PROGRAMCILAR İŞ BAŞINA

©Ali Galip Bayrak

ODTÜ Bilgisayar Topluluğu'nun Bilim ve Teknik için hazırladığı bu sayfa ile bilgisayar bilimlerinin temel problemlerini tanıtmayı amaçlıyoruz. Bu problemler için herhangi bir dilde yazacağınız çözüm kodunu bteknik@tubitak.gov.tr adresine yollayabilirsiniz. Her ay sonunda o ayın çözümlerine ve yapılan değerlendirme sonucu topladığınız puanlara web sitemizden (www.biltek tubitak.gov.tr) ulaşabilirsiniz. Sene sonunda en fazla puan toplayan yarışmacıya özel bir ödül vereceğiz. İlgilenenler için ODTÜ Bilgisayar Topluluğu'nun web sitesi: http://www.cclub.metu.edu.tr/biltek

Yollar

Kral Bora, gelen davet üzerine, ülkesinin güneyinde bulunan Kah şehrine gitmek için yola çıkar. Fakat yolculuk beklediğinden çok uzun sürer, çünkü ülkesindeki yollar ulaşım açısından çok elverişsizdir. Bunun üzerine Kral, baş vezirini çağırır ve der ki: "Ülkedeki bazı yolları yenilememiz gerekiyor. Yenilenecek yolları o şekilde seçmeliyiz ki, bu yolları kullanarak bütün şehirlerden diğer bütün şehirlere gidilebilsin ve olabilecek en az uzunlukta yol yapalım. Bana hemen hangi yolları yenilememiz gerektiğini bul." Bunun üzerine başvezir Murat hemen şehirlerarası yol haritası bulur ve düşünmeye başlar. Sizden istenen başvezir Murat'a yardımcı olmanız.

Varsayımlar

- Ülkede *n* adet şehir vardır ($2 \leq n \leq 100$).
- Şehirler 1'den *n*'e kadar numaralandırılmıştır.
- Ülkede şu anki mevcut yol sayısı *m*'dir.
- Bütün yollar çift yönlüdür.
- Herhangi iki şehri doğrudan bağlayan en fazla bir yol olabilir.
- Birden fazla çözüm olması durumunda herhangi birisinin bulunması yeterlidir.

Girdi

- Girdiler "yollar.gir" isimli dosyadan okunacaktır.
- İlk satırda şehir sayısını ifade eden *n* verilecektir.
- Takip eden satırda mevcut yol sayısını ifade eden m verilecektir.
- Takip eden m adet satırda sırayla bütün yolların ayrıntıları verilecektir. Her satırda üç adet tamsayı bulunacaktır. Bu tamsayılardan birincisi ve ikincisi yolun hangi şehri hangi şehre bağladığını, üçüncüsü ise yolun uzunluğunu ifade edecektir.

Çıktı

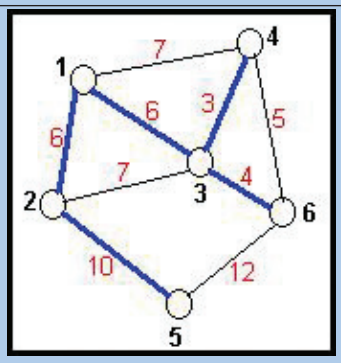
- Çıktılar "yollar.cik" isimli dosyaya yazılacaktır.
- İlk satırda kaç adet yolun yenilenmesi gerektiğini ifade eden *k* basılmalıdır.
- Takip eden k adet satır yenilenmesi gereken yolları göstermelidir. Her satırda iki adet tamsayı bulunmalıdır. Bu tamsayı yenilenmesi gereken yolun hangi şehirler arasında olduğunu belirtmelidir.

Örnek

yollar.gir:
```
6
9
1 2 6
1 4 7
3 6 4
1 3 6
6 5 12
4 6 5
4 3 3
2 3 7
5 2 10
```
yollar.cik:
```
5
5 2
1 2
1 3
6 3
3 4
```
1.şehir için ulaşım:
2'ye: 1-2
3'e: 1-3
4'e: 1-3-4
5'e: 1-2-5
6'ya: 1-3-6

2.şehir için ulaşım:
1'e: 2-1
3'e: 2-1-3
4'e: 2-1-3-4
5'e: 2-5
6'ya: 2-1-3-6

.....



Kırmızı sayılar yolların uzunluklarını, mavi çizgiler yenilenmesi gereken yolları göstermektedir.

Yollar 2

(Sorunun hikayesi "Yollar" sorunun hikayesinin devamı niteliğindedir.)

Başvezir Murat hangi yolların yenilenmesi gerektiğini söylemek için Kral Bora'nın odasına girer. Daha söyleyeceklerine başlamadan Kral hemen konuşmaya başlar: "Yenilenmesi gereken yolları bulurken dikkat etmen gereken bir kaç önemli nokta çıktı Murat. Vezir Özgür'ün söylediğine göre bazı yolları babam kral iken yeniletmiş zaten. Onları tekrar yenilemeye gerek yok. Sen en iyisi bunları da dikkate alarak baştan belirle yenilenmesi gereken yolları." Murat'ın tekrar yardıma ihtiyacı var...

Varsayımlar

- "Yollar" sorusunun varsayımları aynen geçerlidir.
- Daha önceden yenilenmiş yolların sayısı *s*'dir.

Girdi

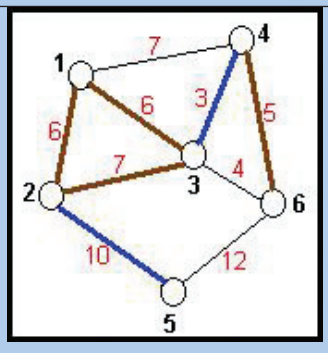
- Girdiler "yollar2.gir" isimli dosyadan okunacaktır.
- İlk satırda şehir sayısını ifade eden *n* verilecektir.
- Takip eden satırda mevcut yol sayısını ifade eden *m* verilecektir.
- Takip eden *m* adet satırda sırayla bütün yolların ayrıntıları verilecektir. Her satırda üç adet tamsayı bulunacaktır. Bu tamsayılardan birincisi ve ikincisi yolun hangi şehri hangi şehre bağladığını, üçüncüsü ise yolun uzunluğunu ifade edecektir.
- Takip eden satırda daha önceden yenilenmiş yolların sayısını ifade eden *s* verilecektir.
- Takip eden *s* satırda sırayla daha önceden yenilenmiş yollar verilecektir. Her satırda iki adet tamsayı bulunacaktır. Bu iki tamsayı yolun hangi şehri hange şehre bağladığını ifade edecektir.

Çıktı

- Çıktılar "yollar2.cik" isimli dosyaya yazılacaktır.
- İlk satırda kaç adet yolun yenilenmesi gerektiğini ifade eden *k* basılmalıdır.
- Takip eden *k* adet satır yenilenmesi gereken yolları göstermelidir. Her satırda iki adet tamsayı bulunmalıdır. Bu tamsayı yenilenmesi gereken yolun hangi şehirler arasında olduğunu belirtmelidir.

Örnek

yollar2.gir:
```
6
9
1 2 6
1 4 7
3 6 4
1 3 6
6 5 12
4 6 5
4 3 3
2 3 7
5 2 10
4
1 2
2 3
1 3
4 6
```
yollar2.cik:
```
2
3 4
5 2
```



Kırmızı sayılar yolların uzunluklarını, kahverengi çizgiler önceden yenilenmiş yolları, mavi çizgiler yenilenmesi gereken yolları göstermektedir.

Geçen Sayımızdaki Soruların Çözümleri

Şehirler

Bu problem, bilgisayar biliminde, "En Kısa Yol (shortest path)" problemi olarak bilinir. Problem için üretilen birden çok çözüm yolu vardır. Anlatacağım çözüm Dijkstra'nın en kısa yol algoritmasıdır.

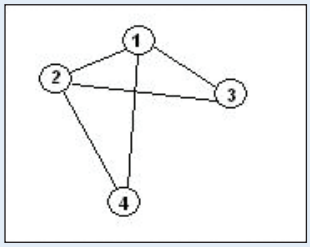
0. Başlangıç şehrinin uzaklığına 0, diğer şehirlerin uzaklıklarına ∞ (sonsuz) verilir. Başlangıç şehri kırmızıya boyanır.

1. Kırmızıya boyanmamış her şehir için:

Eğer en son kırmızıya boyanan şehirden söz konusu şehre doğrudan yol varsa ve "en son kırmızıya boyanmış şehrin uzaklığı + doğrudan yolun uzunluğu" değeri "söz konusu şehrin uzaklığı" değerinden küçükse, "söz konusu şehrin uzaklığı" değeri "en son kırmızıya boyanmış şehrin uzaklığı + doğrudan yolun uzunluğu" değerine eşitlenir ve "söz konusu şehre gelinen yer" olarak "en son kırmızıya boyanmış şehir" atanır.

2. Kırmızıya boyanmamış şehirler arasından uzaklığı en küçük olan seçilir ve kırmızıya boyanır.

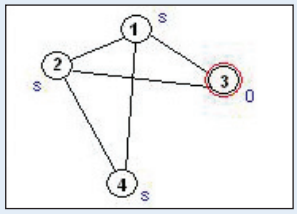
3. Eğer kırmızıya boyanmamış şehir kaldı ise 1. adıma tekrar dönülür. Tüm şehirler kırmızıya boyandı ise bitirilir. Her şehrin uzaklığı, o şehrin başlangıç şehrine olan en kısa uzaklığını verir. "söz konusu şehre gelinen yer" değerlerini takip ederek başlangıç şehrinden o şehre nasıl gelindiği bulunabilir.



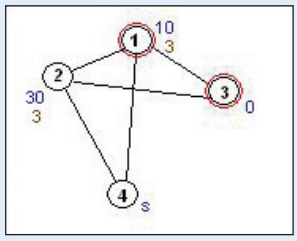
İlk durum:

Siyah çemberler şehirleri, siyah çizgiler şehirler arasındaki doğrudan yolları, turuncu sayılar yolların uzunluklarını belirtmektedir.

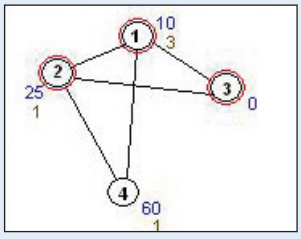
Başlangıç şehri 3 olarak verilmiş, kırmızıya boyanıyor. 3. şehrin uzaklığı 0, diğer şehirlerin



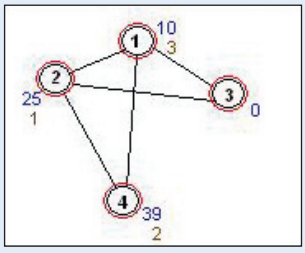
uzaklığı s (sonsuz) olarak atanıyor.



1. ve 2. şehrin uzaklıkları sırasıyla 10 ve 30 olarak atanıyor. 1. şehre ve 2. şehre gelinen yer olarak (kahverengi ile gösteriliyor) 3. şehir atanıyor. En küçük uzaklığa sahip olan 1. şehir kırmızıya boyanıyor.



Aynı işlemler yapılarak değerler güncelleniyor.



Ve bitiyor...

Son durumda 3. şehirden diğer şehirlere olabilecek en kısa yollar:

3-1 arası : uzaklık 10
yol 3-1
3-2 arası : uzaklık 25
yol 3-1-2
3-4 arası : uzaklık 39
yol 3-1-2-4

Altın Toplama

Sorunun çözümünü anlamadan önce bir algoritmaya değinmek gerekiyor. Bilgisayar biliminde BFS (breadth first search) yani genişlik öncelikli arama diye bilinen algoritmayı kullanarak, şekildeki gibi bir zindan verildiğinde, hangi kareye en az kaç adımda gidebileceğimizi bulabiliriz.

3	4	5	6	7	8	9	10
2		4		8	9	10	
1		3		9	10		
	1	2		10			

Şekilde mavi renkteki kareyi başlangıç karesi kabul edelim. Diğer karelere en az kaç hamlede gidebileceğimizi BFS uygularak bulabiliriz. Yapmamız gereken, ilk once başlangıç karesinden gidebilebileceğimiz karelere 1 yazmak, 1 yazan karelerden gidebileceğimiz ve daha önceden birşey yazmadığımız karelere 2 yazmak, ve bu şekilde *k* yazdığımız karelerden gidebileceğimiz ve daha önce gidilmemiş karelere *k+1* yazmak ve sonunda tüm kareleri tamamlamak. Şekilde 10'a kadar olan kareler yazılmış. Açıkça görülüyor ki, mavi kareden turuncu kareye gitmek için en az 15 hamle gerekli.

Sorumuza tekrar dönecek olursak:

İçerisinde altın olan kareleri ve bitiş karesini, bir önceki soruda belirtilen "en kısa yol" algoritmasındaki köşeler (şehirler) gibi düşünelim. Daha sonra, eğer *A* köşesinden *B* köşesine A'daki sayıdan daha az veya eşit adımda ulaşılabiliyorsa A'dan B'ye bir yol olduğunu ve bu yolun uzunluğunun A'dan B'ye kaç adımda gidilebileceği kadar olduğunu kabul edelim(bütün köşe ikilileri için bu işlemi yapalım).

Yani sorumuzda verilen zindan için aşağıdaki gibi bir harita çıkarabiliriz:

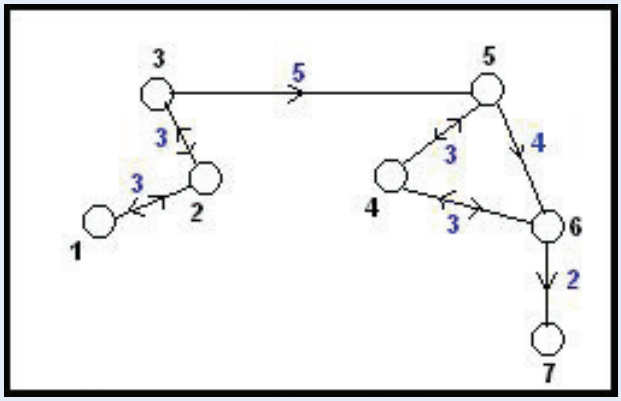
	3					5	
		2			4		
1							6
							7

Karelerin içinde yazan sayılar o karelerin numarası olsun.

Bu haritayı çıkardıktan sonra harita üzerinde, bir önceki soruda bahsettiğimiz "en kısa yol" algoritmasının yönlü graflar üzerindeki uygulamasını kullanarak sonuca ulaşabiliriz.

Şekilde görüldüğü gibi 1'den 7'ye 1-2-3-5-6-7 yolunu kullanarak ulaşabiliriz. 1'den 2'ye, 2'den 3'e, 3'den 5'e... nasıl ulaştığımızı da BFS uygularken her kare için "hangi kareden geldiğimiz" bilgisini tutarak yapabiliriz.

Not: Sorunun farklı çözüm yöntemleri de vardır. Burada BFS ve "en kısa yol" algoritmalarının nasıl kullanılabileceği gösterilmiştir.



Şekilde oklar yolların yönünü göstermektedir. Bazı yollar tek yönlüdür (3-5 arası ve 6-7 arası). Mavi sayılar da yolların uzunluğunu göstermektedir.