

# 1. Yapay Zeka'nın Tanımı

- **Zeka** : (D. Lenat ve E. Feigenbaum'un tanımları)

“Karmaşık bir problemi çözmek için gerekli bilgileri toplayıp birleştirme kabiliyetidir”

“Karmaşık bir problemi, çözüm arama alanını daraltarak kısa yoldan çözebilme kabiliyetidir”

**Hedef** : Bir problemi, etkin ve kısa yoldan çözmek

**Zeka'nın sözlük anlamı** : İnsanın düşünme, akıl yürütme, nesnel gerçekleri algılama, kavrama, yargılama, sonuç çıkarma yeteneklerinin tümü. Ayrıca : Soyutlama, öğrenme ve yeni durumlara uyma gibi yetenekler de zeka kapsamı içindedir.

İnsan zekası ve diğer canlıların zekası

**Yapay zeka** ise, bu özelliklere sahip organik olmayan sistemlerdeki zekadır.

01.10.2003

Y. Doç. Dr. Aybars UĞUR (517 Yapay Zeka Dersi)

1

# Yapay Zekanın Tanımı

YZ tanımlarının türleri : Düşünme, çıkarsama ve davranış (eylem)

Bilim kurgudaki insan benzeri robot, ideal zeki makinedir. Bu, mantıklı zeki programlar yapmaktan farklıdır.

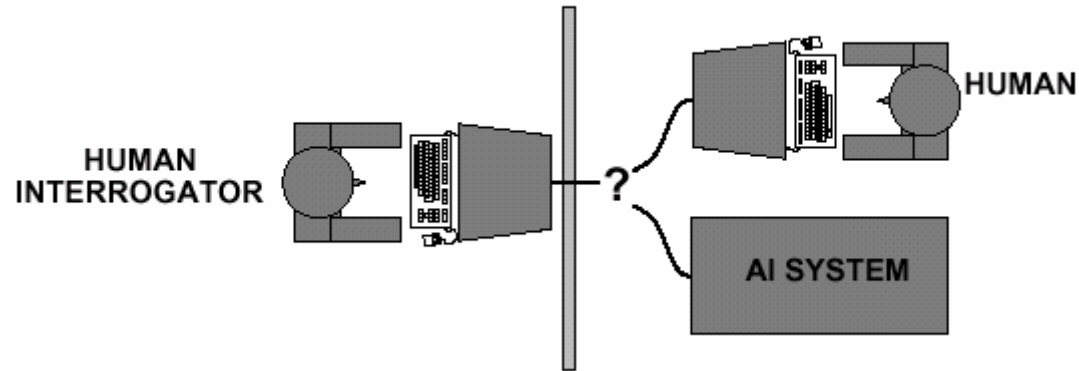
- 1. Kategori : İnsan gibi düşünen sistemler
- 2. Kategori : İnsan gibi davranan sistemler.
- 3. Kategori : Mantıklı düşünen sistemler.
- 4. Kategori : Mantıklı davranan sistemler.

Doğru şeyi yapan sistem mantıklıdır. İnsan mantıksız değildir?

# Turing Testi

İnsan gibi düşünen sistemler : Turing Test

Alan Turing tarafından 1950’de önerildi. Bir makinenin zeki olup olmadığını belirleyen test.



Bir bilgisayarın bu testi geçebilmesi için hangi yeteneklere sahip olması gereklidir?

# Yapay Zekanın Temelleri

AI, yeni bir alan olmasına rağmen, diğer disiplinlerden teknikler ve fikirler alır.

- Felsefe (MÖ 428-)
- Matematik, Algoritma (800-)
- Psikoloji (1879-)
- Bilgisayar Mühendisliği (1940-)
- Dilbilim (1957-)

# Yapay Zeka'nın (YZ) Tarihi (0)

- Yunan Mitolojisi : akıllı makineler, mekanik araçlar ve yapay zeka.
- Abbasiler döneminde (750-1256), 9. yy'da hidrolik prensiplere dayalı otomatik makineler geliştirildi. İlk otomat sistemlerden sonra, bu çalışmalara Selçuklular döneminde “Ebul-İz” devam etmiştir.
- Rönesans döneminde otomatik makineler konusundaki çalışmalara yenileri eklenmiştir (Leonardo da Vinci)
- Pascal, ilk hesap makinesini geliştirmiştir.
- Babbage, ilk programlanabilir bilgisayarı geliştirmiştir (19. yy).
- Sembolik Mantık çalışmaları (Boole, Frege, Russel, Whitehead).
- Turing'in hesaplanabilirlik teorisi (İlk zeki sistemler çalışmaları).
- 1940 : Siberetik (İnsan-makine)

# Yapay Zeka'nın (YZ) Tarihi (1)

- 2. Dünya savaşı sonrası : Asıl gelişmeler, modern bilgisayarların ortaya çıkmasından sonra yaşandı.
- Marvin Minsky, ilk NN bilgisayarı 1951'de yaptı. (Jüri von Neumann)
- 1956 : İlk olarak YZ deyiimi kullanıldı. YZ konusunda düzenlenmiş ilk konferans olan Dartmouth Konferansında, John McCarthy “yapay zeka” terimini türetti.
- Allen Newell, J.C. Shaw ve Herbert Simon'un yazdığı Mantık Kuramcısı (The Logic Theorist) adlı ilk YZ programı tanıtıldı.
- 1957 : Aynı kişiler General Problem Solver'ı yazdılar.
- 1952-1962 : IBM'den Arthur Samuel, satranç oynayabilen ilk programı yazdı. (Daha önce 1950'de Shannon, sonra Turing)
- 1958 : MIT'den John McCarthy, LISP dilini geliştirdi.

## Yapay Zeka'nın (YZ) Tarihi (2)

- 1961 : LISP'te Üniversite 1. Sınıf düzeyindeki matematik problemlerini çözebilen bir program olan Saint (Aziz)'i yazıldı.
- 1962 : İlk endüstriyel robot şirketi kuruldu.
- 1963 : MIT'de, IQ sorularını çözebilen Analogy programı yazıldı. Ivan Sutherland, bilgisayarlarda etkileşimli grafik kullanımını başlattı. Edward A. Feigenbaum ve Julian Feldman, YZ konusundaki ilk makale olan “Bilgisayarlar ve Düşünce'yi” yayınladı.
- 1964 : MIT'deki bir araştırma sonuçları, bilgisayarların doğal dili, basit matematik problemleri çözebilecek kadar anladığını gösterdi.
- 1965 : Joseph Weizenbaum, herhangi bir konuda ingilizce sohbet edebilen ELIZA'yı geliştirdi. Bu programın psikoterapist sürümü, oldukça popüler bir oyuncak haline geldi.

## Yapay Zeka'nın (YZ) Tarihi (3)

- 1968 : Marvin Minsky ve Seymour Papert, sinir ağlarının sınırları konusunda bir makale yayınladı. Sonraki yıl, “Perceptrons” adlı kitabı yayınladılar.
- 1969 : YZ konusundaki ilk uluslararası konferans düzenlendi.
- 1970 : Bilgisayar Destekli Öğretim programı.
- 1971 : İngilizce söylenen komutları yerine getirebilen robot kol.
- 1975 : Öğrenme yeteneğine sahip bir programın bulduğu sonuçların bilimsel dergilerde yayımlanmasının ilk örneği.
- 1979 : Uzman sistemler geliştirilmeye başlandı. Pittsburgh Üniversitesinde ilk iyileştirici program Internist (Stajer) geliştirildi.
- 1980 : Uzman Sistemler, ticari alanda kullanılmaya başlandı. Amerika YZ derneği, ilk ulusal YZ konferansını gerçekleştirdi.



## Yapay Zeka'nın (YZ) Tarihi (4)

- 1984 : Yapay Sinir Ağları yaklaşımı ortaya çıktı
- 1987 : Marvin Minsky, zihnin teorik tanımlamasını yapan “Toplumun Zihni” adlı kitabı yayınladı.
- 1997 : Deep Blue adlı satranç programı, dünya satranç şampiyonu Garry Kasparov’u yendi.
- 1998 : İnternet’in yaygınlaşması ile, YZ tabanlı birçok program geniş kitlelere ulaştı.
- 2000 : Etkileşimli robot oyuncaklar piyasaya sürüldü.

# Yorumlar

- Kolay bir çalışma alanı mıdır?
- Yapay zeka alanı, kendinden beklenenleri gerçekleştirebilmiş midir?
- Yapay zeka günümüzde neleri gerçekleştirebilmektedir?
- Olması gereken nedir?
- Yapay zekanın değişik tanımları?
- Bir programı, yapay zeka olarak kabul etmek için gerekli olan özellikler nelerdir?
- Yapay Zekanın geleceği? (İstenenler, istenmeyenler)
  - Makinelerin kendi bilinçleri olacak mı?
  - Akıllı makineler hayatın içine ne kadar girecek?
  - İnsan beyni modellenilecek, insanın makine kopyası yapılabilecek mi?
  - Makineler insanlığı geçebilecek mi? Bir arı simülatörü yapılabilecek mi?

# Yapay Zeka'nın Amaçları

- Temel amaç, insanların zor yaptığı işleri yapabilecek sistemler üretmek.
- İnsan beyninin fonksiyonlarını, bilgisayar modelleri yardımıyla anlamaya çalışmak.
- İnsanın bilgi kazanma, öğrenme ve buluş yapma gibi zihinsel yeteneklerini araştırmak.
- Öğrenme metotlarını bilgisayar sistemlerine aktarmak.
- İnsan bilgisayar iletişimini kolaylaştıran kullanıcı arabirimleri geliştirmek.
- Yapay uzman sistemler oluşturmak.
- Yapay zekaya sahip robotlar geliştirmek (İşbirliği)
- Bilgisayarları, bilimsel araştırma ve buluşlarda kullanmak.

# YZ'nın Araştırma ve Çalışma Alanları

- Oyunlar (satranç, dama, strateji, diğer...)
- Yapay Yaşam (Hayat)
- Teorem İspatlama (Prolog, Paralel Prolog, Cebrik Mantık Prog.)
- Doğal Dil Anlama ve Çeviri
- Bilgi Tabanlı Sistemler (Bilgi gösterimi, uzman sistemler, bilgi tabanlı simülasyon, genel bilgi sistemleri -CYC, EDR- )
- Makine Öğrenmesi (Bilgi düzeyinde öğrenme, sembol düzeyinde öğrenme, aygıt düzeyinde öğrenme)
- Makine Buluşları (Bilgi Madenciliği, Bilimsel buluşların Modellendirilmesi)
- Robotik (Görev planlama, robot görmesi)
- Şekil Tanıma (Nesne tanıma, Optik Harf Tanıma (OCR), Ses Tanıma)

# Yapay Zeka ile Yapılanlar

- AI, hem sıradan, hem de uzmanlık gerektiren işlerin otomatikleştirilmesi ile ilgilidir. Bir hastalık tedavisinin belirlenmesi için bir program rahatlıkla yazılsa da, iki yaşındaki bir çocuğun yaptığı işler, günümüzde AI araştırmalarının sınırında veya ötesindedir (yüz tanıma, iletişim kurma, ...). Günümüzde YZ'da sınırlı amaçlarla mutlu olunmaktadır :
- Bilgisayarlara belirli işleri yaptırma
- Sınırlı ses tanıma
- Evde veya diğer alanlarda “Akıllı Yardımcılar”
- İnternet'teki basit YZ örnekleri

# Günümüzde YZ Yaklaşımları

- Yapay Sinir Ağları
- Genetik Algoritmalar
- Yapay Zeka'nın İnternet üzerindeki uygulamaları yeni bir alan.
  - Mobile Agent Systems
    - İnternet üzerinde, istenilen konudaki yazıları okuyarak, istenilen araştırma yazısının hazırlanması.
    - İstenilen fiyat ve özellikte ürünün bulunması (E-ticaret)
  - İnternet üzerinde YZ'lı karakterlerle sohbet etmek
    - 1965'te Joseph Weizenbaum E.L.I.Z.A (Bugün A.L.I.C.E)

# Bilgisayar Oyunlarında YZ

- 2D'den 3D'ye. Gerçekçi grafikler.
- Çok gerçekçi karakterler.
- Karakterlerin yürüyüş gibi hareketleri gerçek yaşamdaki gibi.
- İnsanların, kendileri gibi düşünen karakterlerle etkileşim kurmaya yönelimi var.
- Ateş edip puan toplayarak oynanan oyunlar yerine, yapay zeka örnekleri kullanılarak gerçekleştirilen oyunlar istenmeye başlandı (Karakterlerle etkileşim, konuşma, plan yapan karakterler)
- Oyunlarda yapay zeka, monotonluğu aşmayı sağlıyor.

Diğer Alanlar

# Sinemada Bilgisayarlar ve YZ (1)

- 1907'den itibaren bu alanın ilk örneklerinde konu : Kendilerini yapanlara hizmet etmek için tasarlanmış mekanik adamlar kontrolden çıkarak insanlar için tehdit unsuru haline geliyorlardı.
- 1950'lerde Hollywood'un teknolojiye bakışı oldukça olumlu ve iyimserdi. İyi kalpli bilgisayarlar veya YZ bilgisinin yanlış kişilerce ele geçirilmesi konu edildi.
- 1960'larda kötümser bir hava oluştu. Yapay zeka örnekleri nükleer gücü ele geçirirler. İnsanlar yapay hale gelmiştir.
- 1970'lerde korku, yerini bilgisayarların günlük hayattaki tehlikelerine bırakır.
- 1977'de Yıldız Savaşları, 1950'lerin teknoloji taraftarlığını geri getirir.



## Sinemada Bilgisayarlar ve YZ (2)

- 1980'lerdeki filmlerdeki YZ örneklerinin birincil amacı, insanlığı yok etmektir (Terminatör-1984 gibi). Sonraları bilgisayarlar komedi unsuru olmuş, günlük yaşamın parçası haline gelmiş, daha az korkulur olmuştur.
- 1990'larda İnternet'in yaygınlaşmasının sinemada etkileri görülür. Bir tuşla kişinin kimliğini yok etme, sanal dünyalar gibi.
- 2001 yılında Steven Spielberg'in Yapay Zeka filminde ise, sorun yaşayan, bir çocuk robottur.

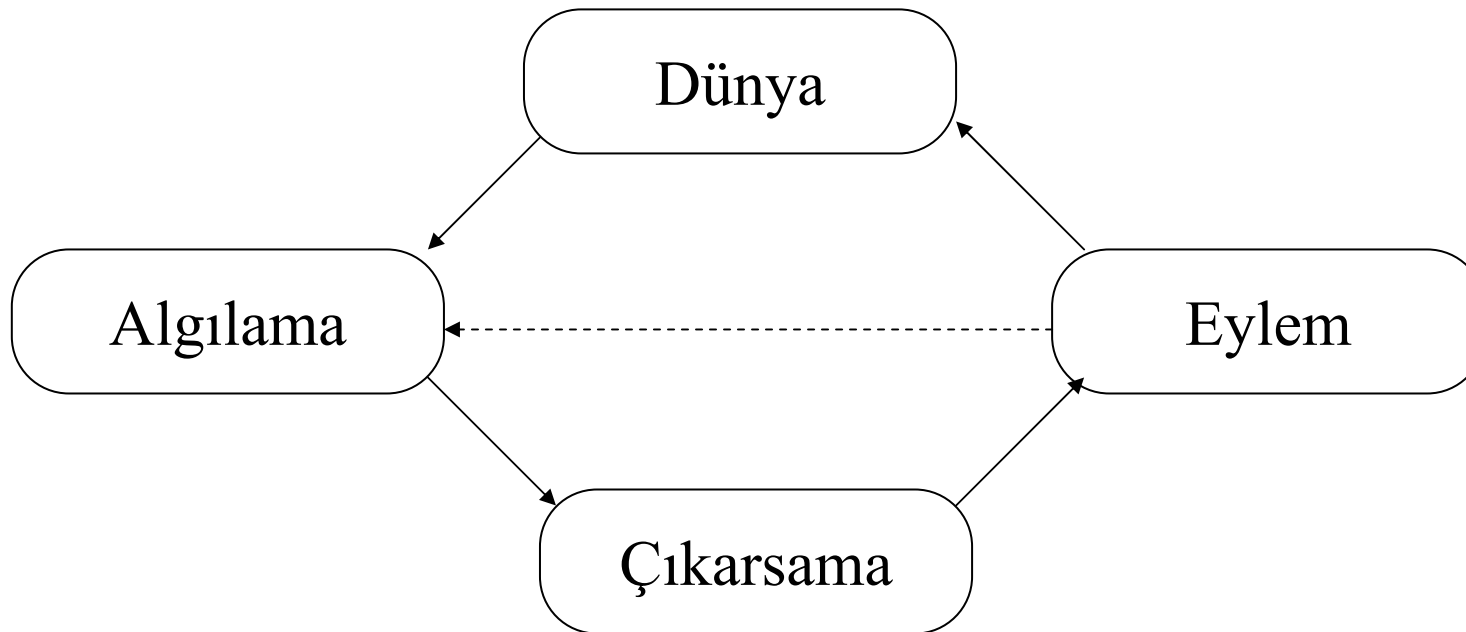
Tamamıyla bilgisayar tabanlı etmenler (robot, bilgisayar)

Humanoid Robot (Kısmen insana benzetilmiş)

Android (Genelde biyolojik, çıkarsaması insan beynininkine benzer)

Cyborg (İnsan+ek)

# Günümüzde AI nedir?



# Örnek AI Sistemi

## 1) Satranç Oynama (Chess Playing)

Deep Blue (IBM) gibi.

- Algılama : Satranç tahtasının ileri özellikleri
- Eylemler : Hareket seçme
- Çıkarsama : Tahta konumlarını değerlendirme sezgisi ve arama.

# Örnek AI Sistemi

## 2) Tıbbi Teşhis (Medical Diagnosis)

Pathfinder (D. Heckerman, Microsoft Research) gibi.

- Algılama : Belirtiler, test sonuçları
- Eylemler : Test önerme, teşhis etme
- Çıkarsama : “Bayesian Inference”, makine öğrenmesi, Monte Carlo simülasyonu.

## Örnek AI Sistemi

3) Kendi Başına Giden Araba (Car driving itself)  
ALVINN (D. Pomerleau, CMU) gibi.

- Algılama :Yolun sayısal kamera görüntüsü
- Eylemler : 64 farklı direksiyon açısı
- Çıkarsama : “Back propagation” eğitilmiş yapay sinir ağı.

# Klasik AI ve Modern AI

- Klasik AI'da, çıkarsama en önemli AI problemi ve Genel Problem Çözücü, altın hedefti.
- Modern AI'da, olasılık, istatistik, karar teorisi ve matematik yoğun olarak kullanılmaktadır. Genel çözücü yerine, özel problemleri çözmeye çalışmaktadır. Yeni alanlar ortaya çıkmıştır.

# PROLOG (Örnek 1)

## Örnek 1 : Basit bir Prolog Örneği

### PREDICATES

nondeterm hoslanir(symbol, symbol).

### CLAUSES

hoslanir(ali,okuma).

hoslanir(ali,film).

### GOAL

write("Ali'nin Hobileri :"),nl,

hoslanir(ali,Ne).

Ekran Çıktısı :  
Ali'nin Hobileri :  
Ne=okuma  
Ne=film  
2 Solutions

# PROLOG (Örnek 2)

## Örnek 2 : Kişilerin Hobileri

DOMAINs

person, activity = symbol

PREDICATES

nondeterm hoslanir(person, activity).

CLAUSES

hoslanir(ali,futbol).

hoslanir(tolga,basketbol).

hoslanir(can,yüzme). hoslanir(canan,tenis).

hoslanir(yesim,Hobi) if hoslanir(tolga,Hobi).

GOAL

hoslanir(yesim,futbol).

GOAL

hoslanir(Kisi,futbol).

Ekran Çıktısı :

Kisi=ali

1 Solution

Ekran Çıktısı :

no



# PROLOG (Örnek 3)

## Örnek 3 : Otomobil Sorgulamaları

### DOMAINs

marka,renk=symbol

yas, fiyat=integer

yol=real

### PREDICATES

araba(marka,yol,yas,renk,fiyat).

### CLAUSES

araba(renault,100,5,gri,20).

araba(opel,20.5,1,mavi,35).

araba(kartal,120,15,siyah,10).

### GOAL

araba(Marka,\_,5,\_,\_).

### Ekran Çıktısı :

Marka=renault

1 Solution

### GOAL

araba(Marka,\_,Yas,Renk,\_) and Yas < 7.

Marka=renault, Yas=5, Renk=gri

Marka=opel, Yas=1, Renk=mavi

2 Solutions

# PROLOG (Örnek 4)

## Örnek 4 : Futbol Turnuvası

### DOMAINS

ulkead, grubu = symbol

### PREDICATES

nondeterm grup(ulkead,grubu).

### CLAUSES

grup(almanya,a).

grup(brezilya,a).

grup(türkiye,a).

grup(kore,a).

grup(çin,b).

grup(amerika,b).

### GOAL

grup(Ulke1,a) and grup(Ulke2,a) and Ulke1  $\neq$  Ulke2.

### Ekran Çıktısı :

Ulke1=almanya, Ulke2=brezilya

Ulke1=almanya, Ulke2=türkiye

Ulke1=almanya, Ulke2=kore

Ulke1=brezilya, Ulke2=almanya

Ulke1=brezilya, Ulke2=türkiye

Ulke1=brezilya, Ulke2=kore

Ulke1=türkiye, Ulke2=almanya

Ulke1=türkiye, Ulke2=brezilya

Ulke1=türkiye, Ulke2=kore

Ulke1=kore, Ulke2=almanya

Ulke1=kore, Ulke2=brezilya

Ulke1=kore, Ulke2=türkiye

12 Solutions

# PROLOG (Örnek 5)

## Örnek 5 : Akrabalık Çıkarsamaları

### PREDICATES

```
nondeterm anne(symbol,symbol).  
nondeterm baba(symbol,symbol).  
nondeterm dede(symbol,symbol).
```

### CLAUSES

```
anne(sue,joe).  
baba(eric,joe).  
baba(john,sue).  
baba(bill,eric).  
baba(george,bill).  
dede(De,T) :- baba(De,G), baba(G,T).  
dede(De,T) if baba(De,G) and anne(G,T).
```

### GOAL

```
dede(Dede,Torun).
```

### Ekran Çıktısı :

```
Dede=bill, Torun=joe  
Dede=george, Torun=eric  
Dede=john, Torun=joe  
3 Solutions
```

# PROLOG (Örnek 6)

## Unification

```
% Örnek 6
/* Uzun Roman
Yazarları */
```

## DOMAINS

```
title, author = symbol
pages = unsigned
```

## PREDICATES

```
book(title,pages).
nondeterm written_by(author,title).
nondeterm long_novelist(author).
```

## CLAUSES

```
written_by(fleming,"DR NO").
written_by(melville,"MOBY DICK").
book("MOBY DICK", 250).
book("DR NO", 310).
```

```
long_novelist(Writer) :-
    written_by(Writer,Title),
    book(Title,Length), Length>300.
```

## GOAL

```
long_novelist(Writer).
```

Writer=fleming 1 Solution
------------------------------

# PROLOG (Örnek 7)

## Listeler

### DOMAINS

```
namelist = name*  
name = symbol
```

### PREDICATES

```
nondeterm member(name,namelist).
```

### CLAUSES

```
member(Name, [Name|_]).  
member(Name, [_|Tail]):-  
    member(Name,Tail).
```

### GOAL

```
member(X,[ali,veli,cemil]).
```

Ekran Çıktısı :  
X=ali  
X=veli  
X=cemil  
3 Solutions

# PROLOG (Örnek 8)

## DOMAINS

**name,address = string**  
**age=integer**  
**list=age\***

## PREDICATES

**nondeterm person(name, address,**  
**age)**  
**sumlist(list,age,integer)**

## CLAUSES

**sumlist([],0,0).**  
**sumlist([H|T],Sum,N):-**  
**sumlist(T,S1,N1),**  
**Sum=H+S1, N=1+N1.**

**person("Sherlock Holmes", "22B Baker Street", 42).**  
**person("Pete Spiers", "Apt. 22, 21st Street", 36).**  
**person("Mary Darrow", "Suite 2, Omega Home", 51).**

## GOAL

**findall(Age,person(\_,\_,Age),L),**  
**sumlist(L,Sum,N),**  
**Ave=Sum/N,**  
**write("Average=",Ave),nl.**

Ekran Çıktısı :  
Average=43  
L=[42,36,51], Sum=129, N=3, Ave=43  
1 Solution

# PROLOG (Örnek 9)

**Döngüler : fail**

**PREDICATES**

**nondeterm country(symbol)**

**print\_countries**

**CLAUSES**

**country("England").**

**country("France").**

**country("Germany").**

**country("Denmark").**

**print\_countries:-**

**country(X),**

**write(X), nl, fail.**

**GOAL**

**print\_countries.**

Ekran Çıktısı :

England

France

Germany

Denmark

no

# PROLOG (Örnek 10)

**Faktöryel, recursion, Cut(!).**

**PREDICATES**

**factorial(unsigned,real)**

**CLAUSES**

**factorial(1,1):-!.**

**factorial(X,FactX):-**

**Y=X-1,**

**factorial(Y,FactY),**

**FactX = X\*FactY.**

**GOAL**

**factorial(3,X).**

Ekran Çıktısı :

X=6

1 Solution



# PROLOG (Örnek 11)

**% Kurallar (case gibi), Cut(!)**

## **PREDICATES**

**action(integer)**

## **CLAUSES**

```
action(1):-!,  
    nl,  
    write("You typed 1.").  
action(2):-!,  
    nl,  
    write("You typed 2.").  
action(3):-!,  
    nl,  
    write("You typed 3.").  
action(_):-  
    write("Other").
```

## **GOAL**

```
write("Type a number from 1 to 3 : "),  
readint(Num),  
action(Num),  
nl, write("Type a String"), readln(Str),  
write(Str),nl.
```

```
Ekran Çıktısı :  
Type a number from 1 to 3 : 3  
  
You typed 3.  
Type a Stringfff  
fff  
Num=3, Str=fff  
1 Solution
```

# PROLOG (Örnek 12)

**Karekök 5**

**PREDICATES**

**run**

**CLAUSES**

**run:-**

**A=sqrt(5),**

**write(A).**

**GOAL**

**run.**

Ekran Çıktısı :  
2.236067977yes

# PROLOG (Örnek 13)

**GOAL**

**A=sqrt(5).**

Ekran Çıktısı :  
A=2.236067977  
1 Solution

## 2. Etmenler (Agents)

Agent, algılayan ve eylemde bulunan bir varlıktır.

Kısaca bir agent, eski algılardan eylemlere bir fonksiyondur.

$$f: P^* \rightarrow A$$

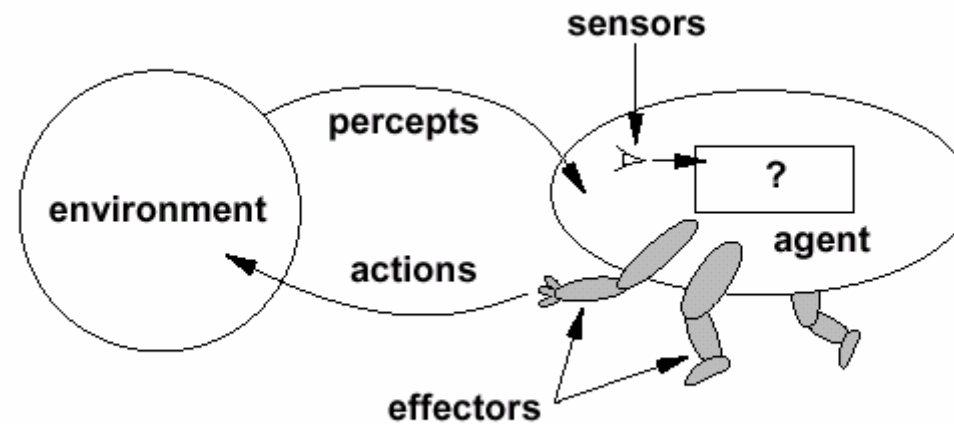
Verilen şart ve işler için, en iyi performansı veren etmen aranır.

Uyarı : Bilgisayar sınırlamalarından dolayı ideal duruma ulaşamaz.

-> Verilen makine kaynakları ile yapılabilecek en iyi programın tasarımı

# Etmen (Agent) nedir?

Etmen veya temsilci (Agent), “Sensor” leri yardımıyla çevreyi algılayan ve “effector” leri yardımıyla eylemde bulunan şeylere verilen isimdir.



# Agent Tasarımı

Dört önemli elemandan oluşur :

- Percepts : Agent'ın aldığı bilgiler
- Actions : Agent'ın yaptıkları
- Goals : Agent'ın ulaşmaya çalıştığı
  - İstenen Durum
- Environment : Agent'ın hareket ettiği yer, ortam.

## Örnek : Otomatik Taksi

- Percepts : Video, motor algılayıcıları, GPS, hızölçer
- Actions : Direksiyon kullanma, hızlanma, fren yapma, korna çalma
- Goals : Emniyetli sürüş, hedefe varma, minimum gaz tüketimi
- Environment : Yollar, trafik, hava, ülke

Örnek : İnternet shopbot

## “Environment” türleri

- Accessible vs Inaccessible : Agent, ortamın durumunu görebiliyor mu?
- Deterministic vs. Stochastic : Aynı koşullarda yapılan aynı hareket hep aynı sonucu veriyor mu?
- Episodic vs. Infinite : İş, bilinen zaman aralığında bitirilebiliyor mu?
- Static vs. Dynamic : Agent düşünürken, ortamda değişiklik oluyor mu?
- Discrete vs. Continuous : Satranç X Robotlar



# AI neden zordur?

- Gerçek ortamlar genelde, “inaccessible”, “stochastic”, “dynamic” ve “continuous” ’dur.
- “Accessible”, “deterministic”, “static”, “discrete” ortamlar bile çok geniştir. (Satranç örneği)
- Sınırlı kaynaklar (bellek, zaman, yetersiz bilgi)

Agent tasarımında “environment” türünün büyük önemi vardır.

# Agent'lar ne yapar?

Agent'lar etkinlikleri seçerler.

## **Agent Çatısı :**

- Input : percept
- Output : action
- Variables : internal memory

## **Algoritma :**

1. Update-memory(percept)
2. action = Choose-best-action(memory)
3. Update-memory(action)
4. return action

# Mantıklı (Rational) Agents

- Agent'lar, mantıklı davranmak zorundadırlar.
- Mantıklı hareket : Verilen bilgiler doğrultusunda, performans ölçüsünün beklenen değerini maksimize eden eylem.
- Beklenen değer olmasının nedeni, ortamların deterministik olmamasından. Belirsizlik ve beklenmeyen, istenmeyen durumlar. Mantıklı hareket, başarılı olmayabilir.

Agent türlerine çalışınız.

## 3. Uninformed Search (Bilgiye Dayanmayan Arama)

### **Arama Yöntemleri**

- Uninformed Search
- Informed Search

Problemlerin Çözümünde Aramanın Önemi

Bir problemde hedef belirtilir.

Hedefe ulaşmak için arama yapılır.

# Terminoloji

Initial State

Goal Test

Successor Function

Path Cost

State Space

Solution

Path

# Örnek Problem 1) 8-puzzle Problemi

## Başlangıç, Bitiş Durumu, Durum Uzayı

Durum Uzayı (State Space) Boyutu :  $9!/2 = 181440$

8-puzzle için 181440, 15-puzzle=?, 24-puzzle=?

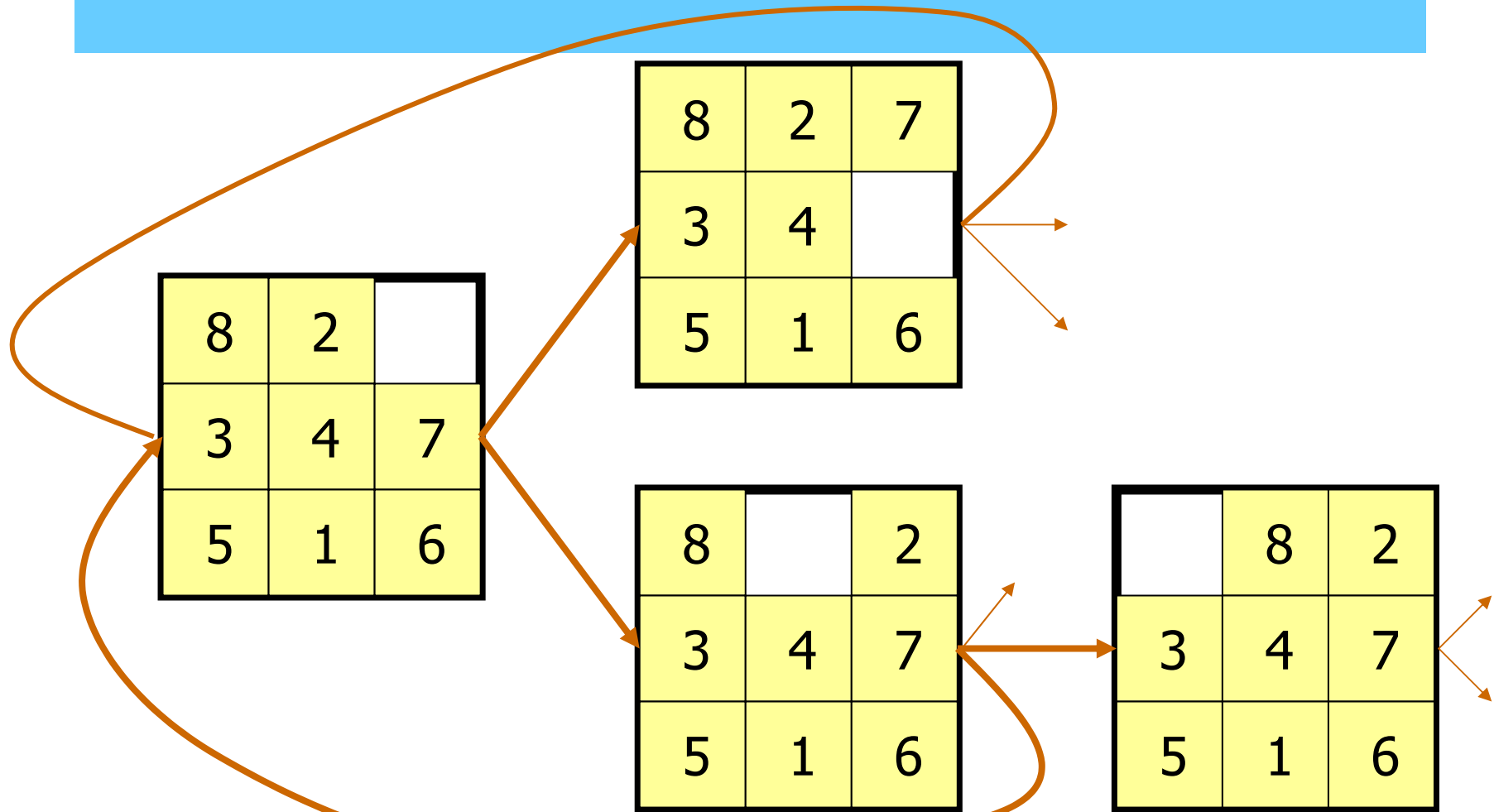
8	2	
3	4	7
5	1	6

Initial state

1	2	3
4	5	6
7	8	

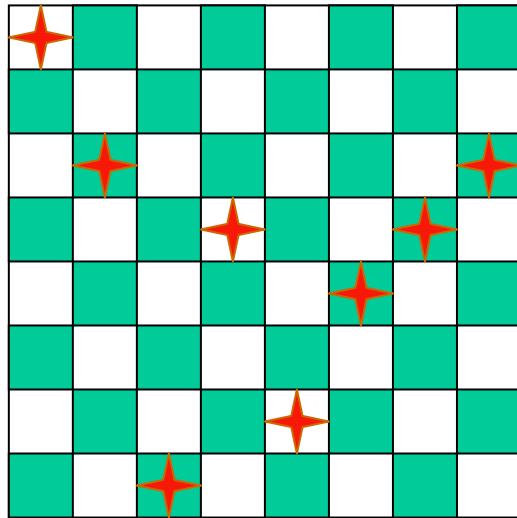
Goal state

# 8-puzzle : Kurallar ve Olası Hareketler



# Örnek Problem 2) 8-queens Problemi

$64^8$  farklı durum var.



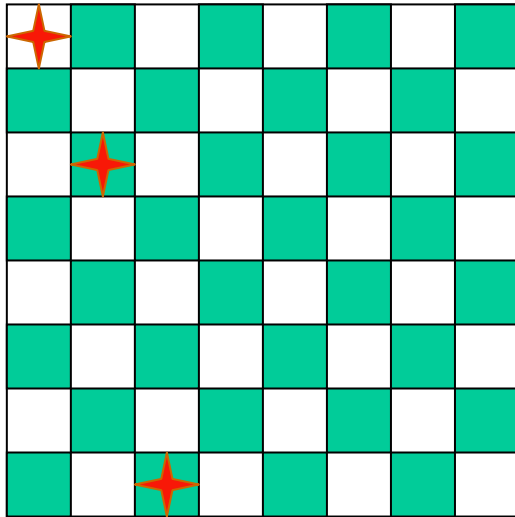
## 1. Yöntem

- Durumlar (States) : 8 vezirin oyun tahtasına herhangi bir şekilde yerleştirilmesi.
- İlk Durum (Initial State) : Tahtada vezir olmadığı durum.
- Sonraki Fonksiyonu (Successor function) : Veziri herhangi bir kareye ekle.
- Hedef Durum (Goal State) : 8 vezirin, tahtaya birbirini alamayacak şekilde yerleştirilmesi



# 8-queens Problemi

2067 farklı durum var.



## 2. Yöntem

- Durumlar :  $k=1..8$ ,  $k$  vezirin, olabilen en sol sütuna, birbirini alamayacak şekilde yerleştirilmesi.
- İlk Durum : Tahtada vezir olmadığı durum.
- Sonraki Fonksiyonu : Bir vezirin, boş olan en soldaki sütundaki bir kareye, diğer vezirler tarafından alınamayacak şekilde yerleştirilmesi.
- Hedef Durum : 8 vezirin tahtada olması.

# Arama Problemleri

- Bir veya daha fazla başlangıç durumu.
- Bir veya daha fazla bitiş durumu (çözüm).
- Çözüm, yol veya hedef düğümdür :
  - 8-puzzle’da yol.
  - 8-queen’de hedef düğüm.

Gerçek Hayat Problemleri :

Robot Navigation, Route Finding, ...

# Önemli Parametreler

- Durum uzayındaki durum sayısı.
- Durumları tutmak için gereken bellek miktarı.

# Sonraki Ders

## Arama Yöntemleri

- Blind Strategies (Bilgiye dayanmayan arama)
- Heuristic Strategies (Sezgisel arama)

# Arama Yöntemleri (Search Strategies)

Her aşamada hangi düğümün açılacağını belirler.

- Un-informed (Blind) Strategies : Durum bilgisinden yararlanmaz.
- Informed (Heuristic) Strategies : Durum bilgisinden yararlanır. Daha umut verici hareketi tercih eder.

# Sezgisel Yöntemlerin Avantajı

8	2	
3	4	7
5	1	6

STATE



N1

Sezgisel yöntemde, yerinde olan kareler sayıldığında N2, N1'den daha umut vericidir.

1	2	3
4	5	
7	8	6

STATE

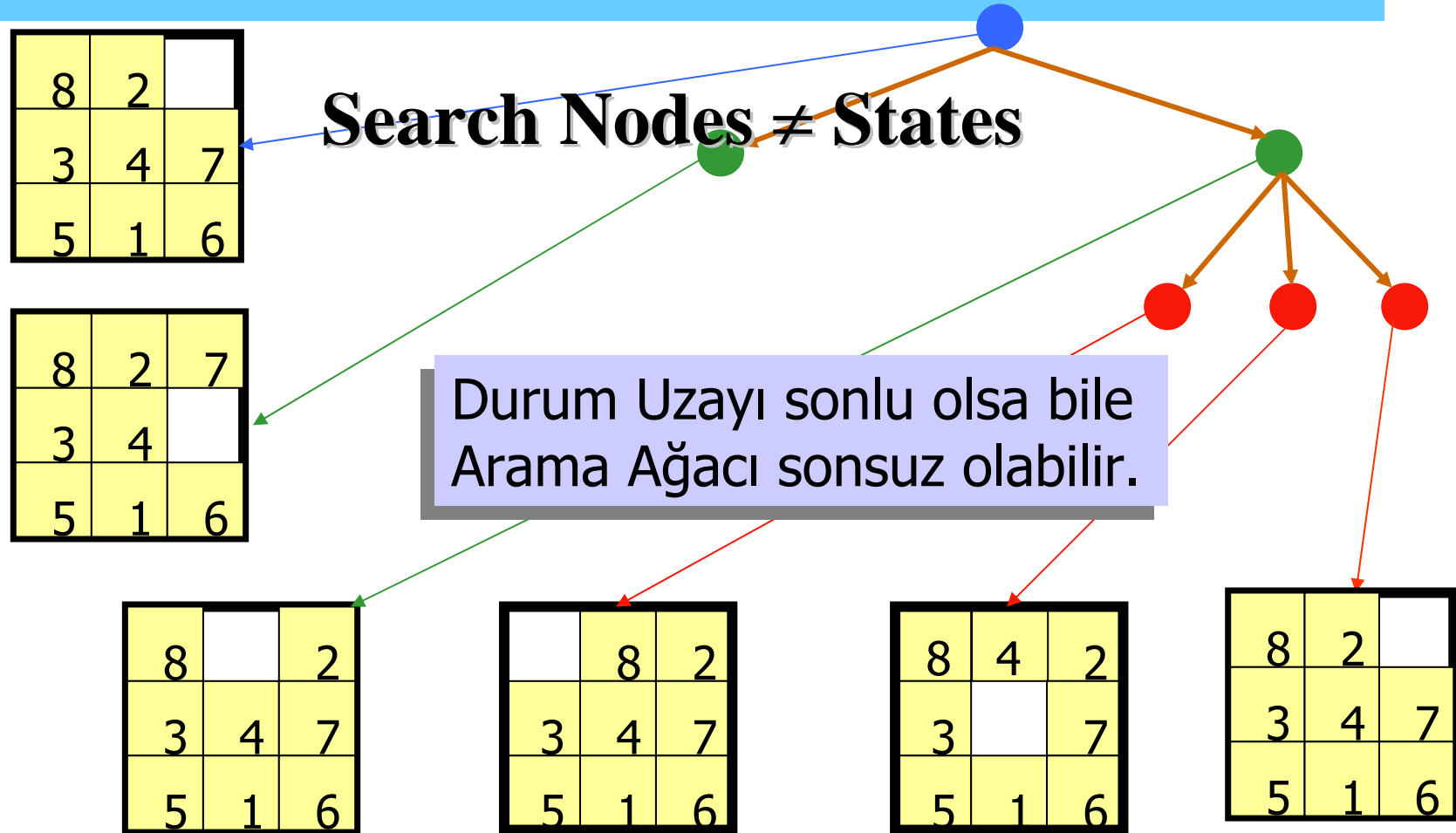


N2

1	2	3
4	5	6
7	8	

Goal state

# 8-puzzle Probleminin Arama Ağacı



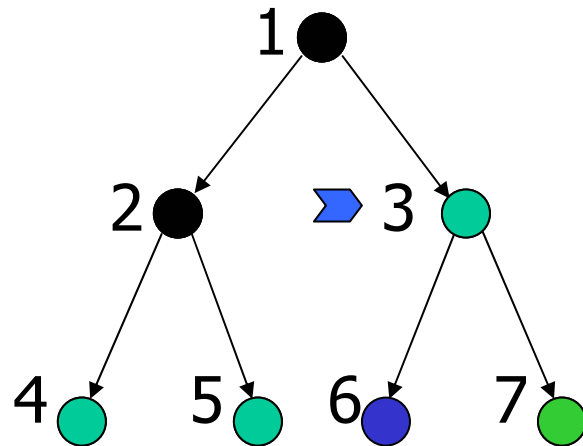
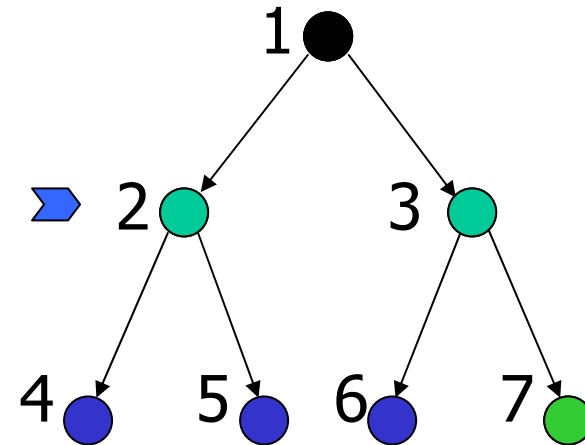
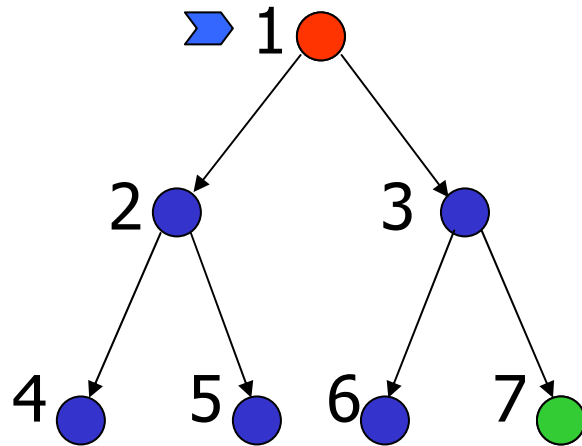
# Sezgisel Olmayan Arama Yöntemleri

- Breadth-First
  - Bidirectional
- Depth-First
  - Depth-limited
  - Iterative deepening
- Uniform-Cost



# Önce Genişliğine Arama

## BFS (Breadth-First Search)

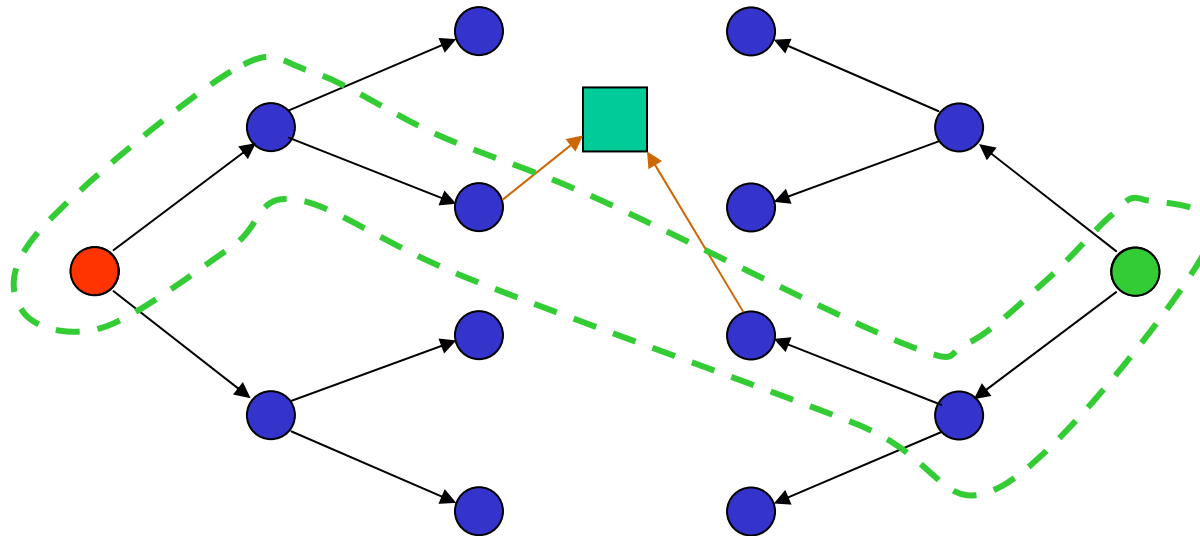


- Düğümleri, kuyruğun sonuna ekler.
- $i$ . Düzeydeki tüm düğümler,  $(i+1)$ . Düzeydeki tüm düğümlerden önce açılır. BFS, hedefe ulaştıran en kısa yolu bulmayı garantiler.

# Çift Yönlü Arama Bidirectional (BF) Search

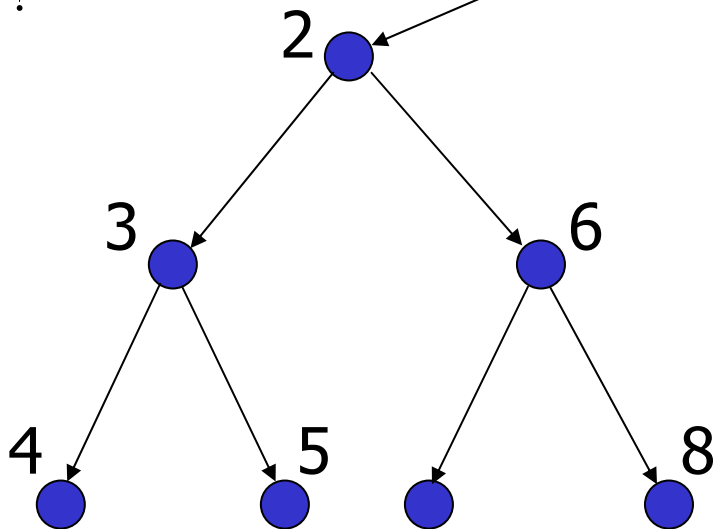
Start ●

● Goal



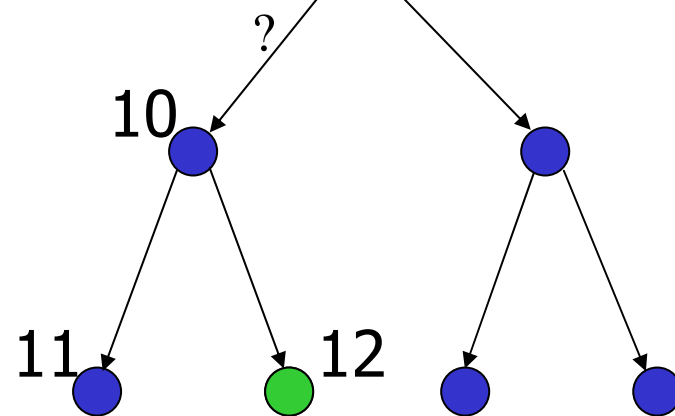
# Önce Derinliğine Arama DFS (Depth-First Search)

✓ DFS, birçok çözüm varsa  
BFS'den etkindir. Kuyruk  
kullanılmasına gerek yoktur.  
?



✗ DFS, tam değildir.?

Çözüm derinliğinin  
büyük olduğunda  
kullanılmamalıdır.



# Derinlik Sınırlandırılmalı Arama

## Depth-Limited Search

k derinliğinde kesilen (cutoff) DFS'dir.

k, altındaki düğümlerin açılmayacağı maximum derinliktir.

Üç olası sonuç vardır :

- Çözüme ulaşılır (Solution)
- Çözüme ulaşamaz (Failure)
- Cutoff (derinlik sınırları içinde sonuca ulaşamaz)

# Yineli Derinleştirmeli Arama

## Iterative Deepening Search

- For  $k=0,1,2,\dots$ :
  - K derinlik sınırı ile DFS yöntemini uygula

DFS ve BFS yöntemlerinin iyi yönlerini birleştirir.

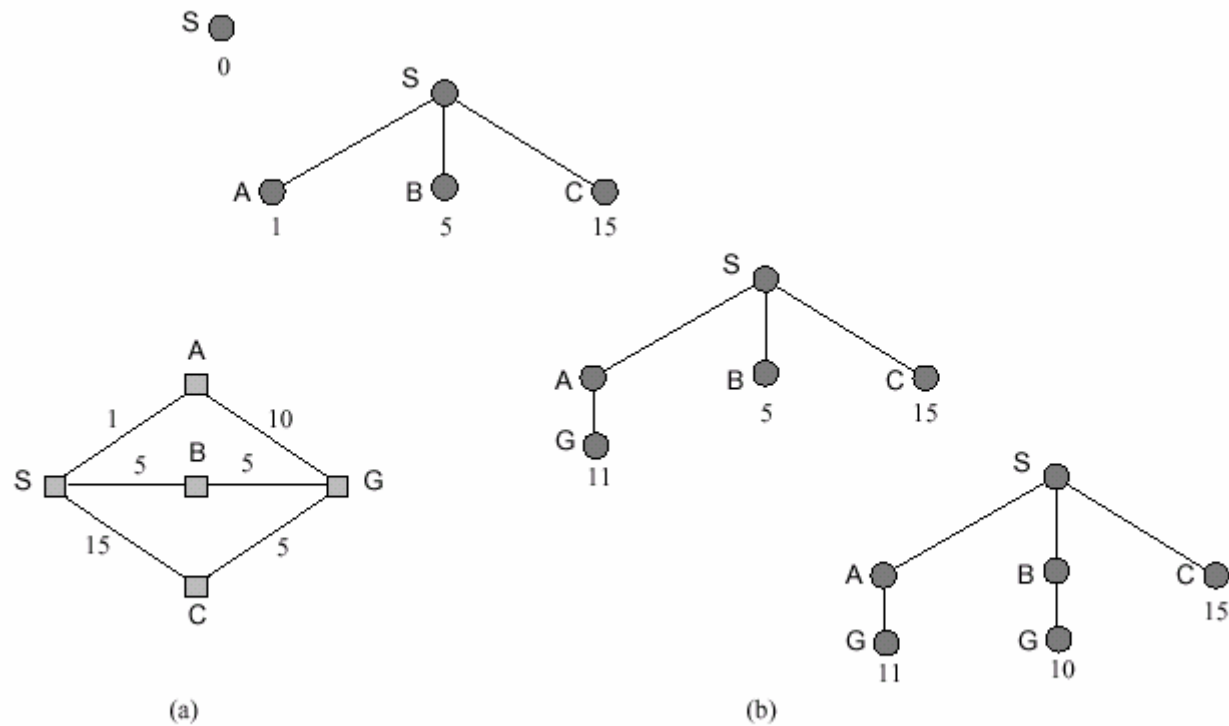
# Sezgisel Olmayan Arama Yöntemleri Karşılaştırma

- BFS, tamdır, optimaldir ancak yer karmaşıklığı yüksektir.
- DFS, yer karmaşıklığı etkindir, ancak tam da değildir, optimal de değildir.
- Yineli derinleştirme, asimptotik olarak optimaldir.

# Uniform Cost Search

- BFS'nin benzeridir.
- Basit kuyruk yerine öncelik kuyruğu kullanır.
- Düğümler, (o ana kadar gelenler içerisinde) artan sırada kuyruğa yerleştirilirler.
- Optimal çözümü bulmayı garantiler.

# Uniform Cost Search Örneği





# Ders Kitabından Okunması Gerekenler

- Tekrarlı Durumların Önlenmesi
- CSP (Constraint Satisfaction Search)

## 4. Heuristic Search (Sezgisel Arama)

- Önceki arama yöntemlerinde düğümlerin açılmasında kullanılan yöntem, başlangıç düğümünden uzaklık bilgisine dayanıyordu.
- Birçok problemde de bilinen budur!
- Hedeften uzaklık tahminlenirse ne olur?
- Hedef biliniyorsa, bu arama olmaz. Sadece gereken düğümler açılır.
- Ama gerçek uzaklık tam olarak bilinmese de tahminlenebilir. Bu tahmine, Heuristic (Sezgi) yani  $h(n)$  denilir.

## Örnek : Rota Planlama

- Bir karayolu sisteminde, bir noktadan diğer noktaya düz hat uzaklığı, uygun bir sezgi ölçüsüdür.
- Her zaman doğru mudur?
- Hayır. Bazı yollar dolambaçlıdır.

# Sezgisel Arama Yöntemleri

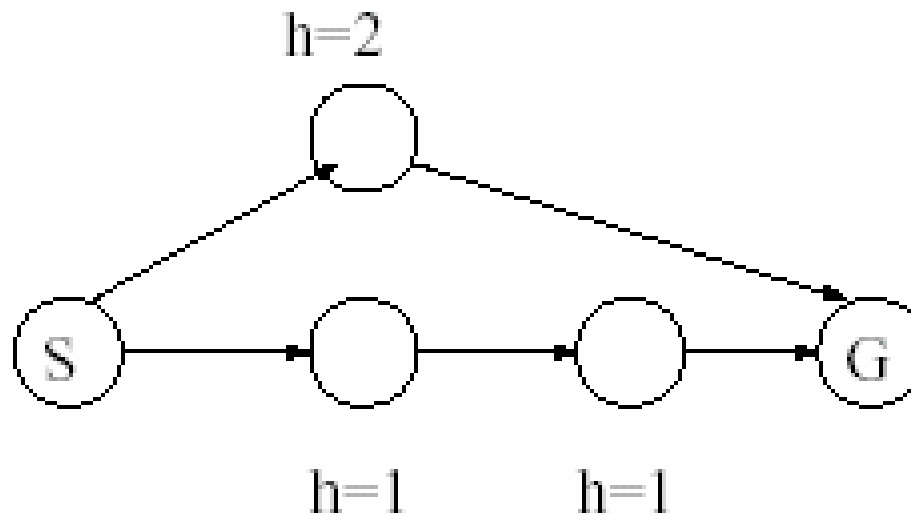
- Best-First Search
- A\* Search
- IDA\* (Iterative Deepening A\*) Search
- SMA\* (Simplified Memory-Bounded A\*) Search

# Best-First Search

- Herhangi bir anda, sezgiye göre en umut verici düğüm açılır.
- Aşağı yukarı uniform-cost search'ün tersidir.
- DFS'ye benzer, ancak sezgi içerir.
- Sezgi, her zaman 0 ise, BFS'ye döner.
- Best-first Search, bir Greedy Yöntemidir.
- Greedy Yöntemleri, uzun vadeli sonuçları önemsemeden, kısa vadeli avantajları en iyi duruma getirir.

# Greedy Yöntemleri

- Tam değildir
- Optimal değildir.
- Zaman ve Yer Karmaşıklıkları kötüdür (üstel)

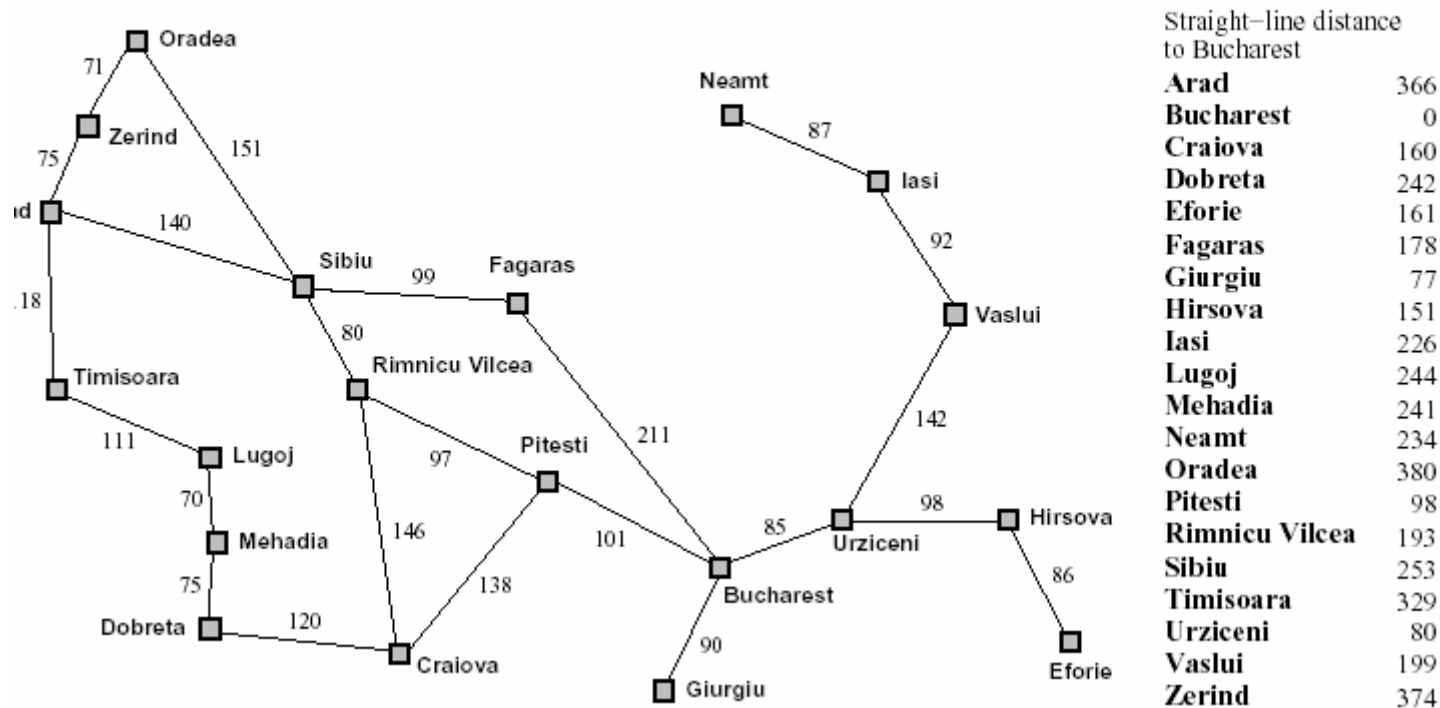


# A\* Search

- Best-first Search'ün en büyük dezavantajı, uzak görüşlü olmamasıdır.
- $g$  : şu anki duruma kadar maliyet ( $c$ ) fonksiyonu
- $h$  : uygun sezgi fonksiyonu olsun.
- Uygun olması, iyimser olması yani hedefe götüren maliyeti hiçbir zaman gerçek değerinden daha fazla tahminlememesi demektir.
- Best-first search, greedy yerine  $f=g+h$  kullanılarak yapıldığında, A\*'a ulaşılır.

# Romanya Haritası (Yol Uzaklıkları ile)

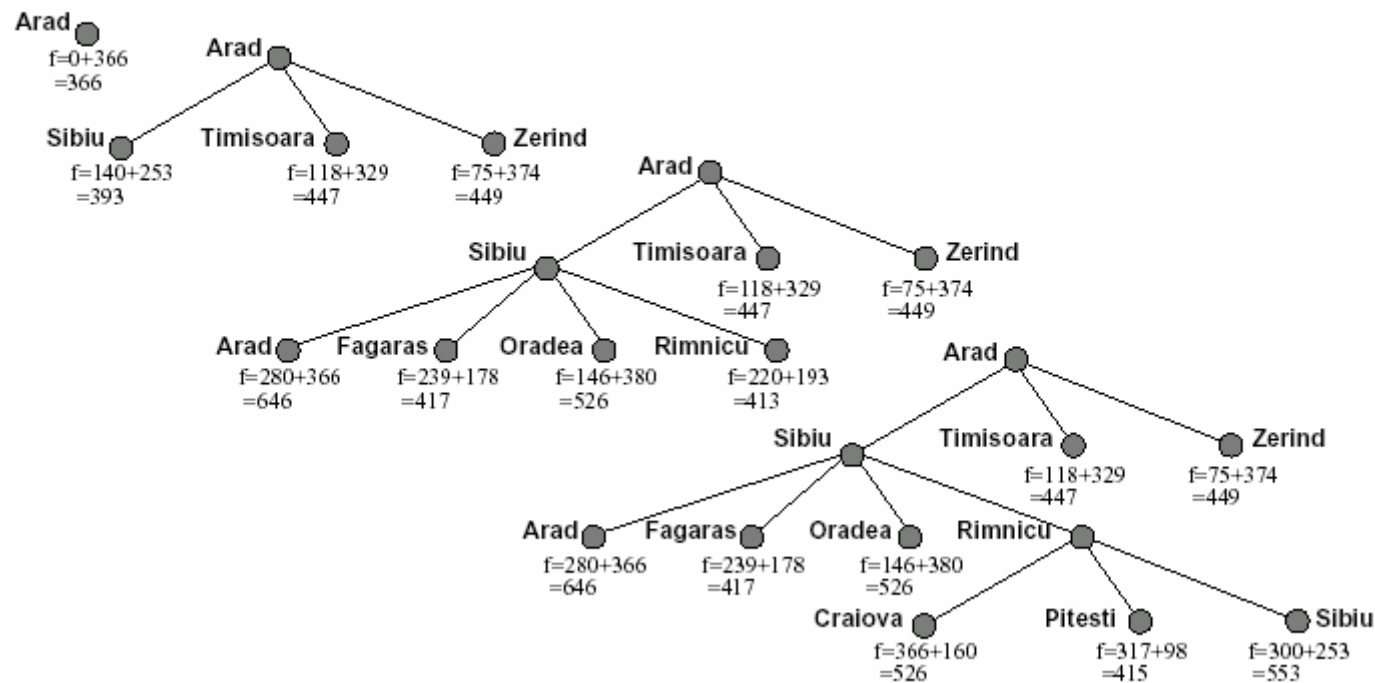
## Example: Romanian navigation





# A\* Aramasının Aşamaları

## Example: Behavior of A\* search

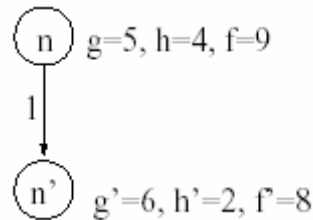


## A\* Aramasının Özellikleri

- Tamdır. Sonuçta, çözüm varsa bulunur. Kökten başlanarak gidilen herhangi bir yolda f tahmini her zaman artar (h, monoton olsa bile). Zaman karmaşıklığı Optimaldir.
- Zaman ve Yer karmaşıklıkları kötü olsa da iyi bir sezgi ile, düzelecektir.

# Non-monotone Heuristics

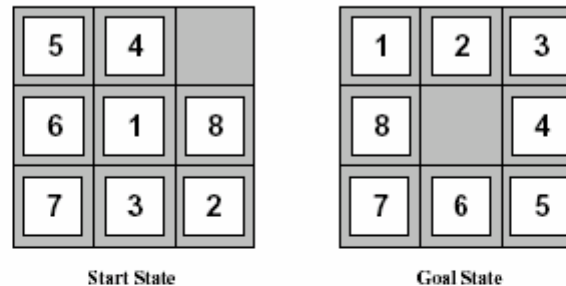
- Bazı sezgilerde  $f$ , yol boyunca düşebilir.  $n'$ ,  $n$  den sonraki düğüm olmak üzere :



- $f(n)=9$  'un anlamı,  $n$  aracılığı ile gıdilebilen yolun maliyeti  $\geq 9$ , ve  $n'$  için de. Basit bir değişiklikle, bu durumun önüne geçilerek,  $F$ 'in yol boyunca azalmaması sağlanır.

$f(n')=g(n')+h(n')$  yerine  $f(n')=\max(g(n')+h(n'), f(n))$   
kullanılır

## Example of admissible heuristics: 8-puzzle



Two possible heuristics:

$h_1(n)$  = number of misplaced tiles = 7

$h_2(n)$  = total Manhattan distance (i.e., no. of squares from desired location of each tile) =  $2+3+3+2+4+2+0+2 = 18$

Intuitively,  $h_2$  seems better: it varies more across the state space, and its estimate is closer to the true cost.

## IDA\* (Iterative-Deepening A\*)

- DFS'ye dayanır. Bir düğümün çocuklarının açılma önceliğini belirlemede f değerinden yararlanır.
- f değeri sınırı vardır (derinlik limiti yerine)
- IDA\*, A\* ile aynı özellikleri taşır ama daha az bellek kullanır.

# Özet

- Sezgisel arama yöntemleri, problem hakkındaki bilgiden yararlanırlar.
- Sezgi (Heuristic), hedefe ulaşmak için kalan maliyetin tahminidir.
- İyi bir sezgi, arama süresini, üstelden doğrusala indirir.
- Best-first Search tam da, optimal de değildir.
- A\*, AI'da anahtar teknolojidir.  $f=g+h$ ,  
(g, harcanan miktar ve h, hedefe ulaşmada harcanması beklenen miktar)

# Diğer Arama Teknikleri

- Gerçek-zamanlı Arama
- Optimizasyon Problemlerinde Arama
- CSP'de Arama

# Gerçek Zamanlı Arama

- Dinamik ortamlarda, arama bitmeden eylemde bulunmak gerekir.
- Hedefe götüren tam yolu aramak yerine, anlık en iyi yolda hareket etmek gerekir.
- Real-time A\*, Korf's Solution? Proposed by Korf in 1990.



# Optimizasyon Problemleri

- Traveling Salesman Problem
- Optimizasyon Problemlerinde Arama
  - Constructive Methods
  - Iterative Improvement/Repair Methods

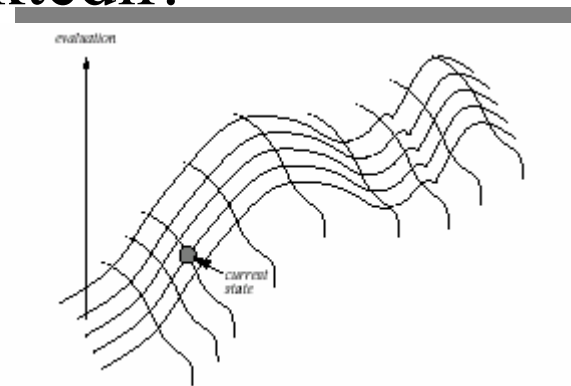
# Yineli Geliştirme Algoritmaları (Iterative Improvement Algorithms)

- Hill Climbing veya Gradient Descent (İrtifa Yokuşu)
- Simulated Annealing

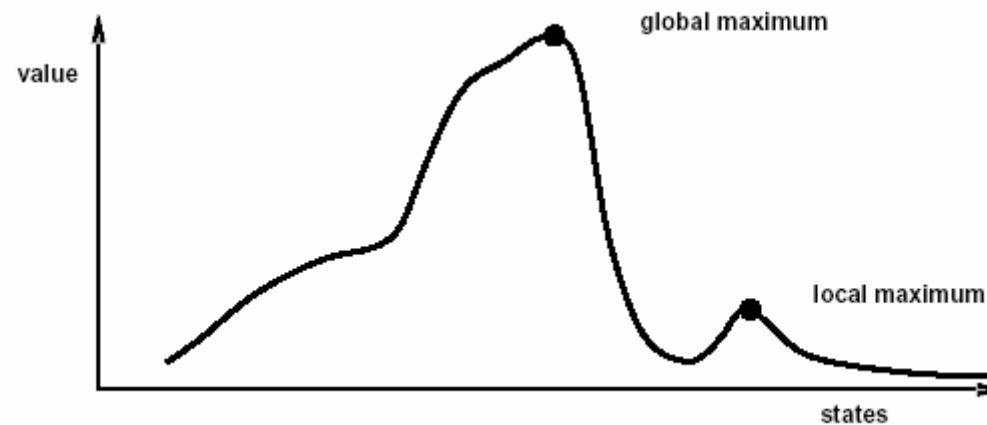
# Hill Climbing

- Yoğun bir siste, Everest Dağına tırmanmaya benzer. Sadece etkin durumun bilgisini tutar.
- Ana düşünce : Her zaman, şimdiki değerlendirmeyi en fazla geliştiren yönde adım at.
- Best-first Search'e benzemektedir.

Öğrenme algoritmalarında popülerdir.



# Hill Climbing : Problemleri



- Local Maximum'da şaşırabilir.
- Yaylada şaşabilir.
- Sıralı tepelerde şaşabilir.

Hızlı Çözüm


Random Restart

# Simulated Annealing

- Ana fikir : Yerel Maksimum'dan kaçmak için, istenmeyen hareketlere izin vermesidir. Yavaş yavaş boyutu ve frekansı azaltılır.
- Random hareket olasılığını kontrol eden T (Temperature) parametresi yeterince yavaş azaltılırsa, en iyi çözüme ulaşılması garantilenir.

# CSP (Constraint Satisfaction Problems) Uygulamaları

- Gerçek CSP'ler
  - Zaman Çizelgesi Problemleri
  - Donanım Konfigürasyonu
  - Nakliye Planlama
  - Fabrika Planlama
  - Kat Planlama



Min-conflicts  
Heuristic

Hill Climbing ve Simulated Annealing, tüm durumlarla çalışır. CSP'lere uygulamada, çığnenen kısıtlamaların sayısı ile Hill Climbing uygun olabilir.

## 5. Game Playing (Oyun Oynama)

- AI'ın en eski ve iyi çalışılmış alanlarındanıdır.
- Nedenleri
  - İnsanlar oyun oynamaktan hoşlanır.
  - Zeka göstergesi kabul edilir.
  - Ortamın net bir tanımı vardır.
  - Sonuçları kolay görülür.
- Oyun Türleri (Satranç, Dama, Tavla, Briç, ...)
  - Perfect vs Imperfect Info (ChessxBridge Hidden Cards)
  - Deterministic vs Stochastic (ChessxBackgammon) ?

# Arama Problemi Olarak Oyun Oynama

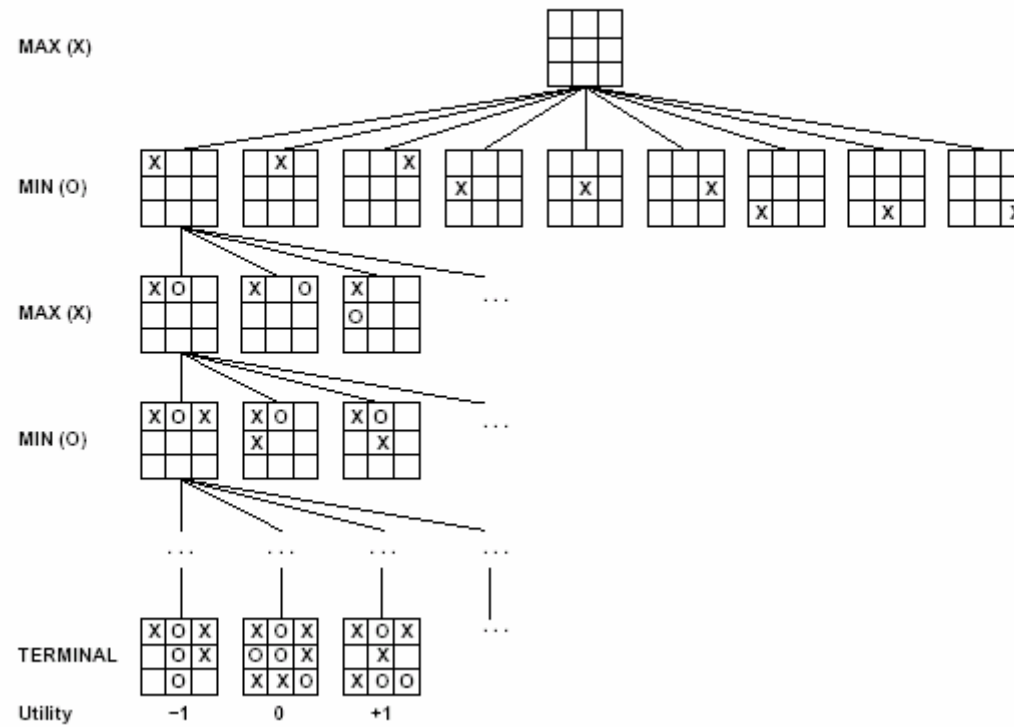
- Tahtanın Durumu
- Geçerli Hareketler
- Uç Durumlar
- Strateji

Rakibin Durumu da düşünülmeli!

- Oyuncunun hareketlerini en iyiye, rakibin hareketlerini en kötüye getiren durumlar.

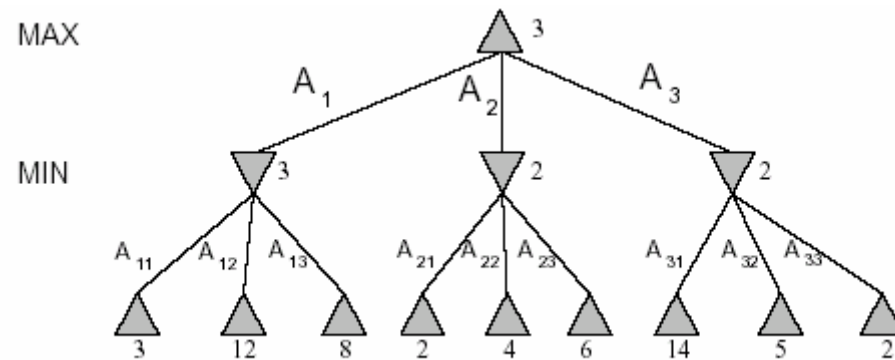


## Example: Tic-Tac-Toe



# Minimax Search

- Uç durumlara ulaşılan kadar tüm arama ağacı açılır.
- Yapraklardan şimdiki duruma doğru geriye dönülür.
- Her min. Düğümünde, çocuk düğümlerin en kötüsü, max düğümün de en iyisi tutulur.
- Oyun ağacı sonlu ise tamdır. Satranç?



# Kaynak Kısıtlamaları ile Başa Çıkmak

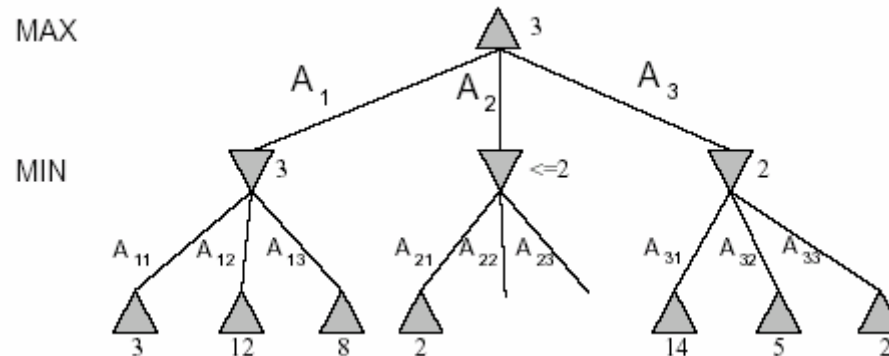
- Hareket için 100 saniyeniz varsa ve saniyede  $10^4$  düğüm dolaşabiliyorsanız, hareket etmeden önce  $10^6$  düğüm dolaşabilirsiniz.
- Standart Yaklaşım
  - Cutoff Test (derinlik sınırlandırmasına dayalı)
  - Evaluation Function (düğüm için, aramanın kesileceği)

# Alpha-Beta Algorithm

- Minimax'a benzer.
- Farklı olarak, arama ağacının sonuçla ilgili olmayan dallarını budar.
- Tüm oyun ağacını açmak (alpha-beta dahil) pek önerilmez. Bir noktada kesilip, ona göre değerlendirilmelidir.
- Şans oyunları da, minimax algoritmasına ek yapılarak gerçekleştirilir.

# Alpha Beta Pruning (Budama)

- Deterministic, Perfect Information Oyunlar için kullanılan standart bir yöntemdir.
- Düşünce Alpha Pruning'e benzer : Bir yol, şimdiki durumdan daha kötüyse elenir.
- Minimax'a da benzer ancak, oyuncunun ve rakibin en iyi yaprak değerinin izini tutar. Örnek :



## Alpha-Beta Algorithm

Instead of MinimaxValue, now we have MaxValue and MinValue

double MaxValue( $s, \alpha, \beta$ )

1. if *cutoff*( $s$ ) return *evaluation*( $s$ )
2. for each  $s' \in \text{Successors}(s)$ 
  - (a)  $\alpha \leftarrow \text{Max}(\alpha, \text{MinValue}(s', \alpha, \beta))$
  - (b) if  $\alpha \geq \beta$  return  $\beta$
3. return  $\alpha$

double MinValue( $s, \alpha, \beta$ )

1. if *cutoff*( $s$ ) return *evaluation*( $s$ )
2. for each  $s' \in \text{Successors}(s)$ 
  - (a)  $\beta \leftarrow \text{Min}(\beta, \text{MaxValue}(s', \alpha, \beta))$
  - (b) if  $\alpha \geq \beta$  return  $\alpha$
3. return  $\beta$

# Alfa Beta Budama'nın Özellikleri

- Budama, sonucu değiştirmez.
- Arama derinliğini, aynı kaynaklarla iki katına çıkarır.
- Satranç gibi oyunlarda,
  - Novice Player (Acemi Oyuncu)
  - Expert Player (Uzman Oyuncu)farkını belirler.

# Özet

- Oyunlar eğlencelidir ve zaman alıcıdır.
- AI'ın birçok önemli yönünü ortaya koyar.
- Mükemmel duruma ulaşamaz  $\Rightarrow$  Yaklaşım
- Grand Prix, otomobil tasarımı için ne ise, Oyunlar da AI için odur.