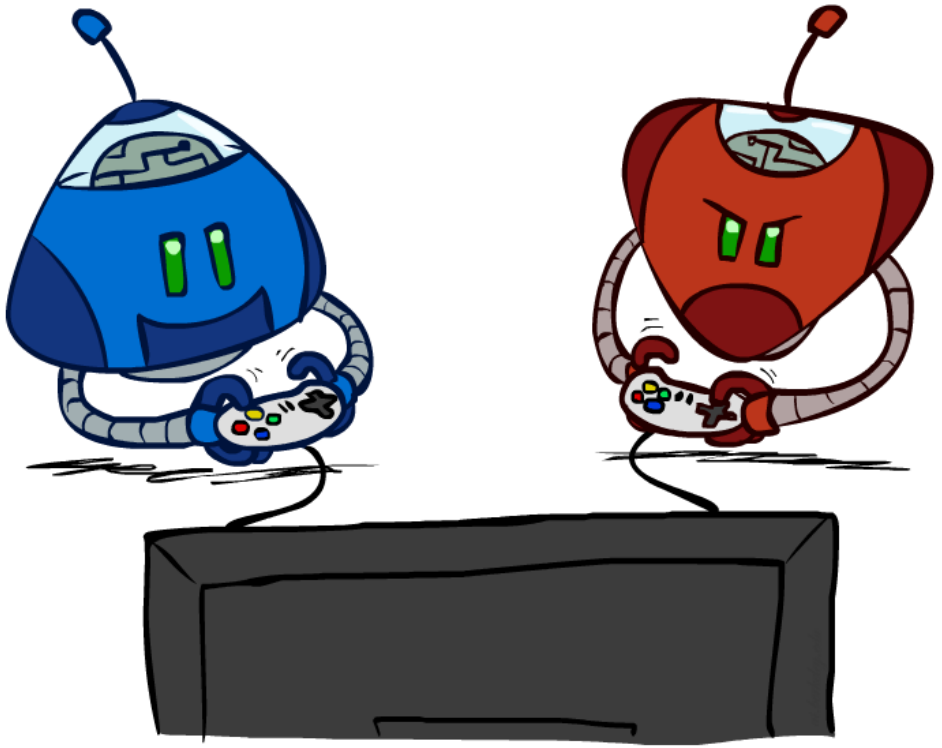
Yapay Zeka

Rekabet Ortamında Arama ve Oyun Ağaçları

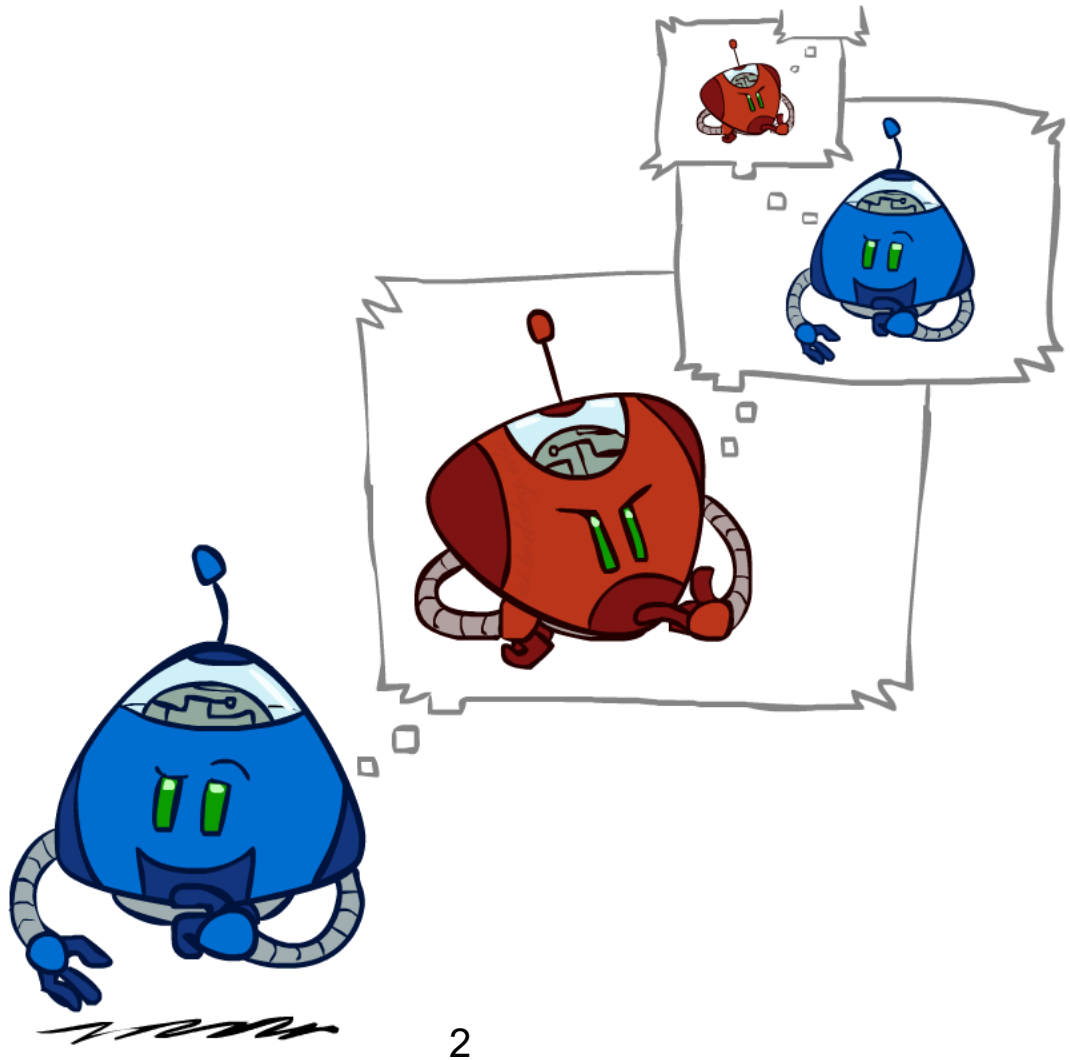


Prof. Sevinç İlhan Omurca / Dr. Öğretim Üyesi Fidan Kaya Gülağız
Kocaeli Üniversitesi

[Slides adapted from Dan Klein and Pieter Abbeel (ai.berkeley.edu).]

Bugün

- Arama
  - Rekabet Ortamında Arama
  - Oyunlarda Arama
- Oyun Teorisi
- Sıfır Toplamlı Oyunlar
- Minimaks Yöntemi
- Alfa-Beta Budaması
- Stokastik Oyunlar

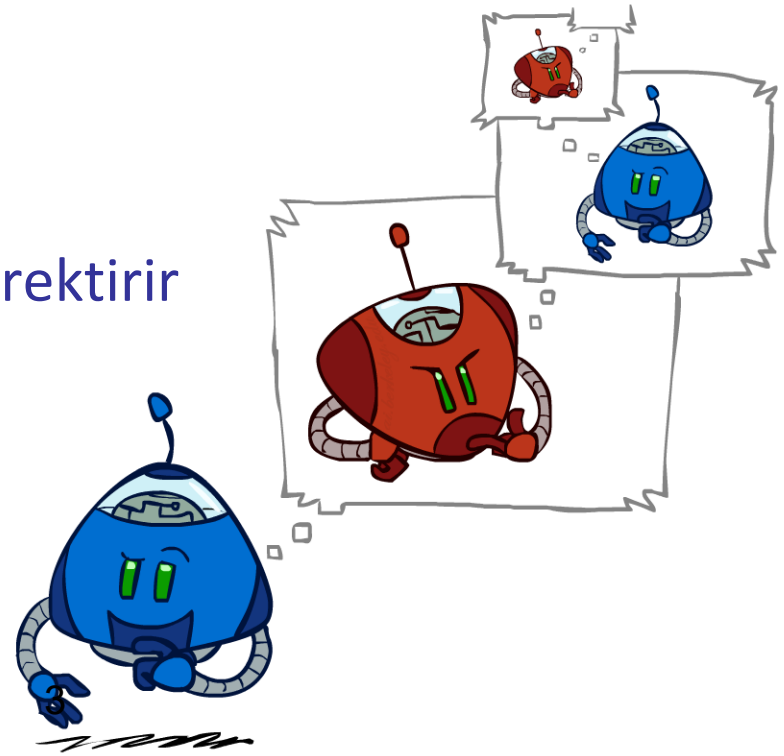


Rekabet Ortamında Arama

- Çok ajanlı ortamlarda davranış?
  - Bir ajan diğer ajanın eylemlerini ve
  - Bu eylemlerin onun refahını nasıl etkilediğini dikkate alır.

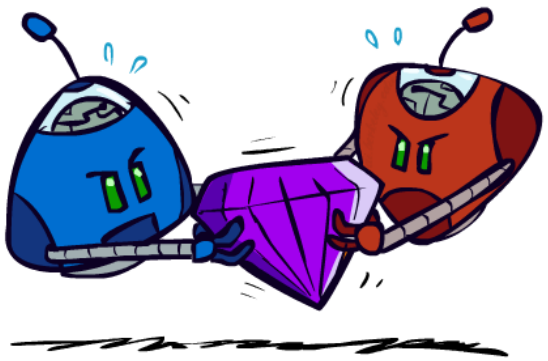
- Diğer ajanların öngörülemezliği?
  - Olası beklenmedik durumları

- Ajanların hedeflerinin çatıştığı ortamlar, rekabetçi arama gerektirir
  - İşte bu ortamlardaki problemler “oyun” olarak adlandırılır !



Oyunlarda Arama

- Oyunlar her zaman rekabetçi ortamlarda mı gerçekleşir?
  - Rekabetçi ya da iş birliğine dayalı ortamlar olabilir!
- Oyun teorisine göre oyun?
  - Herhangi bir çok ajanlı ortam (işbirliğine dayalı veya rekabetçi), her bir etmen diğeri üzerindeki etkisinin önemli olması koşuluyla bir oyundur.
- Bu derste rekabetçi ortamlardaki arama problemlerini ele alacağız.



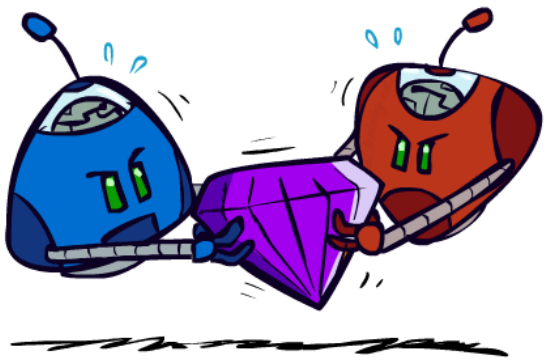
Oyun Teorisi

- Oyun Teorisi iki temel teoreme dayanır
  - Minimum-Maksimum Teoremi
  - Denge Teoremi (Nash Dengesi)

- Doğuş:
  - 1928 Jon von Neuman ispatladığı **Minimaks Teoremi**

- Gelişim:
  - 1950 John Forbes Nash ispatladığı **Denge Teoremi**

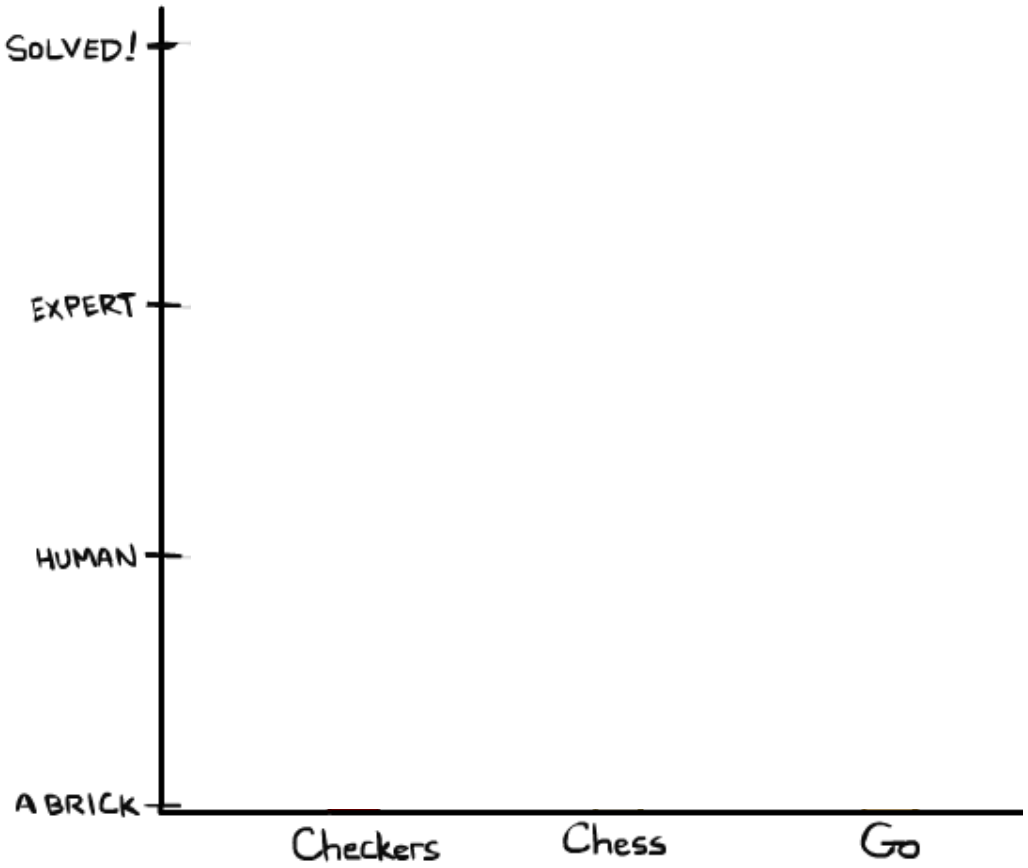
Analiz + Strateji + Taktik = Hamle



Oyun Oynama ve AI – Tarihsel Gelişim

- **Dama:**
  - 1950: İlk bilgisayar oyuncusu
  - 1994: İlk bilgisayar şampiyonu:
    - Chinook, şampiyon Marion Tinsley'nin 40 yıllık saltanatını sonlandırdı
  - 2007: Checkers çözüldü!
- **Satranç:**
  - 1997: Deep Blue, Gary Kasparov'u yendi.
  - Günümüzde AI temelli programlar eskiye göre daha iyidir.
- **Go:**
  - Dallanma faktörü b, b > 300
  - Klasik programlar pattern knowledge tabanlı teknikleri kullanıyordu.
  - Güncel programlar gelişmiş Monte Carlo (randomized) genişleme tekniğini kullanırlar.

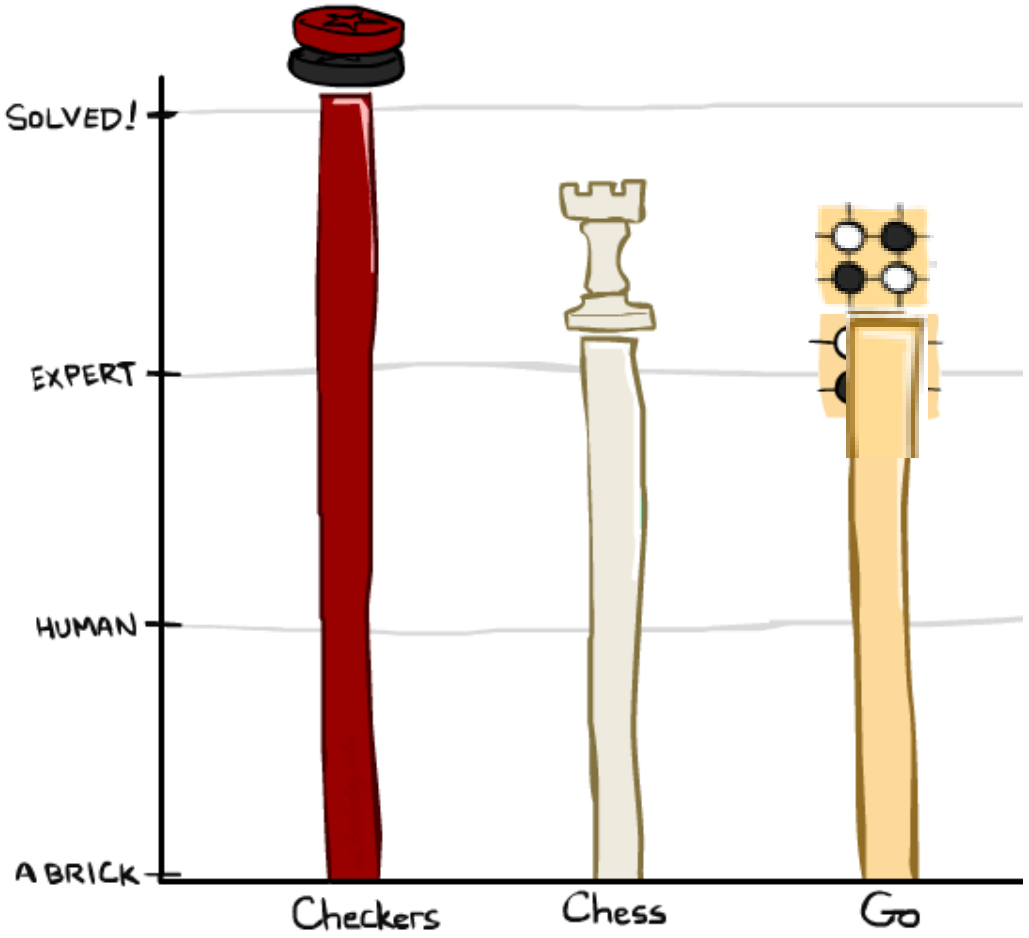
	Satranç	GO
Tahta Boyutu	8 X8	19 X 19
Her oyunda ortalama hareket sayısı	100	300
Her dönüşteki dallanma faktörü	35	235



Oyun Oynama ve AI – Tarihsel Gelişim

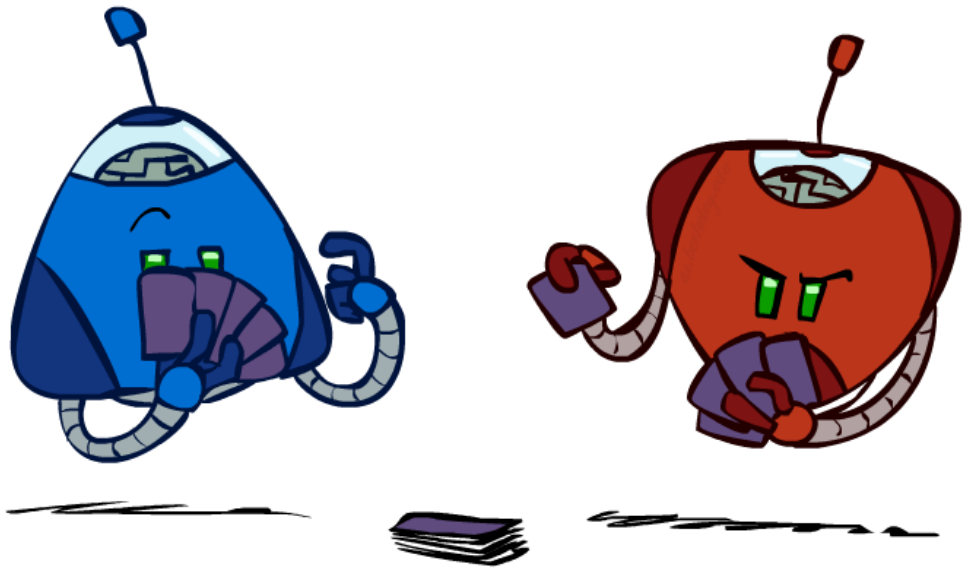
- **Dama:**
  - 1950: İlk bilgisayar oyuncusu
  - 1994: İlk bilgisayar şampiyonu:
    - Chinook, şampiyon Marion Tinsley'nin 40 yıllık saltanatını sonlandırdı
  - 2007: Checkers çözüldü!
- **Satranç:**
  - 1997: Deep Blue, Gary Kasparov'u yendi.
  - Günümüzde AI temelli programlar eskiye göre daha iyidir.
- **Go:**
  - 2016: Alpha GO , 9 X 9 Go oyununda insan şampiyonu yendi !
  - Monte Carlo Ağacı Araması tekniği kullanıldı.

	Satranç	GO
Tahta Boyutu	8 X8	19 X 19
Her oyunda ortalama hareket sayısı	100	300
Her dönüşteki dallanma faktörü	35	235



Oyun Türleri

- Yapay Zekada oyunlar 3 sınıf altında incelenir



- Oyunlar:
  - Rastgele Sonuçlu
    - İskambil, tavla,…
  - Ustalık Gerektiren
    - Futbol, Basketbol, Golf, …
  - Stratejik
    - Satranç, Dama, Go, …

- Stratejik oyunlar daha çok ilgi çekmektedir.
- Oyunlarda oyunculardan mümkün olan durumlardan en iyisini seçmesi beklenmektedir.

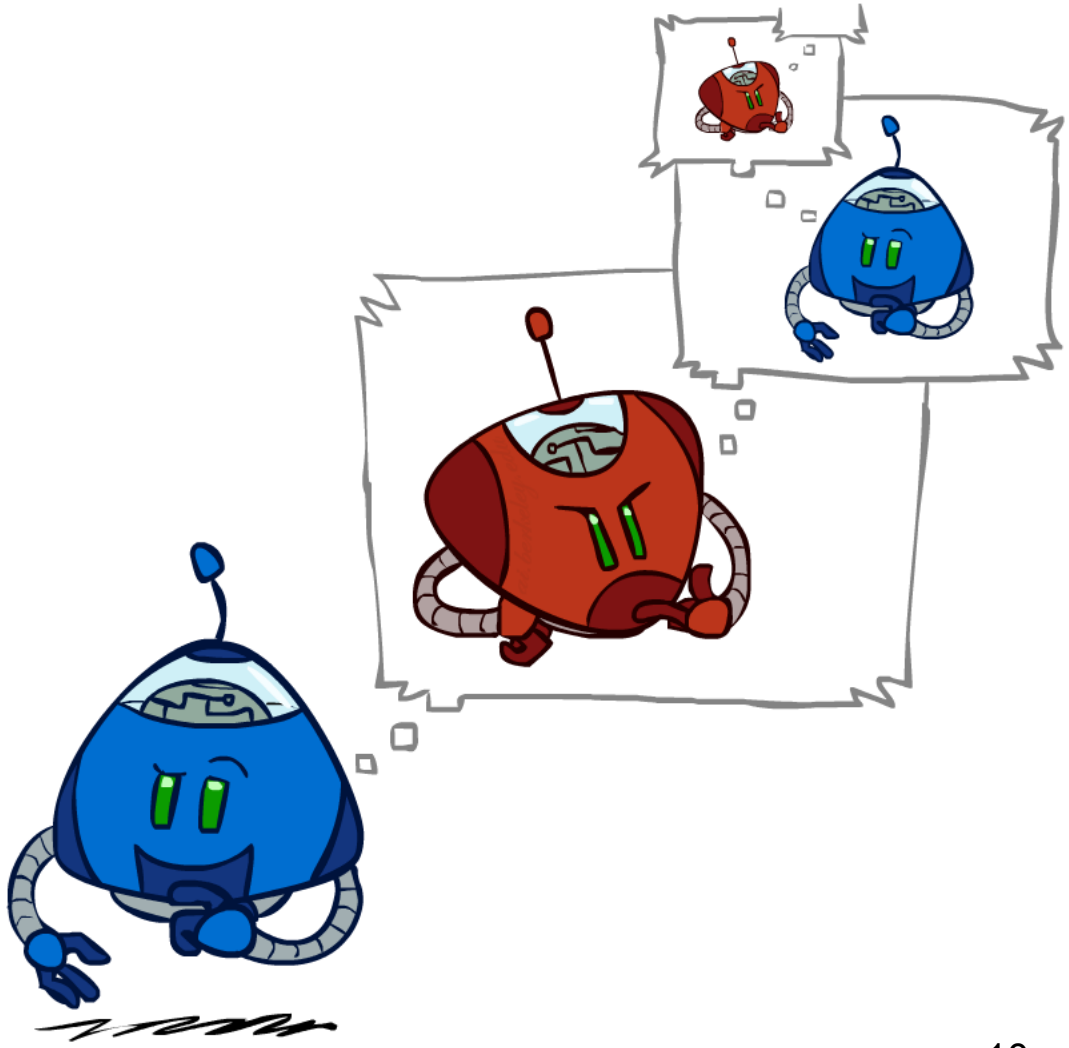
Oyun Türleri

- Oyunları farklı açılardan sınıflandırmak da mümkün!
- Örneğin:
  - Deterministik ya da stokastik?
  - Bir, iki, ya da daha çok oyunuculu?
  - Sıfır Toplamlı?
  - Mükemmel bilgili ya da değil ?

- Oyunlar & AI
  - Oyunların farklı türleri için bir durumdan diğer durma geçiş için önerilerde bulunacak bir politika/strateji geliştirecek algoritmalar istiyoruz.

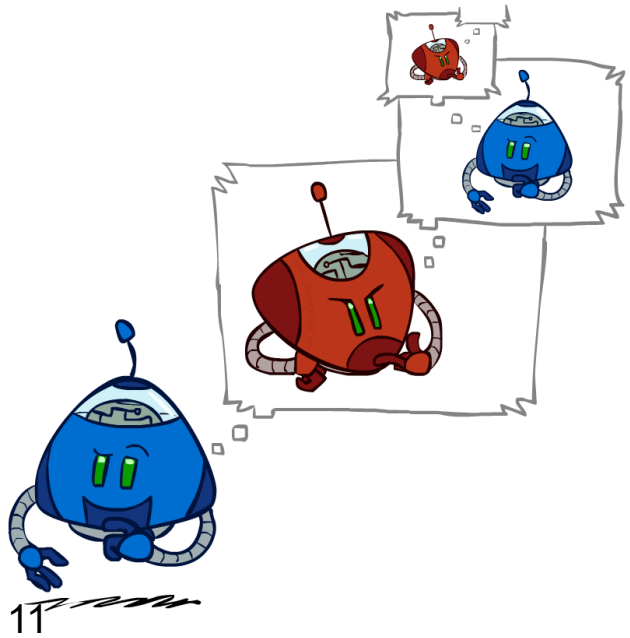
	Deterministik	**Olasılıklı**
Tam Bilgili	Satranç, Go, Dama	Tavla, Monopoly
Eksik Bilgili	Battleships, Stratego	Poker, Scrabble, Briç

Sıfır Toplamlı Oyunlar



Sıfır Toplamlı Oyunlar

- Stratejik oyunların çözümünde oyun teorisi kullanılır.
- Bu teorinin yardımıyla taraflar kendi kazançlarını optimize etmek amacı ile bilgi edinmektedir.
- İki kişilik stratejik oyunlar
  - Taraflardan biri bilgisayar olarak düşünülebilir.
  - Bir tarafın kazanması diğer tarafın kaybetmesine eşittir.
  - Bu oyunlara **sıfır toplamlı iki taraflı oyunlar** denir.

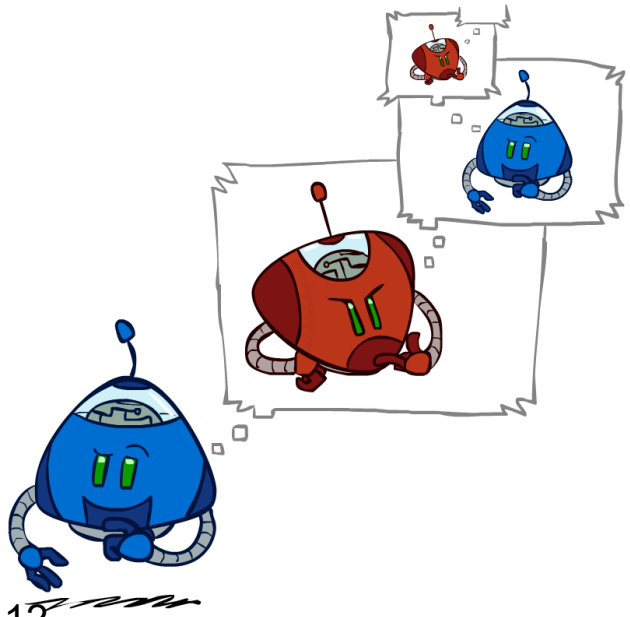


Sıfır Toplamlı Oyunlar

- Sıfır toplamlı oyunları anlamak için ***kazanç matrisinin*** oluşturulması gerekir.

- Kazanç matrisi oluşturma ?
  - Önce her oyuncu için olası gidişler yazılır.
  - Birinci oyuncu m adet gidişe sahip olsun. A1,…Am
  - İkinci oyuncu n adet gidişe sahip olsun B1,…Bn
  - Bu stratejilere karşılık gelen durumlar aij olsun.
  - m x n oyun için kazanç matrisi:

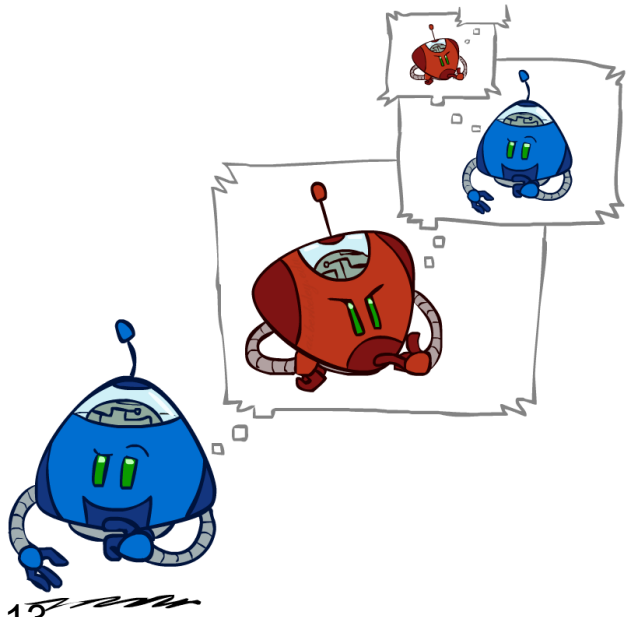
A\B	B_1	B_2	…	B_n
A_1	a_{11}	a_{12}	…	a_{1n}
A_2	a_{21}	a_{22}	…	a_{2n}
…	…	…	…	…
A_m	a_{m1}	a_{m2}	…	a_{mn}



Kazanç Matrisi Örnek: Kartların Açılması

- Kartların Açılması Oyunu:
  - Ahmetin üzerinde 1 yazılmış beyaz ve siyah kartı var.
  - Hasanın üzerine 1 yazılmış siyah ve 0 yazılmış beyaz kartı var.
- Oyuncular kartların renklerini ve sayılarını görmeden aynı zamanda 1 tanesini açar.
  - **Renkler aynı ise Ahmet sayıların farkına eşit değerini kazanır**.
  - **Renkler farklı ise Hasan bu değeri kazanır**.
- Kazanç matrisi:

A\H	H_S	H_B
A_S	0	-1
A_B	0	1

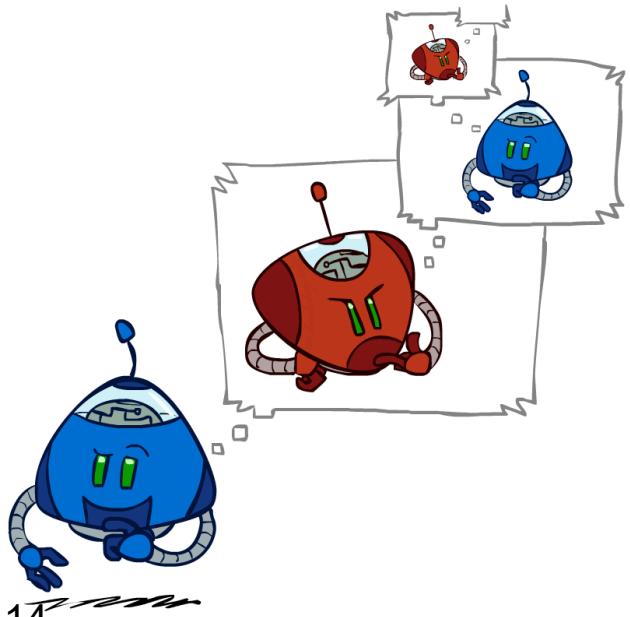


Kazanç Matrisi Örnek: Para Atma Oyunu

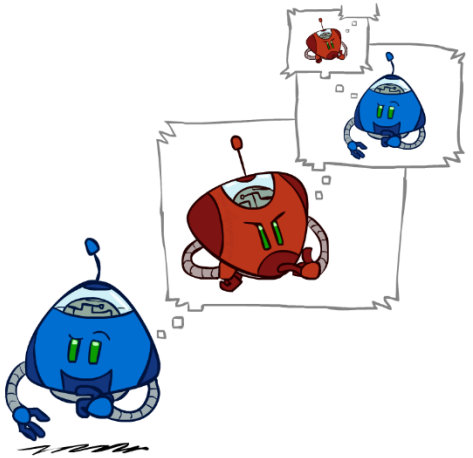
- Para Atma Oyunu:
  - Ali ve Veli'nin aynı değerli iki madeni parası vardır.
  - Paraları masanın üzerine atarlar ve üzerini kapatırlar.
  - Paraların yüzeyleri aynı ise Ali, farklı ise Veli kazanır.

- Kazanç matrisi:

A\V	V_T	V_Y
A_T	2	-2
A_Y	-2	2



Kazanç Matrisi



- Kesin belirlenmiş oyun:
  - Her oyuncunun kısıtlı gidişi olan oyunlardır.
  - Kazanç matrisine bakılarak önceden kimin kazanacağı söylenebilir.

- Kazanç matrisinin minimaks noktası:
  - Aynı anda satırında minimum olan eleman,

    Bu satırla kesişen sütun için maksimum degere sahip olur.

A\H	H_S	H_B
A_S	0	-1
A_B	0	1

- **Kesin belirlenmiş oyun = Kazanç matrisinde minimax noktası olan oyun**
- **Minimaks 0 olan oyun = Dürüst Oyun**

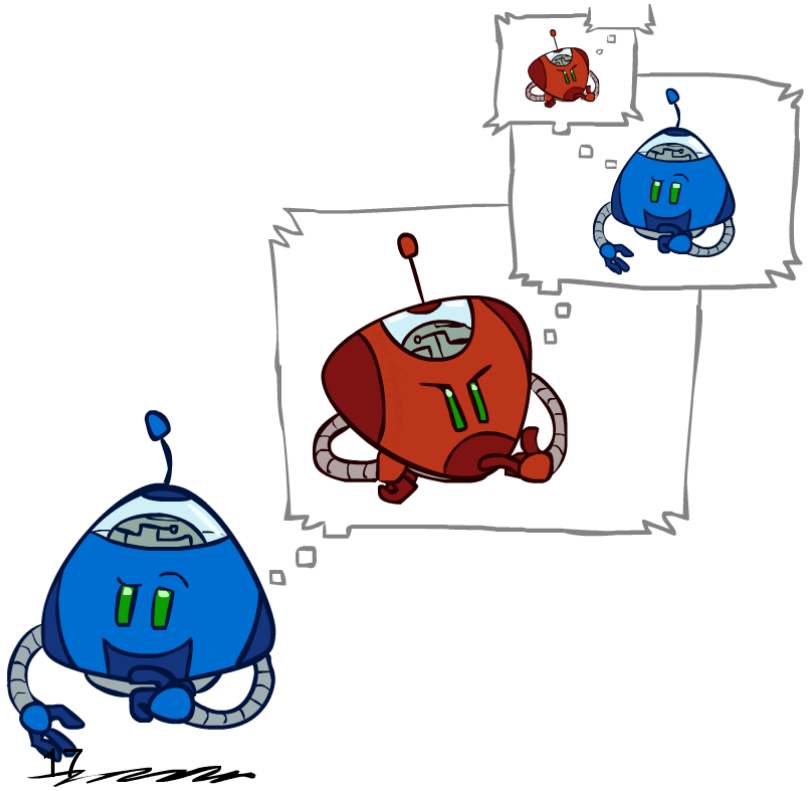
Kazanç Matrisi: Minimaks Mantığı

- Mantık:
  - A oyuncusu 1. stratejiyi (A_1) seçerse B 'de A'ya en küçük kazanç sağlayacak B_{1j} stratejisini seçer.
- Böylece her stratejinin garantilenmiş bir alt değeri olur.
  - $\alpha_l = \min \alpha_{lj}$ (Tüm j değerleri için)
- Rakip bizim için her zaman en kötü stratejiyi seçeceğinden elde edilen α_l değerleri arasında en büyük olan en iyi stratejimiz olacaktır
  - $\alpha = \max \alpha_l$ (Tüm i değerleri için)
- Sonuçta α değeri garantilenmiş kazancı sağlayan en küçük değerdir.
  - $\alpha = \max_i \min_j \alpha_{lj}$
- Benzer şekilde bizim için karlı olan durum rakip için kötüdür ve rakibin garantilenmiş kazanç değeri:
  - $\beta = \min_j \beta_j = \min_j \max_i \alpha_{lj}$

Kazanç Matrisi: Minimaks Mantığı

- α :
  - Bizim en iyi stratejimizde garantilenmiş minimum kazanç
- β :
  - Rakibimizin en iyi oyununda elde edeceğimiz maksimum kazancı ifade eder.
- Kazanç Matrisi:

A\B	B_1	B_2	...	B_n	α_i
A_1	a_{11}	a_{12}	...	a_{1n}	α_1
A_2	a_{21}	a_{22}	...	a_{2n}	α_2
...	...	...	...	...	...
A_m	a_{m1}	a_{m2}	...	a_{mn}	α_m
β_j	β_1	β_2	...	β_n	

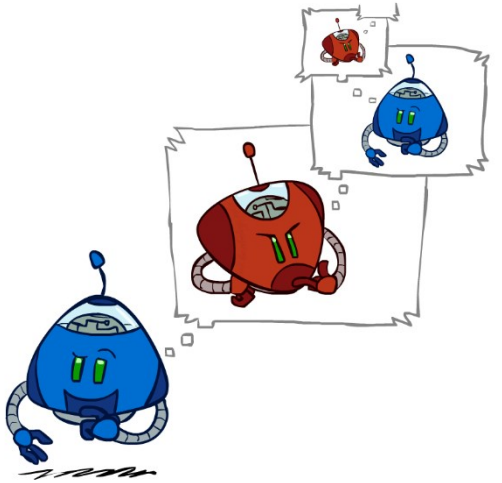


Kazanç Matrisi: Minimaks Mantığı

- Örnek:

A\B	B_1	B_2	B_3	α_i
A_1	0.9	0.4	0.2	0.2
A_2	0.3	0.6	0.8	**0.3**
A_3	0.5	0.7	0.2	0.2
β_j	0.9	**0.7**	0.8	

- Alt değer α : 0.3 ve üst değer β :0.7 olur.
- Minimax stratejilerinin kararlı olduğu oyunlarda alt α ve üst β değeri birbirine eşit olmaktadır.
  - α=β ise bu değerlerin ortak değerine “oyunun değeri” denir. Ve “v” ile temsil edilir.
- Ortalama kazanç oyunun değerine eşittir.
- Taraflar bu optimal stratejiye bağlı kalmadığında kaybedebilir.
  - Von Neumann- Oyun teorisi Ana Teoreminin Temeli
  - Teoreme göre her sonlu oyunun bir bir değeri vardır:
    - $\alpha \leq v \leq \beta$

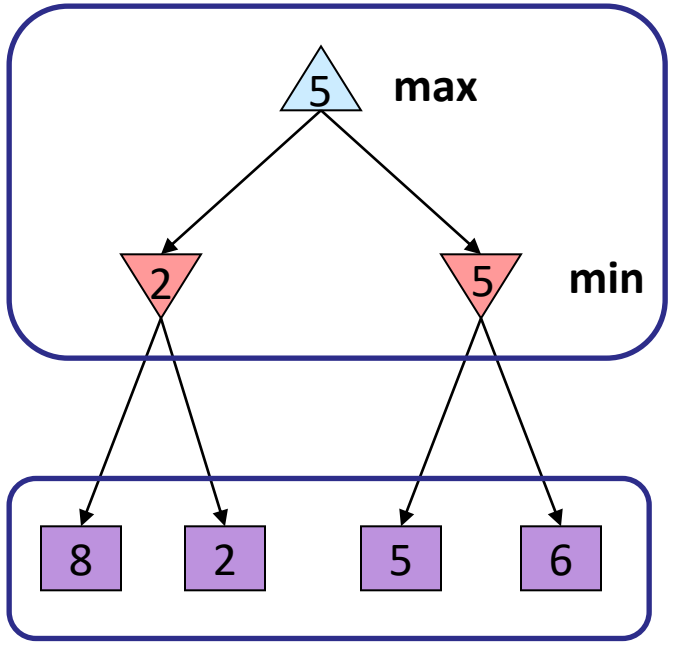


Rekabet Ortamında Arama (Minimaks)

- Minimaks, deterministik oyunlar için mükemmeldir.
- Deterministik Sıfır Toplamlı Oyunlar:
  - Tic-tac-toe, Satranç, Dama
  - Bir oyuncu sonucu maksimize eder!
  - Diğer oyuncu sonucu minimize eder!

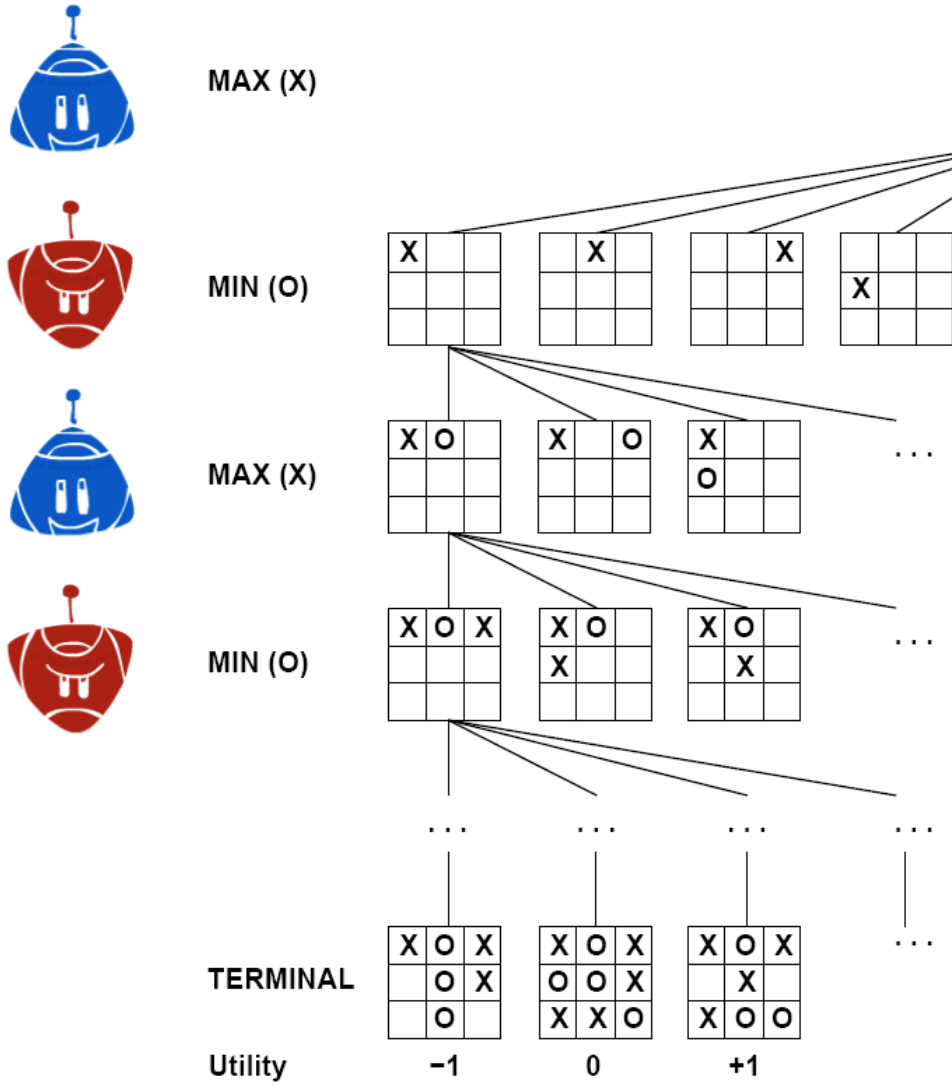
- Minimaks Arama:
  - Durum uzayı bir ağaç üzerinden temsil edilir.
  - Oyuncular sırayla oynarlar.
  - Her düğümün minimaks değeri hesaplanır.

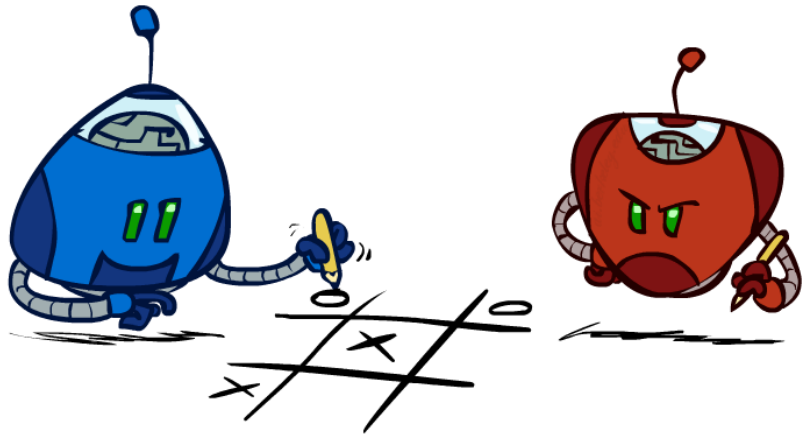
Rekürsif olarak hesaplanmış minimaks değerleri



Oyunun parçası olan terminal değerler

Tic-Tac-Toe Oyun Ağacı





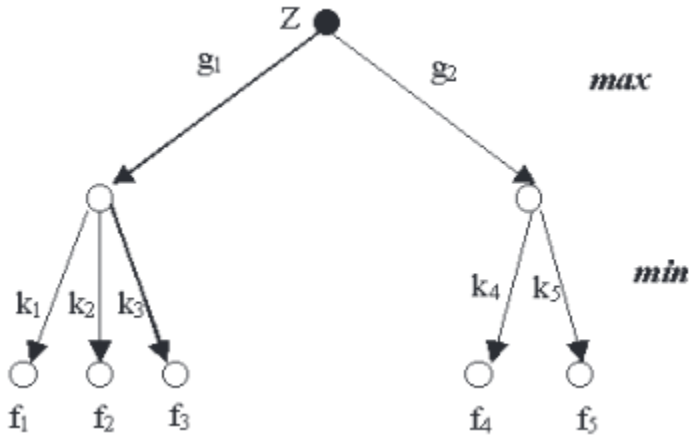
Minimaks Yöntemi

- **Ana fikir:**
  - En yüksek minimaks değerine sahip pozisyona hareket et
- Adımlar:
  - Oyun ağacı belirli bir derinliğe kadar araştırılır
  - Her hedef ara durumu için özel sezgisel fonksiyon değerleri hesaplanır.
  - Bu düğümlerin değerlerinden faydalanarak köke doğru hareketlenilir ve ağacın düğümlerinin değerleri kesinleştirilir.
  - Son olarak program bu değerlere göre en iyi hamleyi yapar.
- Araştırma derinliği arttıkça kararlar daha çok akıllılık gösterecektir.

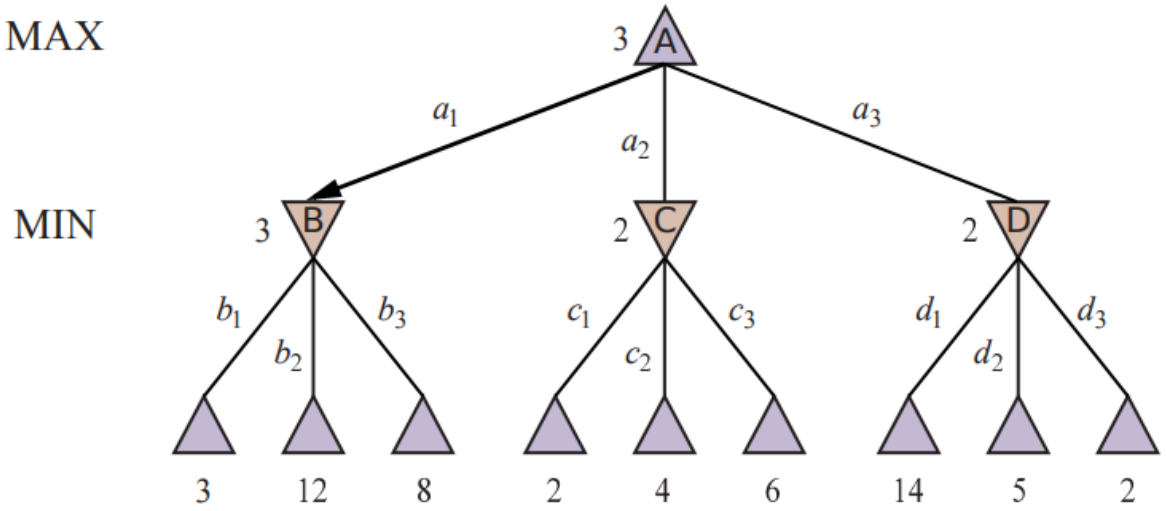
Minimaks Yöntemi

- Yöntemde oyun ağacı belirli bir derinliğe kadar genişletilir
  - Uç düğümlerin değerlendirilmesi yapılır.
- **Biz sezgisel fonksiyon değerini maksimum yapmaya çalışırken**
  - **Rakip bizim sezgisel fonksiyon değerimizi minimum yapmaya çalışır.**
- Biz:
  - Uygun seviye içinden en iyi gidişimizi uç durumların maksimumunu değerlendirerek yapacağız.
- Fonksiyon değeri büyük olduğu sürece galibiyet şansı yüksek, değer küçük olduğu sürece rakibin şansı yüksek.
- **Oyunculardan biri sürekli yüksek (MAX) diğeri küçük (MIN) değerleri takip eder.**

Minimaks Yöntemi : 2 Seviyeli Oyun Ağacı

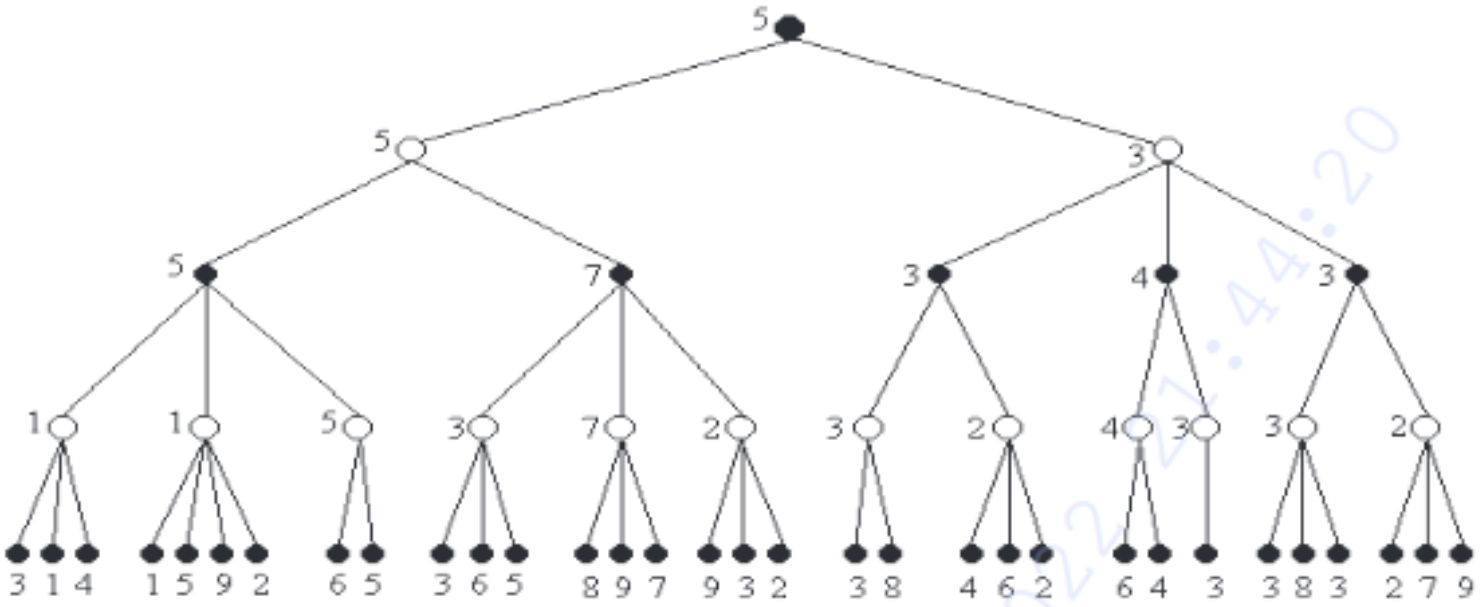


Z = max { min (f1, f2, f3), min (f4, f5) }



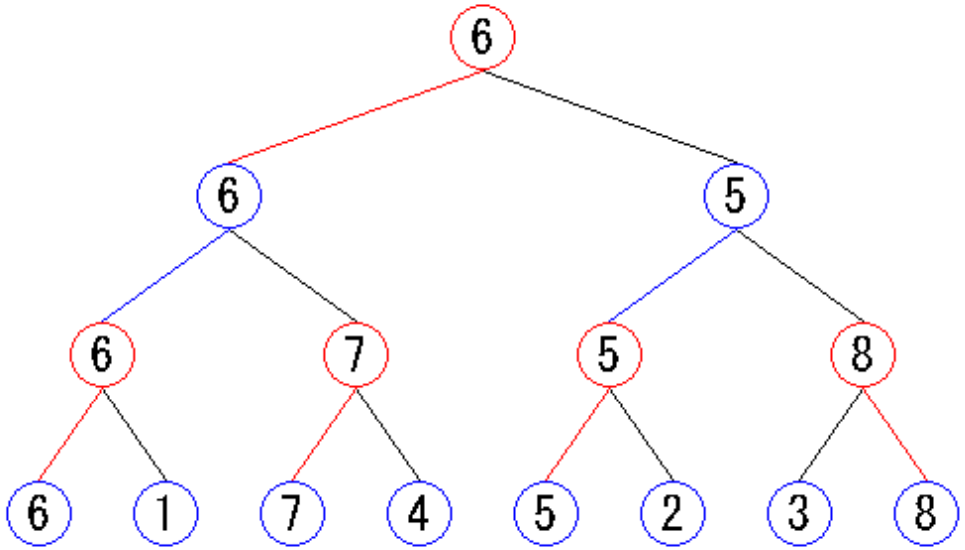
Minimaks Yöntemi : 4 Seviyeli Ağaç

- MAX gidişler VEYA düğümlerine
- MIN ise VE düğümlerine uygun gelmektedir.

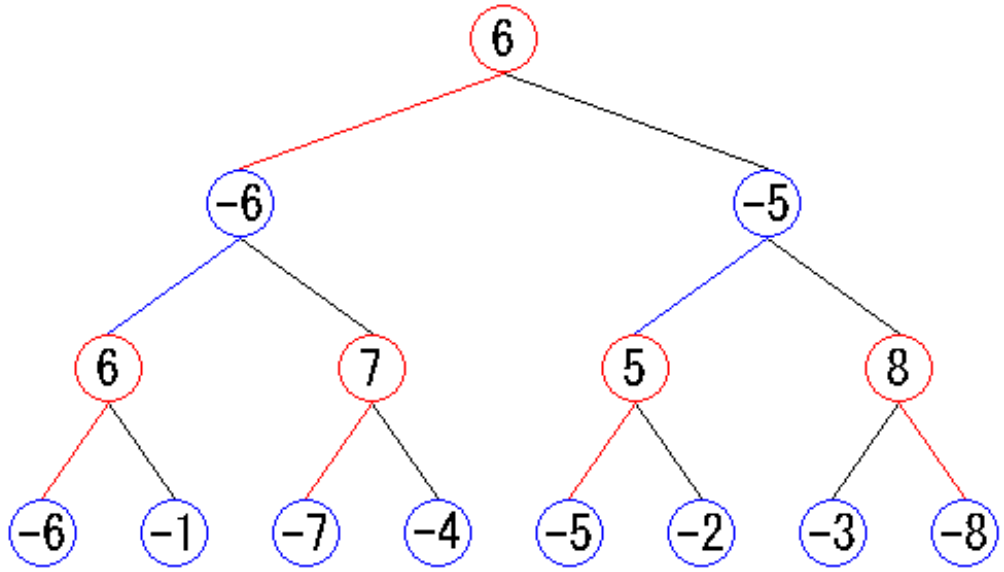


NegaMaks

- Her seviye için minimum puanli düğümün seçilmesi

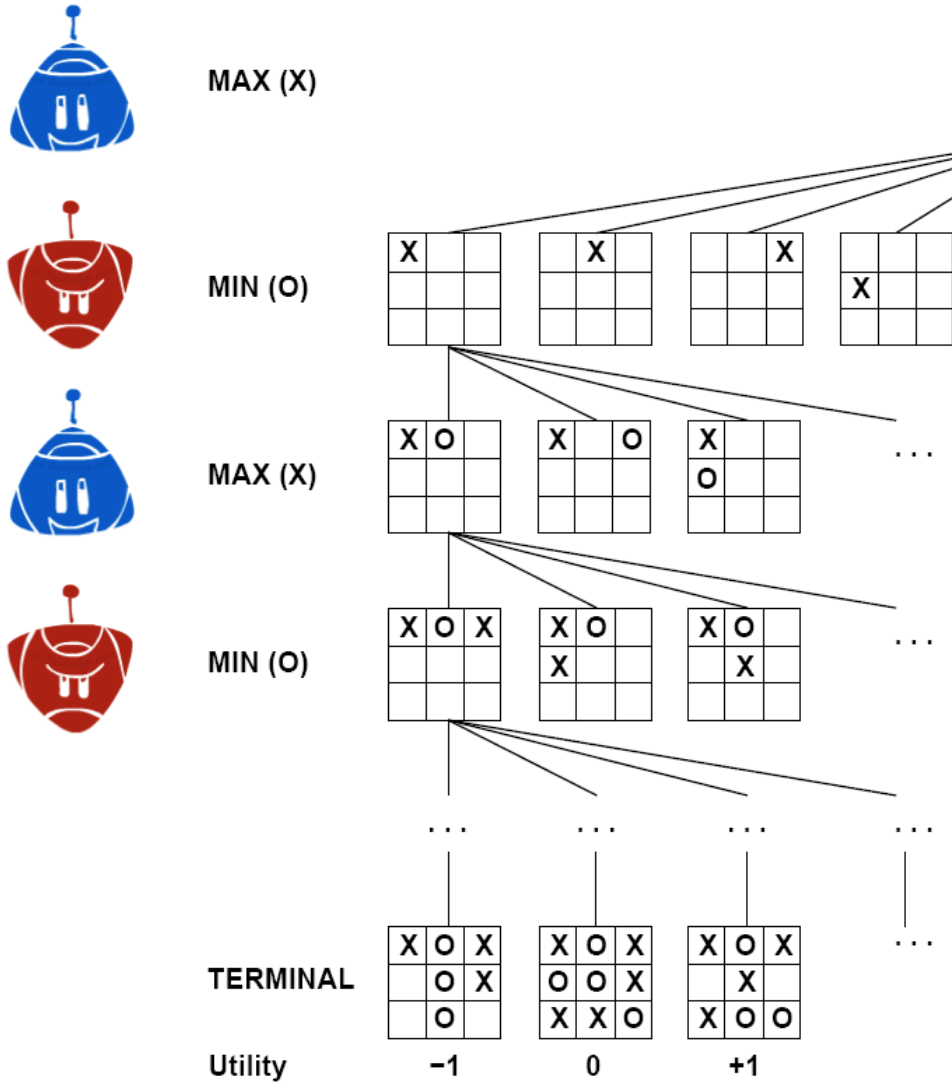


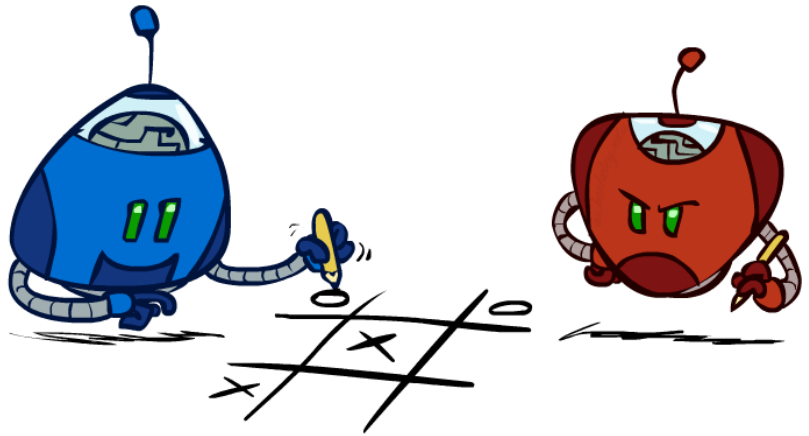
Minimaks



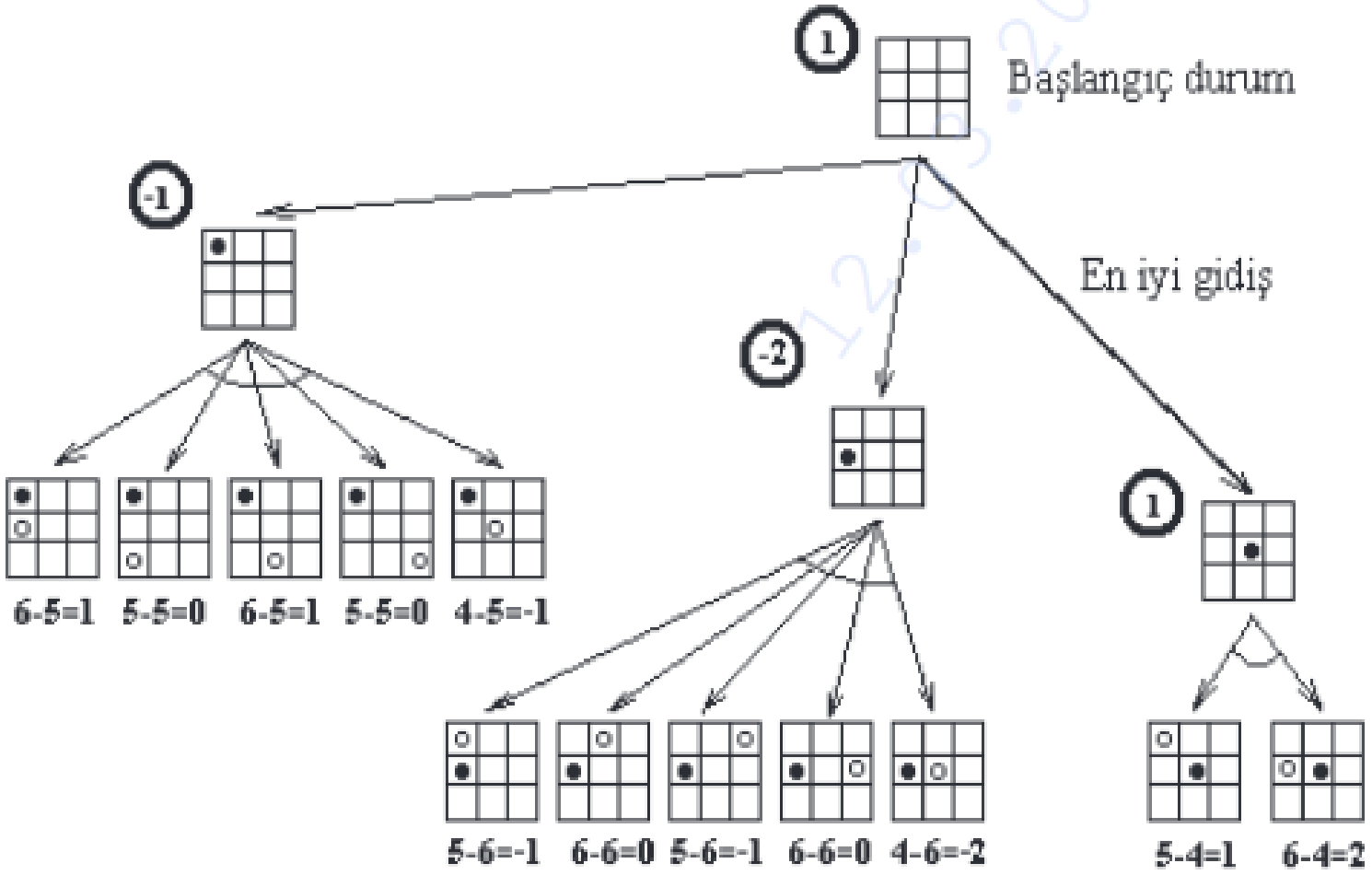
Negamaks

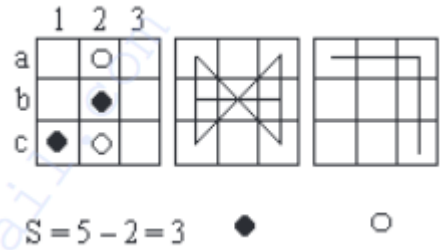
Tic-Tac-Toe Oyun Ağacı

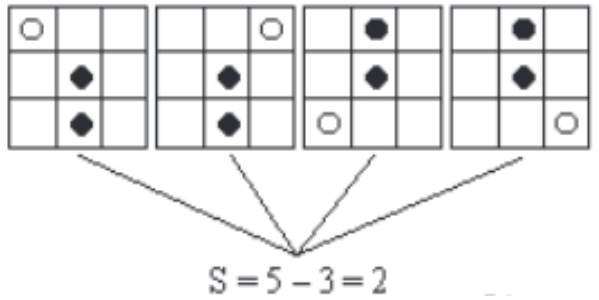


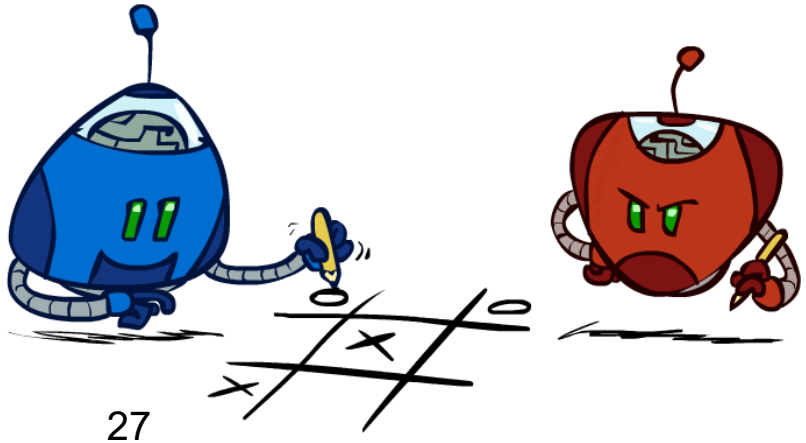


Minimaks Tic-Tac-Toe

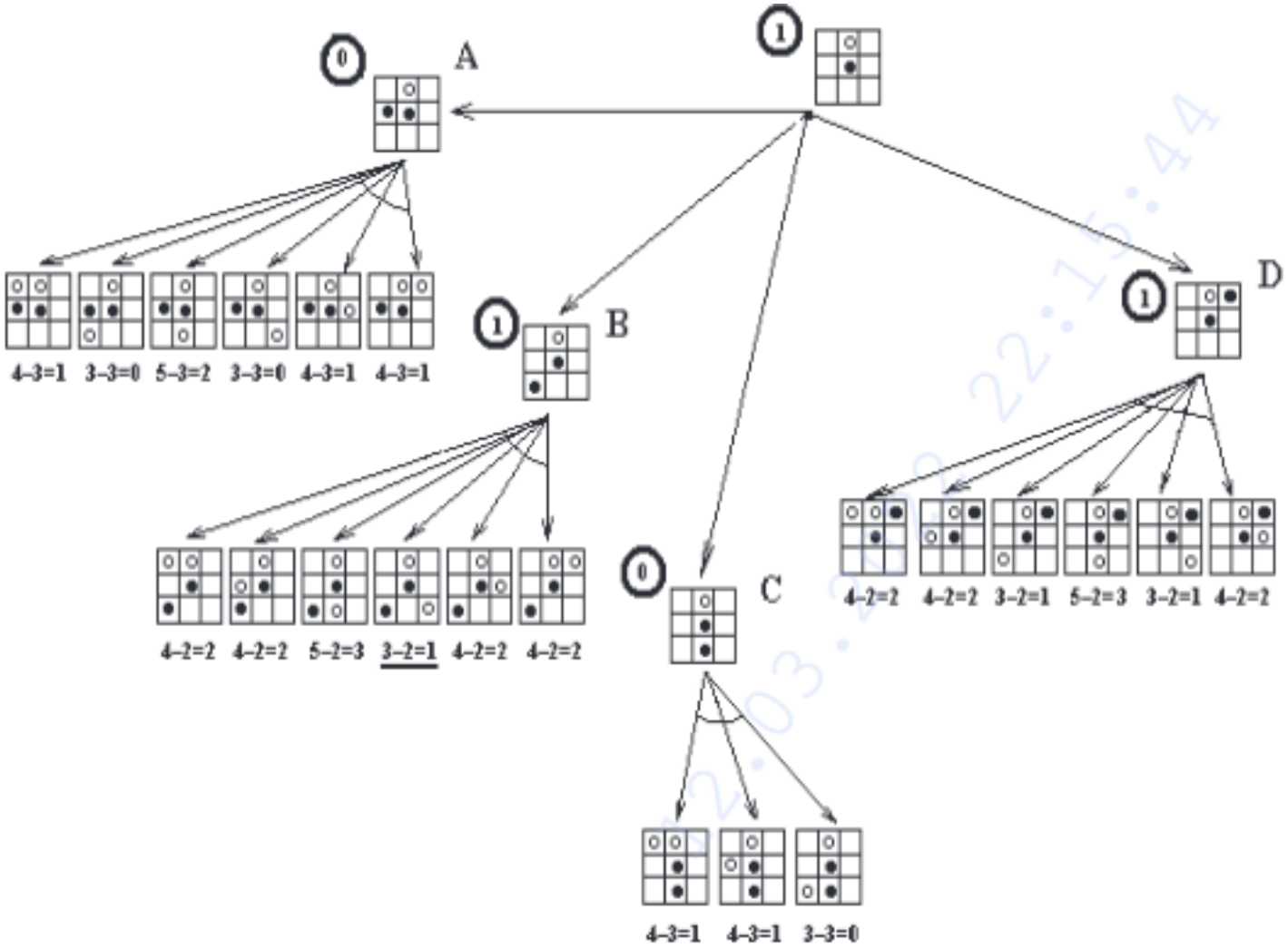


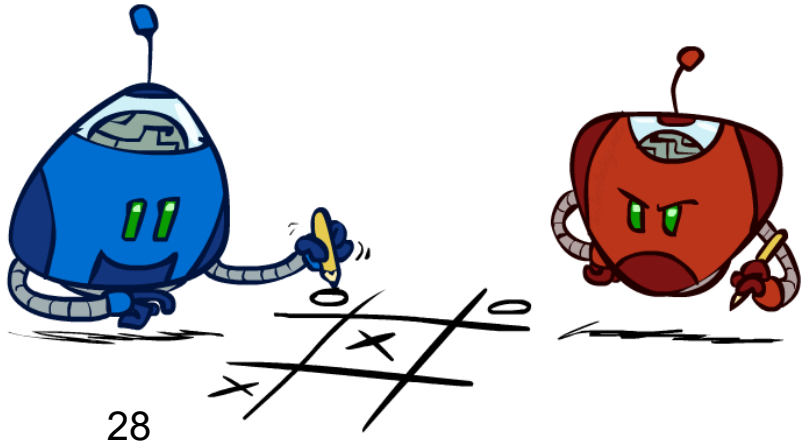






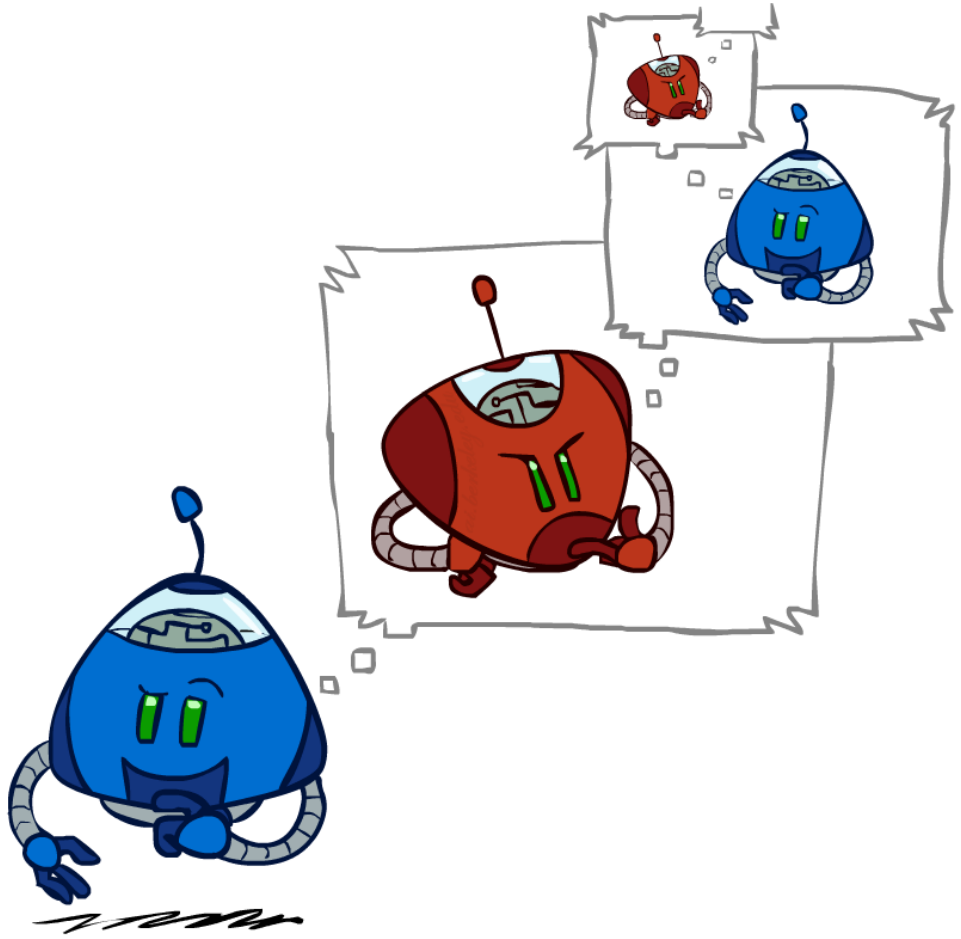
Minimaks Tic-Tac-Toe





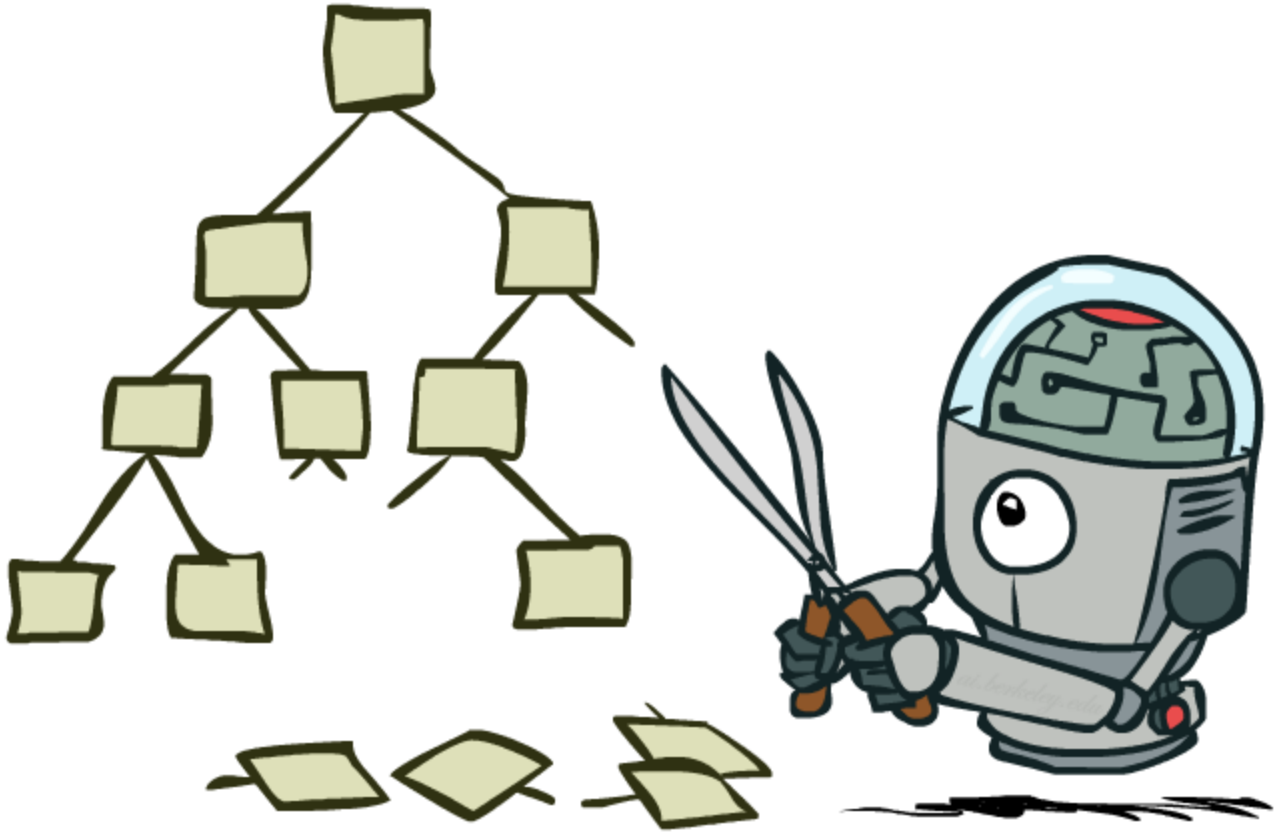
Minimaks Verimliliği

- Minimaks ne kadar verimlidir?
  - Tıpkı DFS gibi:
  - Zaman: $O(b^m)$
  - Yer: O(bm)
- Örnek: Satranç için, $b \approx 35$, $m \approx 100$
  - Arama süresi $b^m \approx 35^{100} \approx 10^{154}$
- Evren
  - Atomların sayısı $\approx 10^{78}$
  - Yaşı $\approx 10^{18}$ saniye
  - Saniyede işlenecek durum sayısı: 10^8 moves/sec
  - $10^8 \times 10^{78} \times 10^{18} = 10^{104}$
  - **Kesin çözüm tamamen olanıksızdır!**
  - **Tüm ağacı keşfetmemiz gerekiyor mu?**



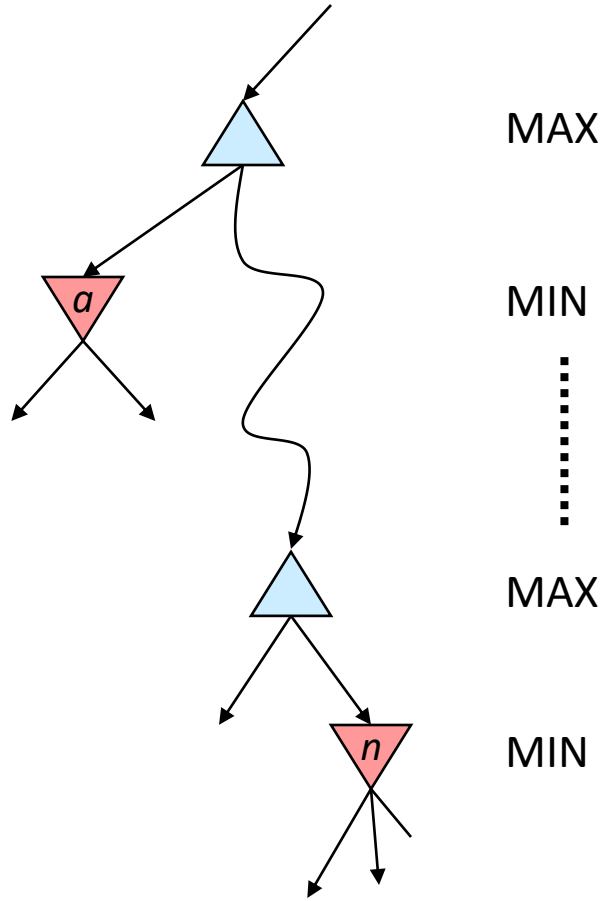
Kaynak Sınırları

Oyun Ağacı Budama



Alpha-Beta Budaması

- **Amaç:** Kötü olmayan gidişin bulunması.
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.
- **MIN açısından:**
  - Beta: onun için garantilenmiş değerler arasından en kötüsüdür.
  - Bu şekilde aranan fonksiyon değeri (alfa,beta) aralığında olur.
- Herhangi bir durum değeri (alfa,beta) aralığı dışında olursa:
  - Araştırılan durumun önem taşımadığı söylenir.
  - Ağacın belirli bir bölümü değerlendirilmez
- MAX açısından da durum simetriktir.



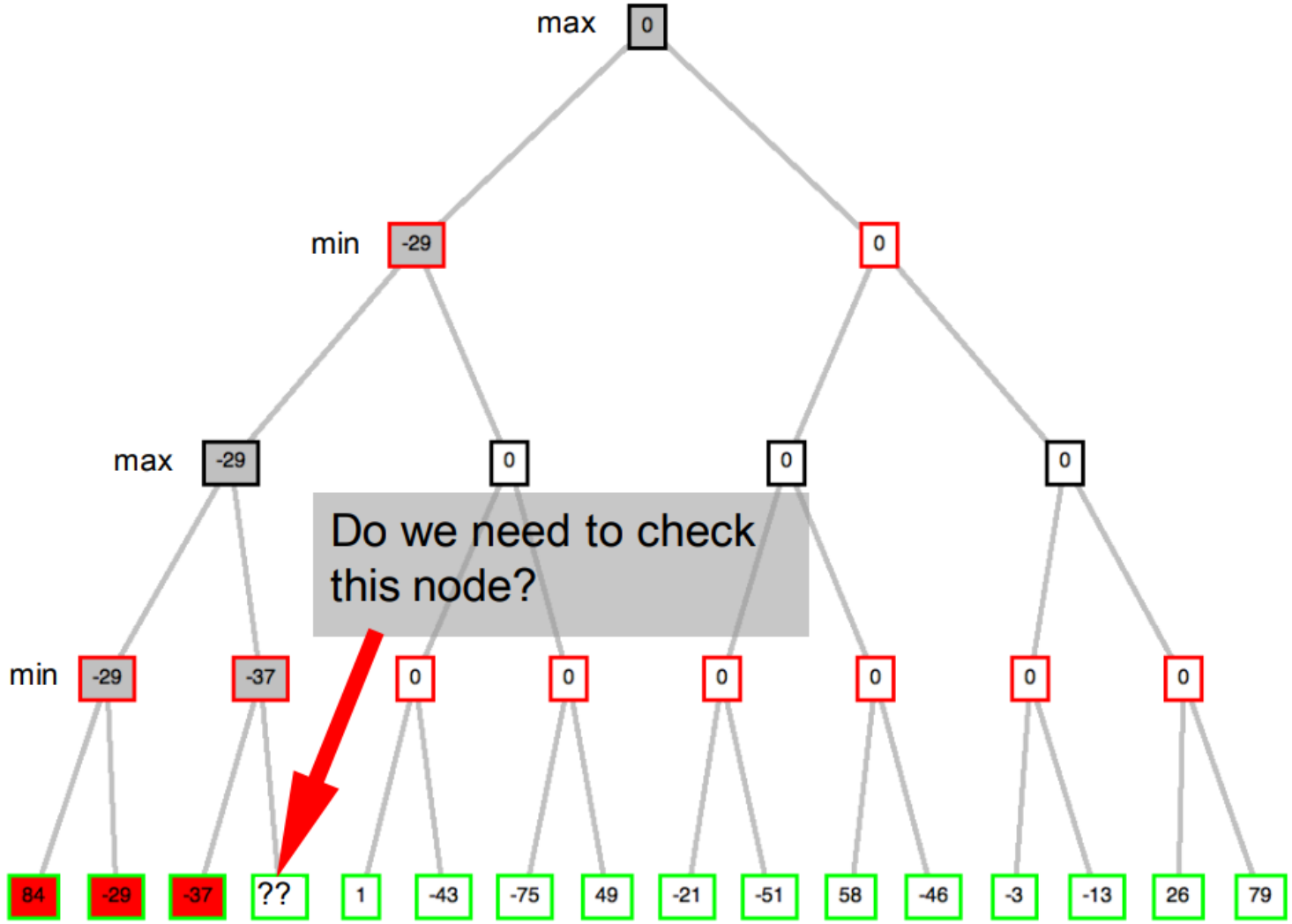
Alpha-Beta Budaması

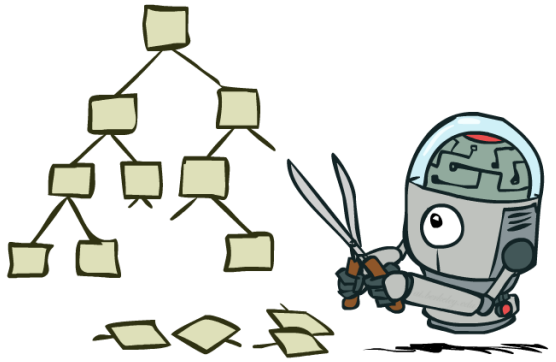
α: Kök yolundaki MAX'ın en iyi seçeneği
β: Kök yolundaki Min'in en iyi seçeneği

```
def max-value(state, α, β):
    initialize v = -∞
    for each successor of state:
        v = max(v, value(successor, α, β))
        if v ≥ β return v
        α = max(α, v)
    return v
```

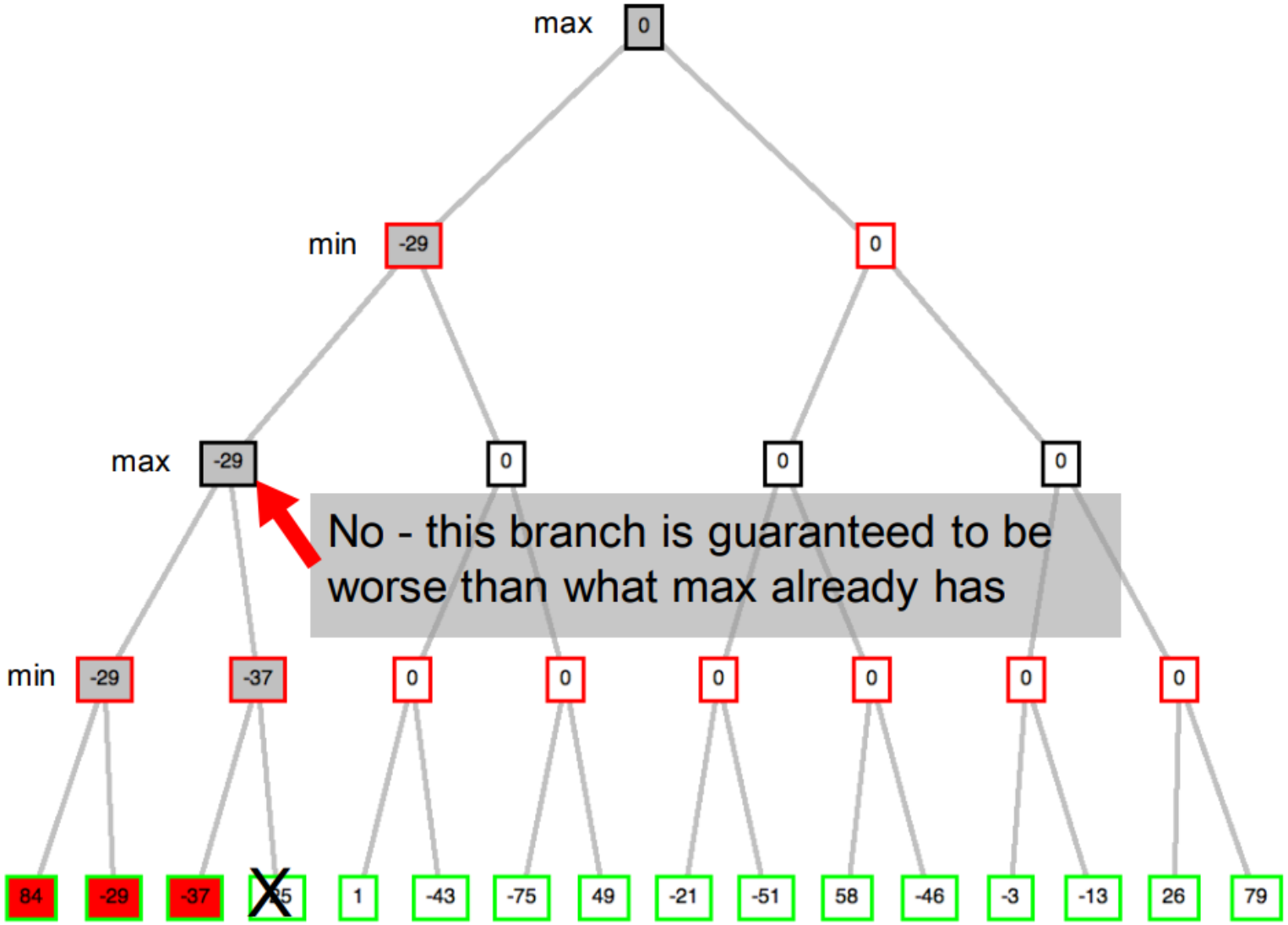
```
def min-value(state, α, β):
    initialize v = +∞
    for each successor of state:
        v = min(v, value(successor, α, β))
        if v ≤ α return v
        β = min(β, v)
    return v
```

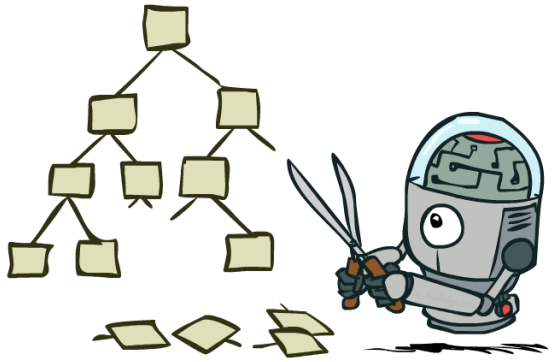
Alpha-Beta Budaması Örnek



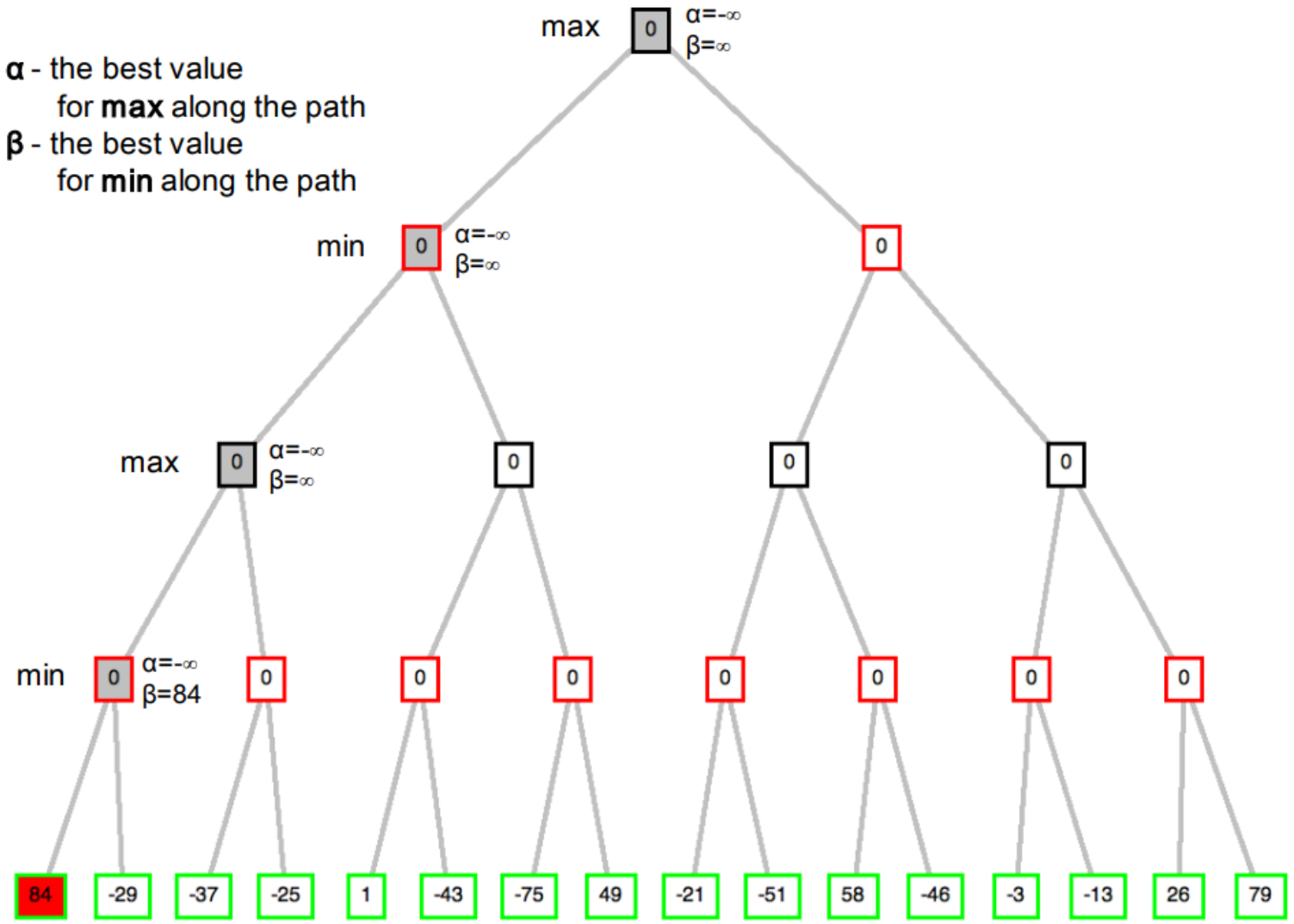


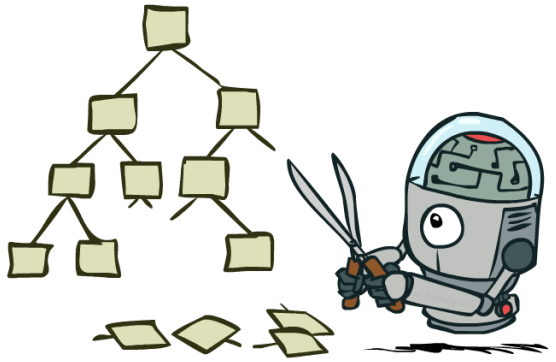
Alpha-Beta Budaması Örnek



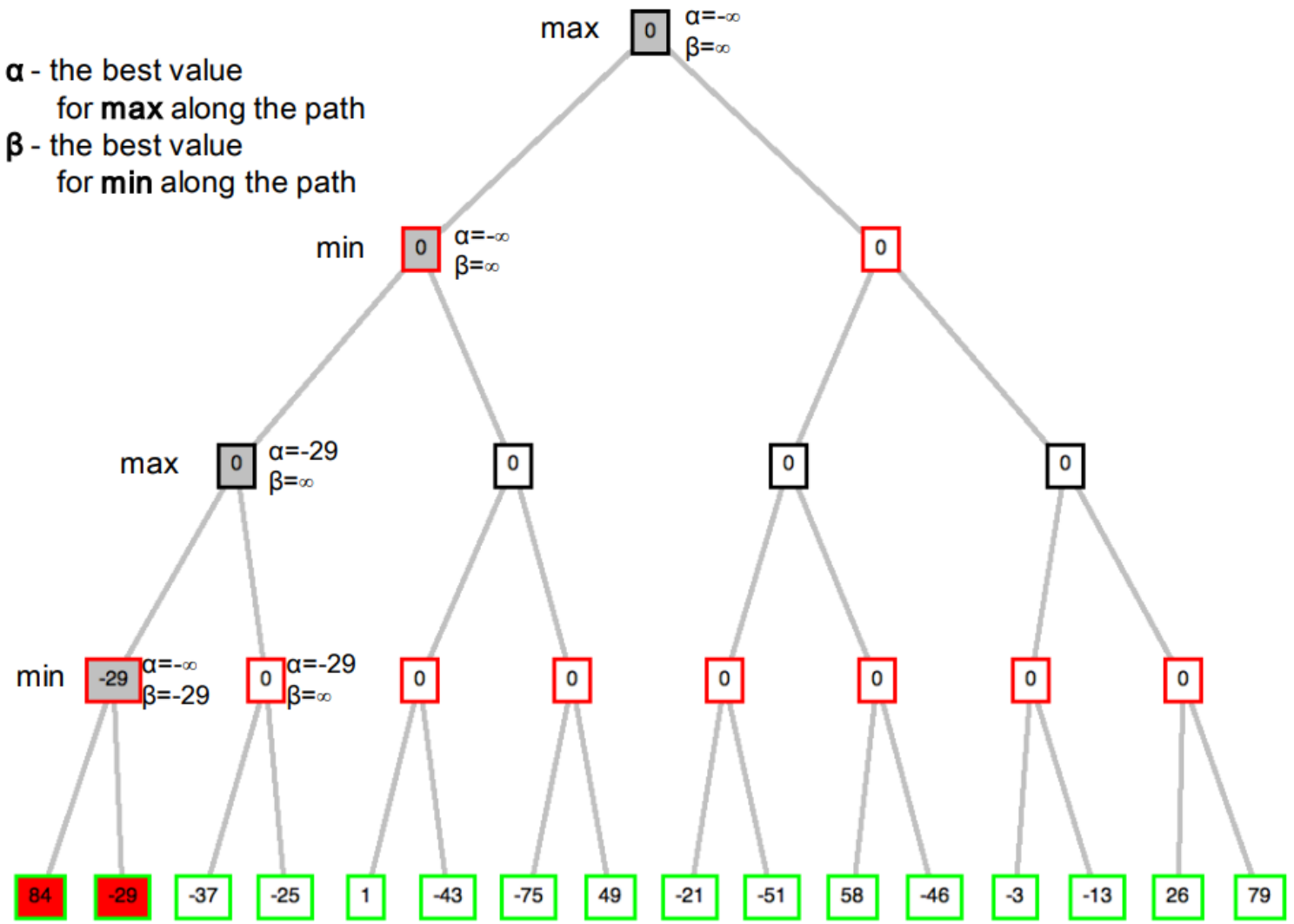


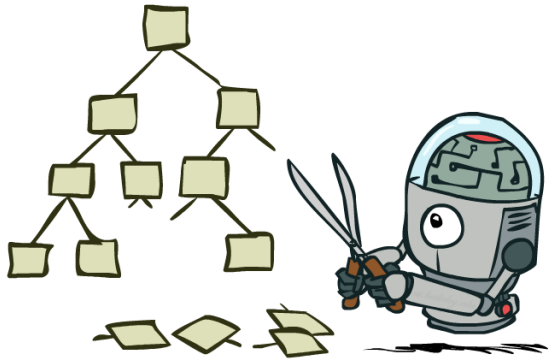
Alpha-Beta Budaması Örnek



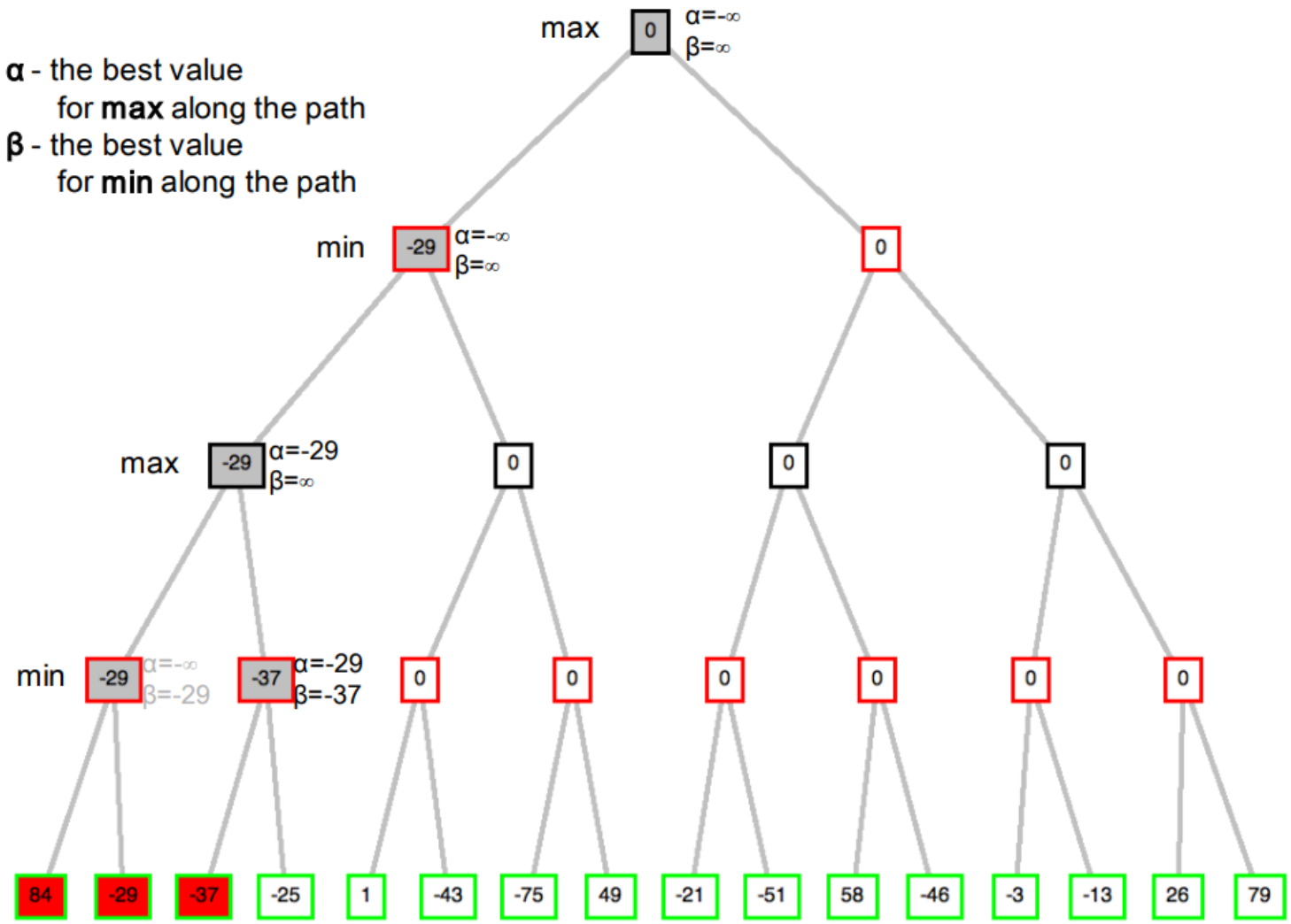


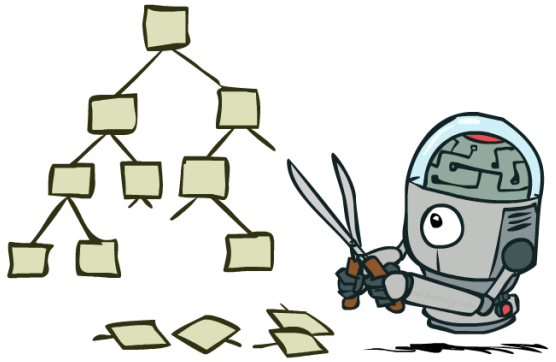
Alpha-Beta Budaması Örnek



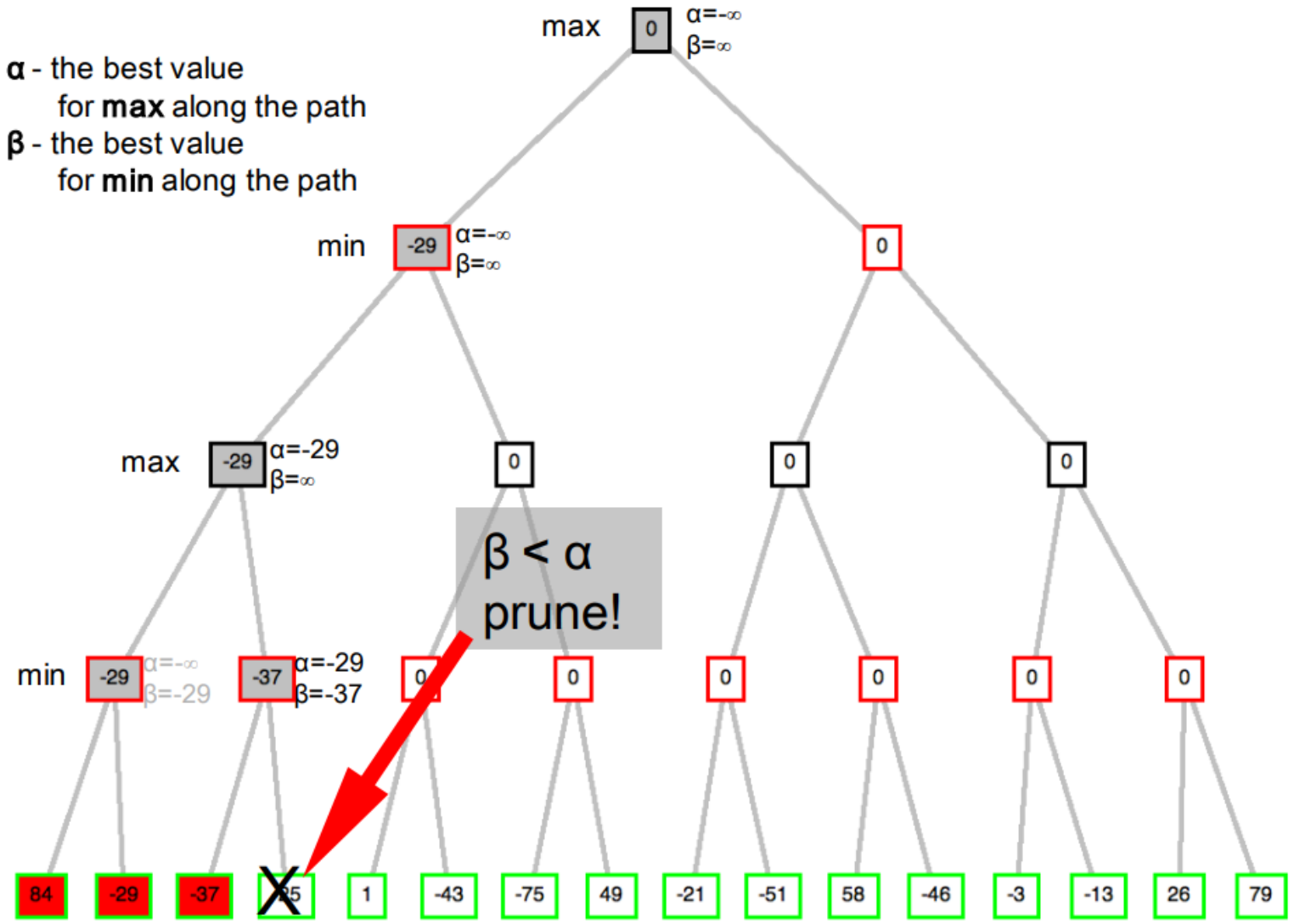


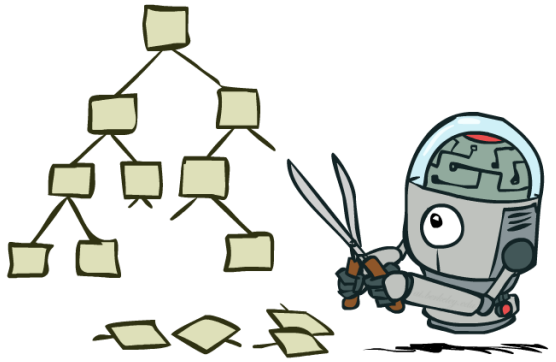
Alpha-Beta Budaması Örnek



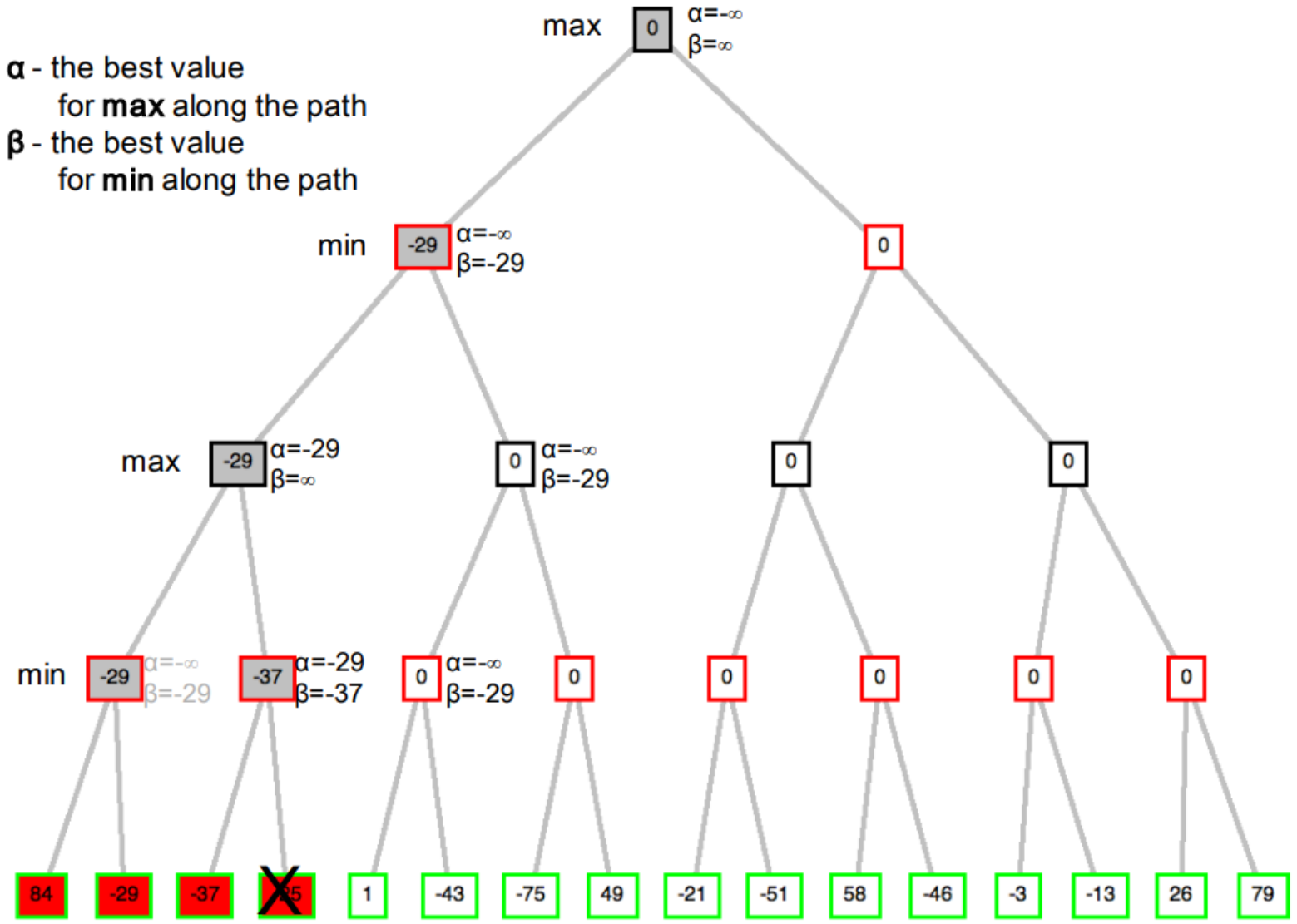


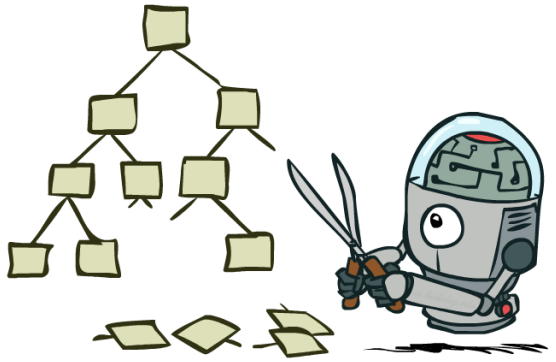
Alpha-Beta Budaması Örnek



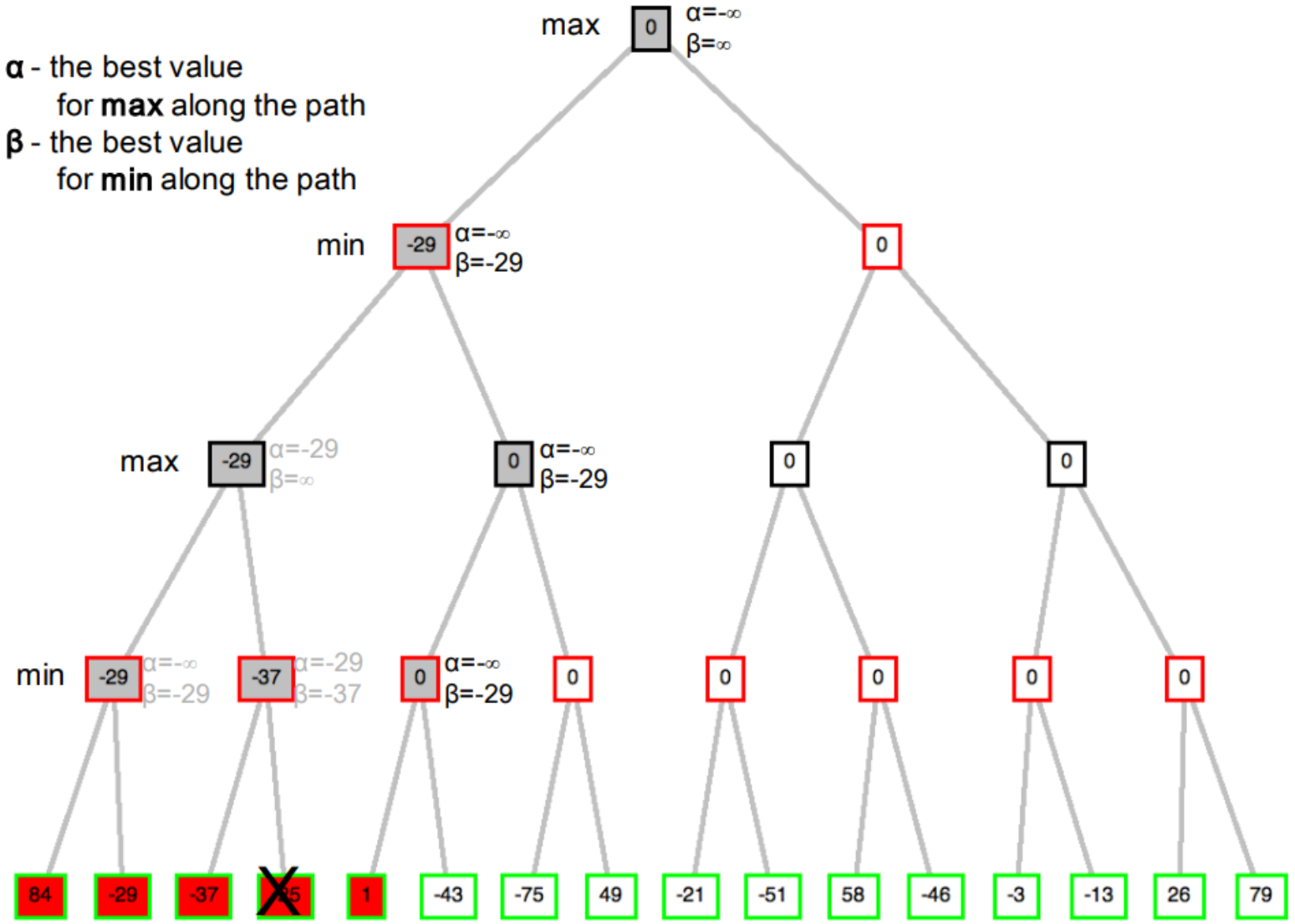


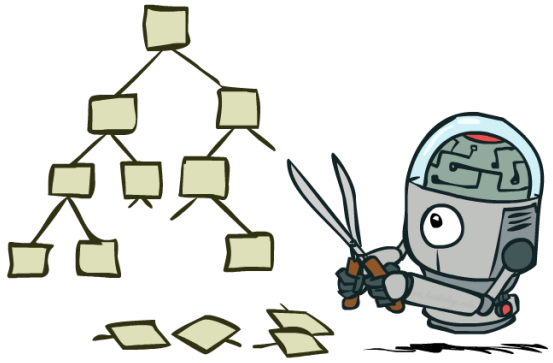
Alpha-Beta Budaması Örnek



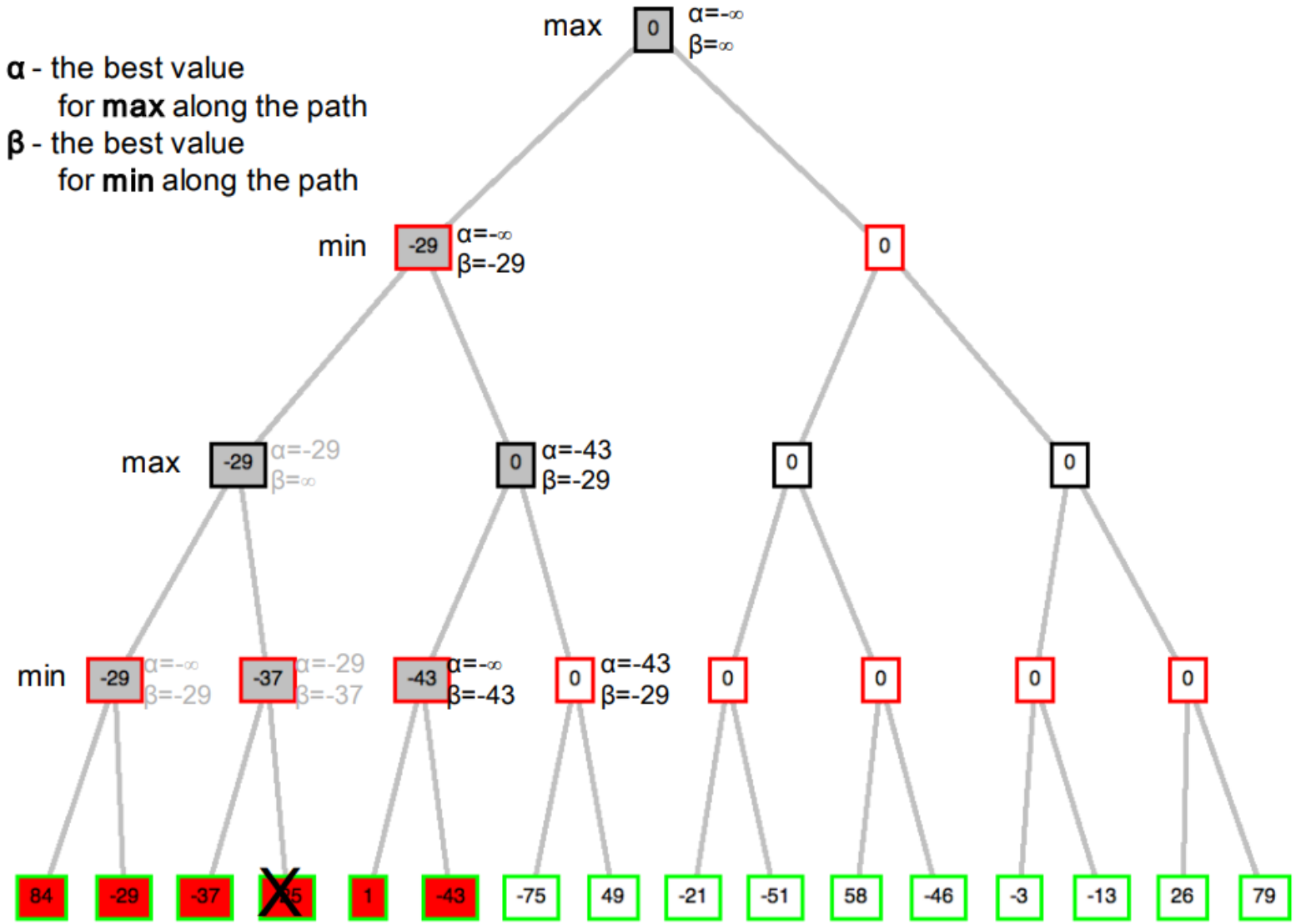


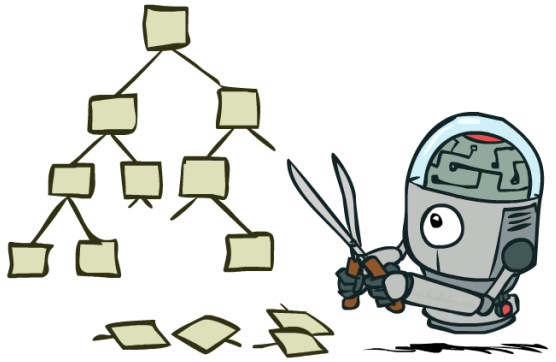
Alpha-Beta Budaması Örnek



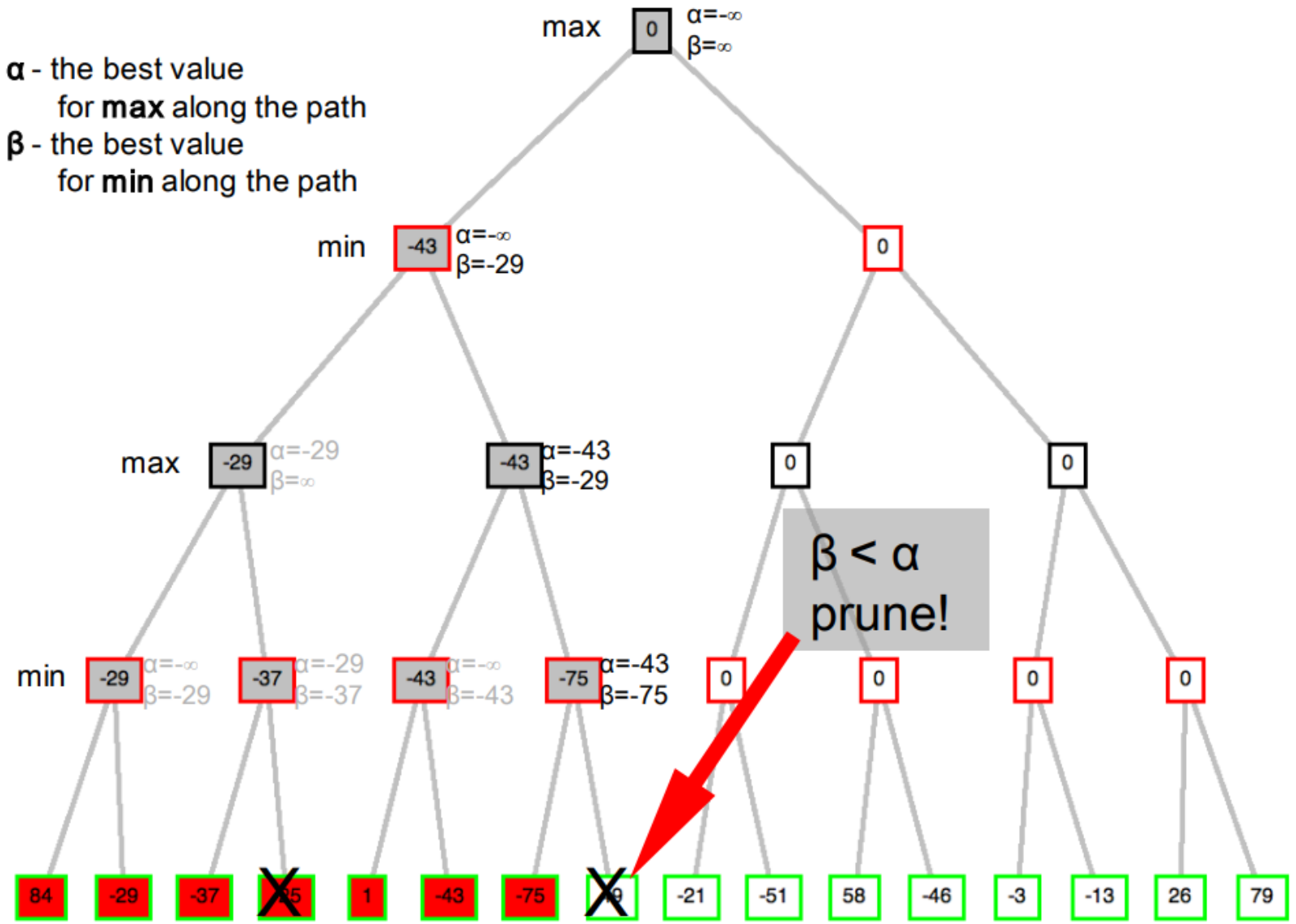


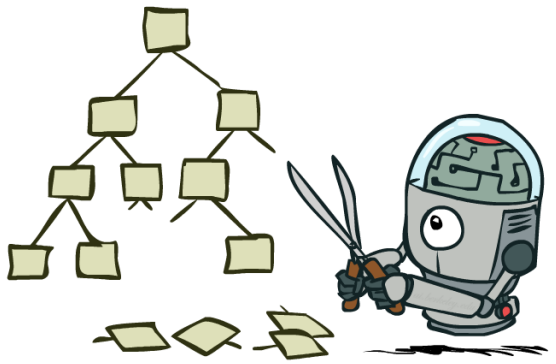
Alpha-Beta Budaması Örnek



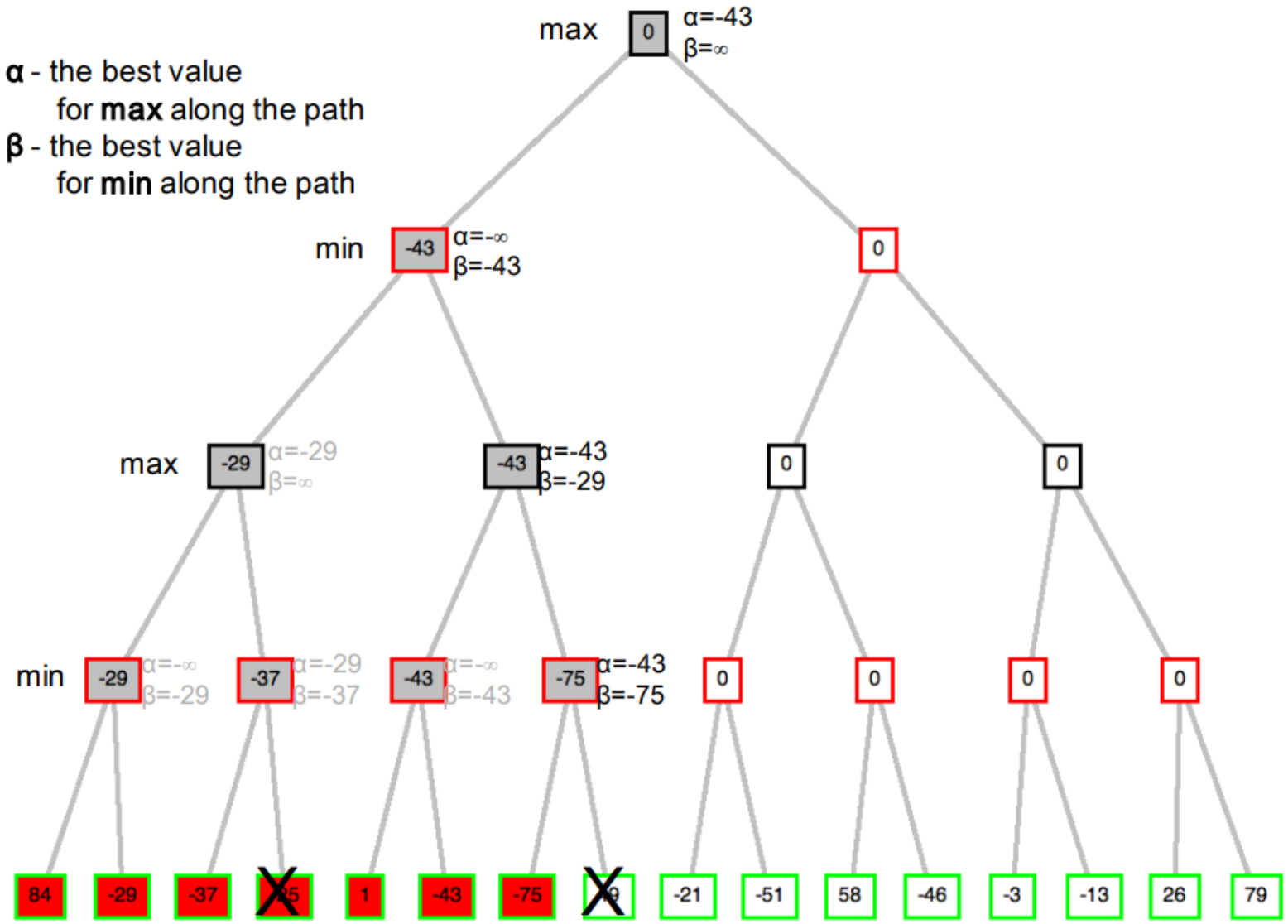


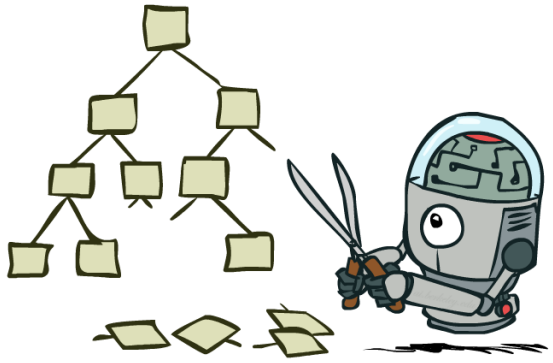
Alpha-Beta Budaması Örnek



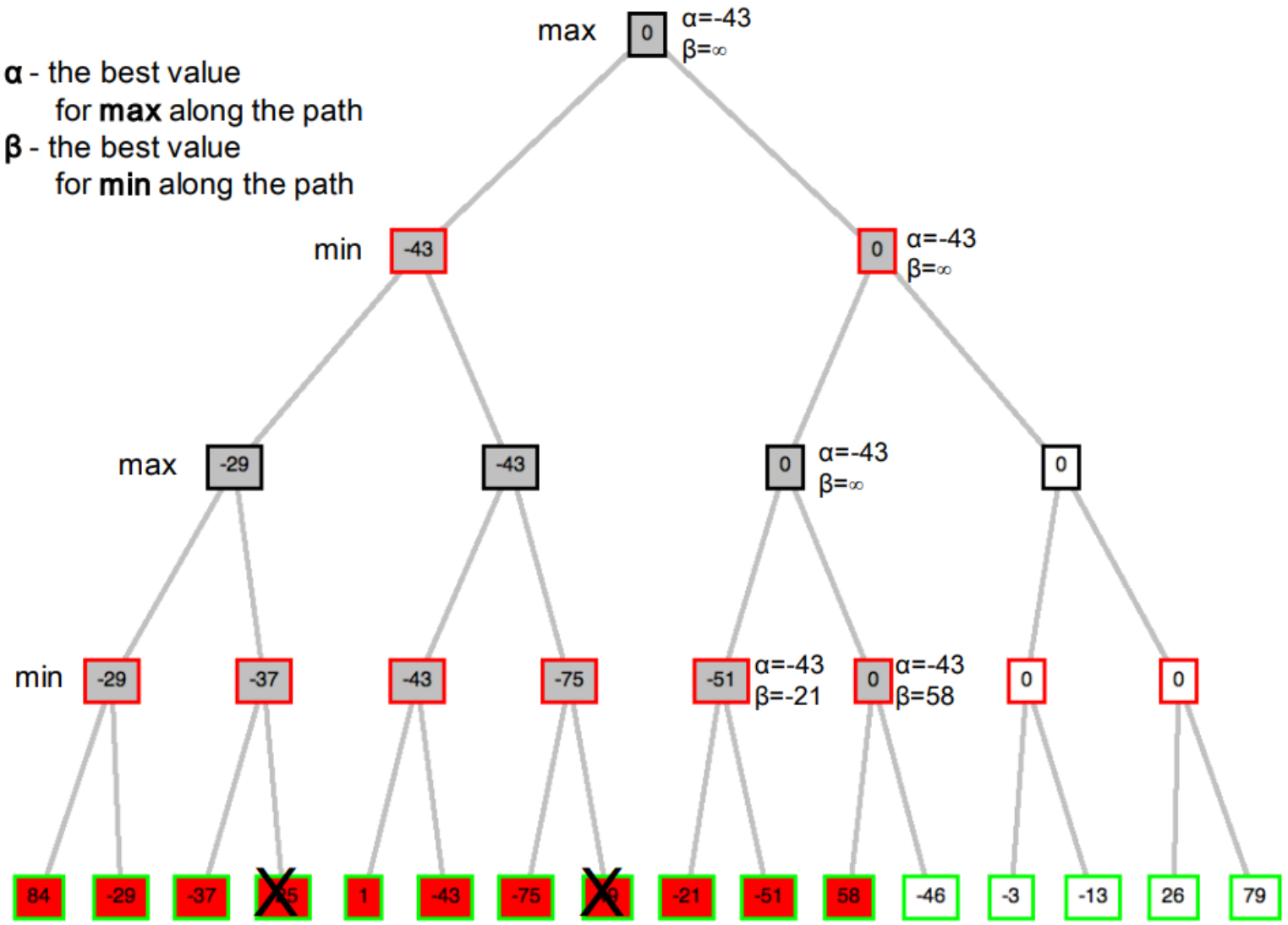


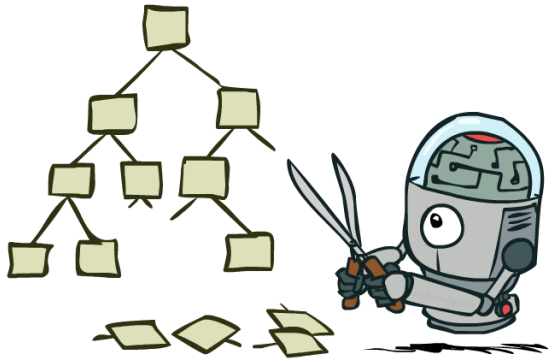
Alpha-Beta Budaması Örnek



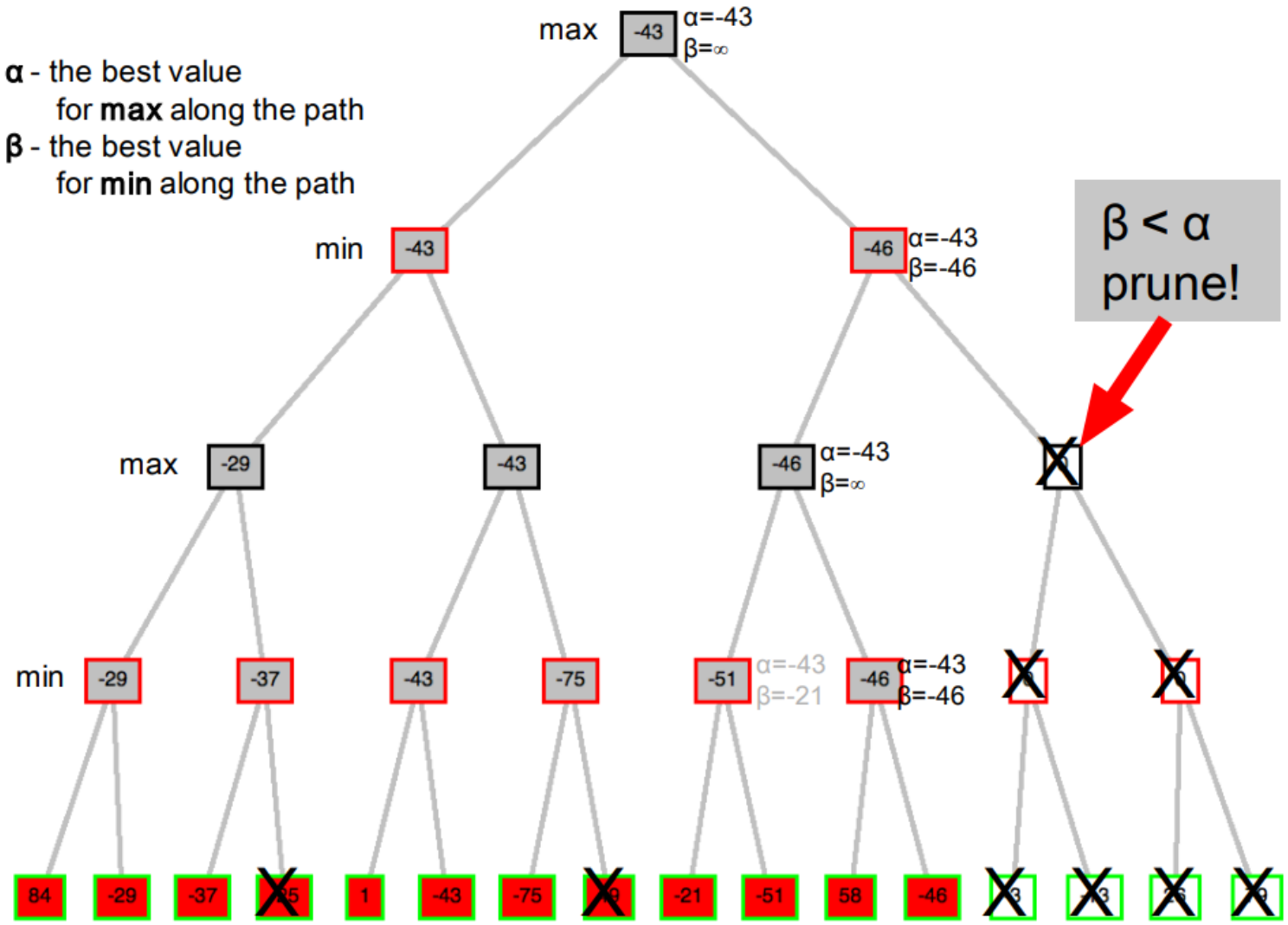


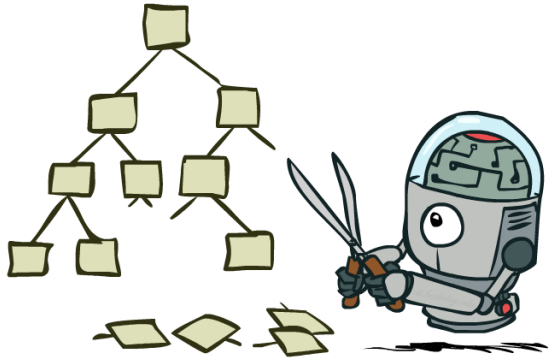
Alpha-Beta Budaması Örnek



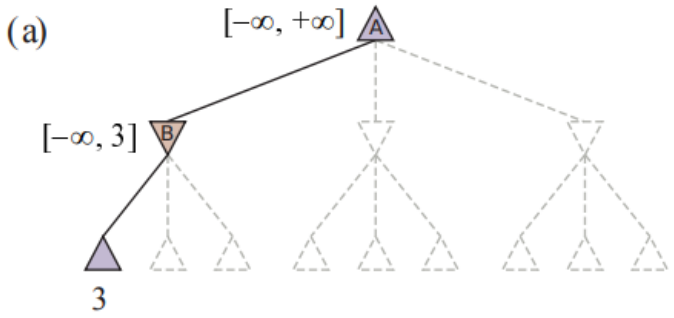


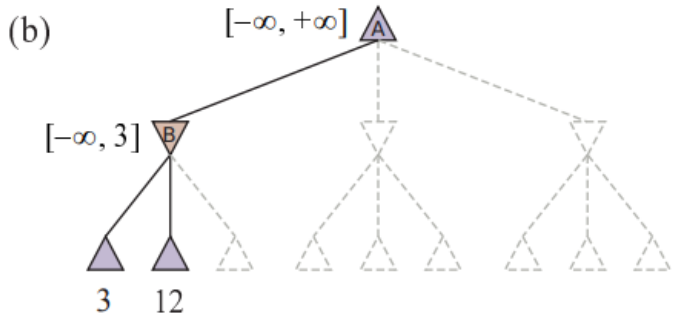
Alpha-Beta Budaması Örnek

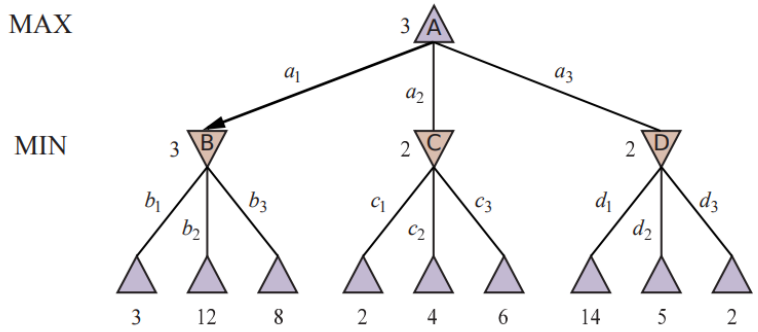


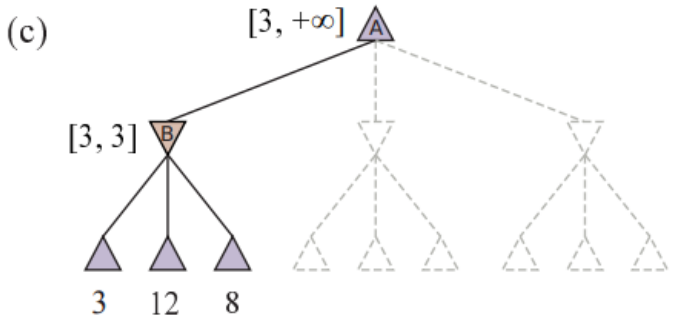


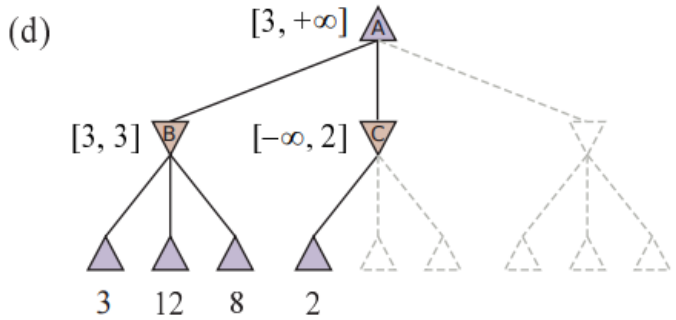
Alpha-Beta Budaması Örnek



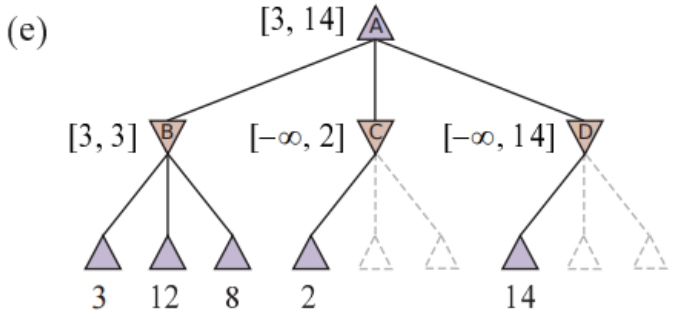


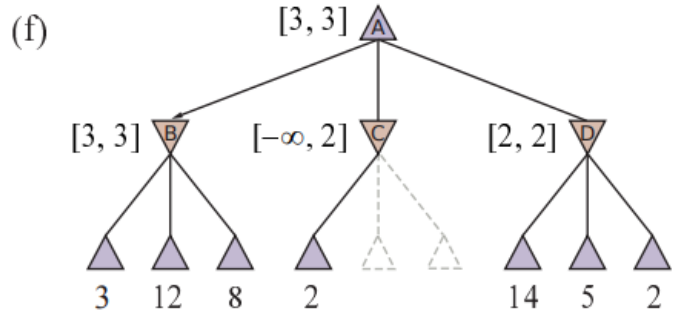


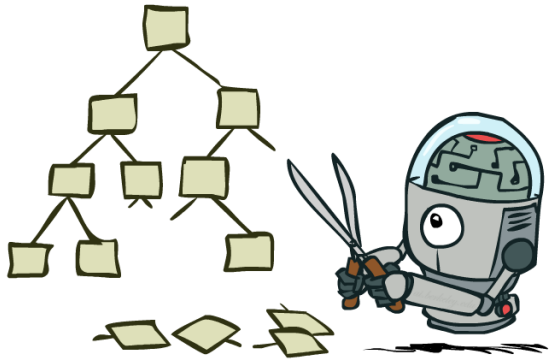




$$\begin{aligned}\text{MINIMAX}(root) &= \max(\min(3,12,8),\min(2,x,y),\min(14,5,2))\\ &= \max(3,\min(2,x,y),2)\\ &= \max(3,z,2) \qquad \text{where } z=\min(2,x,y)\le 2\\ &= 3.\end{aligned}$$

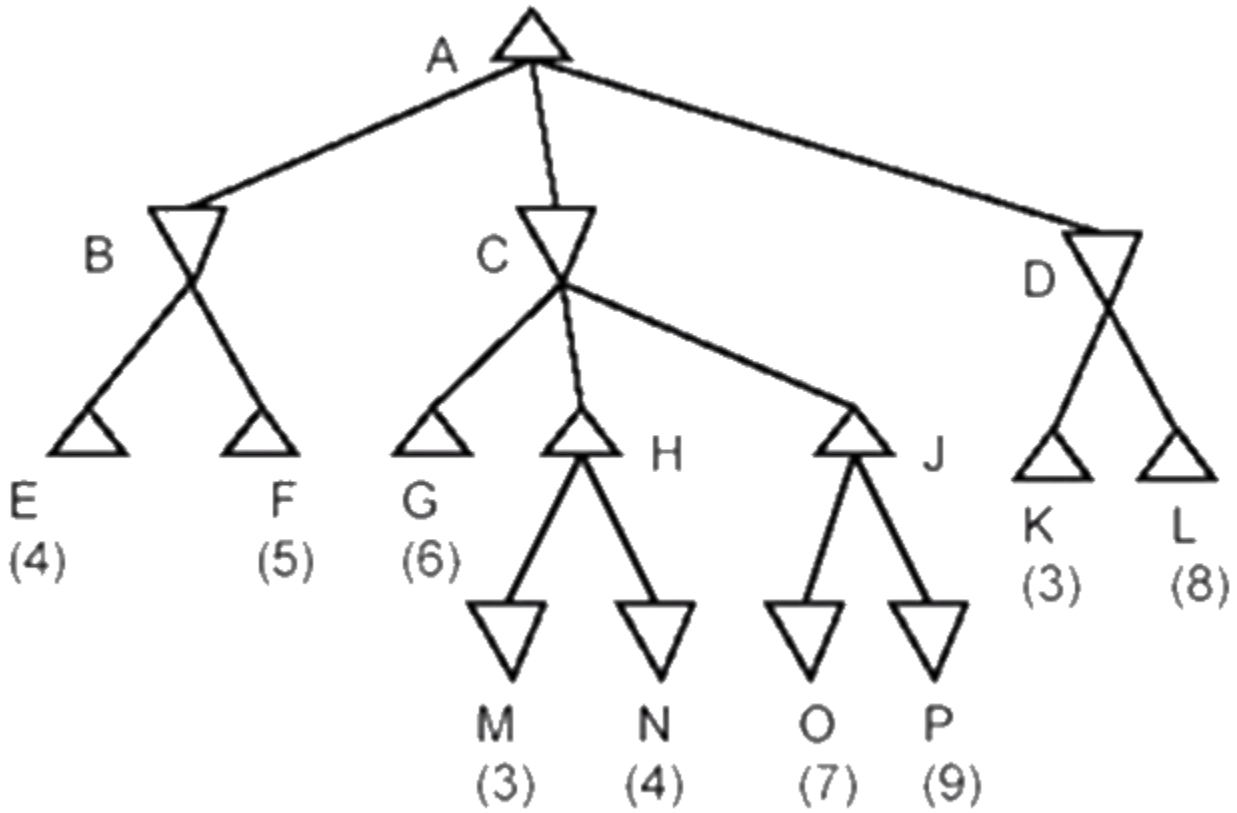


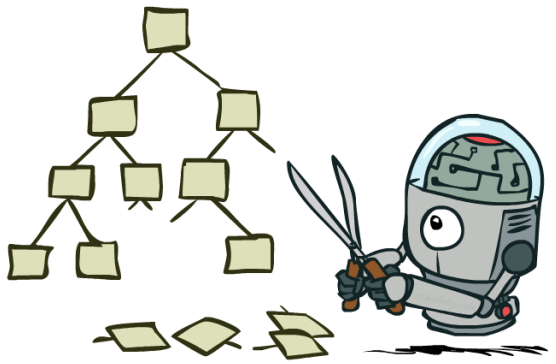




Alpha-Beta Budaması Örnek

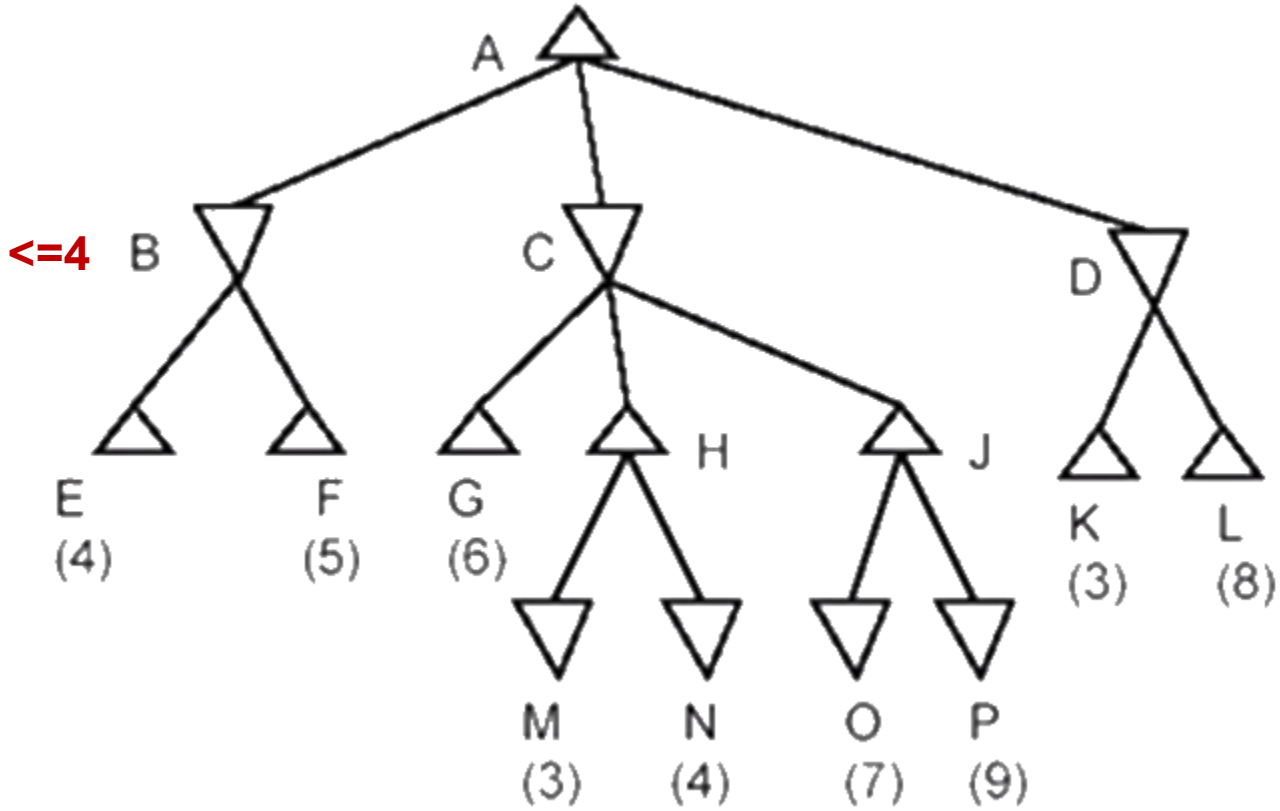
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

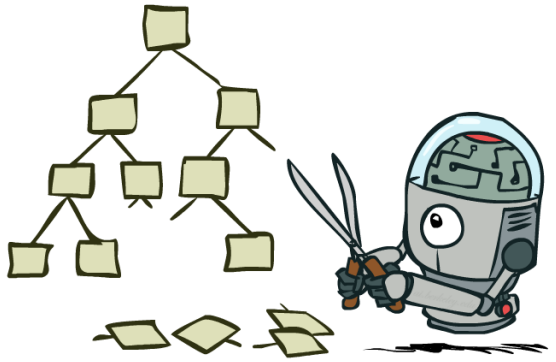




Alpha-Beta Budaması Örnek

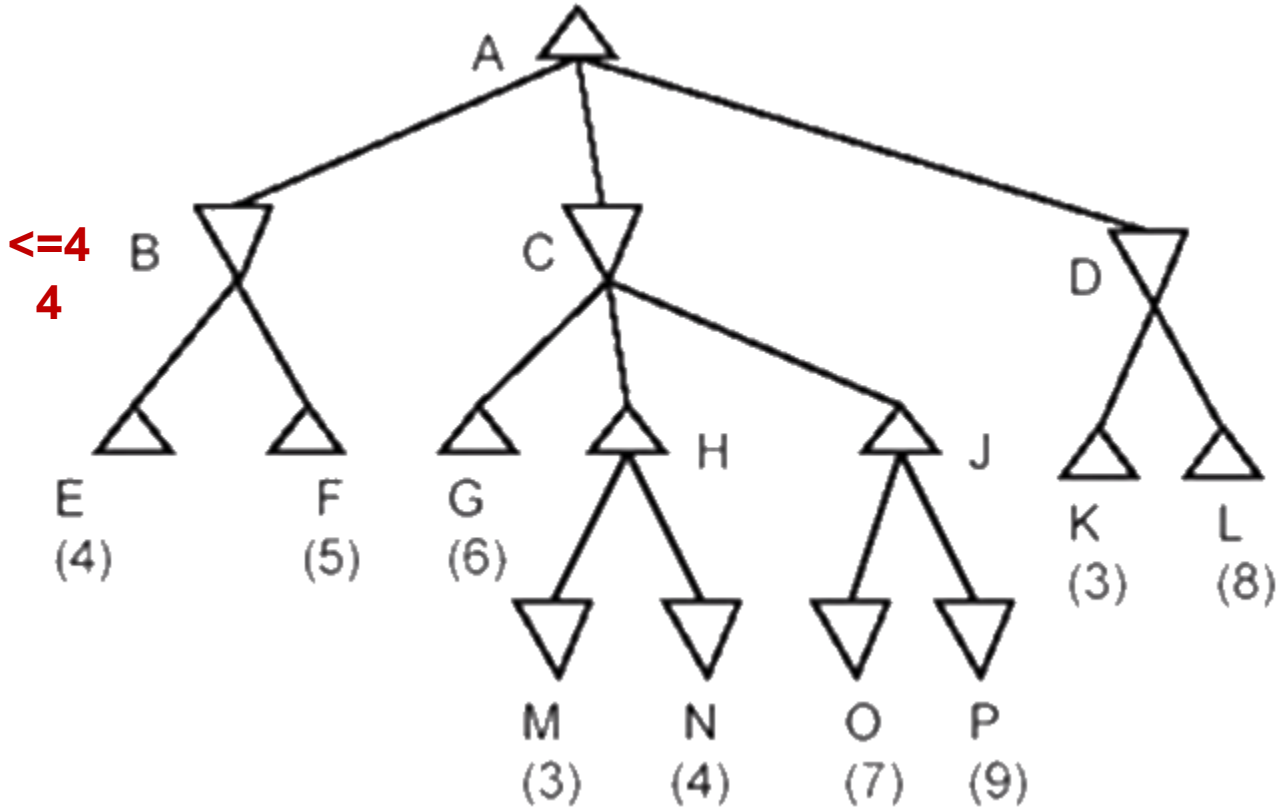
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

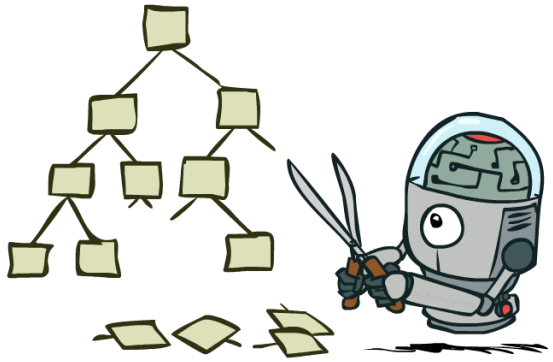




Alpha-Beta Budaması Örnek

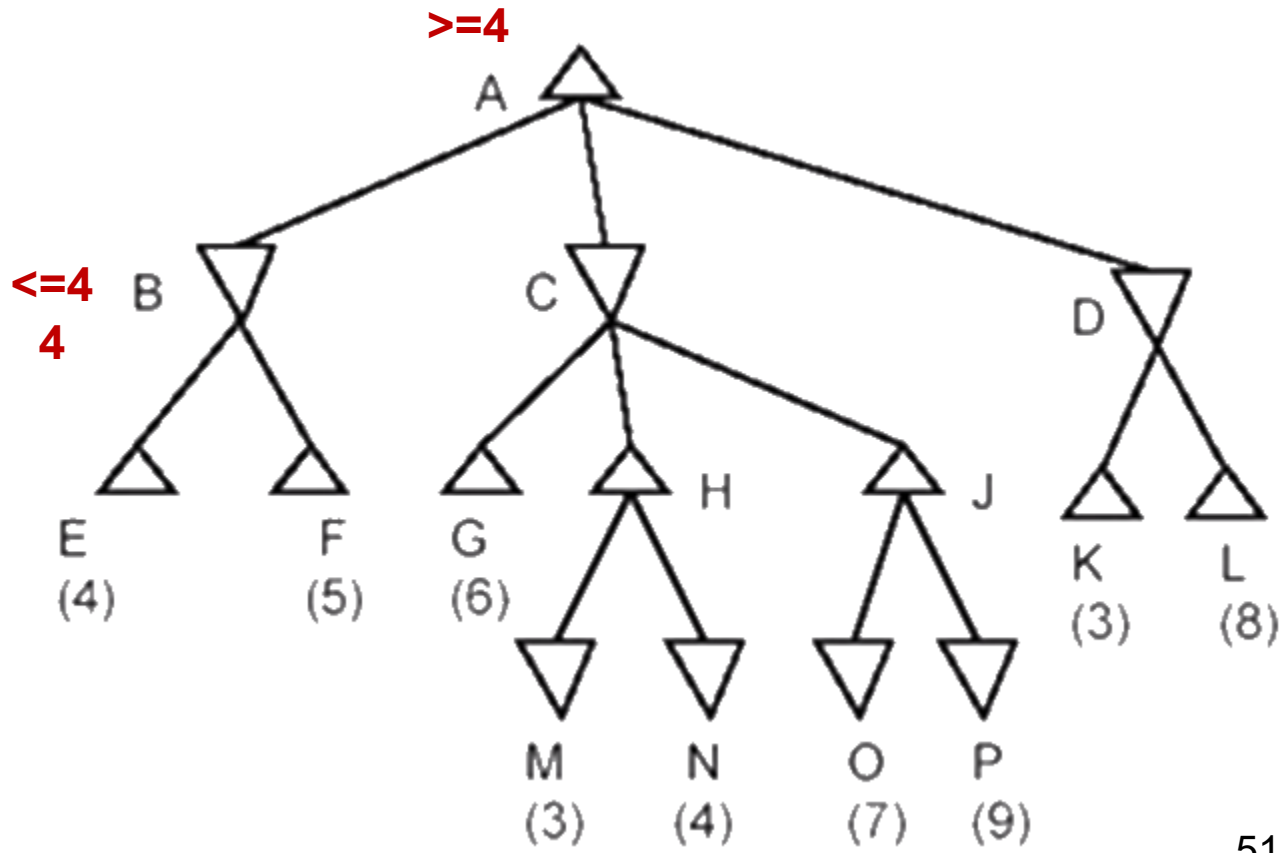
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

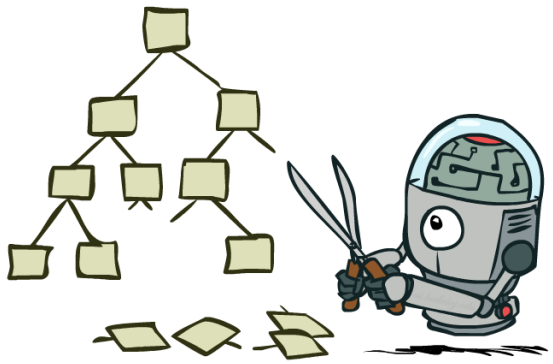




Alpha-Beta Budaması Örnek

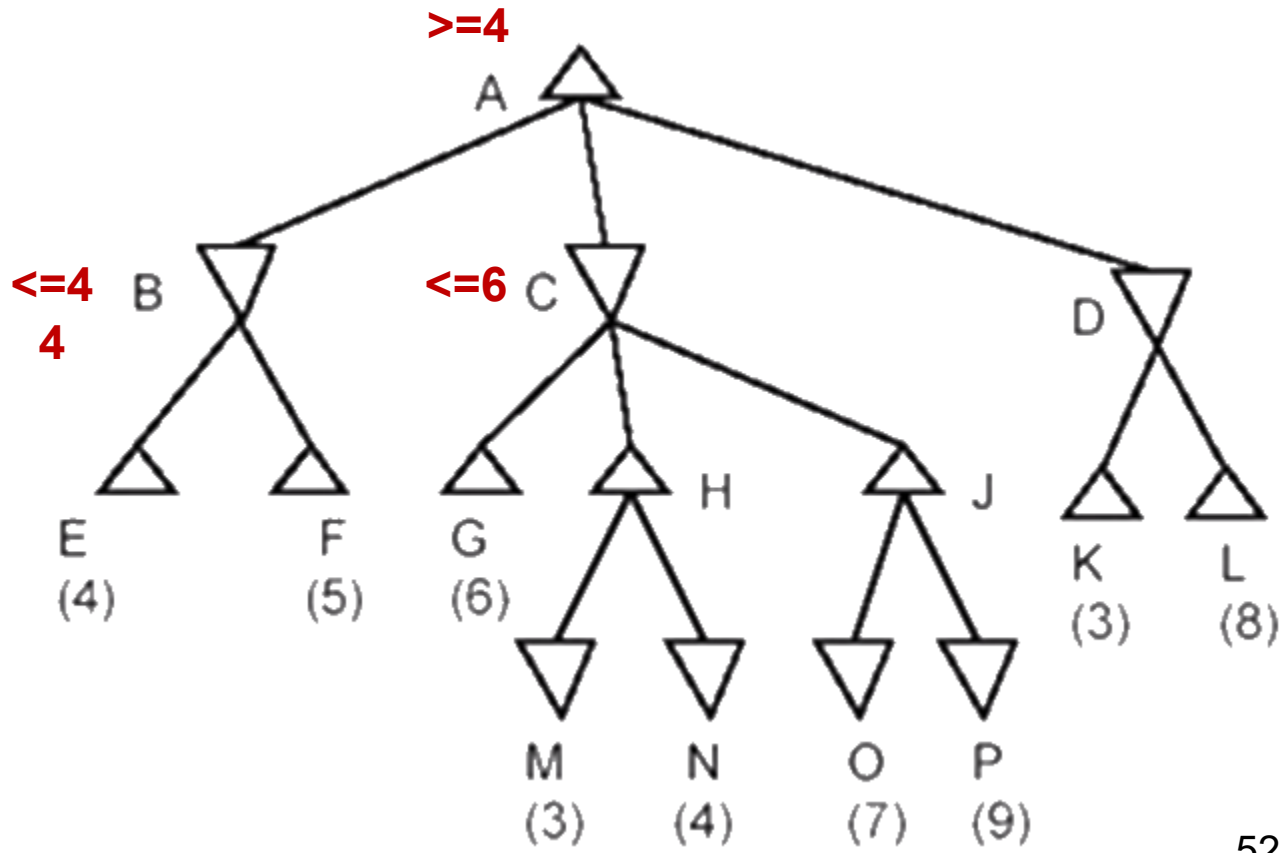
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

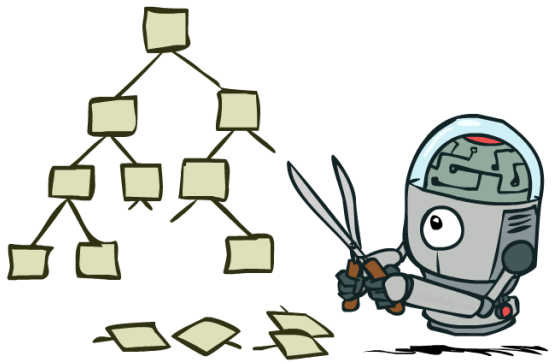




Alpha-Beta Budaması Örnek

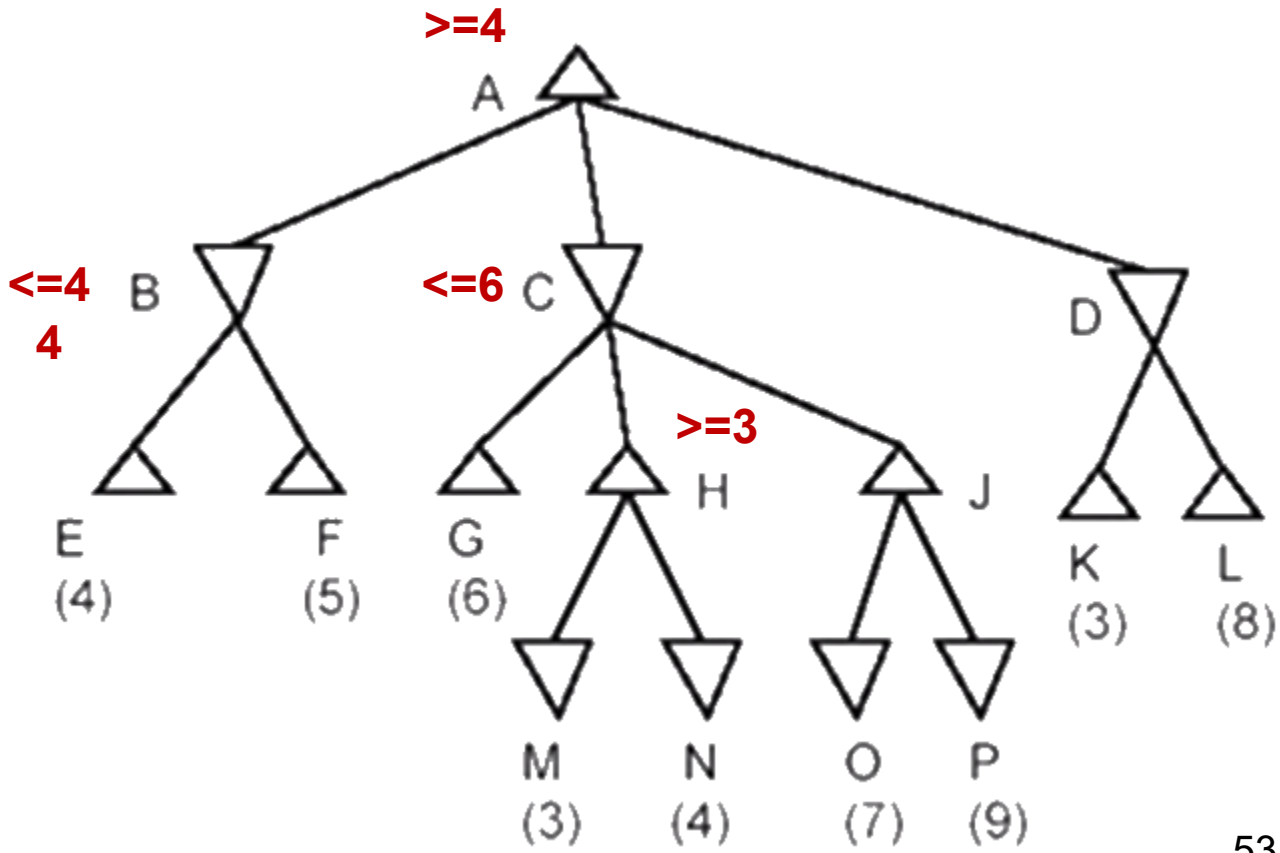
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

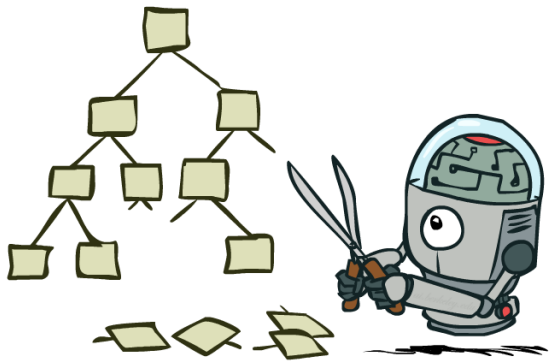




Alpha-Beta Budaması Örnek

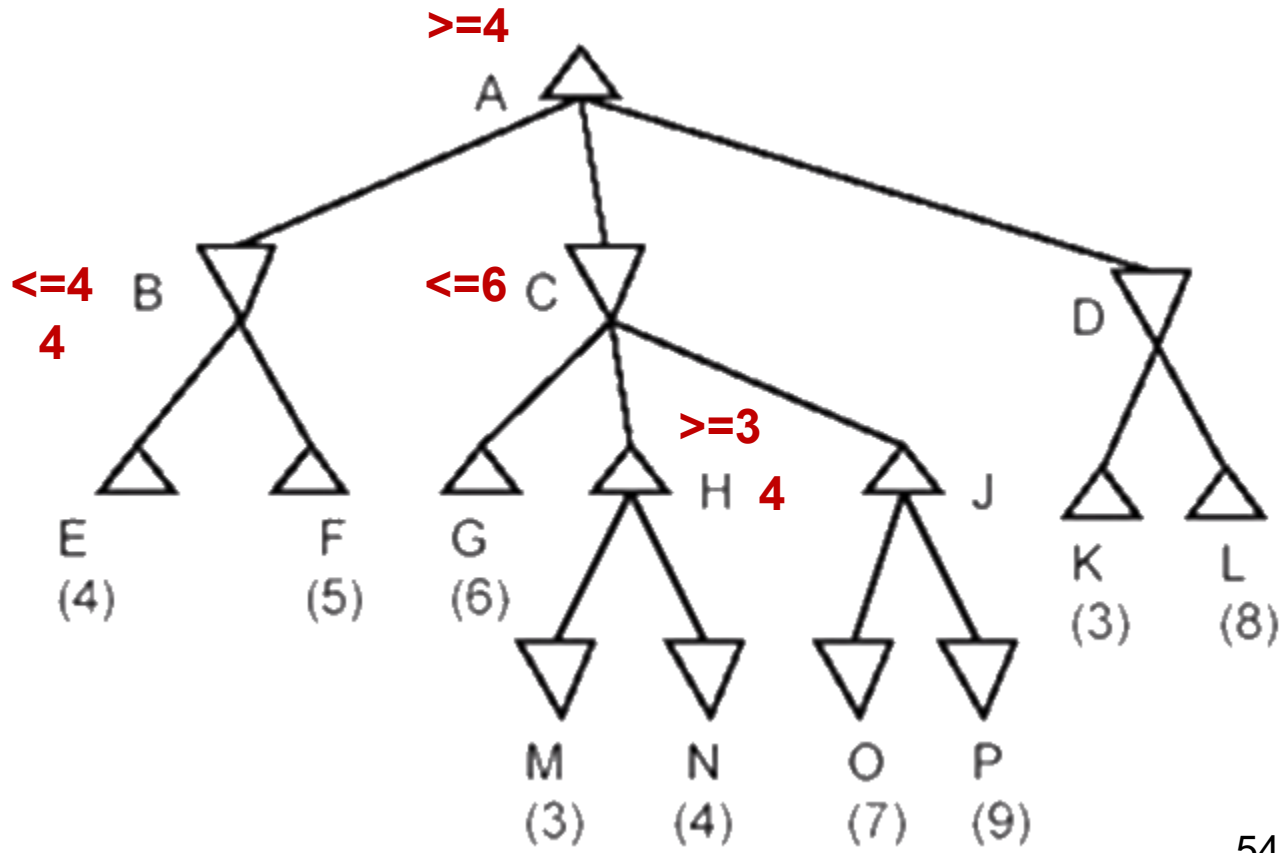
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

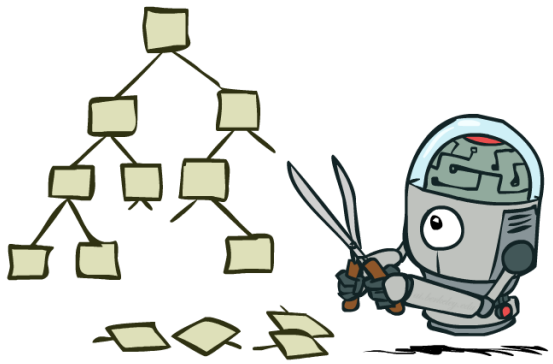




Alpha-Beta Budaması Örnek

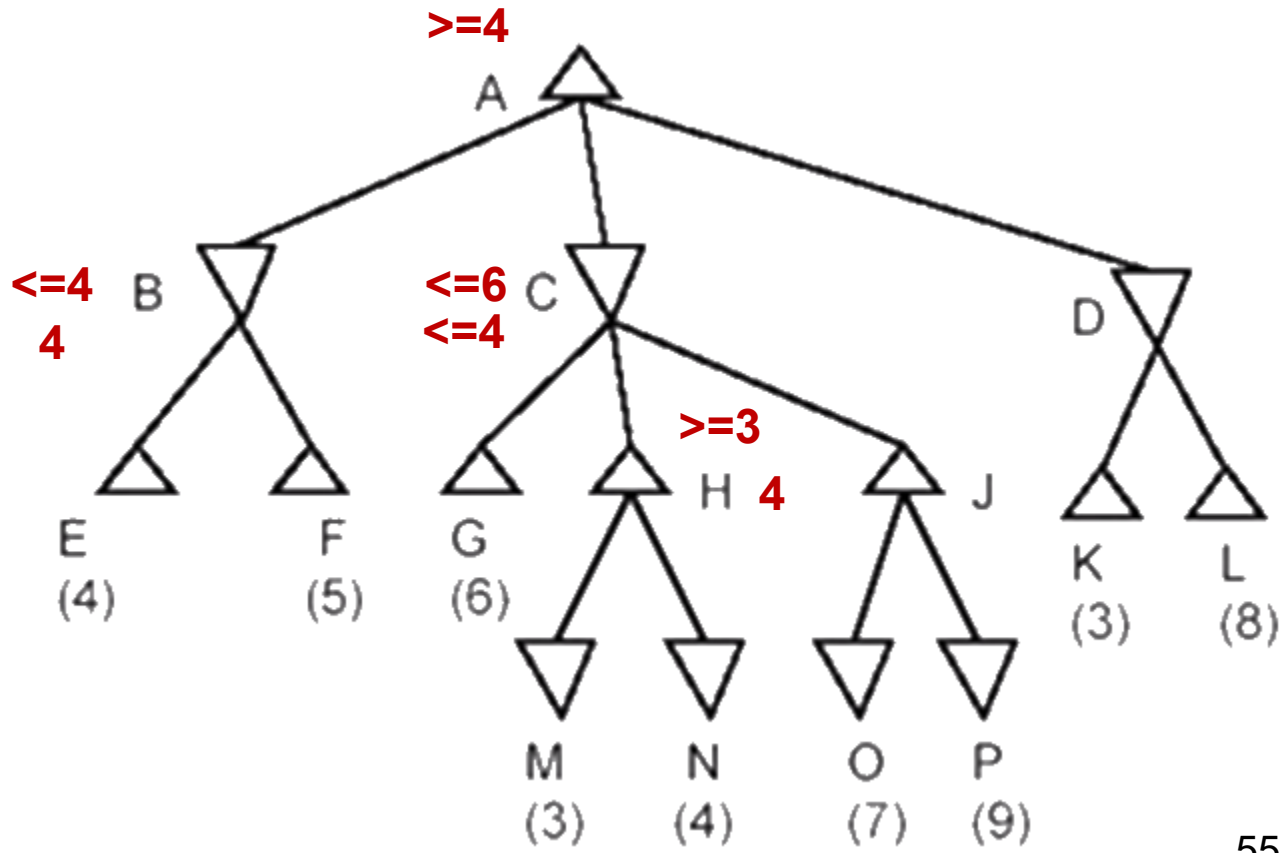
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

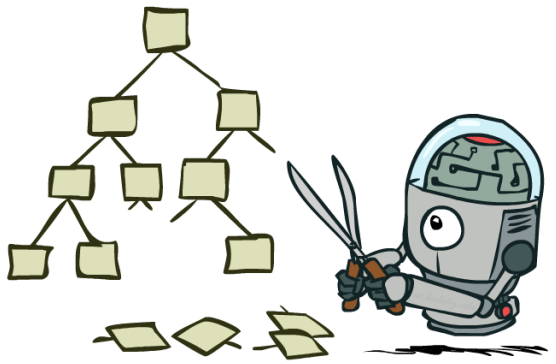




Alpha-Beta Budaması Örnek

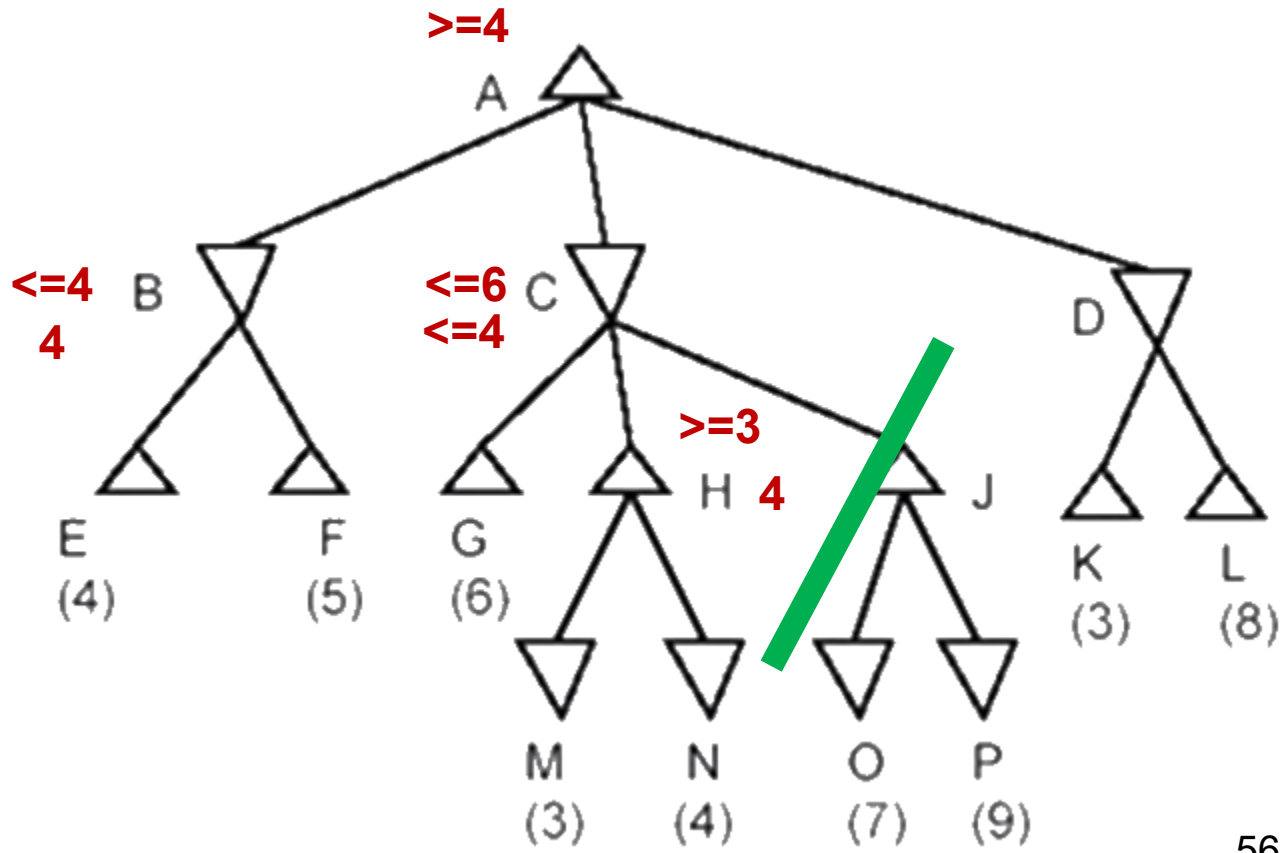
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

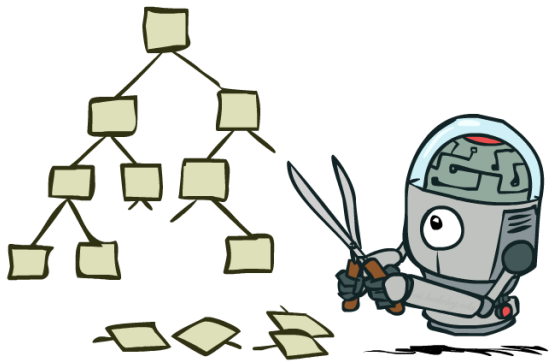




Alpha-Beta Budaması Örnek

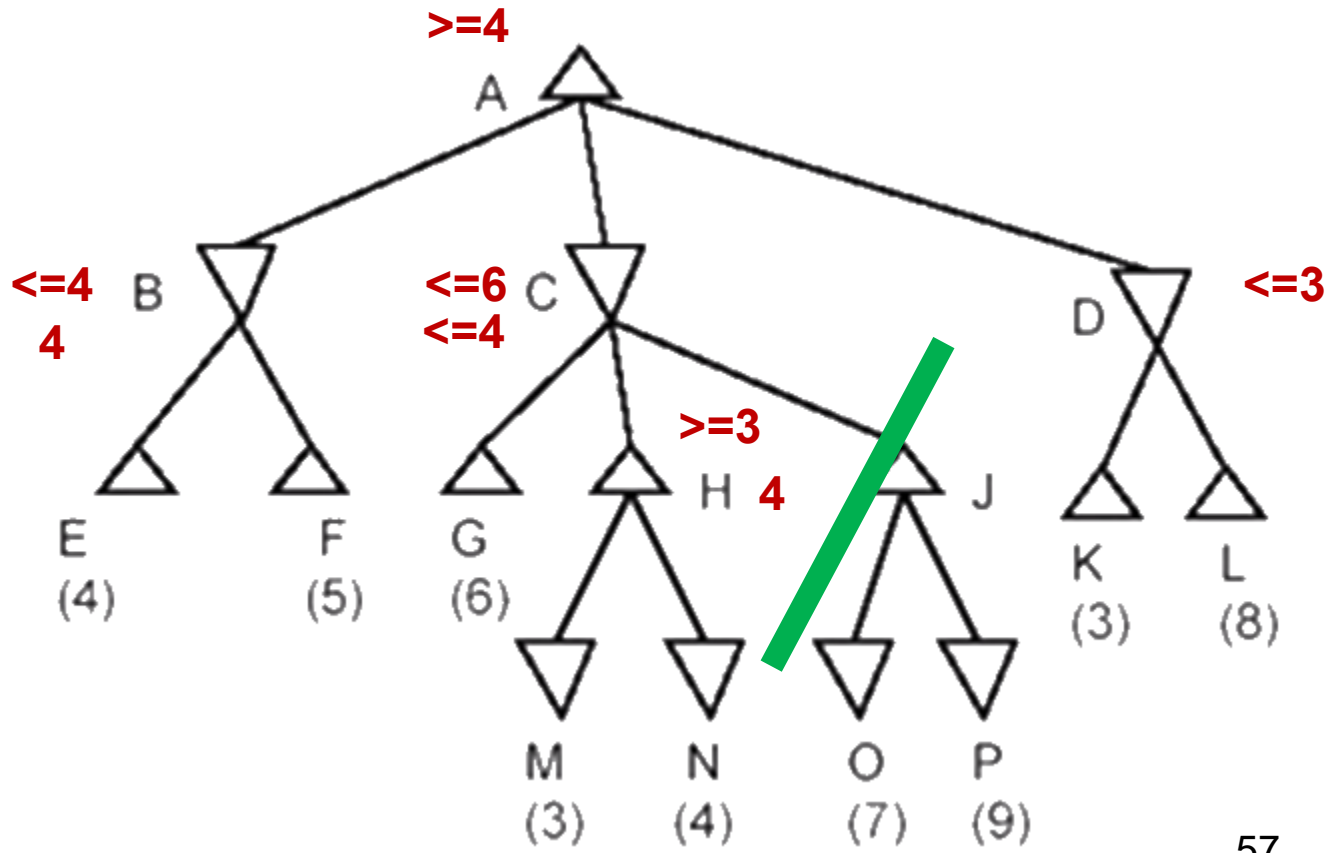
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

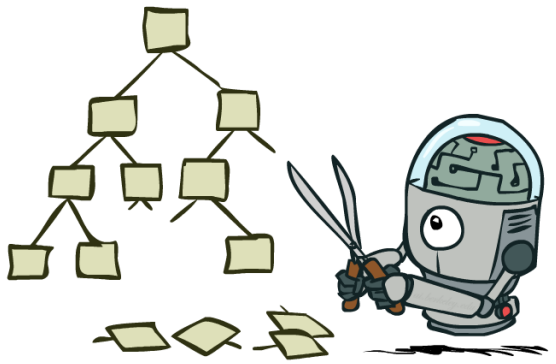




Alpha-Beta Budaması Örnek

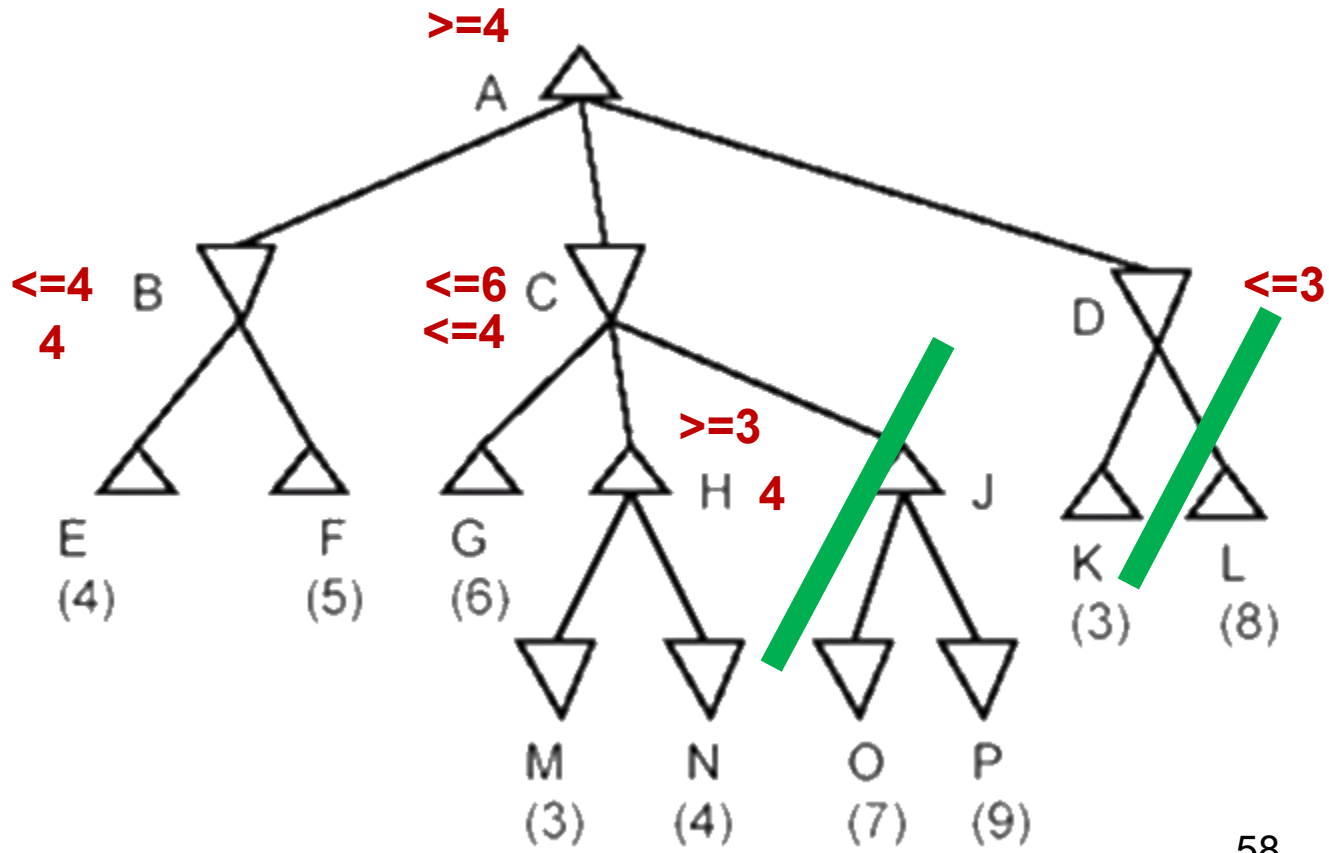
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.

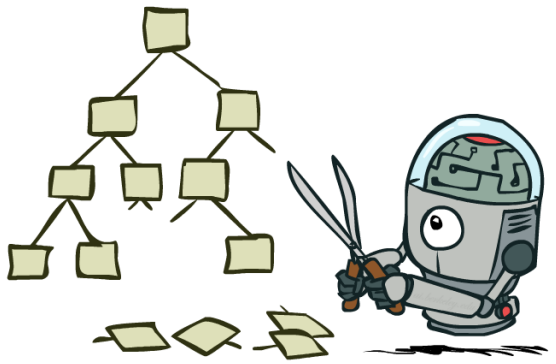




Alpha-Beta Budaması Örnek

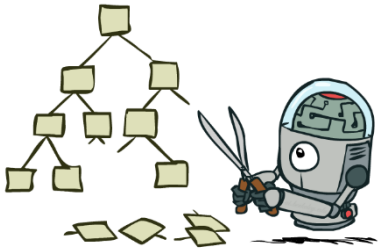
- **Alfa:** Max oyuncusu için garantilenmiş en küçük değerlendirmedir.
- **Beta:** MAX'ın alabileceği fonksiyon değerlerinin en büyüğüdür.



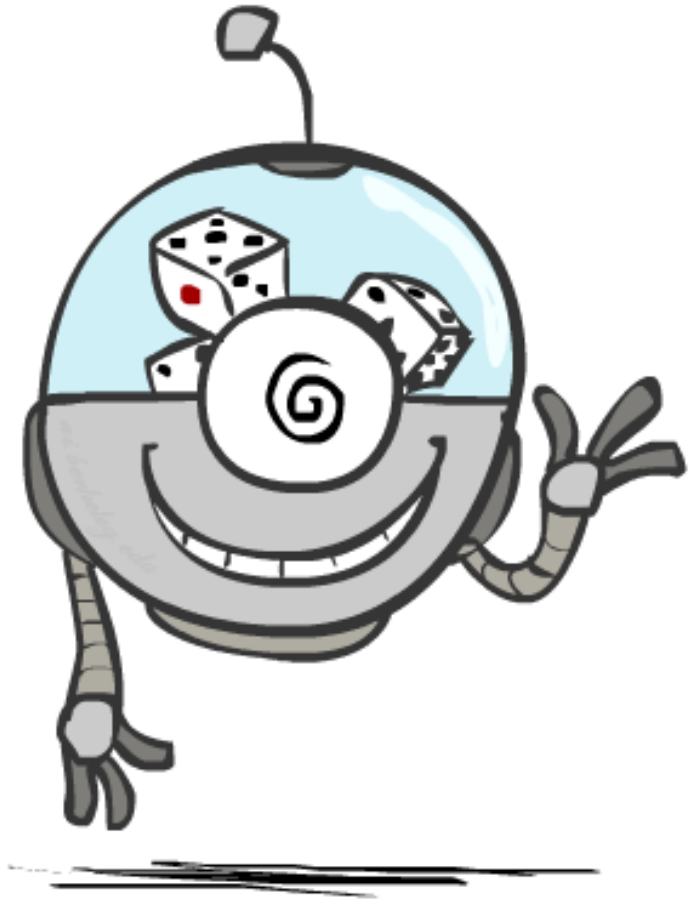


Alpha-Beta Budaması Özellikleri

- Budama işlemi nihai sonucu etkilemez.
  - Yani Alpha-Beta Budaması ile Minimaks tekniğinin sonucu aynıdır.
- İyi bir hareket sıralaması, budama işleminin etkinliğini arttırır.
- "Mükemmel sıralama" ile zaman karmaşıklığı :
  - $O(b^{m/2})$
- Arama derinliğini iki katına çıkarır.
- 'Hangi hesaplamaların alakalı olduğu' hakkında basit bir akıl yürütme örneği olarak düşünülebilir.
  - Bir tür üst akıl yürütme

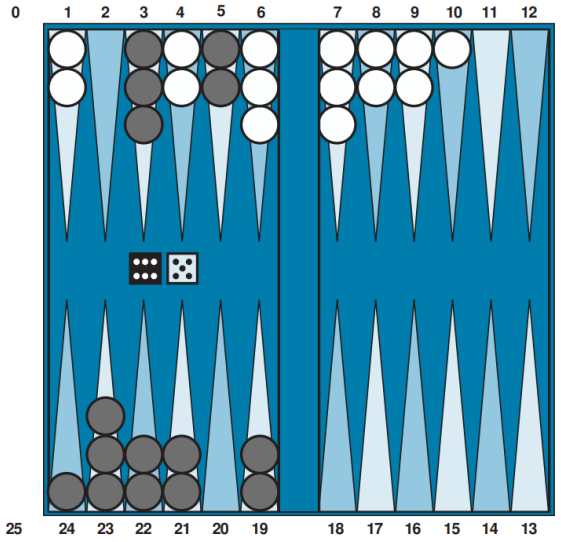


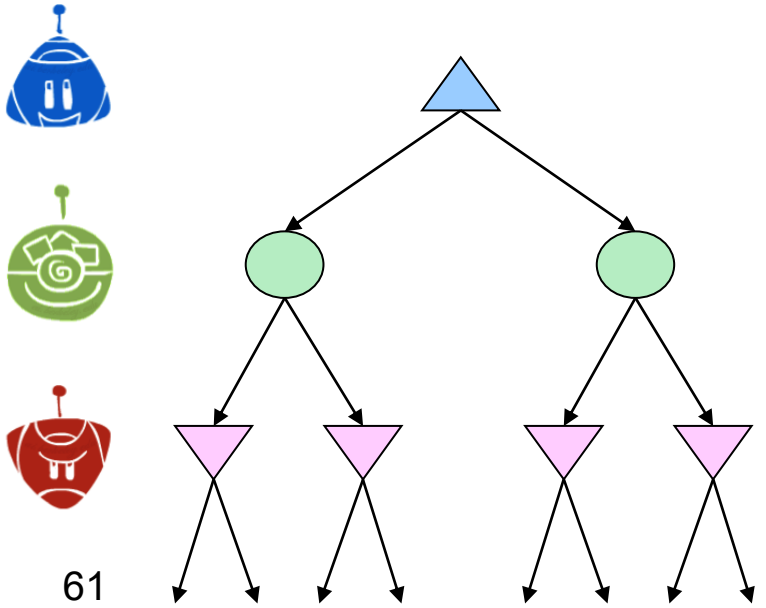
Stokastik (Rastgele Sonuçlu) Oyunlar



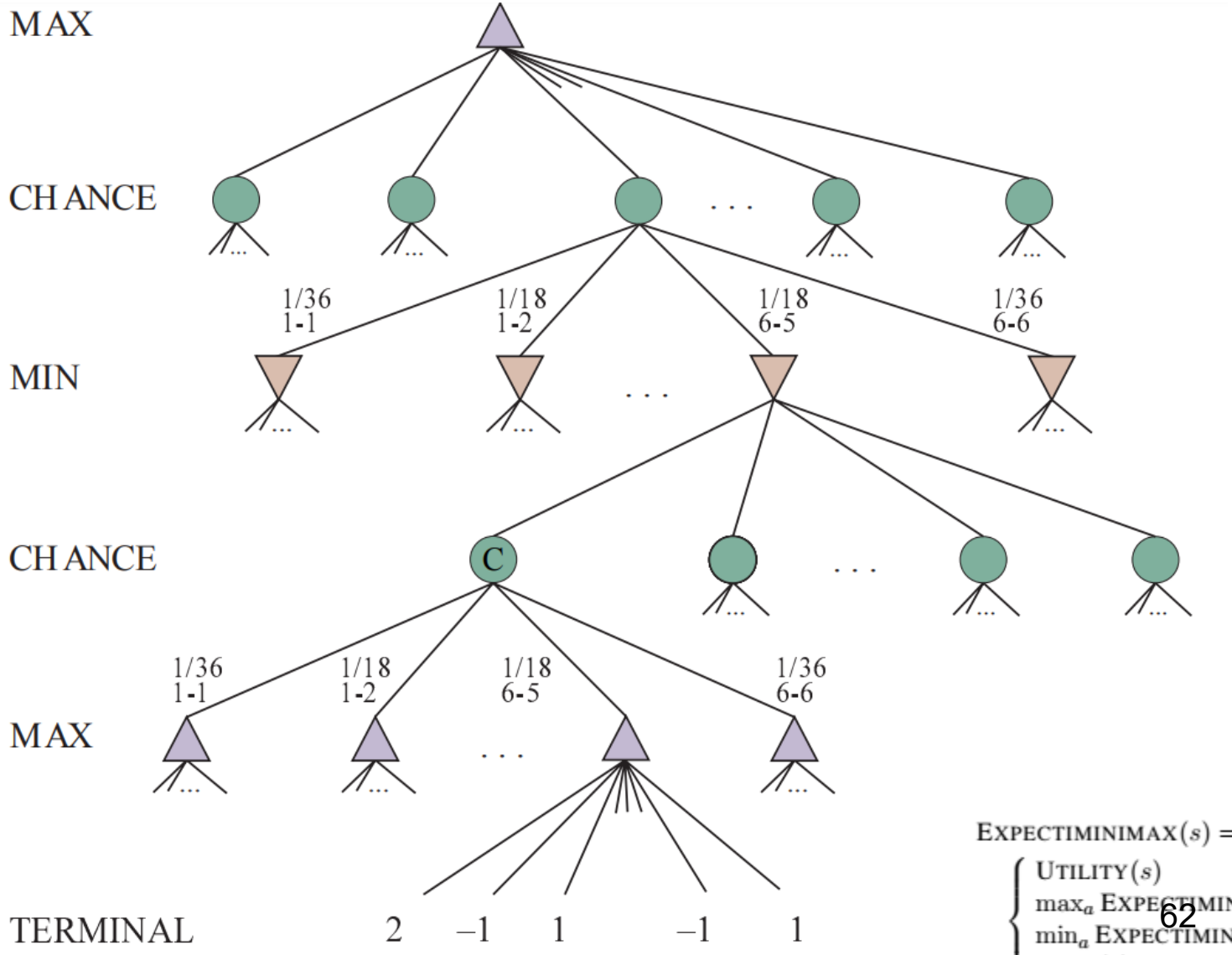
Stokastik Oyunlar

- Tavla gibi şans faktörü ve stratejinin etkin olduğu oyunlar
- Başka bir ajanın eylemi hakkında olasılıklı bir inanca sahip olmak, ajanın hamleleri rastgele yaptığı anlamına gelmez!
- Bu tür oyunların ağaç temsilinde 3 tür düğümden bahsedilir:
  - max düğümleri
  - min düğümleri
  - şans düğümleri
- **Çözüm:**
  - **Expectiminimax**



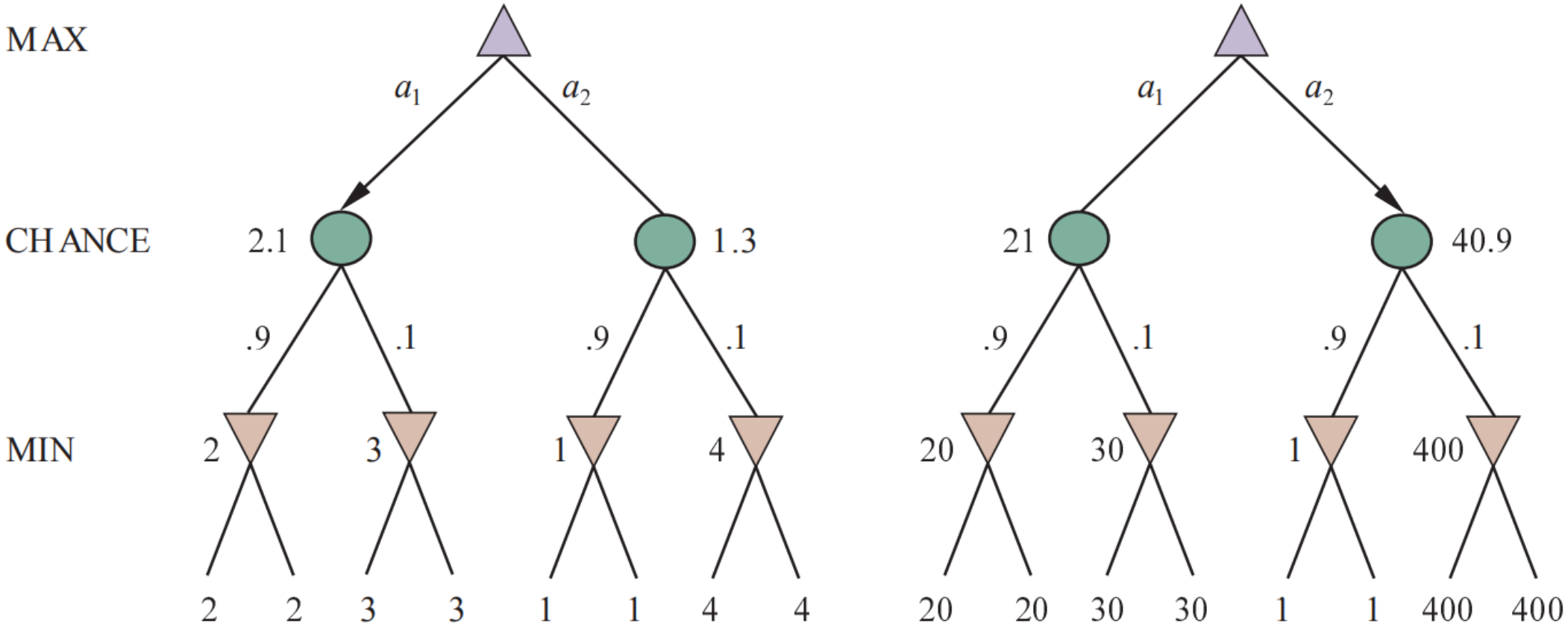


Stokastik Oyunlar



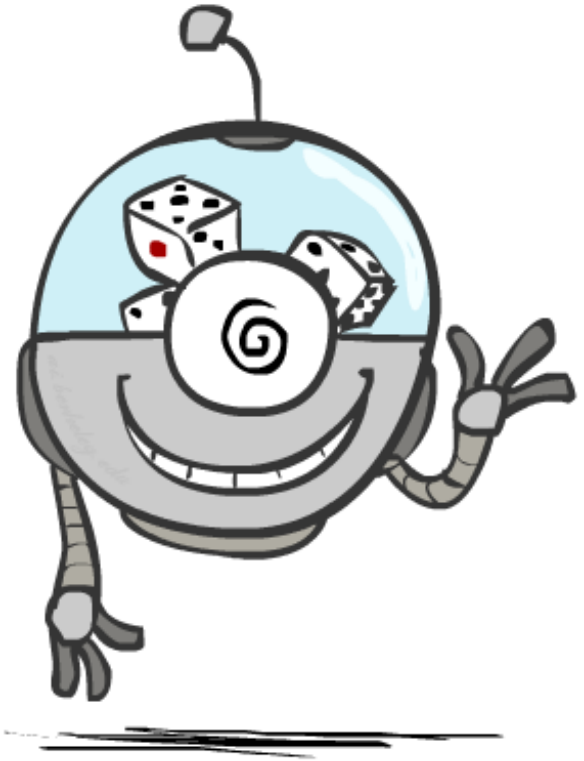
$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\ \max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\ \min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \\ \sum_r P(r)\text{EXPECTIMINIMAX}(\text{RESULT}(s,r)) & \text{if PLAYER}(s) = \text{CHANCE} \end{cases}$$

Stokastik Oyunlar



Stokastik Oyunlar Analiz

- Zaman Karmaşıklığı:
  - $O(b^m)$ yerine $O(b^m n^m)$
  - n: Şans faktörünün çıktılarının yani dallanmalarının sayısı
- Hem yer hem zaman karmaşıklığı fazladır.
- Karmaşıklık sebebiyle derinine arama uygunlanması zordur.
- Budama algoritmaları uygulanabilir.



Referanslar

- Artificial Intelligence A Modern Approach, Stuart Russell and Peter Norvig, Prentice Hall Series in Artificial Intelligence.
- Yapay Zeka, Vasif Vagifoğlu Nabiyev, Seçkin Yayıncılık
- http://aima.cs.berkeley.edu/instructors.html
- https://courses.cs.washington.edu/courses/csep573/11wi/lectures/05-games.pdf
- http://www.cs.bilkent.edu.tr/~duygulu/Courses/CS461/Notes/Games.pdf
- http://aytugonan.cbu.edu.tr/YZM3217/LectureNotes/YZM3217_lecture6.pdf