

# Game Theory

## Giriş

Üzerinde duracağımız oyunlar; iki tarafın da rakibinin bütün muhtemelen hamlelerini görebildiği, şansa dayanmayan, iki taraf da **mükemmel** oynayacağı için kazananın baştan belli olduğu oyunlardır. Sonlu olan her oyunun kazanma stratejisi vardır, burada bazı basit oyunların kazanma stratejilerinden bahsedeceğiz.

## Kazanmak-Kaybetmek

Oyunun sona erdiği durumlar vardır, ortadan taş çekemeyen, hamle yapamayan, taşlarını kaybeden vs. oyunu kaybeder gibi. Hamle yapamayanın mağlup sayıldığı bir oyunda, rakibi hamle yapamayacak duruma sokma fırsatı elde etmek, kazanmak demektir. Yani siz hamlenizi yaptığınızda rakibinize 'kaybeden durum' kalıyorsa, sizin durumunuz 'kazanan durum'dur. Eğer bir durumda, oyunu kaybeden duruma çevirebilecek bir hamle yapma olanağı varsa o durum 'kazanan durum'dur. Çünkü siz hamlenizi yaptığınızda rakibiniz kaybeder.

Bir durumdayken, yapılan farklı hamleler sonucu farklı durumlara ulaşılabilir. Biz oynadıkten sonra sıra rakibimize geleceği için, oyunun vaziyetini 'kaybeden durum'a götürecek hamlelerden birini yapmalıyız.

**Kısaca, o anki halden kaybeden duruma bir geçiş mümkünse, o durum 'kazanan durum'dur.** Aşağıdaki oyun bunun bir örneğidir.

-->Ortada N tane taş var, her hamlede 1,3 veya 4 taş çekiliyor. Hamle yapamayan kaybediyor. Diğer bir deyişle son taşı alan kazanıyor. Öyleyse;

- **N = 0**, kaybeden durumdur. O anda sıra kimdeyse hamle yapamaz, rakibi kazanır.
- **N = 1, N = 3, N = 4** kazanan durumlardır. Sıra bizdeyken ortadaki **N** taşın hepsini alır, rakibimize taş bırakmıyoruz.
- **N = 2** kaybeden durumdur, çünkü ortadan 1 taş almak zorundayız, sıra rakibimize gelince diğer taşı alır, kaybederiz.
- **N = 5** kazanan durumdur, çünkü ortadan 3 taş alırsak, sıra rakibe geçtiğinde 2 taş kalır, **N = 2** kaybeden durum olduğu için biz kazanmış oluruz.

- **N = 6** kazanan durumdur, çünkü ortadan 4 taş alırsak, sıra rakibe geçtiğinde 2 taş kalır, **N = 2** kaybeden durum olduğu için biz kazanmış oluruz. **Dikkat**, 1, 3 ve 4 taş alabilirdik ortadan. 1 taş alsaydık rakibimize **N = 5** durumunu bırakmış olacaktık, yani kazanan durumu. 2 taş alsaydık rakibimize **N = 4** durumunu bırakmış olacaktık, bu da kazanan durumdu. Bu yüzden 4 taş aldık, rakibimize **N = 2** kaybeden durumunu bırakıp oyunu kazanmış olduk.
- **N = 7** kaybeden durumdur, çünkü ortadan 1, 3, 4 taş alalım, rakibimize her türlü kazanan bir durum bırakmak zorundayız.
- Özetleyecek olursak, herhangi bir **X** için, eğer **X-1**, **X-3** ve **X-4** durumlarından herhangi biri kaybeden durumda **X** kazanan bir durumdur, aksi takdirde kaybeden bir durumdur.

Bu soru için örnek kodlar:

```

    ...
    cin >> N ;

    win[0]=0;
    win[1]=1;
    win[2]=0;
    win[3]=1;
    win[4]=1;

    for(int i=5;i<=N;i++){
        ...
        if( win[i-1] == 0 || win[i-3] == 0 || win[i-4] == 0 )
            win[i] = 1;
        else
            win[i] = 0;
    }

```

```

int f( int N ){

    if(N==0 || N==2) return 0;
    if(N==1 || N==3) return 1;

    if(f(N-1)==0 || f(N-3)==0 || f(N-4)==0) return 1;
    return 0;
}

```

**Örnek Soru :** [http://community.topcoder.com/stat?c=problem\\_statement&pm=6856](http://community.topcoder.com/stat?c=problem_statement&pm=6856)

**İpucu :** Bu tarz problemlerin sonuçları, bir yerden sonra periyodik olarak tekrar etmeye başlar.

## Nim

Nim, çeşitli varyasyonları olan, birçok matematik probleminde geçen bir oyundur. Ortada duran N kasenin her birinde çeşitli sayıarda bilyeler vardır. Her hamlede sırası gelen oyuncu bir kase seçer, ve seçtiği kaseden dilediği miktarda bilye alır. Son taşı alan kazanır, hamle yapamayan kaybeder. Her hamlede en az bir bilye alınmak zorunda. Kim kazanır?

**Cevap:** Torbalardaki taş sayılarına  $N_1, N_2, N_3, \dots, N_k$  diyelim. **Ancak ve ancak  $N_1 \text{ xor } N_2 \text{ xor } N_3 \text{ xor } N_4 \dots \text{ xor } N_k = 0$  olduğu durumda ikinci oyuncu kazanır.**

Torbaların sayılarını ikilik tabana çevirip alt alta yazdığını düşünün. Örneğin:

$$\begin{array}{rcl} 6 = (110)_2 & & 1 \ 1 \ 0 \\ 9 = (1001)_2 & & 1 \ 0 \ 0 \ 1 \\ 3 = (11)_2 & & 1 \ 1 \\ \hline & & 1 \ 1 \ 0 \ 0 \end{array}$$

**Önemli:** Bir sütunda çift sayıda 1 biti varsa o sütunun xor'u 0 olur. Tek sayıda ise 1 olur. Bunu iyice oturtup aşağıdaki açıklamaya geçebiliriz.

- **Kaybeden durumlardan yalnızca kazanan durumlara yol vardır:**

- Eğer kaybeden durumdaysak, yani sayılarımızın xor'u 0 ise, biz hamle yaptığımızda sayılarından biri azalacağı için, sonuç xor'umuz değişecek. Çünkü sayılarımızda birinin azalması demek, **en az bir** adet bitinin 1'den 0'a dönüşmesi demektir.

- **Kazanan durumlardan en az bir tane kaybedene yol vardır:**

- Eğer kazanan durumdaysak, yani sayılarımızın xor'u 0'dan farklı ise, sonuç xor'umuzdaki en soldaki 1 bitini bulalım. **Önemli** dediğim nottaki gibi, tüm olay o sütundaki 1'lerin sayısının tek veya çift olması. En soldaki 1 bitini bulduğumuzda, o sütunu 1 olan sayılarımızdan birini alalım, o torbadan taş çekersekiz çünkü. Daha sonra o sayının o bitini 0 yapalım, bu da sayıyı azaltmak demek oluyor, yani hamlemizi yapıyoruz. Böylece sonuç xor'umuzun o en soldaki 1 olan biti artık 0 oldu. (teklik çiftlik durumunu hatırlayın). Eğer sıkıntı çıkarın başka bitler varsa, taş çekmek için seçtiğimiz torbayı temsil eden sayının bitleriyle dilediğimiz gibi oynayarak diğer sütunları da düzeltebiliriz. Çünkü o en soldaki 1 bitini 0'a dönüştürdüğümüze göre artık sağ taraftaki bitlerde nasıl bir değişiklik yaparsak yapalım sayı yine azalmış

olma durumunu koruyacaktır. Diğer bitlerdeki sıkıntıları da düzeltip, bütün 1 olan sütunları 0 yapıp, sonuç xor'umuzu 0'a eşitledikten sonra sıramızı tamamlayıp rakibimize xor'ları 0 olan bir durum bırakıyoruz. Yani kaybeden durumu.

### Örnek kod:

```
int main() {
    ...
    cin >> N ;
    ...
    int res=0;
    for(int i=1;i<=N;i++) {
        ...
        scanf(" %d",&ar[i]);
        ...
        res^=ar[i];
    }
    if(res==0) puts("2");
    else puts("1");
    return 0;
}
```

### Örnek Sorular:

[http://community.topcoder.com/stat?c=problem\\_statement&pm=7424](http://community.topcoder.com/stat?c=problem_statement&pm=7424)

[http://community.topcoder.com/stat?c=problem\\_statement&pm=6239](http://community.topcoder.com/stat?c=problem_statement&pm=6239)

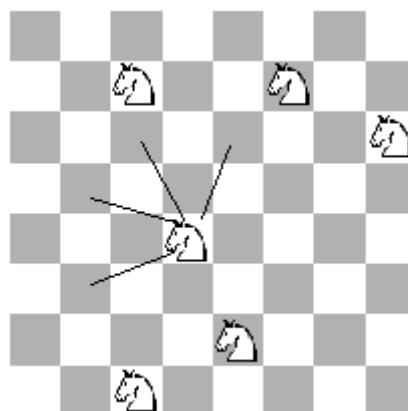
İkinci problem biraz zor olabilir. İpucu: Grupların sayıları arasındaki farklara dikkat edin.

[http://community.topcoder.com/tc?module=Static&d1=match\\_editorials&d2=srm309](http://community.topcoder.com/tc?module=Static&d1=match_editorials&d2=srm309)

Bu da çözümü, kolay gelsin.

## Grundy Numbers

**N x N** lik bir satranç tahtası üzerinde **K** adet at olduğunu düşünün. Aynı karede birden fazla at bulunabiliyor, hamle yapamayan kaybediyor. Ancak bildiğimiz at gibi değil, bunlar biraz kısıtlanmış hareket ediyorlar. Şekildeki gibi:



Aynı anda bir karede bulunan at sayısı hakkında bir sınır bulunmadığı için, isterseniz K farklı satranç tahtası düşünüp her birinde 1 tane at varmış gibi de oynayabilirsiniz.

Bu problemi çözerken her kareye bir grundy number vereceğiz. **Grundy number, o kareden ulaşabildiğimiz karelerin grundy numberlarından farklı olan en küçük sayıya eşittir.** Daha sonra her bir atın bulunduğu karenin grundy numberlarını xor'layıp, 0 olup olmamasına göre kazananı belirleyeceğiz. Çünkü bu problem, o noktadan sonra aslında Nim denk oluyor. Nim oynamayı biliyorsak bunu da oynarız.

0	0	1	1	0	0	1	1
0	0	2	1	0	0	1	1
1	2	2	2	3	2	2	2
1	1	2	1	4	3	2	3
0	0	3	4	0	0	1	1
0	0	2	3	0	0	2	1
1	1	2	2	1	2	2	2
1	1	2	3	1	1	2	0

Grundy numberlarının tablosu.

Grundy numberlarımızı xor'ladıkten sonra elde ettiğimiz sayı 0 ise ikinci oyuncu, değilse birinci oyuncu kazanacaktır. Tıpkı nim gibi.

## Bu problemi Nim'e nasıl indirgeriz?

- Nim'de bir gruptaki taş sayısını A'dan B'ye düşürebiliyorsak, bu oyunda da grundy number'i A olan bir kareden B olan bir kareye gidebiliriz demektir. Bütün  $A > B$  durumları için geçerlidir bu. Çünkü bir karenin grundy number'ını, oradan ulaşamadığımız en küçük sayı olarak tanımlamıştık. Bu yüzden A'dan küçük bütün sayılarla ulaşabiliriz. Eğer ulaşamadığımız herhangi biri olsaydı zaten grundy number olarak A'yı değil o sayıyı alırdık.
- Eğer bir at 0'dan büyük olan bir karedede bulunuyorsa, kendisinden küçük olan bir kareye ilerleyebiliriz. Nim'deki taş çekme gibi, burada da grundy number'ımızı küçültebiliriz.
- Eğer bir at 0 olan bir karedede duruyorsa ve daha büyük bir kareye hareket ederse, her zaman bir hamle daha yaparak tekrar 0 olan bir kareye döndürmenin yolu vardır. Böylece 0'dan çıkışın daha yüksek bir sayıya giden bütün hamleleri bu şekilde yok sayabiliriz.

Örnek Soru:

[http://community.topcoder.com/stat?c=problem\\_statement&pm=2987&rd=5862](http://community.topcoder.com/stat?c=problem_statement&pm=2987&rd=5862)

**Kayıbeden duruma yol varsa orası kazanan durumdur. Ancak ve ancak, oradan sadece kazanan durumlara yol varsa orası kaybeden durumdur.** Kuralı her zaman işe yarar. Oyun soruları bazen korkutucu olabilir. Ama genelde yukarıda dejindiğimiz problemlerin türevi olurlar. Ayrıca genelde kodları çok kısa ve kolay olur. Korkmadan sakin sakin düşünün, çözürlüler :)  
Aşağıdaki örneklerde benzer şeyler çıkabilir karşınıza. Evirip çevirip aynı şeyler sorarlar.

- Eğer birden fazla at seçme şansı olsaydı, ve seçilen atların hepsi oynanma zorunluluğu olsaydı?**
- Ancak ve ancak bütün atların bulunduğu kareler 0 olursa orası kaybeden durum olur.
- Eğer birden fazla at seçebilseydik, ama hepsini birden seçme iznimiz olmasaydı ne olurdu?**
- Ancak ve ancak bütün atların grundy numberları aynıysa kaybeden durum olur.

**Ödev: Eğer bir hamlede bütün atları oynamaya mecburiyetimiz olsaydı ne olurdu?**