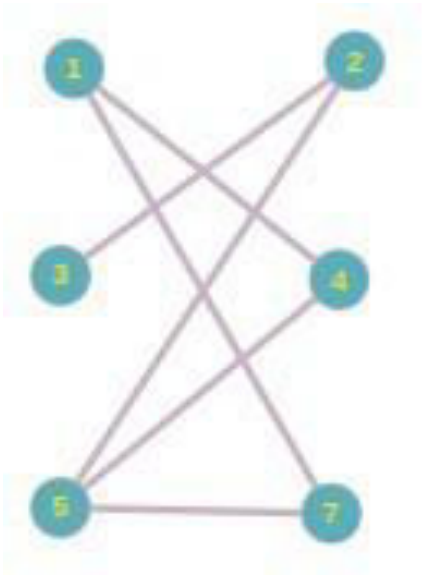
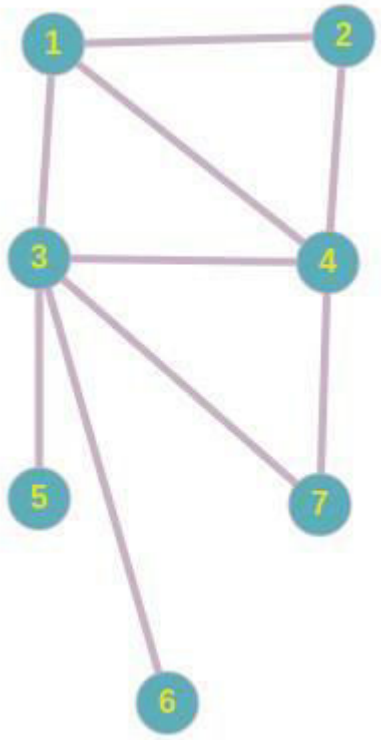
1.

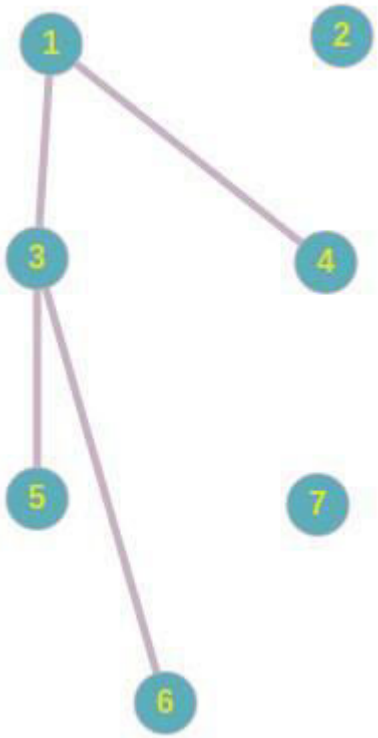
1)



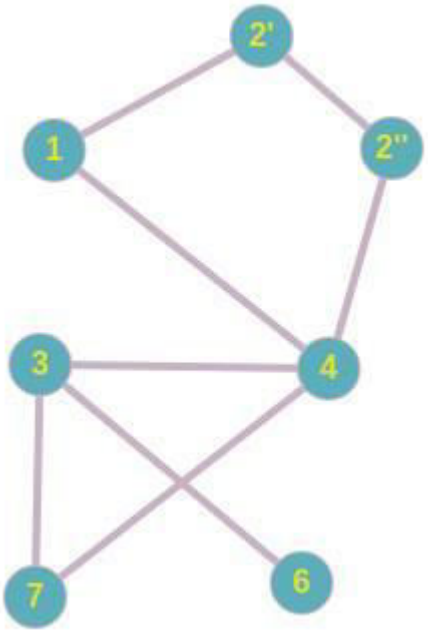
2)



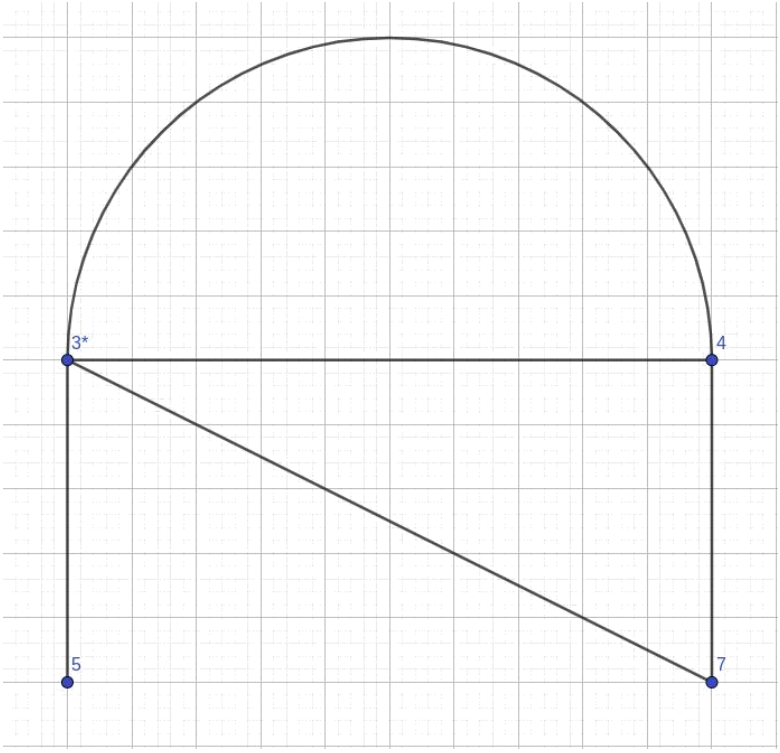
3)



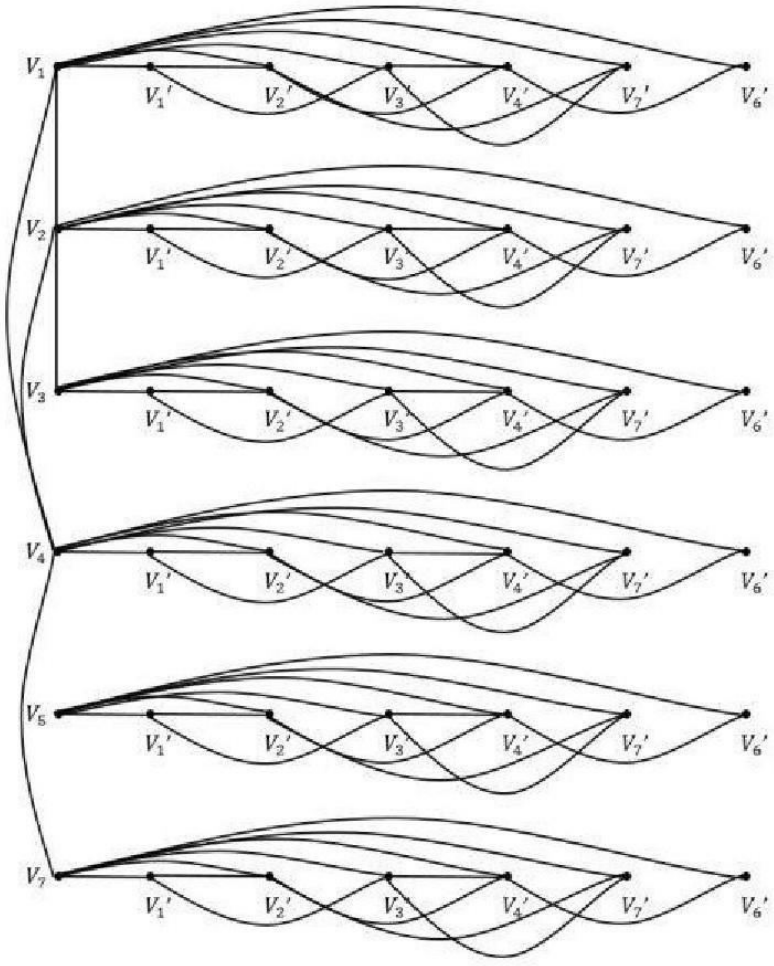
4)



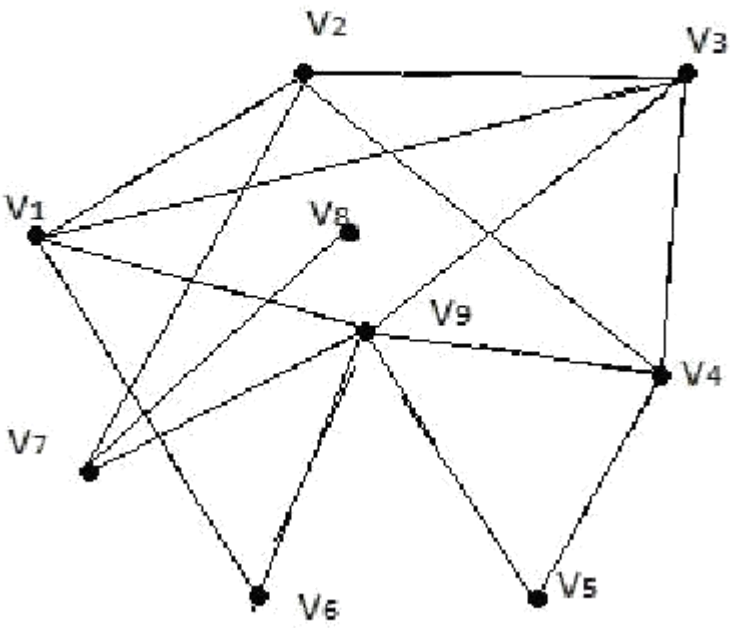
5)



6)



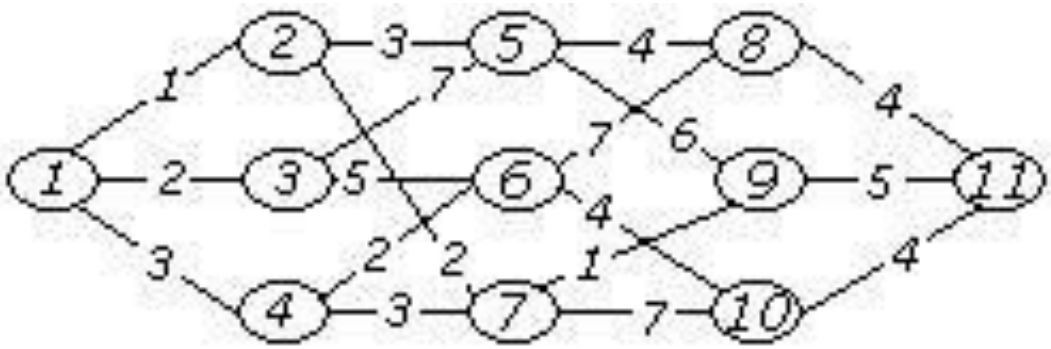
2.



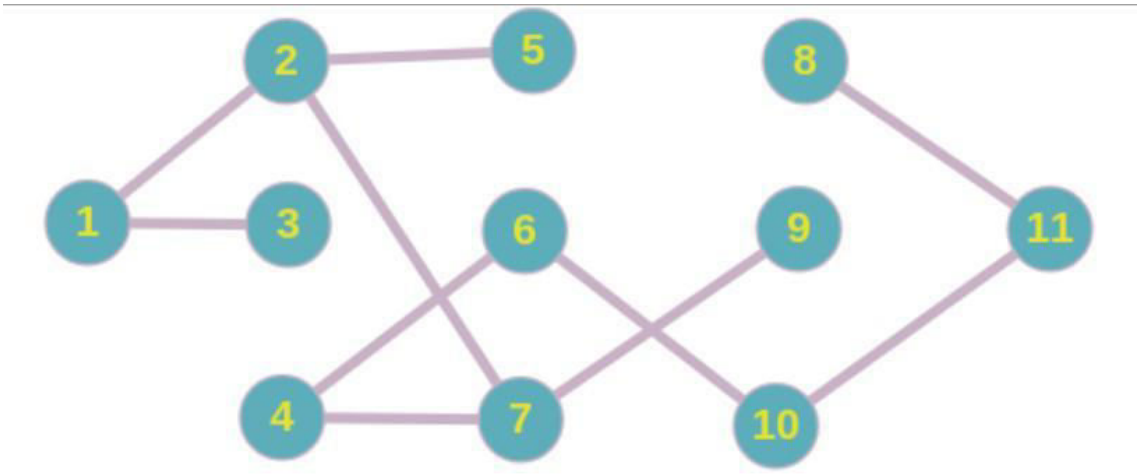
	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	0	1	1	0	0	1	0	0	1
V2	1	0	1	1	0	0	1	0	0
V3	1	1	0	1	0	0	0	0	1
V4	0	1	1	0	1	0	0	0	1
V5	0	0	0	1	0	0	0	0	1
V6	1	0	0	0	0	0	0	0	1
V7	0	1	0	0	0	0	0	1	1
V8	0	0	0	0	0	0	1	0	0
V9	1	0	1	1	1	1	1	0	0

Діаметр: 3

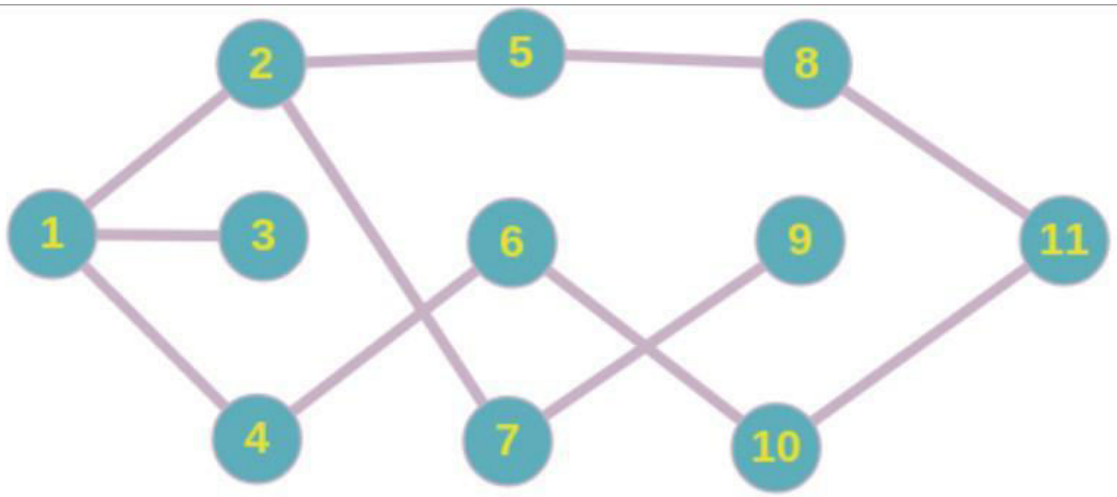
3.



Краскала:



Прима:




```
#include <stdio.h>

int makeTrees(int n, int A[n][n]);
void removeRepeated(int n, int A[n][n]);
int areInDifferentTrees(int n, int A[n][n], int first, int second);
void addToTree(int n, int A[n][n], int first, int second);

int main()
{
   // the adjecency matrix of our graph (with weight)
   //         1  2  3  4  5  6  7  8  9  10 11
```

```
int A[11][11] = {
    /*1*/ { 0, 1, 2, 4, 0, 0, 0, 0, 0, 0, 0 },
    /*2*/ { 1, 0, 0, 0, 3, 0, 2, 0, 0, 0, 0 },
    /*3*/ { 2, 0, 0, 0, 7, 6, 0, 0, 0, 0, 0 },
    /*4*/ { 4, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0 },
    /*5*/ { 0, 3, 7, 0, 0, 0, 0, 7, 5, 0, 0 },
    /*6*/ { 0, 0, 6, 2, 0, 0, 0, 7, 0, 3, 0 },
    /*7*/ { 0, 2, 0, 3, 0, 0, 0, 0, 5, 4, 0 },
    /*8*/ { 0, 0, 0, 0, 7, 7, 0, 0, 0, 0, 4 },
    /*9*/ { 0, 0, 0, 0, 5, 0, 5, 0, 0, 0, 1 },
   /*10*/ { 0, 0, 0, 0, 0, 3, 4, 0, 0, 0, 4 },
   /*11*/ { 0, 0, 0, 0, 0, 0, 0, 4, 1, 4, 0 }
};

removeRepeated(11, A);

/**
 * Prints verticles sorted by weight
 */
printf("\nVerticles sorted by weight:");

// weight, 7 is max weight
for (int i = 1; i <= 7; i++)
{
    printf("\n%d: ", i);
    // first edge
    for (int j = 1; j <= 11; j++)
    {
        // second edge
        for (int k = 1; k <= 11; k++)
        {
            if (A[j - 1][k - 1] == i)
```

```
            {
                printf("%d-%d; ", j, k);
            }
        }
    }
}

/**
 * Checks sorted vertivles and adds one to our path only if two
edges are in different trees
 */

int B[11][11];
makeTrees(11, B);

printf("\n\nOur path: ");
// weight, 7 is max weight
for (int i = 1; i <= 7; i++)
{
    // first edge
    for (int j = 1; j <= 11; j++)
    {
        // second edge
        for (int k = 1; k <= 11; k++)
        {
            if (A[j - 1][k - 1] == i && areInDifferentTrees(11, B, j, k))
            {
                addToTree(11, B, j, k);
                printf("%d-%d; ", j, k);
            }
        }
    }
```

```
    }
    printf("\n\n");

    return 0;
}

int makeTrees(int n, int A[n][n])
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            A[i][j] = 0;
        }
    }
    for (int i = 0; i < n; i++)
    {
        A[i][i] = i + 1;
    }

    return A[n][n];
}

void removeRepeated(int n, int A[n][n])
{
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (j < i)
            {
                A[i][j] = 0;
```

```
        }
      }
   }
}

int areInDifferentTrees(int n, int A[n][n], int first, int second)
{
   int temp1;
   int temp2;

   // line
   for (int i = 0; i < n; i++)
   {
      temp1 = 0;
      temp2 = 0;
      // first element
      for (int j = 0; j < n; j++)
      {
         if (A[i][j] == first)
         {
            temp1 = 1;
         }
      }
      // second element
      for (int k = 0; k < n; k++)
      {
         if (A[i][k] == second)
         {
            temp2 = 1;
         }
      }
```

```
        if (temp1 && temp2)
        {
            return 0;
        }
    }

    return 1;
}

void addToTree(int n, int A[n][n], int first, int second)
{
    int scndLine;
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (A[i][j] == second)
            {
                scndLine = i;
            }
        }
    }

    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
        {
            if (A[i][j] == first)
            {
                for (int k = 0; k < n; k++)
                {
                    if (A[scndLine][k])
```

```
                {
                    A[i][k] = A[scndLine][k];
                    A[scndLine][k] = 0;
                }
            }
        }
    }
}
```