Taller algoritmo dijkstra

Santiago Jose Hernandez Rendon

Isaias Acosta

ejercicio en clase a matriz

	a	b	c	d	e	f	g
a	0	2	4	0	0	0	0
b	2	0	0	5	0	0	0
c	4	0	0	8	0	0	0
d	0	5	8	0	10	15	0
e	0	0	0	10	0	6	2
f	0	0	0	15	6	0	6
g	0	0	0	0	2	6	0

Codigo:

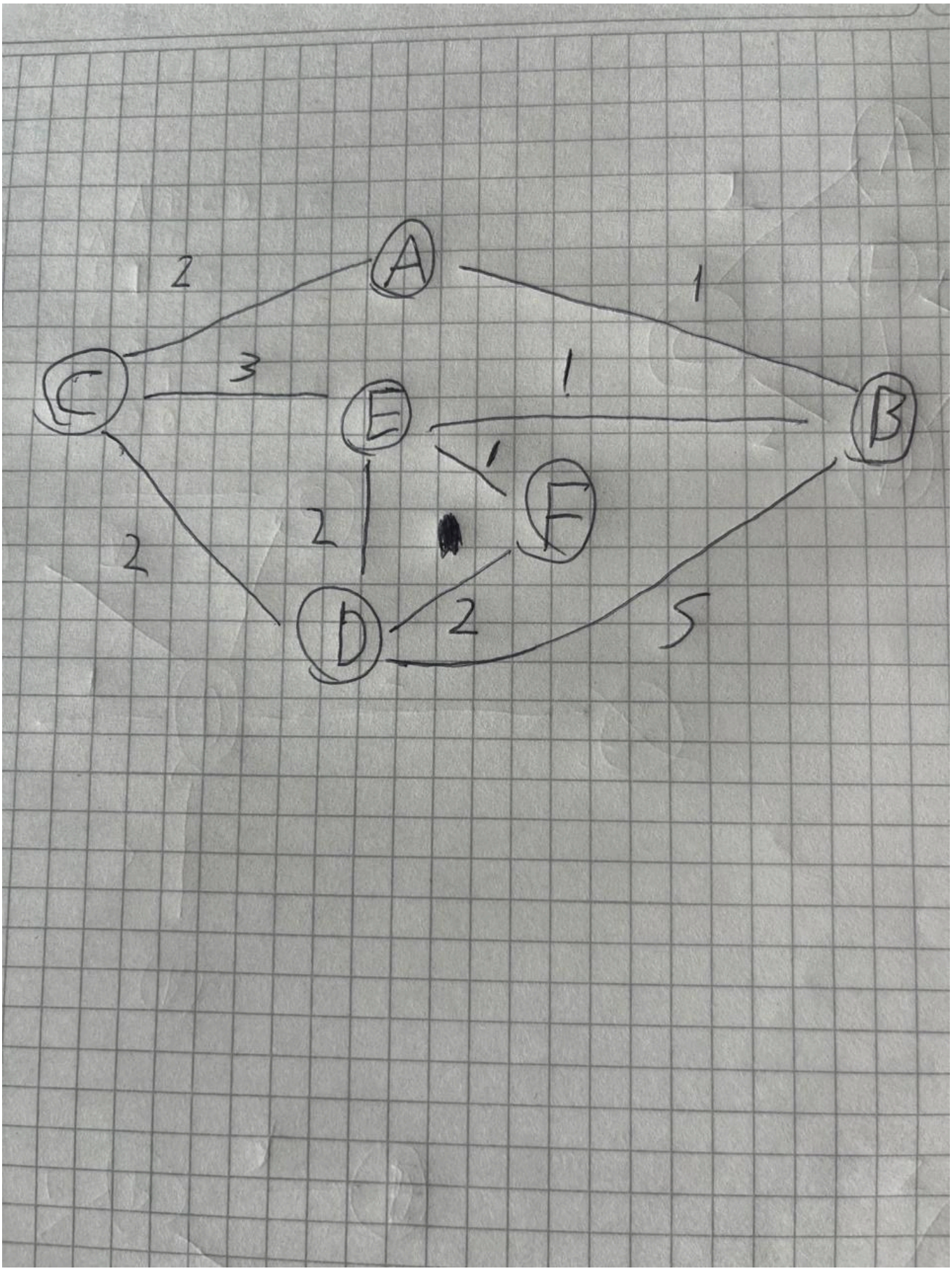
```
//santiago jose hernandez rendon

//estructuras de datos lineales

//john corredor

//pontificia universidad javeriana

//Taller algoritmo dijkstra

#include <iostream>

#include <limits>
```

```
using namespace std;



int miniDist(int distance[], bool Tset[], int MAX_INT) {

    int minimum = MAX_INT, ind = -1;



    for (int k = 0; k < 6; k++) {

        if (!Tset[k] && distance[k] <= minimum) {

            minimum = distance[k];

            ind = k;

        }

    }

    return ind;

}



void DijkstraAlgo(int graph[6][6], int src) {

    int distance[6];

    bool Tset[6];

    int MAX_INT = numeric_limits<int>::max();



    for (int k = 0; k < 6; k++) {
```

```
        distance[k] = MAX_INT;

        Tset[k] = false;

    }



    distance[src] = 0;



    for (int k = 0; k < 6; k++) {

        int m = miniDist(distance, Tset, MAX_INT);

        Tset[m] = true;



        for (int j = 0; j < 6; j++) {

      if (!Tset[j] && graph[m][j] && distance[m] != MAX_INT && distance[m] +
graph[m][j] < distance[j]) {

                distance[j] = distance[m] + graph[m][j];

            }

        }

    }



    cout << "Vertice \t\t Distancia desde la fuente al Vertice" << endl;

    for (int k = 0; k < 6; k++) {

        char str = 65 + k;
```

```
        cout << str << "\t\t\t" << distance[k] << endl;

    }

}



int main() {

    int graph[6][6] = {

        {0, 1, 2, 0, 0, 0},

        {1, 0, 0, 5, 1, 0},

        {2, 0, 0, 2, 3, 0},

        {0, 5, 2, 0, 2, 2},

        {0, 1, 3, 2, 0, 1},

        {0, 0, 0, 2, 1, 0}

    };



    DijkstraAlgo(graph, 0);

    return 0;

}
```

Matriz a grafo:


A
B
C
D
E
F
2
1
3
1
1
2
2
2
5