  
  
  
**Adı : Hüseyin Burak**

**Soyadı: NEGİZ**

**Numara: 18MY03026**

**Öğretmen: Nilgün İNCEREİS**

**Minimum Yolun Bulunması Projesi Raporu**

**İÇİNDEKİLER**

* **Giriş**
* **MİNİMUM YOLUN BULUNMASI PROBLEMİ NEDİR?**
* **MİNİMUM YOLUN BULUNMASI PROBLEMİ NERELERDE KULLANILIR?**
* **MİNİMUM YOLUN BULUNMASI PROJESİ İLE İLGİLİ GELİŞTİRİLEN ALGORİTMALAR NELERDİR VE NASIL ÇALIŞIR?**
* **PROJEMDE HANGİ ALGORİTMAYI VE HANGİ DİLİ KULLANDIM?**
* **DİJKSTRA ALGORİTMASI NEDİR?**
* **DİJKSTRA ALGORİTMASI NASIL İŞLER?**
* **Özgeçmiş**
* **Kaynakça**
* **Ekler**

**GİRİŞ**

**Minimum yolun bulunması problemi birçok alanda hayatımızı kolaylaştırmak için uygulanmıştır. Yapılması planlanan projelerin çoğunda büyük bir rol oynayan minimum yolun bulunması problemi günümüzün teknolojisinde hala yerini korumakta. Minimum yolun bulunması projesi yapılacak herhangi bir eylemi en kısa yoldan yapılmasını amaçlıyor. Bu problem fiziksel dağıtımda ve lojistikte merkezi bir rol oynamaktadır. Dağıtım firmaları için öncelikle üzerinde durulması gereken bir konu bu olmuştur. Bir şirketin başarısı; ürettiği ürünün kalitesiyle, müşteri sayısının fazlalığıyla ve ürettiği ürünleri en kısa zamanda ve en az maliyetle müşterilerine ulaştırmasıyla yakından alakalıdır. Bu şirket ile talep noktası arasındaki en kısa mesafenin hesaplanması ve takip edilecek rotanın belirlenmesi değişik kısa yol algoritmaları kullanılarak hesaplanabilmektedir. Burada amaç, ürünü en az maliyetle, en kısa zamanda ve en uygun yolu kullanarak müşteriye ulaştırmaktır.**

**MİNİMUM YOLUN BULUNMASI PROBLEMİ NEDİR?**

**Minimum yolu bulmak, iki düğüm arasında en maliyetle gidilebilen bir yolun varlığını belirleme problemidir. Minimum yol problemi herhangi bir düğümden bir başka düğüme, her bir düğümden tüm düğümlere ya da tüm düğümler için hesaplanabilir.**

**MİNİMUM YOLUN BULUNMASI PROBLEMİ NERELERDE KULLANILIR?**

**Bir nakliye firmasının en az maliyetli taşıma ağını yapmasına yardımcı olur ya da bir şehir veya ülkenin su, doğalgaz vb ihtiyaçlarının, ihtiyaç sahiplerine en az maliyetle ulaşmasında büyük rol oynar. Bilgisayar üzerinde yer alabileceği alanlardan bir kaçı da internet ağ trafiği protokolü yönlendirme ve oyun programlamaladır. Bu yöntem az adette düğüme sahip problemlerin çözümünde işlem görebilmiş olsa bile düğüm sayısı arttıkça da bir o kadar verimsiz bir yapıya bürünür.**

**MİNİMUM YOLUN BULUNMASI PROJESİ İLE İLGİLİ GELİŞTİRİLEN ALGORİTMALAR NELERDİR VE NASIL ÇALIŞIR?**

1. **Dijkstra Algoritması**
2. **Bellman Ford Algoritması**
3. **Floyd-Warshall Algoritması**
4. **Kruskal Algoritması**
5. **Prim Algoritması**

**Dijkstra Algoritması:** **Dijkstra’nın algoritması belirli bir başlangıç noktasına göre en kısa yolu bulan bir algoritmadır; bir başlangıç düğümünden, diğer tüm düğümlere olan en kısa yolu belirler. Ağırlıklı ve yönlü graflar için geliştirilmiştir. Graf üzerindeki her bir kenarın ağırlığı sıfır veya sıfırdan büyük olmalıdır. Ancak burada da eksi maliyetli çevrim olmaması gerekir. Dijkstra’nın algoritması en kısa yolu belirlerken Greedy yaklaşımını kullanır. Her defasında, bir sonraki düğüme ilerleme Greedy yaklaşımına göre yapılır.**

***Bellman Ford Algoritması* : Bellman ve Ford Algoritması, Dijkstra algoritmasında olduğu gibi bir başlangıç düğümünden diğer tüm düğümlere olan en kısa yolu bulmaktadır. Bazı durumlarda grafın eksi maliyetli değerler alması söz konusu olabilmektedir. Dijkstra algoritması kenar maliyetleri sıfır veya artı olan graflar için doğru sonuç verirken Dijkstra’dan farklı olarak Bellman ve Ford Algoritmasının kenar maliyetleri eksi olan graflar için de çalışıyor olmasıdır.**

***Floyd-Warshall Algoritması*: Hem Floyd'un hem de Dijkstra'nın algoritması, köşeler arasındaki en kısa yolu bulmak için kullanılabilir. En büyük fark, Floyd'un algoritmasının tüm köşeler arasındaki en kısa yolu ve Dijkstra'nın algoritmasının tek bir tepe ile diğer tüm köşeler arasındaki en kısa yolu bulmasıdır.**

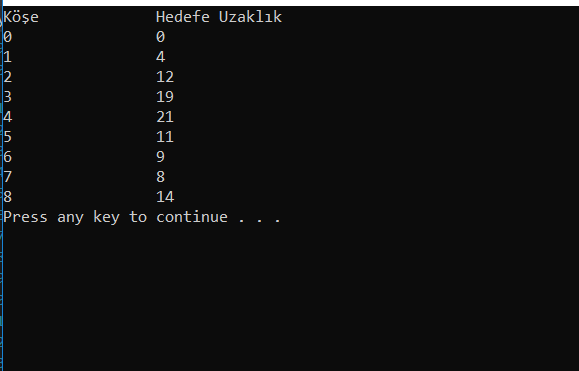
***Kruskal Algoritması*: Kruskal Algoritması daha az maliyetli kenarları tek tek değerlendirerek yol ağacını bulmaya çalışan bir algoritmadır. En küçük yol ağacını belirlemek amacıyla kullanılır. Algoritmik ifadesi davranışsal olarak basit olmasına karşın, gerçekleşmesi için bazı yardımcı fonksiyonlara gereksinim duyar.**

***Prim Algoritması*: Prim Algoritması en az maliyeli kenardan başlanarak, bu kenarın uçlarından en az maliyetle genişleyecek kenarın seçilmesi esasına dayanan bir algoritmadır. Tek kenarlık bir yol ağacından başlanarak adım adım diğer kenarların belirlenmesi esasına dayanır. Kruskal algoritmasındaki gibi, yol ağacının belirlenmesi aşamasında birden çok yol ağacı yoktur; sadece başlangıçta bir tane kenardan oluşan bir yol ağacı vardır ve sonraki adımlarda bu yol ağacına eklenmek için seçilen kenarın yol ağacı üzerinde bulunan bir düğüme komşu olması şarttır.**

**PROJEMDE HANGİ ALGORİTMAYI VE HANGİ DİLİ KULLANDIM?**

**Ben projemi C# dilinde Microsoft Visual Studio 2017 programını kullanarak yaptım. Projemde Dijkstra Algoritmasını kullandım.**

**Programın ekran çıktısı;**

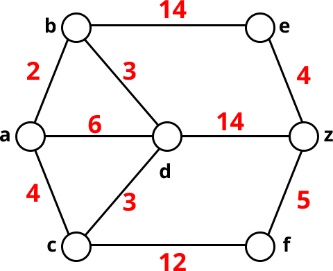
****

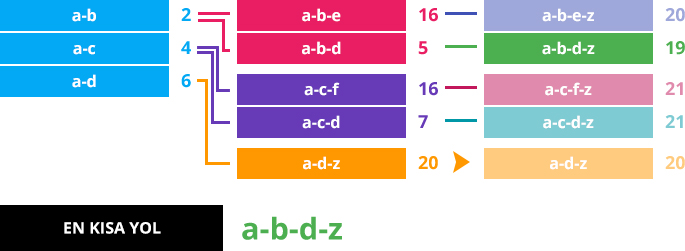
**DİJKSTRA ALGORİTMASI NEDİR?**

**Minimum yol problemleri üzerine geliştirilmiş ve araştırmalarımda karşıma çıkan “Dijkstra” algoritmasıdır. *Edsger Wybe Dijkstra* Hollandalı matematikçi ve bilgisayar bilimcidir. Bu adamın geliştirdiği bu algoritma sayesinde navigasyonumuzu açıp en kısa yola ulaşabiliyoruz. Temelde en kısa yolu bulmaya odaklanmış bir algoritmadır. Dijkstra ayrıca for, while gibi döngülerin ortaya çıkarılmasında pay sahibidir.**

**DİJKSTRA ALGORİTMASI NASIL İŞLER?**

**Dijkstra algoritması prensibi gereği kaynak noktasına ihtiyaç duyar ve etiketleme yöntemi ile çalışır. İki çeşit etiketleme mevcuttur bunlardan biri geçici diğeri ise kalıcı etiketlemedir. Geçici etiketleme yaparken iki düğüm arası mesafe hesaplanır ve yazılır. Sonraki süreçte yapılan hesaplamalarda daha kısa bir yola rastlanmaz ise geçici etiket, kalıcı etiket olarak adlandırılmaya başlar. Şayet başka bir erişimde olan yol daha kısa mesafeye sahipse yeni yol kalıcı etiket olmaya başlar.**

** Yandaki şekilde belirtilen en kısa yolu bulmak için gösterimlerimizi şöyle yapabiliriz;**

****

**ÖZGEÇMİŞ**

**1999 yılında İstanbul’da doğdum ve hala bu şehirde yaşıyorum. 2005 yılında zafer ilk öğretim okuluna başladım ve 2013 yılında mezun oldum. 2013 yılında Yenilevent Anadolu Lisesine başladım ve 2018 yılında mezun oldum. Bilgisayara olan ilgimden dolayı 2018 yılında Okan Üniversitesi’nde “Bilgisayar Programcılığı” bölümüne girdim. Okan Üniversitesinde aldığım eğitimle kendimi algoritma ve programlama konusunda geliştirdim. Şuan Okan Üniversitesinde eğitimime son senemde devam etmekteyim.**

**KAYNAKÇA**

* **http://bilinisterim.com**
* **http://bilisim.io**
* **http://muhendisbeyinler.net**
* **http://algoritmauzmani.com**
* **http://girisimzel.com**
* **http://programiz.com**

**EKLER**

**Program kodları:**

using System;

class HBN

{

static int V = 9;

int minDistance(int[] dist,

bool[] sptSet)

{

// Min. Değeri başlat

int min = int.MaxValue, min\_index = -1;

for (int v = 0; v < V; v++)

if (sptSet[v] == false && dist[v] <= min)

{

min = dist[v];

min\_index = v;

}

return min\_index;

}

// Yazdırmak için program işlevi

// inşa edilen mesafe dizisi

void printSolution(int[] dist)

{

Console.Write("Köşe \t\t Hedefe "

+ "Uzaklık \n");

for (int i = 0; i < V; i++)

Console.Write(i + " \t\t " + dist[i] + "\n");

}

// Dijkstraları uygulayan fonksiyon

// tek kaynaklı en kısa yol algoritması

// matris gösterimi

void dijkstra(int[,] graph, int src)

{

int[] dist = new int[V];

// en kısa tutacak

// src ile i arasındaki mesafe

// tepe noktası varsa sptSet[i] geçerli olur

// src - i sonlandırıldı

bool[] sptSet = new bool[V];

// Tüm mesafeleri şu şekilde başlat:

// INFINITE ve stpSet [] false olacak

for (int i = 0; i < V; i++)

{

dist[i] = int.MaxValue;

sptSet[i] = false;

}

// Kaynak tepe noktasının mesafesi

// her zaman sıfır

dist[src] = 0;

// Tüm köşeler için en kısa yolu bulun

for (int count = 0; count < V - 1; count++)

{

// Minimum mesafe tepe noktasını seçin

int u = minDistance(dist, sptSet);

// Seçilmiş tepe noktasını işlenmiş olarak işaretleme

sptSet[u] = true;

// dağıtım değerini güncelle

// toplanan tepe noktasının köşeleri.

for (int v = 0; v < V; v++)

// dist [v] 'yi yalnızca içinde değilse güncelleyin

if (!sptSet[v] && graph[u, v] != 0 && dist[u] != int.MaxValue && dist[u] + graph[u, v] < dist[v])

dist[v] = dist[u] + graph[u, v];

}

//mesafe dizisini yazdır

printSolution(dist);

}

// Sürücü Kodu

public static void Main()

{

/\* Örneği oluşturalım

yukarıda anlatılan grafik \*/

int[,] graph = new int[,] { { 0, 4, 0, 0, 0, 0, 0, 8, 0 },

{ 4, 0, 8, 0, 0, 0, 0, 11, 0 },

{ 0, 8, 0, 7, 0, 4, 0, 0, 2 },

{ 0, 0, 7, 0, 9, 14, 0, 0, 0 },

{ 0, 0, 0, 9, 0, 10, 0, 0, 0 },

{ 0, 0, 4, 14, 10, 0, 2, 0, 0 },

{ 0, 0, 0, 0, 0, 2, 0, 1, 6 },

{ 8, 11, 0, 0, 0, 0, 1, 0, 7 },

{ 0, 0, 2, 0, 0, 0, 6, 7, 0 } };

HBN t = new HBN();

t.dijkstra(graph, 0);

}

}