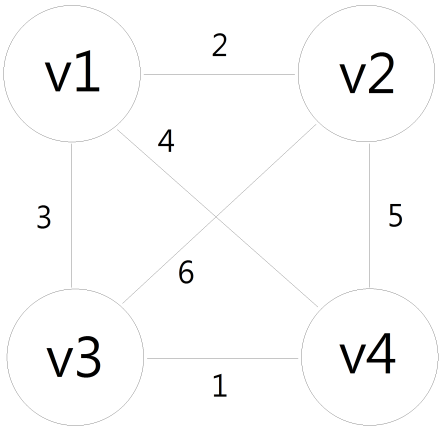
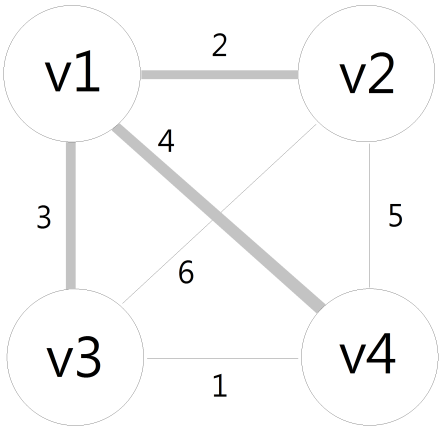
四電子二B 曾皓暐 B10213147 Homework#2

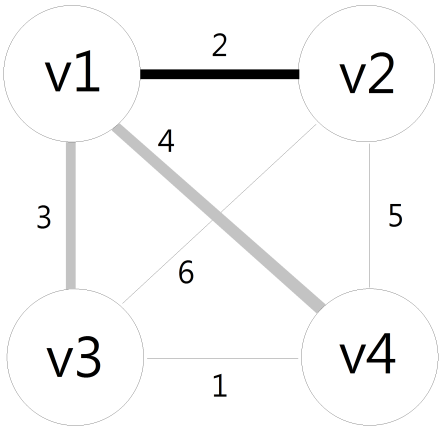
Graph 1:



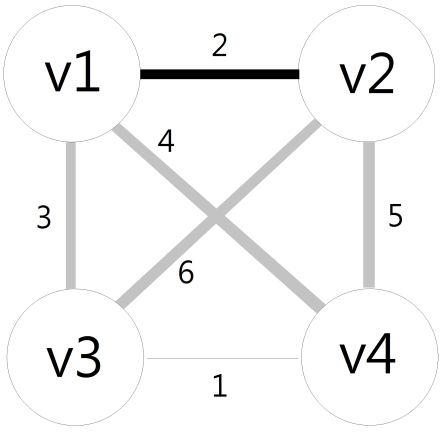
Step 1:



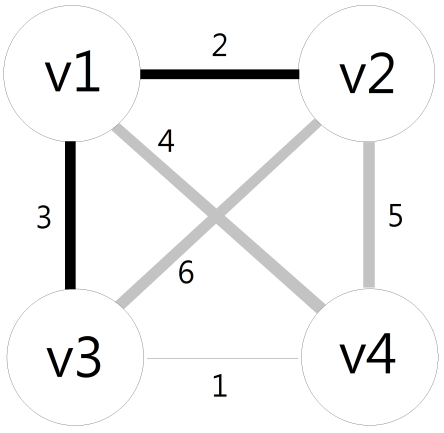
Step 2:



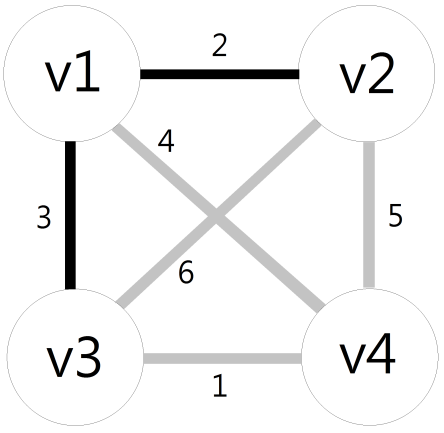
Step 3:



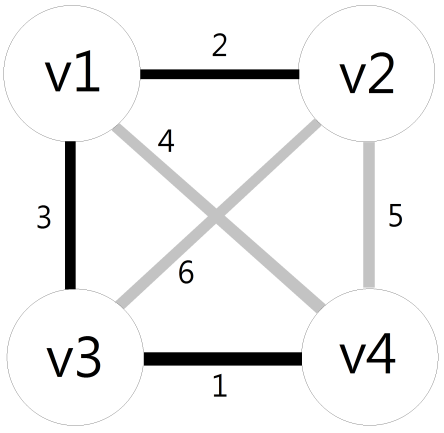
Step 4:



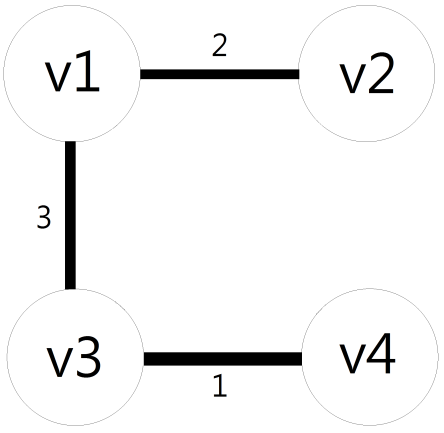
Step 5:



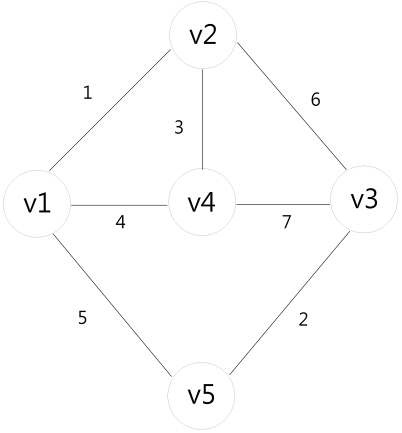
Step 6:

 We have gotten enough edges(n-1=3), so don’t have to search anymore.

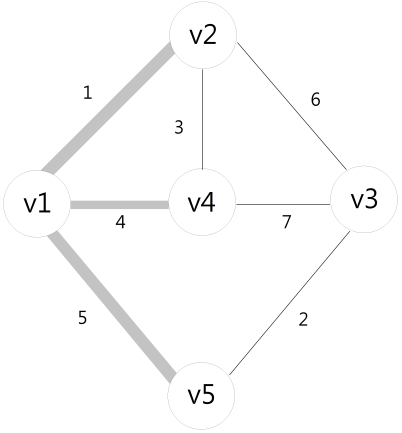
Result:

 The weight of the MST is 6.

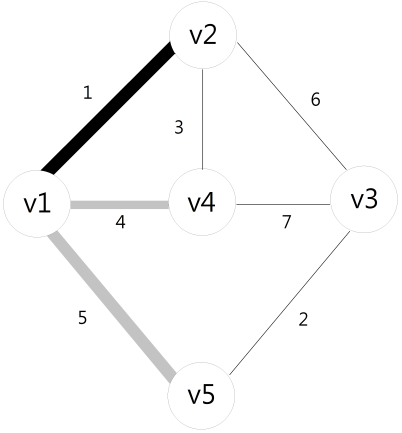
Graph 2:



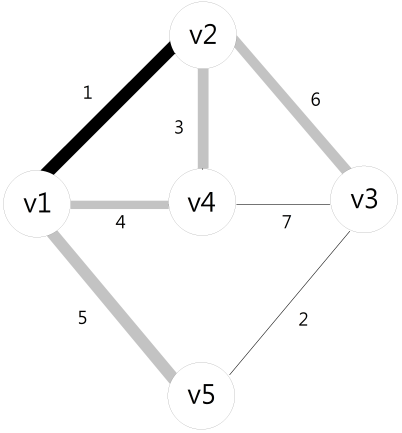
Step 1:



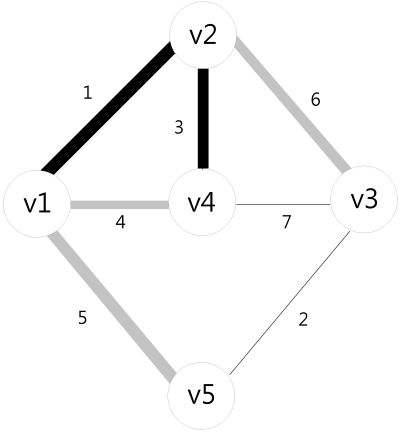
Step 2:



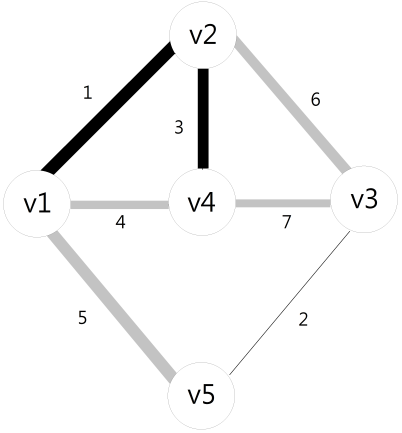
Step 3:



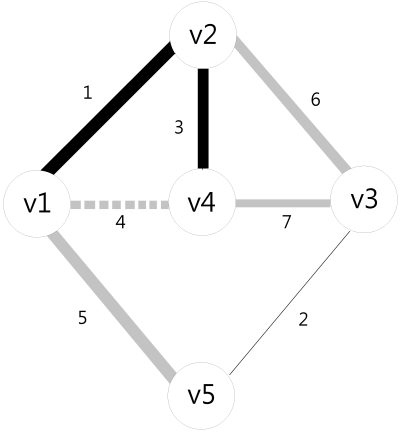
Step 4:



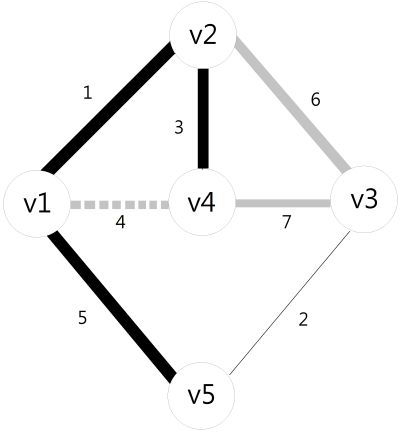
Step 5:



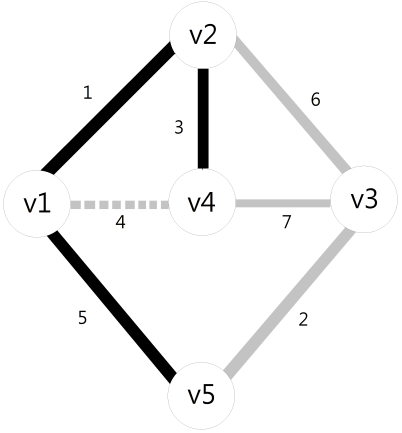
Step 6:

 Edge v1-v4 is the smallest weight we’ve searched. But it will generate a loop, cannot choose it.

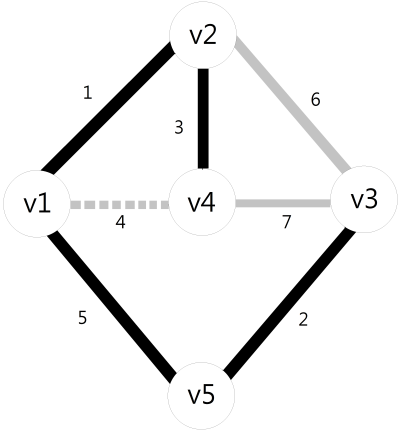
Step 7:



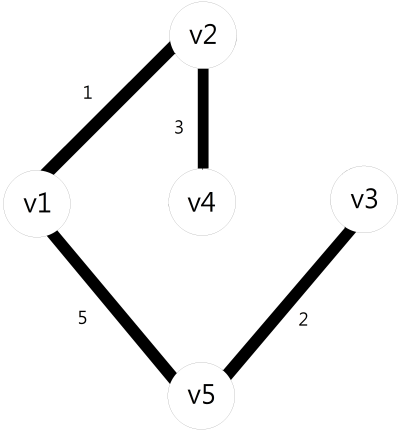
Step 8:



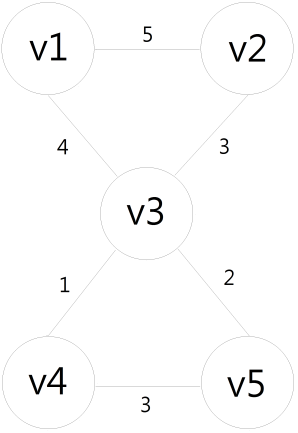
Step 9:

 We have gotten enough edges(n-1=4), so don’t have to search anymore.

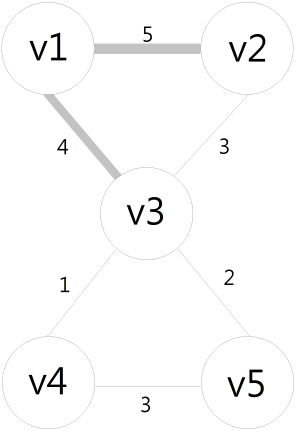
Result:

 The weight of the MST is 11.

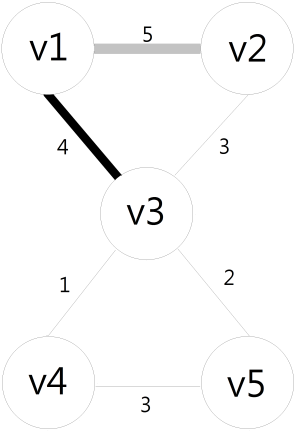
Graph 3:



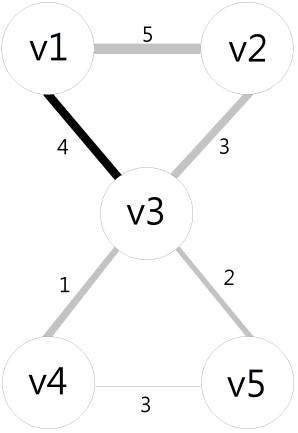
Step 1:



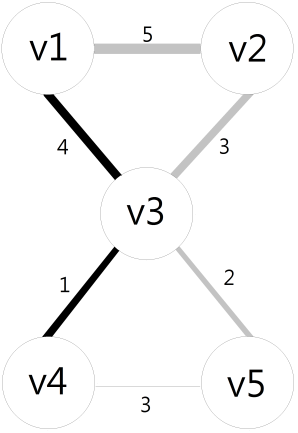
Step 2:



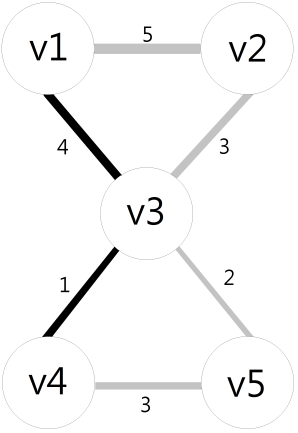
Step 3:



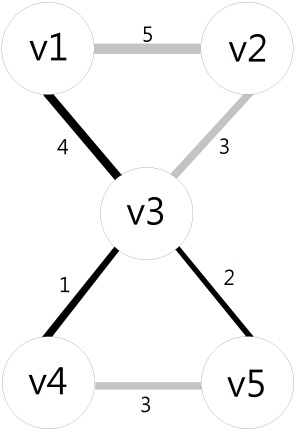
Step 4:



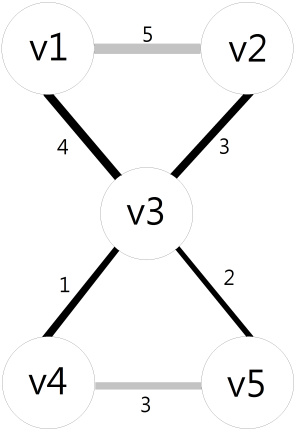
Step 5:



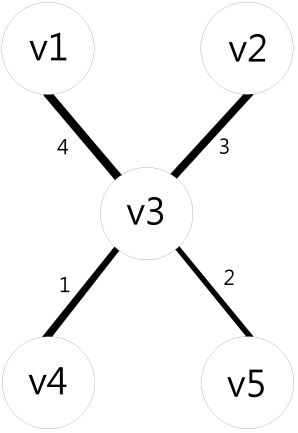
Step 6:



Step 7:

 Edge v2-v3 and v4-v5 have the same weight. Pick one of them whatever you want. Now, we have gotten enough edges(n-1=4), so don’t have to search anymore.

Result:

 The weight of the MST is 10.

Program code

#include <stdio.h>

#include <stdlib.h>

#define Empty -1

#define Size 50

const int ASCII0=48;

int Table[Size][Size],Subtree[Size][3],Searched\_Edges[Size][2];

int Selected\_Edges[Size][3];

int TVN,SEC=0,n\_Edge;

int main(int argc, char \*argv[]) {

int i,n;

open\_file();

modify\_Table();

TVN=Table[0][1]; //total vertex number

n\_Edge=0;

find\_Subtree(1);

while(n\_Edge<TVN-1){

n=choose\_Edge();

n\_Edge++;

if(n\_Edge<TVN-1) find\_Subtree(n);

else break;

}

show\_Result();

system("pause");

return 0;

}

void open\_file(void) {

FILE \*fp;

int i,count=0;

char fname[50],line[Size];

/\*Enter a file name\*/

printf("Enter a file:");

gets(fname);

/\*convert txt to array\*/

fp = fopen(fname, "r");

if(fp!=NULL) {

printf("Opening file:[%s]\n", fname);

while(fgets(line,Size,fp)!=NULL) {

printf("%s",line);

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

Table[count][i]=line[i];

}

count++;

}

fclose(fp);

}

else{

printf("Open [%s] file fail.", fname);

exit(1);

}

printf("\n");

}

void modify\_Table(void){

int i,j,n,sum=0;

/\*Convert ASCII to integer and remove the control characters\*/

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

n=1;

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

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

if(Table[i][j]>=ASCII0 && Table[i][j]<=ASCII0+9)

sum=sum\*10+Table[i][j]-ASCII0; //For the value is bigger than 10

else{

Table[i][n]=sum;

sum=0;

n++;

}

}

else break;

}

for(;n<Size;n++)Table[i][n]=Empty; //Fill "Empty" to unused space

}

}

void find\_Subtree(int v1){

int i,v2,weight,bottom;

int j,min,i\_min,temp;

char flag;

bottom=Size-1;

/\*find edges\*/

for(v2=1;v2<=TVN;v2++){

weight=Table[v1][v2];

if(weight!=0 && weight!=Empty){

/\*check the edge has been searched or not\*/

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

if((v1==Searched\_Edges[i][0] && v2==Searched\_Edges[i][1])

|| (v1==Searched\_Edges[i][1] && v2==Searched\_Edges[i][0])){

flag=0;

break;

}

else flag=1;

}

while (flag==1){

Subtree[bottom][0]=v1;

Subtree[bottom][1]=v2;

Subtree[bottom][2]=weight;

bottom--;

Searched\_Edges[SEC][0]=v1;

Searched\_Edges[SEC][1]=v2;

SEC++;

flag=0;

}

}

}

/\*sort the Subtree\*/

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

if(bottom==Size-1) break;

while(Subtree[i][0]==0 && bottom!=Size-1){

/\*move 0 down to bottom\*/

bottom++;

Subtree[i][0]=Subtree[bottom][0];

Subtree[i][1]=Subtree[bottom][1];

Subtree[i][2]=Subtree[bottom][2];

Subtree[bottom][0]=0;

Subtree[bottom][1]=0;

Subtree[bottom][2]=0;

}

}

if(Subtree[0][0]==0){

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

Subtree[j-1][0]=Subtree[j][0];

Subtree[j-1][1]=Subtree[j][1];

Subtree[j-1][2]=Subtree[j][2];

}

}

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

/\*comparision\*/

min=Subtree[i][2];

for(j=i+1;Subtree[j][0]>0;j++)

while(Subtree[j][2]<min){

min=Subtree[j][2];

i\_min=j;

flag=1;

}

while(flag==1){

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

/\*swap\*/

temp=Subtree[i][j];

Subtree[i][j]=Subtree[i\_min][j];

Subtree[i\_min][j]=temp;

}

flag=0;

}

}

}

int choose\_Edge(void){

int i,j,v2;

char flag=0;

i=0;

do{

v2=Subtree[i][1];

flag=loop\_Check(v2);

if(flag==0) break;

else {

printf("v%d-v%d(%d) will create a loop. Ignore it.\n",

Subtree[i][0],Subtree[i][1],Subtree[i][2]);

for(j=0;j<=2;j++) Subtree[i][j]=0;

}

i++;

}while(flag!=0);

for(j=0;j<=2;j++) Selected\_Edges[n\_Edge][j]=Subtree[i][j];

v2=Selected\_Edges[n\_Edge][1];

for(j=0;j<=2;j++) Subtree[i][j]=0;

return v2;

}

int loop\_Check(int v2){

int i,j; char flag=0;

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

if(v2==Selected\_Edges[i][1]){

flag=1;

break;

}

else flag=0;

}

return flag;

}

void show\_Result(void) {

int i,sum=0;

for(i=0;i<n\_Edge;i++)sum=sum+Selected\_Edges[i][2];

printf("MST has found!\n");

printf("The final path is\n");

for(i=0;i<n\_Edge;i++) printf("v%d-v%d(%d) -> ",Selected\_Edges[i][0],

Selected\_Edges[i][1],Selected\_Edges[i][2]);

printf("The weight of the MST is %d\n",sum);

}

