****

[**BİL 501**](javascript:__doPostBack('grdOgretimProgramiYariyil1$cell1_1$lbtnLink','')) **- Ayrık Yapılar Dersi Projesi**

**Kruskal ve Prim Algoritmaları**

Recep Ali YILMAZ  
Mustafa YILMAZ

Yusuf COŞKUN

Yasin YENER

A.Veysel TOPLU

Maltepe Üniversitesi  
Fen Bilimleri Enstitüsü   
Bilgisayar Mühendisliği Yüksek Lisans Programı

**İçindekiler**

[Giriş 2](#_Toc378017045)

[1. Kruskal Algoritması 3](#_Toc378017046)

[1.1 Psuedo Kodu 3](#_Toc378017047)

[1.2 Örnek 4](#_Toc378017048)

[1.3 Kaynak Kodu 8](#_Toc378017049)

[2. Prim Algoritması 14](#_Toc378017050)

[2.1 Psuedo Kodu 14](#_Toc378017051)

[2.2 Örnek 15](#_Toc378017052)

[2.3 Kaynak Kodu 19](#_Toc378017053)

[3. Hazırladığımız Uygulamanın Çalışması 27](#_Toc378017054)

[4. Kodların Çalıştırılabilir Toplu Hali 30](#_Toc378017055)

[Kaynakça 57](#_Toc378017056)

# Giriş

Kruskal ve Prim algoritmaları bağlı düğümler içerisinde en kısa şekilde tüm düğümleri dolaşmayı sağlar (Minimum Spanning Tree Solving). Ağırlık değerlerine sahip bir G grafı içerisinde en küçük yol ağacını hesaplar. Algoritma en küçük yolları alıp bir döngü(loop) yapmadan tüm düğümleri dolaşarak, tüm düğümleri en kısa yoldan nasıl dolaşılabileceğini bulur. Tüm düğümlerin bağlı ve yolların çift yönlü tek ağırlıklı olması gerekmektedir. Yani A düğümünden B düğümüne gitmenin maliyeti x ise B düğümünden A düğümüne gitmenin de maliyetinin x olması gerekmektedir.

# Kruskal Algoritması

Kruskal algoritması bağlı düğümler içerisinde en kısa şekilde tüm düğümleri dolaşmayı sağlar (Minimum Spanning Tree Solving). Algoritma 1956 yılında Joseph Kruskal tarafından yazılmış hala çok yaygın olarak kullanılan bir algoritmadır. Graf üzerindeki düğümler, aralarında bağlantı olmayan N tane bağımsız küme gibi düşünülür. Daha sonra bu kümeler tek tek maliyeti en az olan kenarlarla birleştirilir (çevrim oluşturmayacak şekilde ). Düğümler arasında bağlantı olan tek bir küme oluşturulmaya çalışılır. Küme birleştirme işleminde en az maliyetli olan kenardan başlanılır; daha sonra kalan kenarlar arasından en az maliyetli olanlar seçilir.

## Psuedo Kodu

Yol ağacını oluşturan kenarların tutulduğu Yol Ağacı(YA) dizisi oluşturulur.

Grafdaki kenarları içeren K dizisi oluşturulur. Yol uzunluğuna başlangıç değerleri verilir.

while(K!=0) && Yol Uzunluğu < N {

K içerisindeki en küçük maliyetli Ki kenarını al

Ki’yi K’den sil.

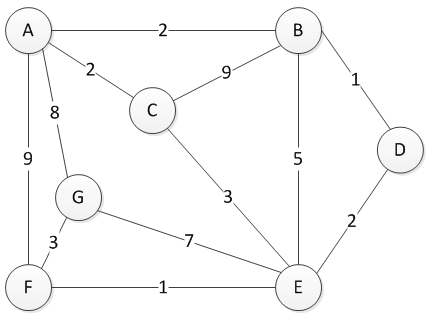
İf (Ki YA’ya eklendiğinde çevrim oluşmuyorsa) { Ki’yi YA’ya ekle

Yoluzunluğu++

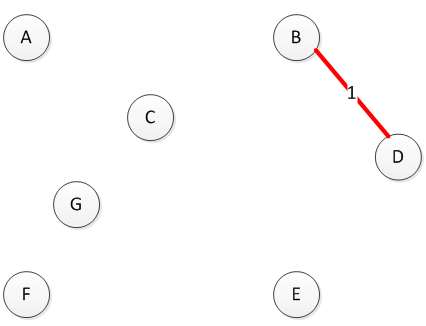
}

}

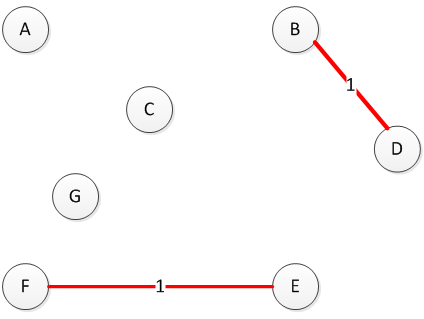
## Örnek



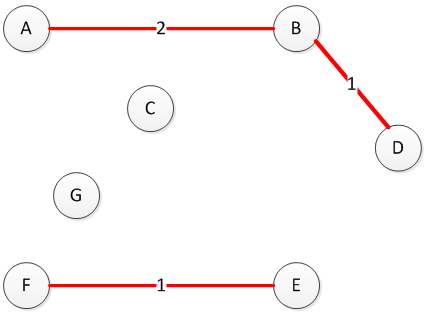
**Adım 1:** En küçük ağırlıklı yollar bulunuyor eğer aynı aygırlık 2 yol var ise matriste ilk gelen yol çiziliyor



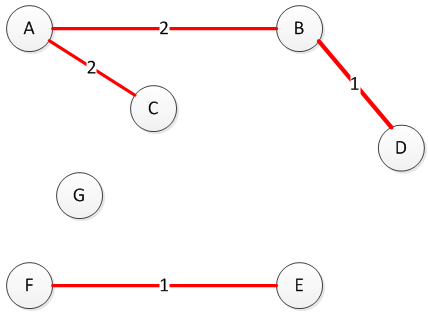
**Adım 2:** Aynı ağırlıktaki 2.yol olan F-E bağlantısını çizer.



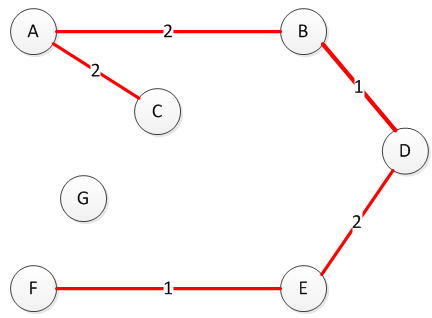
**Adım 3:** ikinci büyük ağırlıklı bağlantıyı çiziyor. Döngü oluşturmadığı için birleştirilir.



**Adım4:** A-C bağlantısı çizilir.

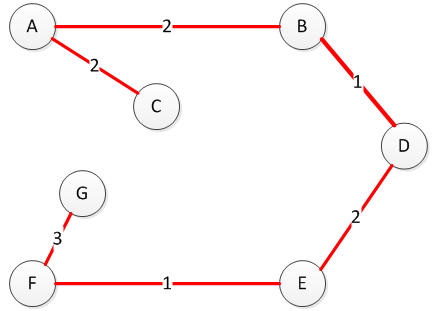


**Adım 5:** E-D döngü oluşturmadığı için birleştirilir.



**Adım 6:** Kalan düğümler arasında az maliyetli olanlar C-E ve G-F düğümleridir. Önce C-E ye bakılır. Döngü oluşturduğu için çizilmez. G-F ‘ ye bakılır ve çevrim oluşturmadığı için bağlanır Her düğüm dolaşıldığı için ve diğer düğümler döngü oluşturacağı için diğer yollar çizilmez.

Kruskal ile graf’ ın çözümü;



## Kaynak Kodu

class Kruskal{

public $x;

public $y;

public $xVal;

public $yVal;

public $Matris;

public $sonucMatris=array(array());

public $copy1;

public $mainy;

public $mainx;

public function \_\_construct($mtr){

$this->Matris=$mtr;

$this->x=count($this->Matris);

$this->y=$this->x;

$this->bosMatris();

$this->copy1=$this->Matris;

$line=0;

$vsay=0;

for ($i=0;$i<count($this->Matris)+6;$i++){

$small=$this->smallPoint();

$this->mainy=$this->yVal;

$this->mainx=$this->xVal;

if ($this->cycleCheck($this->xVal,$this->yVal)){

$this->sonucMatris[$this->xVal][$this->yVal]=$this->copy1[$this->xVal][$this->yVal];

$this->vertex\_addr[$vsay][0]=$this->xVal;

$this->vertex\_addr[$vsay][1]=$this->yVal;

$vsay++;

}

}

}

public function firstSmall($mtrs){

$small=0;

for ($m=0;$m<$this->x;$m++){

for ($n=0;$n<$this->y;$n++){

if (($mtrs[$m][$n]!=-1)&&($mtrs[$m][$n]!=0)){

$small=$mtrs[$m][$n];

$this->xVal=$m;

$this->yVal=$n;

return $small;

}

}

}

return $small;

}

public function smallPoint(){

$small=$this->firstSmall($this->Matris);

for ($m=0;$m<$this->x;$m++){

for ($n=0;$n<$this->y;$n++){

if ($small>$this->Matris[$m][$n] && $this->Matris[$m][$n]!=-1 && $this->Matris[$m][$n]!=0){

$small=$this->Matris[$m][$n];

$this->xVal=$m;

$this->yVal=$n;

}

}

}

$this->Matris[$this->xVal][$this->yVal]=0;

$this->Matris[$this->yVal][$this->xVal]=0;

return $small;

}

public function cycleCheck($x,$y,$ki=""){

/\*Yeni eklenecek Vertex köşelerini eski vertex köşeleri ile karşılatırıyor Döngü oluşturuyor ise false gönderiyor ve Yeni Vertex ile Edge eklenmiyor\*/

$x1="";

$y1="";

$buldu=true;

for ($i=0;$i<count($this->vertex\_addr);$i++){

if ($ki!=$i){

if ($this->vertex\_addr[$i][0]==$y){

$y1=$this->vertex\_addr[$i][1];

if ($y1==$this->mainx){

return false;

}

$buldu=$this->cycleCheck($x,$y1,$i);

if ($buldu==false){

return false;

}

}elseif ($this->vertex\_addr[$i][1]==$y){

$y1=$this->vertex\_addr[$i][0];

if ($y1==$this->mainx){

return false;

}

$buldu=$this->cycleCheck($x,$y1,$i);

if ($buldu==false){

return false;

}

}

}

}

return $buldu;

}

public function matrisyaz($mtr){

/\*Parametre ile gelen matrisi yazdırıyor\*/

for ($i=0;$i<count($mtr);$i++){

echo "Adım ".($i+1)."> <b>";

for ($j=0;$j<count($mtr[0]);$j++){

echo $this->get\_vad($mtr[$i][$j]+1);

if ($j==0){

echo "-";

}

}

echo "</b><br>";

}

return true;

}

public function bosMatris(){

/\* Boş sonuç Matrisi oluşturuluyor\*/

for($i=0;$i<count($this->Matris);$i++){

for($j=0;$j<count($this->Matris[0]);$j++){

$this->sonucMatris[$i][$j]=0;

}

}

return true;

}

public function get\_vad($no){

//Vertex i adlandırıyor

switch ($no){

case "1":$vad="A";break;

case "2":$vad="B";break;

case "3":$vad="C";break;

case "4":$vad="D";break;

case "5":$vad="E";break;

case "6":$vad="F";break;

case "7":$vad="G";break;

case "8":$vad="H";break;

case "9":$vad="I";break;

case "10":$vad="J";break;

case "11":$vad="K";break;

}

return $vad;

}

}

# Prim Algoritması

Prim algoritması en küçük kapsayan ağacını oluşturabilmek için kullanılan bir algoritmadır. Prim algoritmasına ismini veren Robert C. Prim tarafından 1957 yılında geliştirilmiştir (Prim, 1957). Kruskal algoritmasından tek farkı bir sonraki kenarı nasıl seçtiğidir. Başlangıçta, herhangi bir noktayı, ağaç oluşturmaya başlamak için seçilir. Oluşturulan ağaca kenar eklemek için, şu ana kadar oluşturulmuş ağaç üzerinden erişilebilen ve daha önceden ağaca katılmamış olan en küçük ağırlıklı kenar seçilir. Eğer bu kenarın ağaca katılması, bir çember oluşmasına sebep olmuyorsa, ağaca eklenir. Ağaçtaki kenar sayısı (N-1)'e ulaşana kadar ikinci adıma geri dönülür.

## Psuedo Kodu

Yol ağacını oluşturan kenarların tutulduğu Yol Ağacı(YA) dizisi oluşturulur.

Grafdaki kenarları içeren K dizisi oluşturulur. Yol uzunluğuna başlangıç değerleri verilir.

Rastgele bir Ki seçilir

while(K!=0) && Yol Uzunluğu < N {

K içerisindeki Ki ye komşu olan en küçük maliyetli Ki kenarını al

Ki’yi K’den sil.

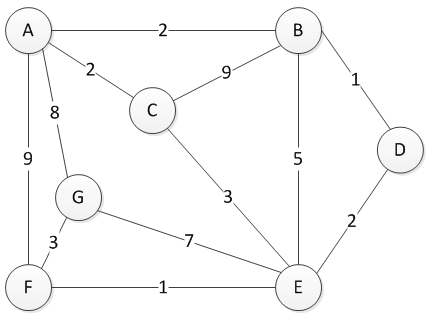
İf (Ki YA’ya eklendiğinde çevrim oluşmuyorsa) { Ki’yi YA’ya ekle

Yoluzunluğu++

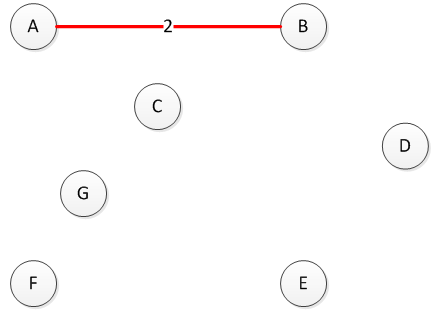
}

}

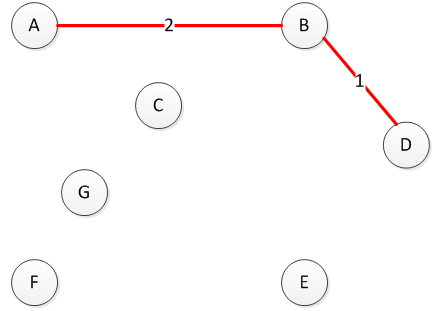
## Örnek



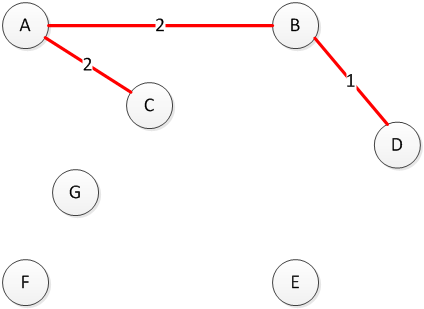
**Adım 1:** ilk adımda kurala göre herhangi bir noktadan başlanabilir bizim programız da ve algoritmamızda matristeki en küçük vertex seçilerek başlanmıştır. Bu örnekte ilke bağlantı ve bu bağlantıda en küçük ağırlık A-B noktası olduğu için seçim bu şekilde yapılmıştır.



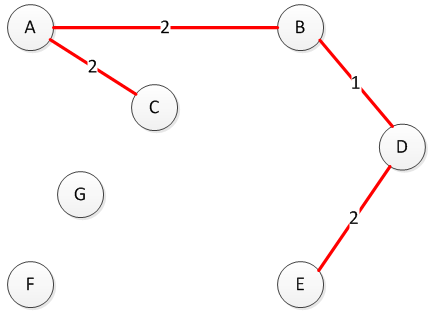
**Adım 2:** A-B bağlantılarındaki tüm uçlar kontrol edilir ve ağırlığı en düşük olan uç ile devam edilir. Örneğimizde B-D seçimi ile devam edilmektedir.



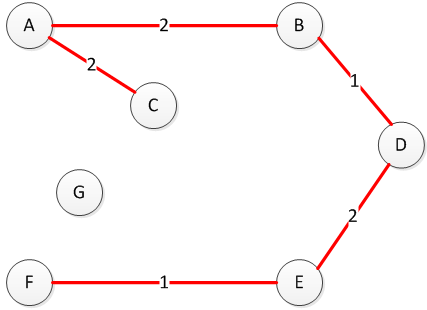
**Adım 3:** A-C seçimi ile devam edilmektedir.



**Adım 4:** D-E seçimi ile devam edilmektedir.



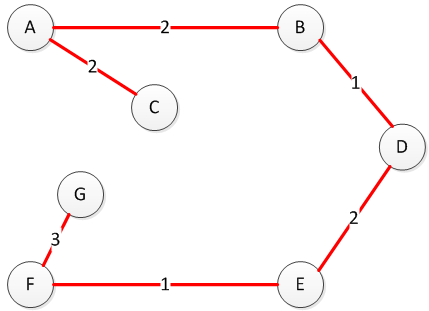
**Adım 5:** E-F seçimi ile devam edilmektedir.



**Adım 6:** F-G seçimi ile tüm noktalara ulaşıldığı için en kısa yol bulunmuştur.

Prim ile graf’ ın çözümü;

Çözümler en kısa yolu bulduğu için sonuç olarak aynı çıkmıştır fakat yol seçimleri algoritmaların farklıları sebebiyle farklı şekilde oluşmuştur.



## Kaynak Kodu

class Prim{

public $Matris;

public $sonucMatris=array(array());

public $Xval;

public $Yval;

public $Xval1;

public $Yval1;

public function \_\_construct($mtrs){

/\*Class oluşturulduğunda bu class ilk çalışıyor\*/

$this->Matris=$mtrs;

$this->bosMatris();//sonucMatris 0 değerler ile üretiliyor

$this->lineSmall($this->Matris[0]); //İlk satırın en küçük değerine aranıyor ve getirliyor

$this->sonucMatris[0][$this->Yval]=$this->Matris[0][$this->Yval]; //Sonuc matrise gelen değerin ağırlıpı atanıyor

$this->Matris[0][$this->Yval]=-1;//Kullanılan değerler tekarar değerlendirilmemsi için -1 ile sıfırlanıyor

$this->Matris[$this->Yval][0]=-1;

$this->vertex\_addr[0][0]=0;

$this->vertex\_addr[0][1]=$this->Yval;

for($i=0;$i<count($this->Matris);$i++){

$small=$this->vertexscan();

if ($small!=0){

$vl=count($this->vertex\_addr);

if ($this->cycleCheck($this->Xval1,$this->Yval1)){

$this->sonucMatris[$this->Xval1][$this->Yval1]=$this->Matris[$this->Xval1][$this->Yval1];

$this->vertex\_addr[$vl][0]=$this->Xval1;

$this->vertex\_addr[$vl][1]=$this->Yval1;

}

$this->Matris[$this->Xval1][$this->Yval1]=-1;

$this->Matris[$this->Yval1][$this->Xval1]=-1;

}

}

}

public function lineSmall($dz){

/\*Parametre ile gelen $dz satırının en küçük olanı bulunuyor\*/

$small=0;

for ($i=0;$i<count($dz);$i++){

if (($dz[$i]!=-1)&&($dz[$i]!=-2)){

if ($small==0){

$small=$dz[$i];

$this->Yval=$i;

}elseif($small>$dz[$i]){

$small=$dz[$i];

$this->Yval=$i;

}

}

}

return $small;

}

public function vertexscan(){

/\*Daha önce oluşturulan Vertex lerin uçlarından hangisinin en ufak olduğunu arıyor. Eğer en ufak tan 2 tane var ise indisi ufak olanı önce alıyor \*/

$mainSmall=0;

for ($i=0;$i<count($this->vertex\_addr);$i++){

$x1=$this->vertex\_addr[$i][0];//Alınan veretxlerin tutulduğu diziden değerler alınıyor

$x2=$this->vertex\_addr[$i][1];//

$s1=$this->lineSmall($this->Matris[$x1]);

$y1=$this->Yval;

$s2=$this->lineSmall($this->Matris[$x2]);

$y2=$this->Yval;

if ($s1>$s2){

if ($s2>0){

if ($mainSmall==0){

$mainSmall=$s2;

$this->Xval1=$x2;

$this->Yval1=$y2;

}elseif ($mainSmall>$s2){

$mainSmall=$s2;

$this->Xval1=$x2;

$this->Yval1=$y2;

}

}else{

if ($mainSmall==0){

$mainSmall=$s1;

$this->Xval1=$x1;

$this->Yval1=$y1;

}elseif ($mainSmall>$s1){

$mainSmall=$s1;

$this->Xval1=$x1;

$this->Yval1=$y1;

}

}

}else{

if ($s1>0){

if ($mainSmall==0){

$mainSmall=$s1;

$this->Xval1=$x1;

$this->Yval1=$y1;

}elseif ($mainSmall>$s1){

$mainSmall=$s1;

$this->Xval1=$x1;

$this->Yval1=$y1;

}

}else{

if ($s2>0){

if ($mainSmall==0){

$mainSmall=$s2;

$this->Xval1=$x2;

$this->Yval1=$y2;

}elseif ($mainSmall>$s2){

$mainSmall=$s2;

$this->Xval1=$x2;

$this->Yval1=$y2;

}

}

}

}

}

return $mainSmall;

}

public function cycleCheck($x,$y,$ki=""){

/\*Yeni eklenecek Vertex köşelerini eski vertex köşeleri ile karşılatırıyor Döngü oluşturuyor ise false gönderiyor ve Yeni Vertex ile Edge eklenmiyor. Kendi kendini çağıran fonksiyon bir nevi Drilling yapıyor\*/

$x1="";

$y1="";

$buldu=true;

for ($i=0;$i<count($this->vertex\_addr);$i++){

if ($ki!=$i){

if ($this->vertex\_addr[$i][0]==$y){

$y1=$this->vertex\_addr[$i][1];

if ($y1==$this->mainx){

return false;

}

$buldu=$this->cycleCheck($x,$y1,$i);

if ($buldu==false){

return false;

}

}elseif ($this->vertex\_addr[$i][1]==$y){

$y1=$this->vertex\_addr[$i][0];

if ($y1==$this->mainx){

return false;

}

$buldu=$this->cycleCheck($x,$y1,$i);

if ($buldu==false){

return false;

}

}

}

}

return $buldu;

}

public function matrisyaz($mtr){

/\*Parametre ile gelen matrisi yazdırıyor\*/

for ($i=0;$i<count($mtr);$i++){

echo "Adım ".($i+1)."> <b>";

for ($j=0;$j<count($mtr[0]);$j++){

echo $this->get\_vad($mtr[$i][$j]+1);

if ($j==0){

echo "-";

}

}

echo "</b><br>";

}

return true;

}

public function bosMatris(){

/\* Boş sonuç Matrisi oluşturuluyor\*/

for($i=0;$i<count($this->Matris);$i++){

for($j=0;$j<count($this->Matris[0]);$j++){

$this->sonucMatris[$i][$j]=0;

}

}

return true;

}

public function get\_vad($no){

//Vertex i adlandırıyor

switch ($no){

case "1":$vad="A";break;

case "2":$vad="B";break;

case "3":$vad="C";break;

case "4":$vad="D";break;

case "5":$vad="E";break;

case "6":$vad="F";break;

case "7":$vad="G";break;

case "8":$vad="H";break;

case "9":$vad="I";break;

case "10":$vad="J";break;

case "11":$vad="K";break;

}

return $vad;

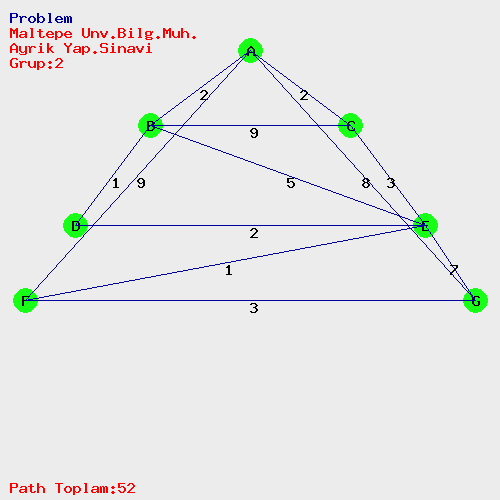
}

}

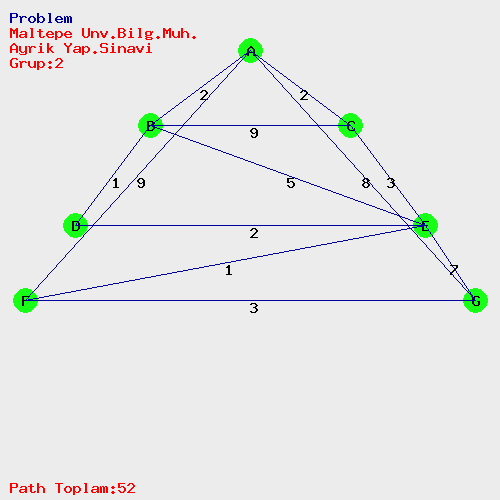
# Hazırladığımız Uygulamanın Çalışması

Bu ödev için kodlarımızı PHP dilinde geliştirdik. Kod’ a probleme ait grafın matisi oluşturuluyor ve giriliyor kruskal ve prim algoritmalarını işletip çözüm için yeni bir matris oluşturuyor. Kodda yer alan grafik çizim bölümü giriş ve sonuç matrislerini kullanarak matrisleri grafikler olarak ekranda gösteriyor. Ekranda çözüm grafiklerinin yanında çözüme hangi vertexleri kullanarak ve hangi adımlar ile ulaştığını gösteren bir tablo yer almaktadır bu tablo sayesinde her iki algoritmanın takip ettiği yolları ve adımları karşılaştırmalı olarak görebilmekteyiz.

Örnek problemin programımız ile çizilmiş görüntüsü;



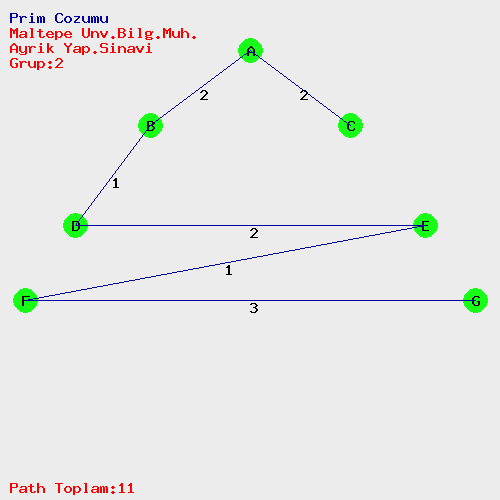
Örnek problemin kruskal algoritması ile çözümü ve grafik olarak görüntülenmesi;



Kruskal ile çözüme ulaşırken gidilen adımlar;

Adım 1> B-D  
Adım 2> E-F  
Adım 3> A-B  
Adım 4> A-C  
Adım 5> D-E  
Adım 6> F-G

Örnek problemin prim algoritması ile çözümü ve grafik olarak görüntülenmesi;



Prim ile çözüme ulaşırken gidilen adımlar;

Adım 1> A-B  
Adım 2> B-D  
Adım 3> A-C  
Adım 4> D-E  
Adım 5> E-F  
Adım 6> F-G

# Kodların Çalıştırılabilir Toplu Hali

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>Maltepe Ünv.Bilgisayar Müh.Yük.Lisansı Ayrık Yapılar Dersi</title>

</head>

<body>

<?php

$matris1=array(array(-1,2,2,-1,-1,9,8),

array(2,-1,9,1,5,-1,-1),

array(2,9,-1,-1,3,-1,-1),

array(-1,1,-1,-1,2,-1,-1),

array(0,5,3,2,-1,1,7),

array(9,-1,-1,-1,1,-1,3),

array(8,-1,-1,-1,7,3,-1)

);

$matris2= array(array(-1, 5, 11,-1,-1,-1),

array(5,-1,20,10,-1,-1),

array(11,20,-1,10,15,-1),

array(-1,10,10,-1,5,20),

array(-1,-1,15,5,-1,15),

array(-1,-1,-1,20,15,-1)

);

//1 2 3 4 5 6 7 8 9 10

$matris3=array(array(-1, 9,-1,-1,-1,-1,-1,-1,-1,-1),//1

array( 9,-1,-1,-1,-1,-1, 7,-1,-1,-1),//2

array(-1,-1,-1,-1, 2,-1,-1,-1,-1,-1),//3

array(-1,-1,-1,-1,-1, 1,-1,-1, 3,-1),//4

array(-1,-1, 2,-1,-1, 2,-1,-1,-1,-1),//5

array(-1,-1,-1, 1, 2,-1,10,-1, 8, 7),//6

array(-1, 7,-1,-1,-1,10,-1, 8, 8,-1),//7

array(-1,-1,-1,-1,-1,-1, 8,-1,-1,-1),//8

array(-1,-1,-1, 3,-1, 8, 8,-1,-1,-1),//9

array(-1,-1,-1,-1,-1, 7,-1,-1,-1,-1)//10

);

//1 2 3 4 5 6 7 8 9 10

$matris4=array(array(-1, 2,-1,-1,-1,-1,-1,-1,10,-1),//1

array( 2,-1, 7,-1, 2,-1,-1,-1,-1,-1),//2

array(-1, 7,-1, 6,-1,-1,-1,-1,-1, 5),//3

array(-1,-1, 6,-1,-1, 6,-1,-1,-1,-1),//4

array(-1, 2,-1,-1,-1,10,-1,-1,-1,-1),//5

array(-1,-1,-1, 6,10,-1,-1, 6,-1,-1),//6

array(-1,-1,-1,-1,-1,-1,-1, 7,-1, 3),//7

array(-1,-1,-1,-1,-1, 6, 7,-1, 4,-1),//8

array(10,-1,-1,-1,-1,-1,-1, 4,-1,-1),//9

array(-1,-1, 5,-1,-1,-1, 3,-1,-1,-1)//10

);

$matris=array(array());

$matris=$matris4; //Yukarıda tanımlanaan matrisin değişken adını yazın

$a=new Kruskal($matris);

$b=new Prim($matris);

echo ("<table border='1'>");

echo("<tr><td colspan='4' align='center'><b>

Maltepe Üniversitesi Fen Bilimleri Enstitüsü Bilgisayar Mühendisliği Yük. Lisansı<br>

Ayrık Yapılar Dersi Final Sınavı Projesi (24 Ocak 2014)<br>

</b></td></tr>");

echo("<tr><td colspan='4' align='center'>");

$c=new graph\_yap($matris,"gp1","Problem");

echo("</td></tr>");

echo("<tr><td colspan='2' align='center'>Kruskal Yöntemi İle Çözüm</td><td colspan='2' align='center'>Prim Yöntemi İle Çözüm</td></tr>");

echo("<tr><td>");

$c=new graph\_yap($a->sonucMatris,"gp2","Kruskal Cozumu");

echo("</td><td valign='top'>");

$a->matrisyaz($a->vertex\_addr);

echo("</td>");

echo("<td>");

$c=new graph\_yap($b->sonucMatris,"gp3","Prim Cozumu");

echo("</td><td valign='top'>");

$b->matrisyaz($b->vertex\_addr);

echo("</td></tr>");

echo("<tr><td colspan='4' align='center'><b>

Ders Hocası:Prof. Dr.İlhami Yavuz<br>

Öğrenciler: 1-)Ahmet Veysel Toplu 2-)Mustafa Yılmaz 3-)Recep Ali Yılmaz 4-)Yasin Yener 5-)Yusuf Coşkun <br>

</b></td></tr>");

echo("</table>");

class Prim{

public $Matris;

public $sonucMatris=array(array());

public $Xval;

public $Yval;

public $Xval1;

public $Yval1;

public function \_\_construct($mtrs){

/\*Class oluşturulduğunda bu class ilk çalışıyor\*/

$this->Matris=$mtrs;

$this->bosMatris();//sonucMatris 0 değerler ile üretiliyor

$this->lineSmall($this->Matris[0]); //İlk satırın en küçük değerine aranıyor ve getirliyor

$this->sonucMatris[0][$this->Yval]=$this->Matris[0][$this->Yval]; //Sonuc matrise gelen değerin ağırlıpı atanıyor

$this->Matris[0][$this->Yval]=-1;//Kullanılan değerler tekarar değerlendirilmemsi için -1 ile sıfırlanıyor

$this->Matris[$this->Yval][0]=-1;

$this->vertex\_addr[0][0]=0;

$this->vertex\_addr[0][1]=$this->Yval;

for($i=0;$i<count($this->Matris);$i++){

$small=$this->vertexscan();

if ($small!=0){

$vl=count($this->vertex\_addr);

if ($this->cycleCheck($this->Xval1,$this->Yval1)){

$this->sonucMatris[$this->Xval1][$this->Yval1]=$this->Matris[$this->Xval1][$this->Yval1];

$this->vertex\_addr[$vl][0]=$this->Xval1;

$this->vertex\_addr[$vl][1]=$this->Yval1;

}

$this->Matris[$this->Xval1][$this->Yval1]=-1;

$this->Matris[$this->Yval1][$this->Xval1]=-1;

}

}

}

public function lineSmall($dz){

/\*Parametre ile gelen $dz satırının en küçük olanı bulunuyor\*/

$small=0;

for ($i=0;$i<count($dz);$i++){

if (($dz[$i]!=-1)&&($dz[$i]!=-2)){

if ($small==0){

$small=$dz[$i];

$this->Yval=$i;

}elseif($small>$dz[$i]){

$small=$dz[$i];

$this->Yval=$i;

}

}

}

return $small;

}

public function vertexscan(){

/\*Daha önce oluşturulan Vertex lerin uçlarından hangisinin en ufak olduğunu arıyor. Eğer en ufak tan 2 tane var ise indisi ufak olanı önce alıyor \*/

$mainSmall=0;

for ($i=0;$i<count($this->vertex\_addr);$i++){

$x1=$this->vertex\_addr[$i][0];//Alınan veretxlerin tutulduğu diziden değerler alınıyor

$x2=$this->vertex\_addr[$i][1];//

$s1=$this->lineSmall($this->Matris[$x1]);

$y1=$this->Yval;

$s2=$this->lineSmall($this->Matris[$x2]);

$y2=$this->Yval;

if ($s1>$s2){

if ($s2>0){

if ($mainSmall==0){

$mainSmall=$s2;

$this->Xval1=$x2;

$this->Yval1=$y2;

}elseif ($mainSmall>$s2){

$mainSmall=$s2;

$this->Xval1=$x2;

$this->Yval1=$y2;

}

}else{

if ($mainSmall==0){

$mainSmall=$s1;

$this->Xval1=$x1;

$this->Yval1=$y1;

}elseif ($mainSmall>$s1){

$mainSmall=$s1;

$this->Xval1=$x1;

$this->Yval1=$y1;

}

}

}else{

if ($s1>0){

if ($mainSmall==0){

$mainSmall=$s1;

$this->Xval1=$x1;

$this->Yval1=$y1;

}elseif ($mainSmall>$s1){

$mainSmall=$s1;

$this->Xval1=$x1;

$this->Yval1=$y1;

}

}else{

if ($s2>0){

if ($mainSmall==0){

$mainSmall=$s2;

$this->Xval1=$x2;

$this->Yval1=$y2;

}elseif ($mainSmall>$s2){

$mainSmall=$s2;

$this->Xval1=$x2;

$this->Yval1=$y2;

}

}

}

}

}

return $mainSmall;

}

public function cycleCheck($x,$y,$ki=""){

/\*Yeni eklenecek Vertex köşelerini eski vertex köşeleri ile karşılatırıyor Döngü oluşturuyor ise false gönderiyor ve Yeni Vertex ile Edge eklenmiyor. Kendi kendini çağıran fonksiyon bir nevi Drilling yapıyor\*/

$x1="";

$y1="";

$buldu=true;

for ($i=0;$i<count($this->vertex\_addr);$i++){

if ($ki!=$i){

if ($this->vertex\_addr[$i][0]==$y){

$y1=$this->vertex\_addr[$i][1];

if ($y1==$this->mainx){

return false;

}

$buldu=$this->cycleCheck($x,$y1,$i);

if ($buldu==false){

return false;

}

}elseif ($this->vertex\_addr[$i][1]==$y){

$y1=$this->vertex\_addr[$i][0];

if ($y1==$this->mainx){

return false;

}

$buldu=$this->cycleCheck($x,$y1,$i);

if ($buldu==false){

return false;

}

}

}

}

return $buldu;

}

public function matrisyaz($mtr){

/\*Parametre ile gelen matrisi yazdırıyor\*/

for ($i=0;$i<count($mtr);$i++){

echo "Adım ".($i+1)."> <b>";

for ($j=0;$j<count($mtr[0]);$j++){

echo $this->get\_vad($mtr[$i][$j]+1);

if ($j==0){

echo "-";

}

}

echo "</b><br>";

}

return true;

}

public function bosMatris(){

/\* Boş sonuç Matrisi oluşturuluyor\*/

for($i=0;$i<count($this->Matris);$i++){

for($j=0;$j<count($this->Matris[0]);$j++){

$this->sonucMatris[$i][$j]=0;

}

}

return true;

}

public function get\_vad($no){

//Vertex i adlandırıyor

switch ($no){

case "1":$vad="A";break;

case "2":$vad="B";break;

case "3":$vad="C";break;

case "4":$vad="D";break;

case "5":$vad="E";break;

case "6":$vad="F";break;

case "7":$vad="G";break;

case "8":$vad="H";break;

case "9":$vad="I";break;

case "10":$vad="J";break;

case "11":$vad="K";break;

}

return $vad;

}

}

class Kruskal{

public $x;

public $y;

public $xVal;

public $yVal;

public $Matris;

public $sonucMatris=array(array());

public $copy1;

public $mainy;

public $mainx;

public function \_\_construct($mtr){

$this->Matris=$mtr;

$this->x=count($this->Matris);

$this->y=$this->x;

$this->bosMatris();

$this->copy1=$this->Matris;

$line=0;

$vsay=0;

for ($i=0;$i<count($this->Matris)+6;$i++){

$small=$this->smallPoint();

$this->mainy=$this->yVal;

$this->mainx=$this->xVal;

if ($this->cycleCheck($this->xVal,$this->yVal)){

$this->sonucMatris[$this->xVal][$this->yVal]=$this->copy1[$this->xVal][$this->yVal];

$this->vertex\_addr[$vsay][0]=$this->xVal;

$this->vertex\_addr[$vsay][1]=$this->yVal;

$vsay++;

}

}

}

public function firstSmall($mtrs){

$small=0;

for ($m=0;$m<$this->x;$m++){

for ($n=0;$n<$this->y;$n++){

if (($mtrs[$m][$n]!=-1)&&($mtrs[$m][$n]!=0)){

$small=$mtrs[$m][$n];

$this->xVal=$m;

$this->yVal=$n;

return $small;

}

}

}

return $small;

}

public function smallPoint(){

$small=$this->firstSmall($this->Matris);

for ($m=0;$m<$this->x;$m++){

for ($n=0;$n<$this->y;$n++){

if ($small>$this->Matris[$m][$n] && $this->Matris[$m][$n]!=-1 && $this->Matris[$m][$n]!=0){

$small=$this->Matris[$m][$n];

$this->xVal=$m;

$this->yVal=$n;

}

}

}

$this->Matris[$this->xVal][$this->yVal]=0;

$this->Matris[$this->yVal][$this->xVal]=0;

return $small;

}

public function cycleCheck($x,$y,$ki=""){

/\*Yeni eklenecek Vertex köşelerini eski vertex köşeleri ile karşılatırıyor Döngü oluşturuyor ise false gönderiyor ve Yeni Vertex ile Edge eklenmiyor\*/

$x1="";

$y1="";

$buldu=true;

for ($i=0;$i<count($this->vertex\_addr);$i++){

if ($ki!=$i){

if ($this->vertex\_addr[$i][0]==$y){

$y1=$this->vertex\_addr[$i][1];

if ($y1==$this->mainx){

return false;

}

$buldu=$this->cycleCheck($x,$y1,$i);

if ($buldu==false){

return false;

}

}elseif ($this->vertex\_addr[$i][1]==$y){

$y1=$this->vertex\_addr[$i][0];

if ($y1==$this->mainx){

return false;

}

$buldu=$this->cycleCheck($x,$y1,$i);

if ($buldu==false){

return false;

}

}

}

}

return $buldu;

}

public function matrisyaz($mtr){

/\*Parametre ile gelen matrisi yazdırıyor\*/

for ($i=0;$i<count($mtr);$i++){

echo "Adım ".($i+1)."> <b>";

for ($j=0;$j<count($mtr[0]);$j++){

echo $this->get\_vad($mtr[$i][$j]+1);

if ($j==0){

echo "-";

}

}

echo "</b><br>";

}

return true;

}

public function bosMatris(){

/\* Boş sonuç Matrisi oluşturuluyor\*/

for($i=0;$i<count($this->Matris);$i++){

for($j=0;$j<count($this->Matris[0]);$j++){

$this->sonucMatris[$i][$j]=0;

}

}

return true;

}

public function get\_vad($no){

//Vertex i adlandırıyor

switch ($no){

case "1":$vad="A";break;

case "2":$vad="B";break;

case "3":$vad="C";break;

case "4":$vad="D";break;

case "5":$vad="E";break;

case "6":$vad="F";break;

case "7":$vad="G";break;

case "8":$vad="H";break;

case "9":$vad="I";break;

case "10":$vad="J";break;

case "11":$vad="K";break;

}

return $vad;

}

}

class graph\_yap{

public $w=500;

public $h=500;

public $Siyah;

public $Beyaz;

public $Mavi;

public $Kirmizi;

public $Yesil;

public $Gri;

public $Resim;

public $x=array();

public $y=array();

public $graph;

public $pTotal=0;

public function \_\_construct($gr,$fname,$baslik){

$this->Resim=@imagecreate($this->w,$this->h);

$this->Siyah = imagecolorAllocate($this->Resim, 0, 0, 0);

$this->Beyaz = imagecolorallocate($this->Resim, 255, 255, 255);

$this->Mavi = imagecolorallocate($this->Resim, 0, 0, 160);

$this->Kirmizi = imagecolorAllocate($this->Resim, 255, 0, 0);

$this->Yesil = imagecolorAllocate($this->Resim, 23, 255, 23);

$this->Gri = imagecolorAllocate($this->Resim, 236, 236, 236);

imagefilledrectangle($this->Resim, 0, 0, $this->w, $this->h ,$this->Gri);

$this->graph=$gr;

$this->vertex\_yerlestir();

$this->edge\_yerlestir();

imagestring($this->Resim,5,10,10,$baslik,$this->Mavi);

imagestring($this->Resim,5,10,25,"Maltepe Unv.Bilg.Muh.",$this->Kirmizi);

imagestring($this->Resim,5,10,40,"Ayrik Yap.Sinavi",$this->Kirmizi);

imagestring($this->Resim,5,10,55,"Grup:2",$this->Kirmizi);

if ($baslik=="Problem"){

imagestring($this->Resim,5,10,480,"Path Toplam:".$this->pTotal/2,$this->Kirmizi);

}else{

imagestring($this->Resim,5,10,480,"Path Toplam:".$this->pTotal,$this->Kirmizi);

}

$File=$fname;

$filename = sprintf("%s.png",$File);

ImagePNG($this->Resim,$filename);

ImageDestroy($this->Resim);

printf("<img src='%s'> \n", $filename);

}

public function get\_vad($no){

//Vertex i adlandırıyor

switch ($no){

case "1":$vad="A";break;

case "2":$vad="B";break;

case "3":$vad="C";break;

case "4":$vad="D";break;

case "5":$vad="E";break;

case "6":$vad="F";break;

case "7":$vad="G";break;

case "8":$vad="H";break;

case "9":$vad="I";break;

case "10":$vad="J";break;

case "11":$vad="K";break;

}

return $vad;

}

public function vertex\_yerlestir(){

$vno=count($this->graph);

$i=0;

if ($vno<=$i) break;

$m=250;

$uz=200;

$this->x[$i]=$m;

$this->y[$i]=$m-200;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=1;

if ($vno<=$i) return;

$this->x[$i]=$m-100;

$this->y[$i]=$m-125;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=2;

if ($vno<=$i) return;

$this->x[$i]=$m+100;

$this->y[$i]=$m-125;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=3;

if ($vno<=$i) return;

$this->x[$i]=$m-175;

$this->y[$i]=$m-25;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=4;

if ($vno<=$i) return;

$this->x[$i]=$m+175;

$this->y[$i]=$m-25;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=5;

if ($vno<=$i) return;

$this->x[$i]=$m-225;

$this->y[$i]=$m+50;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=6;

if ($vno<=$i) return;

$this->x[$i]=$m+225;

$this->y[$i]=$m+50;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=7;

if ($vno<=$i) return;

$this->x[$i]=$m-175;

$this->y[$i]=$m+125;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=8;

if ($vno<=$i) return;

$this->x[$i]=$m+175;

$this->y[$i]=$m+125;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=9;

if ($vno<=$i) return;

$this->x[$i]=$m-100;

$this->y[$i]=$m+200;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

$i=10;

if ($vno<=$i) return;

$this->x[$i]=$m+100;

$this->y[$i]=$m+200;

$vad=$this->get\_vad($i+1);

$this->vertex\_yap($this->x[$i],$this->y[$i],25,$vad);

return true;

}

public function edge\_yerlestir(){

$dzlen=count($this->graph);

for ($i=0;$i<$dzlen;$i++){

for ($j=0;$j<$dzlen;$j++){

$k=$this->graph[$i][$j];

if (($k!=0)&&($k!=-1)){

$this->edge\_yap($this->x[$i],$this->y[$i],$this->x[$j],$this->y[$j],$k);

}

}

}

return true;

}

public function vertex\_yap($x,$y,$g,$vad){

imagefilledellipse($this->Resim,$x,$y,$g,$g,$this->Yesil);

imagestring($this->Resim,5,$x-3,$y-7,$vad,$this->Siyah);

return true;

}

public function edge\_yap($x1,$y1,$x2,$y2,$ag){

if ($y1==$y2){

$txc=$this->mtlk(($x2-$x1)/2);

if ($x1>=$x2){

$txc+=$x2;

}else{

$txc+=$x1;

}

$txy=$y1;

}else{

$txc=$this->mtlk(($x2-$x1)/2);

if ($x1>=$x2){

$txc+=$x2;

}else{

$txc+=$x1;

}

$txy=$this->mtlk(($y2-$y1)/2);

if ($y1>=$y2){

$txy+=$y2;

}else{

$txy+=$y1;

}

}

imageline($this->Resim,$x1,$y1,$x2,$y2,$this->Mavi);

imagestring($this->Resim,5,$txc,$txy,$ag,$this->Siyah);

$this->pTotal+=$ag;

return true;

}

public function mtlk($d){

if ($d<0){

$d\*=-1;

}

return $d;

}

}

?>

</body>

</html>

# Kaynakça

* Wikipedia

http://en.wikipedia.org/wiki/Prim's\_algorithm  
http://en.wikipedia.org/wiki/Kruskal's\_algorithm

Son Erişim Zaman: 18.01.2014

* Papaioannou Panagiotis  
  Computer Engineering Department of the University of Patras  
  http://students.ceid.upatras.gr/~papagel/project/kruskal.htm  
  http://students.ceid.upatras.gr/~papagel/project/prim.htm

Son Erişim Zaman: 18.01.2014

* Agustín J. González V.  
  Ph.D. Computer Science, [Old Dominion University,](http://www.cs.odu.edu/) U.S.A. http://profesores.elo.utfsm.cl/~agv/elo320/animation/prim/prim\_kruskal.html

Son Erişim Zaman: 18.01.2014

* Onur Öçalan

[Prim ve Kruskal Algoritmaları](http://ocalanonur.com/2013/02/10/prim-ve-kruskal-algoritmalari/)  
http://ocalanonur.com/tag/prim-algorithm-c-code/

Son Erişim Zaman: 18.01.2014

* [Mehmet Ali ECER](http://www.yazgelistir.com/Yazar/mehmetaliecer)  
  [C# & VB.Net](http://www.yazgelistir.com/Kategori/csharp-vbnet) / Kruskal Algoritması(Minimum Spanning Tree)

http://www.yazgelistir.com/makale/kruskal-algoritmasi-minimum-spanning-tree-

Son Erişim Zaman: 18.01.2014

* http://enm.blogcu.com/yoneylem-arastirmasi-kruskal-algoritmasi/9707427

Son Erişim Zaman: 18.01.2014