YZM 3217- YAPAY ZEKA

DERS#6: REKABET ORTAMINDA ARAMA

Oyun Oynama

- **Çoklu vekil ortamı**-her bir vekil karar verirken diğer vekillerin de hareketlerini dikkate almalı ve bu vekillerin onun durumunu nasıl etkileyeceğini bilmelidir.

- Olasılık-diğer vekillerin hareketlerinin tahmin edile bilmemesi
  - –“Önceden tahmin edilemeyen" karşı taraf
  - – Rakibin her olası cevabına karşı bir hareketin belirlenmesi

Oyun Oynama

- Soyut ve önemli bir Yapay Zeka problemi
- Yapay Zekanın en eski uygulama alanlarından biri
  - Zeka gerektiren rekabetin soyut ifadesi
  - Durum ve etkinliklerin kolay ifade edilebilirliği
  - Dış dünyadan çok az bilginin gerekli olması
- Birçok durumda tamamen erişilebilir bir durum uzayı ile karşı karşıyayız.

Oyun Oynama

- **Zorlayıcı bir problem olabilir:** belirsizlik
  - Karşıdaki oyuncunun (rakibin) yapacağı hamleler
  - Problemin karmaşıklığı (tam arama -> mümkün değil) --- **Arama Uzayının Boyutu**

- **Satranç oyunu:** Her durumda yaklaşık 15 hareket, 80 karşılıklı hamle => 15^{80} düğümlü bir arama ağacı
- **Go oyunu:** Her durumda yaklaşık 200 hareket, 300 karşılıklı hamle => 200^{300} düğümlü bir arama ağacı

Oyun Türleri

- **Tam bilgili / Eksik bilgili**
  - Oyuncuların ardışık olarak stratejileri seçtiği ve diğer oyuncuların seçiminin ne olduğunun farkında olduğu bir oyun türüdür.
  - Oyuncuların yalnızca diğer oyuncuların ne yapacağını tahmin ederek, bir başka oyuncunun hamlesini bilmeden hareket etmesidir

- **Belirli / Şansa Dayalı**

Oyun Türleri

- **Satranç / Dama /Go Oyunu:**
  - Tam Bilgili, Belirlenmiş
- **Tavla:**
  - Tam Bilgili, Şansa Dayalı
- **Kart Oyunları:**
  - Eksik Bilgili, Şansa Dayalı

Oyun Türleri

	Belirli	Şansa Dayalı
Tam Bilgili	Satranç, Go	Backgammon, monopoly
Eksik Bilgili	?	Bridge, poker

İki Oyunculu Tam Bilgili Oyunlar

- İki etmen (oyuncu) bulunmaktadır.
- Oyunu, oyunculardan birisinin kazanması ya da oyuncuların berabere kalması şeklinde sonlanır.
- Her bir oyuncu ortama ilişkin tam ve eksiksiz bilgiye sahiptir.

İki Oyunculu Tam Bilgili Oyunlar

Başlangıç Durumu: Oyuna ilişkin ilk durum ve kimin oyuna başlayacağı

Operatörler: Uygulanabilir hamleler

Sonlanma Testi: Oyun Bitti mi?

Fayda Fonksiyonu: Sonuç (Kazandı:+1, Kaybetti:-1, Berabere Kaldı :0 gibi)

- İki oyuncu **(MIN** ve **MAX)** kendi hamlelerini gerçekleştirirken oyunu kazanma şanslarını artırmayı amaçlıyor.
- Bir oyuncunun zaferi, diğerinin yenilgisi anlamına geliyor.
- Dolayısıyla, rakibin yaptığı hamle ne olursa olsun, bize oyunu kazandıracak stratejiyi belirlememiz gerekli!

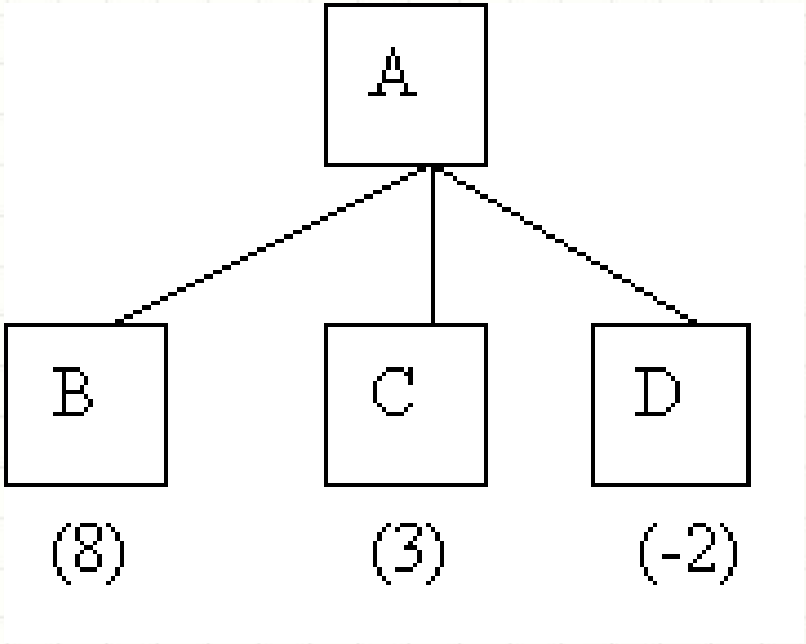
MINIMAX Algoritması

- MINIMAX algoritması, derinlik öncelikli, derinlik sınırlı bir arama prosedürüdür.
- Mevcut durumda, bir sonraki adımdaki olası hamleler oluşturularak devam edilir.
- Ardından, statik değerlendirme fonksiyonu kullanılarak, en uygun hamle seçilir.
- Statik değerlendirme fonksiyonu, oyuncu için yüksek, rakip için düşük değer döndüren bir fonksiyondur.

MINIMAX Algoritması

Yandaki oyun ağacında,

- **10: Oyuncu oyunu kazandı.**
- **-10: Rakip oyunu kaybetti.**
- **0: Oyuncular berabere kaldı.**

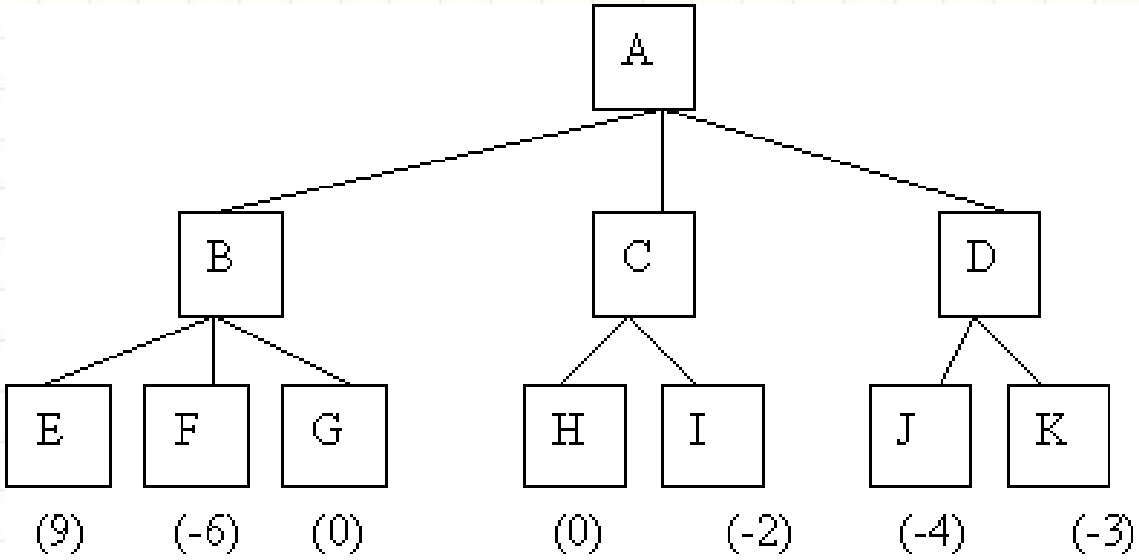


İki oyuncu aynı bilgi seviyesinde olduğu takdirde, sıra oyuncuda olduğunda; statik değerlendirme fonksiyonu değeri yüksek olan; sıra rakipte olduğunda ise; statik değerlendirme fonksiyonu değeri düşük olan hamlenin seçilmesi amaçlanır.

Burada, oyunu kazanma şansımızı artırmak için B'yi seçmeliyiz.

MINIMAX Algoritması

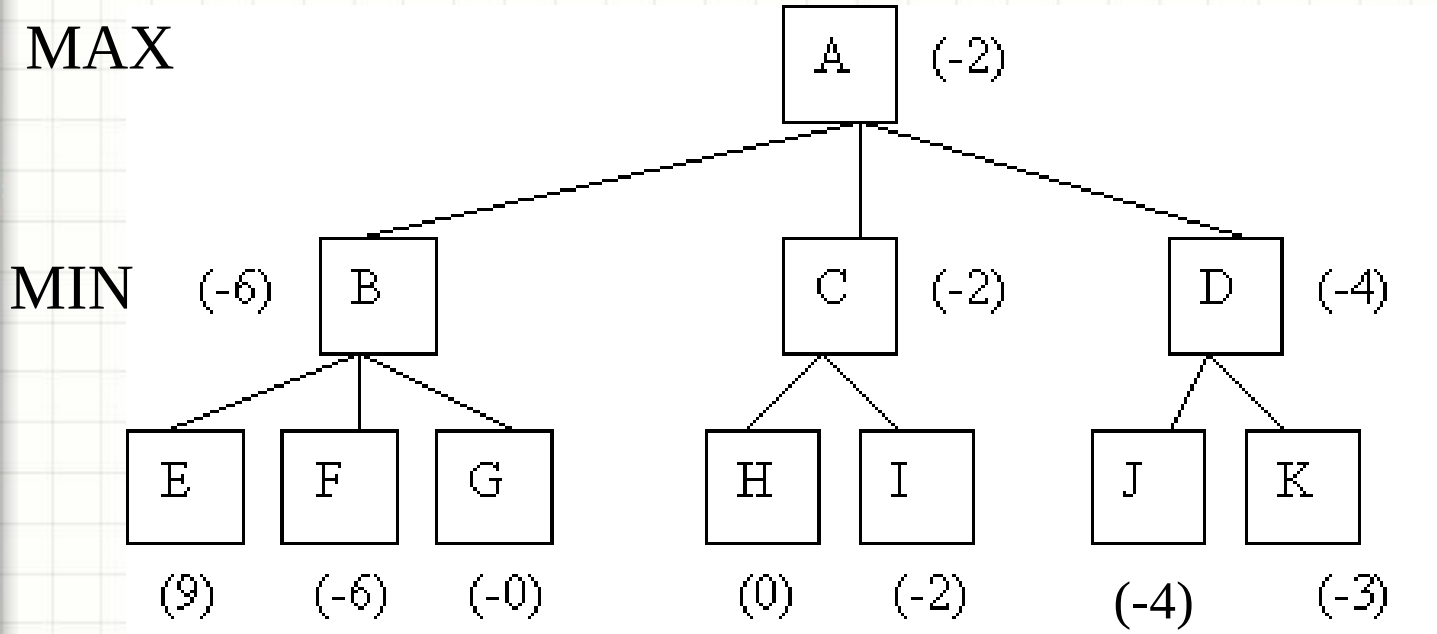
Genellikle arama ağacı **birkaç adım sonraki hamleler** de göz önünde bulundurularak açılır:



Eğer biz B düğümünü seçersek; aslında; F düğümüne erişebilmesi, rakibimizi avantajlı çözüme ulaştırmamız anlamına geliyor.

MINIMAX Algoritması

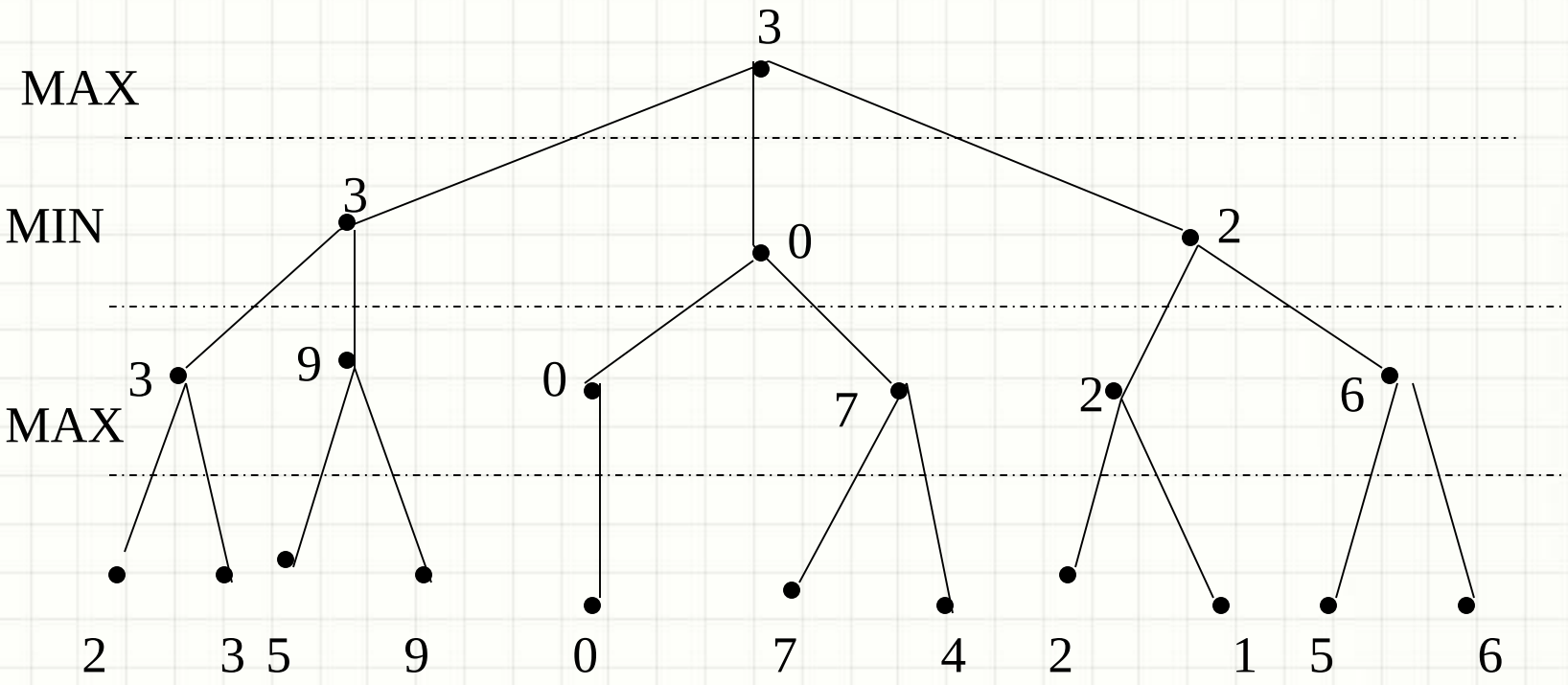
Statik Değerleme Fonksiyonu değerini **yukarıya doğru yaymalı** ve C düğümünü seçmeliyiz.



MINIMAX Algoritması

- Birçok oyun için arama uzayına ilişkin tüm ağacın oluşturulması hesaplama maliyeti bakımından **uygulanabilir değil!**
- Arama sonlandığından, en iyi olası hamleye ilişkin bir kestirimde bulunulmalı.
- Hangi hamlenin seçileceği, ağacın yaprak düğümlerinden başlanarak, statik değerlendirme fonksiyonunun uygulanması ile yapılabilir. Değerlendirme fonksiyonu herhangi bir düğümün değerini temsil eder.
- **MAX oyuncusu:** en yüksek değere; **MIN oyuncusu** en düşük değere doğru hamlelerini yapmayı amaçlar.
- Sıfıra yakın değerlendirme fonksiyonu değerleri, ne MAX ne de MIN oyuncusu için tercih edilebilir değildir.

MINIMAX Algoritması



Tic-Tac-Toe

MAX oyuncusunun ‘X’, MIN oyuncusunun ‘O’ hamleleri yaptığını varsayalım:

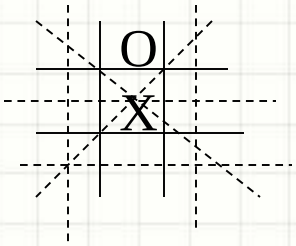
- İlk olarak oyunu **MAX oyuncusu** başlayacak.
- Derinlik sınırı değeri 2 için, tüm olası düğümleri oluşturup statik değerlendirme fonksiyonunu uygulayalım:

Statik değerlendirme fonksiyonu:

- e(p) = **MAX** oyuncusu için halen tamamlanabilir olan toplam satır, sütun ve köşegen sayısı – **MIN** oyuncusu için halen tamamlanabilir olan toplam satır, sütun ve köşegen sayısı
- e(p) = ∞ Eğer p düğümü MAX oyuncusu için kazanma ise.
- e(p) = - ∞ Eğer p düğümü MIN oyuncusu için kazanma ise.

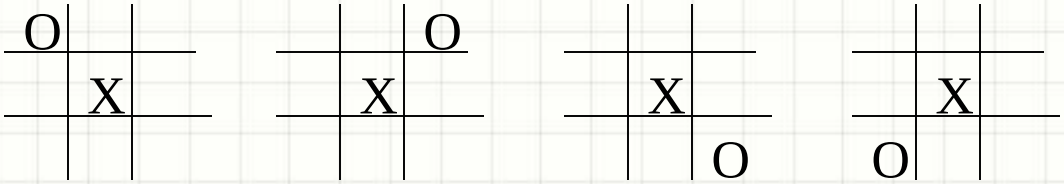
Tic-Tac-Toe

- Aşağıdaki p düğümü için:

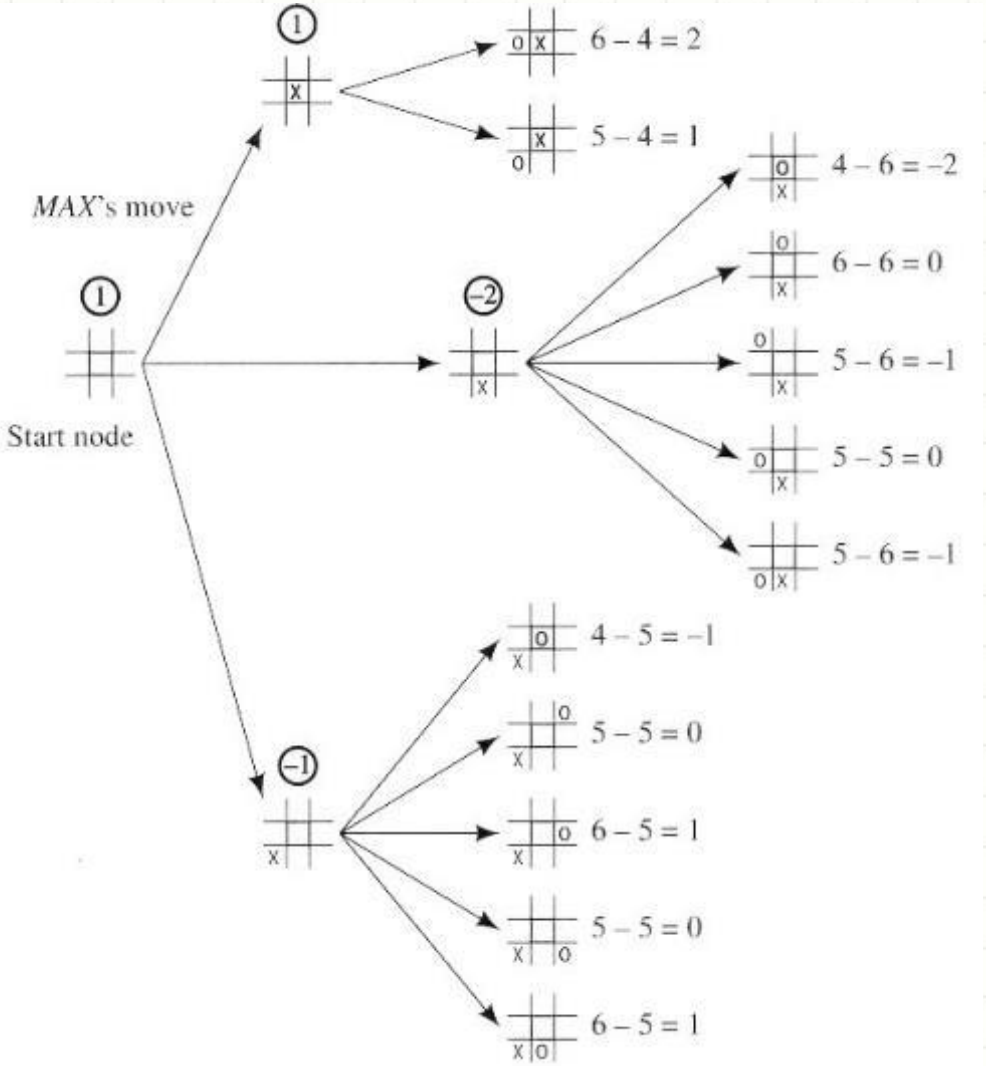


e(p) = 6 –4 = 2

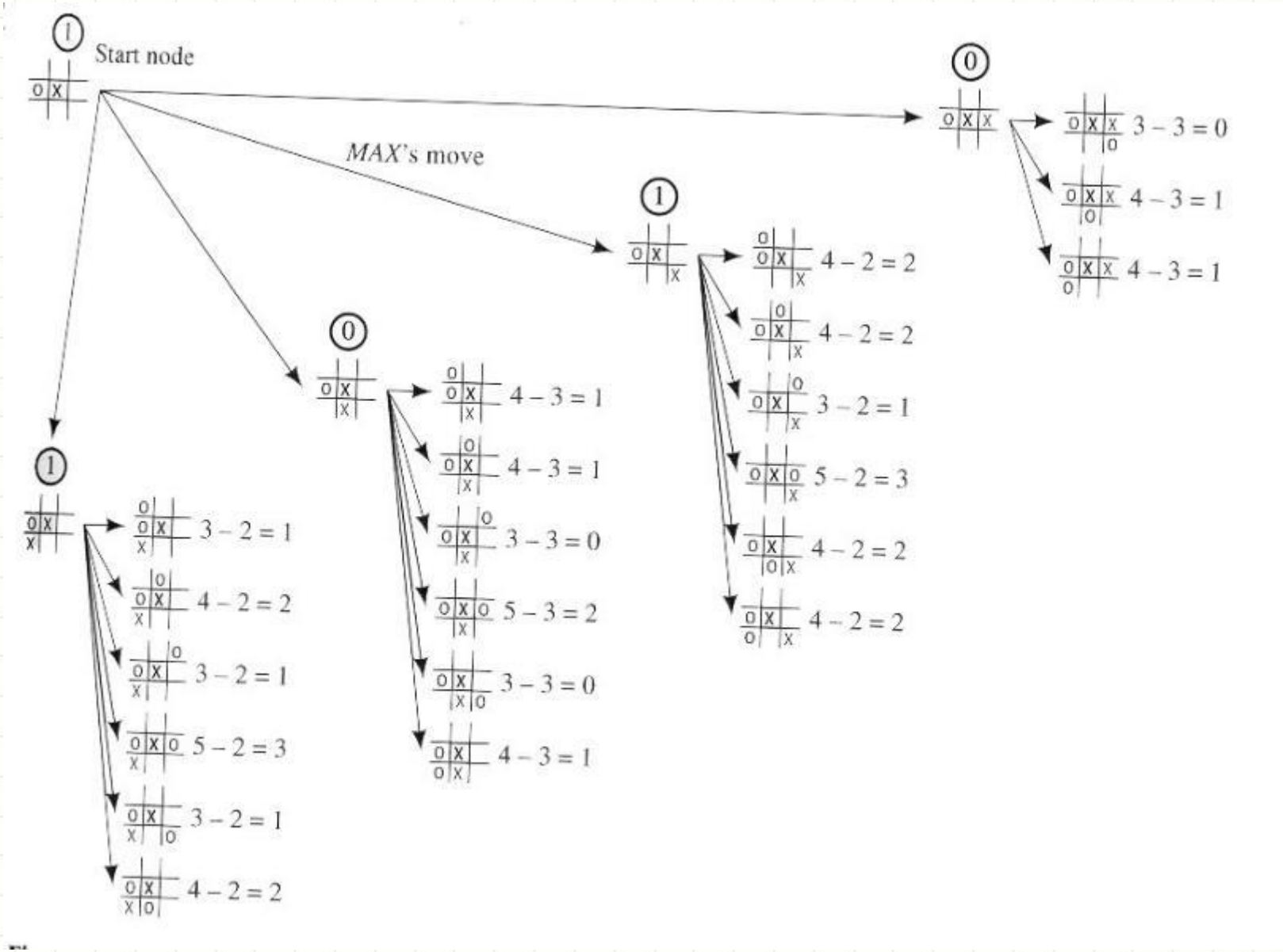
Simetrik durumlar elenebilir:



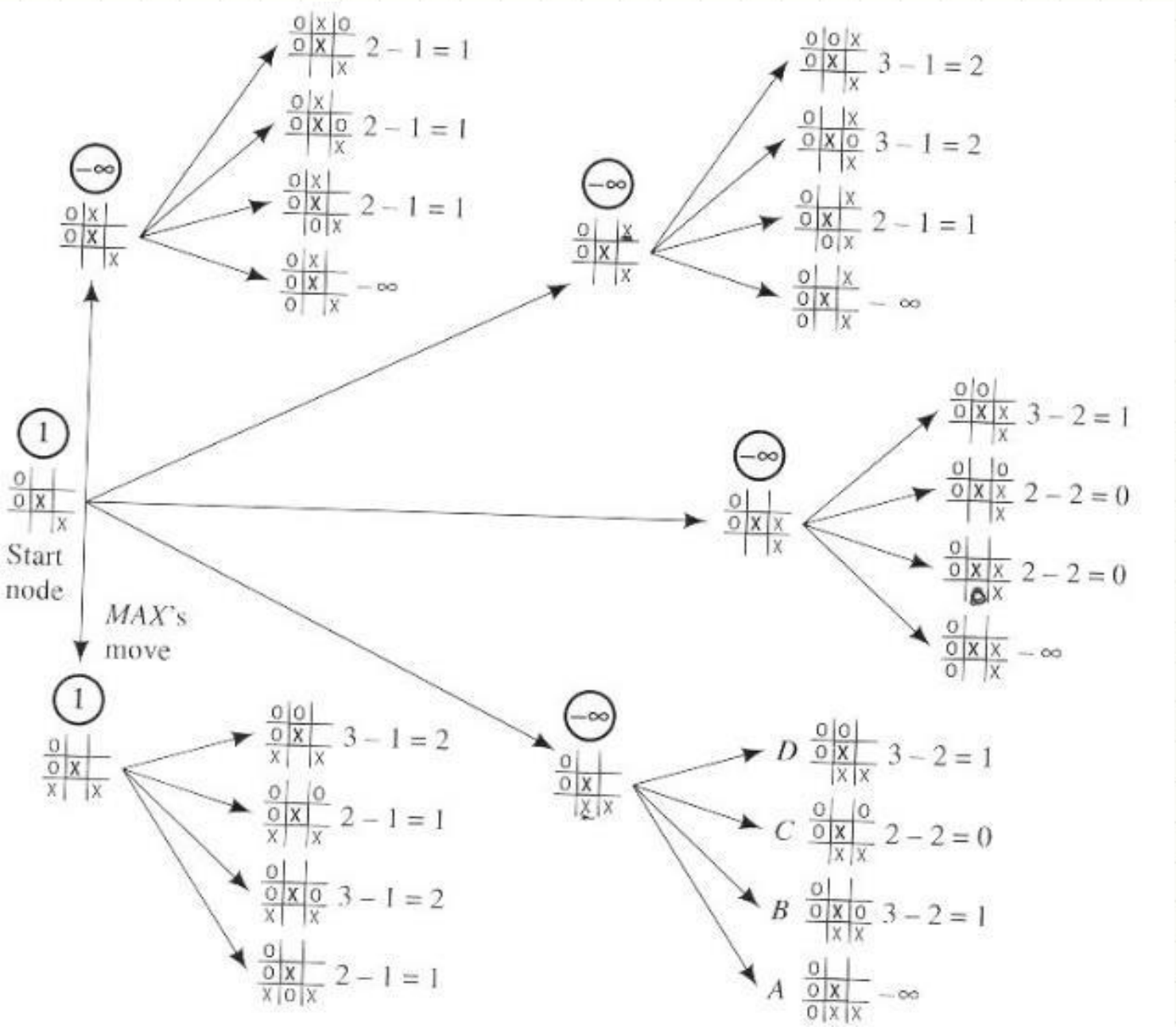
MINIMAX Algoritması



MINIMAX Algoritması



MINIMAX Algoritması



Alfa-Beta Budama Algoritması

- Tüm ağacın (yukarıdan aşağıya doğru derinine) oluşturulmasına ve değerlerin tüm ağaç boyu yayılmasına **gerek kalmayabilir.**

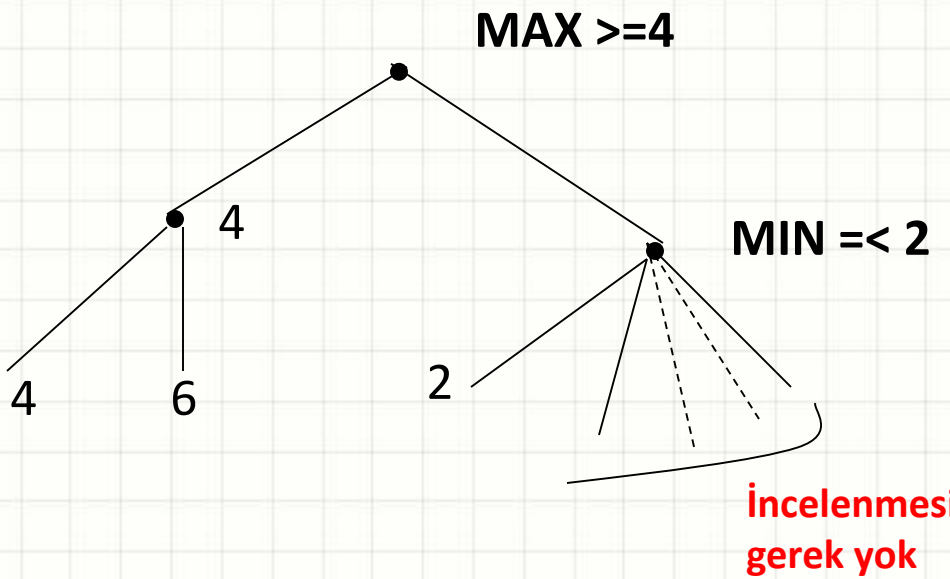
- Edinilmiş bazı değerler ,ağacın üretilmemiş kısımlarının fazla olduğu ve üretilmesine gerek kalmadığı bilgisini verebilir.

- **Temel fikir:** oyun ağacında her bir düğüme bakmadan da doğru çözümü bulmak mümkündür. Bu halde ağacın bakılmayan kısmı budanmış oluyor.

Alfa-Beta Budama Prosedürü

- α **Kesim**

Eğer güncel maksimum değer, kendisinden sonra gelen düğümlerin minimum değerinden büyükse; alt ağaçta geriye kalan düğümlerin incelenmesine gerek yoktur.

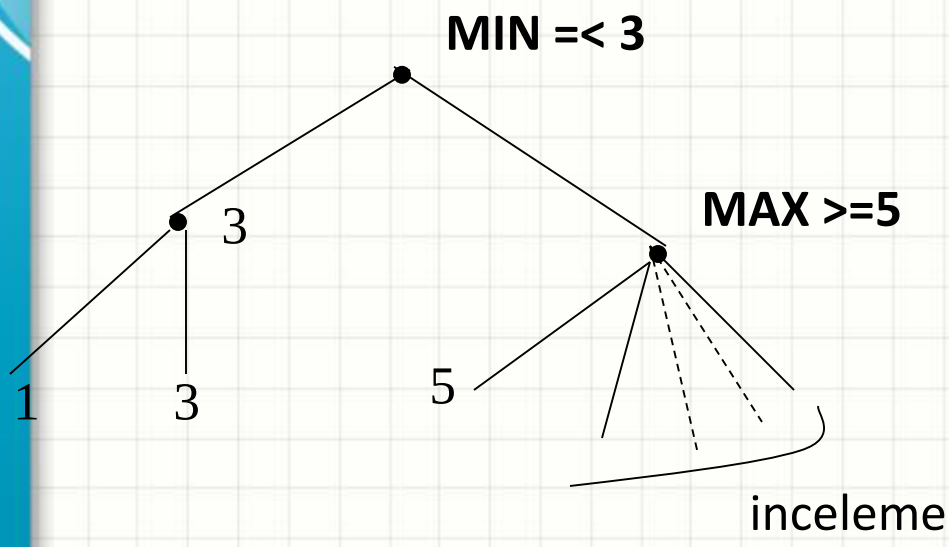


Sağ Alt Ağaç 2 ya da daha düşük bir değer döndürecek. Dolayısıyla, MAX her zaman sol yol üzerinden ilerleyecek.

Alfa-Beta Budama Prosedürü

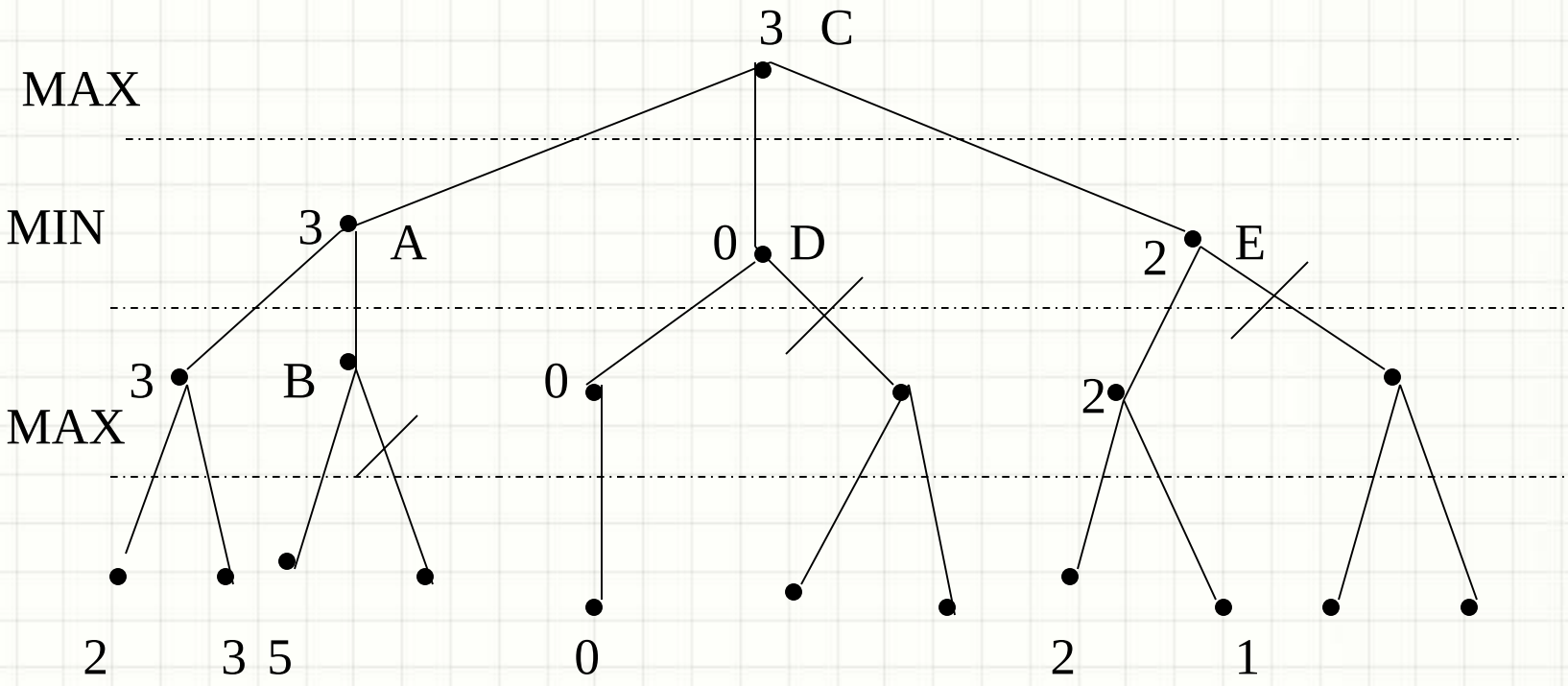
β Kesim

Eğer güncel minimum değer, daha sonra gelen düğümün maksimum değerinden küçükse; o zaman sağ alt ağaca bakmaya gerek yoktur.



Sağ Alt Ağaç en az 5 olabilir. Dolayısıyla, MIN oyuncusu her zaman sol alt ağaç üzerinden ilerleyecek.

Alfa-Beta Budama Prosedürü Örneği



1) A: β= 3

B: β budanmış

2) C α = 3

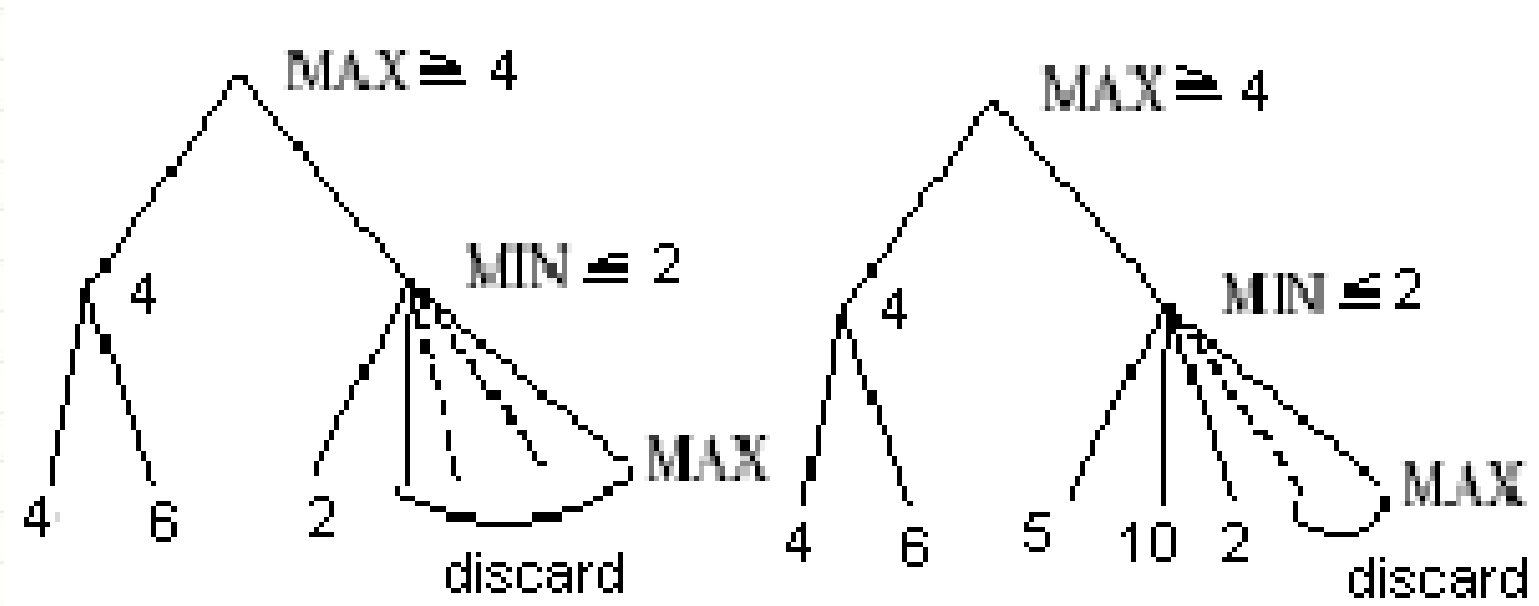
D : α budanmış, 0<3

E : α budanmış, 2<3

α - β Prosedürünün Arama Etkinliği

- Budama arama algoritmasının son sonucunu etkilemez.
- Alfa-beta budama algoritmasının işletilebilmesi için, arama ağacının en azından bir bölümünün en yüksek derinlik derecesine kadar açılmış olması gereklidir.
- Alfa-beta budama algoritmasının performansı, düğümlerin sıralanmasına dayalı olarak artırılabilir.

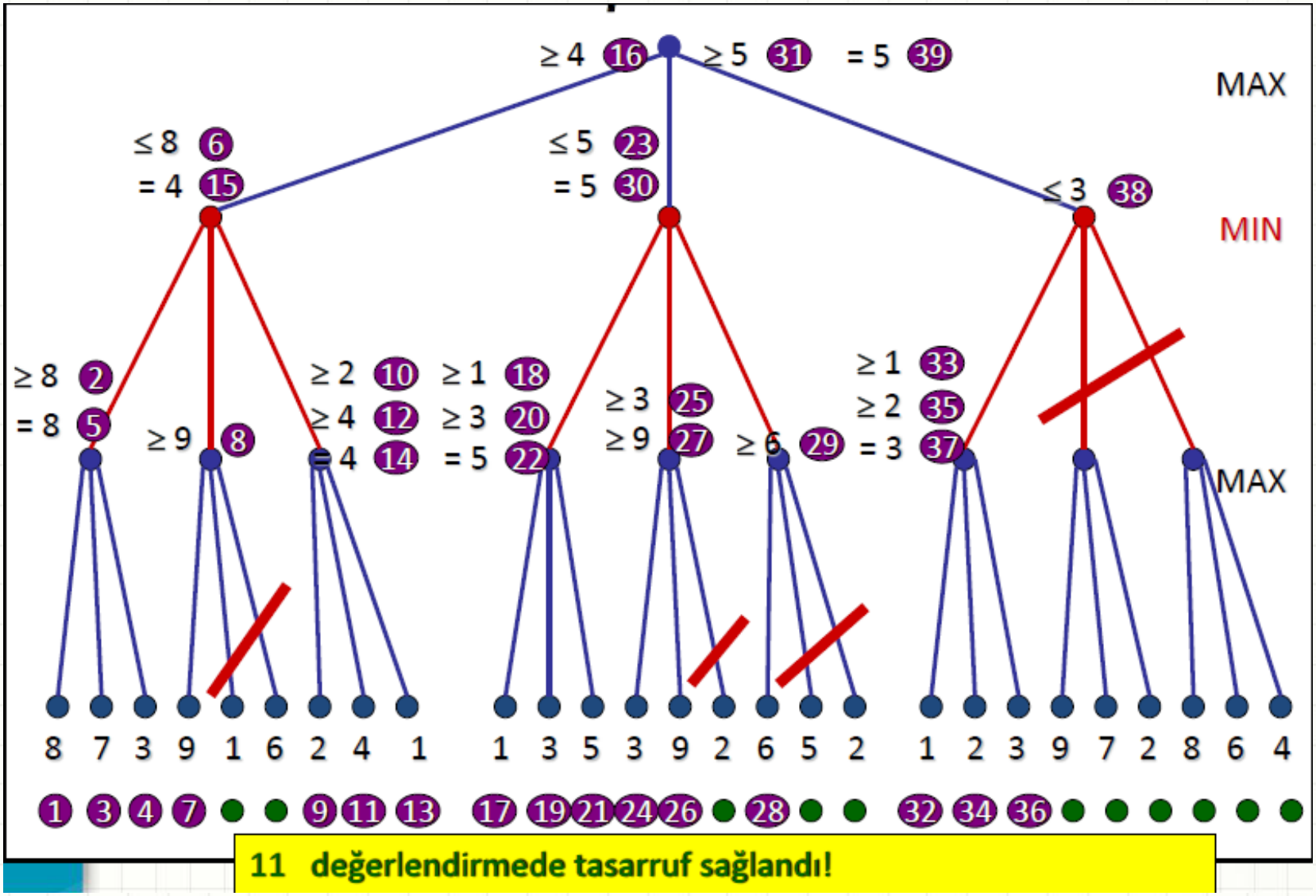
Etkin bir Budama için Sıralama Önemli



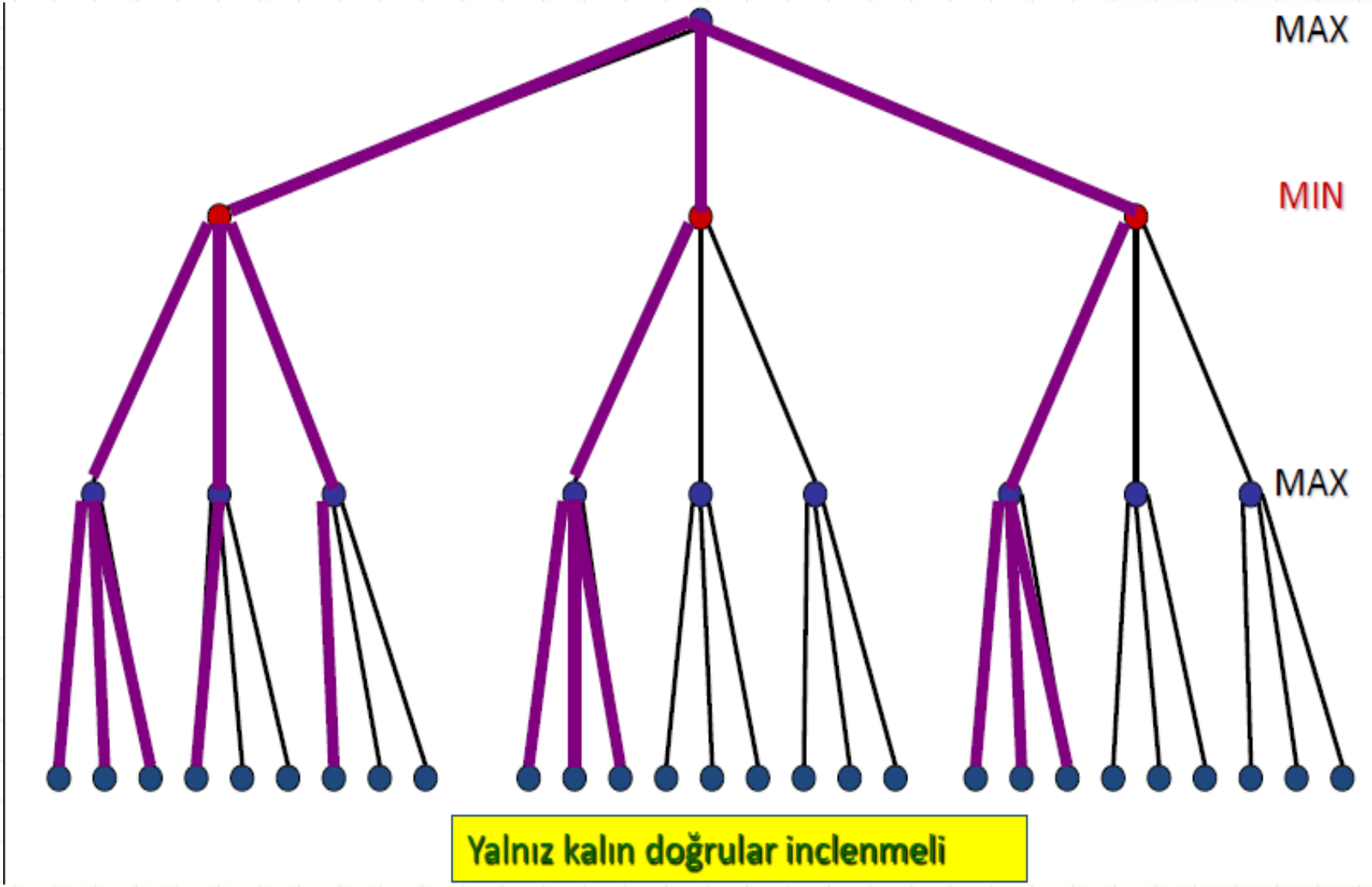
MIN için: Değerleme fonksiyonun artan sırada sıralanması;

MAX için: Değerleme fonksiyonu değerlerinin azalan sırada sıralanması daha iyidir.

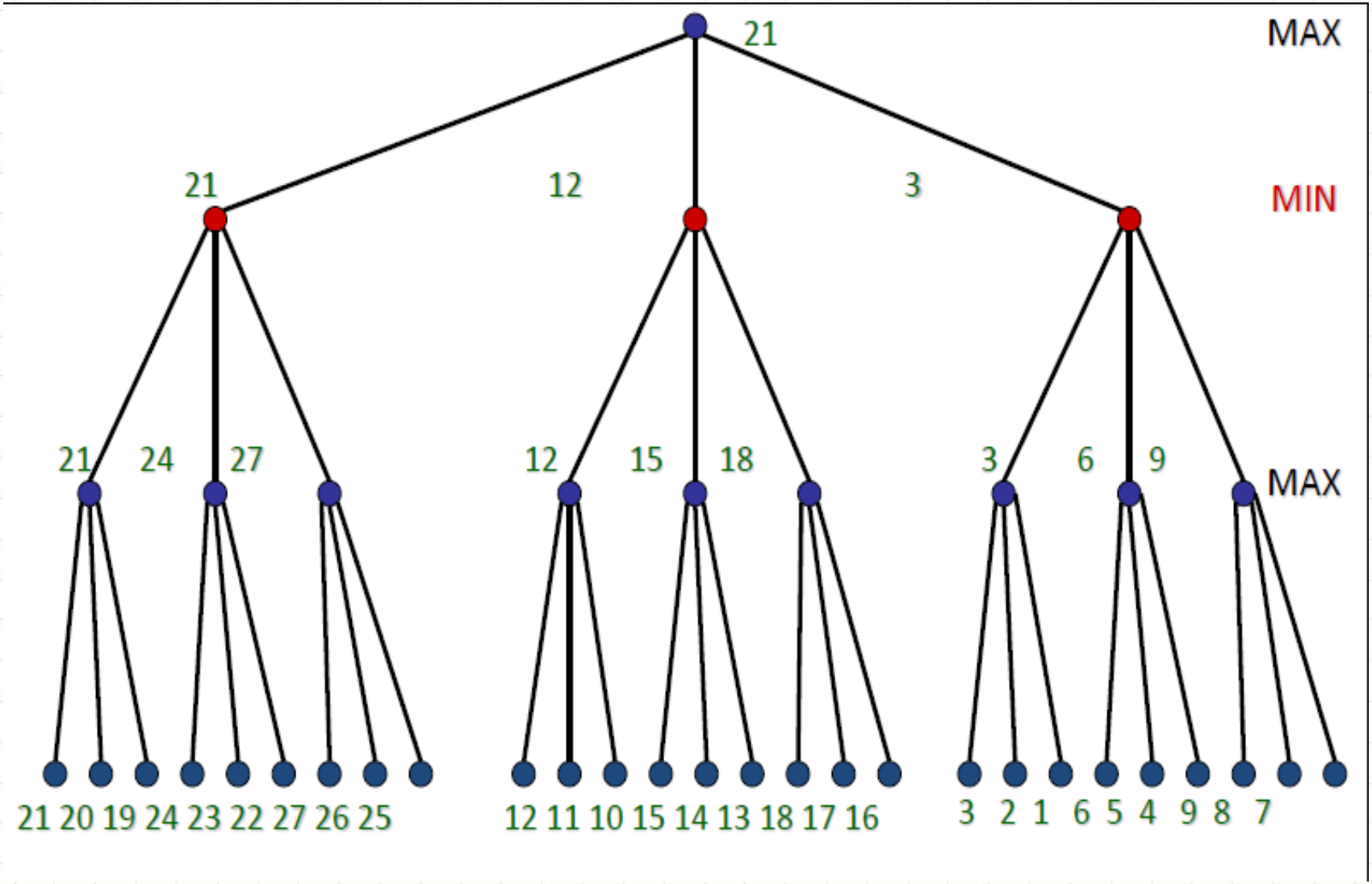
α - β Örneği


≥ 4 16
≥ 5 31
= 5 39
MAX
≤ 8 6
= 4 15
≤ 5 23
= 5 30
≤ 3 38
MIN
≥ 8 2
= 8 5
≥ 9 8
≥ 2 10
≥ 4 12
= 4 14
≥ 1 18
≥ 3 20
= 5 22
≥ 3 25
≥ 9 27
≥ 6 29
≥ 1 33
≥ 2 35
= 3 37
MAX
8 7 3 9 1 6 2 4 1
1 3 5 3 9 2 6 5 2
1 2 3 9 7 2 8 6 4
1 3 4 7 9 11 13 17 19 21 24 26 28 32 34 36
11 değerlendirmede tasarruf sağlandı!

α - β Örneği



α - β Örneği


21
MAX
21
12
3
MIN
21
24
27
12
15
18
3
6
9
MAX
21 20 19 24 23 22 27 26 25
12 11 10 15 14 13 18 17 16
3 2 1 6 5 4 9 8 7