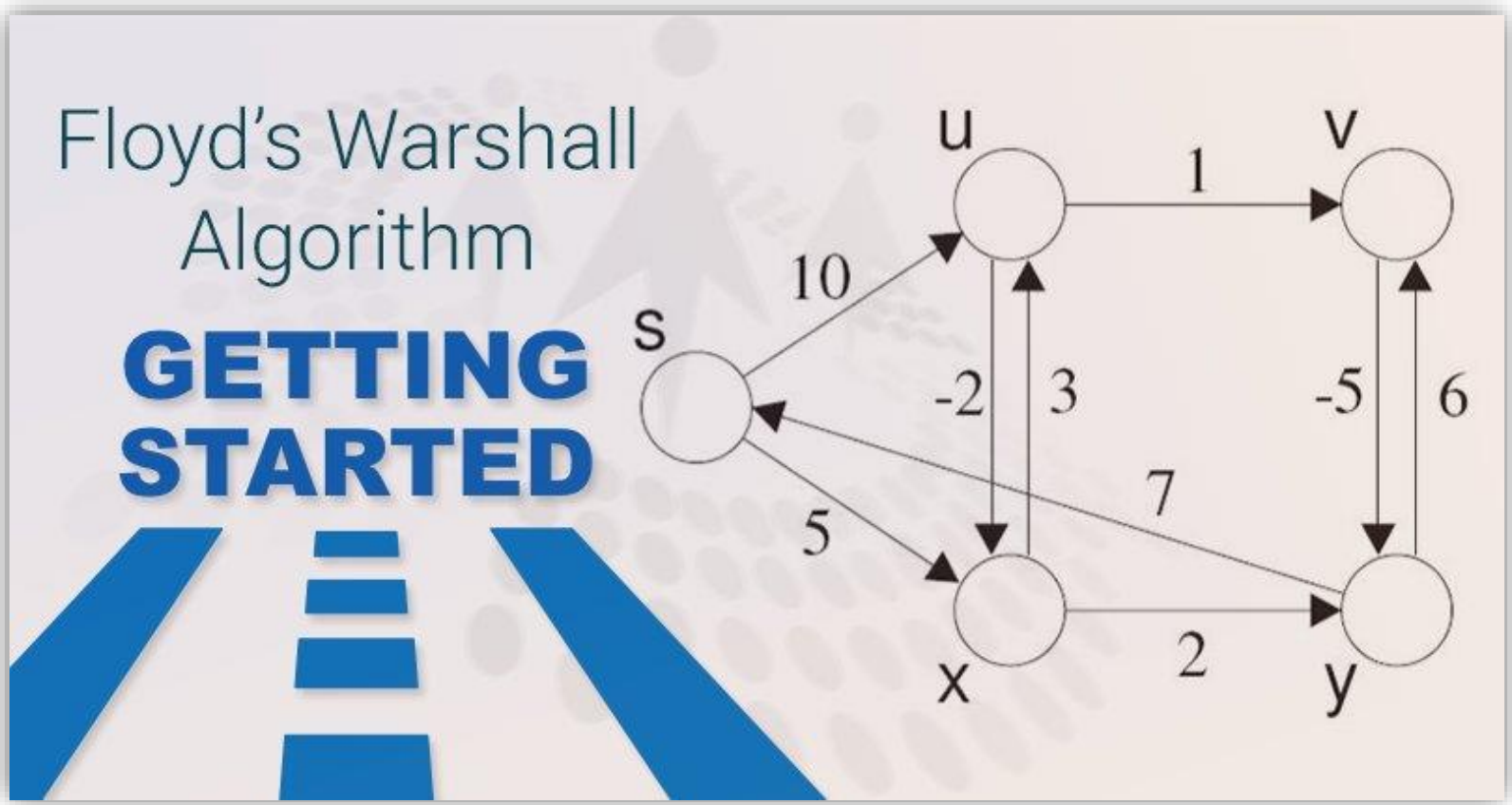
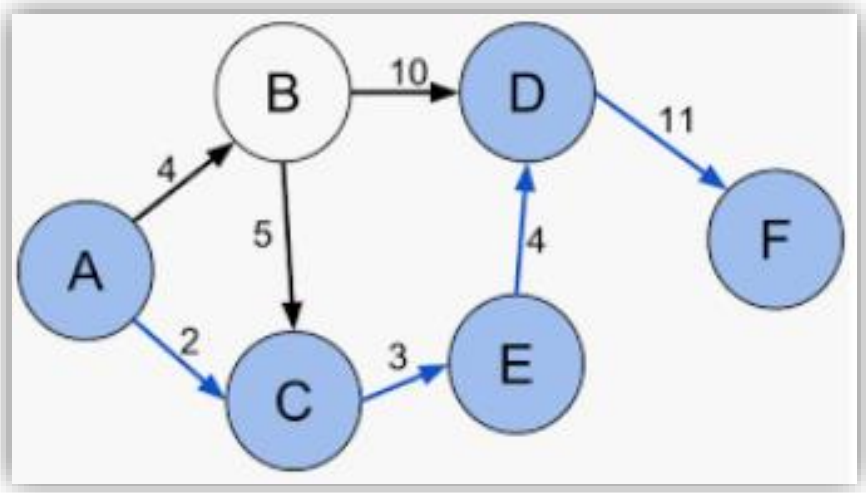

Floyd's Warshall Algorithm
GETTING STARTED
u
v
s
x
y
10
1
-2
3
-5
6
7
5
2

Shortest Path Problem - what?

➢ En kısa yolu **(shortest path)** bulmak, temelde iki düğüm arasında **(node-to-node)** en az maliyetle gidilebilen bir yolun varlığını belirleme problemidir.

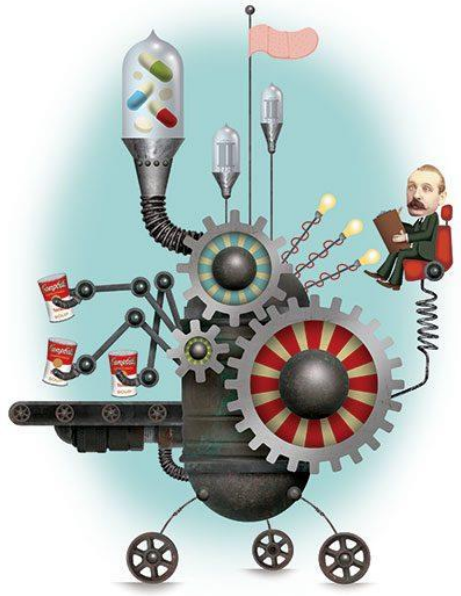
- **Dijktra:** Bir düğümden diğer tüm düğümlere en kısa yollar. (sıfır veya sıfırdan büyük)
- **Bellman ve Ford:** Bir düğümden diğerlerine en kısa yollar (eksi maliyetli graflarda).
- **Floyd Warshall Algorithm:** Tüm düğümler için en kısa yollar.



History and naming – how?

- **Bernard Roy** in 1959
- **Stephen Warshall** in 1962
  - ✓the same as algorithms previously published

    for finding the transitive *closure of a graph*
- **Robert Floyd** in 1962
  - ✓published in its *currently* recognized form
- **Peter Ingerman** in 1962
  - ✓The modern formulation of the algorithm as three nested for-loops was first described by Peter Ingerman



History and naming – how?

- **The algorithm is also known as**
  - *The* **Floyd's algorithm**
  - *The* **Roy–Warshall algorithm**
  - *The* **Roy–Floyd algorithm,** *or*
  - *The* **WFI algorithm**

Floyd Warshall Algorithm

- En kısa yolu bulmak için en genel algoritmadır.
- **Yoğun graflarda** kullanılması daha iyi seçim olur.
- Çok seyrek graflarda **Dijkstra** algoritmasının $|D|$ kez çalıştırılmasıyla daha iyi sonuç elde edilir.


```
for (aradüğüm sayacı < düğüm sayısı) {
     for (satır sayacı < düğüm sayısı) {
          for (sütun sayacı < düğüm sayısı) {
               if (ara düğüm üzerinden gitmek daha kısa ise) {
                    Aradüğümlü maliyeti en küçük maliyet kabul et;
                    Rota bilgisini güncelle;
                }
          }
     }
}
```

Time Complexity

- ✓ **İç içe üç çevrim ile gerçekleştirilen Floyd'un Algoritmasının karmaşıklığı;**
- ✓ $\mathbf{O} = \left(N^3\right)$ **olmaktadır.**

Floyd Warshall Algoritması

- Algoritma, n düğümlü şebekeyi n satırlı ve n sütunlu kare matris olarak gösterir.
- Matrisin (i,j) elemanı, i. düğümden j. düğüme olan d_{ij} uzaklığı verir.
- i doğrudan j'ye bağlı ise d_{ij} sonlu, bağlı değilse sonsuzdur.

D_0 =

	1	2	. . .	j	. . .	n
1	-	d_{12}		d_{1j}		d_{1n}
2	d_{2i}	-	. . .	d_{2j}	. . .	d_{2n}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
i	d_{i1}	d_{i2}	. . .	d_{ij}	. . .	d_{in}
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	d_{n1}	d_{n2}	. . .	d_{nj}	. . .	-

D_0 = Başlangıç uzaklık matrisi

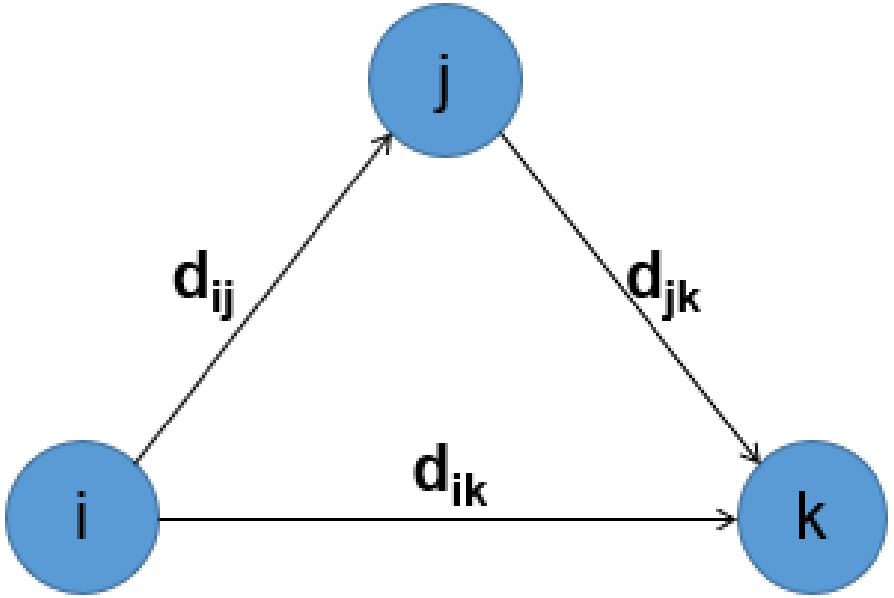
S_0 =

	1	2	. . .	j	. . .	n
1	-	2		j		n
2	1	-	. . .	j	. . .	n
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
i	1	2	. . .	j	. . .	n
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
n	1	2	. . .	j	. . .	-

S_0 = Düğüm sırası matrisi

Floyd Warshall Algoritması

- Eğer $\mathbf{d_{ij}+d_{jk} \leq d_{ik}}$ ise, i'den başlayıp j'den geçerek k'ya ulaşmak daha kısadır.
- Bu durumda i'den k'ya doğrudan yolu**, i→j → k dolaylı yolu ile değiştirmek optimumu verir.**
- Bu **üçlü işlem değişimi,** aşağıdaki adımlar kullanılarak şebekeye sistematik olarak uygulanır.



Floyd Warshall Algoritması

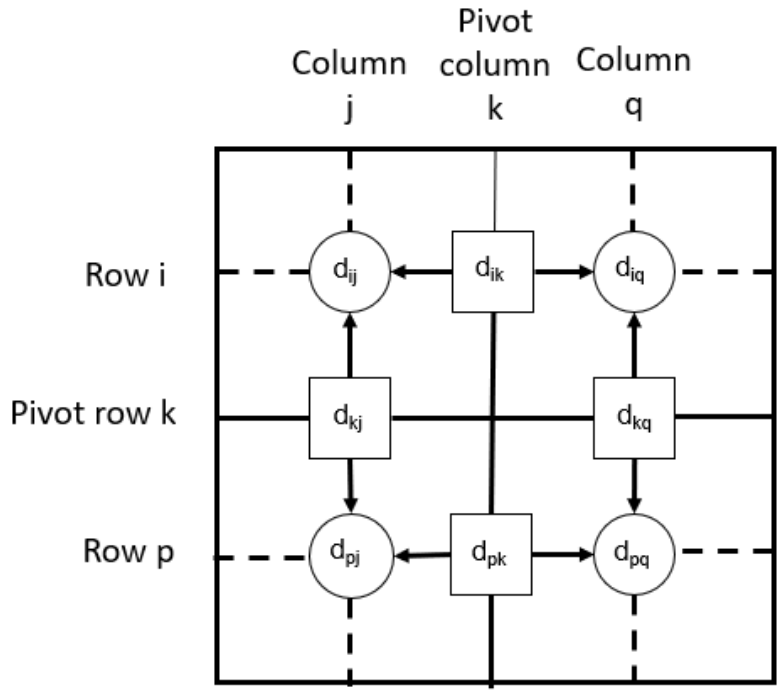
❖ k. Genel Adım

- k. Satırı ve k. Sütunu **anahtar (pivot) satır** ve **anahtar (pivot) sütun** olarak tanımla.
- Tüm i ve j'ler için D_{k-1}'deki her bir d_{ij} elemanına üçlü işlemi uygula. Eğer burada;

  $d_{ik}+d_{kj} \leq d_{ij}$, (i ≠ k, j ≠ k ve i ≠ j) sağlanıyorsa aşağıdaki değişiklikleri yap.

  a) D_{k-1}'de d_{ij}'yi $d_{ik}+d_{kj}$ ile değiştirerek D_k' yı oluşturduk.

  b) S_{k-1}' de S_{ij}' yi k ile değiştirerek S_k' yı oluşturduk. k = k + 1 olarak belirle ve k. adımı tekrar et.



❖ **Eğer anahtar satır ve sütundaki karelerle gösterilen elemanların toplamı, daireyle gösterilen ilgili arakesit elemanından küçükse, arakesit uzaklığı yerine anahtar uzaklıkların toplamını yazmak optimumdur.**

Floyd Warshall Algoritması

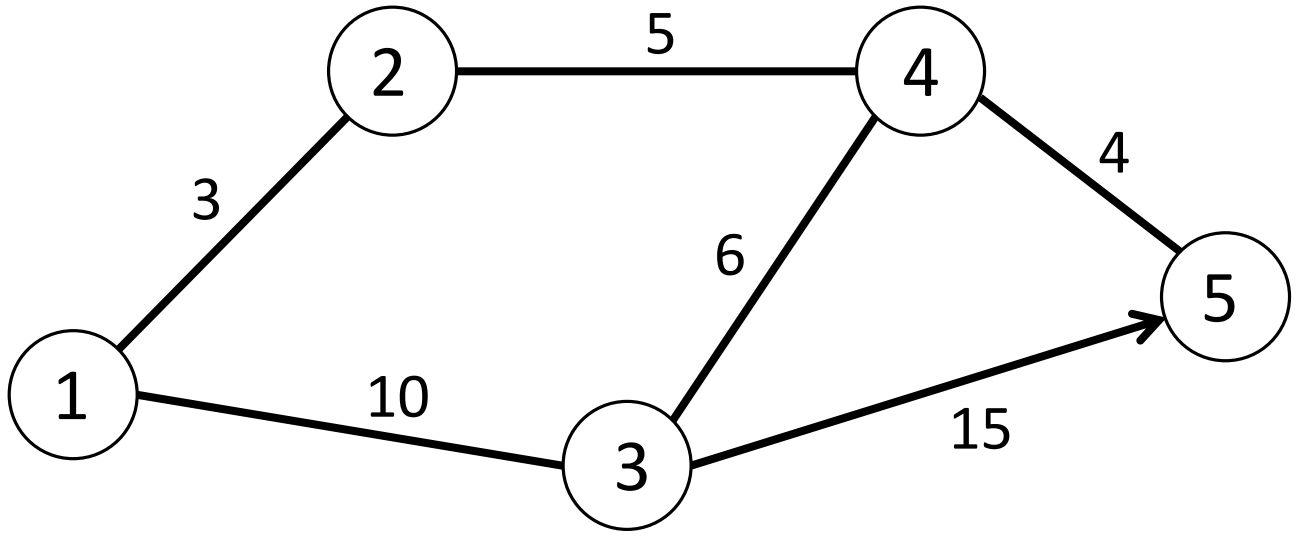
❖ k. Genel Adım

- n adım sonra **i** ve **j** düğümleri arasındaki en kısa yolu $\mathbf{D_n}$ ve $\mathbf{S_n}$ matrislerinden aşağıdaki kuralları kullanarak belirleriz.
- **1.** D_n' deki d_{ij}, i ve j düğümleri arasındaki en kısa yolu verir.
- **2.** S_n'deki **i→j → k** yolunu veren k = S_{ij} ara düğümünü belirle.
- Eğer $\mathbf{S_{ik} = k}$ **ve** $\mathbf{S_{kj} = j}$ **ise dur;** yolun tüm ara düğümleri bulunmuştur.
- Aksi halde i ve k düğümleri ve k ve j düğümleri arasındaki prosedürü tekrar et.

Ek kısa yol problemi - Floyd Warshall Algoritması

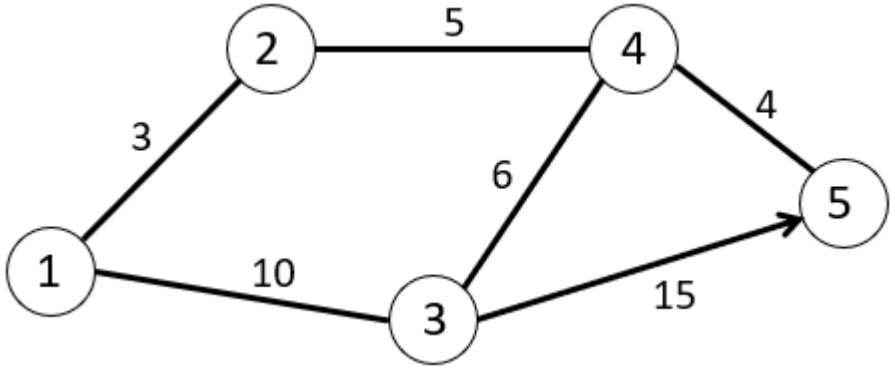
Örnek: Aşağıdaki şebeke için her iki düğüm arasındaki en kısa yolları bulun. Uzaklıklar km cinsinden bağlantıların üzerinde belirtilmiştir.

(3,5) bağlantısı 5. düğümden 3. düğüme trafiğin olmadığı **tek yönlü** bir bağlantıdır. Diğer düğümlerde iki yönde de trafik akışına izin verilmektedir.



Ek kısa yol problemi - Floyd Warshall Algoritması

Example: 0. iteration



D_0

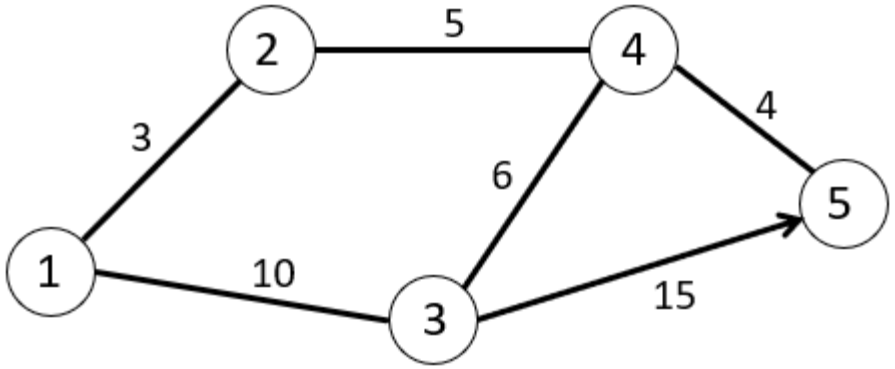
	1	2	3	4	5
1	-	3	10	∞	∞
2	3	-	∞	5	∞
3	10	∞	-	6	15
4	∞	5	6	-	4
5	∞	∞	∞	4	-

S_0

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	3	4	5
3	1	2	-	4	5
4	1	2	3	-	5
5	1	2	3	4	-

Ek kısa yol problemi - Floyd Warshall Algoritması

Example: 1. iteration



D_1

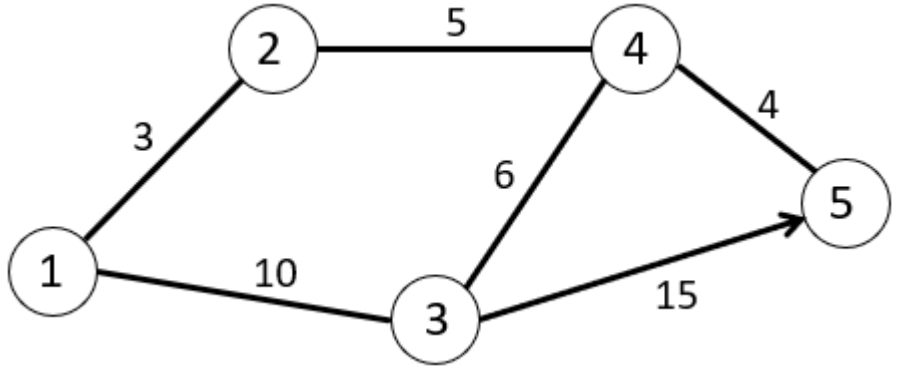
	1	2	3	4	5
1	-	3	10	∞	∞
2	3	-	**13**	5	∞
3	10	**13**	-	6	15
4	∞	5	6	-	4
5	∞	∞	∞	4	-

S_1

	1	2	3	4	5
1	-	2	3	4	5
2	1	-	**1**	4	5
3	1	**1**	-	4	5
4	1	2	3	-	5
5	1	2	3	4	-

Ek kısa yol problemi - Floyd Warshall Algoritması

Example: 2. iteration



D_2

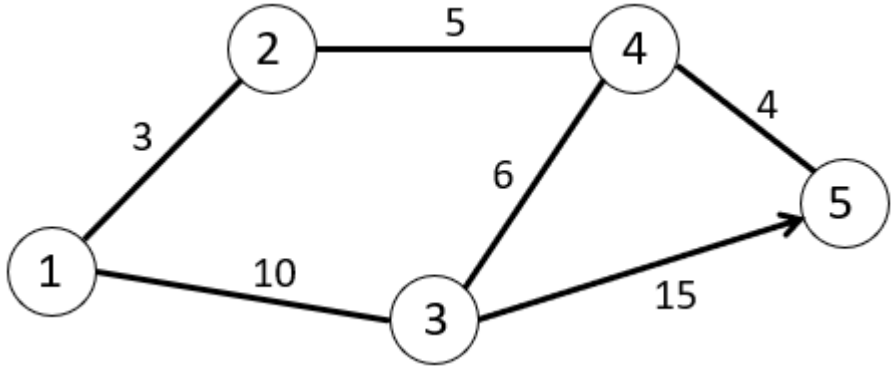
	1	2	3	4	5
1	-	3	10	**8**	∞
2	3	-	13	5	∞
3	10	13	-	6	15
4	**8**	5	6	-	4
5	∞	∞	∞	4	-

S_2

	1	2	3	4	5
1	-	2	3	**2**	5
2	1	-	1	4	5
3	1	1	-	4	5
4	**2**	2	3	-	5
5	1	2	3	4	-

Ek kısa yol problemi - Floyd Warshall Algoritması

Example: 3. iteration



D_3

	1	2	3	4	5
1	-	3	10	8	**25**
2	3	-	13	5	**28**
3	10	13	-	6	15
4	8	5	6	-	4
5	∞	∞	∞	4	-

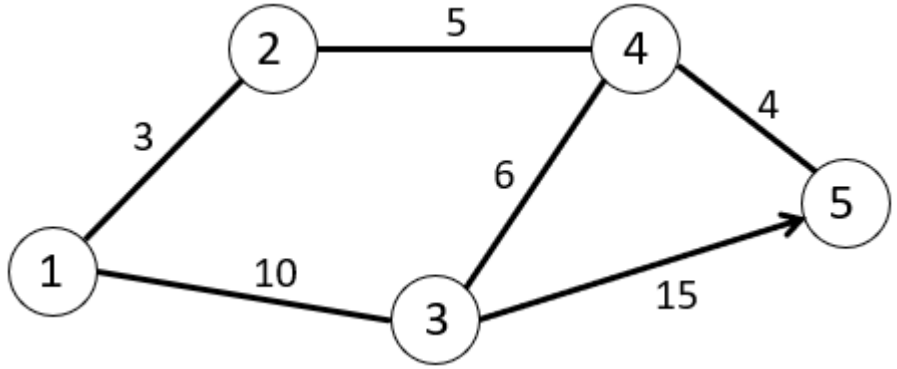
S_3

	1	2	3	4	5
1	-	2	3	2	**3**
2	1	-	1	4	**3**
3	1	1	-	4	5
4	2	2	3	-	5
5	1	2	3	4	-

Ek kısa yol problemi - Floyd Warshall Algoritması

Example: 4. iteration

Herhangi bir iyileşme yoktur.
İşlemler bitmiştir.



D_4

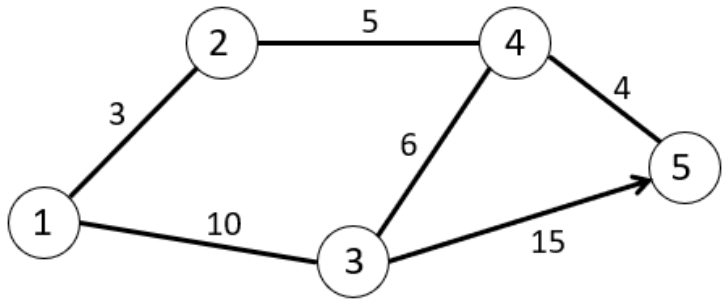
	1	2	3	4	5
1	-	3	10	8	**12**
2	3	-	**11**	5	**9**
3	10	**11**	-	6	**10**
4	8	5	6	-	4
5	**12**	**9**	**10**	4	-

S_4

	1	2	3	4	5
1	-	2	3	2	**4**
2	1	-	**4**	4	**4**
3	1	**4**	-	4	**4**
4	2	2	3	-	5
5	**4**	**4**	**4**	4	-

Ek kısa yol problemi - Floyd Warshall Algoritması

Example: 4. iteration



Herhangi bir iyileşme yoktur.
İşlemler bitmiştir.

D_4

	1	2	3	4	5
1	-	3	10	8	**12**
2	3	-	**11**	5	**9**
3	10	**11**	-	6	**10**
4	8	5	6	-	4
5	**12**	**9**	**10**	4	-

S_4

	1	2	3	4	5
1	-	2	3	2	**4**
2	1	-	**4**	4	**4**
3	1	**4**	-	4	**4**
4	2	2	3	-	5
5	**4**	**4**	**4**	4	-

Örneğin 1 düğümünden 5 düğümüne **shortest path** d_{15} = 12' dir.
Route : $S_{15} = 4$; $S_{14} = 2$; $S_{12} = 2$
O halde: 1'den 5'e yol 1 → 2 → 4 → 5

Kaynaklar

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein - Introduction to Algorithms-MIT Press (2009)

[2] Anany Levitin - Introduction to the Design and Analysis of Algorithms-Pearson (2012)

[3] Harsh Bhasin - Algorithms Design and Analysis-Oxford (2015)

[4] Rıfat Çölkesen – Veri Yapıları ve Algoritmalar – Papatya Yayıncılık (2014)

[5] https://en.wikipedia.org/wiki/Floyd–Warshall_algorithm