Republic of the Philippines

**BOHOL ISLAND STATE UNIVERSITY**

**MAIN CAMPUS**

Tel: 038-4113289 Telfax: 038-5017516 

6300 Tagbilaran City

Vision: A premier S&T university for the formation of world class and virtuous human resource for sustainable development in Bohol and the Country.

Mission: BISU is committed to provide quality higher education in the arts and sciences, as well as in the professional and technological fields;

undertake research and development, and extension services for the sustainable development of Bohol and the country.

**Project #1**

**CpE 412 –** Software Engineering

**Kolor Matching**

**using Kruskal Algorithm**

Submitted by:

Perin, Max Angelo D.

Submitted to:

Engr. Edgar Uy II

1. **Objectives and Goals**

Objective of this project is used to find the minimum spanning tree in a graph. This is a greedy algorithm that examines each edge of the graph and only keeps the connections that are the smallest while still keeping a connection to that node.

1. **Software Development Model**

Problem Definition/ Concept Exploration

Requirements Specification

Software Design

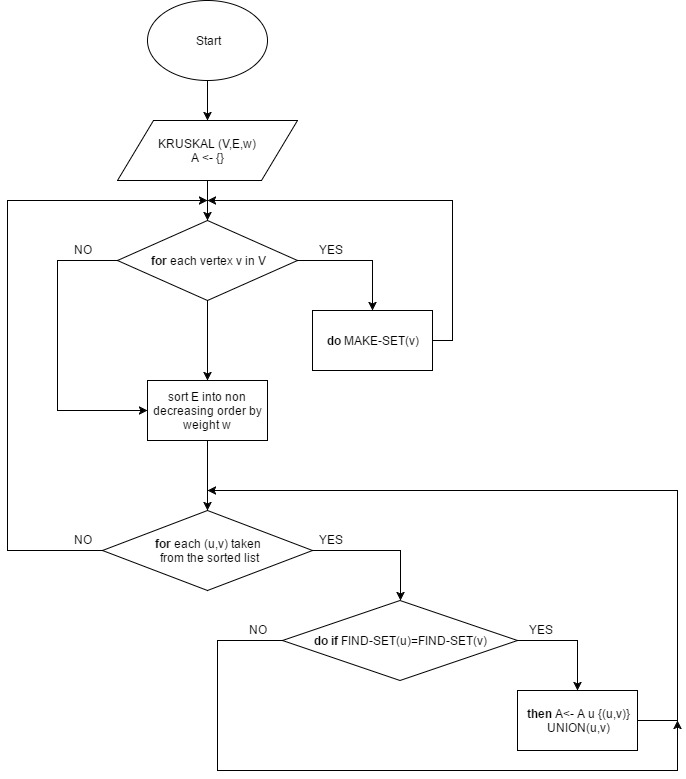
Design Implementation

Verification and Test

Software Deployment

Operation and Maintenance

1. **Flowchart**

****

1. **Algorithm**

Start with an empty set A, and select at every stage the shortest edge that has not been chosen or rejected, regardless of where this edge is situated in the graph.

KRUSKAL(V, E, w)

A ← { }           ▷ Set A will ultimately contains the edges of the MST  
**for** each vertex v in V  
    **do** MAKE-SET(v)  
sort E into nondecreasing order by weight w  
**for** each (u, v) taken from the sorted list  
    **do if** FIND-SET(u) = FIND-SET(v)  
        **then** A ← A ∪ {(u, v)}  
            UNION(u, v)  
**return** A

1. **Algorithm Analysis**

Initialize the set A:          O(1)

First for loop:                 |V| MAKE-SETs

Sort E:                          O(E lg E)

Second for loop:            O(E) FIND-SETs and UNIONs

Assuming the implementation of disjoint-set data structure, already seen in Chapter 21, that uses union by rank and path compression: O((V + E) α(V)) + O(E lg E)

Since G is connected, |E| ≥ |V| − 1⇒ O(E α(V)) + O(E lg E).

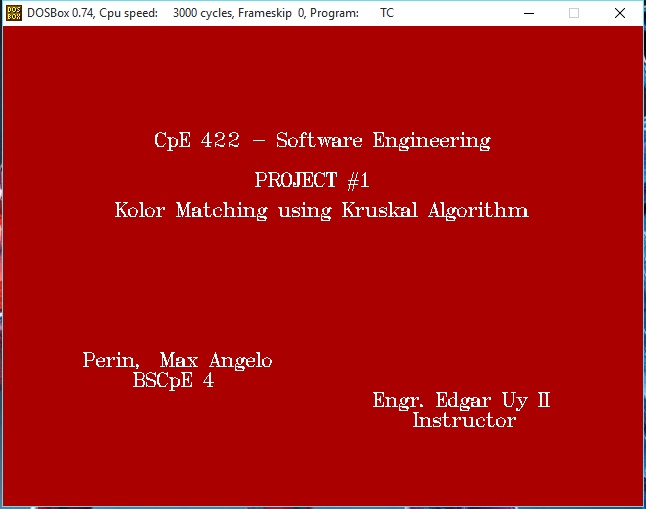
α(|V|) = O(lg V) = O(lg E).

Therefore, total time is O(E lg E).

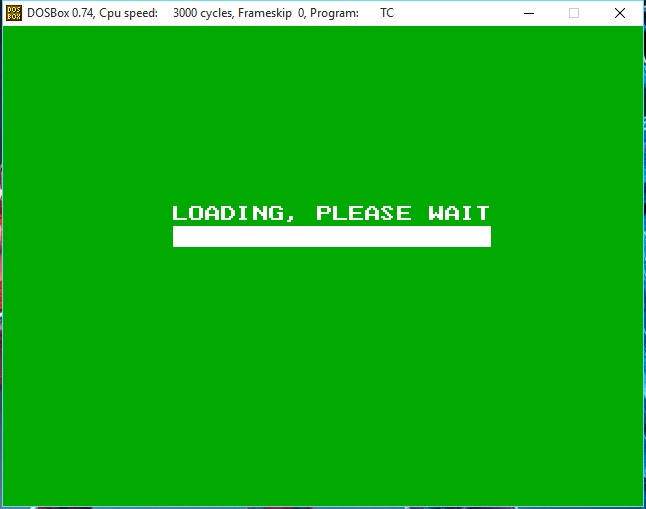
|E| ≤ |V|2 ⇒lg |E| = O(2 lg V) = O(lg V).

Therefore, O(E lg V) time. (If edges are already sorted, O(E α(V)), which is almost linear.)

1. **Snapshots of the Functionality**

****

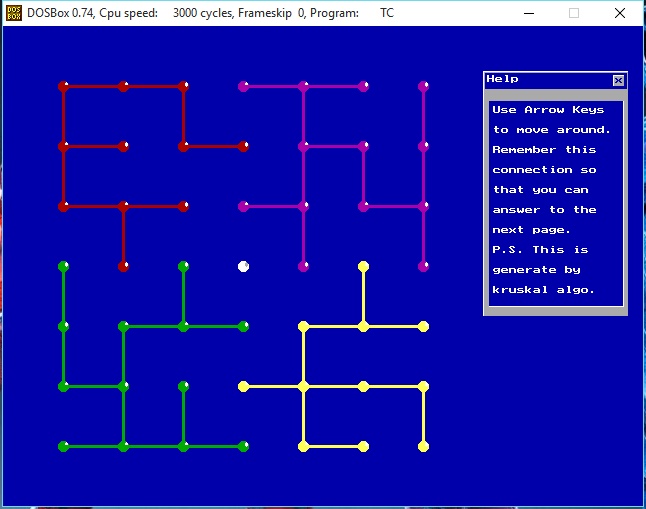
**CPE 422 – Software Engineering, Project No. 1**

****

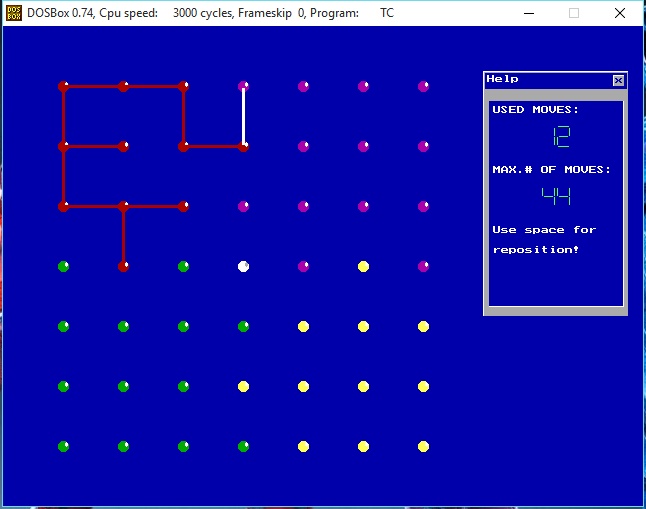
**Loading Screen.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

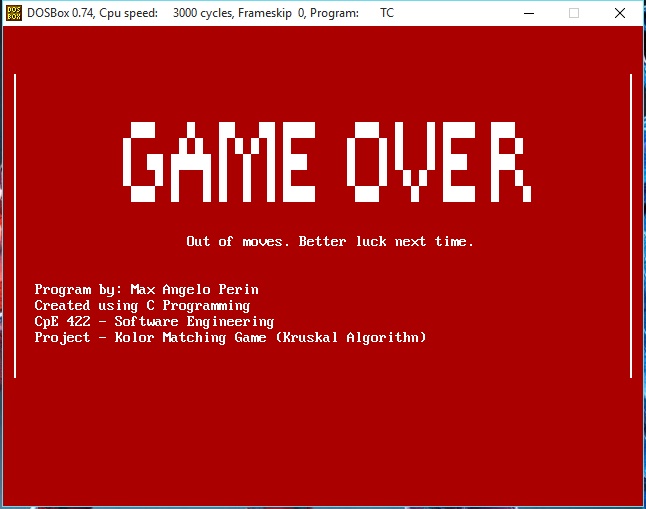
**These 49 nodes has corresponding distances from node 0 to node 49. By the help of adjacency matrix, it is easy to find the shortest path.**

****

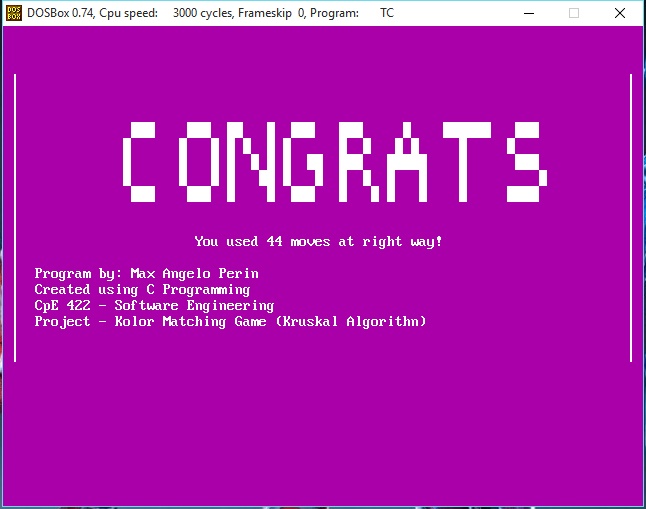
**Using kruskal algorithm, it selects the shortest paths which this connection is generate.**

****

**So I applied it into a game called “Kolor Matching” which you should memorize the kruskal algorithm generated connection and match it carefully. Use “arrow keys” to move then press “enter” if you choose that node and “space” for reposition the cursor.**

****

**A “game over” will pop out if you don’t use the moves properly.**

****

**And a “congrats” will print if you answer it perfectly!**

1. **Code**

Using MDOS Turbo C:

#include <graphics.h>

#include <stdlib.h>

#include <stdio.h>

#include <conio.h>

#include <comm.h>

#include <dos.h>

#include <time.h>

#define MAX 49

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* GLOBAL VARIABLES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

int xmax,ymax,xdum=2,ydum=2,vert;

int i,j,l,q;

int x1=60,x2=60,y1=60,y2=60;

long n;

int m;

int moves=0,cmoves=0;

int t=44;

char a;

int m;

int mnode,vnode,hnode;

float octave[7]={130.81, 146.83, 164.81, 174.61, 196, 220, 246.94};

int board[8][8]={

{ 0, 0, 0, 0, 0, 0, 0, 0},

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

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

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

{ 0, 4, 1, 4, 5, 2, 3, 2},

{ 0, 4, 4, 4, 4, 3, 3, 3},

{ 0, 4, 4, 4, 3, 3, 3, 3},

{ 0, 4, 4, 4, 4, 3, 3, 3}

};

int board2[7][7]={

{ 1, 1, 1, 4, 4, 4, 4},

{ 1, 1, 1, 1, 4, 4, 4},

{ 1, 1, 1, 4, 4, 4, 4},

{ 2, 1, 2, 0, 4, 3, 4},

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

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

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

};

int p3[14][14]={

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

};

long from[100],to[100];

/\* Adjacency Matrix 49 Nodes by 49 Nodes \*/

long G[MAX][MAX]={

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

{2,0,3,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,7,0,5,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,5,0,6,0,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,6,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,4,0,0,0,0,0,2,0,7,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,1,0,0,0,0,0,7,0,5,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,4,0,0,0,0,0,0,0,3,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,5,0,0,0,0,0,3,0,6,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,4,0,0,0,0,0,6,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,0,0,0,0,4,0,0,0,0,0,2,0,1,0,0,0,0,0,3,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,7,0,4,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,4,0,5,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,5,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

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

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,0,0,6,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,1,0,0,0,0,0,5,0,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,1,0,7,0,0,0,0,0,6,0,0,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,2,0,3,0,0,0,0,0,7,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,3,0,0,0,0,0,0,0,6,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,3,0,0,0,0,0,7,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,5,0,0,0,0,0,3,0,4,0,0,0,0,0,1,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,4,0,0,0,0,0,0,0,1,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,2,0,0,0,0,0,7,0,5,0,0,0,0,0,2,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,5,0,2,0,0,0,0,0,6,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,2,0,0,0,0,0,0,0,4},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,7,0,0,0,0,0,0,0,1,0,0,0,0,0},

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,2,0,4,0,0,0},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0},

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

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,6,0,0,0,0,0,3,0,5},

{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,4,0,0,0,0,0,5,0},

};

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* STRUCTURES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

struct pct{

int con1; int con2;

} p[15][15];

typedef struct edge{

long u,v,w;

} edge;

typedef struct edgelist{

edge data[100];

long n;

} edgelist;

edgelist elist;

edgelist spanlist;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* PROTOTYPES \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void putlinev(int i,int j,int k);

void putline(int i,int j,int k);

void putdot(int i,int j,int k);

void reposition();

void graphinit();

void drawlines();

void drawlines1();

void drawdots();

void drawall();

void intro();

void move();

void dim();

void kruskal();

long find(long belongs[],long vertexno);

void union1(long belongs[],long c1,long c2);

void sort();

void print();

void help();

void win(char \*text,int sx,int sy,int ex,int ey,int ck);

void winp(int sx,int sy,int ex,int ey,int state);

void Lcd(int x,int y,int n);

void LCD(int left,int top,int NUM);

void check();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* M A I N \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

main(){

long i,j,total\_cost;

clrscr(); n=49; kruskal(); print();

intro();

graphinit(); dim(); graphinit(); setbkcolor(BLUE); drawdots(); drawlines1(); delay(500); closegraph();

dim(); graphinit(); setbkcolor(BLUE);

drawdots(); move();

getch(); closegraph();

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* FUNCTIONS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void graphinit(){

int driver=DETECT,mode;

initgraph(&driver,&mode,"c:\\tc\\bgi");

}

void dim(){

start: clrscr();

xmax=8; ymax=8;

}

void intro(){

int x=170;

graphinit();setbkcolor(RED);

settextstyle(TRIPLEX\_FONT, HORIZ\_DIR,2);

setcolor(WHITE);

/\* FRONT \*/

outtextxy(62,100," CpE 422 - Software Engineering ");

outtextxy(62,140," PROJECT #1 ");

outtextxy(62,170," Kolor Matching using Kruskal Algorithm ");

outtextxy(70,320," Perin, Max Angelo ");

outtextxy(70,340," BSCpE 4 ");

outtextxy(70,360," Engr. Edgar Uy II ");

outtextxy(70,380," Instructor ");

getch();

clrscr();

graphinit();

setbkcolor(GREEN);

setcolor(WHITE);

settextstyle(DEFAULT\_FONT,HORIZ\_DIR,2);

outtextxy(170,180,"LOADING, PLEASE WAIT");

/\* LOADING BAR \*/

for(i=0;i<318;++i) {

sound(i\*4);

delay(10);

line(x,200,x,220);

x++;

nosound();

}

for (j=0;j<2;j++){

sound(i\*4);

delay(100);

nosound();

delay(100);

sound(i\*4);

delay(100);

nosound();

delay(100);

sound(i\*4);

delay(100);

nosound();

}

}

void check(){

for(i=0;i<=xmax+4;i++){

for(j=0;j<=ymax+4;j++){

if (p[i][j].con2){

mnode=((j/2)\*7)+(i/2)+1;

hnode=((j/2)\*7)+(i/2)+2;

for (m=0;m<spanlist.n;m++){

if(mnode==from[m] && hnode==to[m]||mnode==to[m] && hnode==from[m]){

cmoves++;

}

}

}

if(p[i][j].con1){

mnode=((j/2)\*7)+(i/2)+1;

vnode=((j/2)\*7)+(i/2)+8;

for (m=0;m<spanlist.n;m++){

if(mnode==from[m] && vnode==to[m]||mnode==to[m] && vnode==from[m]){

cmoves++;

}

}

}

}

}

}

void drawdots(){

int x,y;

x=60;y=60;

/\*cleardevice();\*/

win("Help",480,45,625,290,1);

setfillstyle(1,0);

bar(486,75,620,280);

winp(486,75,620,280,1);

outtextxy(490,80,"USED MOVES:");

outtextxy(490,100,"");

outtextxy(490,120," ");

outtextxy(490,140,"MAX.# OF MOVES:");

outtextxy(490,160,"");

outtextxy(490,180," ");

outtextxy(490,200,"Use space for");

outtextxy(490,220,"reposition!");

outtextxy(490,240,"");

Lcd(540,100,moves);

Lcd(540,160,t);

for(i=1;i<xmax;i++){

for(j=1;j<ymax;j++){

if(board[j][i]!=-1){

switch(board[j][i]){

case 1:putdot(x,y,RED);break;

case 2:putdot(x,y,GREEN);break;

case 3:putdot(x,y,YELLOW);break;

case 4:putdot(x,y,5);break; /\*5=PINK\*/

case 5:putdot(x,y,15);break;

}

}

x+=60;

}

x=60;

y+=60;

}

}

void putdot(int i,int j,int k){

setcolor(k);

setfillstyle(1,k);

setcolor(k);

fillellipse(i,j,5,5);

if(k!=0){

if(k==15){setcolor(7);setfillstyle(1,7);}

else {setfillstyle(1,15);setcolor(15);}

fillellipse(i+3,j-2,1,2);

}

}

void putline(int i,int j,int k){

setcolor(k);

line(i\*30+x1+1,j\*30+y1-1,i\*30+x1+58,j\*30+y1-1);

line(i\*30+x1+1,j\*30+y1,i\*30+x1+58,j\*30+y1);

line(i\*30+x1+1,j\*30+y1+1,i\*30+x1+58,j\*30+y1+1);

}

void putlinev(int i,int j,int k){

setcolor(k);

line(i\*30+x2-1,j\*30+y2+2,i\*30+x2-1,j\*30+y2+58);

line(i\*30+x2,j\*30+y2+2,i\*30+x2,j\*30+y2+58);

line(i\*30+x2+1,j\*30+y2+2,i\*30+x2+1,j\*30+y2+58);

}

void drawlines(){

mnode=0;

for(i=0;i<=xmax+4;i++){

for(j=0;j<=ymax+4;j++){

if (p[i][j].con2){

mnode=((j/2)\*7)+(i/2)+1;

hnode=((j/2)\*7)+(i/2)+2;

for (m=0;m<spanlist.n;m++){

if(mnode==from[m] && hnode==to[m]||mnode==to[m] && hnode==from[m]){

switch(board2[(j/2)][(i/2)]){

case 1:putline(i,j,RED);break;

case 2:putline(i,j,GREEN);break;

case 3:putline(i,j,YELLOW);break;

case 4:putline(i,j,5);break; /\*5=PINK\*/

}

}

}

}

if(p[i][j].con1){

mnode=((j/2)\*7)+(i/2)+1;

vnode=((j/2)\*7)+(i/2)+8;

for (m=0;m<spanlist.n;m++){

if(mnode==from[m] && vnode==to[m]||mnode==to[m] && vnode==from[m]){

switch(board2[(j/2)][(i/2)]){

case 1:putlinev(i,j,RED);break;

case 2:putlinev(i,j,GREEN);break;

case 3:putlinev(i,j,YELLOW);break;

case 4:putlinev(i,j,5);break; /\*5=PINK\*/

}

}

}

}

}

}

}

void reposition(){

if(xdum>=xmax && !vert) xdum=xdum-2;

if(ydum>=ymax && vert) ydum=ydum-2;

}

void drawall(){

drawdots(); drawlines();

}

void move(){

while(a!=ESC){

begin:

a=getch();

if (moves==44){

check();

if (cmoves!=44){

graphinit();

setbkcolor(RED);

printf(" \n\n\n ");

printf("³ ³\n ");

printf("³ ³\n ");

printf("³ ³\n ");

printf("³ ÛÛÛ Û ÛÛ ÛÛ ÛÛÛÛ ÛÛÛ Û Û ÛÛÛÛ ÛÛÛ ³\n ");

printf("³ Û Û Û Û Û Û Û Û Û Û Û Û Û Û Û ³\n ");

printf("³ Û ÛÛ ÛÛÛÛÛ Û Û Û ÛÛÛ Û Û Û Û ÛÛÛ ÛÛÛ ³\n ");

printf("³ Û Û Û Û Û Û Û Û Û Û Û Û Û Û Û ³\n ");

printf("³ ÛÛÛ Û Û Û Û Û ÛÛÛÛ ÛÛÛ Û ÛÛÛÛ Û Û ³\n ");

printf("³ ³\n ");

printf("³ ³\n ");

printf("³ Out of moves. Better luck next time. ³\n ");

printf("³ ³\n ");

printf("³ ³\n ");

printf("³ Program by: Max Angelo Perin ³\n ");

printf("³ Created using C Programming ³\n ");

printf("³ CpE 422 - Software Engineering ³\n ");

printf("³ Project - Kolor Matching Game (Kruskal Algorithn) ³\n ");

printf("³ ³\n ");

printf("³ ³\n ");

for ( i = 4 ; i >= 0 ; i-- ){

sound ( octave[i] \* 4 ) ;

delay ( 200 ) ;

}

nosound() ;

getch();

break;

}

else if (cmoves==44){

graphinit();

setbkcolor(5);

printf(" \n\n\n ");

printf("³ ³\n ");

printf("³ ³\n ");

printf("³ ³\n ");

printf("³ ÛÛÛ ÛÛÛ ÛÛ Û ÛÛÛ ÛÛÛ Û ÛÛÛÛÛÛ ÛÛÛ ³\n ");

printf("³ Û Û Û Û Û Û Û Û Û Û Û Û Û ³\n ");

printf("³ Û Û Û Û Û Û Û ÛÛ ÛÛÛ ÛÛÛÛÛ Û ÛÛÛ ³\n ");

printf("³ Û Û Û Û ÛÛ Û Û Û Û Û Û Û Û ³\n ");

printf("³ ÛÛÛ ÛÛÛ Û Û ÛÛÛ Û Û Û Û Û ÛÛÛÛ ³\n ");

printf("³ ³\n ");

printf("³ ³\n ");

printf("³ You used 44 moves at right way! ³\n ");

printf("³ ³\n ");

printf("³ Program by: Max Angelo Perin ³\n ");

printf("³ Created using C Programming ³\n ");

printf("³ CpE 422 - Software Engineering ³\n ");

printf("³ Project - Kolor Matching Game (Kruskal Algorithn) ³\n ");

printf("³ ³\n ");

printf("³ ³\n ");

for ( i = 0 ; i < 7 ; i++ ){

sound ( octave[i] \* 8 ) ;

delay ( 80 ) ;

}

nosound() ;

getch();

break;

}

}

if(!vert){

putline(xdum,ydum,BLUE);

if(a==RIGHT && xdum+1<xmax+2)

{ drawall(); xdum=xdum+2; putline(xdum,ydum,15); sound(700); delay(100); nosound();}

if(a==LEFT && xdum-1>=1)

{ drawall(); xdum=xdum-2; putline(xdum,ydum,15); sound(700); delay(100); nosound(); }

if(a==UP && ydum-1>=1)

{ drawall(); ydum=ydum-2; putline(xdum,ydum,15); sound(700); delay(100); nosound(); }

if(a==DOWN && ydum+1<=ymax+3)

{ drawall(); ydum=ydum+2; putline(xdum,ydum,15); sound(700); delay(100); nosound();}

if(a==ENTER && !p[xdum][ydum].con2) { moves++; sound(900); delay(100); nosound( ); p[xdum][ydum].con2=1;

drawall(); }

if(a==SPACE) { sound(700); delay(100); nosound(); if(vert) vert=0; else vert=1; reposition();

drawall(); putlinev(xdum,ydum,15); goto begin;}

}

if(vert){

putlinev(xdum,ydum,BLUE);

if(a==RIGHT && xdum<xmax+3)

{ drawall(); xdum=xdum+2; putlinev(xdum,ydum,15); sound(700); delay(100); nosound(); }

if(a==LEFT && xdum-1>=1)

{ drawall(); xdum=xdum-2; putlinev(xdum,ydum,15); sound(700); delay(100); nosound(); }

if(a==UP && ydum-1>=1)

{ drawall(); ydum=ydum-2; putlinev(xdum,ydum,15); sound(700); delay(100); nosound();}

if(a==DOWN && ydum+1<ymax+2)

{ drawall(); ydum=ydum+2; putlinev(xdum,ydum,15); sound(700); delay(100); nosound();}

if(a==ENTER && !p[xdum][ydum].con1) { moves++; sound(900); delay(100); nosound( ); p[xdum][ydum].con1=1;

drawall(); }

if(a==SPACE) { sound(700); delay(100); nosound( ); if(vert) vert=0; else vert=1; reposition();

drawall(); putline(xdum,ydum,15); goto begin; }

}

}

}

void drawlines1(){

int m;

int mnode,vnode,hnode;

mnode=0;

win("Help",480,45,625,290,1);

setfillstyle(1,0);

bar(486,75,620,280);

winp(486,75,620,280,1);

outtextxy(490,80,"Use Arrow Keys");

outtextxy(490,100,"to move around.");

outtextxy(490,120,"Remember this");

outtextxy(490,140,"connection so");

outtextxy(490,160,"that you can");

outtextxy(490,180,"answer to the");

outtextxy(490,200,"next page.");

outtextxy(490,220,"P.S. This is");

outtextxy(490,240,"generate by");

outtextxy(490,260,"kruskal algo.");

for(i=0;i<xmax+6;i++){

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

mnode=((j/2)\*7)+(i/2)+1;

hnode=((j/2)\*7)+(i/2)+2;

vnode=((j/2)\*7)+(i/2)+8;

m=random (7);

sound(octave[m]\*4);

delay(100);

nosound();

for (m=0;m<spanlist.n;m++){

if(p3[i][j]==2){

if(mnode==from[m] && hnode==to[m]||mnode==to[m] && hnode==from[m]){

switch(board2[j/2][i/2]){

case 1:putline(i,j,RED);break;

case 2:putline(i,j,GREEN);break;

case 3:putline(i,j,YELLOW);break;

case 4:putline(i,j,5);break; /\*5=PINK\*/

}

}

else if(mnode==from[m] && vnode==to[m]||mnode==to[m] && vnode==from[m]){

switch(board2[j/2][i/2]){

case 1:putlinev(i,j,RED);break;

case 2:putlinev(i,j,GREEN);break;

case 3:putlinev(i,j,YELLOW);break;

case 4:putlinev(i,j,5);break; /\*5=PINK\*/

}

}

}

}

}

}

}

void win(char \*text,int sx,int sy,int ex,int ey,int ck){

setfillstyle(1,7);

bar(sx,sy,ex,ey);

setfillstyle(1,7);

setcolor(15);

line(sx,sy,sx,ey);

line(sx,sy,ex,sy);

line(sx,sy,sx,ey);

line(sx,sy,ex,sy);

setcolor(0);

line(ex,sy,ex,ey);

line(ex,sy,ex,ey);

line(sx,ey,ex,ey);

line(sx,ey,ex,ey);

if(ck==1){

settextstyle(0,0,0);

setfillstyle(1,1);

bar(sx+2,sy+2,ex-2,sy+17);

setcolor(15);

outtextxy(sx+4,sy+4,text);

setfillstyle(1,7);

bar(ex-15,sy+4,ex-4,sy+15);

setcolor(15);

line(ex-15,sy+4,ex-4,sy+4);

line(ex-15,sy+4,ex-15,sy+15);

setcolor(0);

line(ex-15,sy+15,ex-4,sy+15);

line(ex-4,sy+4,ex-4,sy+15);

setcolor(1);

outtextxy(ex-13,sy+5,"x");

setfillstyle(1,7);

}

}

void winp(int sx,int sy,int ex,int ey,int state){

if(state==1){

setcolor(0);

line(sx,sy,sx,ey);

line(sx,sy,ex,sy);

line(sx,sy,sx,ey);

line(sx,sy,ex,sy);

setcolor(15);

line(ex,sy,ex,ey);

line(ex,sy,ex,ey);

line(sx,ey,ex,ey);

line(sx,ey,ex,ey);

}

else {

setcolor(15);

line(sx,sy,sx,ey);

line(sx,sy,ex,sy);

line(sx,sy,sx,ey);

line(sx,sy,ex,sy);

setcolor(0);

line(ex,sy,ex,ey);

line(ex,sy,ex,ey);

line(sx,ey,ex,ey);

line(sx,ey,ex,ey);

}

}

void Lcd(int x,int y,int n){

int a,b;

if(n<10){

LCD(x,y,0);

LCD(x+15,y,n);

}

else if(n<100){

a=n/10;b=n%10;

LCD(x,y,a);

LCD(x+15,y,b);

}

}

void LCD(int left,int top,int NUM){

int i;

setcolor(10);

switch(NUM){

case 1:

line(left+11,top+2,left+11,top+9);

line(left+11,top+11,left+11,top+18);

break;

case 2:

line(left,top,left+10,top);

line(left+11,top+2,left+11,top+9);

line(left+1,top+10,left+9,top+10);

line(left-1,top+11,left-1,top+18);

line(left,top+20,left+10,top+20);

break;

case 3:

line(left,top,left+10,top);

line(left+11,top+2,left+11,top+9);

line(left+1,top+10,left+9,top+10);

line(left+11,top+11,left+11,top+18);

line(left,top+20,left+10,top+20);

break;

case 4:

line(left-1,top+2,left-1,top+9);

line(left+11,top+2,left+11,top+9);

line(left+1,top+10,left+9,top+10);

line(left+11,top+11,left+11,top+18);

break;

case 5:

line(left,top,left+10,top);

line(left-1,top+2,left-1,top+9);

line(left+1,top+10,left+9,top+10);

line(left+11,top+11,left+11,top+18);

line(left,top+20,left+10,top+20);

break;

case 6:

line(left,top,left+10,top);

line(left-1,top+2,left-1,top+9);

line(left+1,top+10,left+9,top+10);

line(left-1,top+11,left-1,top+18);

line(left+11,top+11,left+11,top+18);

line(left,top+20,left+10,top+20);

break;

case 7:

line(left,top,left+10,top);

line(left-1,top+2,left-1,top+9);

line(left+11,top+2,left+11,top+9);

line(left+11,top+11,left+11,top+18);

break;

case 8:

line(left,top,left+10,top);

line(left-1,top+2,left-1,top+9);

line(left+11,top+2,left+11,top+9);

line(left+1,top+10,left+9,top+10);

line(left-1,top+11,left-1,top+18);

line(left+11,top+11,left+11,top+18);

line(left,top+20,left+10,top+20);

break;

case 9:

line(left,top,left+10,top);

line(left-1,top+2,left-1,top+9);

line(left+11,top+2,left+11,top+9);

line(left+1,top+10,left+9,top+10);

line(left+11,top+11,left+11,top+18);

line(left,top+20,left+10,top+20);

break;

case 0:

line(left,top,left+10,top);

line(left-1,top+2,left-1,top+9);

line(left+11,top+2,left+11,top+9);

line(left-1,top+11,left-1,top+18);

line(left+11,top+11,left+11,top+18);

line(left,top+20,left+10,top+20);

break;

}

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* KRUSKAL ALGORITHM HERE \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

void kruskal(){

long belongs[49],i,j,cno1,cno2;

elist.n=0;

for(i=1;i<n;i++){

for(j=0;j<i;j++){

if(G[i][j]!=0){

elist.data[elist.n].u=i;

elist.data[elist.n].v=j;

elist.data[elist.n].w=G[i][j];

elist.n++;

}

}

}

sort();

for(i=0;i<n;i++){

belongs[i]=i;

}

spanlist.n=0;

for(i=0;i<elist.n;i++){

cno1=find(belongs,elist.data[i].u);

cno2=find(belongs,elist.data[i].v);

if(cno1!=cno2){

spanlist.data[spanlist.n]=elist.data[i];

spanlist.n=spanlist.n+1;

union1(belongs,cno1,cno2);

}

}

}

long find(long belongs[],long vertexno){

return(belongs[vertexno]);

}

void union1(long belongs[],long c1,long c2){

long i;

for(i=0;i<n;i++){

if(belongs[i]==c2){

belongs[i]=c1;

}

}

}

void sort(){

long i,j;

edge temp;

for(i=1;i<elist.n;i++){

for(j=0;j<elist.n-1;j++){

if(elist.data[j].w>elist.data[j+1].w){

temp=elist.data[j];

elist.data[j]=elist.data[j+1];

elist.data[j+1]=temp;

}

}

}

}

void print(){

long i,cost=0;

for(i=0;i<spanlist.n;i++){

/\*printf("\n%ld\t%ld\t%ld",spanlist.data[i].u+1,spanlist.data[i].v+1,spanlist.data[i].w);\*/

from[i]=spanlist.data[i].u+1;

to[i]=spanlist.data[i].v+1;

/\*printf("\n%ld %ld\n",from[i],to[i]);\*/

cost=cost+spanlist.data[i].w;

}

/\*printf("\n\nCost of the spanning tree=%ld",cost);\*/

}