8 ŞEBEKE PROBLEMLERİ

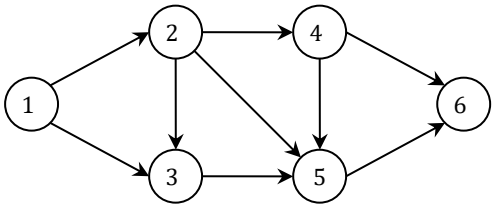
Bazı optimizasyon problemlerinde karar değişkenleri ve kısıtlar arasındaki ilişkiler bir şebekeyle ifade edilebilir. Bu tür problemler şebeke problemleri olarak isimlendirilir. Şebeke problemlerine örnek olarak aşağıdaki problemleri verebiliriz.

- En kısa yol problemi
- Enküçük kapsayan ağaç problemi
- En büyük akış problemi
- Minimum maliyetli akış problemi
- Gezgin satıcı problemi
- Araç rotalama problemi

İzleyen bölümlerde bu problemler ayrıntılı bir biçimde ele alınacaktır.

8.1 Temel Kavramlar

Şebeke düğümler ve bu düğümleri birbirine bağlayan ayrıtlardan oluşur. Düğümler kümesi V, ayrıtlar kümesi E olmak üzere bir şebeke $G(V, E)$ şeklinde gösterilir. Örneğin aşağıdaki şebekede düğümler kümesi $V = \{1, 2, 3, 4, 5, 6\}$ ve ayrıtlar kümesi $E = \{(1,2), (1,3), (2,3), (2,4), (2,5), (3,5), (4,5), (4,6), (5,6)\}$ şeklindedir.

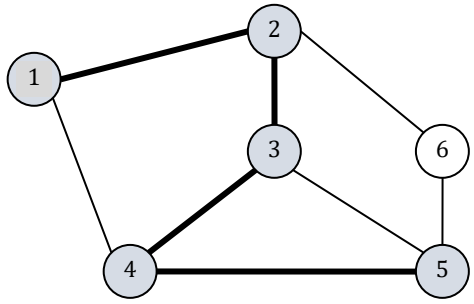


Şebekedeki ayrıtlar yönlü veya yönsüz olabilir. Şebekedeki düğümler ve ayrıtlar probleme göre farklı anlamlar taşır. Tablo 8.1'de farklı problemler için düğümlerin ve ayrıtların ne alama geldiği açıklanmaktadır.

Tablo 8.1 Düğümler, ayrıtlar ve ayrıtların özellikleri

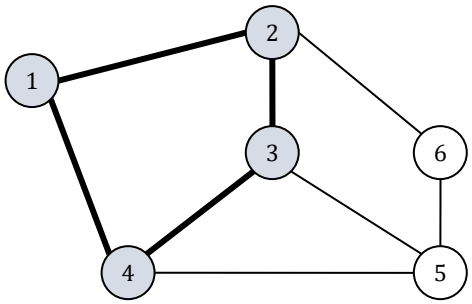
Problem	**Düğümler**	**Ayrıtlar**	**Ayrıtların özellikleri**
Enkısa yol	Şehirler	Yollar	Mesafeler, süre, taşıma maliyeti
Enküçük örten ağaç	Evler	Kablolar, borular	Mesafe, kapasite
Çizelgeleme	Makineler	Rotalar	İşlem süreleri, üretim miktarları

Bir şebekede birbirlerine ayrıtlarla kesintisiz olarak bağlı düğüm sırasına **yol** denir. Örneğin aşağıdaki şebekede 1 - 2 - 3 - 4 - 5 bir 1'den 5'e bir yol iken 1 - 3 - 2 - 6 - 5 bir yol değildir çünkü 1 ile 3 arasında bir bağlantı yoktur.



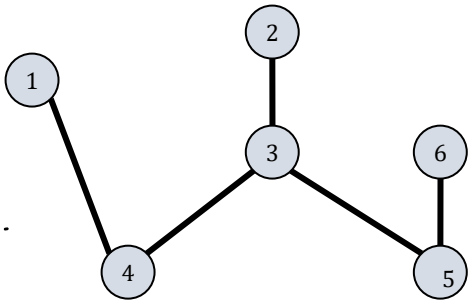
Şekil 8.1 Yol kavramı

Başlangıç ve bitiş düğümü aynı olan yola **döngü** denir. Örneğin 1 - 4 - 3 - 2 - 1 yolu bir döngüdür.



Şekil 8.2 Döngü

Döngü oluşturmayacak şekilde şebekedeki tüm düğümleri birbirine bağlayan ayrıtlar kümesine **ağaç** denir.



Şekil 8.3 Ağaç

Bir şebekede ağaç oluşabilmesi için tüm düğümlerin en az bir ayrıt ile şebekeye bağlı olması gerekir. Bu nedenle şebekede n düğüm var ise ağaç oluşturmak için $n-1$ bir ayrıta bağlantı yapılmalıdır.

8.2 En kısa yol problemi

Yönlü veya yönsüz bir şebekede bir başlangıç düğümü ile bitiş düğümü arasındaki en kısa yolun belirlenmesine en kısa yol problemi denir.

En kısa yol problemi için en iyi çözümü garanti eden farklı yöntemler geliştirilmiştir. Bu yöntemler içinde en çok bilineni Dijkstra algoritmasıdır. Bu algoritmayı açıklamadan önce en kısa yol probleminin kara

r modelini oluşturalım.

n düğümlü bir şebekede başlangıç düğümü 1, bitiş düğümü ise n olarak tanımlayalım. Problemde bir düğümden sonra gidilecek diğer düğüm belirlendiğinden karar değişkenlerini aşağıdaki gibi tanımlayabiliriz.

$$x_{ij} = \begin{cases} 1 & \text{eğer } (i,j) \text{ düğümü kullanılır ise} \\ 0 & \text{diğer durumda} \end{cases}$$

Eğer problemdeki şebekedeki tüm bağlantılar yönsüz ise karar değişkeni sayısını azaltmak için karar değişkenlerini indisler $(i < j)$ şeklinde tanımlayabiliriz.

En kısa yol probleminde amaç başlangıç ve bitiş düğümleri arasında toplam kat edilen mesafenin minimum olmasıdır. d_{ij}: (i,j) düğümleri arasındaki mesafe olarak tanımlandığında amaç fonksiyonu (8.1)'deki gibi yazabiliriz.

$$Enk\ z = \sum_{(i,j)\in E} d_{ij}x_{ij} \qquad \textbf{(8.1)}$$

Problemde üç tip kısıt bulunmaktadır. Birincisi, başlangıç düğümden yalnızca bir düğüme gidilmelidir. Bu kısıtın matematiksel ifadesi (8.2)'deki gibidir.

$$\sum_{k\in L_1} x_{1k} = 1 \qquad \textbf{(8.2)}$$

Burada L_i: i düğümünden gidilebilecek düğümler kümesidir.

İkinci tip kısıt bitiş düğümüne bir düğümden gelinmelidir. Bu kısıtın matematiksel olarak ifadesi (8.3)'deki gibidir.

$$\sum_{k \in A_n} x_{kn} = 1 \qquad \textbf{(8.3)}$$

Burada A_i: i düğümüne gelen düğümler kümesidir

Başlangıç ve bitiş dışında kalan tüm düğümler *transfer* düğümler olarak isimlendirilir. Eğer bir transfer düğümüne bir geliş olmuş ise bu düğümden bir gidiş olmak zorundadır. Örneğin i transfer düğümüne k düğümünden geliş var ise i düğümünden de bir j düğümüne gidiş olmalıdır. Bu kısıtı matematiksel olarak (8.4)'deki gibi yazabiliriz.

$$\sum_{l \in L_i} x_{ij} - \sum_{k \in A_i} x_{ki} = 0 \ \forall i = 2,3, \dots, n-1 \qquad \textbf{(8.4)}$$

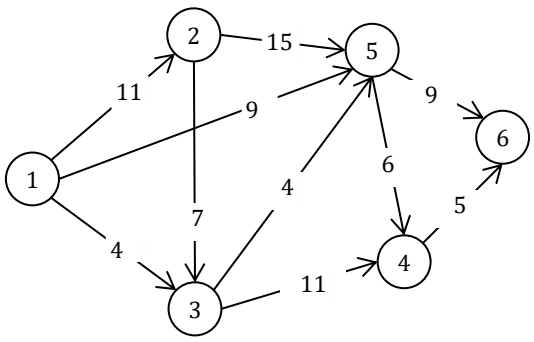
Yönsüz bir şebekede yukarıdaki kısıtlar oluşturulurken sanki küçük indisli düğümden büyük indisli düğüme geliş varmış gibi varsayarak bir düğüme gelen ve giden düğümleri tespit edebiliriz. Örneğin bir şebekede (2, 3) bağlantısı var ise bu hem 2'den 3'e giden bir bağlantı hem de 3'e 2'den gelen bir bağlantı olarak modelde yer almalıdır.

Son olarak, karar değişkenleri 0 veya 1 değerini alması gerekmektedir.

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in E \qquad \textbf{(8.5)}$$

Şimdi aşağıdaki örnek üzerinde enkısa yol probleminin matematiksel modelini oluşturalım.

Örnek 8.1 Aşağıdaki şebekede başlangıç düğümü 1 ve bitiş düğümü 6 olmak üzere en kısa yolu bulan karar modelini yazınız.



Çözüm: Problemin karar modeli aşağıdaki gibidir.

$$Enk\ z = 11x_{12} + 4\,x_{13} + 9x_{15} + 7x_{23} + 15x_{25} + 11x_{34} + 4x_{35} + 5x_{46} + 5x_{54} + 9x_{56}$$

$$\begin{aligned}
-x_{12} - x_{13} - x_{15} &= -1 \quad (1.\text{ düğüm})\\
x_{12} - x_{23} - x_{25} &= 0 \quad (2.\text{ düğüm})\\
x_{13} + x_{23} - x_{34} - x_{35} &= 0 \quad (3.\text{ düğüm})\\
x_{34} + x_{54} - x_{46} &= 0 \quad (4.\text{ düğüm})\\
x_{15} + x_{25} + x_{35} - x_{54} - x_{56} &= 0 \quad (5.\text{ düğüm})\\
x_{46} + x_{56} &= 1 \quad (6.\text{ düğüm})
\end{aligned}$$

Yukarıdaki modelin en iyi çözümünde $x_{13}^* = x_{35}^* = x_{56}^* = 1$ ve $z^* = 17$ olarak bulunmuştur.

8.2.1 Dijkstra Algoritması

Dijkstra algoritması 1959 yılında Edsger Dijkstra tarafından geliştirilmiştir. Algoritmada her yinelemede başlangıç düğümünden bir düğüme en kısa mesafeli yolu bulunmaktadır. Bu işlemlere son düğüme ulaşıncaya kadar devam edilmektedir.

Dijkstra algoritması uygulanırken düğümlere (l_j, i) şeklinde bir etiket verilmektedir. Bu etikete, l_j başlangıç düğümünden bu düğüme toplam mesafeyi, i ise bu düğüme en son hangi düğümden gelindiğini göstermektedir. j düğümüne i düğümünden gelinmiş ise bu düğümün etiketi aşağıdaki gibi hesaplanır.

$$(l_j, i) = (l_i + d_{ij}, i)$$

Algoritmanın başında başlangıç düğümü hariç tüm düğümlere geçici bir etiket verilir. Eğer bir düğüme başlangıç düğümünden en kısa mesafe bulunur ise bu etiket kalıcı hale getirilir.

Dijkstra algoritmasının adımlarını aşağıdaki gibidir.

Adım 1: Başlangıç düğümünü $(0, -)$ şeklinde **kalıcı** olarak, diğer düğümleri $(\infty, -)$ şeklinde geçici olarak etiketle.

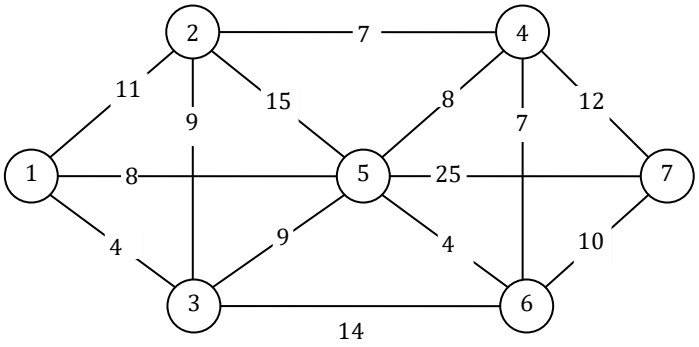
Adım 2: En son kalıcı olarak etiketlenmiş *i.* düğümden erişilebilecek her *j* düğümü için $l_i + d_{ij}$ değerlerini bul. Eğer $l_i + d_{ij} < l_j$ ise j. düğümün etiketini $(l_j, i) = (l_i + d_{ij}, i)$ şeklinde değiştir.

Adım 3: Geçici etiketler içinden minimum l_j değerine sahip düğümün etiketini kalıcı etiket olarak değiştir.

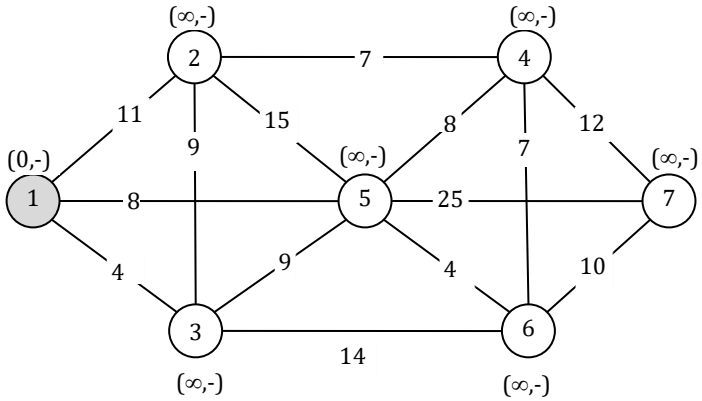
Adım 4: Tüm düğümler kalıcı olarak etiketlenmiş ise dur, aksi takdirde 2. adıma git.

Şimdi Dijkstra algoritmasının adımlarını aşağıdaki yönsüz şebeke üzerinde uygulayalım.

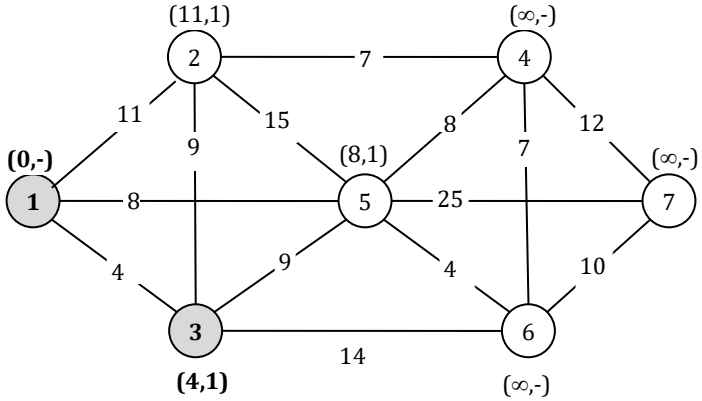
Örnek 8.2 Aşağıdaki şebeke 1'den 7'ye en kısa yolu Dijkstra algoritması ile bulunuz



Birinci düğümü (0,-) şeklinde kalıcı, diğer düğümleri (∞,-) şeklinde geçici olarak etiketleyelim. Şebekede kalıcı etiketli düğümler bundan sonra içi dolu daire olarak gösterilecektir.

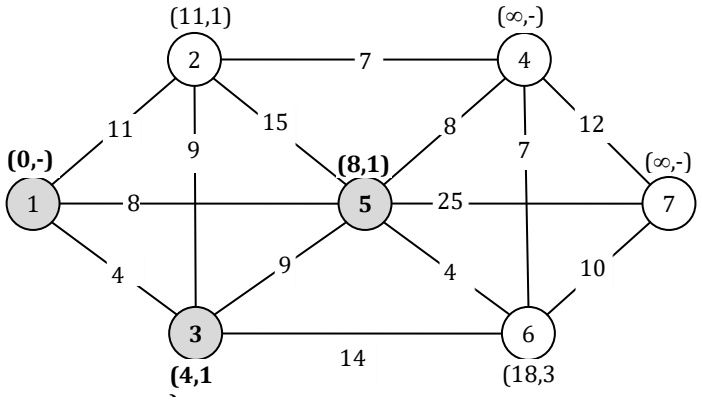


Kalıcı etiketli birinci düğümden direkt olarak erişilebilecek geçici etiketli düğümler kümesi {2, 3, 5}. Birinci düğümden ikinci düğüme gidilir ise toplam mesafe $l_1 + d_{12} = 0 + 11 < \infty$ olduğundan ikinci düğümün geçici etiketi (11,1) olarak değiştirilir. Benzer şekilde, üçüncü düğüm için $l_1 + d_{13} = 0 + 4 < \infty$ olduğundan üçüncü düğümün etiketi (4,1) olarak değiştirilir. Beşinci düğüm için $l_1 + d_{15} = 0 + 8 < \infty$ olduğundan beşinci düğümün etiketi (8,1) olarak değiştirilir. Geçici etiketler içinden en küçük mesafeli üçüncü düğüm kalıcı olarak etiketlenir.

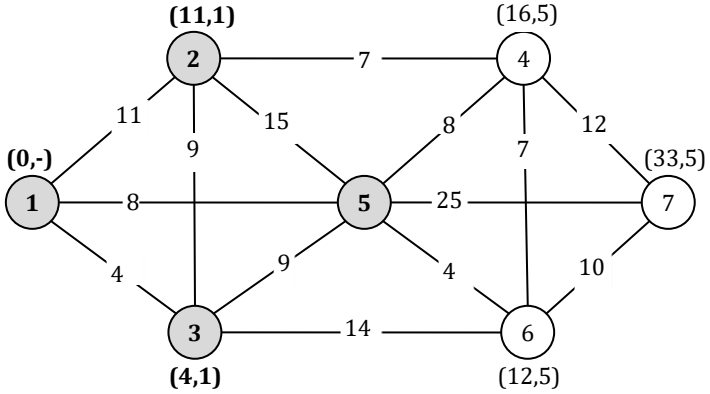


Kalıcı etiketli üçüncü düğümden erişilebilecek geçici etiketli düğümler kümesi {2, 5, 6}. $l_3 + d_{32} = 4 + 9 > 11$ olduğundan ikinci düğümün etiketi değiştirilmez. $l_3 + d_{35} = 4 +$

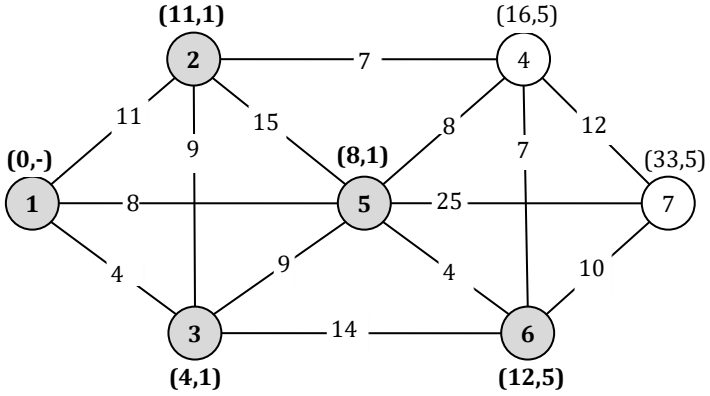
$9 > 8$ olduğundan beşinci düğümün etiketi değiştirilmez. $l_3 + d_{36} = 4 + 14 < \infty$ olduğundan altıncı düğümün etiketi (18, 3) olarak değiştirilir. Geçici etiketler içinden en küçük mesafeli beşinci düğüm kalıcı olarak etiketlenir.



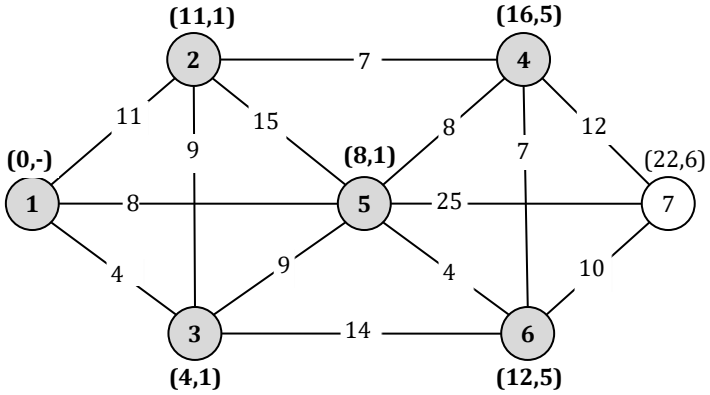
Kalıcı etiketli beşinci düğümden erişilebilecek geçici etiketli düğümler kümesi {2, 4, 6, 7}. ikinci düğüm için $l_5 + d_{52} = 8 + 15 > 11$ olduğundan ikinci düğümün etiketi değişmez. Dördüncü düğüm için $l_5 + d_{54} = 8 + 8 < \infty$ olduğundan dördüncü düğümün etiketi (16,5) olarak değiştirilir. Altıncı düğüm için $l_5 + d_{56} = 8 + 4 < 18$ olduğundan altıncı düğümün etiketi (12, 5) olarak değiştirilir. Yedinci düğüm için $l_5 + d_{57} = 8 + 25 < \infty$ olduğundan yedinci düğümün etiketi (33, 5) olarak değiştirilir. Geçici etiketler içinden en küçük mesafeli ikinci düğüm kalıcı olarak işaretlenir.



Kalıcı etiketli ikinci düğümden erişilebilecek geçici etiketli düğümler kümesi {4}. $l_2 + d_{24} = 11 + 7 > 16$ olduğundan dördüncü düğümün etiketi değişmez. Geçici etiketler içinden en küçük mesafeli altıncı düğüm kalıcı olarak işaretlenir.

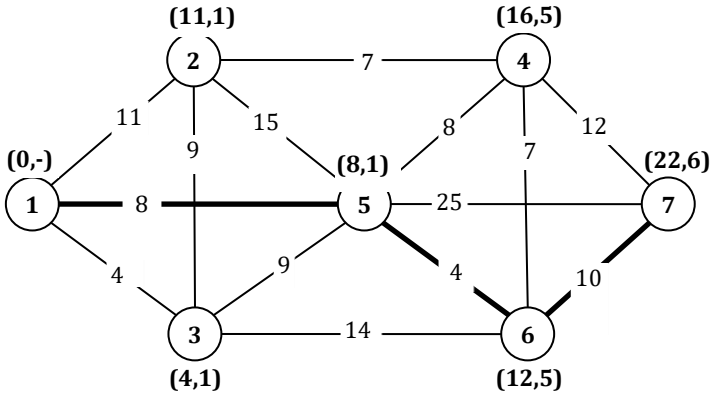


Kalıcı etiketli altıncı düğümünden erişilebilecek geçici etiketli düğümler kümesi {4, 7}. Dördüncü düğüm için $l_6 + d_{64} = 12 + 7 > 16$ olduğundan dördüncü düğümün etiketi değiştirilmez. Yedinci düğüm için $l_6 + d_{67} = 12 + 10 < 33$ olduğundan yedinci düğümün etiketi (22, 6) olarak değiştirilir. Geçici etiketler içinden en küçük mesafeli dördüncü düğüm kalıcı olarak işaretlenir.



Kalıcı etiketli dördüncü düğümden erişilebilecek geçici etiketli düğümler kümesi {7}. $l_4 + d_{47} = 16 + 12 = 28 > 22$ olduğundan yedinci düğümün geçici etiketi değiştirilmez. Sadece yedinci düğüm geçici etikete sahip olduğundan bu düğüm kalıcı olarak işaretlenir ve algoritma sonlandırılır.

En kısa yol bulunurken bitiş düğümden geriye doğru kalıcı etiketli düğümler dikkate alınarak başlangıç düğümüne gidilir. Problemde 7-6-5-1 kalıcı etiketli düğümleri en kısa yolu oluşturmaktadır. Bu yolun toplam mesafesi 22 birimdir. Şekil 8.4'da en kısa yol kalın çizgiler ile gösterilmektedir.

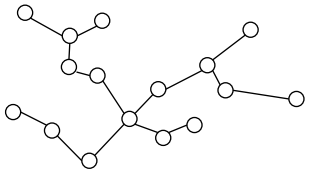


Şekil 8.4 En kısa yol

8.3 Minimum Kapsayan Ağaç Problemi

Yönsüz bir şebekede toplam bağlantı uzunluğunu minimum olacak şekilde tüm düğümlere bağlantı yapılmasına minimum kapsayan ağaç (MKA) (minimum spanning tree, MST) problemi denir. MKA'ya örnek olarak bir şehirde toplam boru hattı uzunluğunun minimum olacak şekilde tüm mahallelere yapılacak doğalgaz bağlantılarının belirlenmesi problemini verebiliriz.

Tüm düğümler birbirine bağlandığında aşağıdaki gibi bir ağaç yapısı oluştuğundan bu probleme enküçük kapsayan ağaç problemi denilmektedir.



Şekil 8.5 Kapsaya ağaç yapısı

Yukarıdaki şekilden de görüldüğü gibi, n düğümlü bir şebekede tüm düğümleri birbirine bağlamak için $n-1$ bağlantıya ihtiyaç vardır ve bu bağlantılar arasında bir döngü oluşturmamalıdır.

E ayrıtlar kümesi, V düğümler kümesi ve d_{ij}: i, j ayrıtının uzunluğu ve karar değişkenleri

$$x_{ij} = \begin{cases} 1 & \text{eğer } i \text{ ve } j \text{ düğümleri bağlanır ise} \\ 0 & \text{diğer durumda} \end{cases}$$

olmak üzere problemin matematiksel modeli (8.6)-(8.9) 'daki gibidir. 8.6'daki amaç fonksiyonunda toplam bağlantı uzunluğu minimum yapılmak istenmektedir.

$$Enk\ z = \sum_{(i,j)\in E} d_{ij}x_{ij} \qquad \textbf{(8.6)}$$

MKA probleminde bağlantıların bir ağaç oluşturması için toplam bağlantı sayısının (n-1) adet olması gerekir. Bu kısıt (8.7)'deki gibi ifade edilir

$$\sum_{(i,j)\in E} x_{ij} = n-1 \qquad \textbf{(8.7)}$$

Problemdeki ikinci tip kısıtlar döngüleri engelleme kısıtlarıdır. Örneğin üç düğüm arasında 1'den 2'ye ve 2'den 3'e olmak üzere 2 bağlantı yapıldığında başka bir bağlantı yapılmaya gerek yoktur, aksi takdirde döngü oluşur. Bu nedenle, üç düğümü içeren bir alt küme için bağlantı sayısı 2 olmak zorundadır. Şebekede olası tüm döngüleri engellemek için V kümesinin tüm alt kümeleri için bu koşulun sağlanması gerekir. S: V kümesinin herhangi bir alt kümesi olmak üzere döngü engelleme kısıtı (8.8)'deki gibi yazılır.

$$\sum_{(i,j)\in E: i\in S, j\in S} x_{ij} = |S|-1 \qquad \forall S \subseteq V \qquad \textbf{(8.8)}$$

V kümesinin alt küme sayısı $2^{|V|} - 1$ olduğundan (8.8)'deki kısıt sayısı şebekedeki düğüm sayısına bağlı olarak üstel bir şekilde artmaktadır. Bu nedenle büyük problemlerde modelinin oluşturulması dahi imkânsız hale gelmektedir.

Son olarak karar değişkenleri 0 veya 1 değeri almak zorundadır.

$$x_{ij} \in \{0,1\} \forall (i,j) \in E \qquad \textbf{(8.9)}$$

Problemdeki kısıt sayısı üstel bir şekilde arttığından problemin çözümünü bulmak için en iyi çözümü garanti eden sezgisel yöntemler kullanmaktadır. Bu yöntemlerden Prim ve Kruskal algoritması izleyen bölümlerde ayrıntılı olarak açıklanmaktadır.

8.3.1 Prim (Jarnik) Algoritması

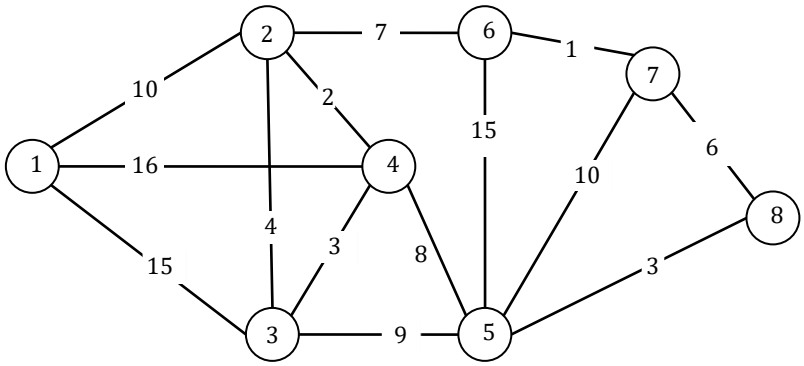
Prim algoritması düğüm bazlı bir algoritma olup her yinelemede en kısa mesafeli bir düğüme bağlantı yapılır. Aşağıda Prim algoritmasının adımları verilmektedir.

Adım 1: Rastgele bir düğümü seç, bu düğümü bağlanmış düğümler kümesine ata.

Adım 2: Bağlanmış düğümler kümesinden erişilebilecek düğümler kümesini belirle. En kısa mesafeli düğüme bağlantı yap ve bu düğümü bağlanmış düğümler kümesini ata.

Adım 3: Tüm düğümlere bağlantı yapılmış ise dur. Değilse 2. adıma git.

Örnek 8.4 Bir doğalgaz şirketi bir şehirde doğal gaz akışını sağlamak için 8 farklı bölgeye regülatör kurmak istemektedir. Şekil 8.6'da regülatörlerin kurulacağı sekiz yer ve bu yerler arasındaki mesafeler aşağıdaki şebekede verilmektedir. Toplam çelik boru uzunluğu minimum olacak şekilde regülatörler arasında nasıl bir bağlantı yapılmalıdır?



Şekil 8.6 Doğal gaz regülatörleri arasındaki potansiyel bağlantılar ve mesafeler

Bu problemi Prim algoritması ile çözelim. Tablo 8.2'deki algoritmanın her yinelemesinde yapılan bağlantılar gösterilmektedir. İlk iki yinelemeyi örnek olarak açıklayalım. Algoritmanın birinci adımında rastgele bir düğüm seçilir, örneğin 4. düğümü seçelim ve bu düğümü bağlanmış düğümler kümesine atayalım. İkinci adımda, bağlanmış düğüm kümesinden direkt olarak erişilebilecek düğümler kümesi {1, 2, 3, 5} olarak belirlenir. Bağlanmış düğümler ve erişilebilecek düğümler arasındaki en kısa mesafe 4. ve 2.

düğümleri arasında olduğundan bu iki düğüm birbirine bağlanır ve bağlanmış düğümler kümesi {2, 4} olarak belirlenir. Üçüncü adımda tüm düğümlere bağlantı yapılmadığından ikinci adıma gidilir. İkinci adımda, erişilebilecek düğümler kümesi {1, 3, 5, 6} olarak belirlenir. Bağlanmış düğümler ve erişilebilecek düğümler arasındaki en kısa mesafe 4 ve 3 düğümeler arasında olduğundan bu bağlantı yapılır ve bağlanmış düğümler kümesi {4, 2, 3} olarak güncellenir.

Tablo 8.2 Prim algoritması

Bağlantı Yapılmış Düğümler Kümesi	Erişilebilir düğümler kümesi	Bağlantı yapılan ayrıt	Mesafe (km)
{4}	{1, 2, 3, 5}	(2, 4)	2
{4, 2}	{1, 3, 5, 6}	(3, 4)	3
{4, 2, 3}	{1, 5, 6}	(2, 6)	7
{4, 2, 3, 6}	{1, 5, 7}	(4, 5)	8
{4, 2, 3, 5, 6}	{1, 7, 8}	(5, 8)	3
{4, 2, 3, 5, 6, 8}	{1,7}	(7, 8)	6
{4, 2, 3, 5, 6, 8, 7}	{1}	(1, 2)	10
{4, 2, 3, 5, 6, 8, 7, 1}	{ }	-	

Bu bağlantılar sonucunda toplam çelik boru uzunluğu 39 km olarak bulunur.

8.3.2 Kruskal Algoritması

Kruskal algoritması ayrıt temelli bir algoritmadır. Algoritmada en kısa mesafeli ayrıttan başlanır döngü oluşturmayacak şekilde sonraki en kısa ayrıta bağlantı yapılarak bir ağaç oluşturulur. Algoritmanın adımları aşağıdaki gibidir.

Adım 1: Ayrıtları küçükten büyüğe sırala

Adım 2: Tüm düğümler için ayrı bir küme oluştur

Adım 3: En kısa ayrıtı seç. Eğer bu ayrıttaki düğümler aynı kümenin elemanı ise 4. adıma git, diğer durumda bu iki düğümü birbirine bağla ve kümeleri birleştir

Adım 4: Tüm düğümlere bağlantı yapıldı ise dur değilse 3. adıma git.

Şimdi Örnek 8.4'ü Kruskal algoritmasıyla çözelim. Önce ayrıtları uzunluklarına göre küçükten büyüğe sıralayalım

(2,4), (3,4), (5,8), (2,3), (7,8) (2,6), (4,5) (3,5) (1,2), (5,7) (6,7) (1,3) (5,6)

Her düğüm için aşağıdaki kümeleri oluşturalım.

{1} {2} {3} {4} {5} {6} {7} {8}

(2, 4) ayrıtı için 2 ve 4 iki farklı kümeye ait olduğundan bu iki düğüme bağlantı yapalım ve kümeleri aşağıdaki gibi birleştirelim.

{1} {2, 4} {3} {5} {6} {7} {8}

(3, 4) ayrıtı için 3 ve 4 iki farklı kümeye ait olduğundan bu iki düğüme bağlantı yapalım ve kümeleri aşağıdaki gibi birleştirelim.

{1} {2, 4, 3} {5} {6} {7} {8}

(5, 8) ayrıtı için 5 ve 8 iki farklı kümeye ait olduğundan bu iki düğüme bağlantı yapalım ve kümeleri aşağıdaki gibi birleştirelim.

{1} {2, 4, 3} {5, 8} {6} {7}

(2, 3) ayrıtı için 2 ve 3 aynı kümeye ait olduğundan bu ayrıt atlayalım. (7,8) ayrıtı için 7 ve 8 iki farklı kümeye ait olduğundan bu iki düğüme bağlantı yapalım ve kümeleri aşağıdaki gibi birleştirelim.

{1} {2, 4, 3} {5, 8, 7} {6}

(2, 6) ayrıtı için 2 ve 6 iki farklı kümeye ait olduğundan bu iki düğüme bağlantı yapalım ve kümeleri aşağıdaki gibi birleştirelim.

{1} {2, 4, 3, 6} {5, 8, 7}

(4, 5) ayrıtı için 4 ve 5 iki farklı kümeye ait olduğundan bu iki düğüme bağlantı yapalım ve kümeleri aşağıdaki gibi birleştirelim.

{1} {2, 4, 3, 6, 5, 8, 7}

(3, 5) ayrıtı için 3 ve 5 aynı kümeye ait olduğundan bu ayrıt atlayalım. (1, 2) ayrıtı için 1 ve 2 iki farklı kümenin elemanı olduğundan bu iki düğüme bağlantı yapalım ve kümeleri aşağıdaki gibi birleştirelim.

{1, 2, 4, 3, 6, 5, 8, 7}

Tüm düğümlere bağlantı yapıldığı için algoritma sonlandırılır. Bağlantı yapılan ayrıtlar (2,4), (3,4), (5,8) (7,8), (2,6), (4,5) (1,2) olup toplam çelik boru uzunluğu 39 km olarak bulunur.

8.4 En Büyük Akış Problemi

Yönlü veya yönsüz bir şebekede bir kaynak (arz) düğümünden talep düğümüne yapılabilecek maksimum akış miktarının belirlenmesine en büyük akış problemi denir. Bir petrol boru hattından taşınabilecek maksimum petrol miktarının belirlenmesi, iki nokta arasında taşınabilecek maksimum yolcu sayısının belirlenmesini enbüyük akış problemine örnek olarak verebiliriz.

İlk önce bu problemin karar modelini yazalım.

Karar değişkenleri: Problemde iki tip karar değişkeni bulunmaktadır. Birincisi şebekedeki her bir ayrıttan yapılacak akış miktarlarının ve ikincisi ise şebekedeki maksimum akış miktarının belirlenmesidir. Şebekedeki maksimum akış miktarı arz

düğümünden çıkan akışların toplamı ve talep düğümüne gelen akışların toplamıdır. Bu karar değişkenlerini aşağıdaki gibi tanımlayalım.

x_{ij}: (i, j) ayrıtındaki akış miktarı

v: Maksimum akış miktarı

Amaç fonksiyonu: Problemde maksimum akış miktarı belirlenmek istendiğinden amaç fonksiyonunu aşağıdaki gibi yazabiliriz.

$$Enb\ z = v \qquad \textbf{(8.5)}$$

Kısıtlar: Problemde dört tip kısıt bulunmaktadır. Birincisi kaynak düğümünden çıkan akışların toplamı, şebekedeki maksimum akış miktarına eşit olmalıdır. Kaynak düğümünü birinci düğüm olarak aldığımızda bu kısıtı matematiksel olarak aşağıdaki gibi yazabiliriz.

$$\sum_{j \in L_1} x_{1j} = v \qquad \textbf{(8.6)}$$

Burada L_1: Kaynak düğümünden çıkan ayrıtların kümesidir.

İkinci tip kısıt talep düğümüne gelen akışların toplamı, şebekedeki maksimum akış miktarına eşit olmalıdır. Son düğümü talep düğümü olarak alırsak matematiksel olarak bu kısıtı aşağıdaki gibi yazabiliriz.

$$\sum_{j \in E_n} x_{jn} = v \qquad \textbf{(8.7)}$$

Burada E_n: Son düğüme gelen akışlar kümesidir.

Üçüncü tip kısıtlar ara düğümlerle ilgilidir. Bir ara düğüme gelen akışların toplamı bu düğümden çıkan akışların toplamına eşit olmalıdır. Matematiksel olarak bu kısıtları aşağıdaki gibi yazabiliriz.

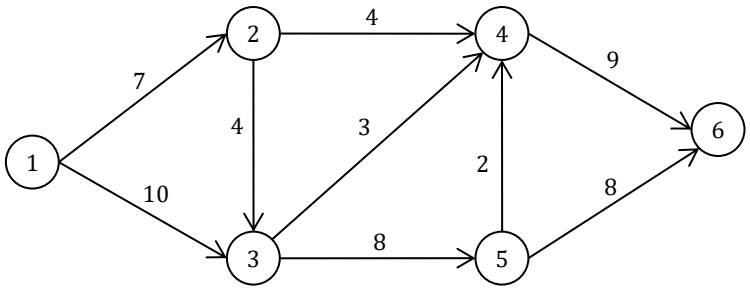
$$\sum_{k \in E_i} x_{ki} = \sum_{j \in L_i} x_{ij} \quad \forall i \neq 1, n \qquad \textbf{(8.8)}$$

Dördüncü kısıt kapasitelerle ilgilidir. Bir ayrıttaki akış miktarı o ayrıtın kapasitesini aşmamalıdır. b_{ij}: (i, j) ayrıtının kapasitesi olmak üzere kapasite kısıtlarını aşağıdaki gibi yazabiliriz.

$$0 \leq x_{ij} \leq b_{ij} \qquad \textbf{(8.9)}$$

Aşağıdaki örnek üzerinde problemin matematiksel modelini yazalım.

Örnek 8.5 Aşağıdaki şebekede 1 no.lu düğüm kaynak ve 6 no.lu düğüm talep düğümü olmak üzere bu şebekede yapılabilecek enbüyük akış miktarı nedir? Problemin doğrusal karar modelini yazınız.



Çözüm: Yukarıda yaptığımız açıklamalar doğrultusunda problemin karar modelinin kısıtları ve amaç fonksiyonunu yazalım.

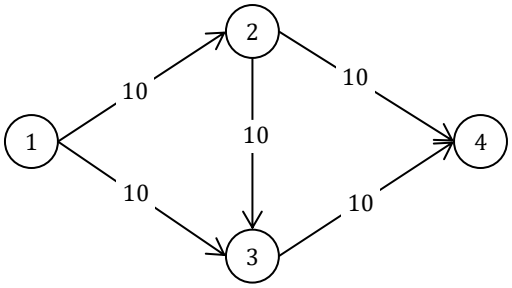
$x_{12} + x_{13} = v$	(Kaynaktan çıkan akış miktarı=Maksimum akış miktarı)
$x_{12} = x_{23} + x_{24}$ $x_{13} + x_{23} = x_{34} + x_{35}$ $x_{24} + x_{34} + x_{54} = x_{46}$ $x_{35} = x_{54} + x_{56}$	(Ara düğümlere gelen akışların toplamı ve bu düğümlerden çıkan akışların toplamına eşit olmalıdır)
$x_{46} + x_{56} = v$	(Talep düğümüne gelen akış miktarı=Maksimum akış miktarı)
$0 \leq x_{ij} \leq b_{ij}$	Kapasite kısıtları
$Enb\ z = v$	Amaç fonksiyonu: Toplam akışı maksimum yapmak

Problemin en iyi çözümünde $v^* = 15, x_{12}^* = 7, x_{13}^* = 8, x_{23}^* = 3, x_{24}^* = 4, x_{34}^* = 3, x_{35}^* = 8, x_{46}^* = 7, x_{56}^* = 8$ olarak bulunur.

8.4.1 Ford-Fulkerson Algoritması

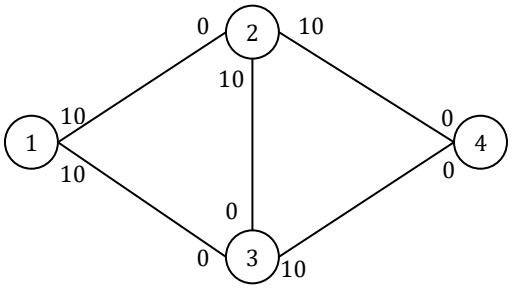
En büyük akış probleminin çözümü için farklı algoritmalar geliştirilmiştir. Bunlar içinde en çok bilinen algoritma Ford-Fulkerson algoritmasıdır (FFA). FFA oldukça basit bir mantığa dayanmaktadır. Algoritmanın her yinelemesinde, kaynak düğümünden talep düğümüne akış miktarı pozitif olan bir güzergâh bulunur ve bu güzergâhlardan gönderilebilecek akış miktarları toplanarak şebekedeki maksimum akış miktarı belirlenir.

Bu algoritmayı basit bir örnek üzerinde açıklamaya çalışalım. Aşağıdaki verilen şebekede 1 kaynak düğümü, 4 ise talep düğümü olsun.



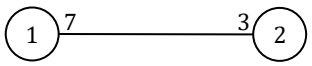
Algoritmada ilk olarak kaynak düğümünden talep düğümüne akış miktarı pozitif olan bir güzergâh belirlenir. Daha sonra bu güzergâhtan yapılabilecek maksimum akış miktarı belirlenir. Örneğin 1-2-4 güzergâhını seçelim. Bu güzergahtan maksimum 10 birim akış yapılabilir. Bu akış yapıldığında (1,2) ve (2,4) ayrıtlarının kapasitelerinin tamamı kullanıldığından bu ayrıtlarda kalan kapasiteler sıfır olur. Daha sonra şebekede kalan kapasiteler dikkate alınarak kaynak düğümünden talep düğümüne akış miktarı pozitif olan bir güzergâh bulunur. Örneğin 1-3-4 güzergâhı, bu güzergahta maksimum 10 birim akış gönderilebilir, böylece (1,3), (3,4) ayrıtlarının kapasitelerinin tamamı kullanılmış olur ve kalan kapasiteler sıfır olur. Kaynak düğümünden talep düğümüne akış miktarı pozitif olan başka bir güzergâh kalmadığından algoritma sonlandırılır. Şebekedeki maksimum akış miktarı her bir güzergahtan yapılan akışların toplamıdır. Bu şebekeden maksimum 10+10=20 birim akış yapılabilir.

Algoritmanın bu noktaya kadar açıklanan kısmında akış miktarları pozitif olan güzergâhlar ve bu güzergahlardan yapılabilecek akışları belirledik. Eğer güzergahları yanlış belirlersek algoritma bu hali ile en iyi çözümü bulamayabilir. Örneğin yukarıdaki şebeke için 1-2-3-4 güzergâhı seçelim. Bu güzergâhtan en fazla 10 birimlik bir akış gönderilebilir. Bu durumda (1, 2) (2, 3) ve (3, 4) ayrıtlarının kapasitelerinin tamamı kullanılmış olacağından şebekede akış miktarı pozitif olan başka bir güzergâh bulunamaz ve maksimum akış miktarı 10 birim olarak bulunur. Yanlış güzergâh seçimini engellemek için algoritma yukarıdaki şebeke yerine **artık şebeke (resudual graph)** olarak isimlendirilen bir şebeke üzerinde uygulanır. Artık şebekede ayrıtlardaki akışlar öne doğru ve geriye doğru akış olarak ikiye ayrılır. Öne doğru yapılan akışların bir kısmı veya tamamı geriye doğru akış yapılarak iptal edilebilir. Artık şebekede öne doğru ve geriye doğru akışlar için iki ayrı kapasite belirlenir. Tek yönlü ayrıtlar için geriye doğru akışlar kapasiteleri sıfır olarak belirlenir. Çift yönlü ayrıtlar için her yön için kapasiteler aynı olarak belirlenir. Örneğin yukarıdaki şebekenin artık şebekesi aşağıdaki gibi oluşturabiliriz.

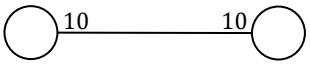


Artık şebekede bir ayrıttan herhangi bir akış yapıldığında artık kapasiteler güncellenir. Bunun için akış yönündeki kapasite akış miktarı kadar azaltılırken, ters yöndeki kapasite akış miktarı kadar arttırılır. Örneğin 1 düğümünden 2 düğümüne 3 birim akış yapıldığında

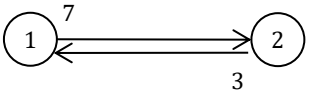
akış yönündeki kapasite 3 birim azaltılırken ters yöndeki akış kapasitesi 3 birim arttırılarak aşağıdaki gibi gösterilir. Böylece 1'den 2'e yapılan akışın bir kısmı veya tamamı 2'den 1'e gönderilerek iptal edilebilir.



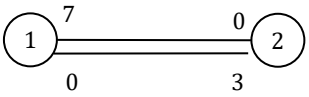
En büyük akış F_F algoritması ile çözülürken şebekenin önce artık şebeke haline dönüştürülmesi gerekir. Akış yönleri dikkate alınarak şebekedeki kapasiteler tespit edilmelidir. Yönsüz şebekeler için artık şebeke oluştururken her iki yön için ayrıt kapasitesi eşit olarak alınır. Örneğin kapasitesi 10 birim olan yönsüz bir ayrıt artık şebekede aşağıdaki gibi gösterilir.



Bazı şebekelerde iki düğüm arasındaki gidiş ve dönüş kapasiteleri aynı olmayabilir. Örneğin aşağıdaki şebekede 1 düğümünden 2 düğümüne 7 birim kapasite varken 2 düğümünden 7 düğümüne 3 birim kapasite bulunmaktadır.



Bu düğümler için artık şebeke aşağıdaki gibi oluşturulur.



Bu açıklamalar doğrultusunda Ford-Fulkerson algoritmasının adımları aşağıdaki gibidir.

Adım 1: Artık şebekeyi oluştur ($b_{ij}^0 = b_{ij}, b_{ji}^0 = b_{ji}$) ve $k = 1$ yap.

Adım 2: Başlangıç düğümünden bitiş düğümüne akış miktarı pozitif olan bir P_k güzergâhı belirle. Eğer akış miktarı pozitif olan bir güzergâh bulunamıyor ise en iyi çözüm bulunmuştur, 6. Adıma git. Değilse 3. Adıma git.

Adım 3: P_k güzergâhından yapılabilecek maksimum akış miktarı v_k'yı belirle

$$v_k = \min_{(i,j)\in P_k} \{b_{ij}^{k-1}\}$$

Adım 4: P_k güzergâhı üzerinde akış yönündeki ayrıtların kapasitelerini aşağıdaki gibi güncelle

$$b_{ij}^k = b_{ij}^{k-1} - v_k$$

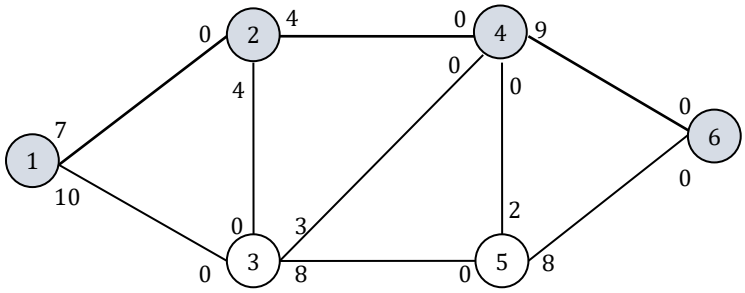
$$b_{ji}^k = b_{ji}^{k-1} + v_k$$

Adım 5: $k = k + 1$ yap ve 2. Adıma git.

Adım 6: Her güzergah için bulunan akış miktarlarını toplayarak toplam akışı belirle

$$v = \sum_{i=1}^{k} v_i$$

Örnek 8.5'i Ford-Fulkerson algoritması ile çözelim. Önce şebekeyi aşağıdaki gibi artık şebeke olarak düzenleyelim. İlk olarak P_1={1, 2, 4, 6} güzergâhını ele alalım.

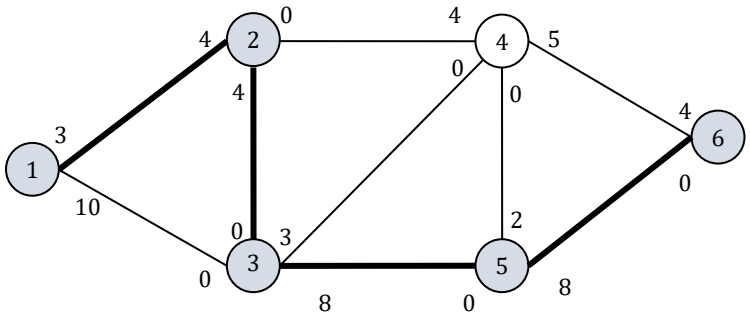


Bu güzergahtan gönderilebilecek en büyük akış miktarı $v_1 = Enk\{7, 4, 9\} = 4$ birimdir. Bu güzergâhtan yapılan akış miktarına göre artık kapasiteleri güncelleyelim.

$$b_{12} = 7 - 4 = 3, b_{24} = 4 - 4 = 0, b_{46} = 9 - 4 = 5$$

$$b_{21} = 0 + 4 = 4, b_{42} = 0 + 4 = 4, b_{64} = 0 + 4 = 4$$

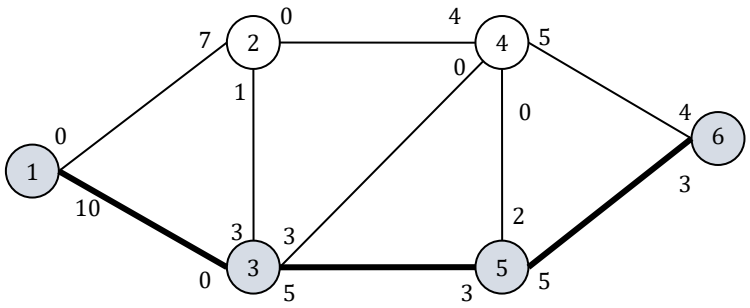
Bu güncelleme sonucunda aşağıdaki artık şebeke elde edilir.



İkinci güzergâh olarak akış miktarı pozitif olan $P_2 = \{1, 2, 3, 5, 6\}$ güzergâhını dikkate alalım. Bu güzergâhtan gönderilebilecek maksimum akış miktarı $v_2 = Enk\{3, 1, 8, 8\} = 3$ birimdir. Bu akış sonucunda ayrıtların kapasiteleri aşağıdaki gibi güncellenir ve aşağıdaki artık şebeke elde edilir.

$$b_{12} = 3 - 3 = 0, b_{23} = 4 - 3 = 1, b_{35} = 8 - 3 = 5, b_{56} = 8 - 3 = 5$$

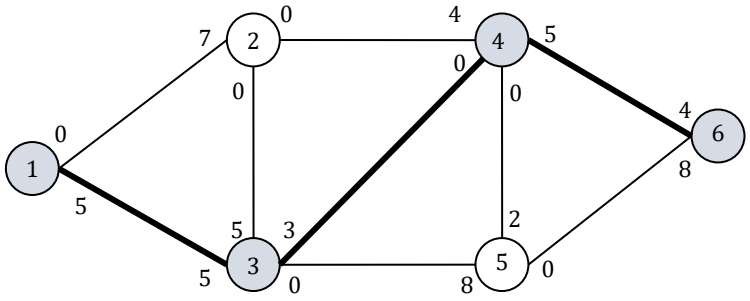
$$b_{21} = 4 + 3 = 7, b_{32} = 0 + 3 = 3, b_{53} = 0 + 3 = 3, b_{65} = 0 + 3 = 3$$



Üçüncü güzergâh olarak akış miktarı pozitif olan $P_3 = \{1, 3, 5, 6\}$ güzergâhını dikkate alalım. Bu güzergâhtan gönderilebilecek en büyük akış miktarı $v_3 = Enk\{10, 5, 5\} = 5$ birimdir. Bu akış sonucunda ayrıtların kapasiteleri aşağıdaki gibi güncellenir ve aşağıdaki artık şebeke elde edilir.

$$b_{13} = 10 - 5 = 5,\ b_{35} = 5 - 5 = 0, b_{56} = 5 - 5 = 0$$

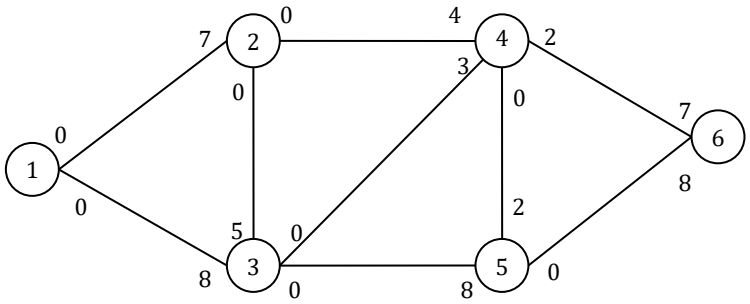
$$b_{31} = 0 + 5 = 5, b_{53} = 3 + 5 = 8, b_{65} = 3 + 5 = 8$$



Dördüncü güzergâh olarak akış miktarı pozitif olan $P_4 = \{1,3,4,6\}$ güzergâhını ele alalım. Bu güzergâhtan gönderilebilecek en büyük akış miktarı $v_3 = Enk\{5, 3, 5\} = 3$ birimdir. Bu akış sonucunda ayrıtların artık kapasiteleri aşağıdaki gibi güncellenir ve aşağıdaki artık şebeke elde edilir.

$$b_{13} = 5 - 3 = 2, b_{34} = 3 - 3 = 0, b_{46} = 5 - 3 = 2$$

$$b_{31} = 5 + 3 = 8, b_{43} = 0 + 3 = 3, b_{64} = 4 + 3 = 7$$



Akış miktarı pozitif olan bir güzergâh olmadığından algoritma sonlandırılır. Şebekedeki talep düğümüne gönderilebilecek maksimum akış miktarı $v = v_1 + v_2 + v_3 + v_4 = 4 + 3 + 5 + 3 = 15$ birim olarak bulunur.

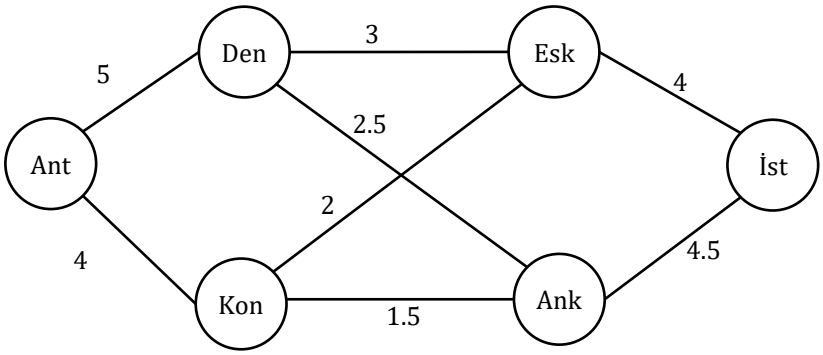
Ayrıtlardaki akış miktarlarının belirlenmesi

Ayrıtlardaki akış miktarlarını belirlemek için artık şebekedeki başlangıçtaki artık kapasitelerden algoritma sonundaki artık kapasiteleri çıkararak tespit edebiliriz. Örneğin (1,2) yönünde ayrıtın başlangıçtaki kapasitesi 7 birim iken algoritma sonundaki kapasitesi 2 birim olmuştur. Bu nedenle 1. düğümden 2. düğüme 5 birimlik taşıma yapılmıştır. Aradaki fark pozitif ise akış yönü 1'den 2'ye doğrudur. Tablo 8.3'de örnek problem için akış miktarları ve yönleri verilmektedir.

Tablo 8.3 Akış miktarları

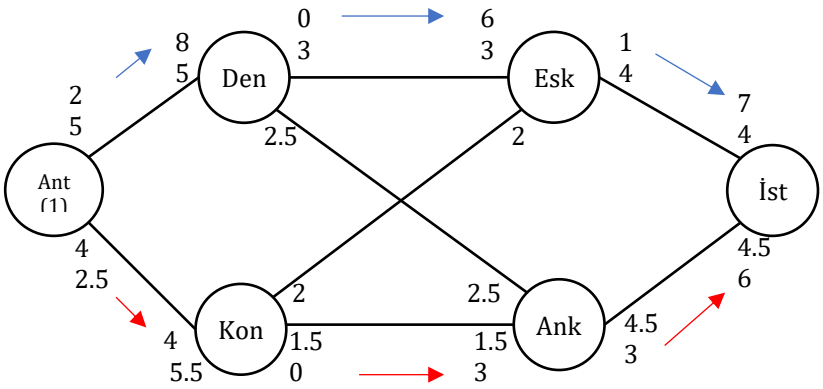
Ayrıtlar	Başlangıç Kapasitesi - Son kapasitesi= Akış miktarı
(1, 2)	7 -2 = 5
(1, 3)	10 - 0 = 10
(2, 3)	1- 0 = 1
(2, 4)	4 - 0 = 4
(3, 4)	3 - 0 = 3
(3, 5)	8 - 0 = 8
(5,4)	2 - 2 = 0
(4, 6)	9 - 2 = 7
(6,5)	0 - 8 = - 8

Örnek 8.6 [1]Antalya ile İstanbul arasındaki telefon görüşmeleri Denizli, Konya, Eskişehir, Ankara santralleri üzerinden aşağıdaki şebekedeki gibi yapılmaktadır. Santrallerin kapasiteleri (1000 hat) olarak şebeke üzerinde gösterilmektedir. Aynı anda Antalya'dan İstanbul'a yapılabilecek maksimum görüşme sayısını Ford – Fulkerson algoritmasını kullanarak bulunuz.



Çözüm: Problemi artık şebeke olarak aşağıdaki gibi gösterebiliriz.

[1]Winston (2003)'den uyarlanmıştır



P1: Ant-Den-Esk-İst güzergâhından en fazla v_1=3 bin görüşme yapılabilir. Bu durumda şebekedeki artık kapasiteler b_{12} = 2, b_{24} = 0, b_{46} = 1; b_{21} = 8, b_{42} = 6, b_{64} = 7 olur.

P2: Ant-Kon-Ank-İzm güzergâhından en fazla v_2=1.5 bin görüşme yapılabilir. Bu durumda şebekedeki artık kapasiteler b_{13} = 2.5, b_{35} = 0, b_{56} = 2; b_{31} = 5.5, b_{53} = 3, b_{65} = 6 olur.

P3:1-2-5-6 güzergâhından en fazla v_3=2 bin görüşme yapılabilir. Bu durumda şebekedeki artık kapasiteler b_{12} = 0, b_{25} = 0.5, b_{56} = 1; b_{21} = 10, b_{52} = 4.5, b_{65} = 8 olur.

P4:1-3-4-6 güzergâhından en fazla v_4=1 bin görüşme yapılabilir. Bu durumda şebekedeki artık kapasiteler b_{13} = 1.5, b_{34} = 1, b_{46} = 0; b_{31} = 6.5, b_{43} = 3, b_{64} = 8 olur.

P5: 1-3-4-2-5-6 güzergâhından en fazla v_5=0.5 bin görüşme yapılabilir. Bu durumda şebekedeki artık kapasiteler b_{13} = 1, b_{34} = 0.5, b_{42} = 5.5; b_{25} = 0, b_{56} = 0, b_{31} = 7, b_{43} = 4.5, b_{24}=0.5, b_{52}=5, b_{65}=8.5 olur.

Antalya İstanbul arasında aynı anda Toplam görüşme sayısı $v_1 + v_2 + v_3 + v_4 + v_5$ = 3 + 1.5 + 2 +1+0.5=8 bin olarak bulunur.