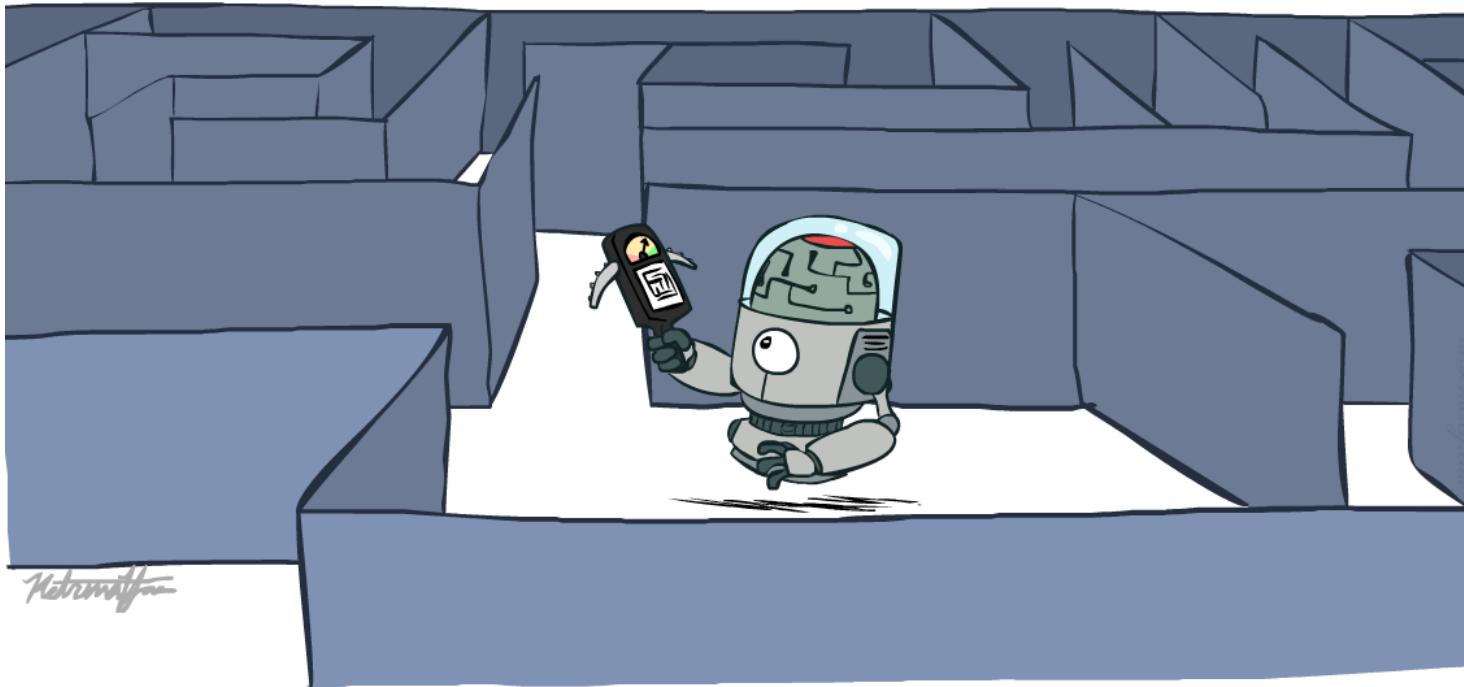


Yapay Zeka

Bilgili Arama



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

[These slides adapted from Dan Klein and Pieter Abbeel]

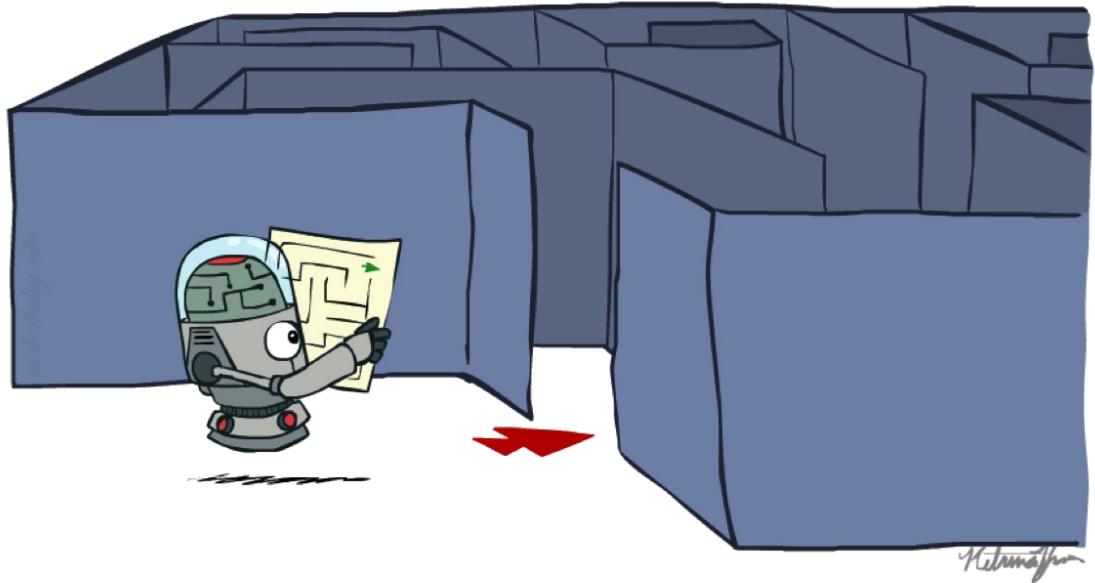
Bugün

- Bilgili Arama
 - En İyi Öncelikli Arama
 - Best-First Search
 - Açı Gözülü Arama
 - Greedy Search
 - A* Algoritması
 - 8 Vezir Problemi
 - Hill Climbing veya Gradient Descent



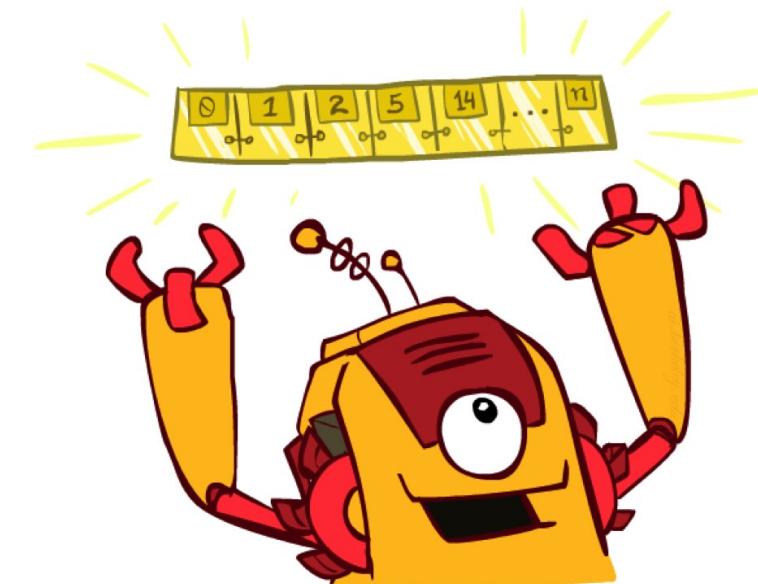
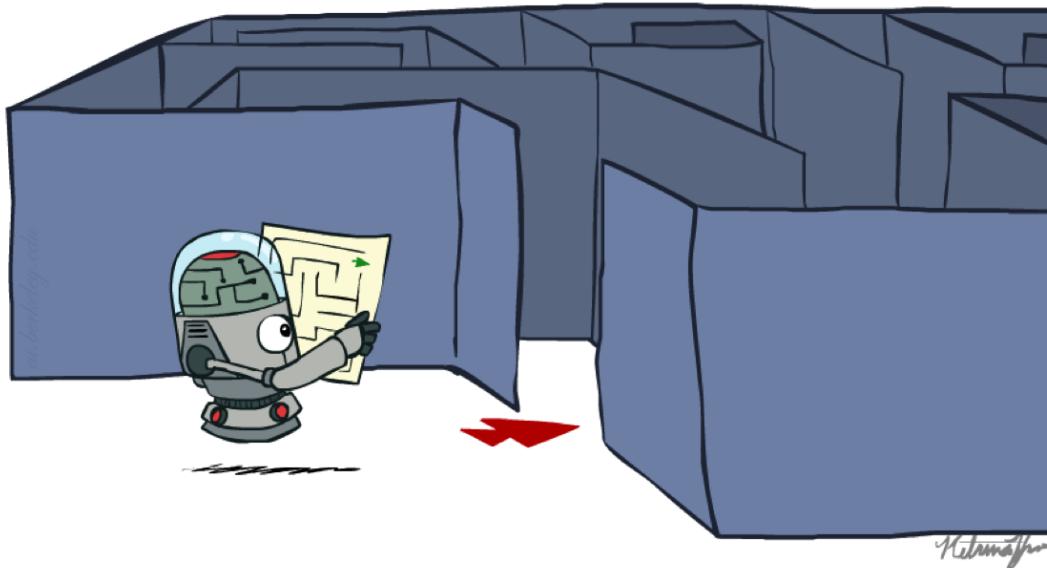
Arama

- Arama yöntemleri ?
 - Hangi düğüm açılacak?
- Bilgisiz Arama:
 - Blind Search
 - Durum bilgisinden yararlanmaz.
- Bilgili Arama:
 - Sezgisel Arama (Heuristics Strategies)
 - Durum bilgisinden yararlanır.



Bilgisiz Aramalar : Özeti

- BFS, tamdır, optimaldir ancak yer karmaşıklığı yüksektir.
- DFS, yer karmaşıklığı etkindir, ancak tam da değildir, optimal de değildir.
- Yinelemeli Derinleşen Arama, yaklaşık olarak optimaldir.
- Kavramsal olarak kuyruk ve yiğin yapıları kullanılarak gerçekleştirilebilirler.

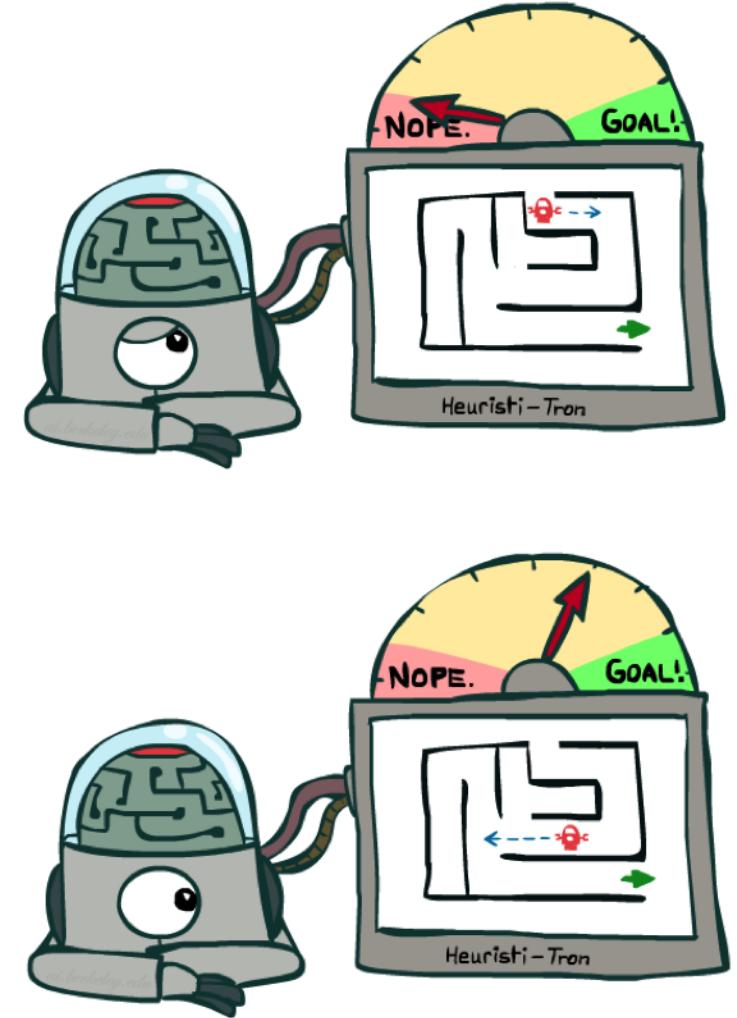
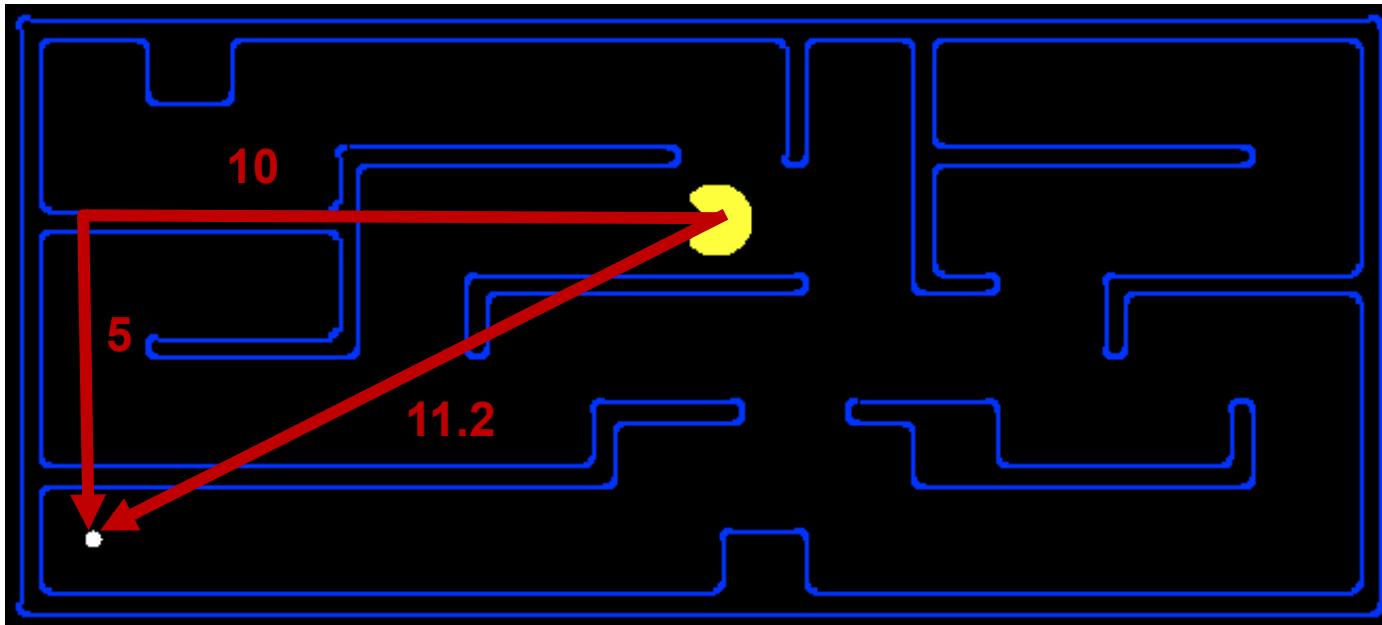


Bilgili Arama



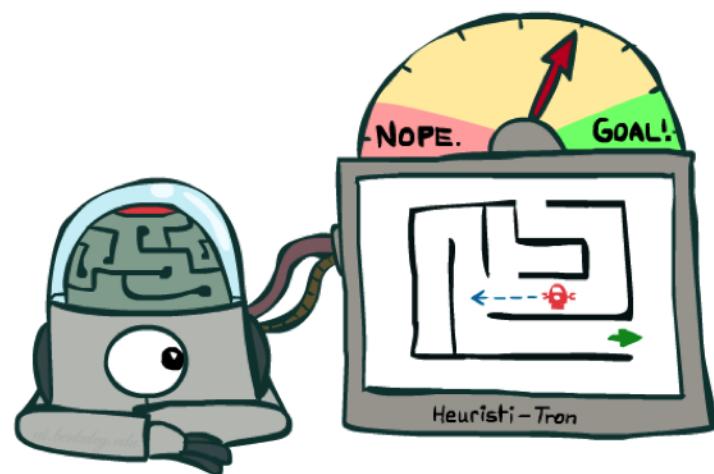
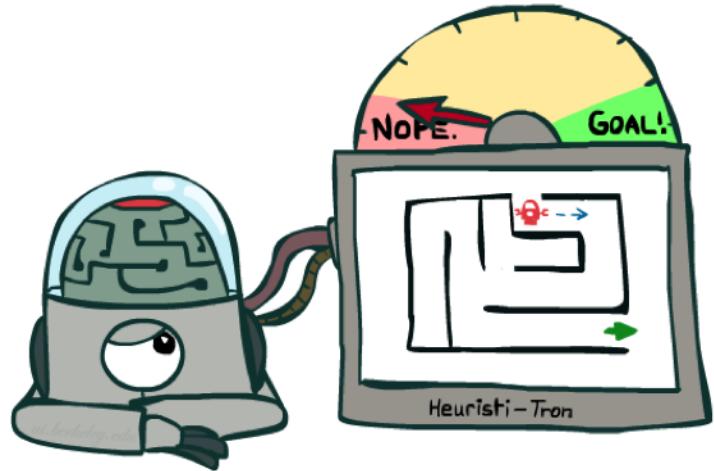
Sezgisel Arama

- Hedefe olan uzaklık kestirilirse ne olur?
- Sezgisellik?
 - Gerçek uzaklık tam olarak bilinmese de kestirilebilir.
- Yapay Zeka da Sezgisellik?
 - İnsanın sezgisel davranışları makinelere nasıl aktarılabilir.



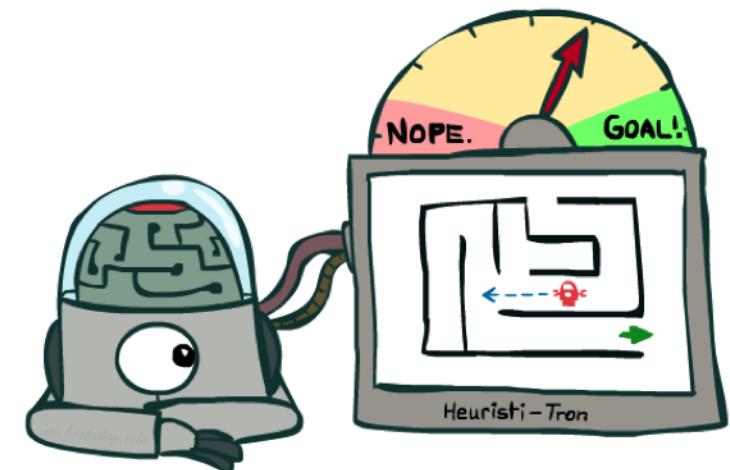
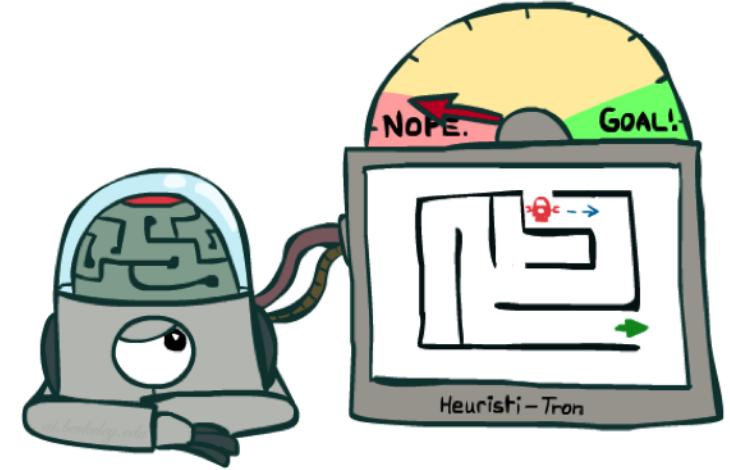
Sezgisel Arama

- Birçok problem aşağıdaki yaklaşımın çözülür:
 1. Olası durumlardan herhangi birini al
 2. Ele alınan duruma mümkün gidişler uygulanarak durumun değiştirilmesi
 3. Durumun değerlendirilmesi
 4. Gereksiz durumların atılması
 5. Eğer sonuca ulaşılmış ise çözümü tamamla değilse yeni değeri ele al işlemi tekrarla
- Algoritmik yaklaşımın hangi adımlar işletilir?
- Sezgisel yaklaşımın hangi adımlar işletilir ?

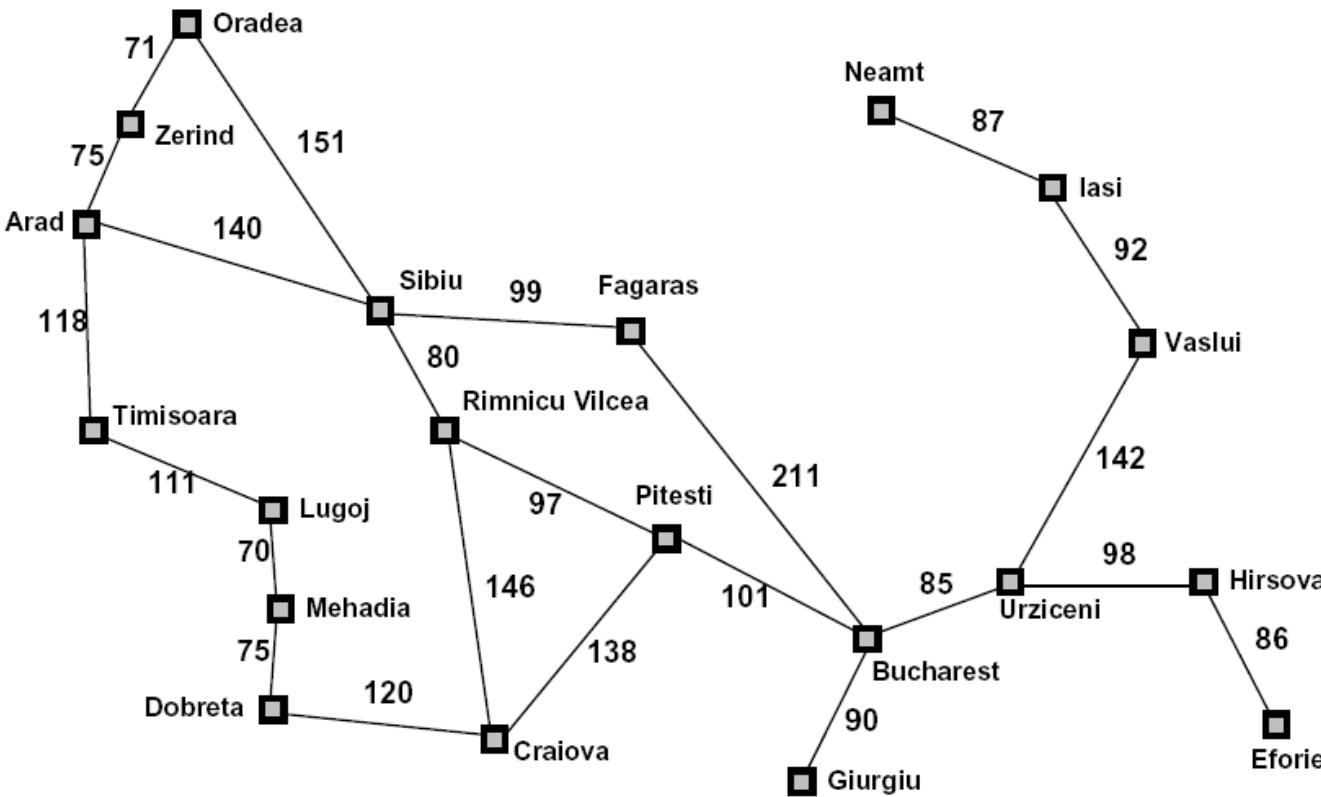


Sezgiselin Kabul Edilebilirliği

- n : Aramadaki herhangi bir durumdur.
- $c(n)$: n düğümünden amaç düğüme optimum yol maliyeti
- $h(n) = n$ düğümünden amaç düğüme maliyetin sezgisel tahmini
 - Robot yol planlaması, $h(n)$ Öklid mesafesi olabilir
 - 8 puzzle, $h(n)$ yerinde olmayan karo sayısı olabilir
- Aramayı yönlendirmek için $h(n)$ kullanan arama algoritmaları sezgisel arama algoritmalarıdır.
- Tüm düğümler için $h(n) \leq c(n)$ ise $h(n)$ kabul edilebilirdir !



Sezgisel Fonksiyon Örneği



Straight-line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

$h(x)$

Sezgisel Arama Yöntemleri

- En İyi Öncelikli Arama
- Açı Gözülü Arama
- A* Algoritması
- 8 Vezir Problemi
- Tepe Tırmanma
- Gradyan Azalma



En İyi Öncelikli Arama

- Her durumun (düğümün) istenilebilirliğini ifade edecek $f(n)$ fonksiyonu kullanılır.
- Genişletilen düğüm daha önce genişletilmemiş istenilirliği en yüksek olan düğümdür.
- Uygulaması:
 - Düğümleri **istenilirliklerine** göre büyükten küçüğe doğru sıralama
- Algoritmanın en önemli bileşeni:
 - $h(n)$: sezgisel fonksyon
 - n 'den bir hedef düğüme en ucuz yolun tahmini maliyeti

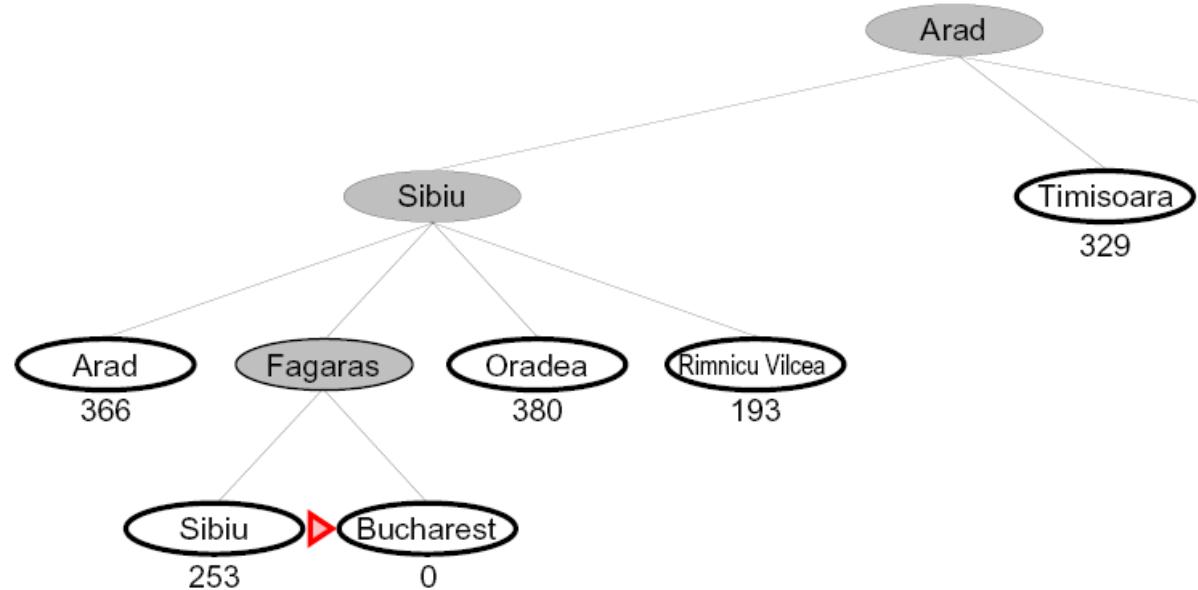
Aç Gözlü Arama



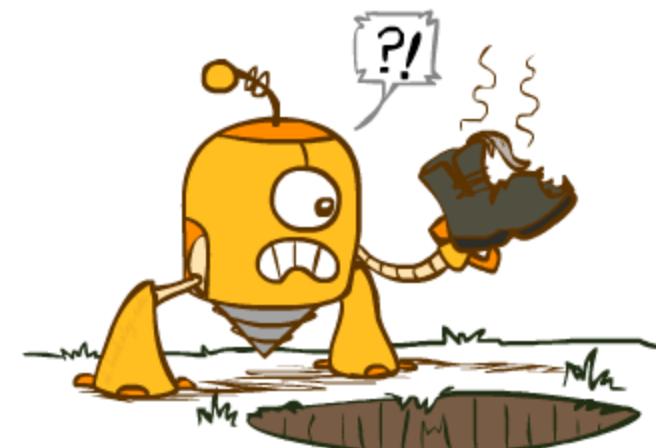
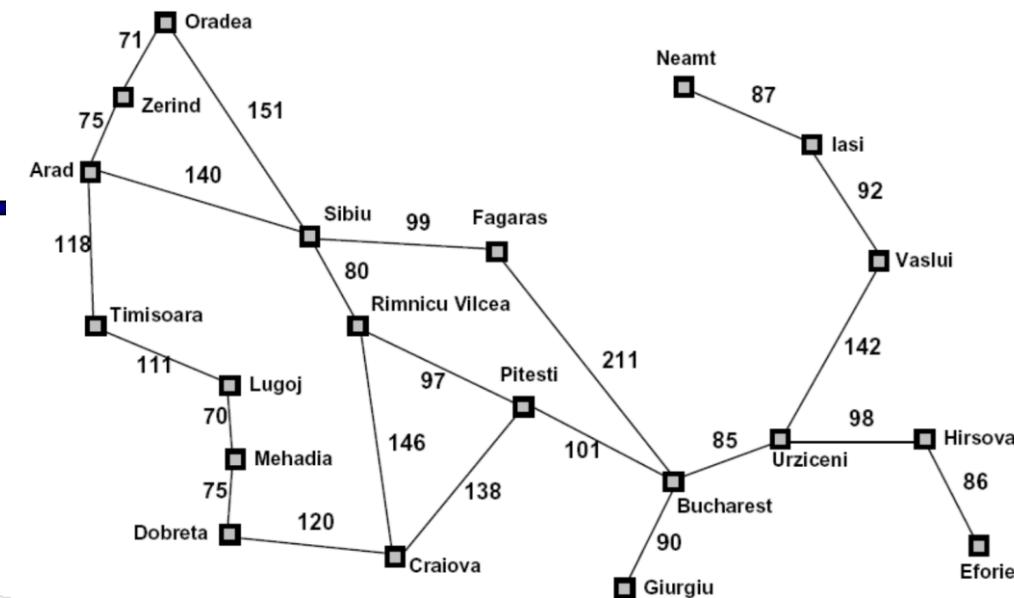
Aç Gözlü Arama

■ Genel Fikir:

- Hedefe en yakın görünen düğümü genişlet



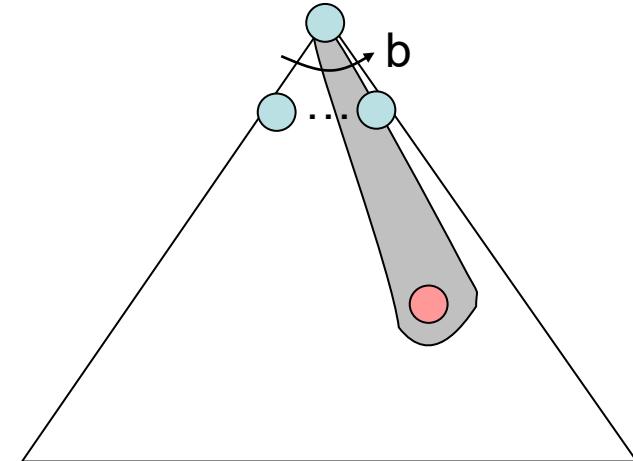
- Ne yanlış gidebilir?



Aç Gözlu Arama

- **Strateji:**

- Amaca en yakın düğümü genişlet
- $f(n) = h(n)$
- $h(n)$: n durumundan hedef duruma kadar hesaplanmış sezgisel maliyet

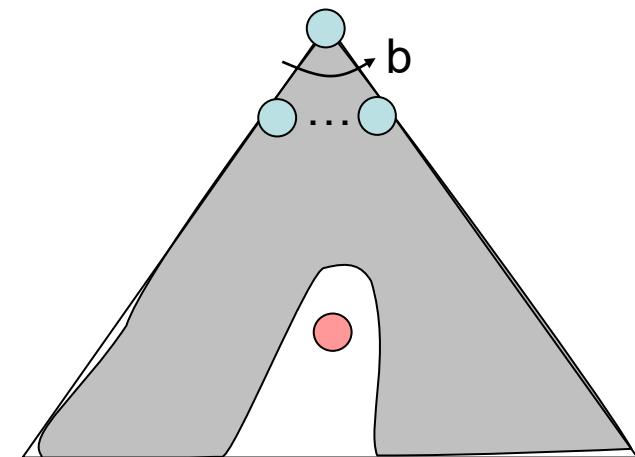


- **Genel Fikir:**

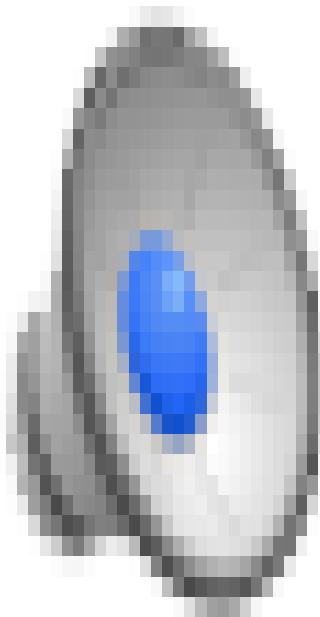
- En İyi Öncelikli arama yanlış yönlendirme yapar.

- **En kötü durumda:**

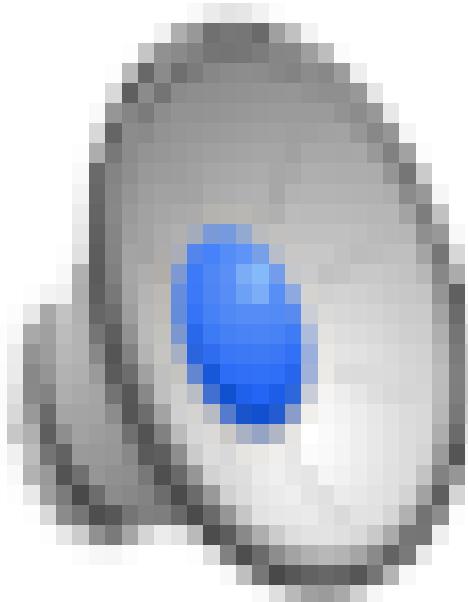
- Kötü yönlendirilmiş DFS gibi çalışır



Aç Gözlu Arama



Küçük Labirentli Pacman Örneği – Açı Gözülü Arama



Aç Gözülü En İyi Öncelikli Arama

- **Aç Gözülü Stratejisi:**

- Amaca en yakın düğümü genişlet
- $f(n) = h(n)$
- $h(n)$: n durumundan hedef duruma kadar hesaplanmış sezgisel maliyet

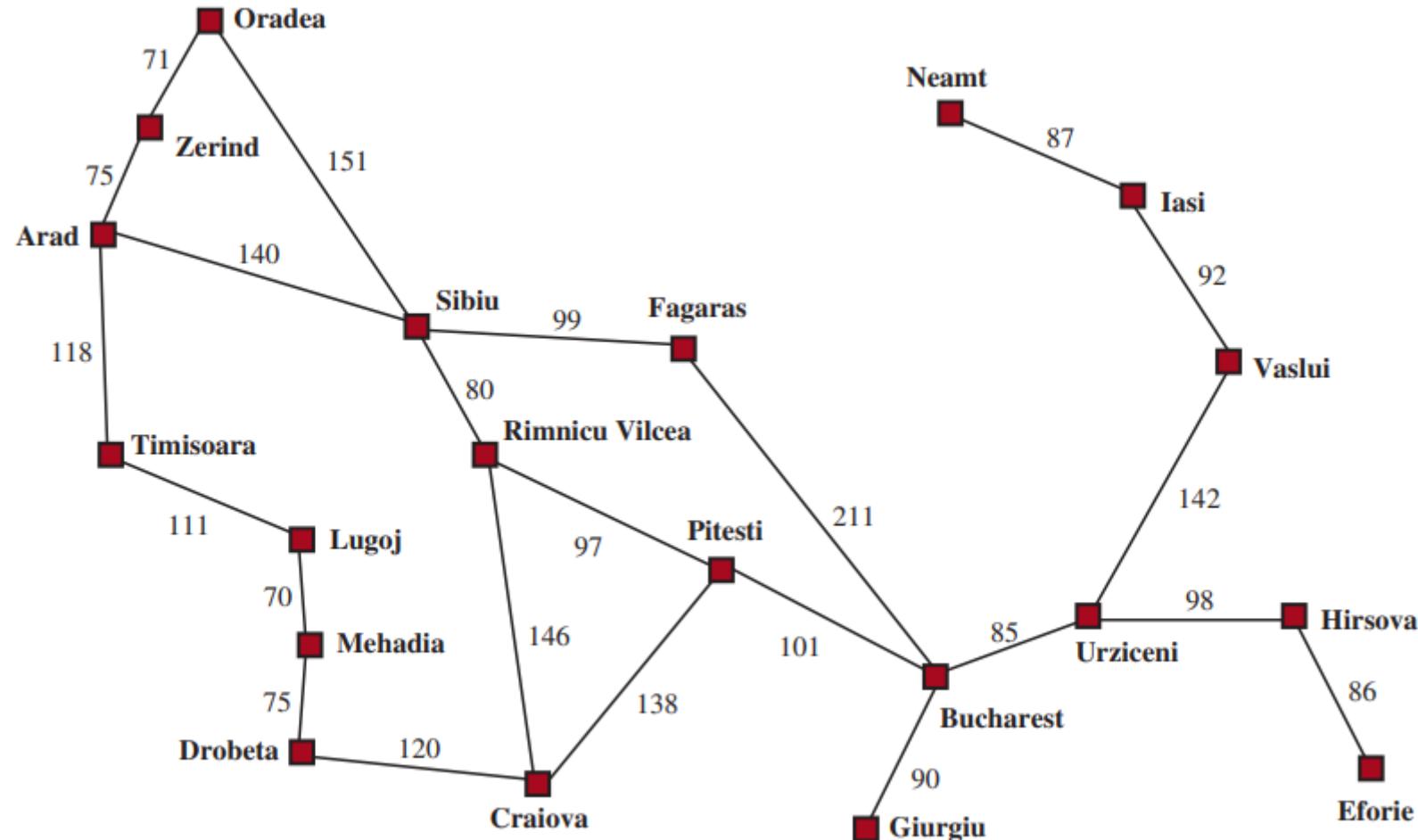
- **Problem: Haritada Arad'dan Bucharest'e yol bulma problemi**

- $h(n)$: n durumundan hedef duruma kuş uçuşu mesafe
- Straight Line Distance
- Çözüm: Arad-Sibiu-Fagaras-Bucharest

- **Analiz:**

- Optimum çözümü aramaz
- Çözüm tam değildir
- Zaman karmaşıklığı: $O(b^m)$
- Uzay karmaşıklığı: $O(b^m)$

Aç Gözlu En İyi Öncelikli Arama Örnek

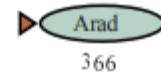


Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

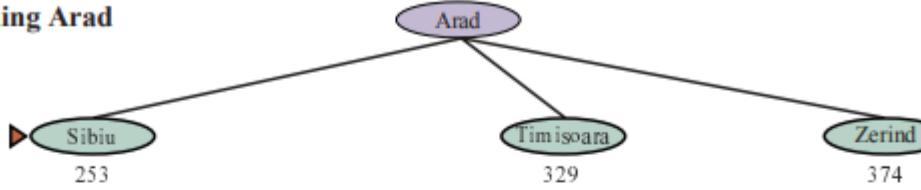
Figure 3.16 Values of h_{SLD} —straight-line distances to Bucharest.

Aç Gözlu En İyi Öncelikli Arama Örnek

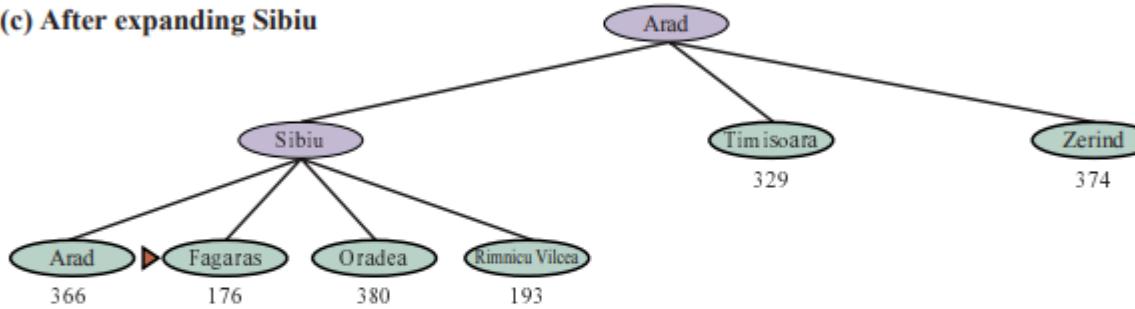
(a) The initial state



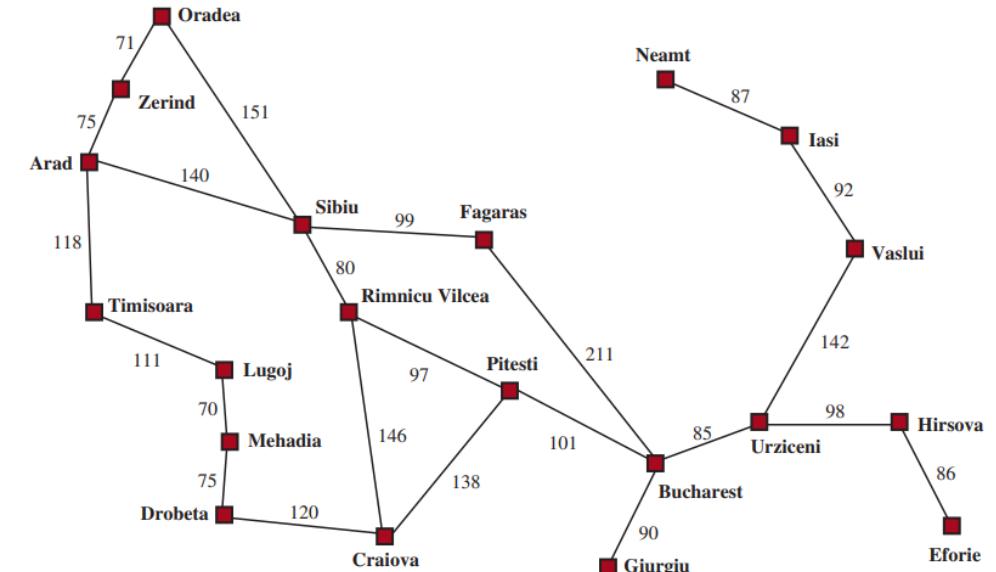
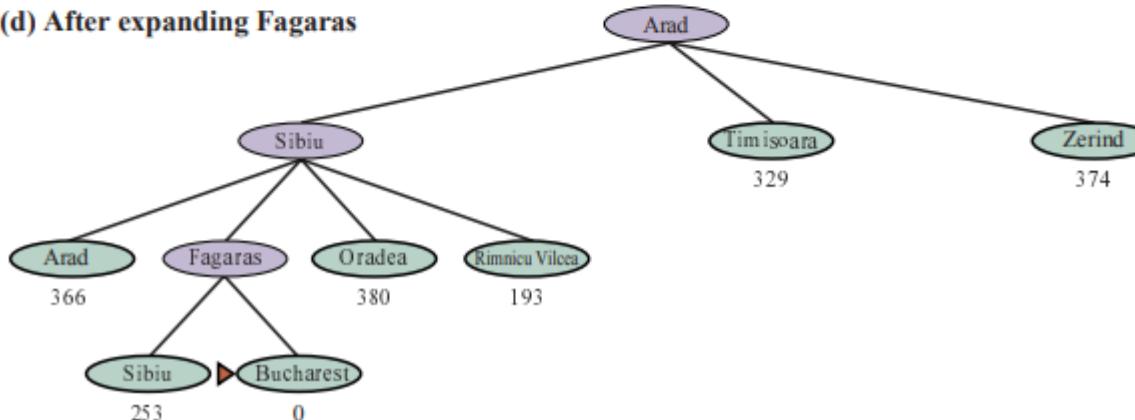
(b) After expanding Arad



(c) After expanding Sibiu



(d) After expanding Fagaras



Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.16 Values of h_{SLD} —straight-line distances to Bucharest.

A* Araması



A* Araması

- Başlangıç düğümü: s düğümü
- s düğümünden x düğümüne kadar yol:
 - $g(x)$: şu anki duruma kadar maliyet fonksiyonu
- x düğümünden hedef f düğümüne kadar yol:
 - $h(x)$: uygun sezgi fonksiyonu olsun.
- $g(x)$: x durumunun gerçek olan o anki değeri
- $h(x)$: x düğümünden çözüme olan gidişlerin sezgisel değeri

- A* Algoritması:
 - $f(x) = g(x) + h(x)$
- Hem optimal hem de tamdır.

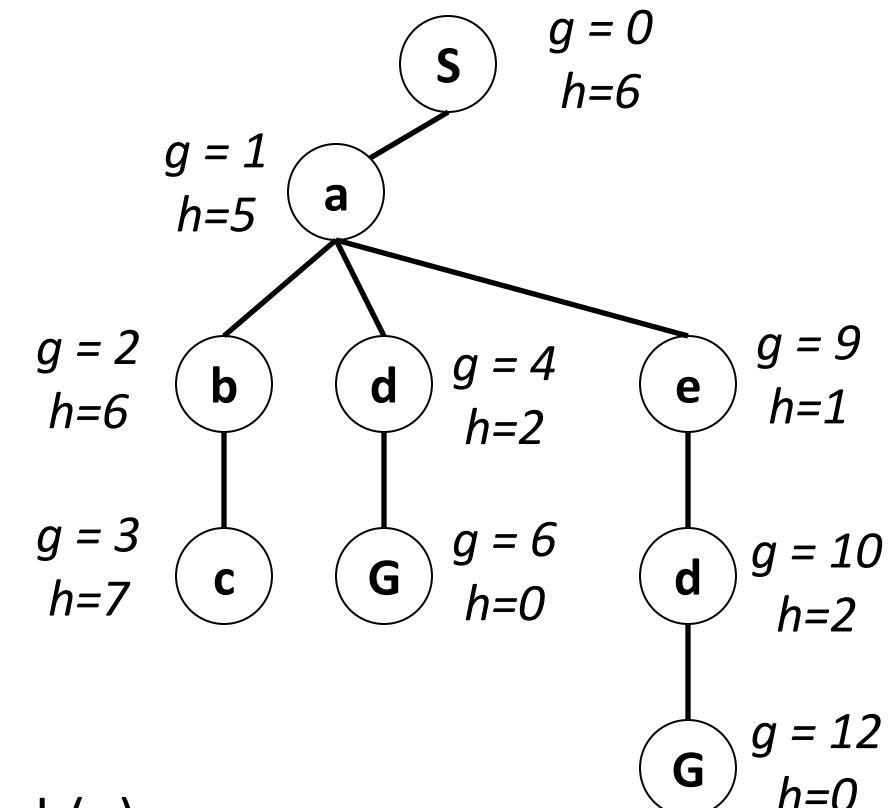
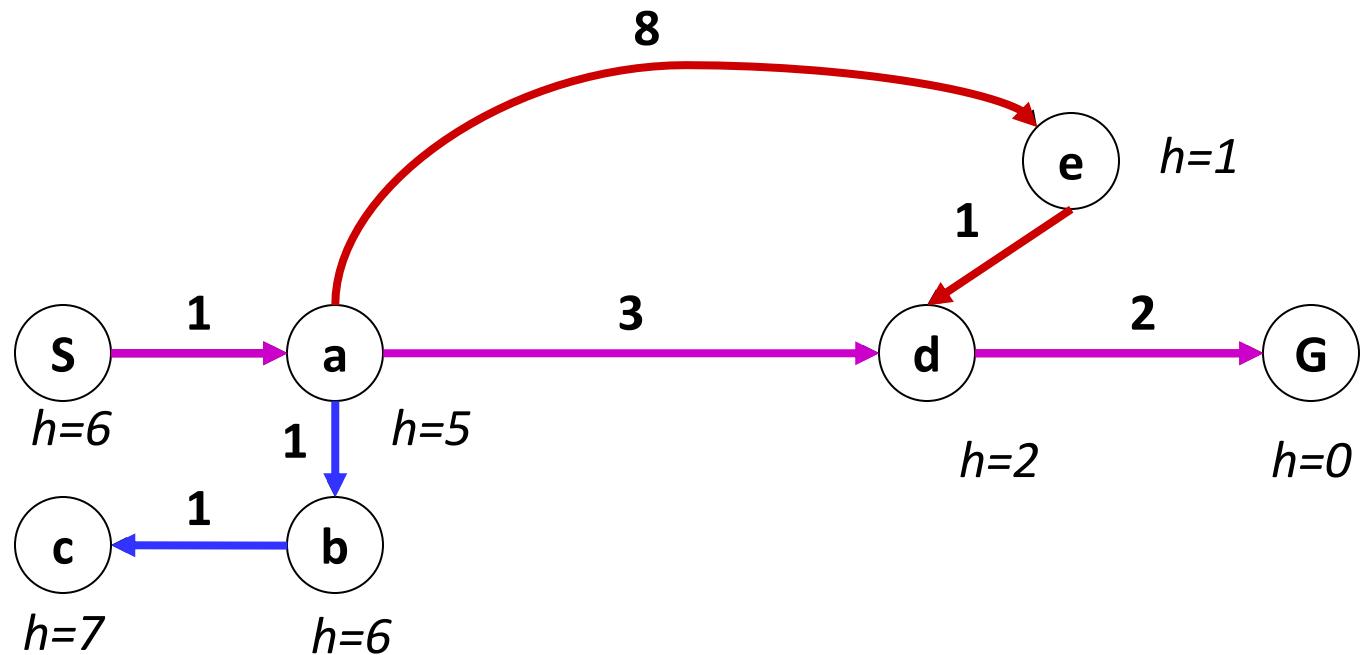


A* Search

1

UCS ve Aç Gözülü Yaklaşımını Birleştirir

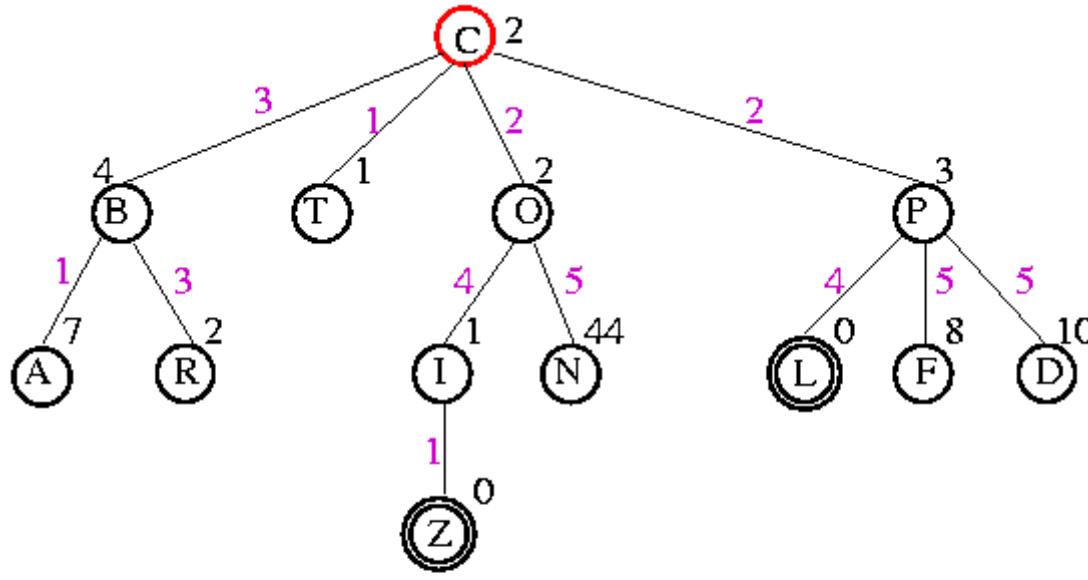
- Uniform-cost yol maliyetine göre sıralar yani: $g(n)$
- Açı gözülü yaklaşımı hedefe yakınlığına göre sıralar : $h(n)$



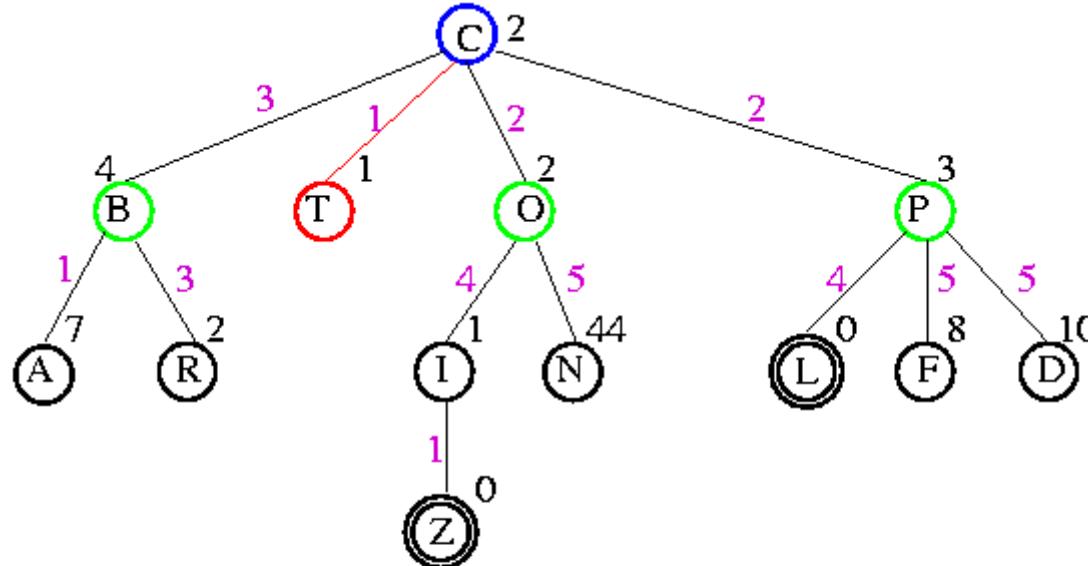
- A* araması bunların birleşimidir: $f(n) = g(n) + h(n)$

Example: Teg Grenager

A* Araması Örnek

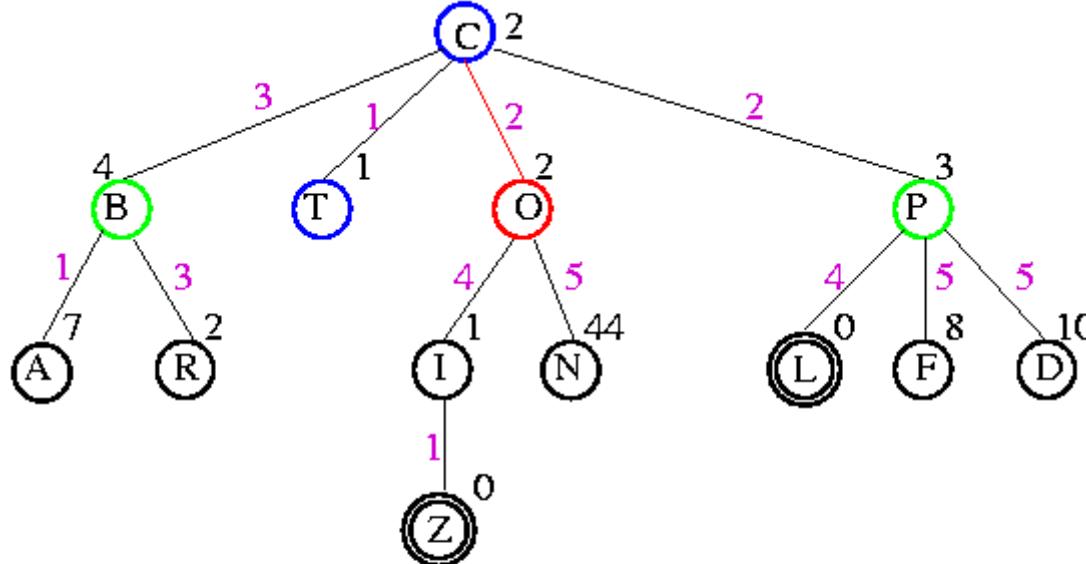


A* Araması Örnek



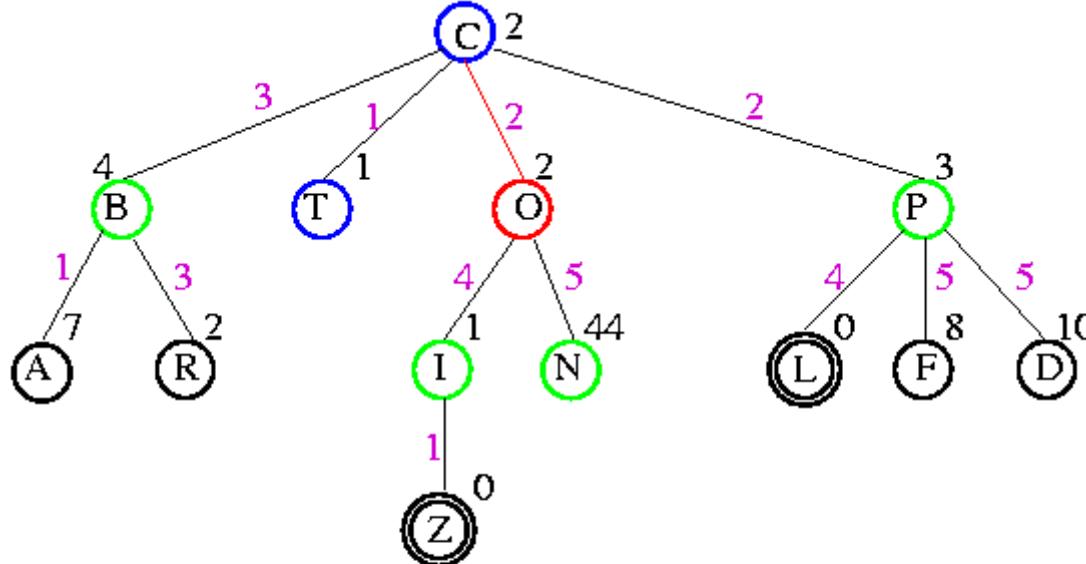
Open List = T ($1+1=2$), O ($2+2=4$), P ($2+3=5$), B ($3+4=7$)

A* Araması Örnek



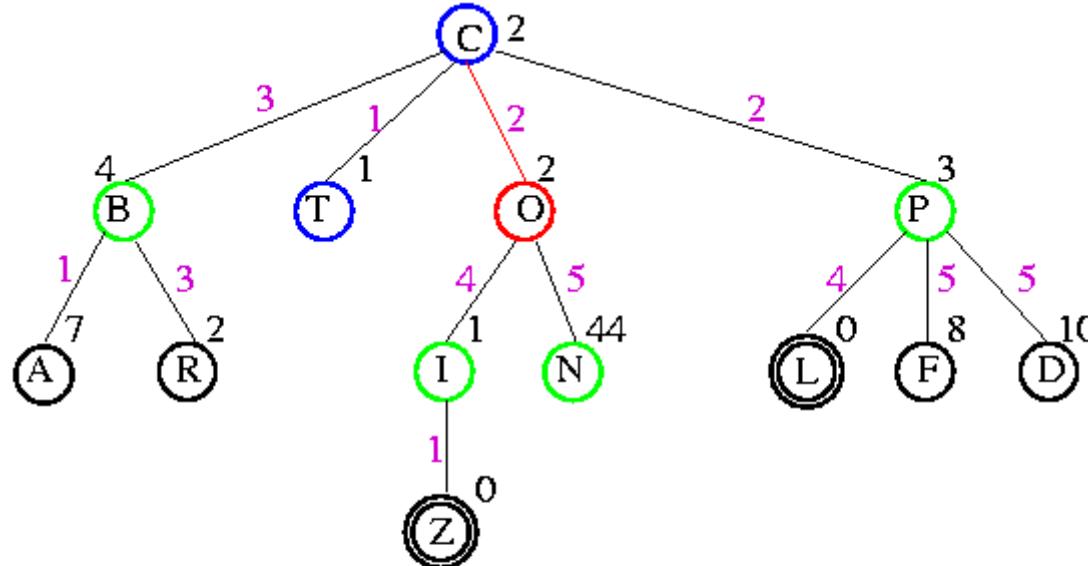
Open List = O (2+2=4), P (2+3=5), B(3+4=7)

A* Araması Örnek



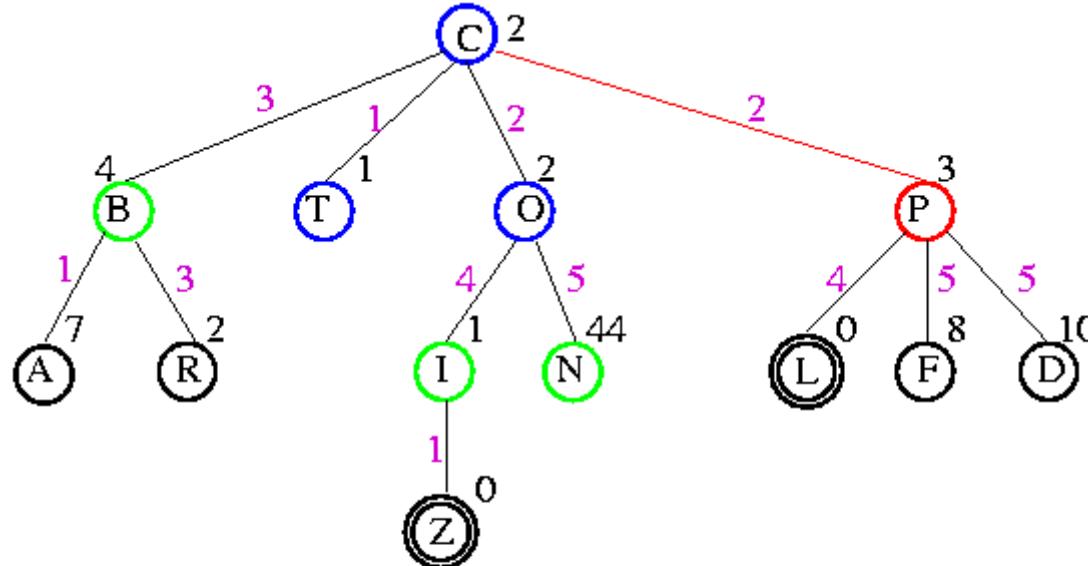
Open List = O (2+2=4), P (2+3=5), B(3+4=7)

A* Araması Örnek



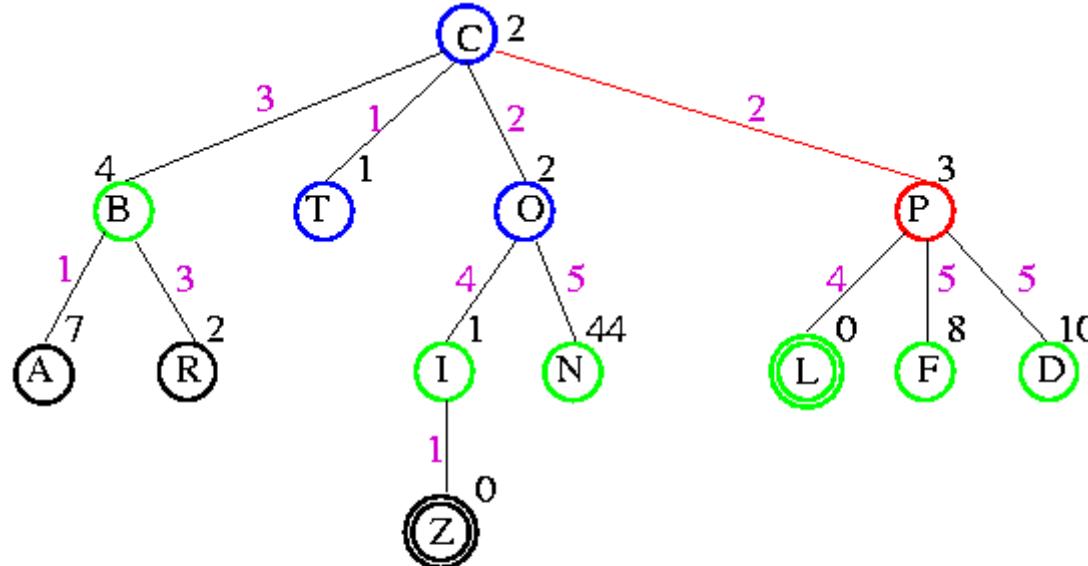
Open List = O ($2+2=4$), P ($2+3=5$), B($3+4=7$)
I ($6+1=7$), N ($7+44=51$)

A* Araması Örnek



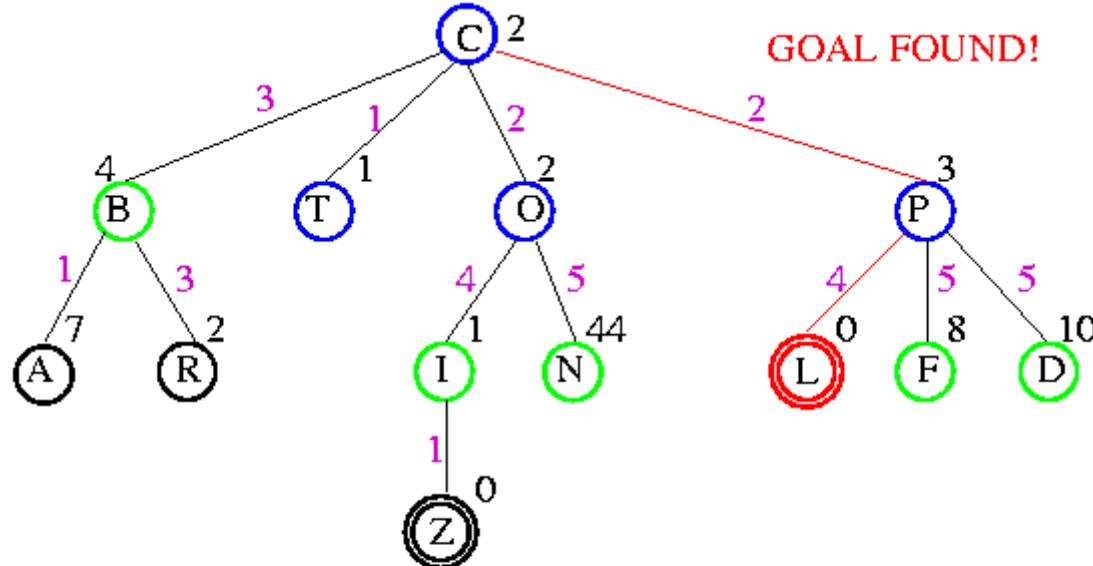
Open List = P (2+3=5), B(3+4=7)
I (6+1=7), N (7+44=51)

A* Araması Örnek



Open List = P (2+3=5), L (6+0=6), B (3+4=7)
I (6+1=7), F (7+8=15), D (7+10=17), N (7+44=51)

A* Araması Örnek



Open List = L ($6+0=6$), B ($3+4=7$)
I ($6+1=7$), F ($7+8=15$), D ($7+10=17$), N ($7+44=51$)

A* Araması Örnek

(a) The initial state



366=0+366

(b) After expanding Arad



393=140+253

447=118+329

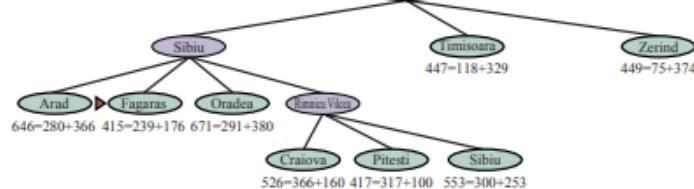
449=75+374

(c) After expanding Sibiu



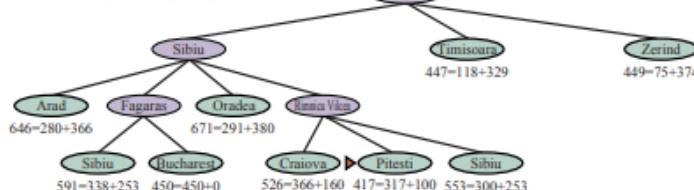
646=280+366 415=239+176 671=291+380 413=220+193

(d) After expanding Rimnicu Vilcea



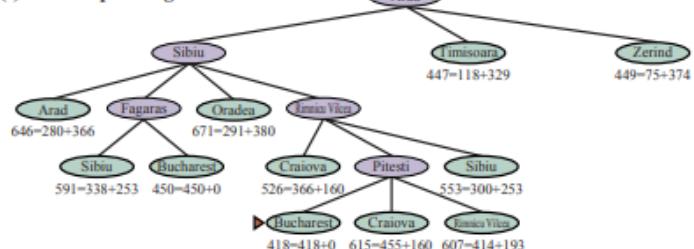
646=280+366 415=239+176 671=291+380
526=366+160 417=317+100 553=300+253

(e) After expanding Fagaras

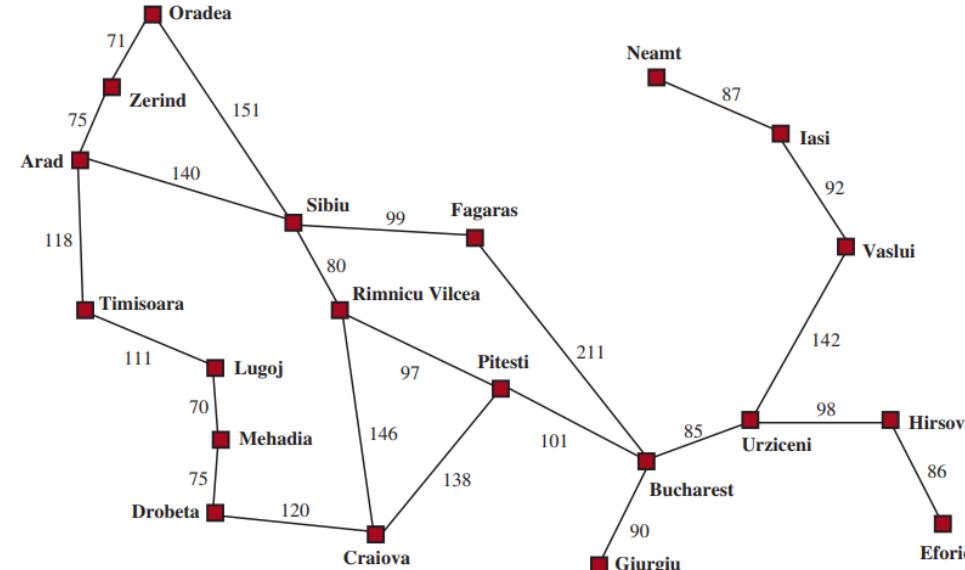


591=338+253 450=450+0
526=366+160 417=317+100 553=300+253

(f) After expanding Pitesti



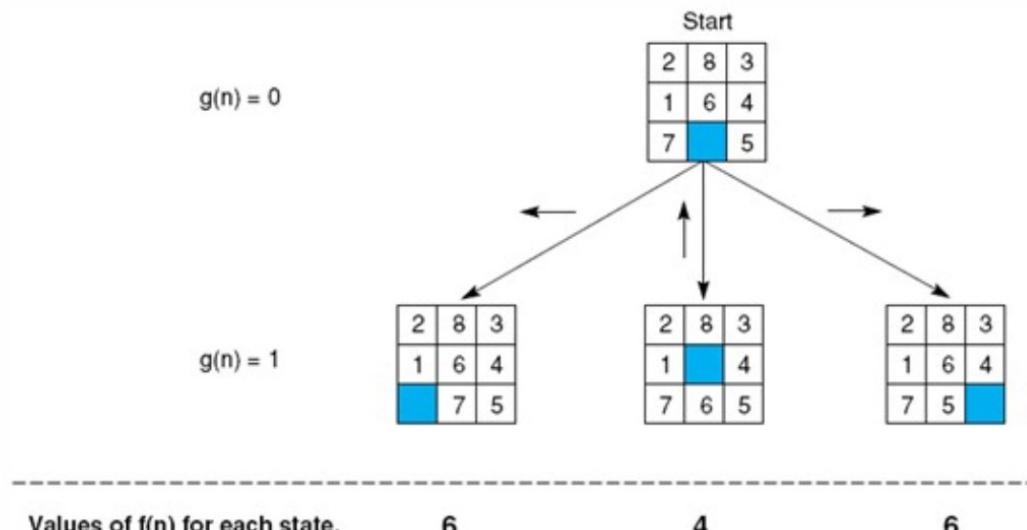
418=418+0 615=455+160 607=414+193



Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.16 Values of h_{SLD} —straight-line distances to Bucharest.

A* Araması Örnek : 8 Puzzle



where:

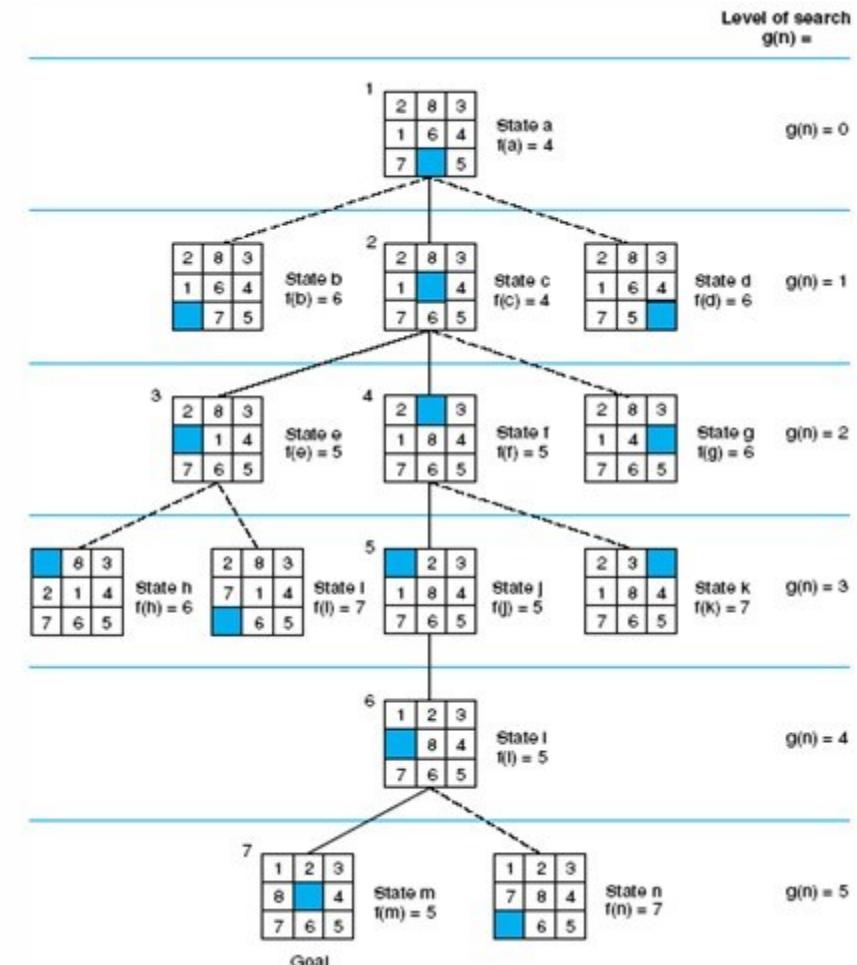
$$f(n) = g(n) + h(n),$$

$g(n)$ = actual distance from n to the start state, and

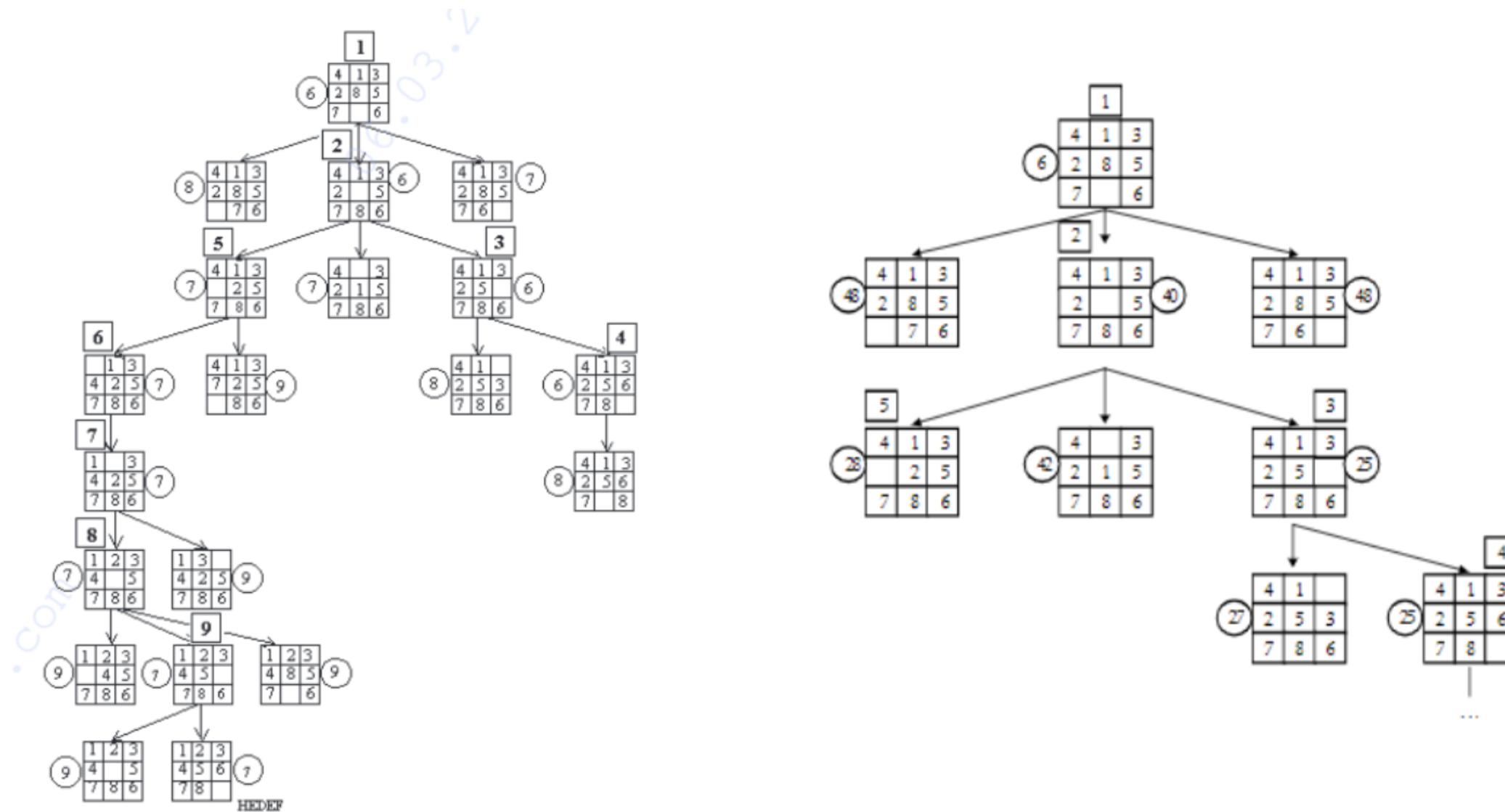
$h(n)$ = number of tiles out of place.

Goal state:

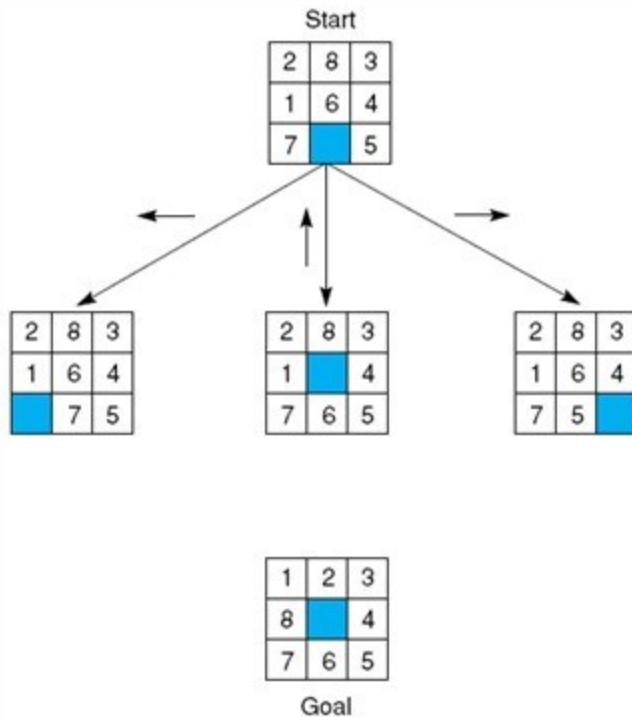
1	2	3
8	4	
7	6	5



A* Araması Örnek : 8 Puzzle



A* Araması Örnek : 8 Puzzle



2 8 3 1 6 4 7 5	5	6	0
2 8 3 1 4 6 7 5	3	4	0
2 8 3 1 6 4 7 5	5	6	0

Tiles out of place Sum of distances out of place 2 x the number of direct tile reversals

Goal

1 2 3 8 4 7 6 5

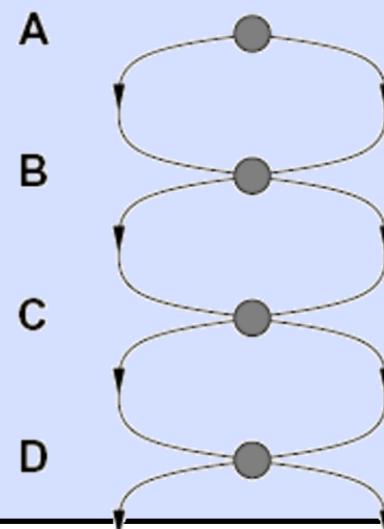
A* Araması Özellikleri

- Tamlik ?
 - Tamdır. Çözüm varsa bulur.
- Zaman Karmaşıklığı:
 - Kullanılan sezgisele göre değişebilir
 - En kötü durumda : Üstel
- Yer Karmaşıklığı
 - Bütün düğümleri bellekte tutar.
 - $O(b^m)$
- Optimallik
 - ?
- Zaman ve Yer Karmaşıklığı Kötü
 - Çözüm ?

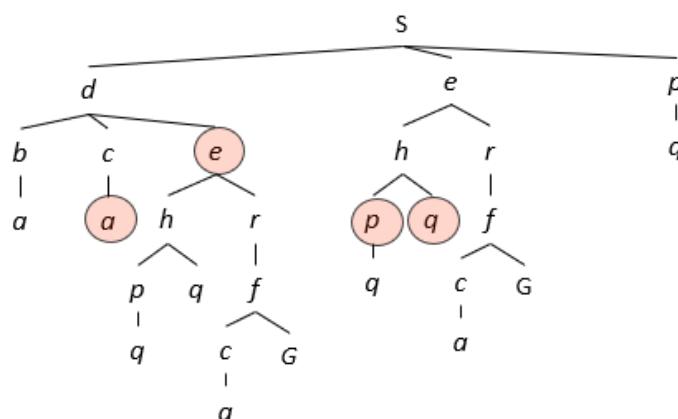
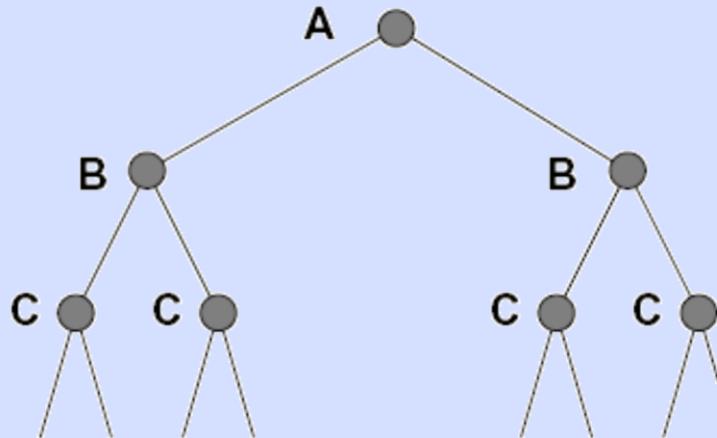


Ağaç Üzerinde Arama – Fazla Çaba!

Durum Grafları

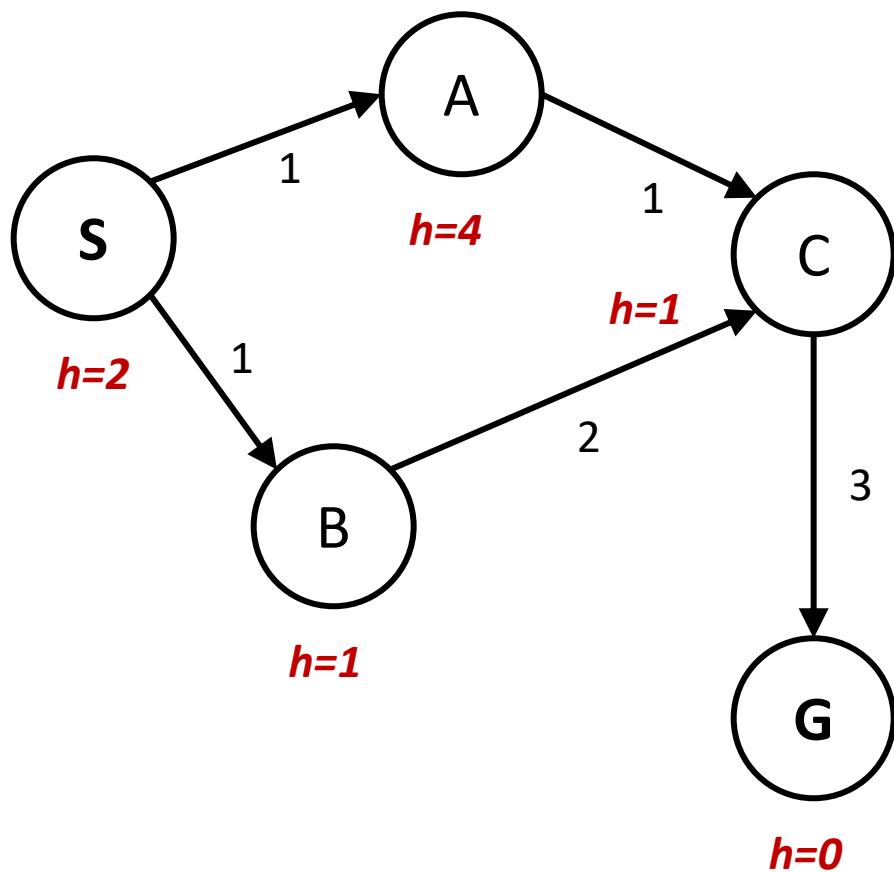


Arama Ağaçları

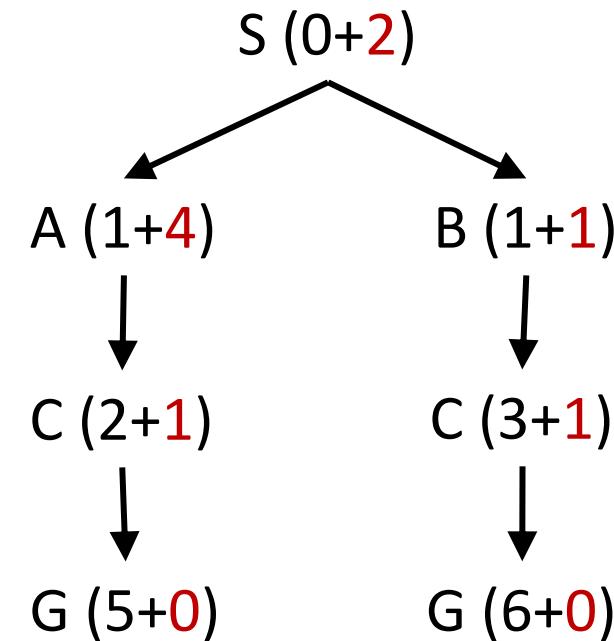


A* Araması?

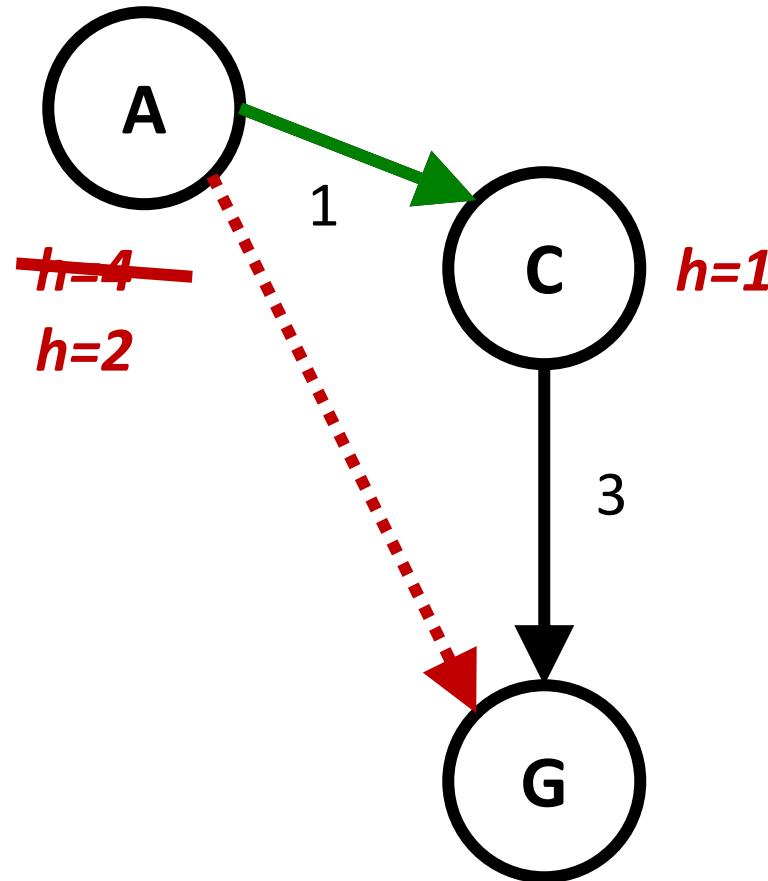
Durum Uzayı Grafi



Arama Ağacı

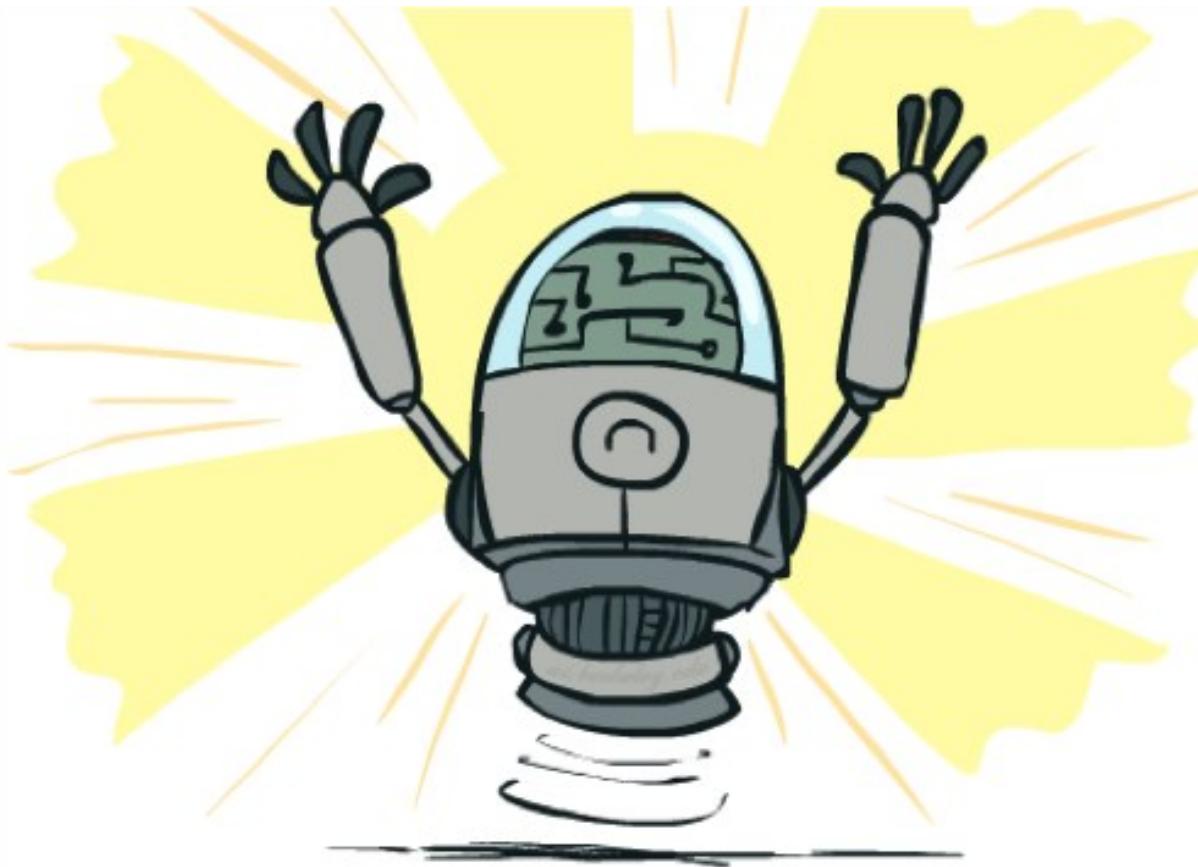


Sezgiselin Tutarlığı



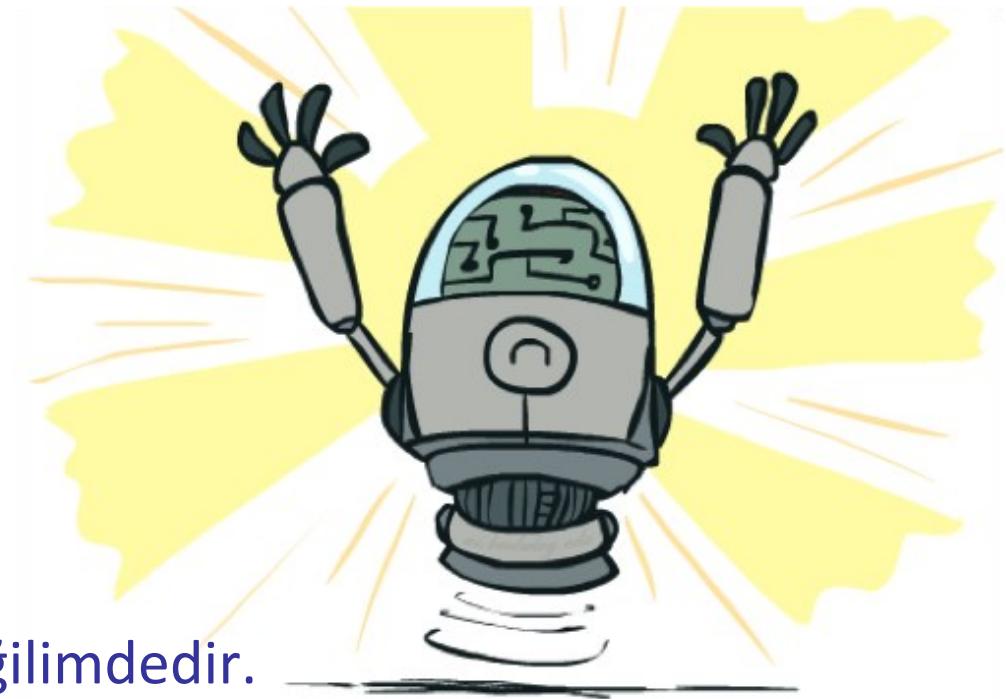
- Ana fikir: tahmini sezgisel maliyet \leq gerçek maliyetler
 - Kabul edilebilirlik: sezgisel maliyet \leq amaca olan gerçek maliyet
$$h(A) \leq A\text{'dan } G\text{^ye gerçek maliyet}$$
 - Tutarlılık : Sezgisel “arc” maliyeti \leq her bir ok için gerçek maliyet
$$h(A) - h(C) \leq \text{cost}(A \text{ to } C)$$
- Tutarlılık Sonuçları:
 - f değeri bir yol boyunca asla azalmaz!
$$h(A) \leq \text{cost}(A \text{ to } C) + h(C)$$
 - Bu durumda A* graf araması optimaldir!

A* Aramasının Optimalliği



Optimallik

- Ağaç Araması:
 - Sezgisel fonksiyon Kabul edilebilirse A* optimaldir.
 - UCS yöntemi ($h = 0$) olan A* aramasıdır.
- Graf Araması:
 - Sezgisel fonksiyon tutarlı ise A* optimaldir.
- Tutarlılık Kabul edilebilirlik anlamına gelir.
- Çoğu Kabul edilebilir sezgisel yöntem tutarlı olma eğilimdedir.

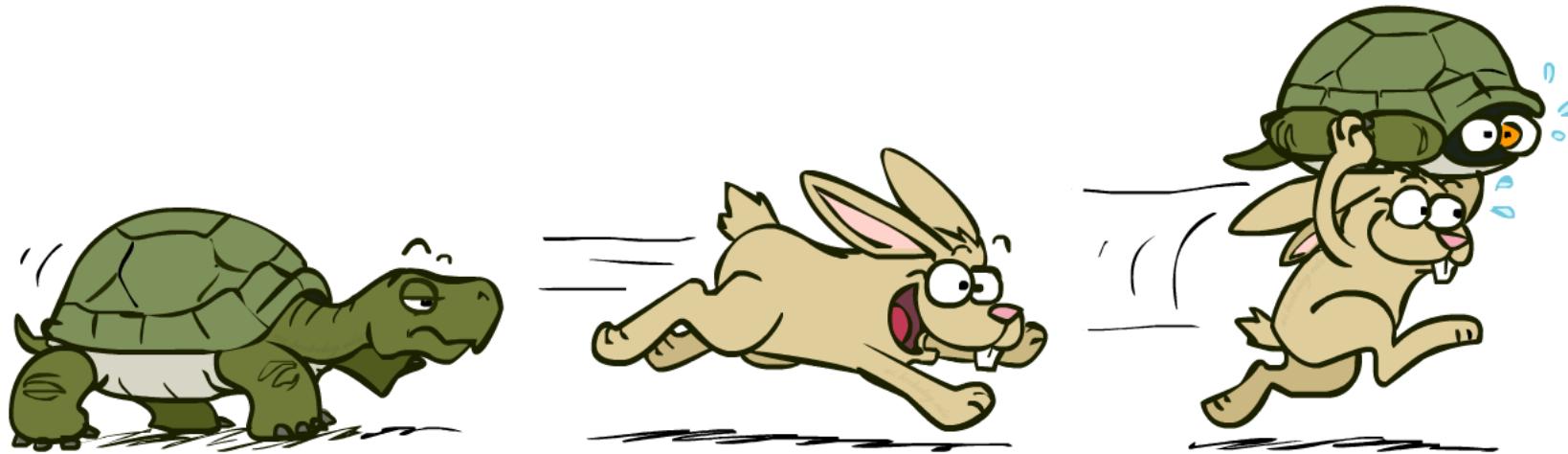


A*: Özeti



A*: Özeti

- A* hem geriye dönük hem de ileriye dönük maliyetleri kullanır.
- A* kabul edilebilir/tutarlı sezgisel yöntemler ile optimaldır.
- Sezgisel fonksiyon tasarıımı önemlidir!

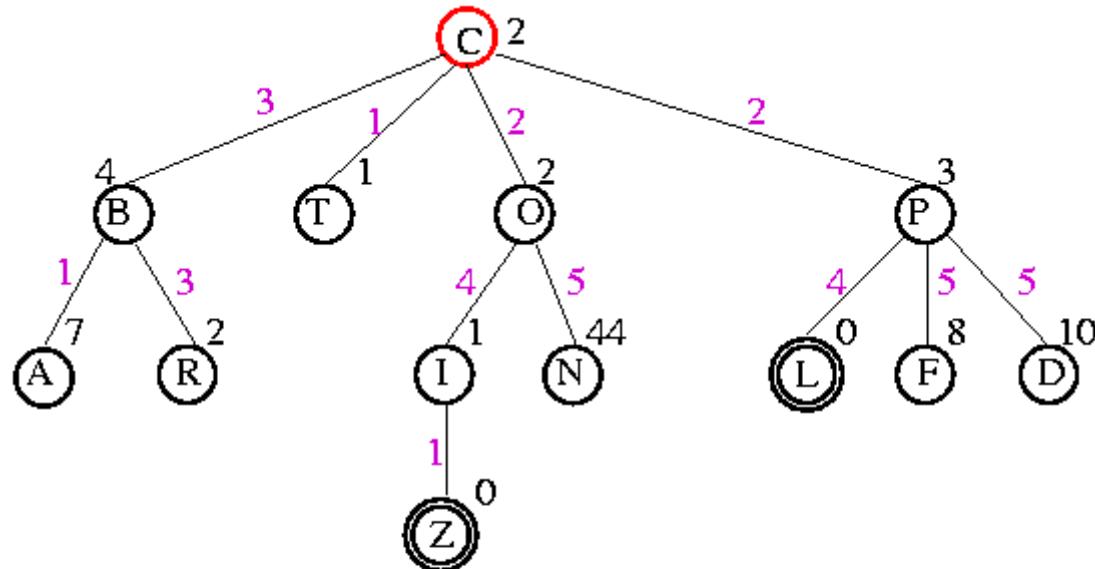


Bellek Sınırlı Sezgisel Arama

- A* algoritmasının bellek gereksinimleri nasıl indirgenebilir ?
 - Iterative Deepening
- Iterative Deepening A*
 - IDA*
- Örnek Algoritmalar
 - Recursive best-first search (RBFS)
 - MA*(memory-bounded A*)

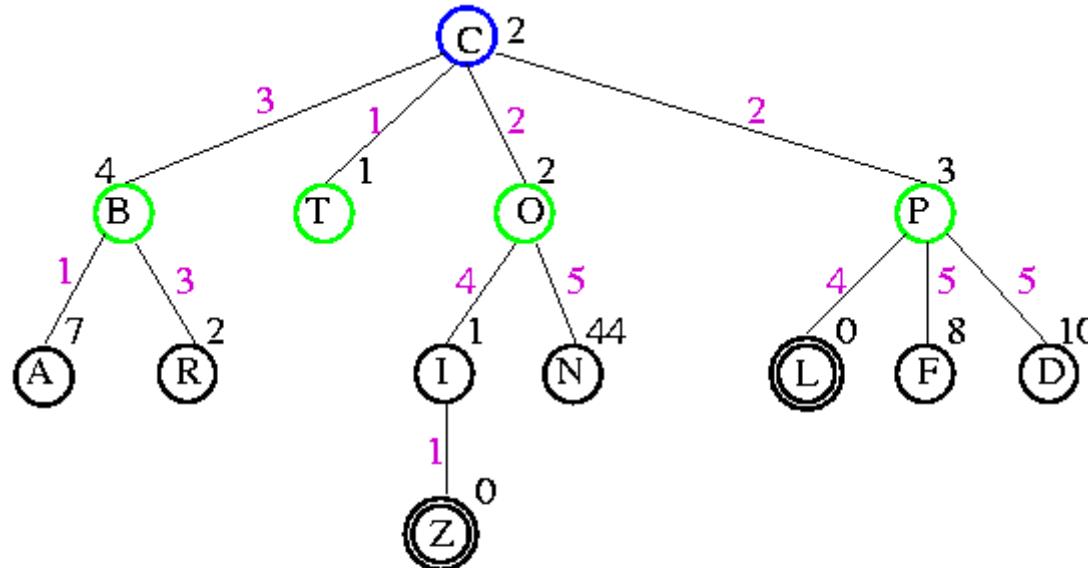


ÖRNEK



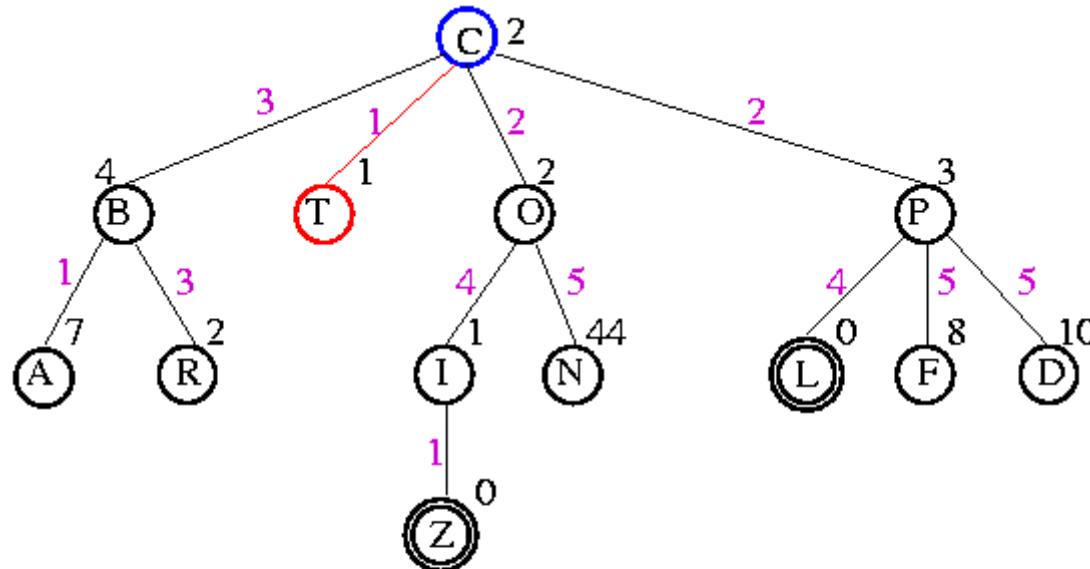
limit = $f(C) = 2$

ÖRNEK



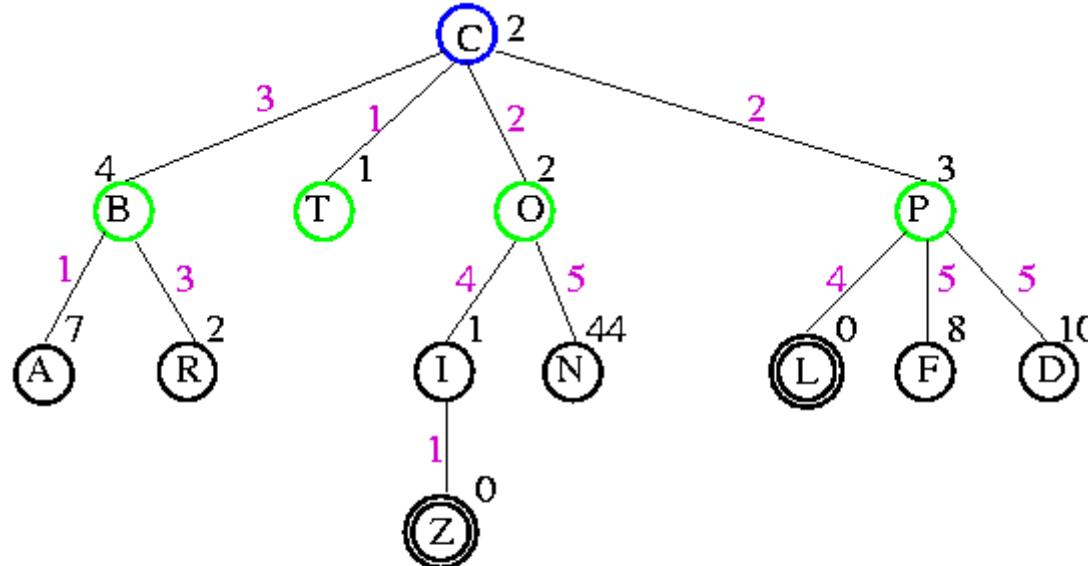
$\text{limit} = f(C) = 2$

ÖRNEK



$\text{limit} = f(C) = 2$

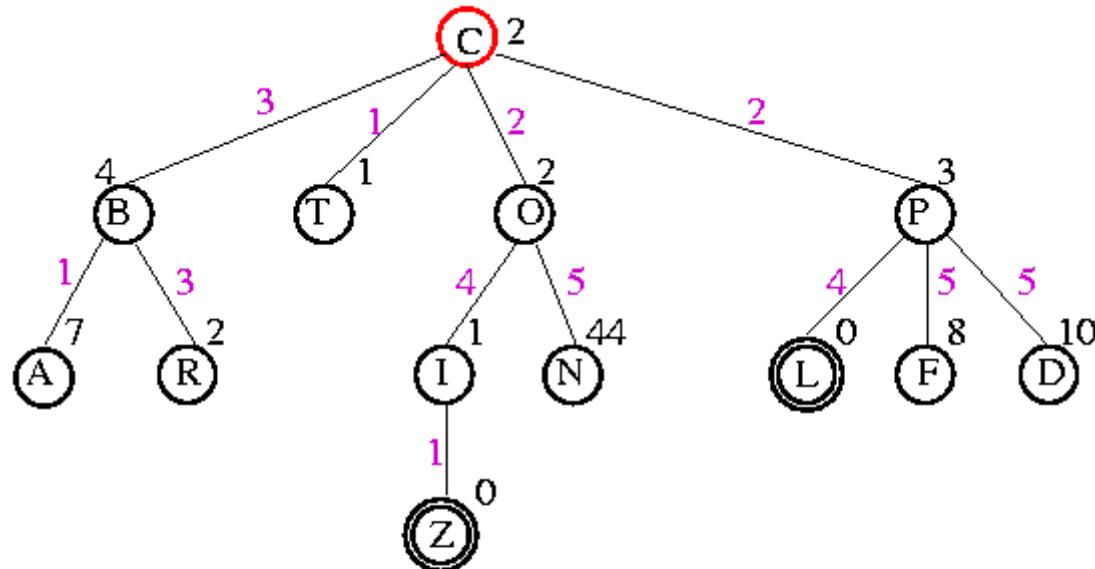
ÖRNEK



$$\text{limit} = f(C) = 2$$

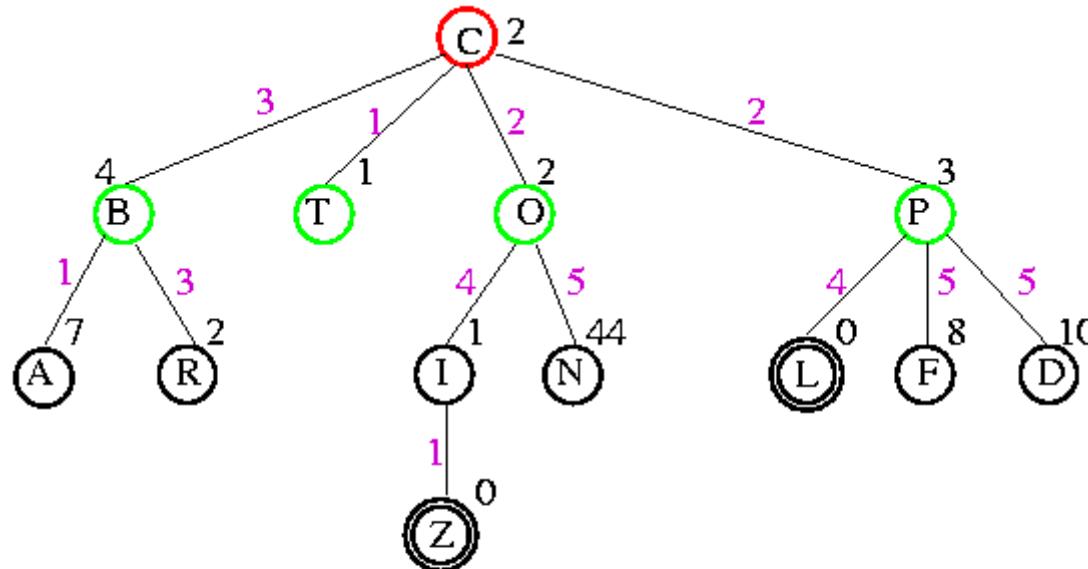
Nodes on frontier: B ($3+4=7$), O($2+2=4$), P($2+3=5$)
New limit = $f(O) = 4$

ÖRNEK



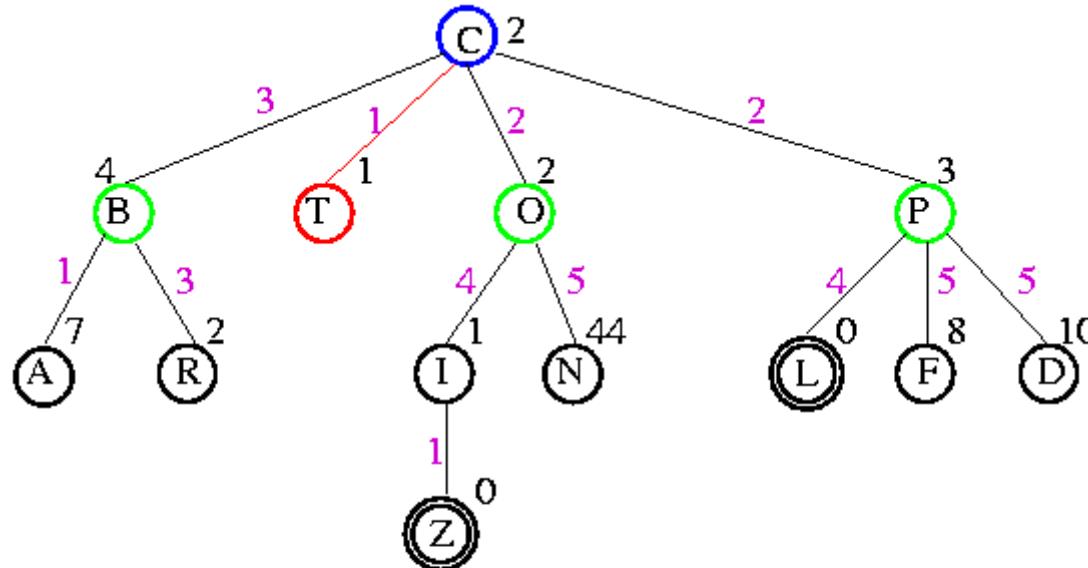
limit = $f(O) = 4$

ÖRNEK



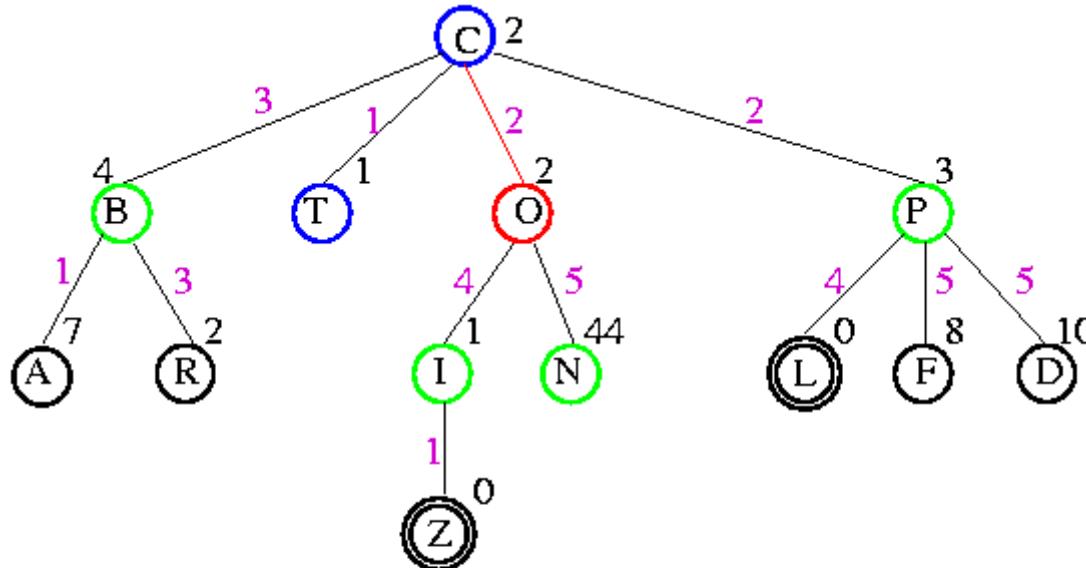
limit = $f(O) = 4$

ÖRNEK



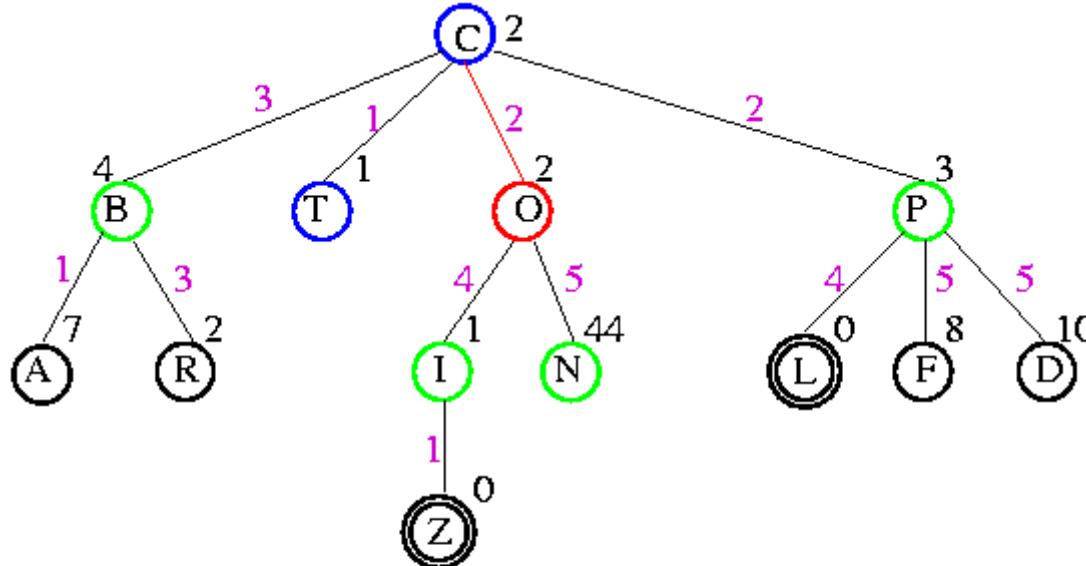
limit = $f(O) = 4$

ÖRNEK



limit = $f(O) = 4$

ÖRNEK



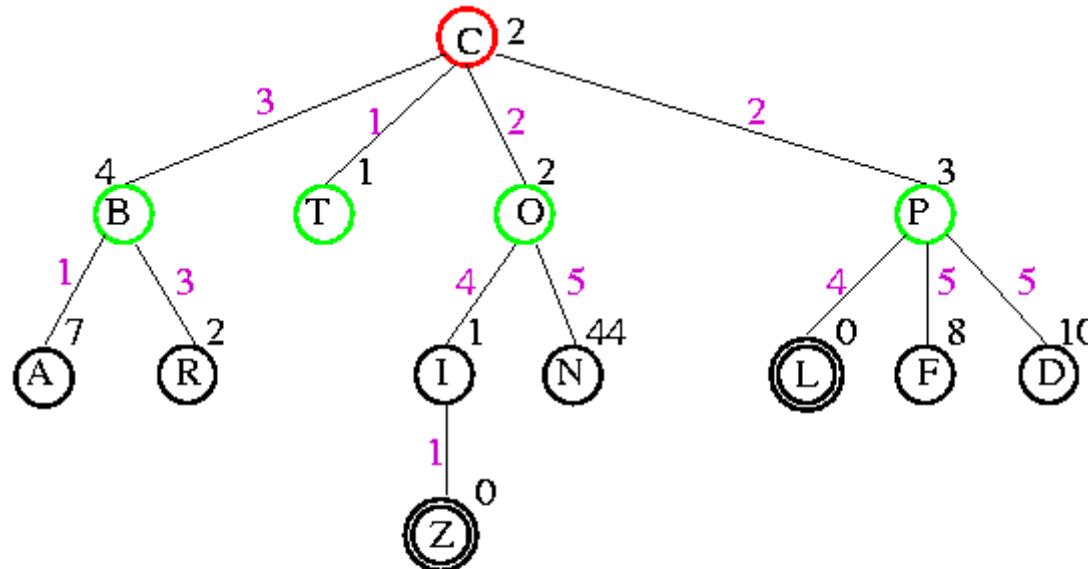
$$\text{limit} = f(O) = 4$$

Nodes on frontier: B ($3+4=7$), P ($2+3=5$)

I ($6+1=7$), N ($7+44=51$)

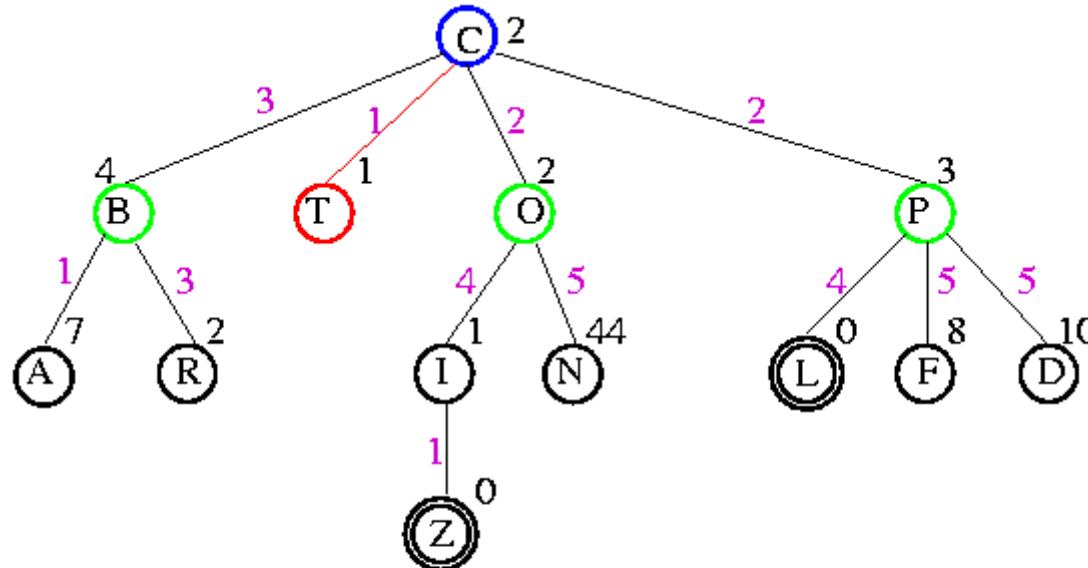
New limit = f(P) = 5

ÖRNEK



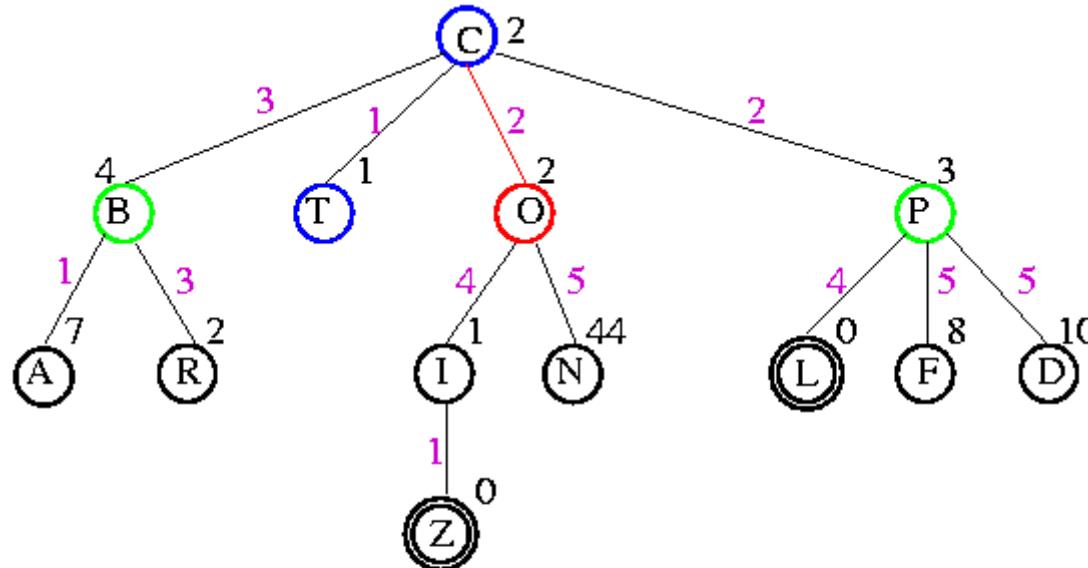
limit = $f(P) = 5$

ÖRNEK



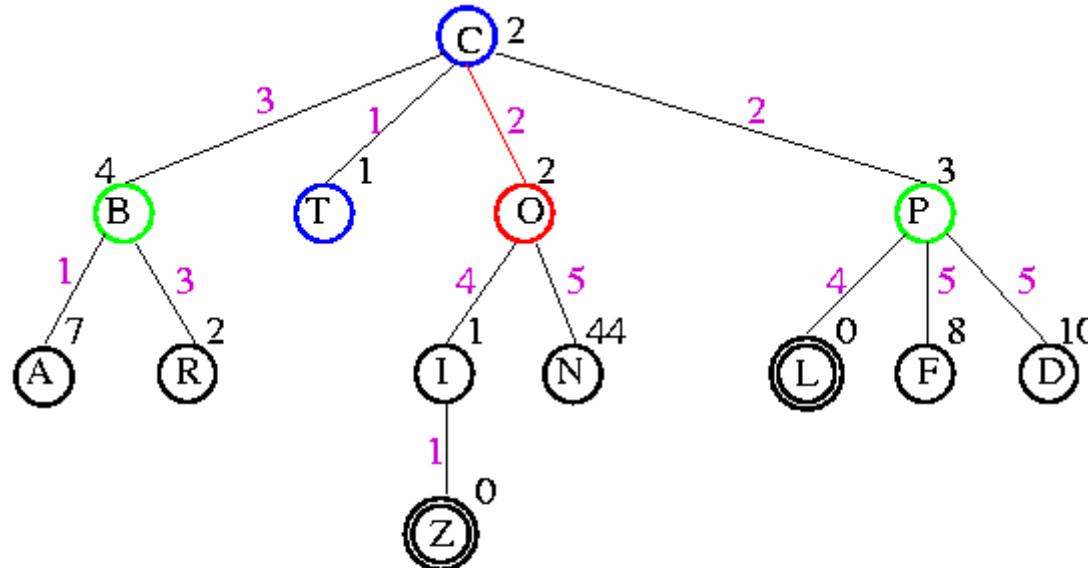
limit = $f(P) = 5$

ÖRNEK



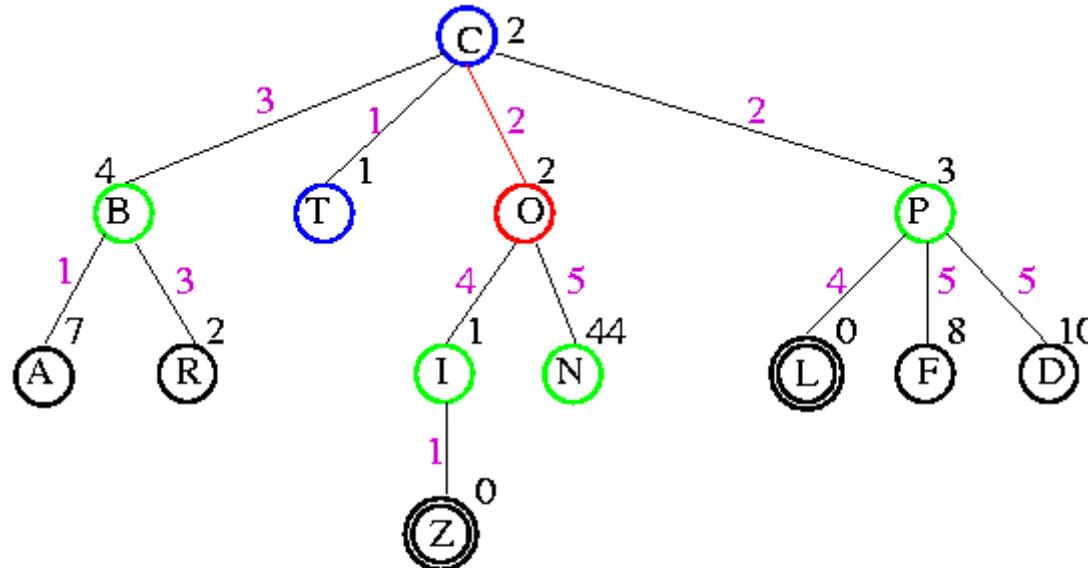
limit = $f(P) = 5$

ÖRNEK



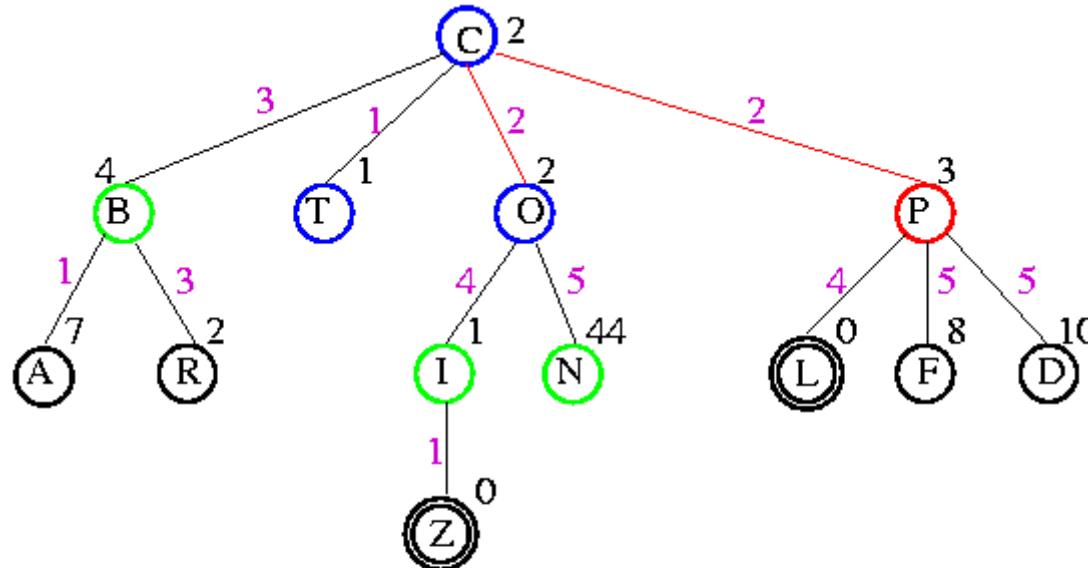
limit = $f(P) = 5$

ÖRNEK



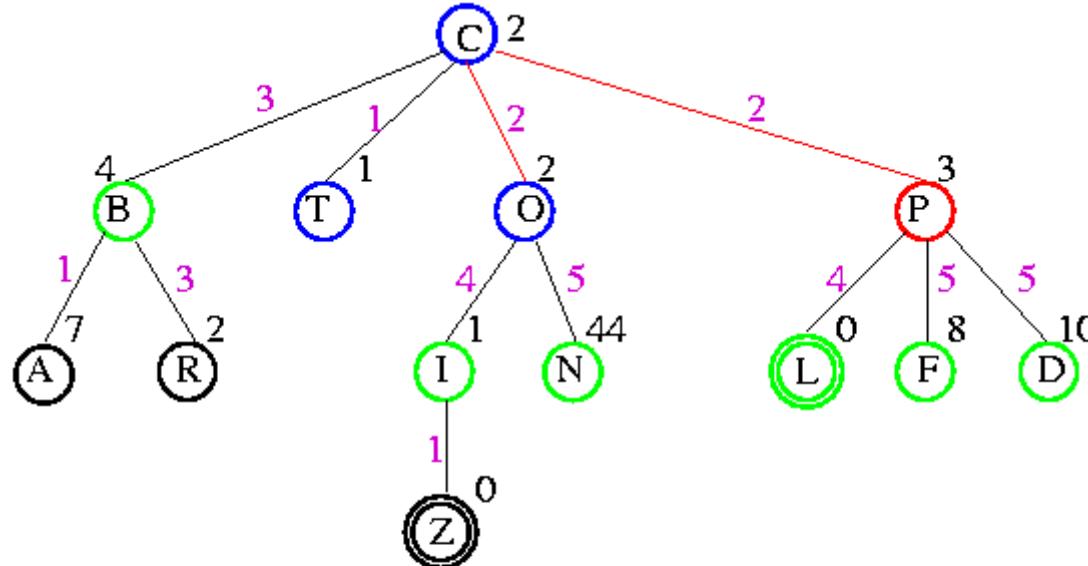
limit = $f(P) = 5$

ÖRNEK



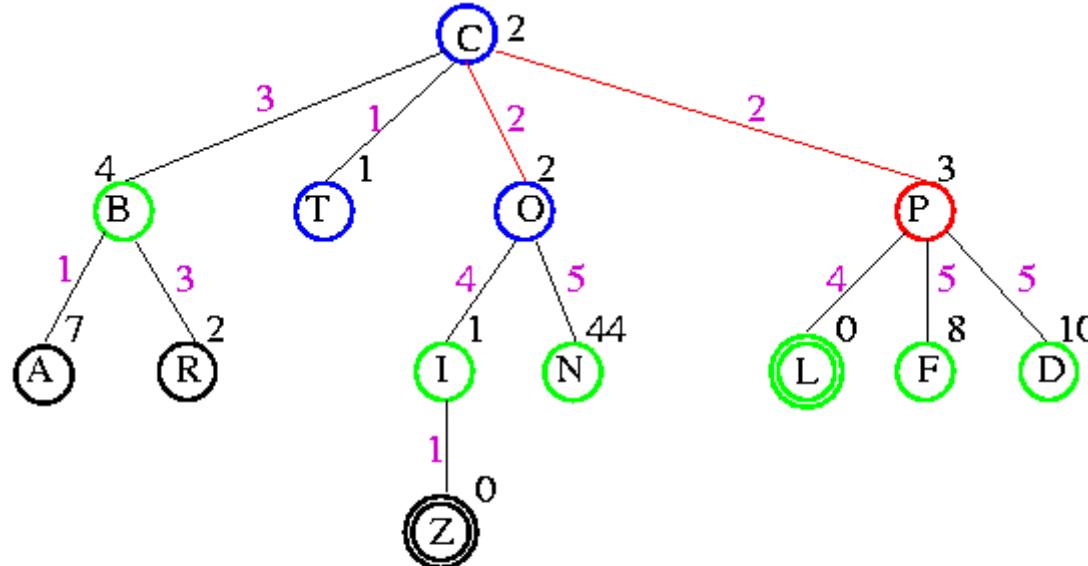
limit = $f(P) = 5$

ÖRNEK



limit = $f(P) = 5$

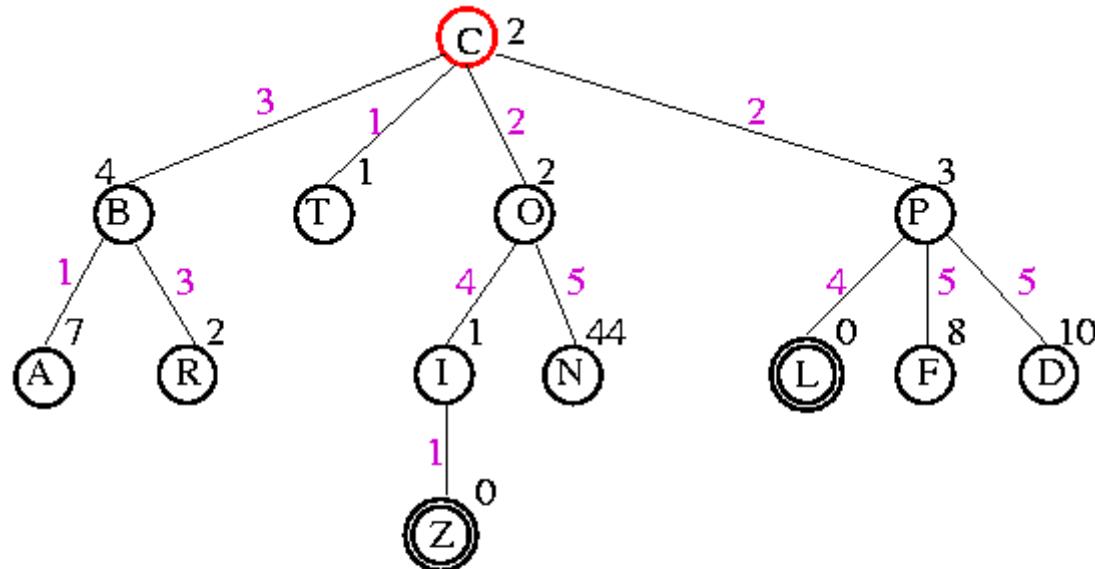
ÖRNEK



$$\text{limit} = f(L) = 6$$

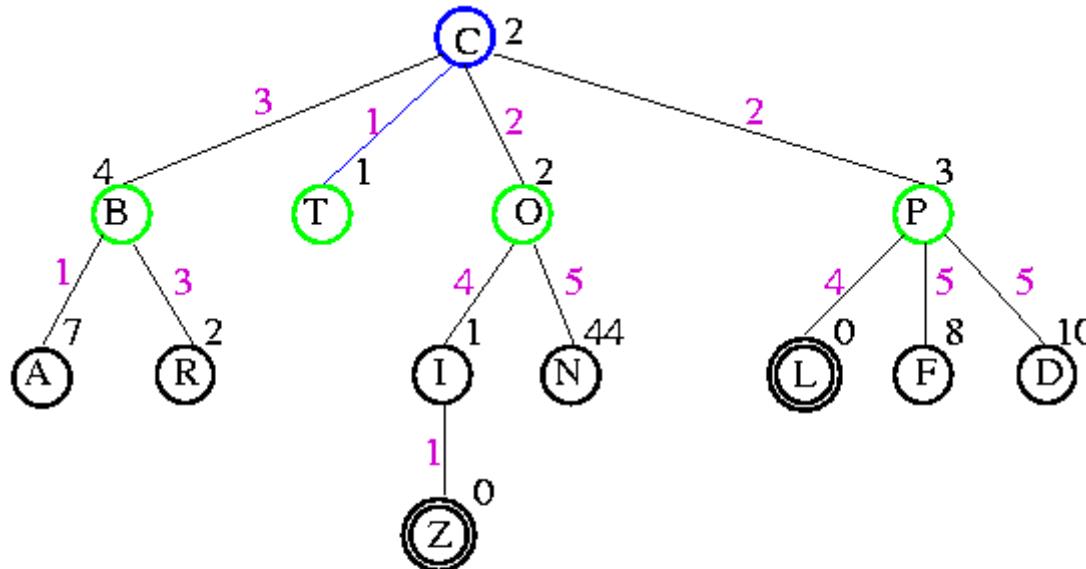
Nodes on frontier: B ($3+4=7$), I ($6+1=7$), N ($7+44=51$)
L ($6+0=6$), F ($7+8=15$), D ($7+10=17$)

ÖRNEK



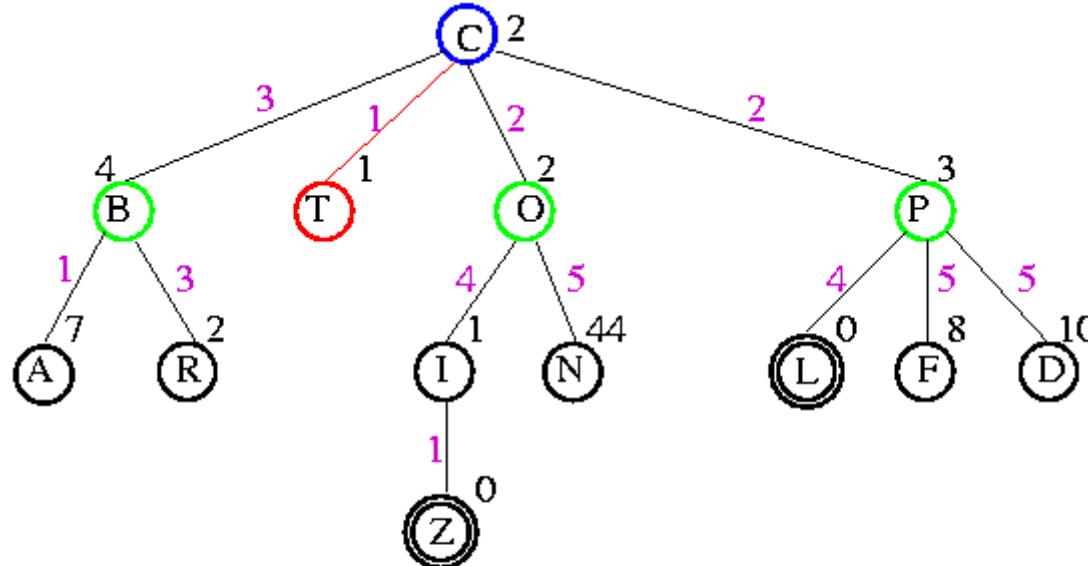
$\text{limit} = f(L) = 6$

ÖRNEK



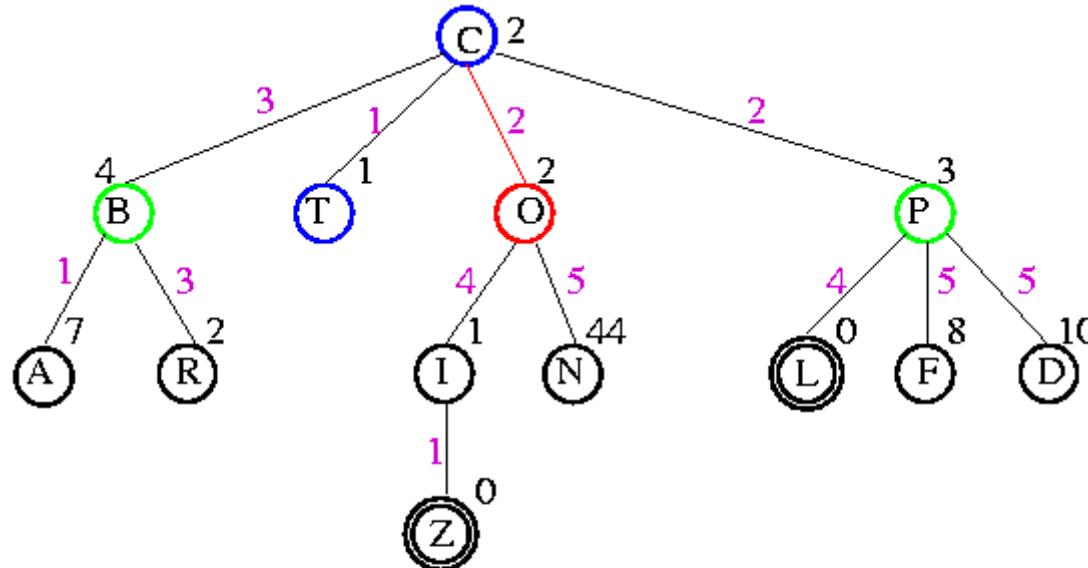
limit = $f(L) = 6$

ÖRNEK



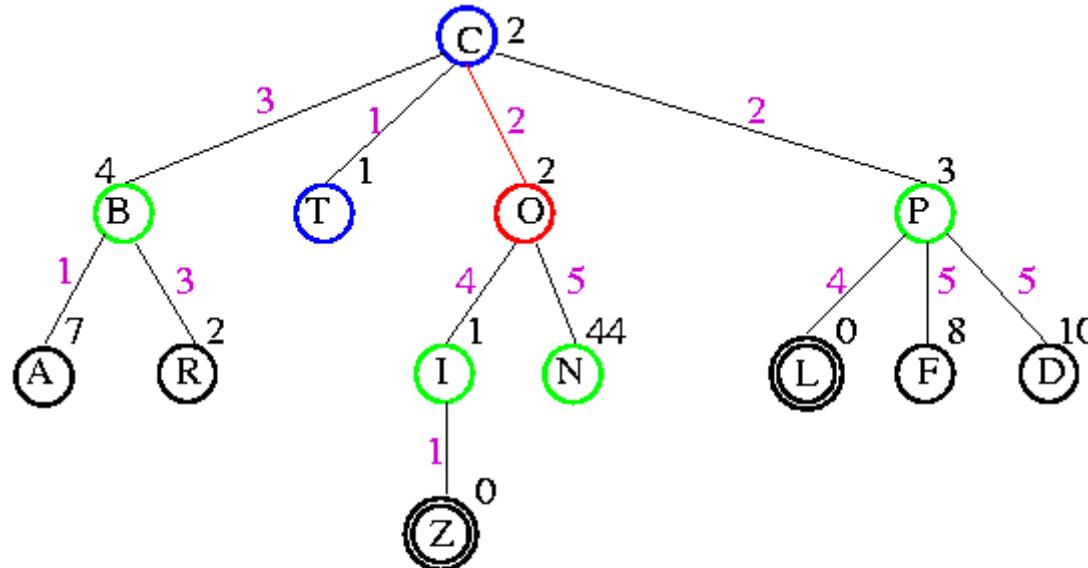
$\text{limit} = f(L) = 6$

ÖRNEK

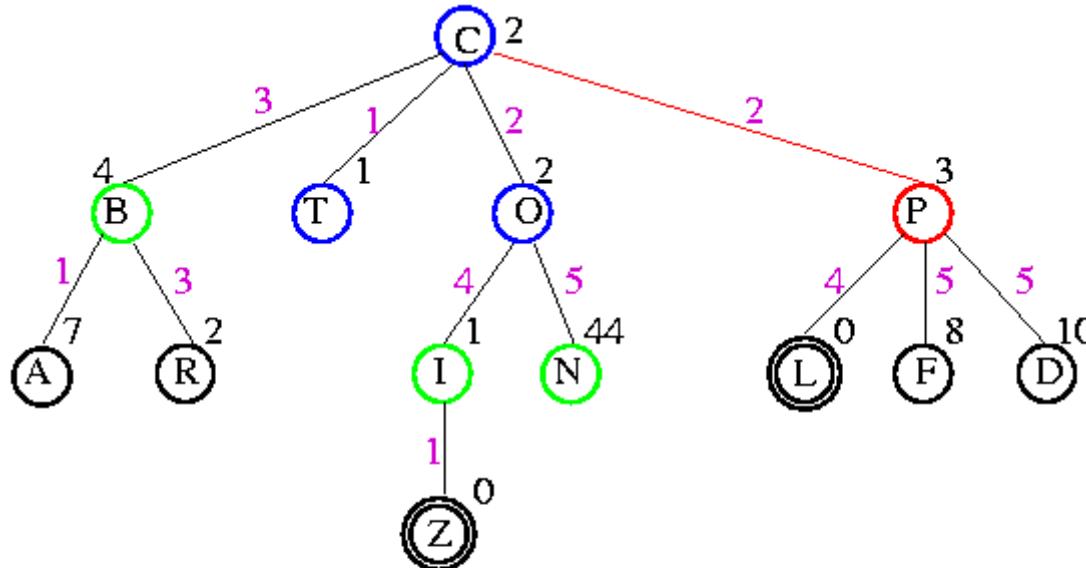


$\text{limit} = f(L) = 6$

ÖRNEK

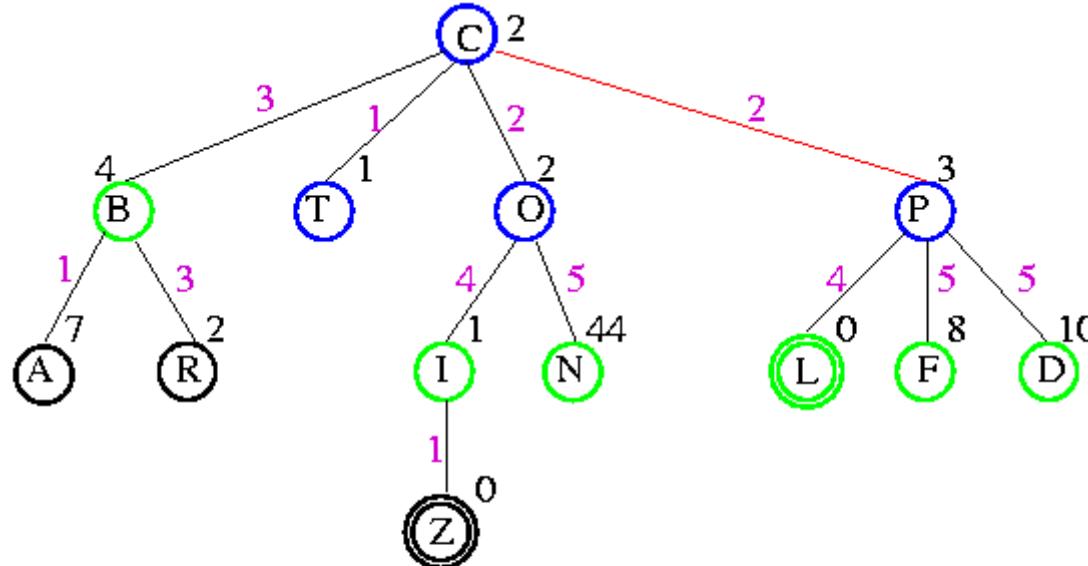


ÖRNEK



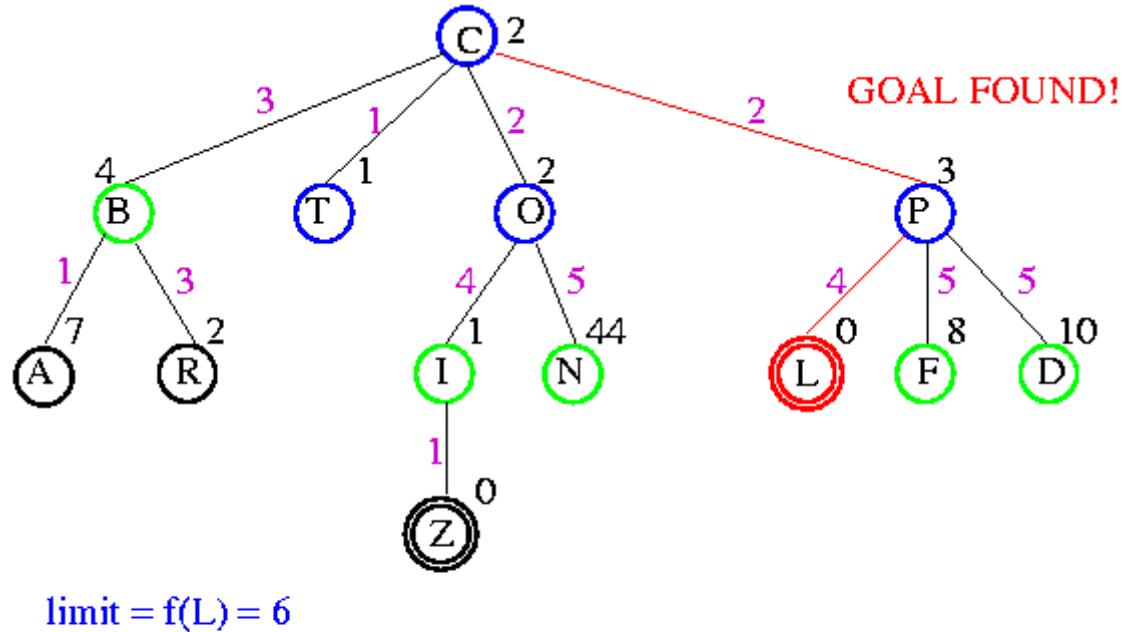
$\text{limit} = f(L) = 6$

ÖRNEK

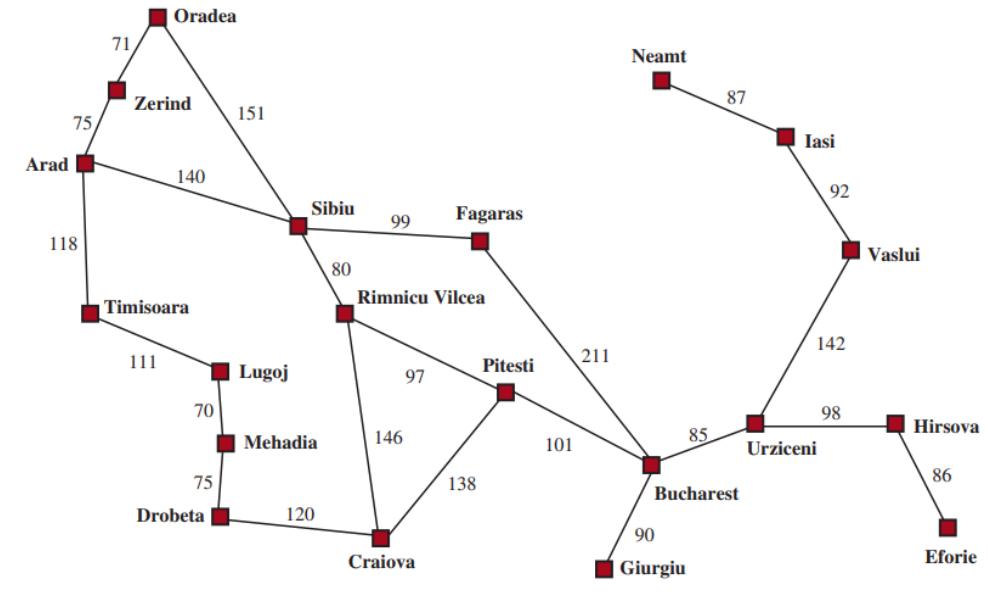
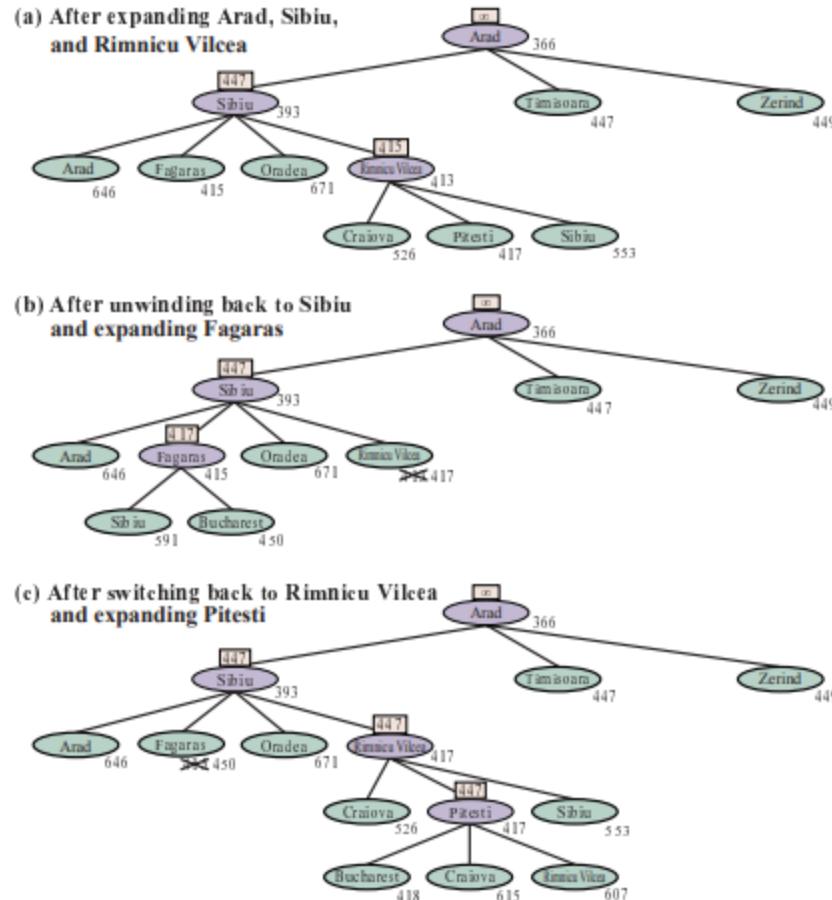


limit = $f(L) = 6$

ÖRNEK



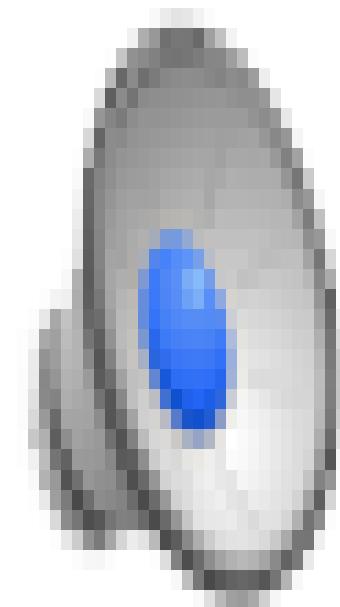
Bellek Sınırlı Sezgisel Arama Örnek



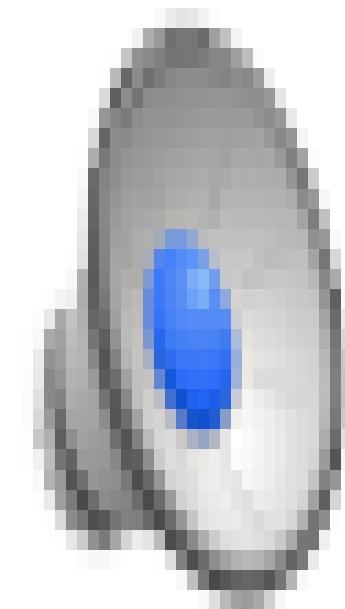
Arad	366	Mehadia	241
Bucharest	0	Neamt	234
Craiova	160	Oradea	380
Drobeta	242	Pitesti	100
Eforie	161	Rimnicu Vilcea	193
Fagaras	176	Sibiu	253
Giurgiu	77	Timisoara	329
Hirsova	151	Urziceni	80
Iasi	226	Vaslui	199
Lugoj	244	Zerind	374

Figure 3.16 Values of h_{SLD} —straight-line distances to Bucharest.

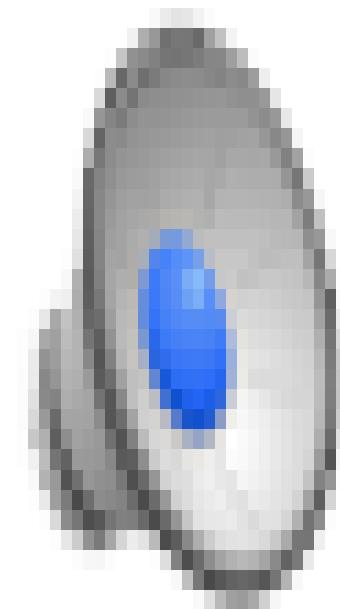
Video of Demo Contours (Empty) -- UCS



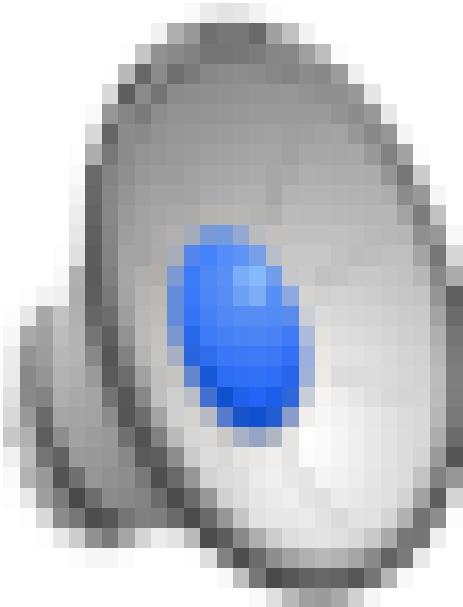
Video of Demo Contours (Empty) -- Greedy



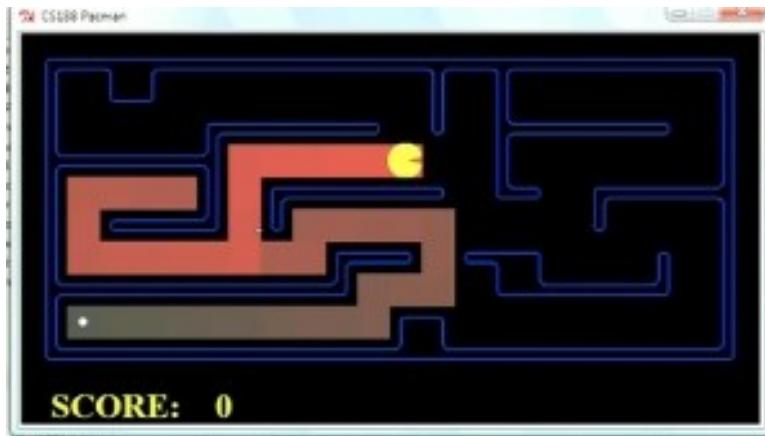
Video of Demo Contours (Empty) – A*



Küçük Labirentli Pacman Örneği – A*



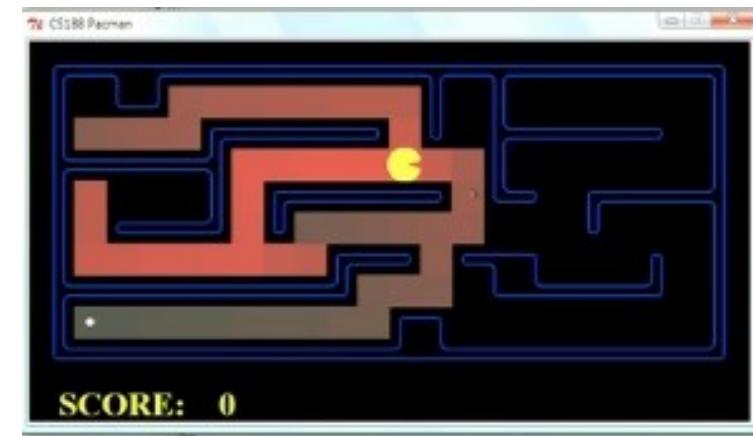
Karşılaştırma



Aç Gözlü

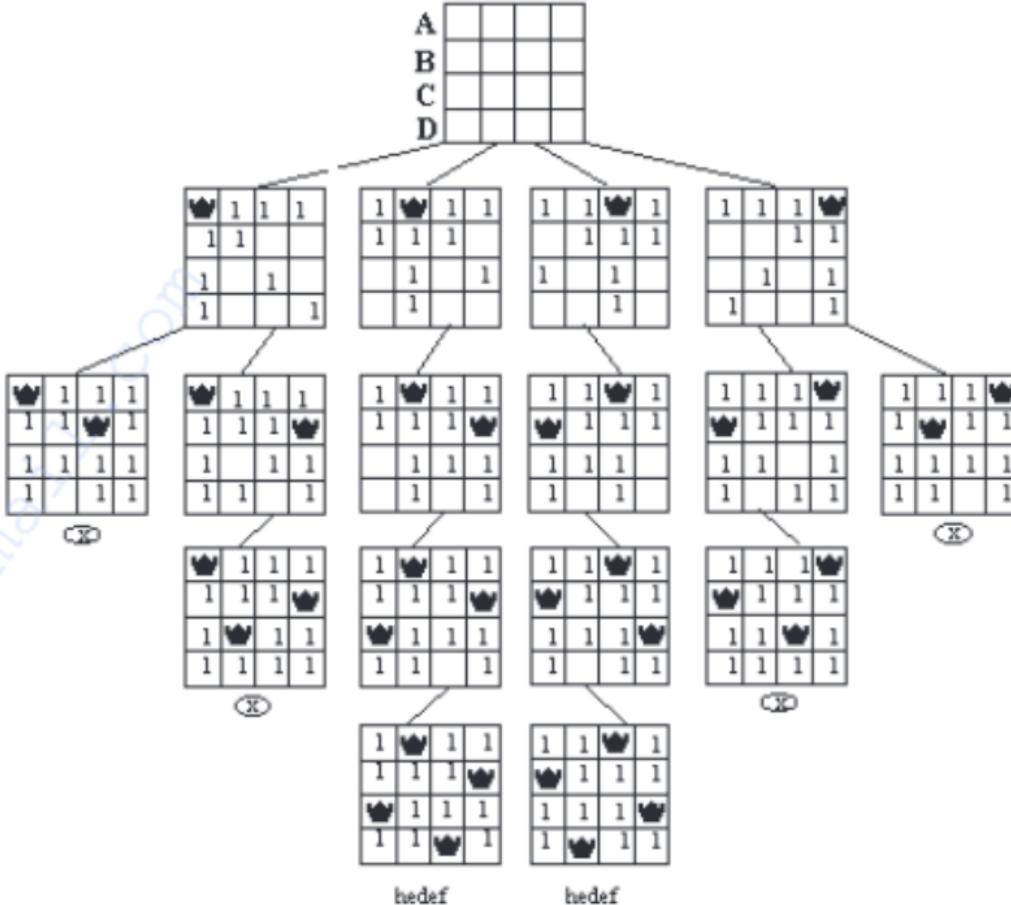


Eşit Maliyetli Arama



A*

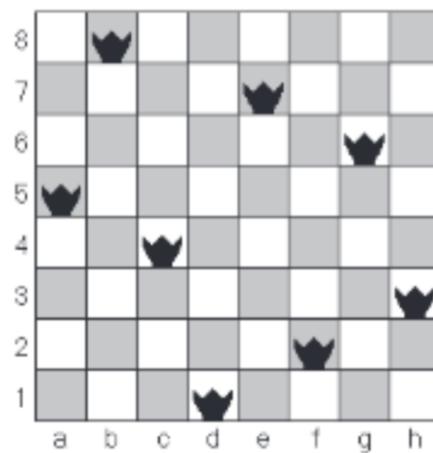
8 Vezir Problemi



N*N problem boyu	Çözüm sayısı	Tek çözümler
1*1	1	1
2*2	0	0
3*3	0	0
4*4	2	1
5*5	10	2
6*6	4	1
7*7	40	6
8*8	92	12
9*9	352	46
10*10	724	92
11*11	2,680	341
12*12	14,200	1,787
13*13	73,712	9,233
14*14	365,596	45,752
15*15	2,279,184	285,053
16*16	14,772,512	1,846,955
17*17	95,815,104	11,977,939
18*18	666,090,624	83,263,591
19*19	4,968,057,848	621,012,754
20*20	39,029,188,884	4,878,666,808
21*21	314,666,222,712	39,333,324,973
22*22	2,691,008,701,644	336,376,244,042
23*23	24,233,937,684,440	3,029,242,658,210
24*24	227,514,171,973,736	28,439,272,956,934
25*25	2,207,893,435,808,352	275,986,683,743,434
26*26	22,317,699,616,364,044	2,789,712,466,510,289
27*27	234,907,967,154,122,528	29,363,495,934,315,694

8 Vezir Problemi (Yaklaşım 1)

- Vezirlerin konumuna göre çözümün aranması
 - k satıra yerleştirilecek k tane vezir $S=\{y_1, y_2, \dots, y_k\}$ ile gösterilsin.
 - Bir önceki örnek durum için $S=\{d, f, h, c, a, g, e, b\}$
- Yatay yöndeki haneler minimum ağırlıklarına (serbestlik derecelerine) göre bakılacaktır.



8	2	1		2	1		
7		2	1		1		
6	1	1		1	2		
5		1	1	1		2	
4	1	1	1	1	1	1	1
3	2	1	1	2	2	2	
2	1	1	1	1	2	2	
1		1		1	2	2	
a	b	c	d	e	f	g	h

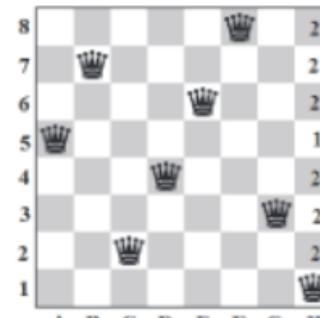
8	2	1	3		2	1	
7	3	2	1			1	
6	1	3	1		1	2	
5	3	1	1	1	3	2	3
4	1	1	1	1	1	1	1
3	2	1	1	1	2	2	2
2	1	1	3	1	2	2	
1		1	2	3	1		2
a	b	c	d	e	f	g	h

3	2	1	4		4	1	6	7
4	3	2	1	4	4	1	2	4
5	1	3	1	5	1	2	4	5
6	3	1	1	1	3	2	3	2
7	1	1	1	1	1	1	1	1
8	2	1	1	1	2	2	2	2
1	1	6	1	3	1	2	2	5
2	3	6	1	2	3	1	8	2
a	b	c	d	E	f	g	h	

8 Vezir Problemi (Yaklaşım 2)

- Sezgisel onarım yöntemi:

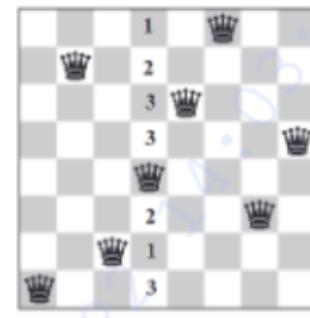
- 1-milyon vezir problemi yaklaşık 50 adımda çözülmektedir.
- İlk önce vezirlerin yerleşimleri rastgele yapılır.
 - Her satır ve sütuna birer tane olacak
- Her sütunda en az tehdit olan satır belirlenir.
- Vezir ilgili satıra taşınır.



(a)



(b)



(c)



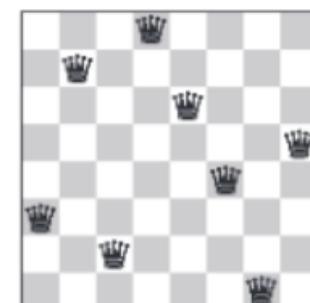
(d)



(e)

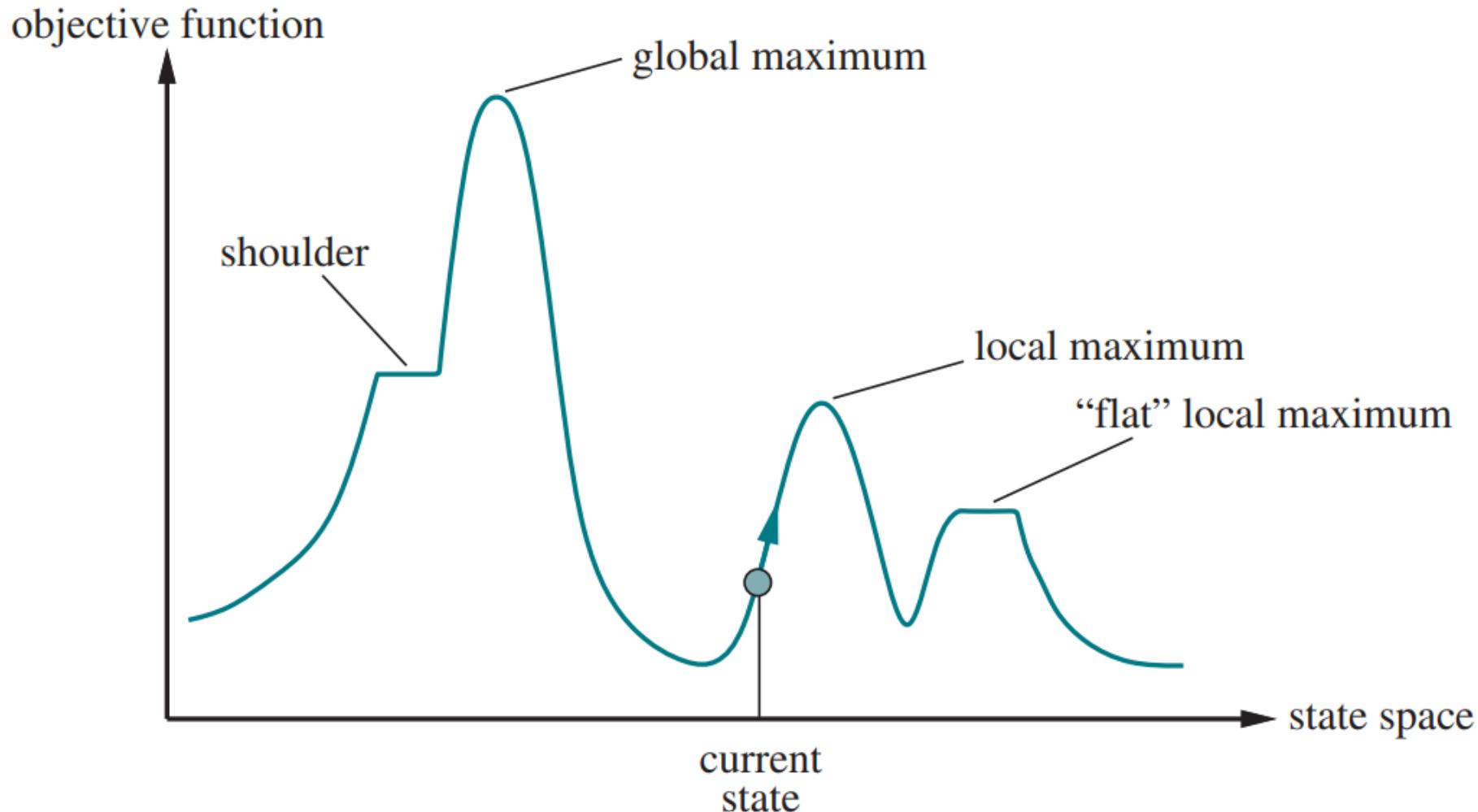


(f)



(g)

Yerel Arama Algoritmaları



Yerel Arama Algoritmaları

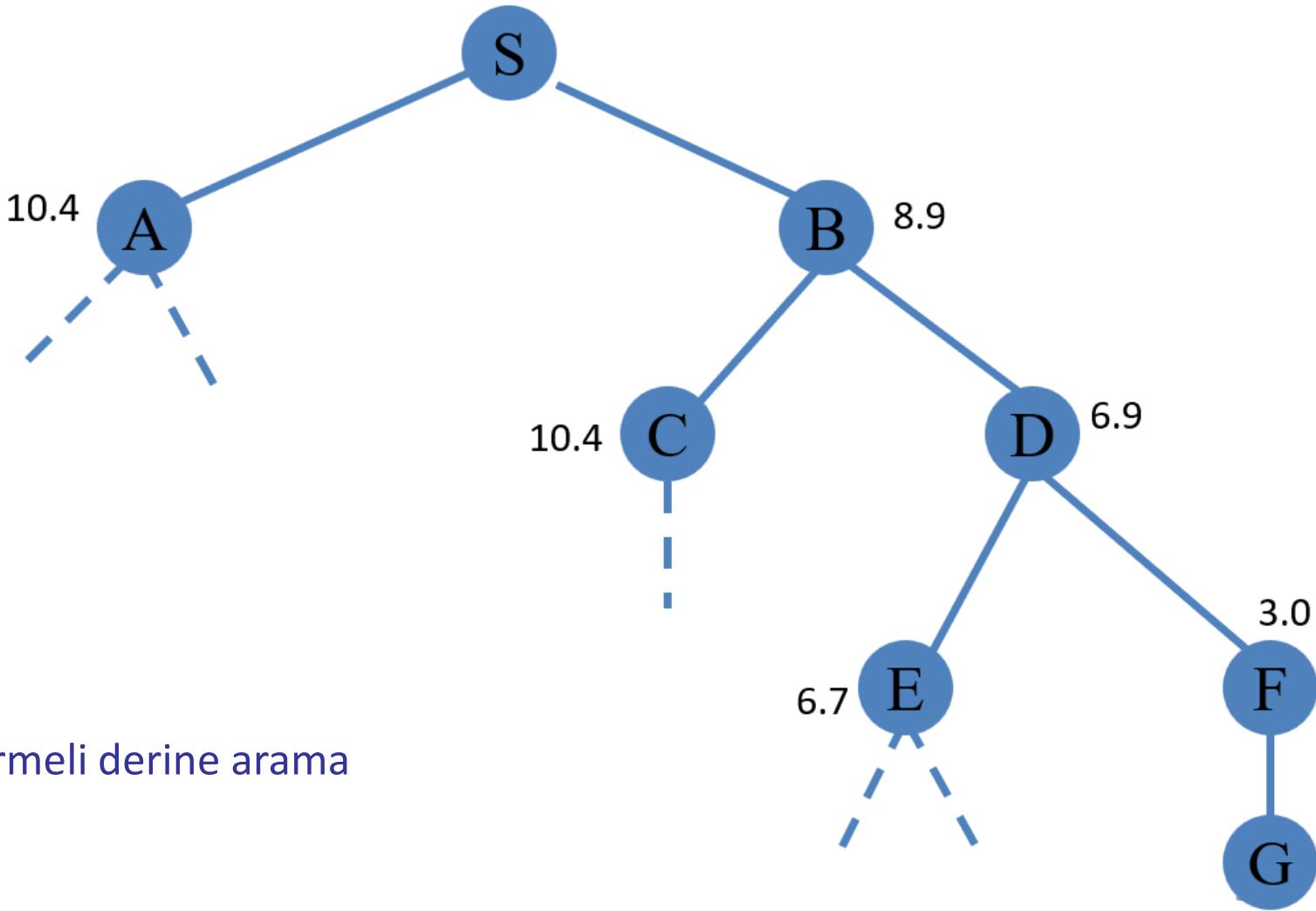
- Tepe Tırmanma
 - Hill Climbing
- Paralel Tepe Tırmanma
 - Local Beam Search
- Gradient Azalan Arama
 - Gradient Descent Search

Tepe Tırmanma Algoritması

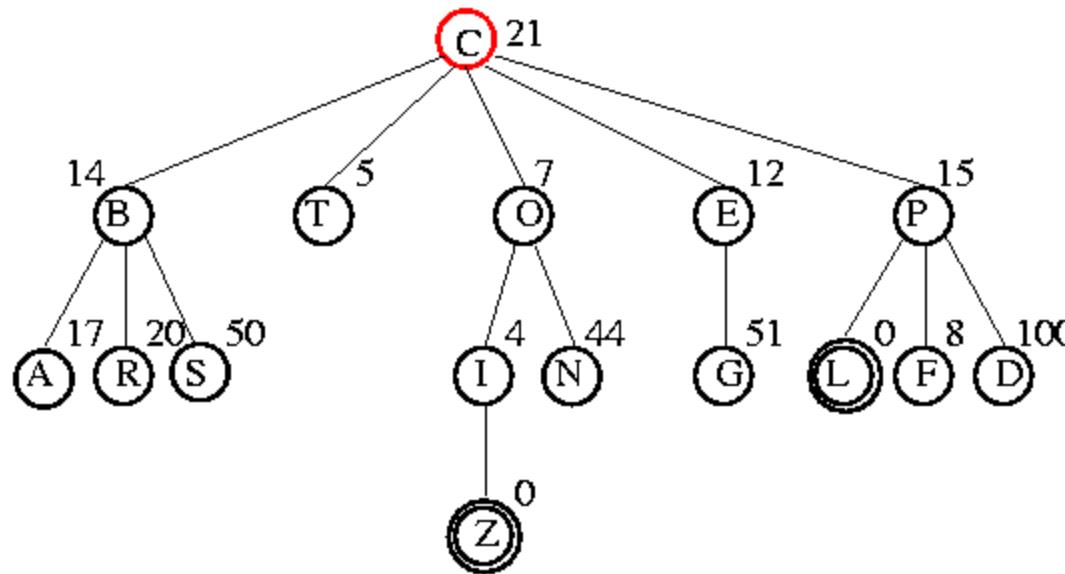
```
function HILL-CLIMBING(problem) returns a state that is a local maximum  
    current  $\leftarrow$  problem.INITIAL  
    while true do  
        neighbor  $\leftarrow$  a highest-valued successor state of current  
        if VALUE(neighbor)  $\leq$  VALUE(current) then return current  
        current  $\leftarrow$  neighbor
```



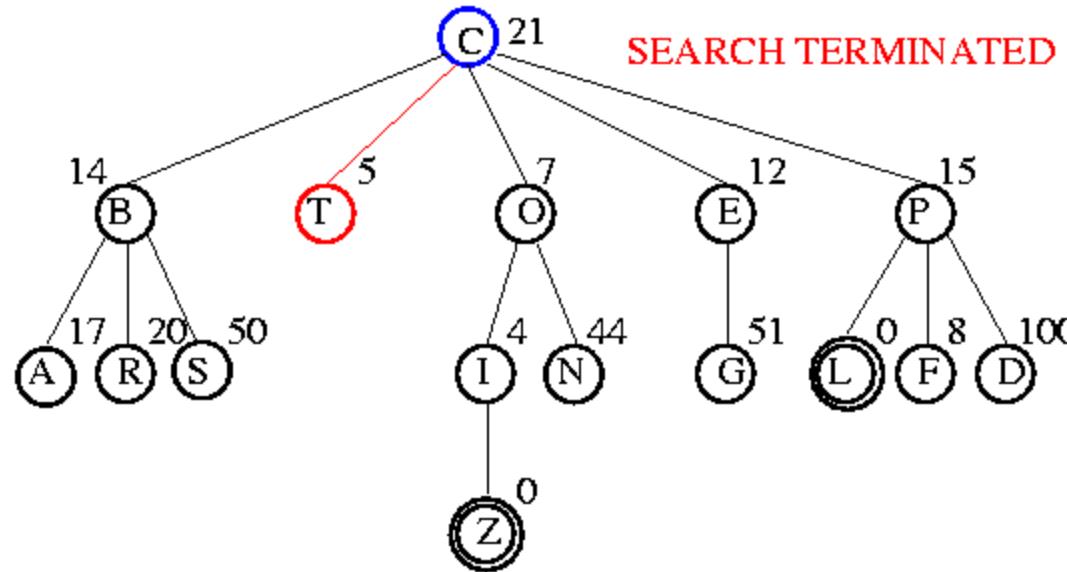
Tepe Tırmanma Algoritması



Tepe Tırmanma Algoritması Örnek

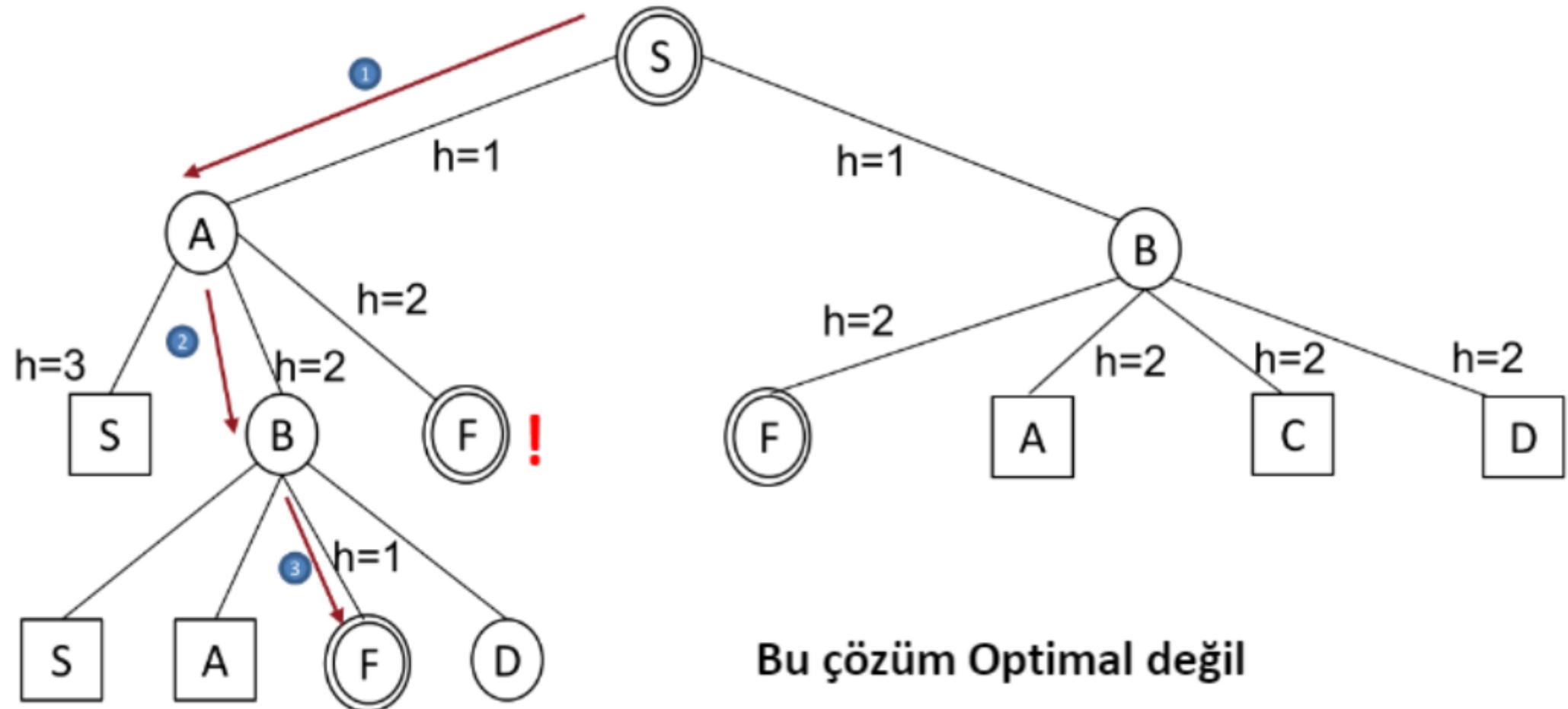


Tepe Tırmanma Algoritması Örnek



SEARCH TERMINATED

Tepe Tırmanma Algoritması Örnek

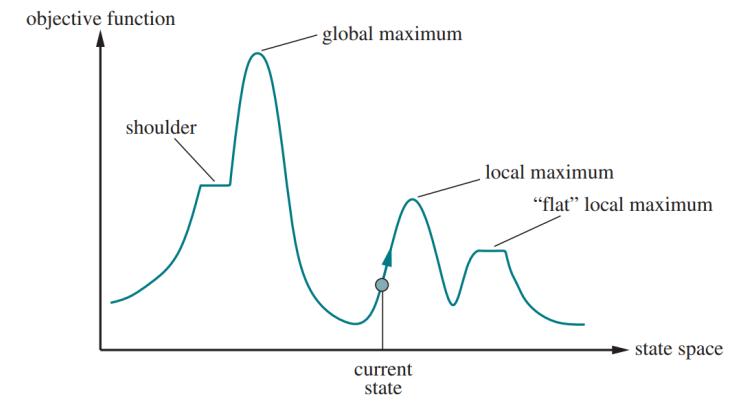


Tepe Tırmanma Algoritması

- Tamlık ?
 - Geri izleme (backtracking) yapmadıkça tam değildir !
- Zaman Karmaşıklığı ?
 - $O(\infty)$
- Uzay Karmaşıklığı ?
 - $O(b)$
- Optimallik ?
 - Yerel minimuma takılır !

Tepe Tırmanma Algoritması Problemleri:

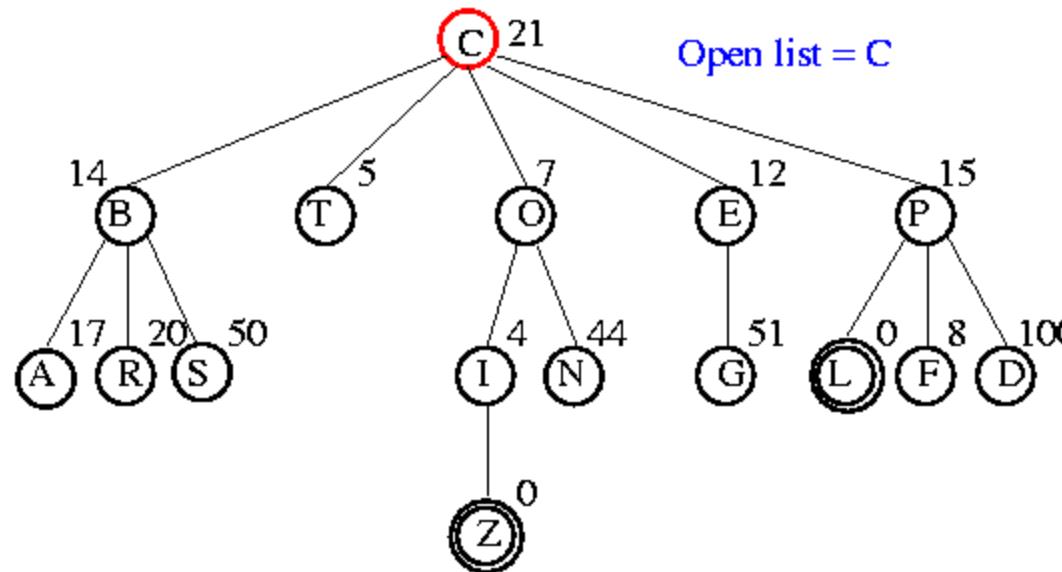
- Başlangıç durumuna bağlı olarak yerel maksimumda takılıp kalabilir
 - Yamaç ve Platolar
- Arama uzayında döngüler varsa kullanılmaz
- İyileştirme:
 - Algoritmayı farklı başlangıç noktalarıyla tekrarlama (Random restart hill climbing)
 - Benzetimli Tavlama (Simulated Annealing)
 - **Paralel Tepe Tırmanma (Local Beam Search)**



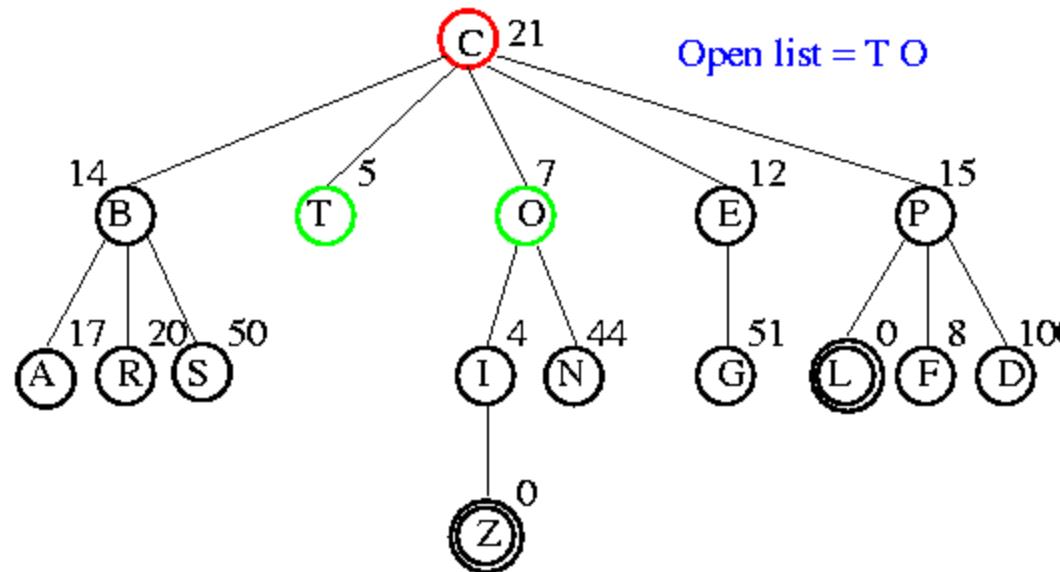
Paralel Tepe Tırmanma

- Değerlendirmek üzere tek bir durum saklamaktansa k adet durum saklar.
- Her adımda k durumu değerlendirilir,
- Sezgisel fonksiyona göre en iyi k tane düğüm seçilerek devam eder.
 - k : “beam genişliği”
 - $k = 1$
 - Tepe Tırmanma
 - $k = \text{sonsuz}$
 - En İyi Öncelikli Arama

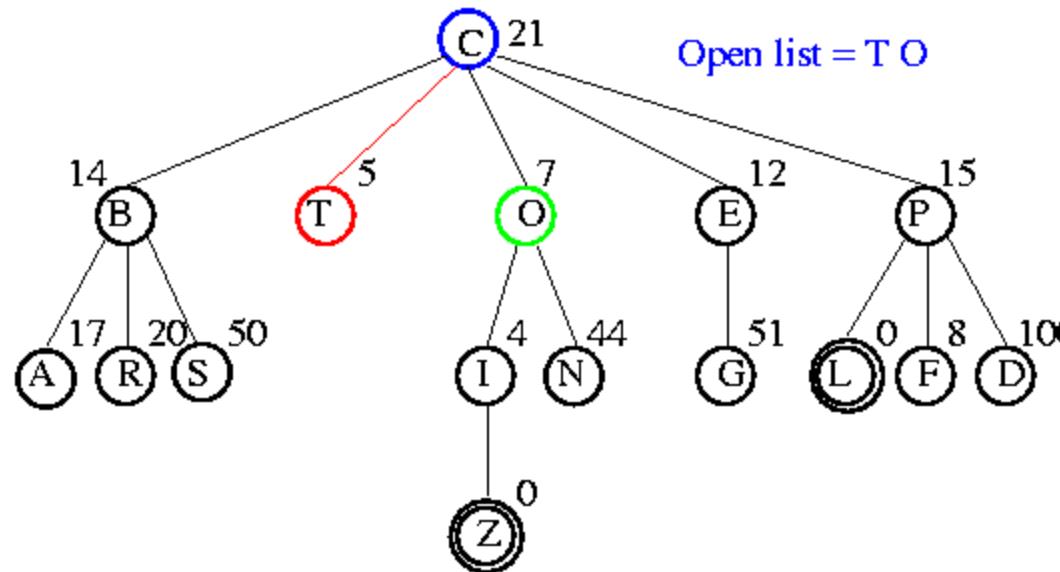
Paralel Tepe Tırmanma Örnek



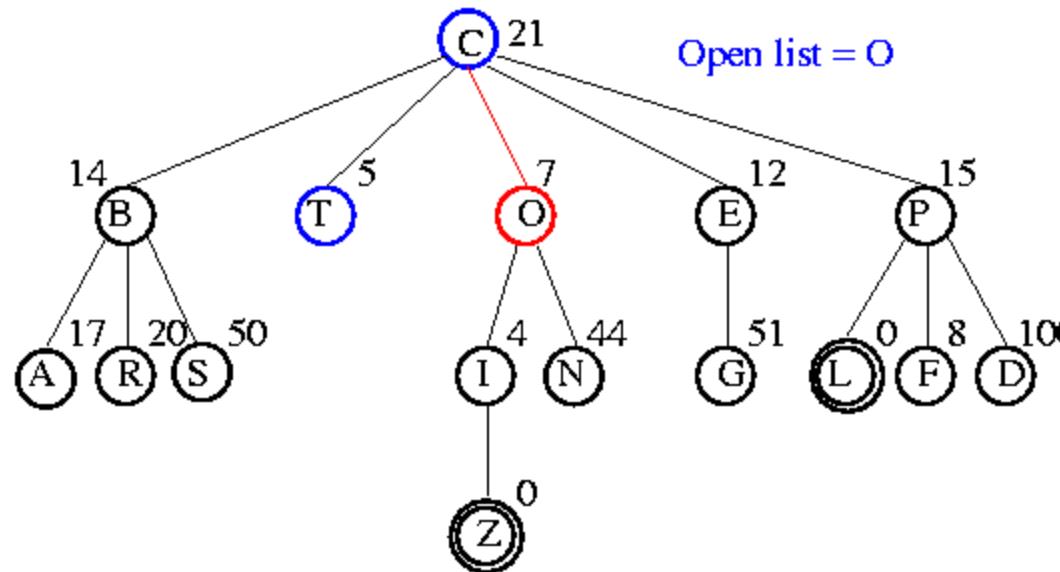
Paralel Tepe Tırmanma Örnek



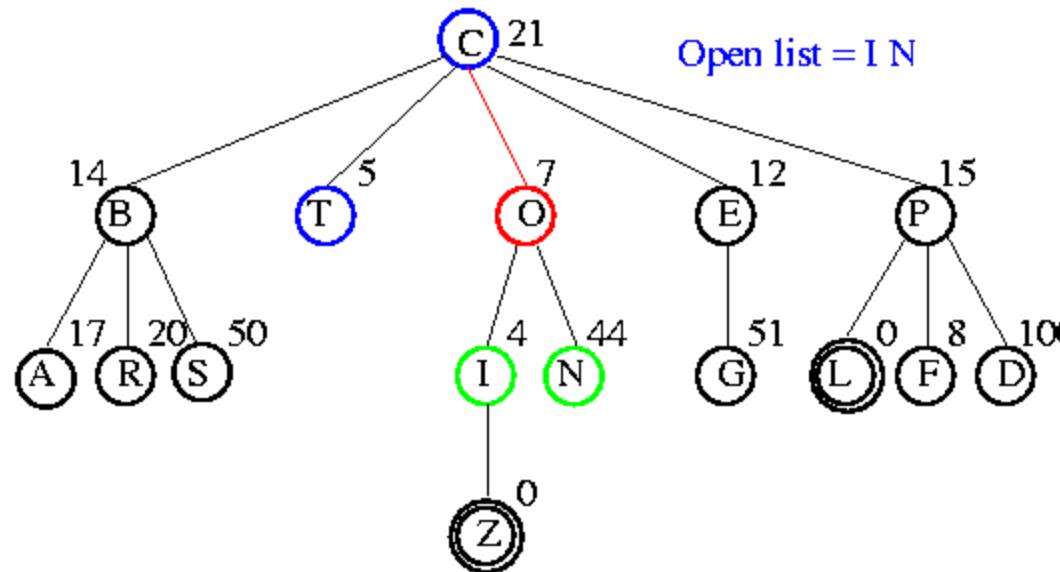
Paralel Tepe Tırmanma Örnek



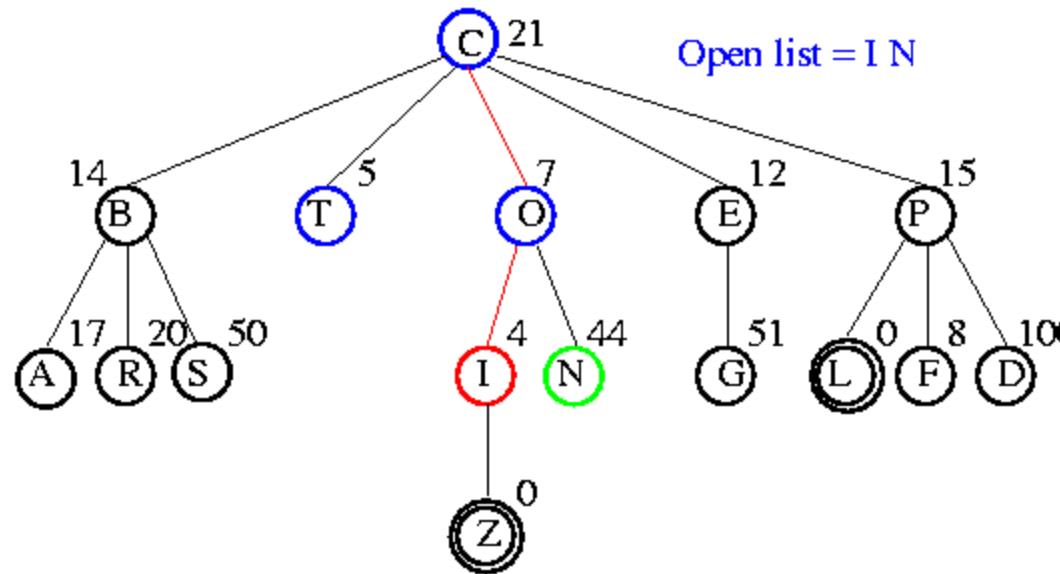
Paralel Tepe Tırmanma Örnek



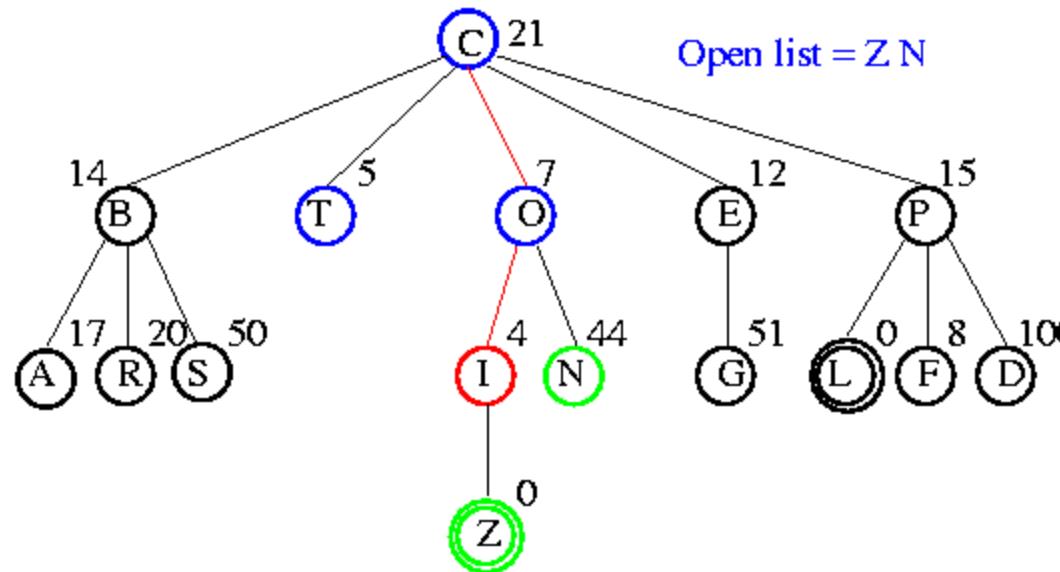
Paralel Tepe Tırmanma Örnek



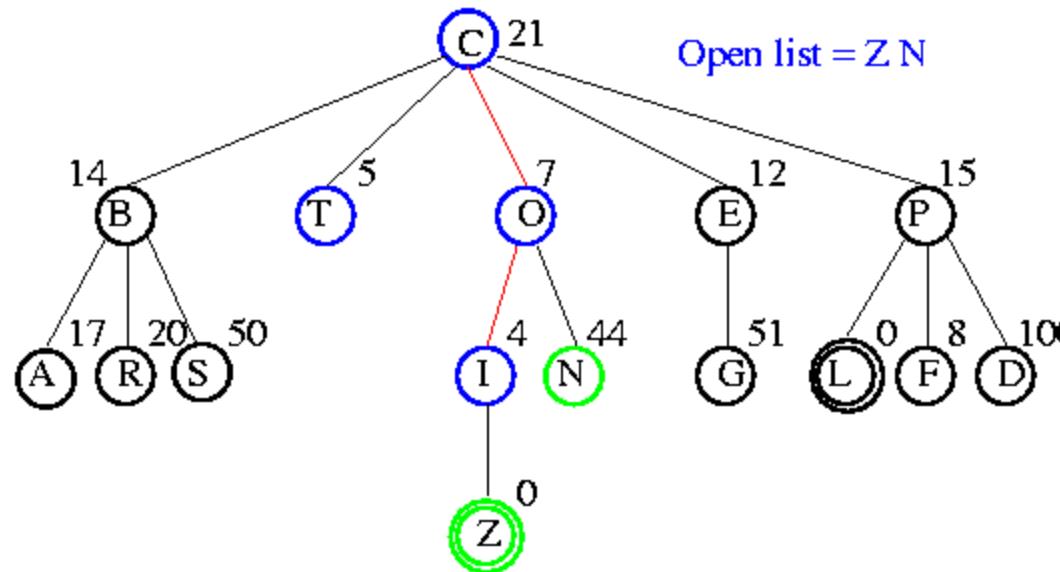
Paralel Tepe Tırmanma Örnek



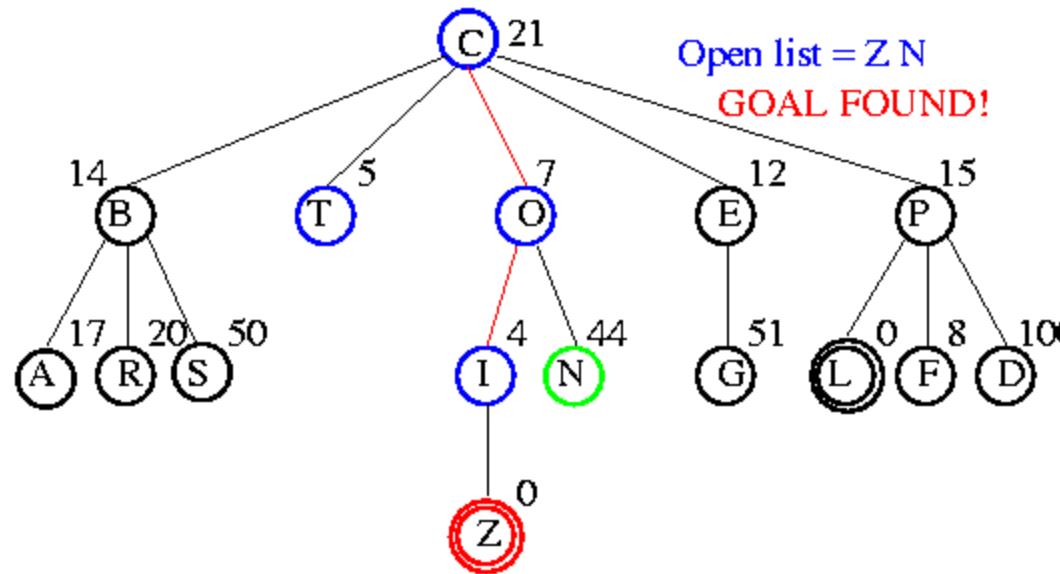
Paralel Tepe Tırmanma Örnek



Paralel Tepe Tırmanma Örnek



Paralel Tepe Tırmanma Örnek



Gradyan Azalan Arama

■ Giriş verileri:

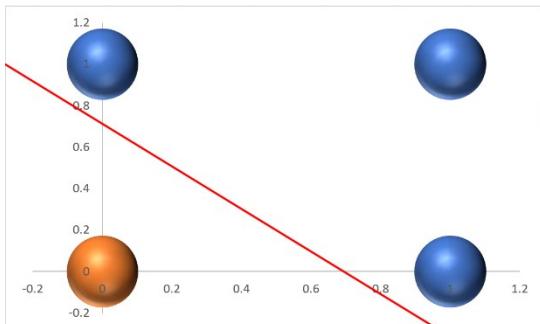
- Doğrusal Olarak Ayrıntırılabilir?

- XOR Problemi

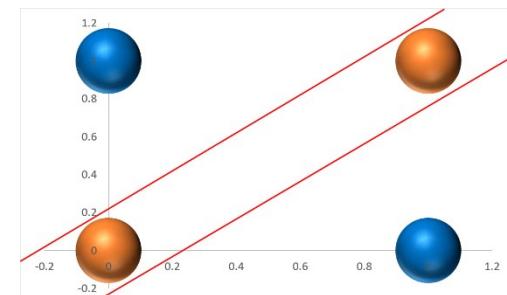
■ Öğrenme Kuralları:

- Algılayıcı Kuralı

- Delta Kuralı



A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1



A	B	A XOR B
0	0	0
0	1	1
1	0	1
1	1	0

Delta Kuralı

- Mantık:

- Beklenen çıktı ile gerçekleşen çıktı arasındaki farkları azaltmak için bağlantıların ağırlık değerlerinin sürekli olarak değiştirilmesi gerekmektedir.

- Hedef:

- Model ürettiği çıktı ile üretilmesi gereken çıktı arasındaki farkların karelerinin ortalamasını minimize etmek.
- Delta kuralı backpropagation algoritmasının temelini oluşturur.
- Eğitim örneklerine en uygun ağırlıkları bulmak için olası ağırlık vektörlerinin hipotez uzayını aramak için gradyan azalma yöntemini kullanmaktadır.

Delta Kuralı

- Doğrusal birim çıkışı:

$$o = w_0 + w_1x_1 + \cdots + w_nx_n$$

$$o(\vec{x}) = \vec{w} \cdot \vec{x}$$

- Öğrenme Hatası

$$E(\vec{w}) \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- D: Eğitim verileri kümesi
- t_d : d giriş verisi için hedef çıkış
- o_d : d örneği için doğrusal birim çıkışı

Gradyan Azalma

- Gradient azalma, bir başlangıç ağırlık vektörü ile başlayıp, sonraki adımlarda güncelliyerek E' 'yi minimum yapacak ağırlık vektörünü tanımlar.
 - W ağırlık vektörü ile başla
 - $E(W)$ yi minimum oluncaya kadar ağırlıkları değiştir.

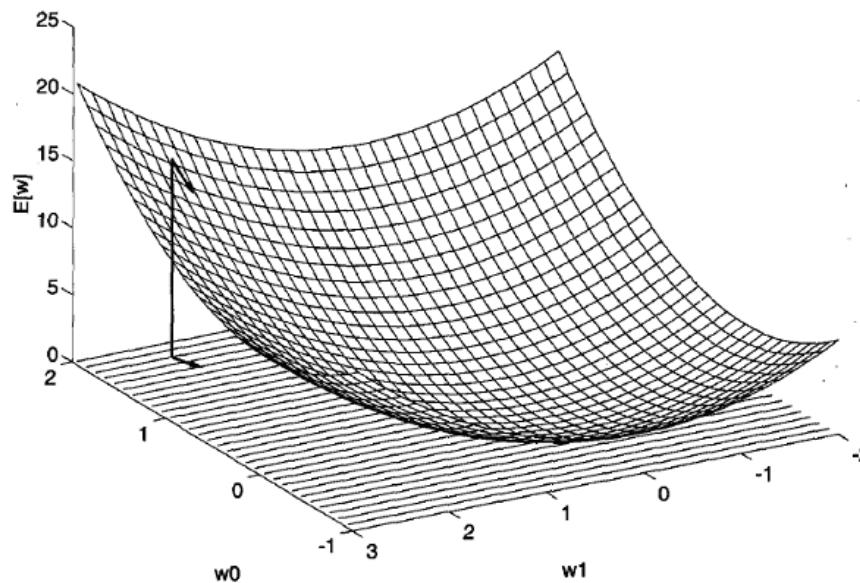
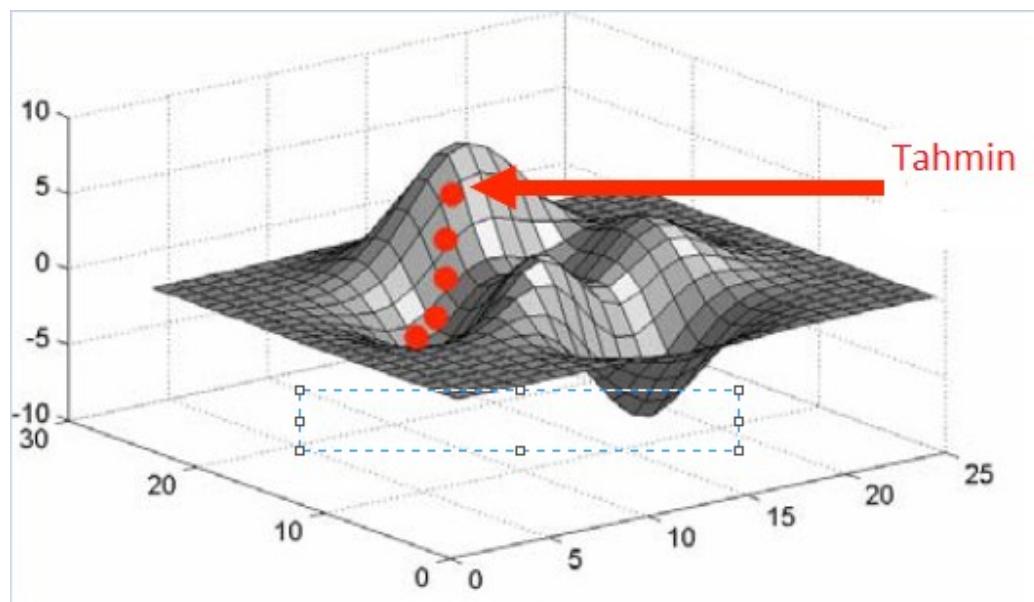


FIGURE 4.4

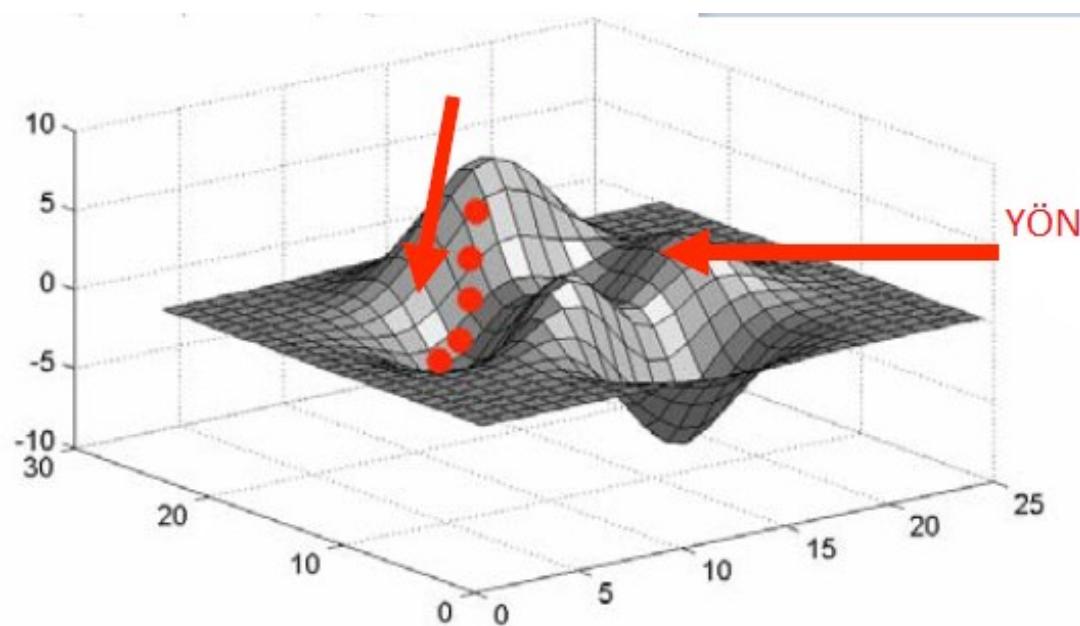
Gradyan Azalan Arama

- Bir nokta ile başla (tahmini noktası):



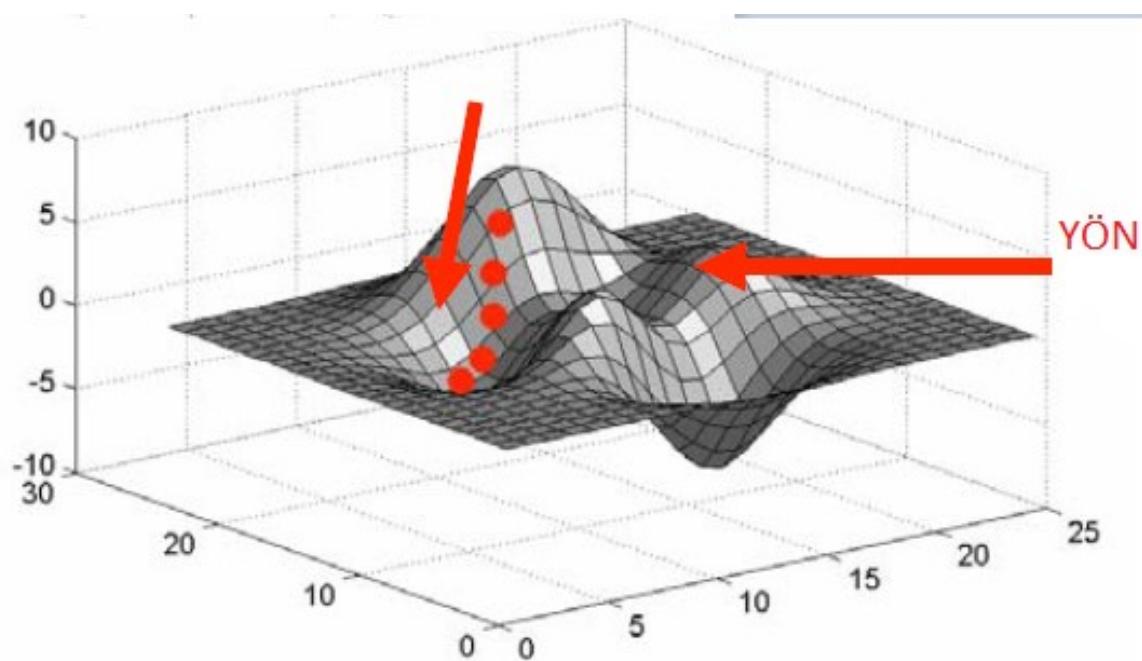
Gradyan Azalan Arama

- Azalma yönünü belirle:



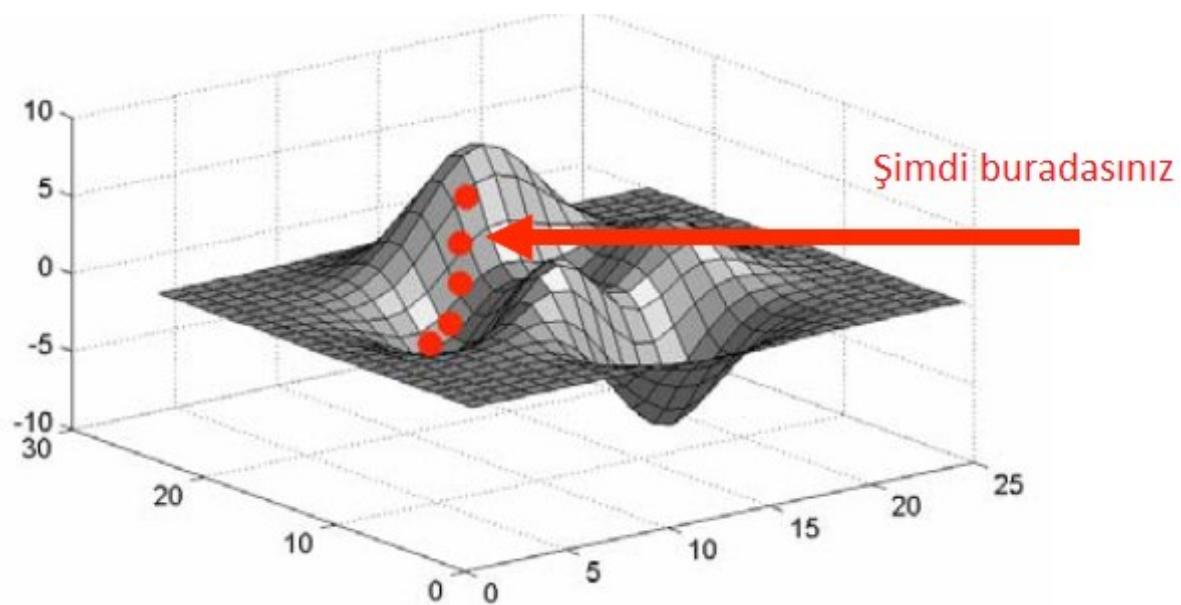
Gradyan Azalan Arama

- Bir adım seç:



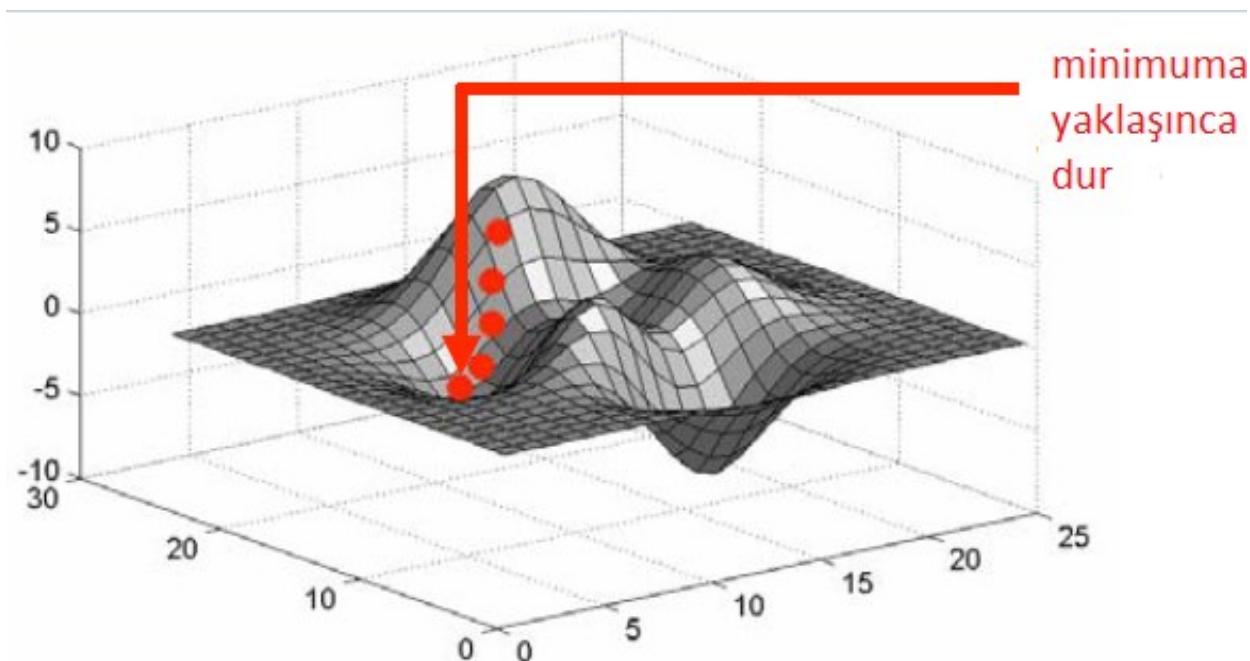
Gradyan Azalan Arama

- Güncelle



Gradyan Azalan Arama

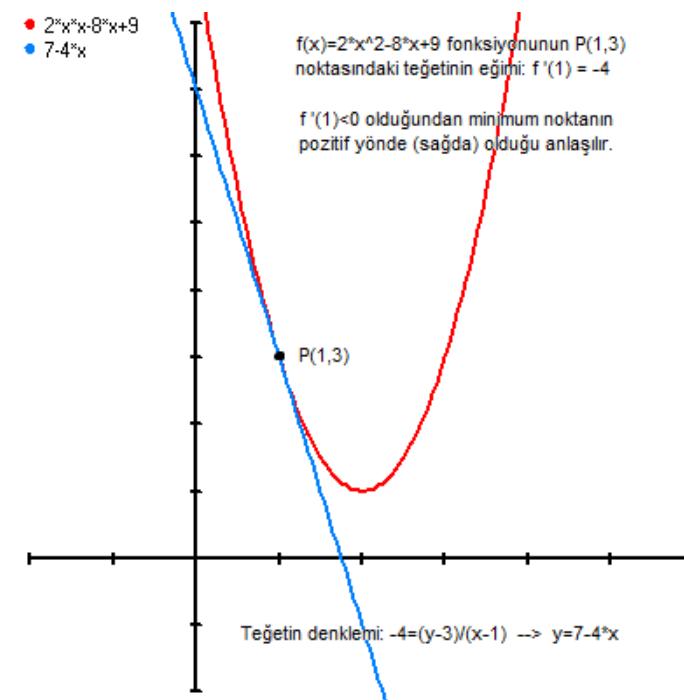
- Durdurma kriteri sağlanan kadar:



Hatadaki en dik düşüş?

- Hata yüzeyindeki en dik düşüşü nasıl hesaplayabiliriz?
 - Hatanın ağırlık vektörünün her bir bileşenine göre türevi alınarak bulunabilir.
- Bu türev vektörü, hatanın ağırlıklara göre gradient'i olarak adlandırılır.

$$\nabla E(\vec{w}) \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \dots, \frac{\partial E}{\partial w_n} \right]$$



Hatadaki en dik düşüş?

- Hata denklemi kullanarak: $E(w) = \frac{1}{2} \sum_{i=1}^m (h_w(x^{(i)}) - y^{(i)})^2$

W değerleri güncellenirken; W'den fonksiyonun türevi çıkarılır:

$$w_i \leftarrow w_i - \eta \frac{\partial}{\partial w_i} E(w) = w_i - \eta \frac{\partial E(w)}{\partial w_i} \quad (\text{her iterasyonda bu güncelleme yapılır.})$$

$$\begin{aligned} \frac{\partial}{\partial w_i} E(w) &= \frac{\partial}{\partial w_i} \frac{1}{2} (h_w(x) - y)^2 \quad (\text{tek giriş olduğu düşünülüyor}) \\ &= 2 \cdot \frac{1}{2} (h_w(x) - y) \frac{\partial}{\partial w_i} (h_w(x) - y) \\ &= (h_w(x) - y) \frac{\partial}{\partial w_i} (w_0 x_0 + \dots + w_i x_i + \dots + w_n x_n - y) \\ &= (h_w(x) - y) x_i \end{aligned}$$

buradan

$$w_i \leftarrow w_i - \eta (h_w(x) - y) x_i \quad \text{elde edilir.}$$

Birden çok eğitim örneği var ise (m tanesi için)

$$w_i \leftarrow w_i - \eta \sum_{j=1}^m (h_w(x^{(j)}) - y^{(j)}) x_i^j \quad (\text{batch gradient descent})$$

Algorithm 1 Gradient Descent

Initialize w_1

for $k = 1$ **to** K **do**

 Compute $\nabla F(w_k) = \sum_{i=1}^n \nabla f_i(w_k)$

 Update $w_{k+1} \leftarrow w_k - \alpha \nabla F(w_k)$

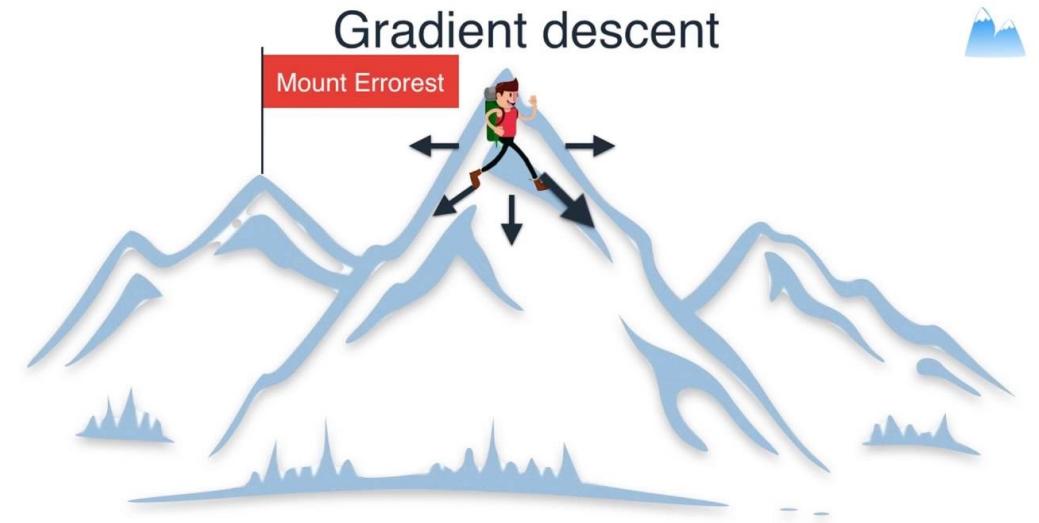
end for

Return w_K .

Tepe Tırmanma vs. Gradyan Azalan Arama



*Gradyan Azalan Arama
Tepe Tırmanma
Algoritmasının Sürekli
Arama Uzayında
Kullanılan Özel Bir
Versiyonudur.*



Özet

- Arama probleminde ilk adım amacın belirlenmesi ve problemin iyi formülize edilmesidir.
- Bir problem 5 kısımdan oluşmalı:

- Başlangıç durumu
- Eylemler
- Eylemlerin sonucunu tanımlayan geçiş modeli
- Amaç testi fonksiyonu
- Yol maliyet fonksiyonu

Problemin tüm çevre bileşenleri durum uzayı (state space),durum uzayında başlangıç durumdan amaç duruma çizilen path çözümüdür.

- Arama algoritmaları için değerlendirme ölçütleri:

- Tamlık
- Optimallik
- Zaman karmaşıklığı
- Yer karmaşıklığı

Karmaşıklık dallanma faktörü (b), ve arama derinliğine (d) göre değişir.

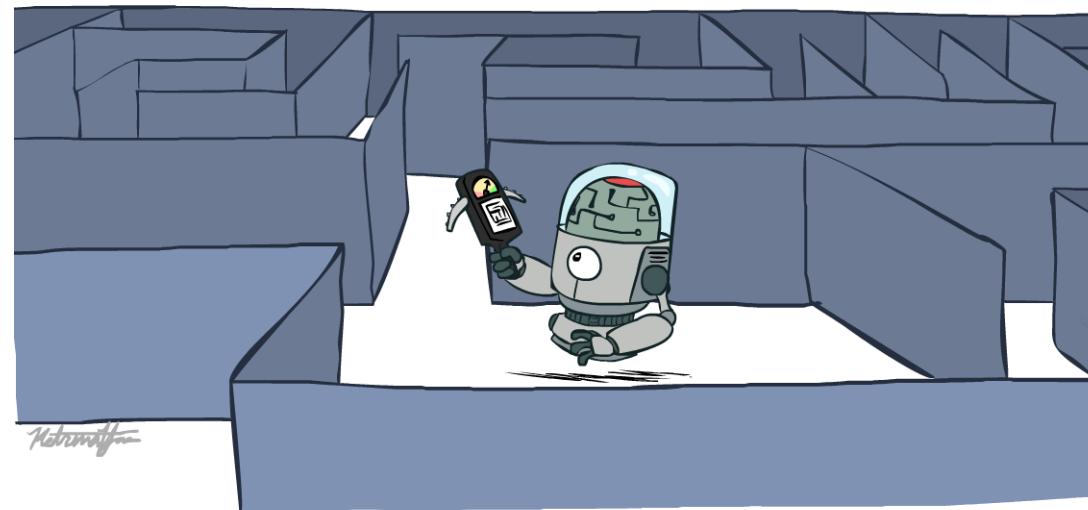
Özet

- **Bilgisiz Arama Teknikleri**
 - **Genişlik Öncelikli Arama**
 - Tamdır, complete, Tüm maliyetler 1 ise optimaldir, yer karmaşıklığı üsteldir.
 - **Sabit Maliyetli Arama**
 - Maliyeti en düşük yolu genişletir
 - Tam ve optimaldir
 - **Derinlik Öncelikli Arama**
 - Keşfedilmemiş en derindeki düğümü genişletir
 - Tam ve optimal değildir ama yer karmaşıklığı linerdir.
 - **Derinlik Sınırlı Arama**
 - Derinlik aramasına bir sınır ekler.
 - **Yinelemeli Derinleşen Arama**
 - Derinlik öncelikli aramayı artan derinlik sınırlarıyla amaca ulaşana kadar tekrarlar.
 - BFS gibi optimal ve tamdır!
 - DFS gibi az bellek gerektirir!
 - **İki Yönlü Arama**
 - Zaman karmaşıklığını büyük ölçüde azaltabilir, ancak her zaman uygulanabilir değildir ve çok fazla alan gerektirebilir.
- **Bilgili Arama Teknikleri**
 - **En İyi Öncelikli Arama**
 - Bir değerlendirme fonksiyonuna göre genişletecek düğümü seçer.
 - **Açgözlü En İyi Öncelikli Arama**
 - Düğümleri minimum $h(n)$ değerine göre genişletir.
 - Optimal değildir ama verimlidir.
 - **A* search Araması**
 - Düğümleri $f(n) = g(n) + h(n)$ fonksiyonuna göre genişletir.
 - Tamdır ve optimaldir.
 - $H(n)$ fonksiyonunun ağaç araması için Kabul edilebilir ve graf araması için tutarlı olması şartıyla
 - Yer karmaşıklığı kötüdür.
 - **RBFS (recursive best-first search) & SMA* (simplified memory-bounded A*)**
 - Sınırlı miktarda bellek kullanan optimal arama algoritmalarıdır.
 - Yeterli zaman verildiğinde A* algoritmasının çözemeyeceği problemleri çözerler.
 - A* algoritması tüm düğümleri hafızada tuttuğu için yetersiz kalır.

Sezgisel arama algoritmalarının performansı sezgisel fonksiyonun kalitesine bağlıdır !

Sonuç

- Sezgisel arama yöntemleri, problem hakkındaki bilgiden yararlanırlar.
- Sezgi (Heuristic), hedefe ulaşmak için kalan maliyetin tahminidir.
- İyi bir sezgi, arama süresini, üstelden doğrusala indirir.



Referanslar

- Artificial Intelligence A Modern Approach, Stuart Russell and Peter Norvig, Prentice Hall Series in Artificial Intelligence.
- Yapay Zeka, Vasif Vagifoğlu Nabihev, Seçkin Yayıncılık
- [Chapter 4 Informed search and Exploration \(bilkent.edu.tr\)](#)
- <http://mail.baskent.edu.tr/~tkaracay/etudio/agora/zv/2011/Hanoi.pdf>
- <https://dokumen.tips/documents/bilgili-arama-yoentemleri.html>
- <https://slideplayer.com/slide/272646/>
- <https://slideplayer.biz.tr/slide/14640497/>
- http://aytugonan.cbu.edu.tr/YZM3217/LectureNotes/YZM3217_lecture5.pdf
- <http://aima.cs.berkeley.edu/instructors.html>
- <https://slideplayer.biz.tr/slide/2660801/>
- <http://nek.istanbul.edu.tr:4444/ekos/TEZ/41491.pdf>
- <https://www.youtube.com/watch?v=ktGm0WCoQOg>