©Ali Galip Bayrak

Güvenlik

ABC şirketinin binası çok güvenli bir şekilde korunmaktadır. Öyle ki, binanın merkezinde bir yere koyulan bir dedektör sayesinde binanın içindeki bütün insanlar algılanabilmektedir. Fakat bu dedektörün algıladığı alan dairesel bir alan olduğu için binanın dışında kalan bazı bölgeler de dedektörün kapsama alandadır ve binanın dışında insan olduğu zaman da uyarı verebilmektedir. Sizden istenen ABC şirketini bu durumdan kurtarmanız, yani gelen uyarının bina içinden olup olmadığını kontrol eden bir program yazmanız.

Varsayımlar

- ABC şirketinin binası bir çokgen şeklindedir ve bu çokgen *n* köşelidir.
- Size çokgen verilirken çokgenin köşeleri saat yönünde ve sırayla verilecektir.

Girdi

- Girdiler "guvenlik.gir" isimli dosyadan okunacaktır.
- İlk satırda çokgenin (binanın şekli) köşelerinin sayısını ifade eden *n* verilecektir.
- Takip eden *n* satırın her birisinde iki adet pozitif tamsayı verilecektir. Bu sayılar sıradaki köşenin x ve y koordinatlarını belirtecektir.
- Takip eden satırda iki adet pozitif tamsayı bulunacaktır. Bu sayılar uyarıya neden olan insanın koordinatlarını ifade edecektir.

Çıktı

- Çıktılar "guvenlik.cik" isimli dosyaya yazılacaktır.
- Uyarıya neden olan insan bina içinde ise "EVET", dışında ise "HAYIR" basılacaktır.

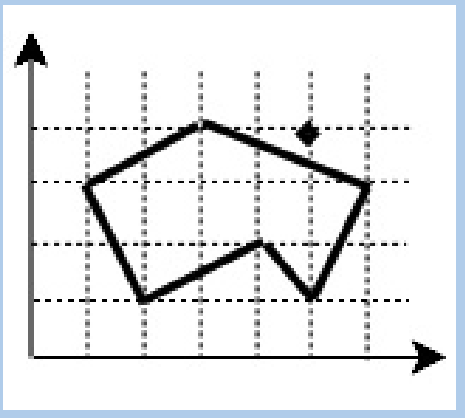
Örnek

guvenlik.gir:

```
6
3 4
6 3
5 1
4 2
2 1
1 3
5 4
```

guvenlik.cik:

```
HAYIR
```



Güvenlik 2

ABC şirketine eski binası yetmeyince şirket yeni binaya taşınmaya karar verir. Yeni taşındıkları bina iki bloktan oluşmaktadır. Güvenliğe önem veren şirket iki bloğa da bir önceki soruda bahsedilen dedektörlerden yerleştirmeye karar verir. Fakat bu kez sistemin farklı bir sorunu vardır. Öyle ki; iki blok kesiştiği için arada kalan alan iki detektör tarafından da algılanacak ve fazladan uyarı alınacaktır. Bunu engellemek için arada kalan alanın sadece birisi tarafından algılanması istenmektedir. Sizden istenen arada kalan alanı bulmanız.

Varsayımlar

- Bloklar dışbükey çokgen şeklindedir. Birinci bloğun *n* köşesi, ikinci bloğun *m* köşesi bulunmaktadır.
- Çokgenler verilirken çokgenin köşeleri saat yönünde ve sırayla verilecektir.
- Arada kalan çokgeni verirken çokgenin köşelerinin birisinden başlayarak sırayla ve saat yönünde vermeniz gerekmektedir.

Girdi

- Girdiler "guvenlik2.gir" isimli dosyadan okunacaktır.
- İlk satırda ilk çokgenin (bloğun şekli) köşelerinin sayısını ifade eden *n* verilecektir.
- Takip eden *n* satırın her birisinde iki adet pozitif tamsayı verilecektir. Bu sayılar sıradaki köşenin x ve y koordinatlarını belirtecektir.
- Takip eden satırda ikinci çokgenin (bloğun şekli) köşelerinin sayısını ifade eden m verilecektir.
- Takip eden *m* satırın her birisinde iki adet pozitif tamsayı verilecektir. Bu sayılar sıradaki köşenin x ve y koordinatlarını belirtecektir.

Çıktı

- Çıktılar "guvenlik2.cik" isimli dosyaya yazılacaktır.
- Çıktının ilk satırında arada kalan çokgenin köşe sayısını ifade eden p verilecektir.
- Takip eden *p* adet satırın her birisinde iki adet reel sayı bulunacaktır (reel sayıların virgülden sonra en fazla iki basamağı basılacaktır). Bu sayılar arada kalan çokgenin köşelerinin x ve y koordinatlarını belirteceklerdir.

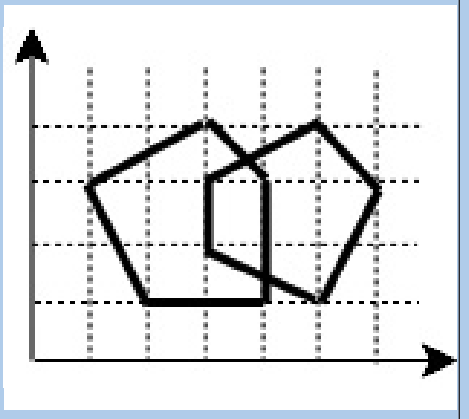
Örnek

guvenlik2.gir:

```
5
1 3
3 4
4 3
4 1
2 1
5
6 3
5 1
3 2
3 3
5 4
```

guvenlik2.cik:

```
5
4 3
4 1.5
3 2
3 3
3.66 3.33
```



Geçen Sayımızdaki Soruların Çözümleri

Mantıksal İfadeler

Bu problemi ünlü bilgisayar bilimcilerinden olan Dijkstra'nın algoritması ile çözebiliriz. Bu algoritma verilen matematiksel bir ifadeyi hesaplamak için kullanılır. Algoritmada, geçen sayıda bahsettiğimiz veri yapılarından yığın kullanılır. Algoritmanın bizim örneğimize uygulamasından bahsedecek olursak:

0. İki adet yığın açılır, bunlardan birisi operatörler için (!, & ve |), diğeri ise önermeler için.

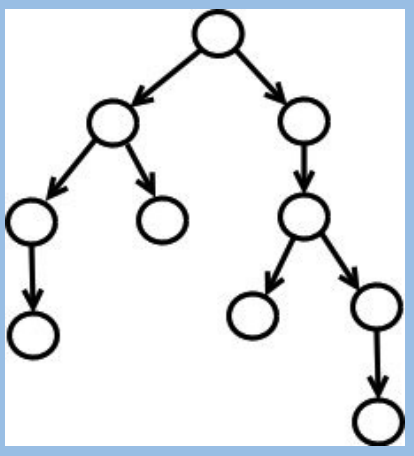
1. İfadenin başından başlanarak sırayla ilerlenir.

2. Sıradaki bir önerme ise değeri önerme yığınına koyulur.

3. Sıradaki bir aç parantez ise operatör yığınına koyulur.

4. Sıradaki bir operatör ise operatör yığınına bakılır.

a. Eğer en üstteki elemanın önceliği kendisininkinden küçük ise sıradaki operatör, operatör yığınına koyulur.



b. Eğer en üstteki elemanın önceliği kendisininkiyle aynı ise:

i. Bu operator ! ise, operator yığınına koyulur

ii. Bu operatör | veya & ise operatör yığınının en üstündeki eleman çekilir, önerme yığınından iki adet eleman çekilir, bu elemanlar operatör yığınından çekilen operatöre göre işleme sokulup sonucu önerme yığınına basılır ve sıradaki operatör operatör yığınına basılır.

c. Eğer en üstteki elemanın önceliği kendisininkinden büyükse, en üstteki eleman bu yığından çekilir.

i. Eğer çekilen eleman ! ise, önerme yığınından bir eleman çekilip ! ile işlenmiş hali önerme yığınına koyulur.

ii. Eğer & veya | ise önerme yığınından iki eleman çekilip bu operatöre göre işleme sokulur ve sonucu tekrar önerme yığınına koyulur Sıradaki operatör operatör yığınına koyulabilene kadar bu işlem devam eder.

5. Sıradaki bir kapa parantez ise, operatör yığınında aç parantez görene kadar operatör yığınından bir eleman çekilir:

a. ! ise önerme yığınından bir eleman çekilerek elemanın ! ile işlenmiş hali önerme yığınına tekrar basılır.

b. & veya | ise önerme yığınından iki eleman çekilerek bu iki elemanın bu operatör ile işlenmiş hali tekrar önerme yığınına basılır

6. Eğer ifadenin sonuna geldiysek yukardaki işlem operatör yığını boşalana kadar devam ettirilir.

7. En sonunda önerme yığınında kalan sonuç bu ifadenin sonucudur.

Mantıksal İfadeler 2

İki çözüm yolundan gidebiliriz:

1. Olası bütün önerme değer dizileri denenir. Verilen sonucu üretenler basılır. Verilen örneğimizi hatırlayacak olursak:

3
p q s
p & q | !p & s
D

Sırasıyla:

YYY, YYD, YDY, YDD, DYY, DYD, DDY, DDD

denenir. Bütün bu olası değer dizilerini şu şekilde üretebiliriz. 0'dan 2n 'e kadar olan sayıların ikili yazılımlarını düşünelim:

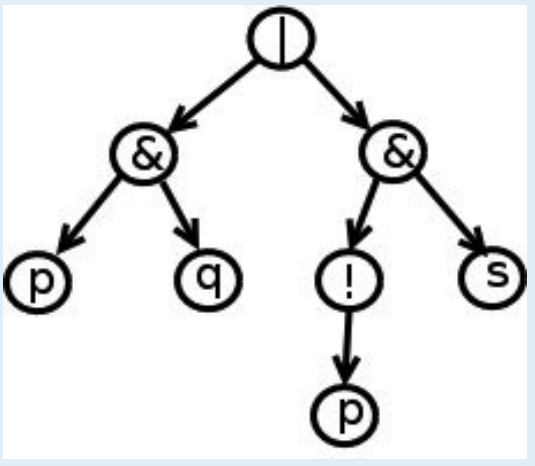
000, 001, 010, 011, 100, 101, 110, 111

Burada 1'leri D, 0'ları Y gibi düşünürsek üstteki dizilimleri elde etmiş oluruz. Daha sonra sırasıyla bütün değer dizilerini bir önceki soruda yazdığımız algoritmayı kullanarak değerlendiririz ve sonucu veriyorsa basarız.

2. Bir önceki soruda bahsedilen Dijkstra'nın algoritmasını biraz düzenlememiz gerekecek. Ama daha önce ağaç ismi verilen veri yapısından bahsedelim. Ağaç bir kökten veya bir kök ve köke bağlı ağaçlardan (çocuklardan) oluşan veri yapısıdır. Şekilde bir ağaç örneği görebiliriz:

Bu ağaç üzerinde özyinelemeli bir algoritma kullanacağız. Yani algoritmayı çalıştırırken önce varsa sağ çocuğu için çalıştırırız, sonra varsa sol çocuğu için çalıştırırız, çıkan sonuçları kendi kökündeki elemana göre işleyip sonucu buluruz.

Bir önceki soruda bahsedilen algoritmada önerme yığınına önermenin değerini değil de önermeler ve operatörlerden oluşan bir ağaç şeklinde basarsak en sonunda önerme yığınında bir ağaç kalır. Verilen örneğimiz için şekildeki gibi bir ağaç oluşur:



Bu ağacı oluşturduktan sonra:

1. Ağacın kökündeki bir önerme ise, o ağacın alması gereken değer önermeye verilir.

2. Ağacın kökündeki ! ise, ağacın alması gereken değerin değil'ini çocuğun alması istenir ve algoritma çocuk için uygulanır.

3. Ağacın kökündeki & ise:

a. Ağacın alması gereken değer D ise, ağacın sol ve sağ çocuğunun D değeri alması istenir ve algoritma çocuklar için uygulanır.

b. Ağacın alması gereken değer Y ise

i. Ağacın sol çocuğunun D, sağ çocuğunun Y olması istenir ve algoritma çocuklar için uygulanır.

ii. Ağacın sol çocuğunun Y, sağ çocuğunun D olması istenir ve algoritma çocuklar için uygulanır.

iii. Ağacın iki çocuğunun da Y olması istenir ve algoritma çocuklar için uygulanır.

4. Ağacın kökündeki | ise:

a. Ağacın alması gereken değer Y ise, ağacın sol ve sağ çocuğunun Y değeri alması istenir ve algoritma çocuklar için uygulanır.

b. Ağacın alması gereken değer D ise

i. Ağacın sol çocuğunun D, sağ çocuğunun Y olması istenir ve algoritma çocuklar için uygulanır.

ii. Ağacın sol çocuğunun Y, sağ çocuğunun D olması istenir ve algoritma çocuklar için uygulanır.

iii. Ağacın iki çocuğunun da D olması istenir ve algoritma çocuklar için uygulanır.

Bu şekilde algoritmayı sonuna kadar uygularsak olası bütün değerleri buluruz. Burada dikkat etmemiz gereken tek nokta, aynı önermeye ağacın farklı dallarında farklı değerler vermemektir.