parcial

1. A. $n+a+b+c$
B. n,a
C. b,c
2. A. $\text{path}[0]$
B. $v, \text{graph}[\text{pos}-1][\text{path}[v]], \text{path}[], v$
C. $\text{graph}[\text{path}[\text{pos}-1]][\text{path}[0]], \text{path}[v], \text{pos}$
3. A). $0 \rightarrow 3 \rightarrow 7 \rightarrow 4 \rightarrow 2 \rightarrow 1 \rightarrow 5 \rightarrow 6 \rightarrow \text{NULL}$
 $1 \rightarrow 5 \rightarrow 2 \rightarrow 6 \rightarrow 4 \rightarrow 0 \rightarrow 3 \rightarrow 7 \rightarrow \text{NULL}$
 $2 \rightarrow 1 \rightarrow 5 \rightarrow 0 \rightarrow 3 \rightarrow 7 \rightarrow 4 \rightarrow 6 \rightarrow \text{NULL}$
 $3 \rightarrow 7 \rightarrow \text{NULL}$
 $4 \rightarrow 2 \rightarrow 1 \rightarrow 5 \rightarrow 0 \rightarrow 3 \rightarrow 7 \rightarrow 6 \rightarrow \text{NULL}$
 $5 \rightarrow \text{NULL}$
 $6 \rightarrow 2 \rightarrow 1 \rightarrow 5 \rightarrow 0 \rightarrow 3 \rightarrow 7 \rightarrow 4 \rightarrow \text{NULL}$
 $7 \rightarrow \text{NULL}$
B). $0 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 5 \rightarrow \text{NULL}$
 $1 \rightarrow 2 \rightarrow 6 \rightarrow 5 \rightarrow 0 \rightarrow 3 \rightarrow 4 \rightarrow 7 \rightarrow \text{NULL}$
 $2 \rightarrow 1 \rightarrow 6 \rightarrow 4 \rightarrow 0 \rightarrow 5 \rightarrow 3 \rightarrow 7 \rightarrow \text{NULL}$
 $3 \rightarrow 7 \rightarrow \text{NULL}$
 $4 \rightarrow 2 \rightarrow 0 \rightarrow 6 \rightarrow 1 \rightarrow 0 \rightarrow 5 \rightarrow 3 \rightarrow 7 \rightarrow \text{NULL}$
 $5 \rightarrow \text{NULL}$
 $6 \rightarrow 2 \rightarrow 1 \rightarrow 4 \rightarrow 5 \rightarrow 0 \rightarrow 3 \rightarrow 7 \rightarrow \text{NULL}$
 $7 \rightarrow \rightarrow \text{NULL}$

	UNIVERSIDAD EAFIT ESCUELA DE INGENIERÍA DEPARTAMENTO DE INFORMÁTICA Y SISTEMAS	Código: ST245
		Estructura de Datos 1

5. Lectura recomendada (opcional)

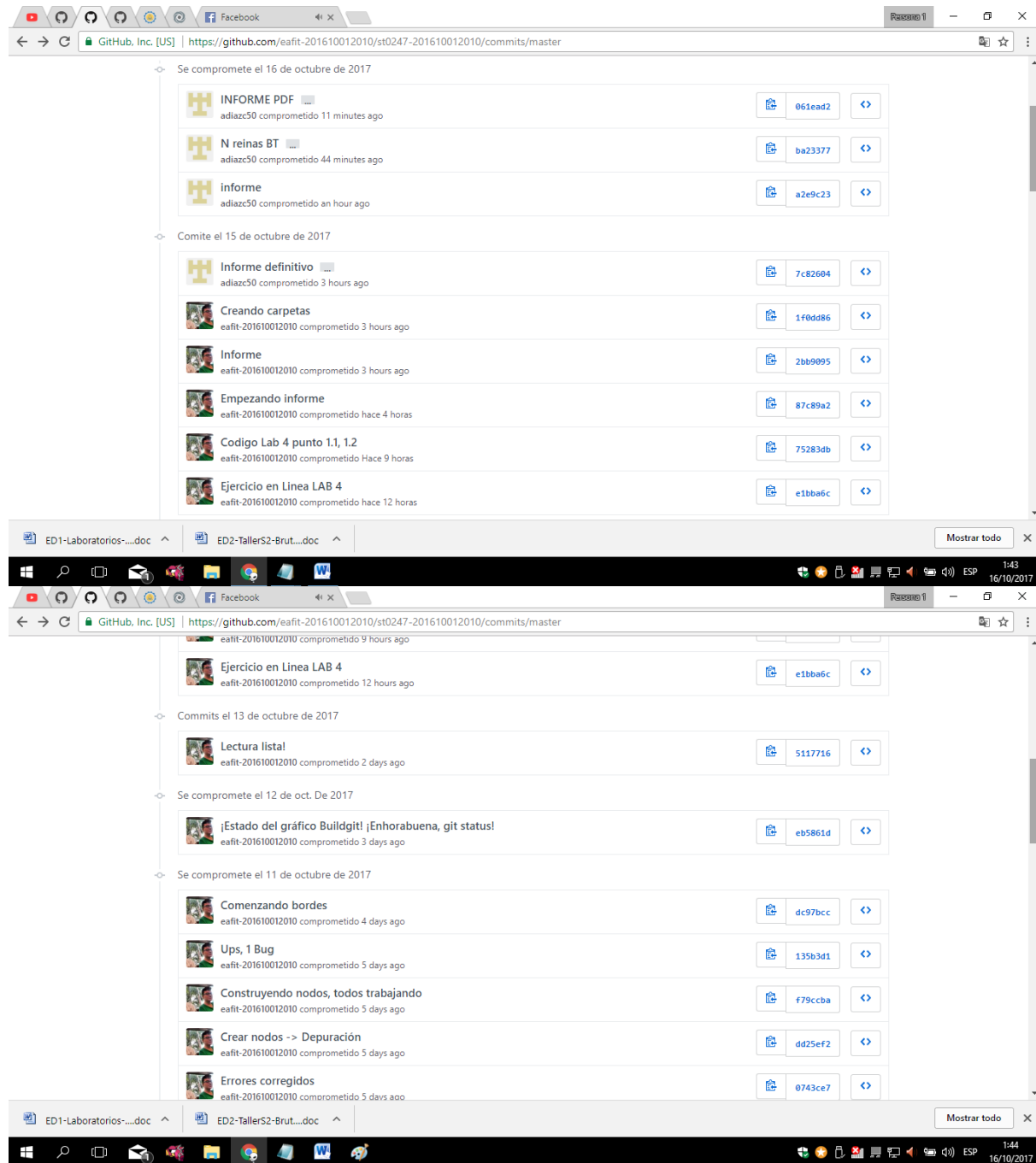
- a) Título
- b) Ideas principales
- c) Mapa de Conceptos

6. Trabajo en Equipo y Progreso Gradual (Opcional)

a) Actas de reunión

Lunes	Martes	Miercoles	JUEVES	VIERNES	SABADO	DOMINGO
	10/10/2017			05/10/2017	14/10/2017	
	7:30 - 9:00 AM			7:30-9:00 AM	2-5 PM	
	Juan Gonzalo			Juan Gonzalo	Juan Gonzalo	
	Quiroz Cadavid			Quiroz Cadavid	Quiroz Cadavid	
	Alejandro			Alejandro	Alejandro	
	Díaz Cano			Díaz Cano	Díaz Cano	
				12/10/2017		
				7:30-9:00 AM		
				Juan Gonzalo		
				Quiroz Cadavid		
				Alejandro		
				Díaz Cano		

b) El reporte de cambios en el código



Se compromete el 16 de octubre de 2017

- INFORME PDF (adiazc50 comprometido 11 minutos ago) 061ead2
- N reinas BT (adiazc50 comprometido 44 minutos ago) ba23377
- informe (adiazc50 comprometido an hour ago) a2e9c23

Comite el 15 de octubre de 2017

- Informe definitivo (adiazc50 comprometido 3 hours ago) 7c82604
- Creando carpetas (eafit-201610012010 comprometido 3 hours ago) 1f6dd86
- Informe (eafit-201610012010 comprometido 3 hours ago) 2bb9095
- Empezando informe (eafit-201610012010 comprometido hace 4 horas) 87c89a2
- Codigo Lab 4 punto 1.1, 1.2 (eafit-201610012010 comprometido Hace 9 horas) 75283db
- Ejercicio en Linea LAB 4 (eafit-201610012010 comprometido hace 12 horas) e1bba6c

Comits el 13 de octubre de 2017

- Lectura lista! (eafit-201610012010 comprometido 2 days ago) 5117716

Se compromete el 12 de oct. De 2017

- ¡Estado del gráfico Buildgit! ¡Enhorabuena, git status! (eafit-201610012010 comprometido 3 days ago) eb5861d

Se compromete el 11 de octubre de 2017

- Comenzando bordes (eafit-201610012010 comprometido 4 days ago) dc97bcc
- Ups, 1 Bug (eafit-201610012010 comprometido 5 days ago) 135b3d1
- Construyendo nodos, todos trabajando (eafit-201610012010 comprometido 5 days ago) f79ccba
- Crear nodos -> Depuración (eafit-201610012010 comprometido 5 days ago) dd25ef2
- Errores corregidos (eafit-201610012010 comprometido 5 days ago) 0743ce7

- c) El reporte de cambios del informe de laboratorio
Todo se hizo en Word.