İstasyonlar (stations)

Singapur'un İnternet Omurgası (SIB) her birine 0'dan $n-1$'e kadar bir **indeks** verilmiş n adet istasyondan oluşmaktadır. Herbiri iki istasyonu birbirine bağlayan ve 0'dan $n-2$'ye kadar numaralandırılmış $n-1$ tane çift yönlü link vardır. Aralarında bir link bulunan iki istasyona komşu istasyonlar denir.

x istasyonundan y istasyonuna bir yol, bütün elemanları birbirinden farklı ve şu şartları sağlayan bir $a_0, a_1, \cdots, a_p$ istasyon dizisi olarak tanımlanmıştır: $a_0 = x$'dir, $a_p = y$'dir ve dizinin her iki ardışık elemanı birbirine komşu istasyonlardır. Herhangi bir x istasyonundan herhangi bir y istasyonuna **sadece ve sadece bir** yol olduğu bilinmektedir.

Herhangi bir x istasyonu herhangi bir y istasyonuna gönderilmek üzere bir paket oluşturabilir. Bu durumda y istasyonuna paketin **hedefi** diyoruz. Bu paket İnternet omurgasında x'den y'ye giden yegane (unique) yoldan, aşağıdaki şekilde yönlendirilecektir. Hedefi y olan bir paket, $z \neq y$ olmak üzere z istasyonuna ulaşmış olsun. Bu durumda z istasyonu şunları yapacaktır:

1. z istasyonunun z'den y'ye giden yegane yol üzerindeki komşusunu bulan bir **yönlendirme prosedürü** çalıştıracak, ve
2. paketi bu komşusuna gönderecektir.

Ancak, istasyonların hafızası sınırlı olup, yönlendirme prosedüründe kullanılmak üzere SIB'deki bütün linklerin listesini hafızada tutamamaktadır.

Göreviniz SIB için iki fonksiyondan oluşan bir yönlendirme prosedürü implement etmektir.

- Birinci fonksiyon n sayısını, SIB'deki linklerin listesini ve $k \geq n-1$ olmak üzere bir k sayısını girdi olarak almaktadır. Bu fonksiyon herbir istasyonu 0 ve k arasında (0 ve k dahil) bir tamsayı ile etiketlemektedir. Farklı istasyonların etiketleri de farklı olmak zorundadır.

- İkinci fonksiyon yönlendirme prosedürü olup, istasyonlar etiketlendikten sonra bütün istasyonlara yüklenmiştir. Bu fonksiyon **sadece** aşağıdaki girdileri almaktadır.

  - s, şu anda paketin bulunduğu istasyonun **etiketi**,
  - t, paketin hedef istasyonunun **etiketi** ($t \neq s$),
  - c, s'nin bütün komşularının **etiketlerinin** listesi.

  Bu fonksiyon s'nin paketin yönlendirilmesi gereken komşusunun **etiketini** döndürecektir.

Bir altgörevde, çözümünüzün alacağı puan kullandığınız en büyük etiket numarasına bağlıdır (genel olarak, ne kadar küçükse o kadar iyidir).

Implementasyon detayları

Aşağıdaki fonksiyonları implement etmelisiniz:

```
int[] label(int n, int k, int[] u, int[] v)
```

- n: SIB'deki istasyon sayısı.
- k: Kullanılabilecek en büyük etiket numarası.
- u and v: Linkleri tanımlayan herbiri $n-1$ uzunluğunda iki dizi. $0 \leq i \leq n-2$ olmak üzere herbir i için, i linki indeksi $u[i]$ ve $v[i]$ olan istasyonları bağlamaktadır.
- Bu fonksiyon n uzunluğunda bir L dizisi döndürmelidir. Herbir i için $(0 \leq i \leq n-1)$ $L[i]$ indeksi i olan istasyonuna verilen etiketi göstermektedir. L dizisinin bütün elemanları birbirinden farklı ve 0 ile k arasındadır (0 ve k dahil).

```
int find_next_station(int s, int t, int[] c)
```

- s: Paketin bulunduğu istasyonun etiketi.
- t: Paketin hedefi olan istasyonun etiketi.
- c: s'nin bütün komşularının etiketlerini tutan bir dizi. c dizisi artan şekilde sıralıdır.
- Bu fonksiyon s'nin paketi yönlendirilmesi gereken komşusunun etiketini döndürmelidir.

Herbir test case bir yada daha fazla birbirinden bağımsız senaryodan oluşmaktadır (yani, farklı SIB tanımlamaları). r senaryodan oluşan bir test case için, yukarıdaki fonksiyonları çağıran bir **program** aşağıda açıklandığı üzere tam olarak iki kere çalıştırılmaktadır.

Programın birinci çalıştırılışında:

- `label` fonksiyonu r defa çağırılmaktadır,
- döndürülen etiketler grading system tarafından depolanmaktadır, ve
- `find_next_station` fonksiyonu çağırılmamaktadır.

Programın ikinci çalıştırılışında:

- `find_next_station` bir veya daha fazla defa çağırılmaktadır. Her çağrıda **rastgele** bir senaryo seçilmektedir ve bu senaryo için `label` fonksiyonu tarafından döndürülmüş etiketler `find_next_station` fonksiyonuna girdi olarak verilmektedir.
- `label` fonksiyonu çağırılmamaktadır.

Programın birinci çalıştırılışında statik yada global değişkenlere kaydedilen bilgiler `find_next_station` fonksiyonu tarafından erişilemeyecektir.

Örnek

Aşağıdaki çağrıya bakınız:

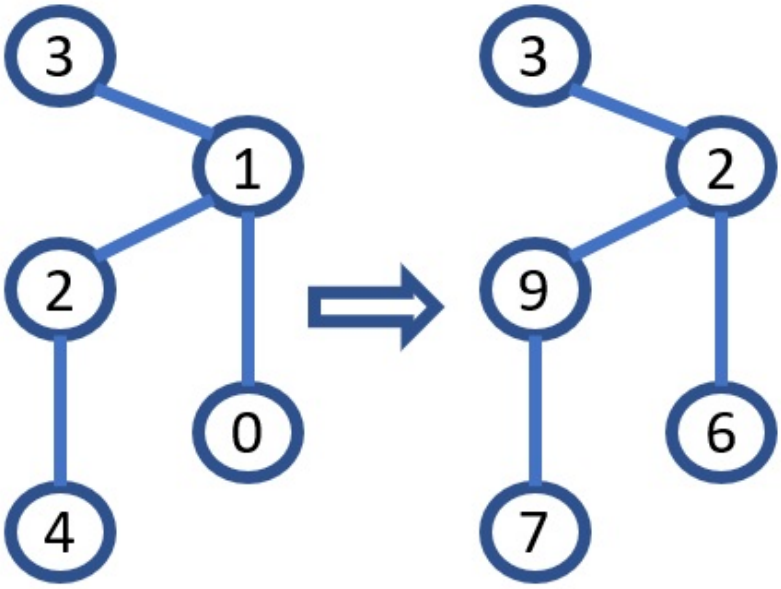
```
label(5, 10, [0, 1, 1, 2], [1, 2, 3, 4])
```

Bu örnekte 5 tane istasyon ve indeksleri $(0, 1)$, $(1, 2)$, $(1, 3)$ ve $(2, 4)$ olan istasyonları bağlayan 4 link vardır. Her etiket 0'dan $k = 10$'a kadar (0 ve 10 dahil) bir tamsayıdır.

Aşağıdaki etiketlemeyi raporlamak için:

Index	Label
0	6
1	2
2	9
3	3
4	7

`label` fonksiyonu $[6, 2, 9, 3, 7]$ dizisini döndürmelidir. Aşağıdaki resimin sol tarafında istasyonların indeksleri sağ tarafında ise etiketleri görülmektedir.



Etiketlerin yukarıdaki şekilde verildiği varsayımıyla aşağıdaki çağrıya bakınız:

```
find_next_station(9, 6, [2, 7])
```

Bu çağrı paketin etiketi 9 olan istasyonda olduğu ve hedef istasyonun ise etiketinin 6 olduğunu göstermektedir. Hedef istasyona olan yoldaki istasyonların etiketleri $[9, 2, 6]$'dır. Bu yüzden, bu çağrı paketin yönlendirilmesi gereken istasyonun etiketini olan 2'yi döndürmelidir (Bu istasyonun indeksi 1'dir).

Aşağıdaki başka bir muhtemel çağrıya bakınız:

```
find_next_station(2, 3, [3, 6, 9])
```

Bu çağrı 3 döndürmelidir çünkü etiketi 3 olan hedef istasyon etiketi 2 olan istasyonun bir komşusudur ve paketi direk olarak almalıdır.

Kısıtlar

- $1 \leq r \leq 10$

`label` fonksiyonunun herbir çağrısı için:

- $2 \leq n \leq 1000$
- $k \geq n - 1$
- $0 \leq u[i], v[i] \leq n - 1$ ($0 \leq i \leq n - 2$ şartını sağlayan her i için)

`find_next_station` fonksiyonunun herbir çağrısında girdi daha önceki `label` çağrılarından rastgele seçilen birinden gelmektedir. Seçilen `label` çağrısının döndürdüğü etiketlerle uyumlu olacak şekilde:

- s ve t iki farklı istasyonun etiketleridir.
- c ise s'nin komşusu olan bütün istasyonların etiketlerinin artan sıralı dizisidir.

Herbir test case için `find_next_station` fonksiyonunun çağrıldığı bütün senaryolardaki c dizilerinin uzunluklarının toplamı $100\ 000$'i geçmemektedir.

Altgörevler

1. (5 puan) $k = 1000$, hiçbir istasyonun 2'den fazla komşusu yoktur.
2. (8 puan) $k = 1000$, link i indeksleri $i + 1$ ve $\left\lfloor \frac{i}{2} \right\rfloor$ olan istasyonları bağlamaktadır.
3. (16 puan) $k = 1\ 000\ 000$, 2'den fazla komşusu olan en fazla bir istasyon vardır.
4. (10 puan) $n \leq 8$, $k = 10^9$
5. (61 puan) $k = 10^9$

$5.$ altgörevde partial skor alabilirsiniz. m `label` fonksiyonun bütün senaryolarda kullandığı en büyük etiket değeri olsun. Bu altgörev için skorunuz aşağıdaki tabloya göre hesaplanacaktır:

Maximum label	Score
$m \geq 10^9$	0
$2000 \leq m < 10^9$	$50 \cdot \log_{5 \cdot 10^5}(\frac{10^9}{m})$
$1000 < m < 2000$	50
$m \leq 1000$	61

Örnek grader

Örnek grader girdiyi aşağıdaki formatta okumaktadır:

- satır 1: r

Altında herbiri bir senaryoyu tarif eden r blok vardır. Herbir bloğun formatı aşağıdaki gibidir:

- satır 1: n k
- satır $2 + i$ $(0 \leq i \leq n - 2)$: $u[i]$ $v[i]$
- satır $1 + n$: q: `find_next_station` fonksiyonuna yapılan çağrı sayısı.
- satır $2 + n + j$ $(0 \leq j \leq q - 1)$: $z[j]$ $y[j]$ $w[j]$: `find_next_station` fonksiyonunun j. çağrısında yer alan istasyonların **indeksleridir**. $z[j]$ paketin bulunduğu istasyon, $y[j]$ paketin hedef istasyonu ve $w[j]$ ise paketin yönlendirileceği istasyondur.

Örnek grader sonucu aşağıdaki formatta yazmaktadır:

- satır 1: m

Daha sonra girdideki ardışık senaryolara karşılık gelen r blok gelmektedir. Herbir bloğun formatı aşağıdaki gibidir:

- satır $1 + j$ $(0 \leq j \leq q - 1)$: Bu senaryoda `find_next_station` fonksiyonunun j. çağrısında **etiketi** döndülen istasyonun **indeksi**.

Örnek grader'ın herbir çalıştırılışında hem `label` fonksiyonunun hem de `find_next_station` çağrıldığına dikkat ediniz.