

Graf Veri Modeli

Graf, bir olay veya ifadenin düğüm ve çizgiler kullanılarak gösterilme şeklidir. Fizik, Kimya gibi temel bilimlerde ve mühendislik uygulamalarında ve tıp biliminde pek çok problemin çözümü ve modellenmesi graflara dayandırılarak yapılmaktadır. Bir graf aşağıdaki gibi tanımlanır.

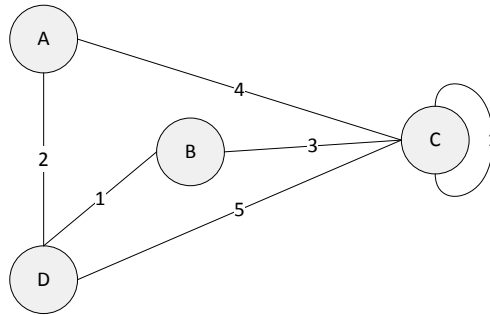
$D = \{d_0, d_1, d_2, \dots, d_{n-1}, d_n\}$ Düğümler kümesi

$K = \{k_0, k_1, k_2, \dots, k_{n-1}, k_n\}$ Kenarlar kümesi

$G(D, K)$ Graf

Basit bir $G(D, K)$ grafi, boş olmayan düğümler kümesine ve bu düğümler kümesindeki elemanları sıralı olma özelliğine bakılmaksızın bağlayan veya ilişkilendiren K kenar kümesine sahiptir.

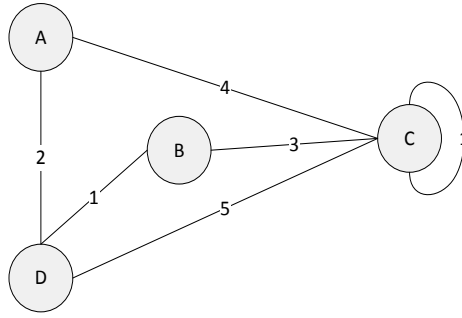
Bir G grafi üzerinde d_j ve d_i adlı iki düğüm, kenarlar kümesinde bulunan bir kenarla ilişkilendiriliyorsa bu iki düğüm birbirine komşu düğümlerdir denir.



Şekil 1 Maliyetli graf

Bir graf üzerindeki kenarların ağırlıkları/değerleri eşit değilse ve her biri farklı bir değer alabiliyorsa, bu graf maliyetli graf¹ olarak adlandırılır ve gösterimde maliyet bilgisi de yer alır. Şekil 1’de maliyetli graf örneğine yer verilmiştir.

¹ Weighted graph



Şekil 2 Yönsüz graf

Bir G grafi üzerindeki kenarlar bağlantının nerden başlayıp nerede sonlandığını belirten yön bilgisine sahip ise yönlü-graf veya yönlendirilmiş graf olarak adlandırılır. Yönlü-graf üzerindeki bir kenar, iki düğüm arasında bir bağlantı olduğunu, ancak ilişkinin oka bağlı olarak tek yönlü olduğunu gösterir; iki düğüm arasında karşılıklı ilişki varsa, ters yönde iki kenarlı ok kullanılır. Şekil 2’de yönsüz graf örneğine yer verilmiştir.

1. Graflara İlişkin Bazı Kavramlar

Bu başlık altında graflara ilişkin bazı kavramlar kısaca açıklanacaktır.

- Yol²: belirtilen iki düğüm arasında doğrudan ya da dolaylı olarak bir bağlantı olmasıdır.
- Yol ağacı³: Bir graf üzerinde, tüm düğümlere herhangi bir çevrim oluşturmadan gidilen yoldur.
- Çevrim⁴: Başlangıç ve bitiş düğümü aynı olan kapalı yola verilen isimdir.
- Dolaşı⁵: Birbirine bitişik olan düğüm-ayrıt-düğüm yapısıdır.
- Gezi⁶: Kenarların tekrarlanmadığı dolaşıdır.
- Gezgin satışı problemi⁷: Maliyetli bir graf üzerinde tüm düğümleri kapsayan ve en az maliyetle yapılan kapalı bir dolaşı bulunması problemidir. Dolaşı

² Path

³ Spanning tree

⁴ Cycle

⁵ Walk

⁶ Trail

⁷ Traveling salesman

sırasında tüm düğümlere uğranır ancak dolaşı sonrası başlangıç noktasına geri dönlüldüğünde alternatiflere göre daha az yol gidilecektir.

- Hamilton grafi⁸: Her bir düğümden yalnızca bir kez geçilmesi koşuluyla kapalı bir dolaşı gerçekleştirilebilen graftır.
- Euler grafi⁹: Üzerinde kenarların tekrarlanmadığı bir dolaşı, yani kapalı bir gezi yapılabilen graftır.
- Graf renklendirme¹⁰: Yalın bir grapta, komşu düğümlere farklı renk atama işlemidir. “Welch ve Powel” algoritması bu amaç için kullanılabilir.

2. Komşuluk Matrisi

Komşuluk matrisi¹¹, düğümlerden düğümlere olan bağlantıyı gösteren bir kare matristir; komşuluk matrisinin elemanları k_i değerlerinden oluşur. Komşuluk matrisi G_{dd} ’nin matrisel şekilde gösterilmesinden oluşur. Eğer komşuluk matrisi $G_{dd} = [a_{ij}]$ ise yönlü-maliyetsiz graflar için:

$$a_{ij} = \begin{cases} 1, & \text{Eğer } (d_i, d_j) \in K \text{ ise} \\ 0, & \text{Diğer durumlarda} \end{cases}$$

basit (yönsüz-maliyetsiz) graflar için ise,

$$a_{ij} = \begin{cases} 1, & \text{Eğer } (d_i, d_j) \in K \text{ ise veya } (d_j, d_i) \in K \text{ ise} \\ 0, & \text{Diğer durumlarda} \end{cases}$$

olur.

Şekil 3’de bir graf ve

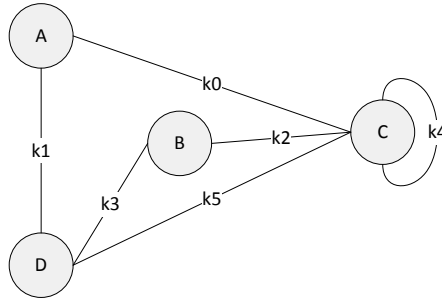
Şekil 4’de grafın tanımlanmış komşuluk matrisi yer almaktadır.

⁸ Hamilton’s graph

⁹ Euler’s graph

¹⁰ Graph coloring

¹¹ Adjacency Matrice



Şekil 3 Graf

A	A	B	C	D
A	0	0	1	1
B	0	0	1	1
C	1	1	1	1
D	1	1	1	0

Şekil 4 Komşuluk matrisi

3. Bitişiklik Matrisi

Bitişiklik matrisi¹², düğümlerle kenar arasındaki bağlantı/bitişiklik ilişkisini gösteren bir matristir; matrisin satır sayısı düğüm, sütun sayısı kenar sayısı kadar olur. Bitişiklik matrisi G_{dk} 'nin matris formunda gösterilmesinden oluşur. Eğer bitişiklik matrisi $G_{dk} = [m_{ij}]$ ise, maliyetsiz graf için:

$$m_{ij} = \begin{cases} 1, & \text{Eğer } (d_i, k_j) \text{ bağlantısı varsa} \\ 0, & \text{Diğer durumlarda} \end{cases}$$

A	k_0	k_1	k_2	k_3	k_4	k_5
A	1	1	0	0	0	0
B	0	0	1	1	0	0
C	1	0	1	0	1	1
D	0	1	0	1	0	1

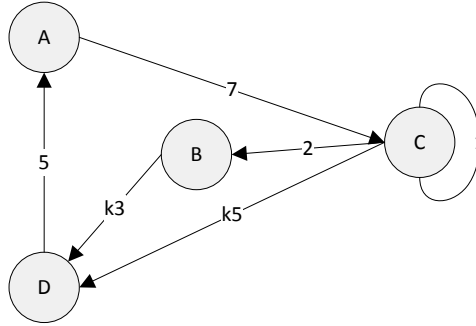
Şekil 5 Bitişiklik matrisi

¹² Incidence matrice

Yönlü olmayan graflarda komşuluk matrisi (A) simetrik özellikte olur; çünkü graf yönlü olmadığı için A düğümünden B düğüme olan bağlantı, aynı zamanda B düğümünden A düğüme olan bağlantıyla eşdeğerdir.

4. Düğüm Derecesi

Düğüm derecesi¹³, düğüme yapılmış bağlantı sayısıdır. Çevrimli kenarlar aynı düğüme hem çıkış hem de giriş yaptığı için dereceyi iki artırır. Yönlü graflarda düğüm derecesi giriş derecesi¹⁴ ve çıkış derecesi¹⁵ olarak ayrı ayrı belirtilir.



Şekil 6 Yönlü ve maliyetli graf

A	A	B	C	D
A	-	-	7	-
B	-	-	-	3
C	-	2	1	4
D	5	-	-	-

Şekil 7 Komşuluk matrisi

5. Tamamlanmış Graf

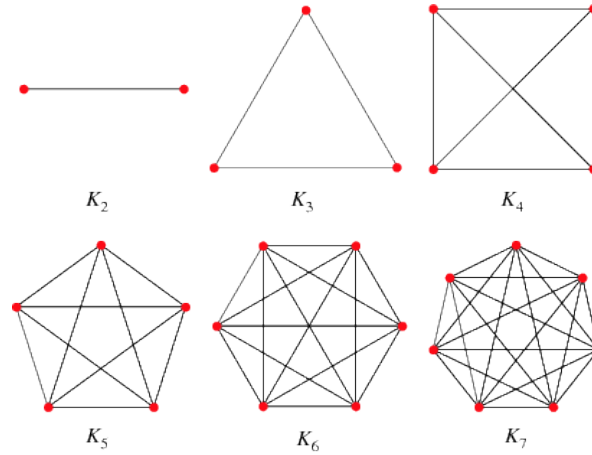
Her bir düğümü arasında doğrudan bağlantısı olan graflara tamamlanmış graf¹⁶ denir. Tamamlanmış bir graf üzerindeki tüm düğümlerin maliyetleri eşittir ve

¹³ Node degree

¹⁴ Input degree

¹⁵ Output degree

düğüm sayısından bir eksiktir. Tamamlanmış graf düğüm sayısı n , indis kullanılarak K_n şeklinde gösterilir. Şekil 8’de tamamlanmış graf örneklerine yer verilmiştir.



Şekil 8 Tamamlanmış graf örnekleri

Tamamlanmış bir K_n grafında düğümlerin derecesi $(n - 1)$, kenar sayısı ise $\frac{n(n-1)}{2}$ dir. Tamamlanmış graf yönlü ise bu bağıntı geçerli değildir. Yönlü-tamamlanmış graf üzerinde her düğümden her düğüme, kendisi de dâhil, bir yol vardır.

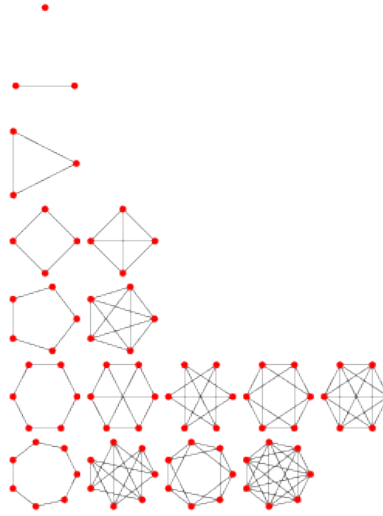
6. Düzenli Graf

Tüm düğümlerin derecesi aynı olan grafa düzenli graf¹⁶ denir. Tamamlanmış graf aynı zamanda düzenli bir graf olup tersi her zaman geçerli değildir. Yani düzenli bir graf, her zaman tamamlanmış bir graf değildir.

Şekil 9’da düzenli graf örneklerine yer verilmiştir. Dikkat edilirse tüm düğümlerin derecesi birbirine eşittir.

¹⁶ Complete graph

¹⁷ Regular graph

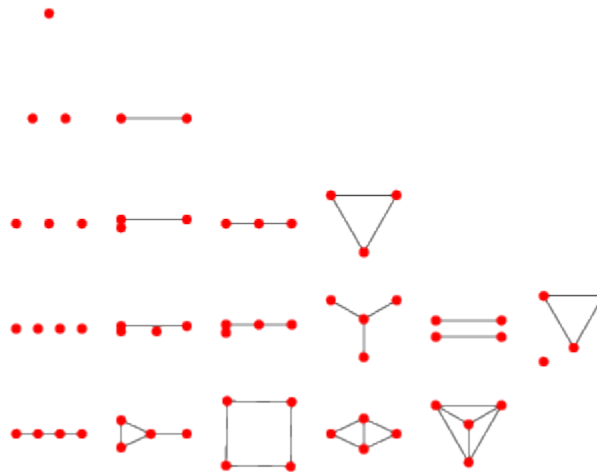


Şekil 9 Düzenli graf

7. Düzlemsel Graf

Hiçbir kenarı birbirini kesmeyen graf türüne düzlemsel graf¹⁸ denir. Düzlemsel bir graf, ayrıtları kesişmeden bir düzleme çizilebilir; harita graf veya kısaca harita olarak da adlandırılır.

Şekil 10'da düzlemsel graf örneklerine yer verilmiştir.



Şekil 10 Düzlemsel graf örnekleri

¹⁸ Planar graph

8. Grafların Bellekte Tutulması

Grafların bellekte tutulması ilgili birçok yöntem vardır. Yaygın olarak kullanılanlar aşağıda listelenmiştir:

- Matris üzerinde
- İki-Dizi üzerinde
- Bağlantılı liste ile
- Dizili-Bağlantılı liste

Grafın seyrek olması, genel olarak bağıl bir kavramdır. Düğüm sayısı N , kenar sayısı M olmak üzere $M \leq 2N$ ise seyrek, $M > 2N$ ise normal, $M > N^2/2$ ise yoğun olduğu söylenebilir. Grafın bellekte tutulma şekli uygulama yapısı ve gereksinimlerine göre farklılık gösterebilir. Ancak genel olarak seyrek bağlantıya sahip grafların matris üzerinde tutulması çok fazla bellek alanı harcar. Matris üzerinde tutulma işlemi yoğun graflar için daha uygun bir seçim gibi görülmektedir. Grafların matris üzerinde tutulmasının en büyük avantajı istenilen bağlantıya doğrudan erişim sağlayabilmesidir. Grafların, iki dizi üzerinde tutulması, matris üzerinde tutulmasına göre daha tasarruflu bir yöntemdir. Grafların tutulması için bağlantılı liste yapısı kullanılırsa, genel olarak çalışma zamanı uzar; yani hız azalır denebilir.

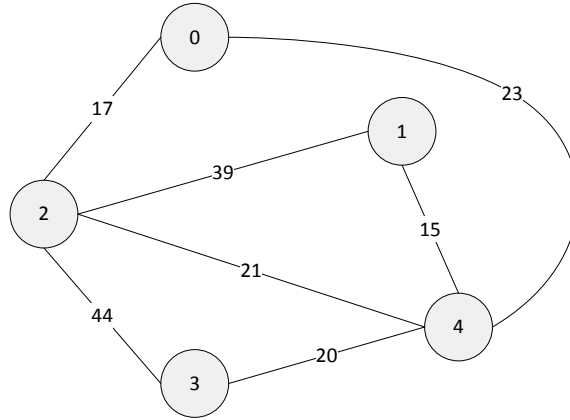
N : düğüm sayısı, M : kenar sayısı, d : düğüm derecesi, c : çevrimli kenar sayısı, b : sizeof(veri türü), a : sizeof(bellek adresi) olmak üzere Şekil 11'de grafın bellekte tutulma biçimlerinin karşılaştırılmasına yer verilmiştir.

Yöntem	Bellek Gereksinimi	Bağlantı Sorgulama	Yeni Bağlantı Ekleme
Matris Üzerinde	N^2b	$O(1)$	$O(1)$
İki-Dizi Üzerinde	$((2M - c) + (N + 1))b$	$O(d)$	$O(M)$
Bağlantılı Liste ile	$N(b + (da))$	$O(N^2)$	$O(N^2)$
Dizili-Bağlantılı Liste ile	$(N + 2M)(b + a)$	$O(d)$	$O(d)$

Şekil 11 Grafın bellekte tutulma biçimlerinin karşılaştırılması

9. Örnek Uygulama

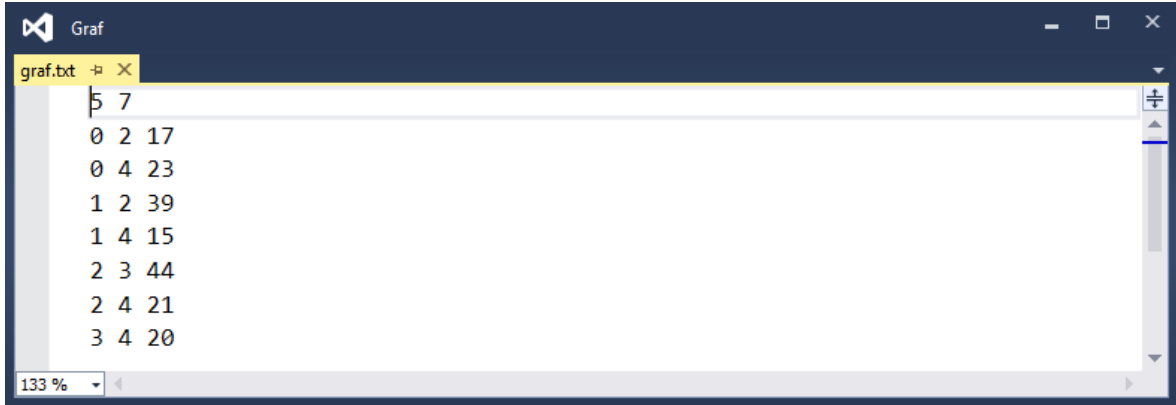
Şekil 12’de örnek bir graf ve maliyet bilgileri yer almaktadır. İlgili uygulamada graf düğüm ve kenar bilgilerinin bir metin dosyasında tutulması ve metin dosyasından alınan bilgilere göre grafın bir matris üzerinde tanımlanması sağlanmıştır. Şekil 13’de grafın komşuluk matrisi görülmektedir. Yapılacak uygulamada komşuluk matrisinin otomatik olarak oluşturulması sağlanacaktır.



Şekil 12 Örnek graf

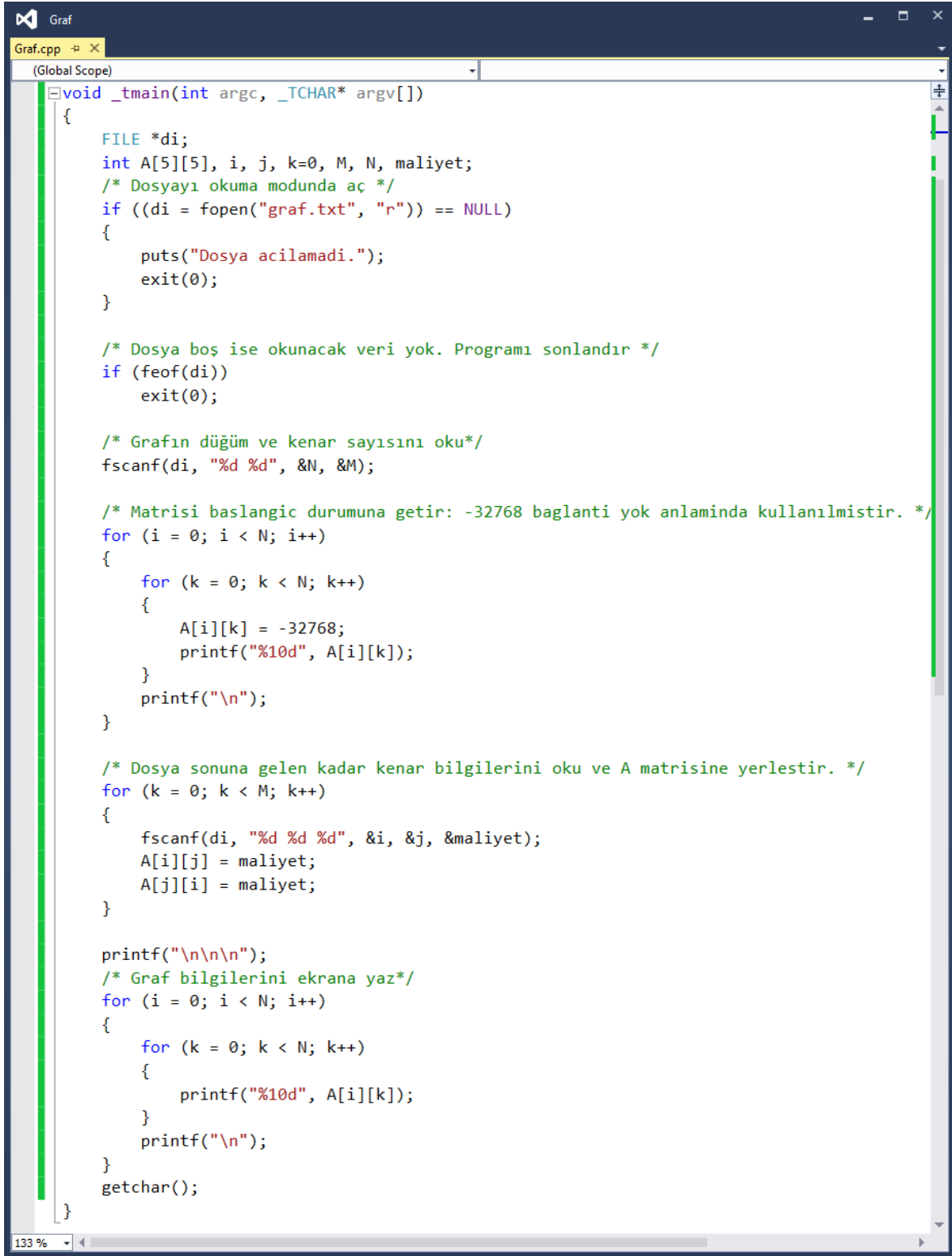
A	0	1	2	3	4
0	-	-	17	-	23
1	-	-	39	-	15
2	17	39	-	44	21
3	-	-	44	-	20
4	23	15	21	20	-

Şekil 13 Komşuluk matrisi



Şekil 14 Graf bilgilerinin tutulduğu metin dosyası

Şekil 14’de graf bilgilerinin tutulduğu metin dosyası görülmektedir. 5 düğüm sayısını 7 ise kenar sayısını temsil etmektedir. 0 2 17 şeklilde sıralanan dizide 0’dan 2. düğüme maliyet bilgisinin 17 olduğu anlaşılmaktadır.



```
void _tmain(int argc, _TCHAR* argv[])
{
    FILE *di;
    int A[5][5], i, j, k=0, M, N, maliyet;
    /* Dosyayı okuma modunda aç */
    if ((di = fopen("graf.txt", "r")) == NULL)
    {
        puts("Dosya acilamadi.");
        exit(0);
    }

    /* Dosya boş ise okunacak veri yok. Programı sonlandır */
    if (feof(di))
        exit(0);

    /* Grafın düğüm ve kenar sayısını oku*/
    fscanf(di, "%d %d", &N, &M);

    /* Matrisi baslangic durumuna getir: -32768 baglanti yok anlaminda kullanilmistir. */
    for (i = 0; i < N; i++)
    {
        for (k = 0; k < N; k++)
        {
            A[i][k] = -32768;
            printf("%10d", A[i][k]);
        }
        printf("\n");
    }

    /* Dosya sonuna gelen kadar kenar bilgilerini oku ve A matrisine yerlestir. */
    for (k = 0; k < M; k++)
    {
        fscanf(di, "%d %d %d", &i, &j, &maliyet);
        A[i][j] = maliyet;
        A[j][i] = maliyet;
    }

    printf("\n\n\n");
    /* Graf bilgilerini ekrana yaz*/
    for (i = 0; i < N; i++)
    {
        for (k = 0; k < N; k++)
        {
            printf("%10d", A[i][k]);
        }
        printf("\n");
    }
    getchar();
}
```

Şekil 15 Programın kaynak kodları

