BELLMAN-FORD ALGORİTMASI
SHORTEST PATH PROBLEM

ZÜLAL GÜDÜK AYVAZ

SUNUM PLANI

- Bellman-Ford Algoritması
- Algoritma Nasıl Çalışır?
- Shortest Path Problem
- Piseudo Code
- Dijkstra ve Bellman-Ford Algoritması

Bellman-Ford Algoritması

Algoritmanın Amacı:

- Bir graph üzerindeki bir kaynaktan, bir hedefe giden en kısa yolu bulmaktır.
- Bir s ∈ V kaynağından tüm v ∈ V'lere bütün kısa yol uzunluklarını bulur.
- Örneğin İstanbul'dan Trabzon'a gidebileceğimiz en kısa yol gibi düşünebiliriz.
- Grafın, negatif ağırlık değerleri olduğunu saptar.

Algoritma Nasıl Çalışır?

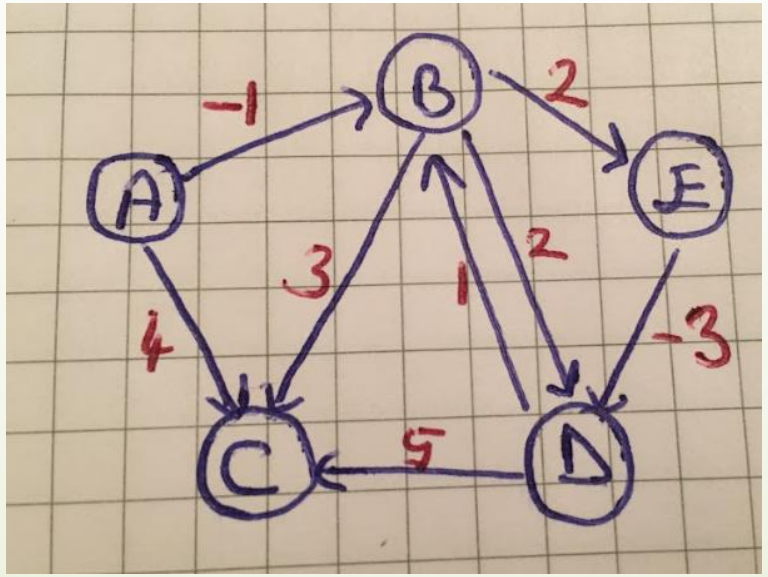
- Algoritma ağırlıklı şekiller (weighted graph) üzerinde çalışır.
- Her düğüm için bir uzaklık tahmini oluşturulur.
- Başlangıç olarak **maliyet(s)=0** değeri, **doğrudan erişilen düğümlere erişim değeri**, **erişilemeyen düğümler için maliyet(u)= ∞** değer olarak atanır.
- En az maliyetli yol hesaplanana kadar tüm kenarlar üzerinden güncelleme yapılır.

Shortest Path Problem

Örnek: Eksi değerlere sahip bir graph üzerinde, **A kaynak** düğümünden **E hedefine** giden en kısa yolu bulalım:

Formül: (Kaynak düğümün mevcut değeri**)** + **(**kaynak düğümden hedef düğüme gitme maliyeti**)** = Sonuç

Sonuç değeri **hedef düğümün mevcut değerinden** küçük mü?

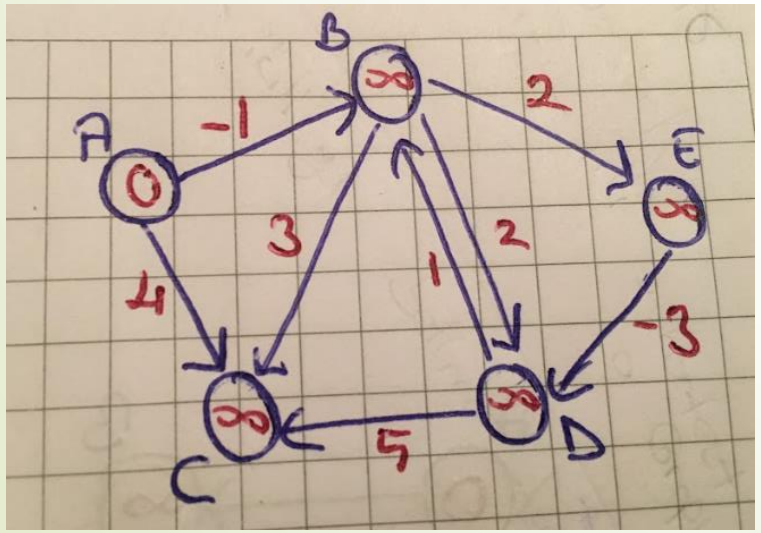


Shortest Path Problem

Adım 1:

- ✓ Öncelikle düğümlere değer ataması yapılır.
- ✓ **Şekil1'**deki gibi başlangıç düğümüne **«0»**, doğrudan erişilen düğümlere erişim değerleri ve erişilemeyen düğümlere **« ∞»** sonsuz değeri atanır.
- ✓ Her bir düğümün kaynak düğümden hedef düğüme olan uzaklığını tutacak, **Tablo1**' deki gibi, bir tablo tutulur.

Şekil1:



Tablo1:

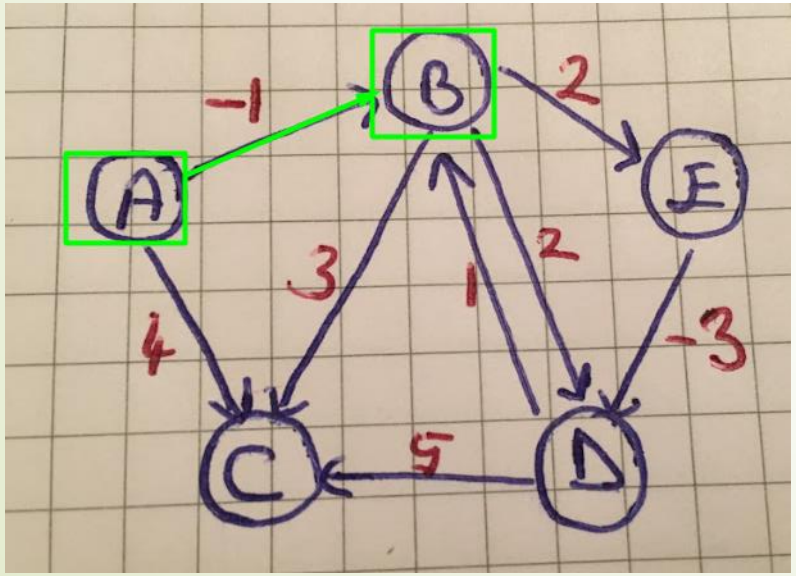
A	B	C	D	E
0	∞	∞	∞	∞

Shortest Path Problem

Adım 2:

- **(A,B; -1) :** A düğümünden B düğümüne gitme maliyeti/uzaklığı «-1». Bu durumda B düğümüne erişmenin yeni değeri ne olur?
- Kaynak düğüm mevcut değeri **A= 0;**
- Kaynak düğümden hedef düğüme gitme maliyeti **(A,B) = -1;**

Şekil2:



Tablo2:

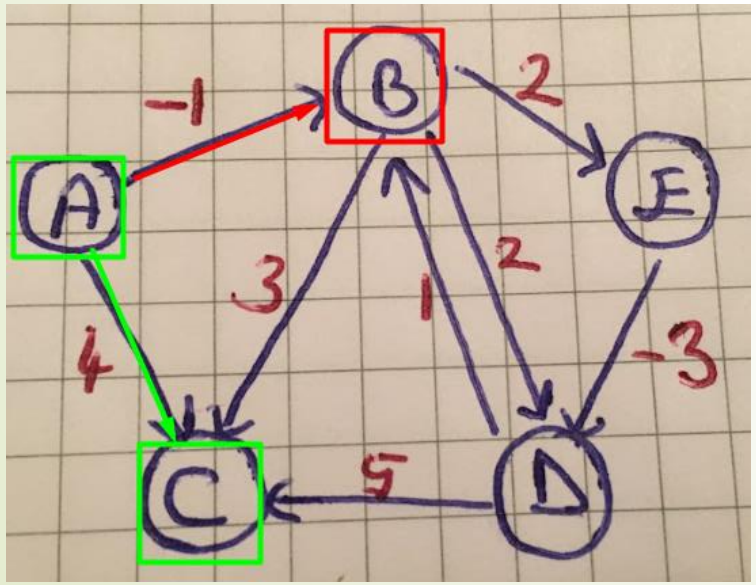
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞

Shortest Path Problem

Adım 3:

- **(A,C; 4) :** A düğümünden C düğümüne gitme maliyeti/uzaklığı «4». Bu durumda C düğümüne erişmenin yeni değeri ne olur?
- Kaynak düğüm mevcut değeri **A= 0 ;**
- Kaynak düğümden hedef düğüme gitme maliyeti **(A,C) = 4;**

Şekil3:



Tablo3:

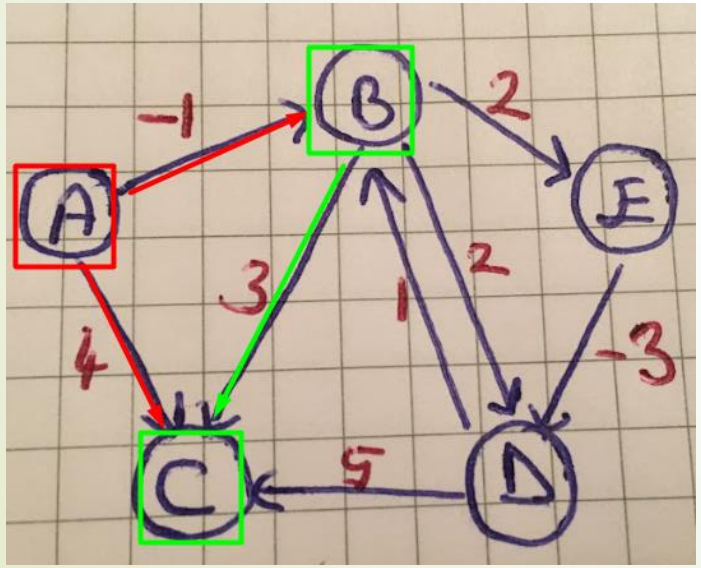
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞

Shortest Path Problem

Adım 4:

- **(B,C; 3) :** B düğümünden C düğümüne gitme maliyeti/uzaklığı «3». Bu durumda C düğümüne erişmenin yeni değeri ne olur?
- Kaynak düğüm mevcut değeri **B= -1 ;**
- Kaynak düğümden hedef düğüme gitme maliyeti **(B,C) = 3;**

Şekil4:



Tablo4:

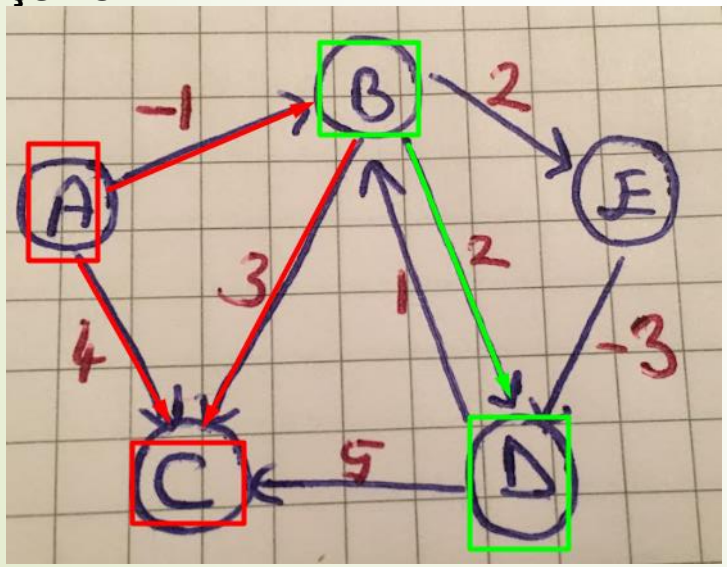
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞

Shortest Path Problem

Adım 5:

- **(B,D; 2) :** B düğümünden D düğümüne gitme maliyeti/uzaklığı «2». Bu durumda D düğümüne erişmenin yeni değeri ne olur?
- Kaynak düğüm mevcut değeri **B= -1 ;**
- Kaynak düğümden hedef düğüme gitme maliyeti **(B,D) = 2;**

Şekil5:



Tablo5:

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	1	∞

Shortest Path Problem

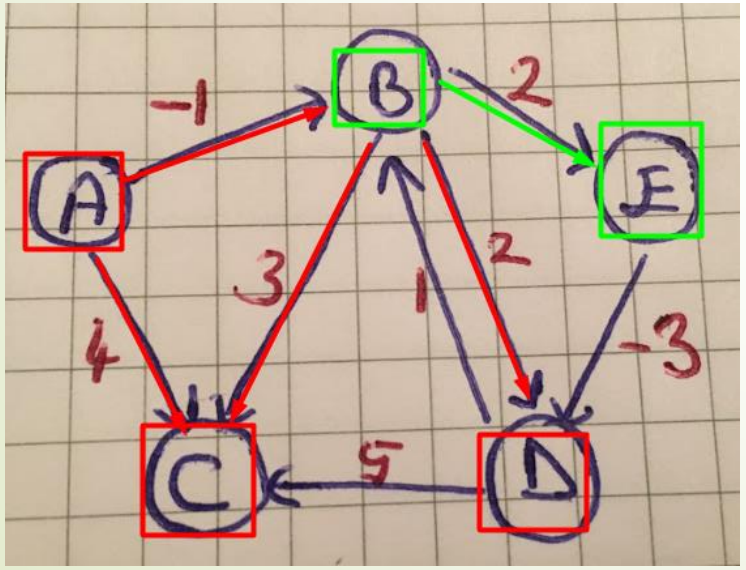
Adım 6:

- **(B,E; 2) :** B düğümünden E düğümüne gitme maliyeti/uzaklığı «2». Bu durumda E düğümüne erişmenin yeni değeri ne olur?

- Kaynak düğüm mevcut değeri **B= -1 ;**
- Kaynak düğümden hedef düğüme gitme maliyeti **(B,E) = 2;**

- B düğümünden başka gidebileceğiz komşu bulunmuyor.
- C düğümüne de baktığımızda gidebileceğimiz bir komşu bulunmuyor.

Şekil6:



Tablo6:

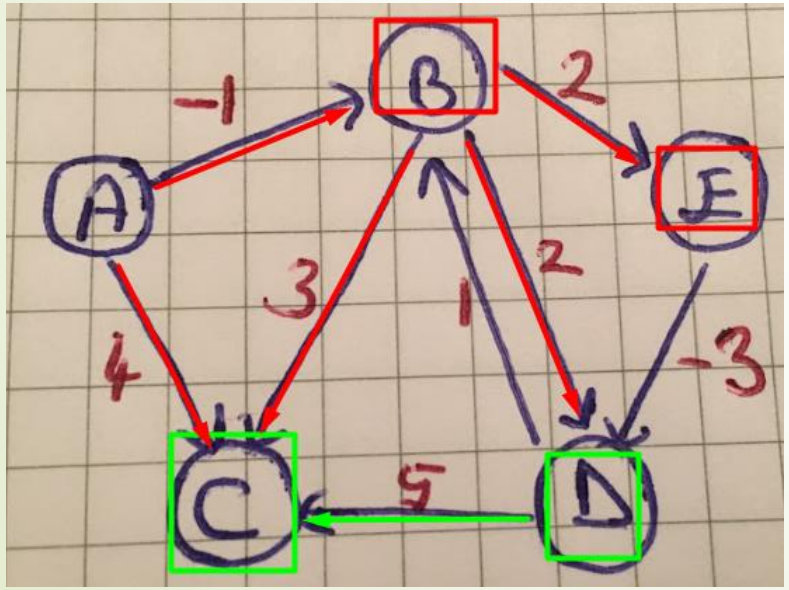
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	1	∞
0	-1	2	1	1

Shortest Path Problem

Adım 7:

- **(D,C; 5) :** D düğümünden C düğümüne gitme maliyeti/uzaklığı «5». Bu durumda C düğümüne erişmenin yeni değeri ne olur?
- Kaynak düğüm mevcut değeri **D= 1 ;**
- Kaynak düğümden hedef düğüme gitme maliyeti **(D,C) = 5;**

Şekil7:



Tablo7:

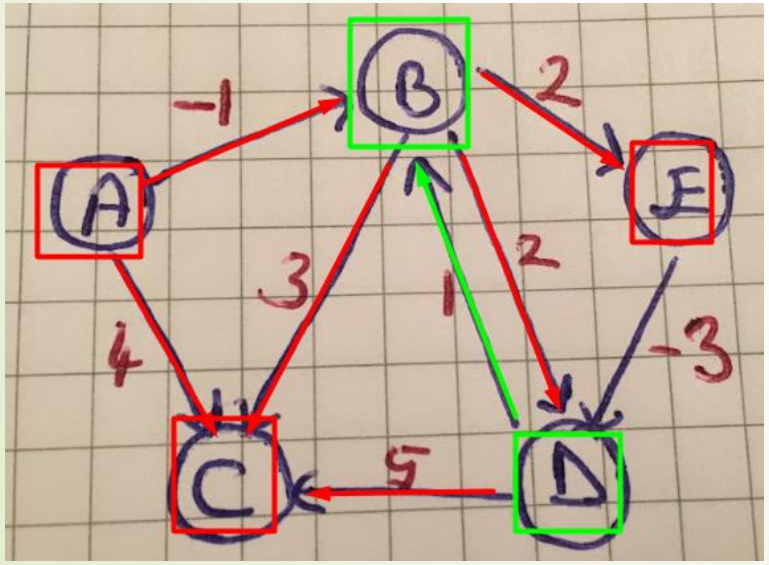
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	1	∞
0	-1	2	1	1

Shortest Path Problem

Adım 8:

- **(D,B; 1) :** D düğümünden B düğümüne gitme maliyeti/uzaklığı «1». Bu durumda B düğümüne erişmenin yeni değeri ne olur?
- Kaynak düğüm mevcut değeri **D= 1 ;**
- Kaynak düğümden hedef düğüme gitme maliyeti **(D,B) = 1;**

Şekil8:



Tablo8:

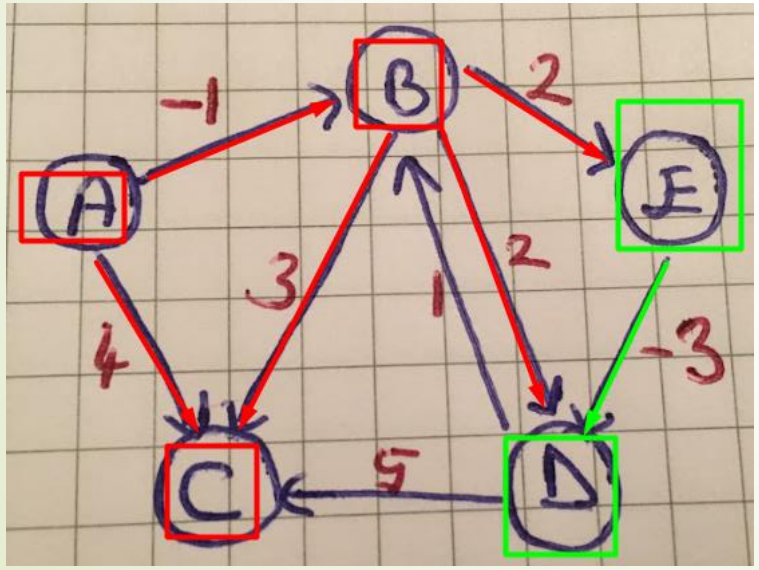
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	1	∞
0	-1	2	1	1

Shortest Path Problem

Adım 9:

- **(E,D; -3) :** E düğümünden D düğümüne gitme maliyeti/uzaklığı «-3». Bu durumda D düğümüne erişmenin yeni değeri ne olur?
- Kaynak düğüm mevcut değeri **E= 1 ;**
- Kaynak düğümden hedef düğüme gitme maliyeti **(E,D) = -3;**

Şekil9:



Tablo9:

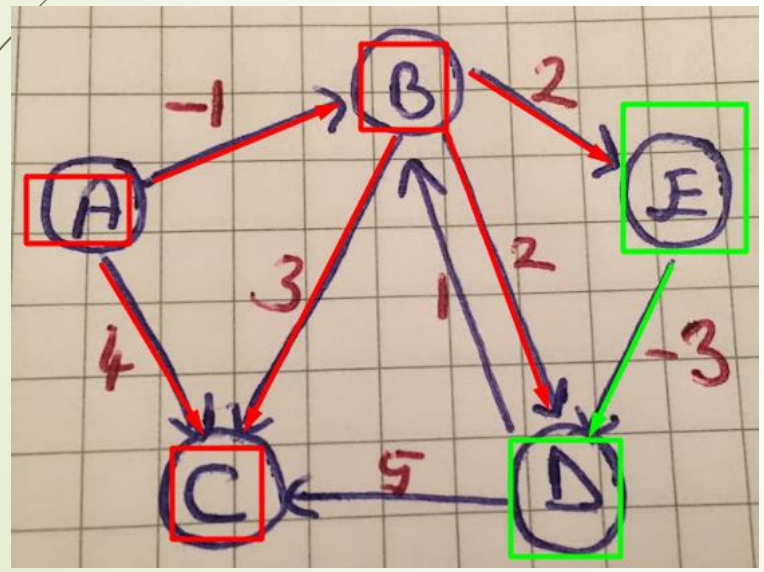
A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	1	∞
0	-1	2	1	1
0	-1	2	-2	1

Shortest Path Problem

Sonuç:

- Kaynak düğümden diğer graf içerisindeki tüm düğümlere erişimin minimum kısa yollarını **Bellman-Ford Algortiması** ile bu şekilde bulabiliyoruz.

Şekil9:



Tablo9:

A	B	C	D	E
0	∞	∞	∞	∞
0	-1	∞	∞	∞
0	-1	4	∞	∞
0	-1	2	∞	∞
0	-1	2	1	∞
0	-1	2	1	1
0	-1	2	-2	1

Piseudo Code

```
for (int i = 1; i <= n; i++) {
    distance[i] = INF;
}
distance[x] = 0;
for (int i = 1; i <= n-1; i++) {
    for (auto e : edges) {
        int a, b, w;
        tie(a, b, w) = e;
        distance[b] = min(distance[b], distance[a]+w);
    }
}
```

Distance[i] -> Kaynak düğüm hariç diğer düğümlerin değeri
Distance[x] -> Kaynak düğümün değeri
a -> Kaynak düğüm
b -> Hedef düğüm
w -> Maliyet

Dijkstra ve Bellman-Ford Algoritması

- Bellman-Ford , Dijkstra algoritmasının iyileştirilmişi olarak düşünülebilir.
- Algoritma aslında Dijkstra algoritmasından daha kötü bir performansa sahiptir ancak graftaki ağırlıkların eksi(-) olması durumunda Dijkstra' nın tersine başarılı çalışır.
- Dijkstra algoritmasında olduğu gibi en küçük değere sahip olan kenardan gitmek yerine bütün graf üzerindeki kenarları test eder. Bu sayede **aç gözlü yaklaşımının(Greedy Approach)** handikabına düşmez ve her düğüme sadece bir kere bakarak en kısa yolu bulmuş olur.

DİNLEDİĞİNİZ İÇİN TEŞEKKÜR EDERİM.