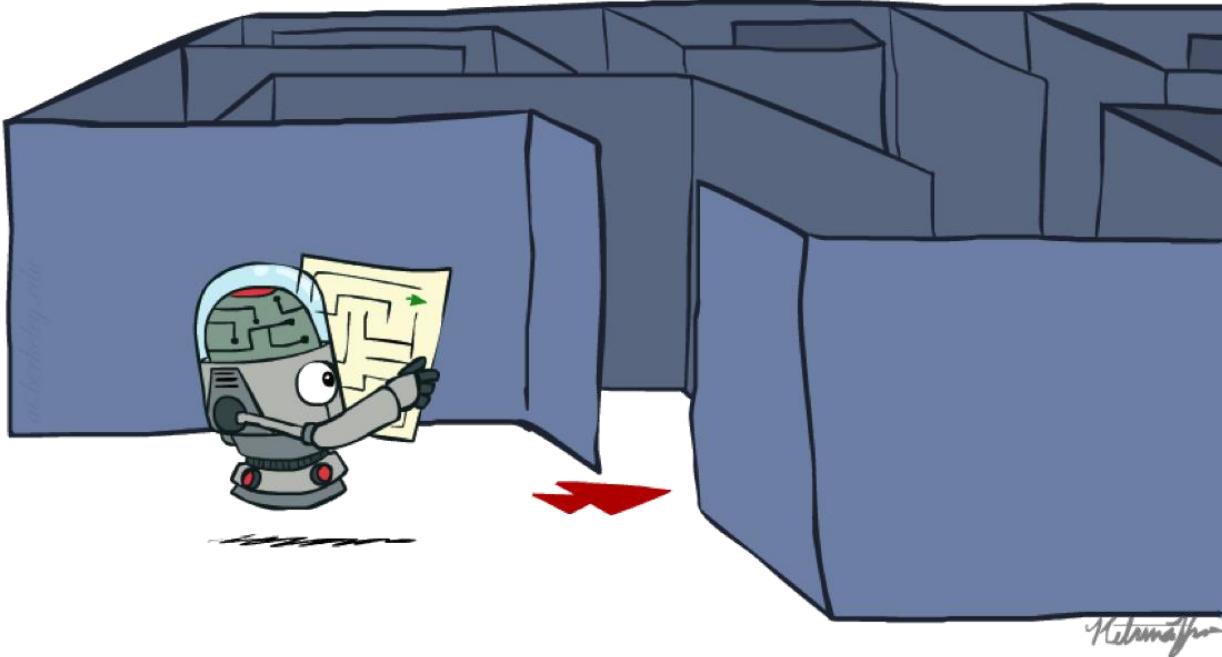


Yapay Zeka

Arama

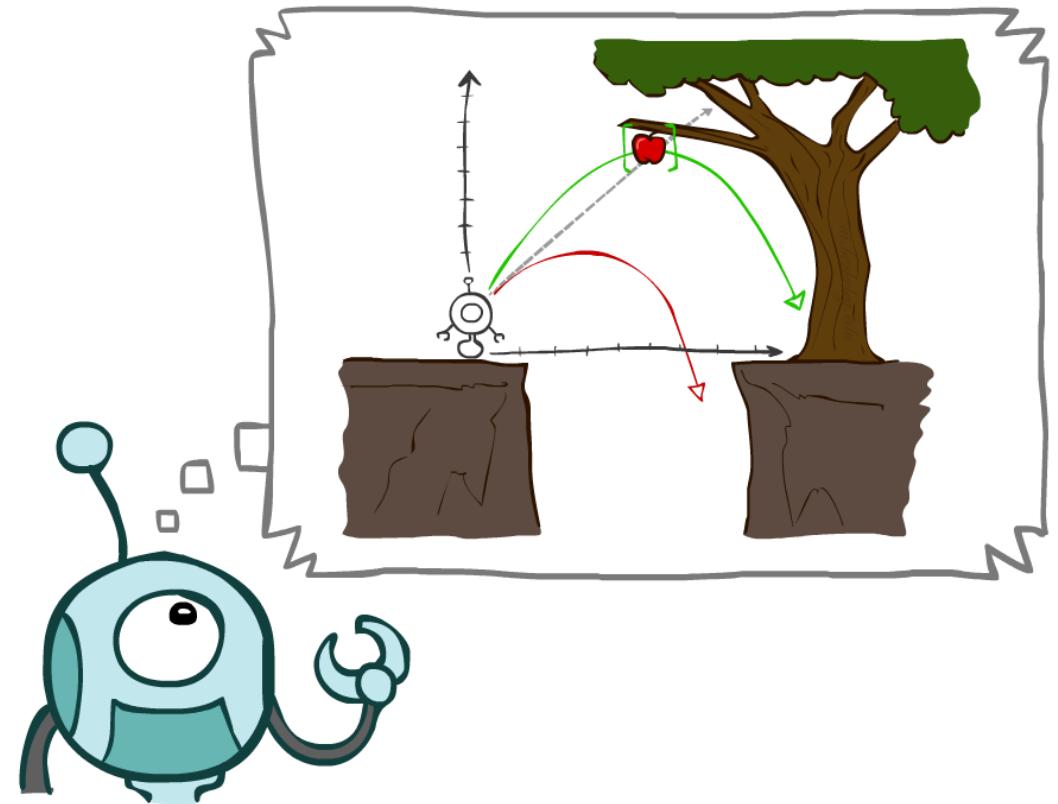


Prof. Sevinç İlhan Omurca / Dr. Öğretim Üyesi Fidan Kaya Gülağız
Kocaeli Üniversitesi

[These slides adapted from Dan Klein and Pieter Abbeel]

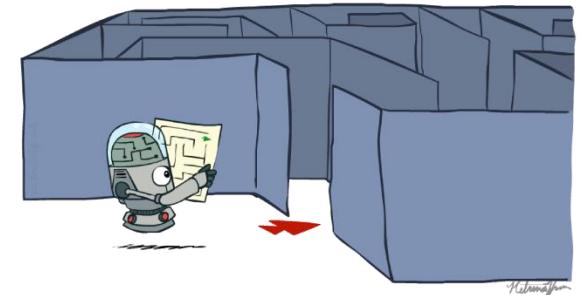
Today

- Arama
- Problem Çözme Ajanları
- Arama Problemi Tanımı
- Örnek Problemler
- Arama Uzayını Görselleştirme
- Problem Çözümü
- Durum Uzayı
- Bilgisiz Arama Algoritmaları

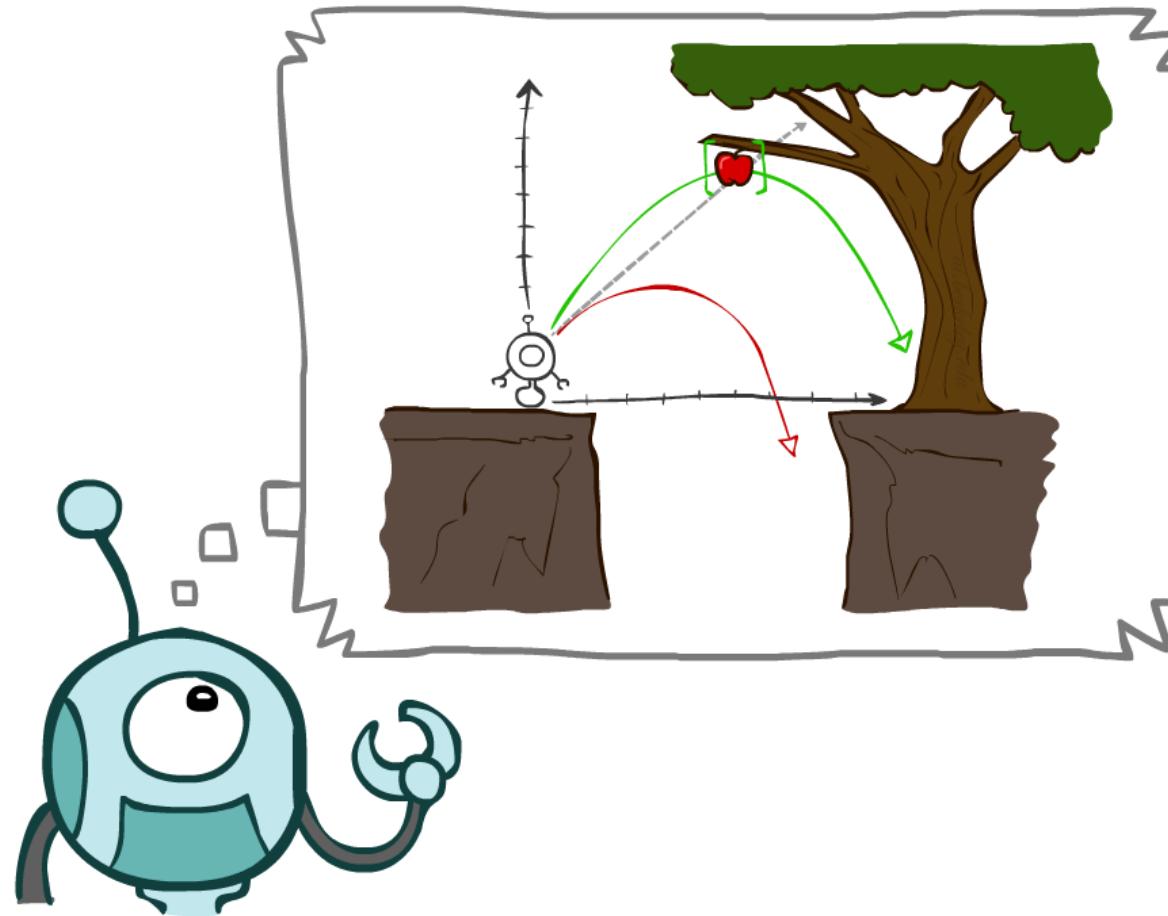


Arama

- Varsayımlar
 - Akıllı ajanlar performanslarını maksimum yapmaya çalışır
- Problem çözmedeki ilk adım?
 - Amaç Formülüzasyonu
- Arama?
 - En iyi eylemeler dizisini bulma işlemidir.
- Arama algoritması?
 - Problemi giriş olarak alır ve bir çözüm döndürür.



Problem Çözme Ajanları

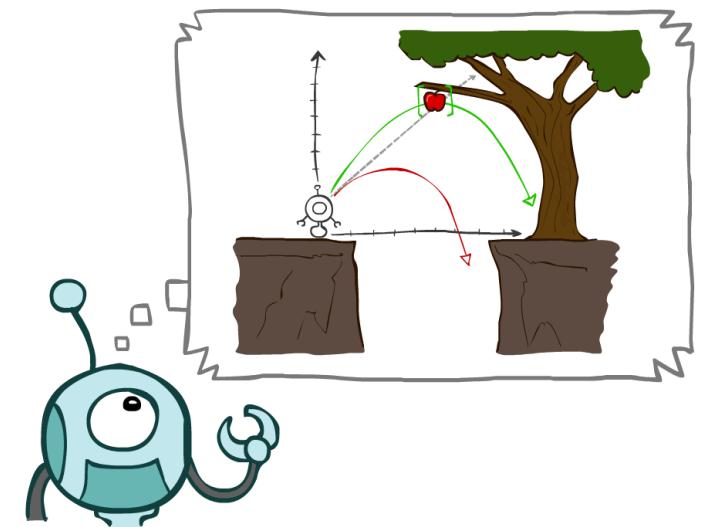


Problem Çözme Ajanları

- Problem çözme ajanları amaç temelli ajan tipidir.
- Ajan tasarıımı
 - “formüle et”, “ara”, “yürüt”

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  persistent: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation

  state  $\leftarrow$  UPDATE-STATE(state, percept)
  if seq is empty then
    goal  $\leftarrow$  FORMULATE-GOAL(state)
    problem  $\leftarrow$  FORMULATE-PROBLEM(state, goal)
    seq  $\leftarrow$  SEARCH(problem)
    if seq = failure then return a null action
  action  $\leftarrow$  FIRST(seq)
  seq  $\leftarrow$  REST(seq)
  return action
```



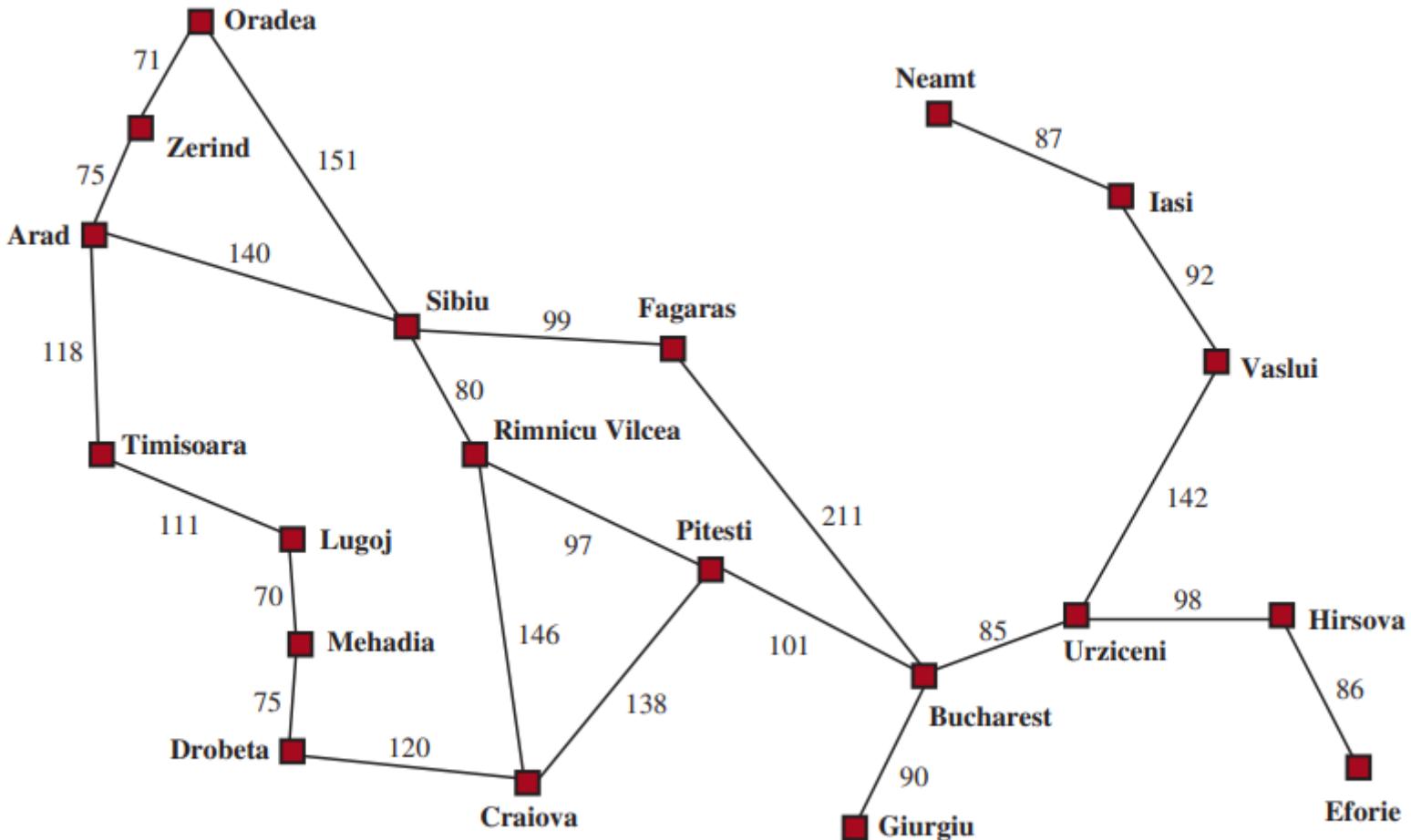
Arama Problemleri



Arama Problemi

- Bir arama problemi 5 bileşen ile tanımlanabilir:
 - Başlangıç durumu : Ajanın başlangıç durumu
 - Eylemler : Ajanın olası eylemleri
 - Geçiş Modeli
 - Durum uzayı :
 - Durum Uzayı Modeli = Başlangıç durumu + Eylemler + Geçiş modeli
 - Amaç Testi
 - Maliyet
 - Adım Maliyeti
 - Yol Maliyeti
- Çözüm, başlangıç durumunu bir hedef durumuna dönüştüren bir dizi eylemdir

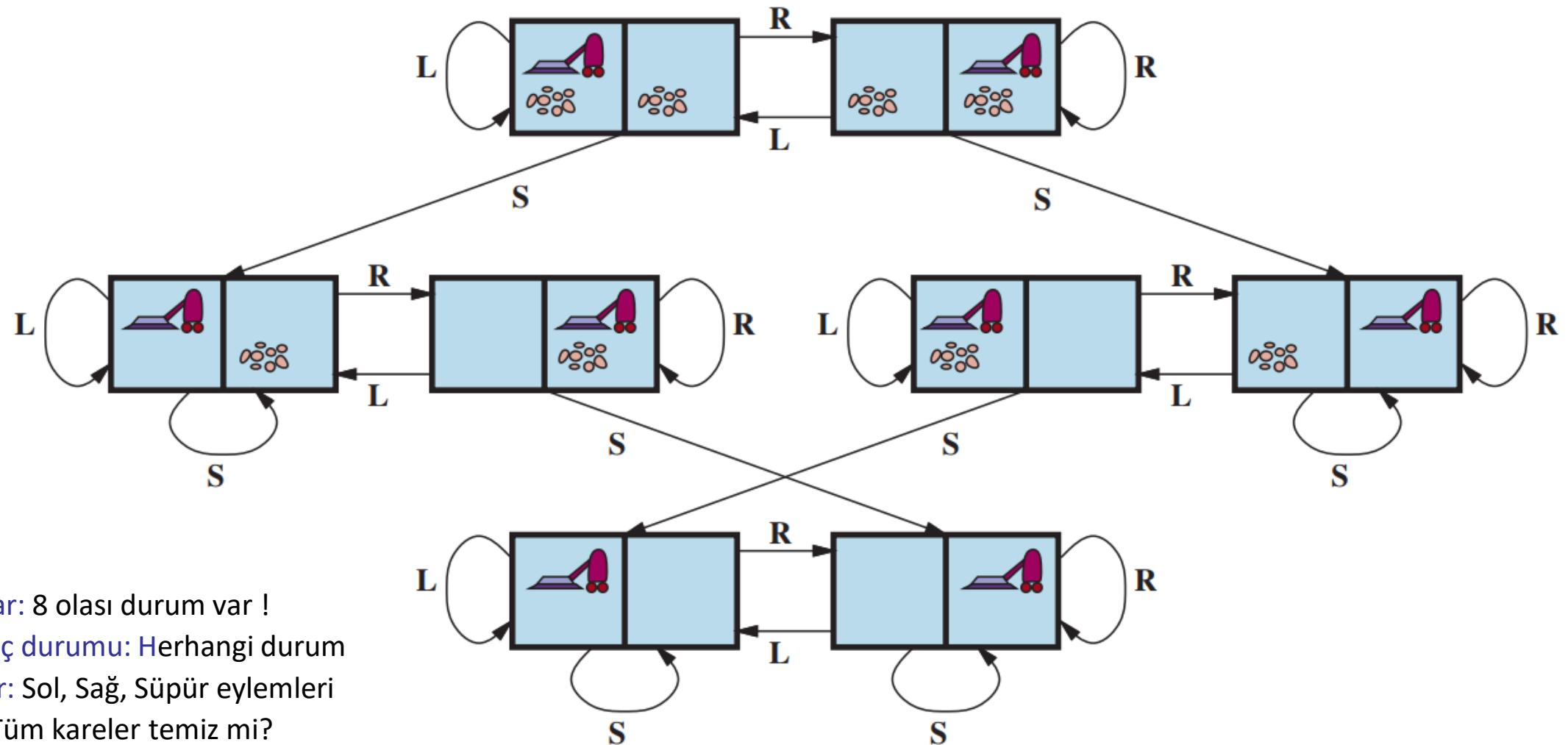
Arama Problemi



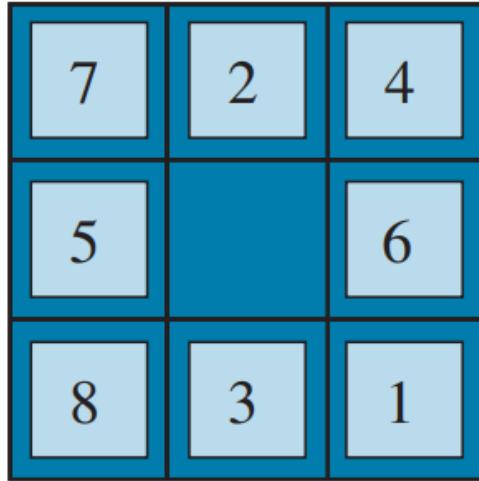
Örnek Problemler



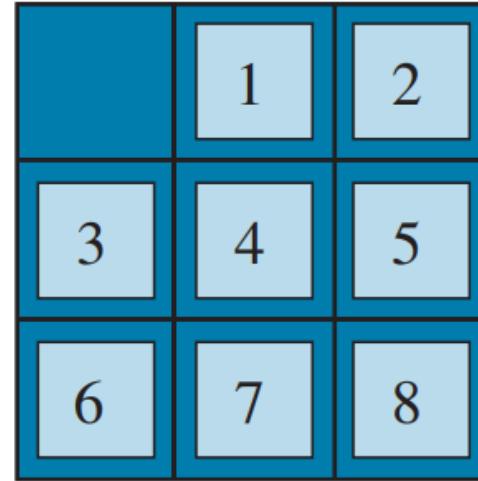
Örnek Problemler: Elektrik Süpürgesi



Örnek Problemler: 8 Puzzle Problemi



Start State

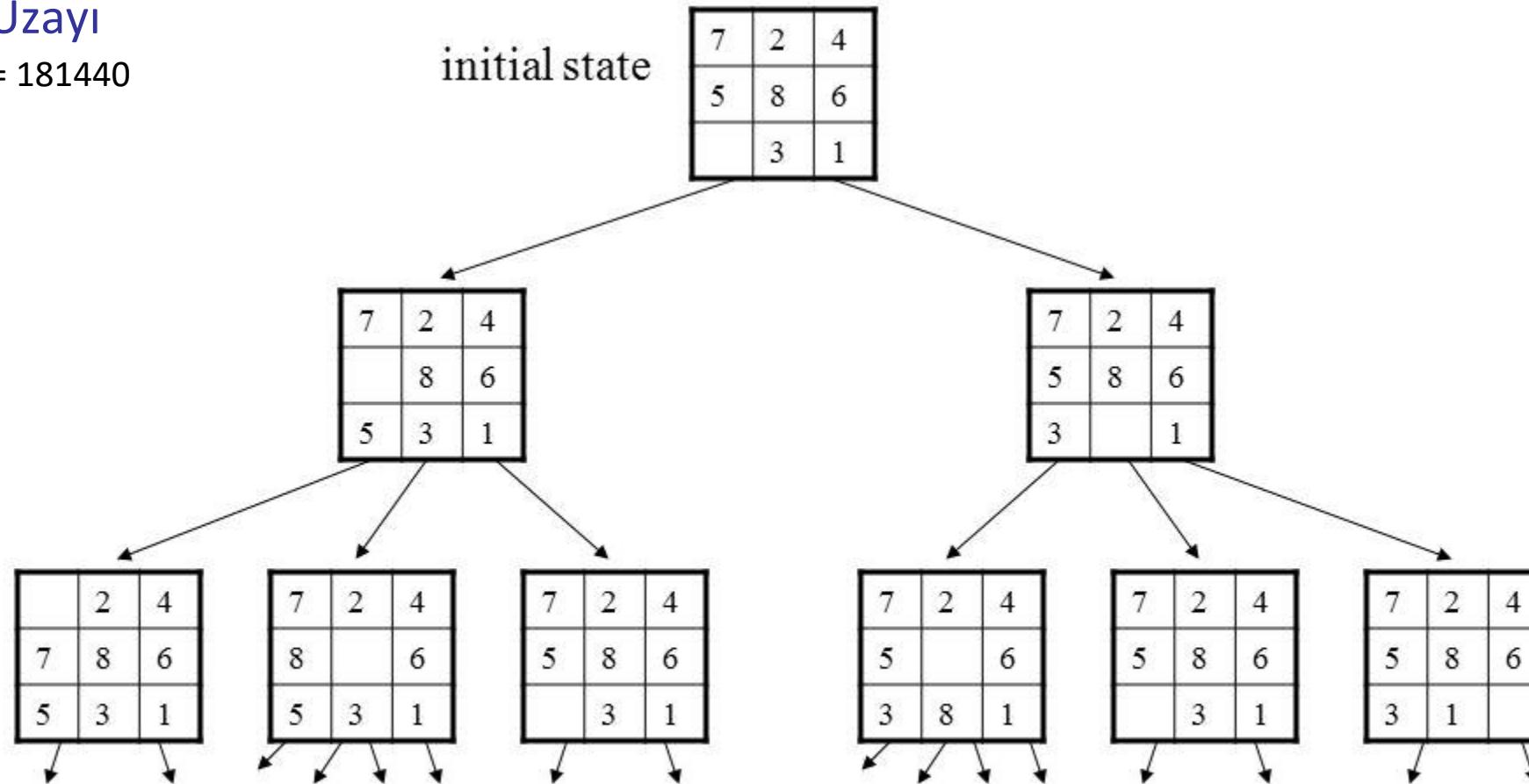


Goal State

- Durumlar: Kareler
- Başlangıç durumu: Belirli bir kare
- Geçişler: Boş kutucuğu sola, sağa, yukarı veya aşağı taşı
- Hedef: Kareler büyük kare etrafında birden sekize kadar sıralanır
- Yol maliyeti: Hareket başına 1 maliyet (çözüm maliyeti çoğunlukla yol uzunluğu ile aynıdır)

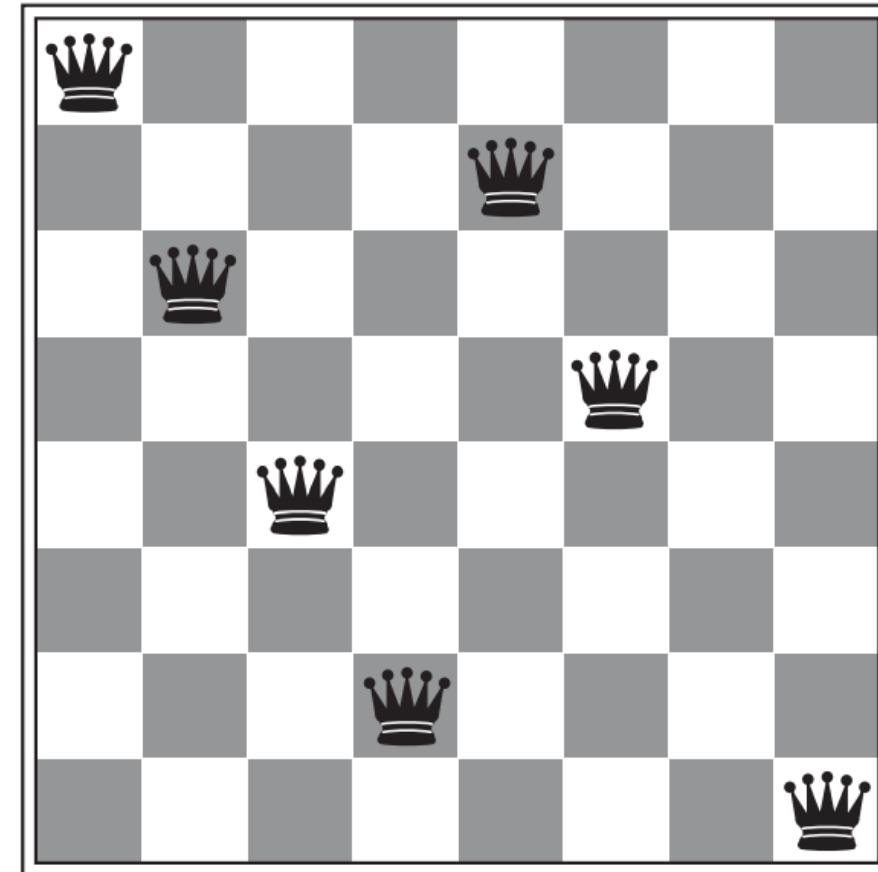
Örnek Problemler: 8 Puzzle Problemi

- Durum Uzayı
 - $9!/2 = 181440$

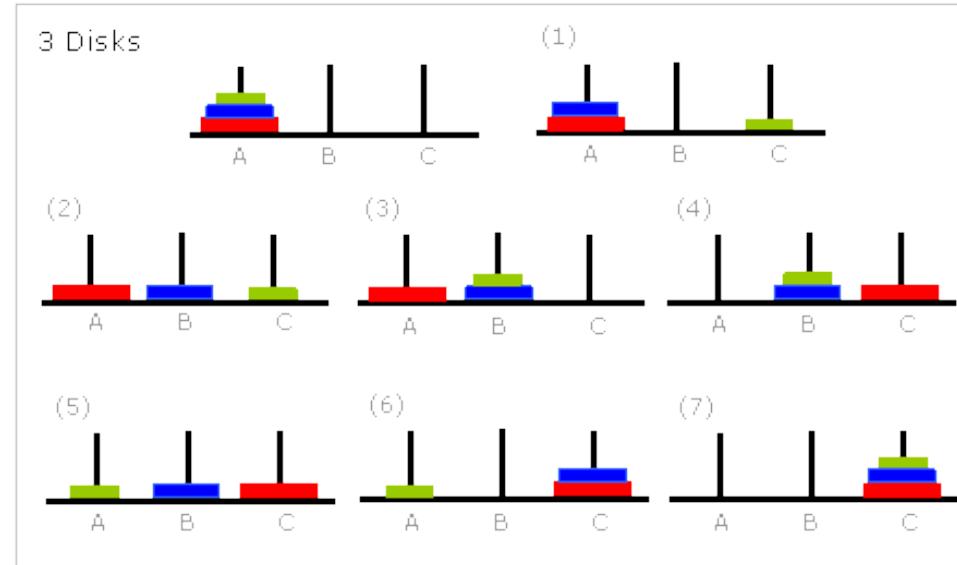


Örnek Problemler: 8 Vezir Problemi

- Durum Uzayı
 - 64^8 durum
- Durumlar: satranç tahtasında 8 vezirin yeri
- Başlangıç durumu: Tahtada vezir yok
- Eylemler: Herhangi bir boş kareye bir vezir ekleyin
- Hedef: 8 vezirin, tahtaya birbirini yenemeyecek şekilde yerleştirilmesi
 - (aynı sıra, sütun veya çaprazda olamaz)
- Yol maliyeti: Yok



Örnek Problemler: Hanoi Kulesi



- Durumlar: çubuklar ve diskler
- Başlangıç durumu: diskler 1 çubuğunda büyükten küçüğe doğru (aşağıdan yukarıya) sıralı
- Eylemler: A diskini 1 çubuğundan 2 çubuğuna hareket ettirin (kısıtlamalara tabi olarak)
 - Bir disk daha küçük disk üstüne taşınamaz
 - Bir disk üstte diskler varsa taşınamaz
- Hedef: Diskler 3 çubuğunda büyükten küçüğe doğru (aşağıdan yukarıya) sıralanacak
- Yol maliyeti: hareket başına 1 birim

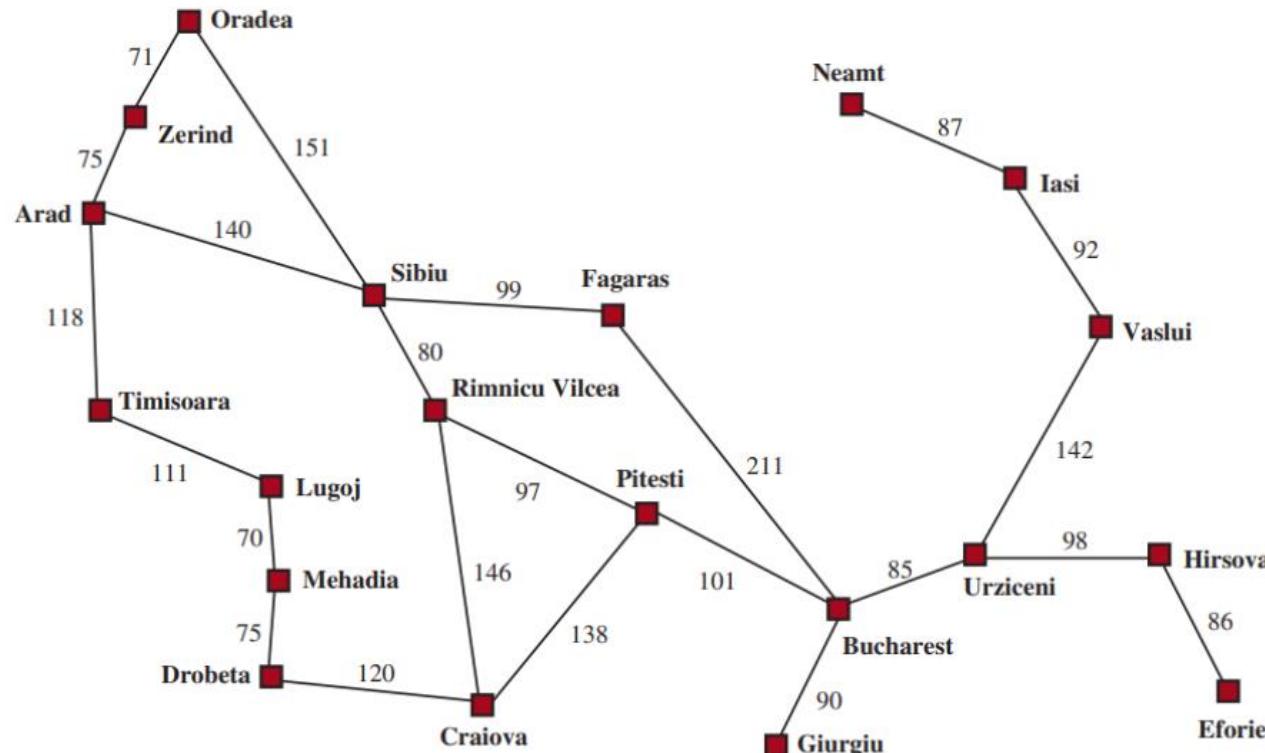
Örnek Problemler: İnsanlar ve Yamyamlar



- **Durumlar:** nehir kıyısındaki insan, yamyam ve tekne sayısı
- **Başlangıç durumu:** nehir kıyısındaki tüm nesneler
- **Eylemler:** x insanı ve y yamyamı nehrin karşı kıyısına taşı.
 - Botta ve nehir kıyısında yamyam sayısı insandan fazla olmayacak.
 - Tekne en fazla m kişi taşıyabiliyor.
- **Hedef:** insanları ve yamyamları nehrin karşı kıyısına taşımak
- **Yol maliyeti:** Nehir geçisi başına 1 birim

Gerçek Dünya Problemi: Yol Bulma

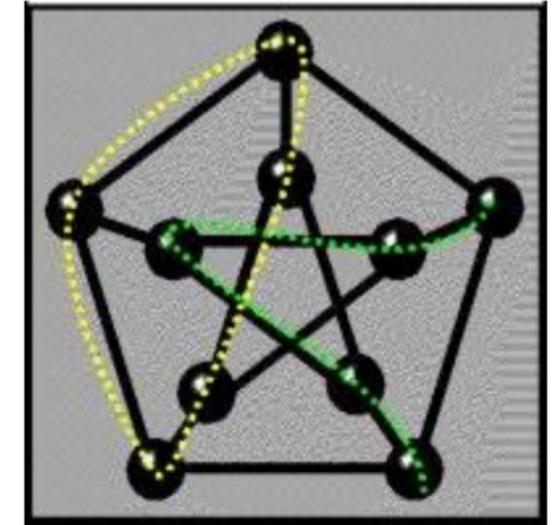
Arad'dan Bucharest'e yol bulma problemi



- Durumlar
 - Şehirler (Her konum = Bir durum)
- Eylemler:
 - Bir konumdan diğer konuma uçuş
- Başlangıç Durumu
 - Arad
- Hedef
 - Durum == Bucharest?
- Yol Maliyeti
 - Pek çok şeye bağlı olabilir
- Çözüm?

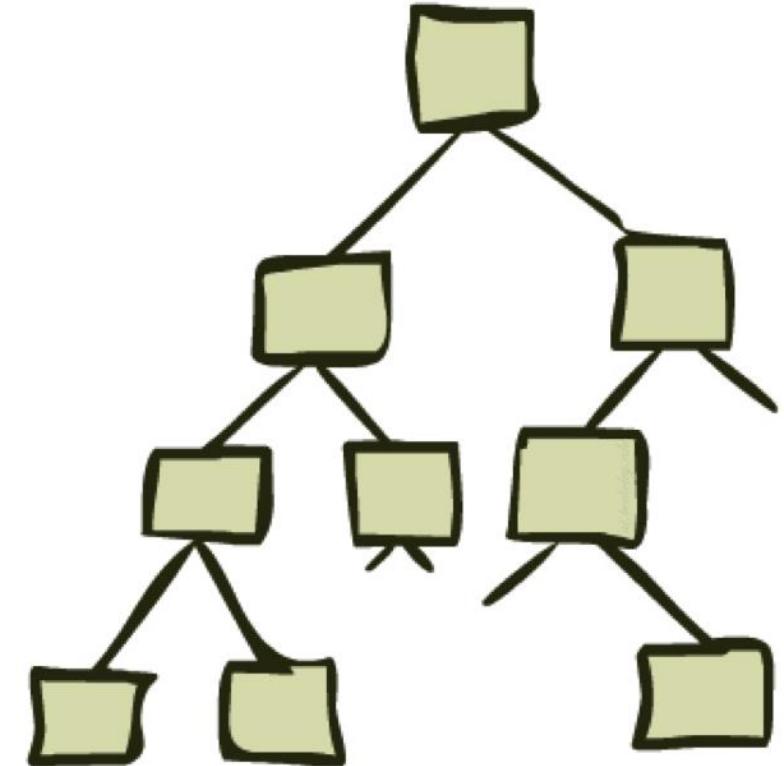
Arama Uzayını Görselleştirme: Graf Veri Yapısı

- YZ problemlerinin çoğunda durum uzayının ve çözüm ağacının gösterilmesinde graflar kullanılır.
- Graflar arama problemlerinin matematiksel gösterimidir.
- Graf:
 - Yalın graf:
 - Bağlantılı graf
- Komşu düğüm: birbiri ile kenar bağlantısı olan düğümlerdir.
- Grafın herhangi düğüm komşuları sayısı onun derecesini belirler.
- Döngü içermeyen bağlantılı graflar ağaç olarak adlandırılır.



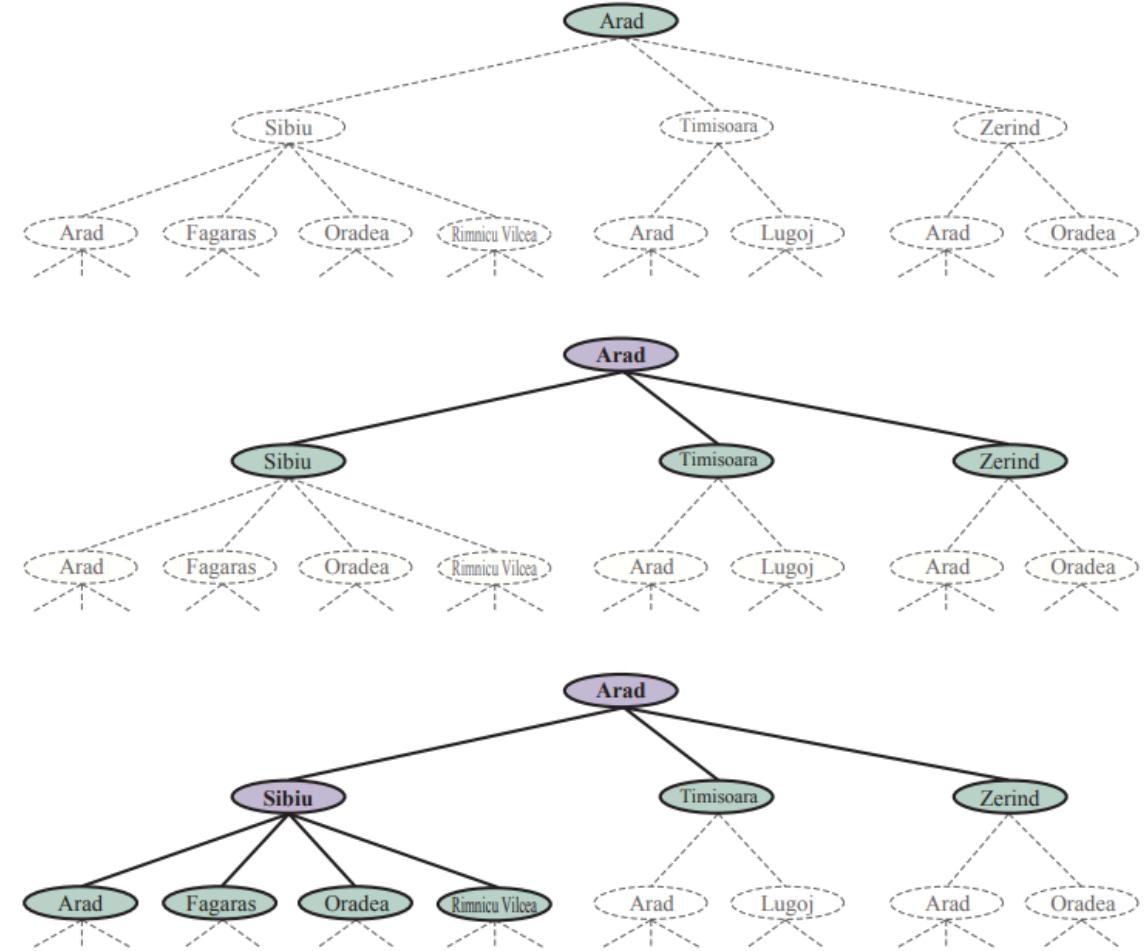
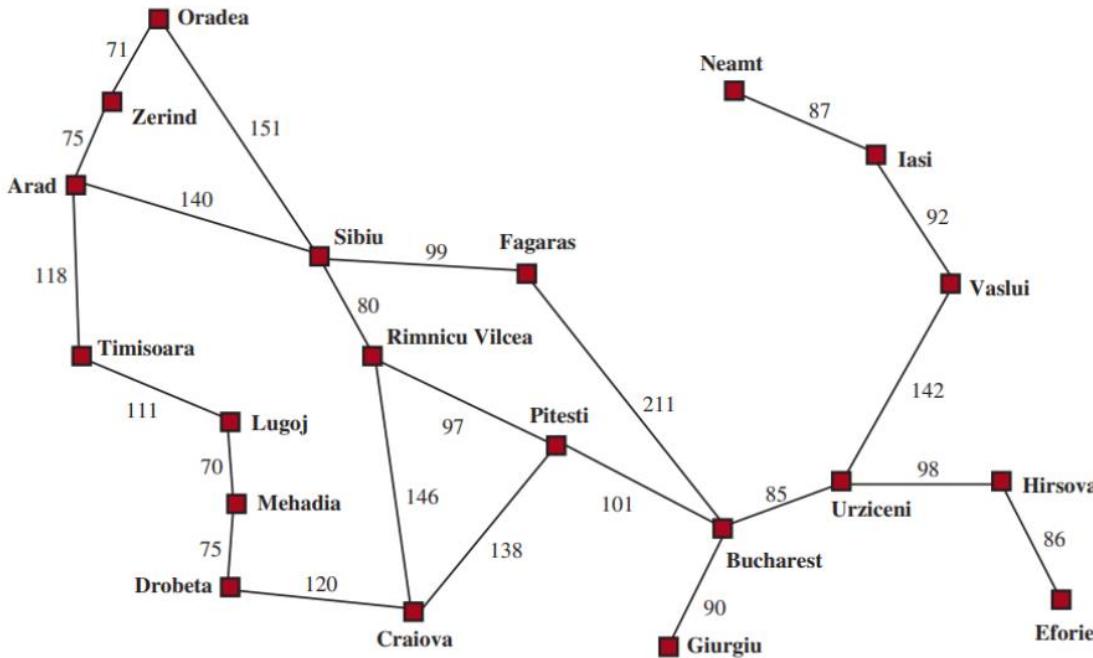
Arama Uzayını Görselleştirme: Arama Ağacı

- Durumlar düğümlerdir
- Eylemler kenarlardır / dallar
- Başlangıç durumu köktür
 - Çözüm, kökten hedef düğüme giden yoldur
- Kenarlar bazen ilişkili maliyetlere sahiptir
- Eylemlerden kaynaklanan durumlar çocuklardır
- Derinlik başlangıç durumdan düğüme olan adımların sayısıdır.



Arama Uzayını Görselleştirme: Arama Ağacı

Arad'dan Bucharest'e yol bulma problemi



Arama Uzayını Görselleştirme

function TREE-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

loop do

if the frontier is empty **then return** failure

 choose a leaf node and remove it from the frontier

if the node contains a goal state **then return** the corresponding solution

 expand the chosen node, adding the resulting nodes to the frontier

function GRAPH-SEARCH(*problem*) **returns** a solution, or failure

initialize the frontier using the initial state of *problem*

initialize the explored set to be empty

loop do

if the frontier is empty **then return** failure

 choose a leaf node and remove it from the frontier

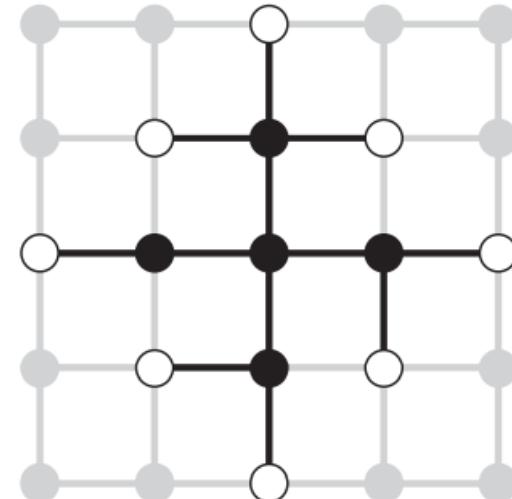
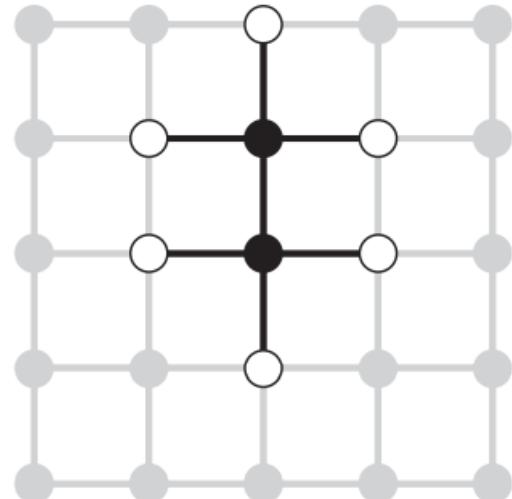
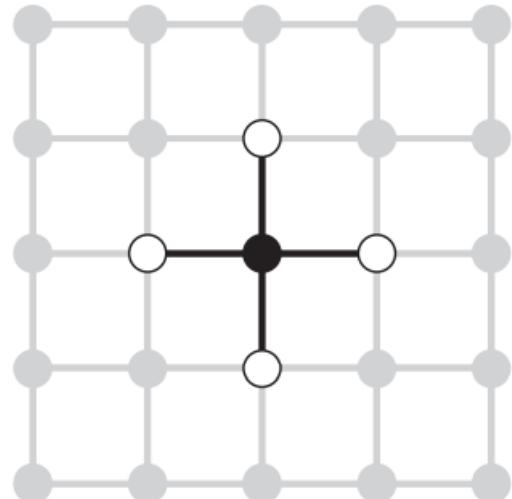
if the node contains a goal state **then return** the corresponding solution

add the node to the explored set

 expand the chosen node, adding the resulting nodes to the frontier

only if not in the frontier or explored set

Frontier / Closed List



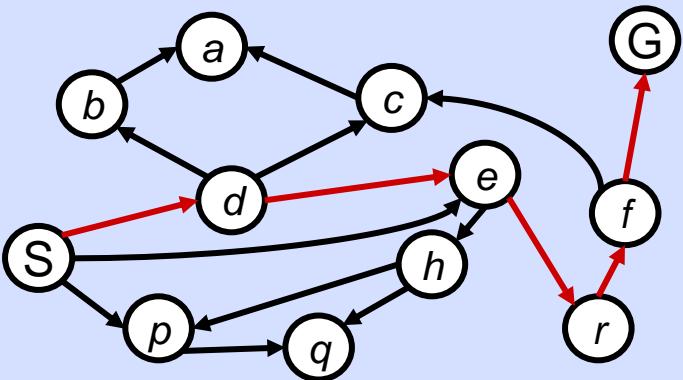
(a)

(b)

(c)

Durum Uzayı Grafi vs. Arama Ağacı

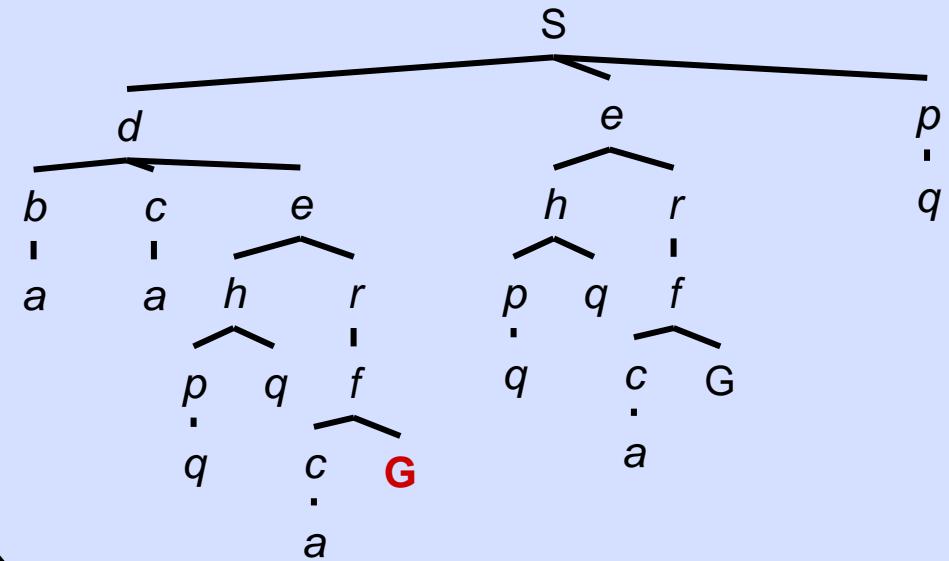
Durum Uzayı Grafi



Arama ağacındaki her DÜĞÜM, durum uzay grafındaki tam bir YOL'dur

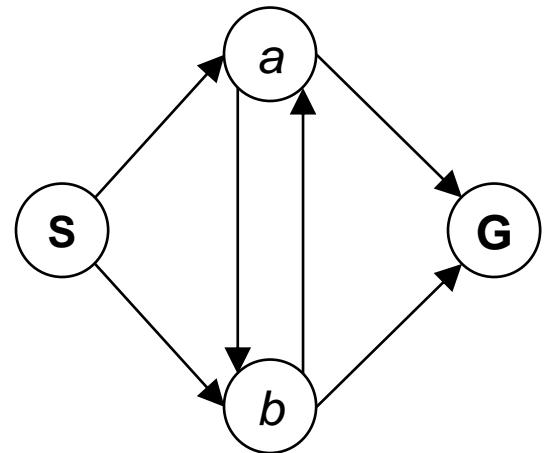
Talebe göre her ikisi de kullanılabilir.
İstenilen mümkün olduğunda küçük boyutlu inşa edilmeleridir.

Arama Ağacı



Quiz: Durum Uzayı Grafi vs. Arama Ağacı

4 durumlu bir graf:

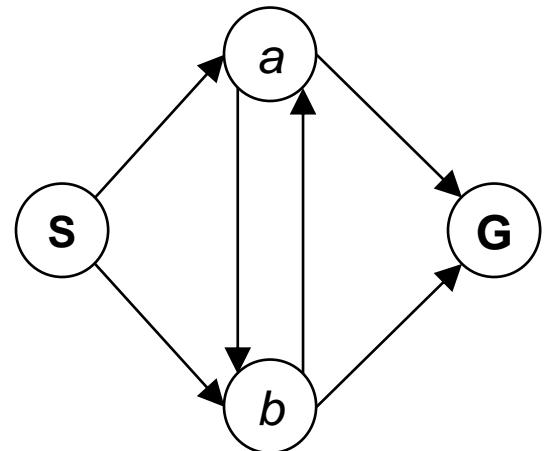


S durumundan başlanırsa arama ağacı ne kadar büyük olur?

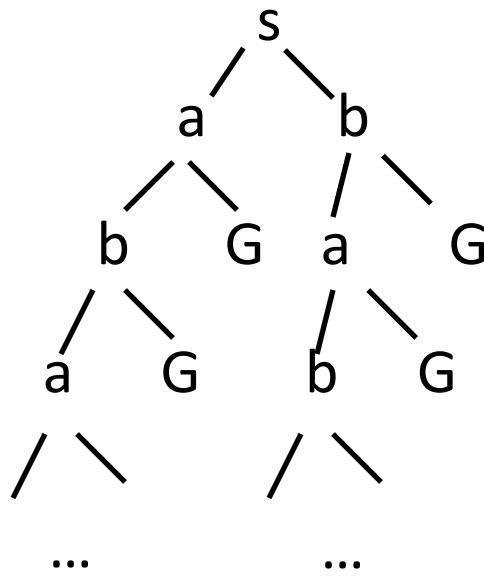


Quiz: Durum Uzayı Grafi vs. Arama Ağacı

4 durumlu bir graf:



S durumundan başlanırsa arama ağacı
ne kadar büyük olur?



Önemli: Arama ağacında pek çok tekrarlı yapı mevcut!

Problem Çözümü Aşamaları

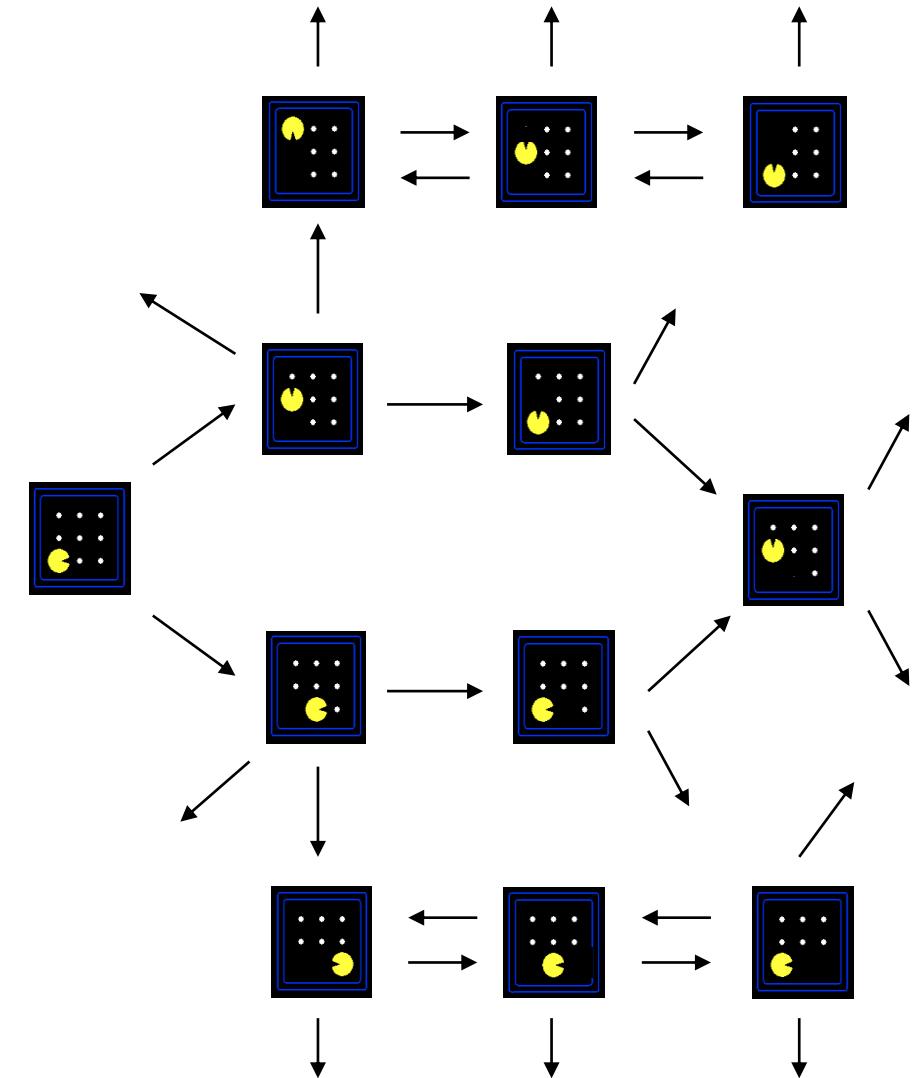
- Problemi anlama
 - Problemin genel resminin çizilmesi
- Çözümü Planlama
 - İlişkilerin belirlenmesi
- Seçilen planı uygulama
 - Çözümün her aşamasının uygulanması
- Sonuçların değerlendirilmesi

Problem Çözme Performansı

- **Tamlık**
 - Algoritma bir çözüm bulmayı garanti ediyor mu?
- **Optimallik**
 - Algoritma optimal çözümü buluyor mu?
- **Zaman Karmaşıklığı**
 - Bir çözüm bulmak ne kadar sürer?
- **Uzay Karmaşıklığı**
 - Ne kadar hafıza gerekiyor?

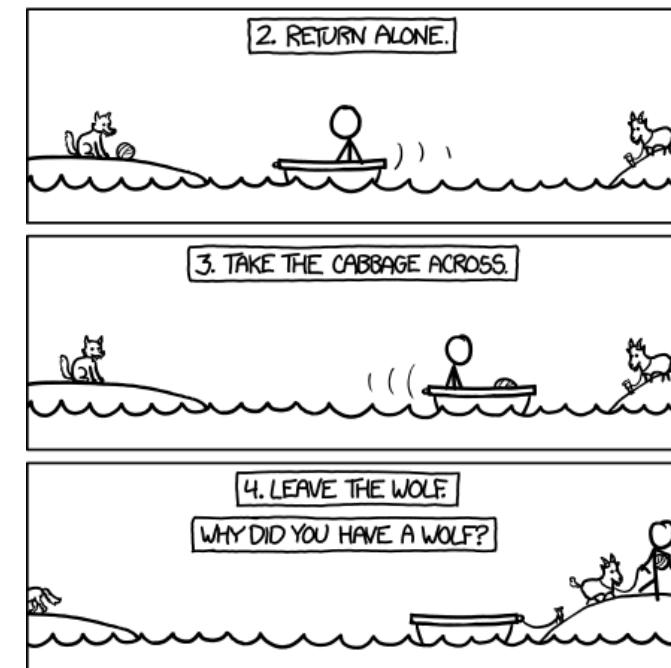
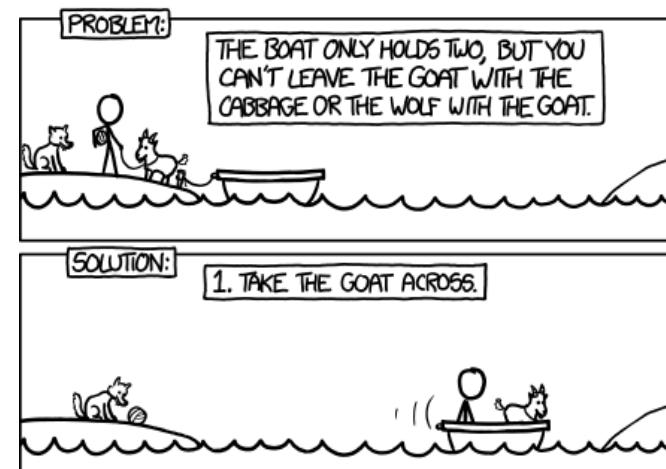
Durum Uzayı

- YZ problemleri sonlu durumlar kümesinden oluşur.
- Çözüm tüm durum uzayı taranarak bulunur.
- Durum uzayı kümesi elemanları
 - Graf yapısı ile temsil edilir.
- Durumlar
 - Düğümler
- Durumlardan durumlara izinli geçişler
 - Yollar
- Çözüm
 - Graftaki minimum yolun bulunması problemidir.



Durum Uzayı Örnek: Kurt – Kuzu – Lahana Problemi

- Çiftçi nehrin sağ kıyısındaki kurt, kuzu ve lahanayı bir tekne ile sol kıyıya geçirmek istiyor.
 - Tekne yalnız 2 nesne alabiliyor
 - Çiftçi yanlarında olduğu sürece kuzu lahanayı, kurt kuzuyu yiyecek
- Çözüm tüm durum uzayı taranarak bulunur.



Durum Uzayı Örnek: Kurt – Kuzu – Lahana Problemi

- Çözüm tüm durum uzayı taranarak bulunur.
- Her nesne 2 farklı konumda olabileceğiinden $2^4 = 16$ farklı durum
 - Başlangıç durum: 1
 - Hedef durum: 16

1. çOKW / Ø

2. çOK / W

3. çOW / K

4. çKW / O

5. ~~OKW/c~~

6. ~~cO/KW~~

7. CK / OW

8. ~~cW/oK~~

9. ~~OK/cw~~

10. OW / cK

11. ~~KW/cO~~

12. ~~c/cOKW~~

13. O / CKW

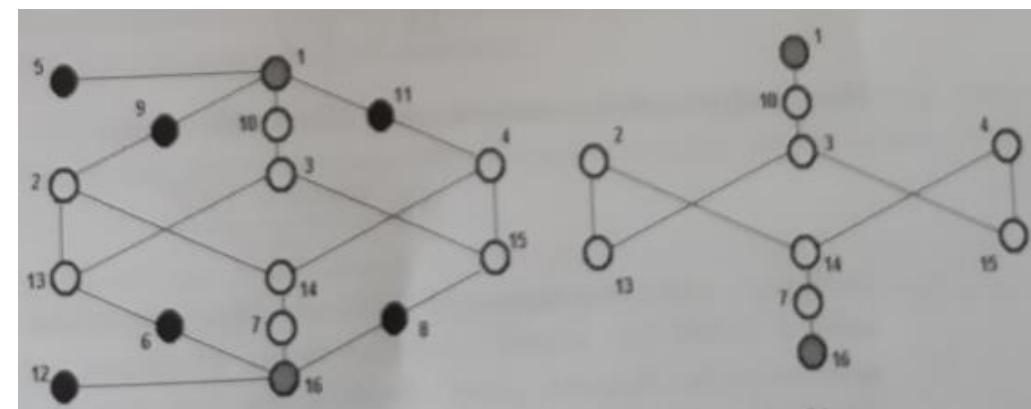
14. K / COW

15. W / COK

16. Ø / COKW

1-10 -3 -13 -2 -14 -7 -16

1-10 -3 -15 -4 -14 -7 -16



Durum Uzayı Örnek: Kurt – Kuzu – Lahana Problemi

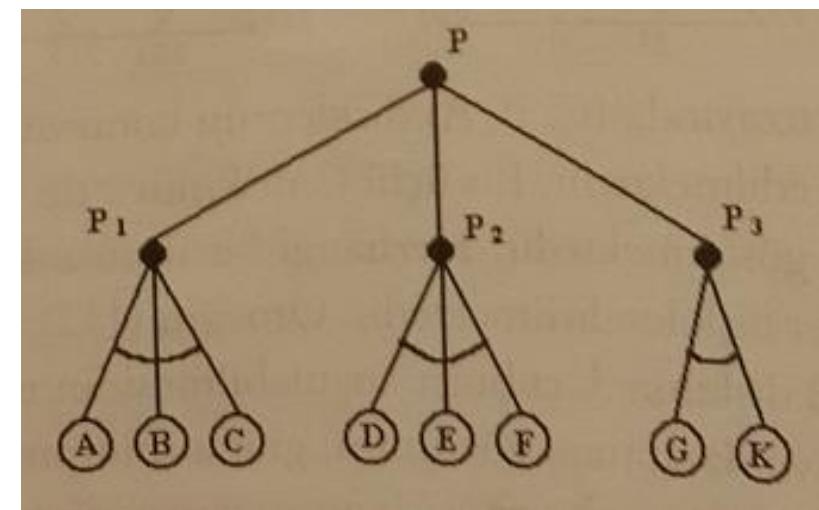
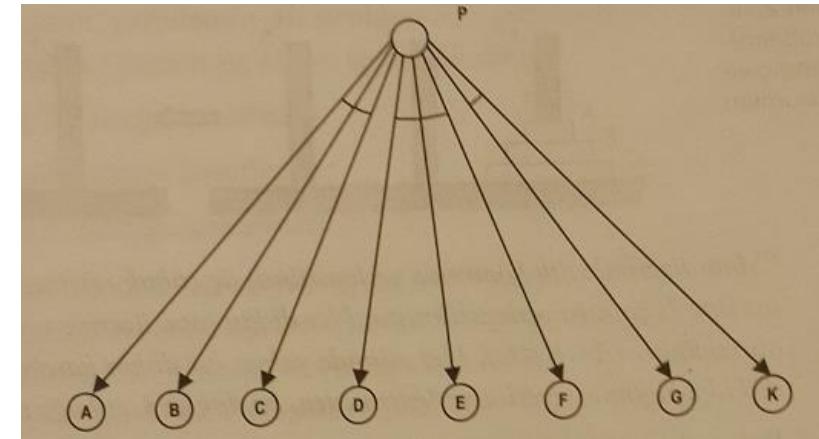
- Çözüm tüm durum uzayı taranarak bulunur.

	Left bank	Right bank	move
0	Cabbage, Wolf, Goat, Farmer		Farmer and Goat to the right
1	Wolf, Cabbage	Goat, Farmer	Farmer to the left
2	Cabbage, Wolf, Farmer	Goat	Farmer and Cabbage to the right
3	Wolf	Cabbage, Goat, Farmer	Farmer and Goat to the left
4	Wolf, Goat, Farmer	Cabbage	Farmer and Wolf to the right
5	Goat	Cabbage, Wolf, Farmer	Farmer to the left
6	Goat, Farmer	Cabbage, Wolf	Farmer and Goat to the right
7		Cabbage, Wolf, Goat, Farmer	

	Left bank	Right bank	move
0	Cabbage, Wolf, Goat, Farmer		Farmer and Goat to the right
1	Wolf, Cabbage	Goat, Farmer	Farmer to the left
2	Cabbage, Wolf, Farmer	Goat	Farmer and Wolf to the right
3	Cabbage	Wolf, Goat, Farmer	Farmer and Goat to the left
4	Cabbage, Goat, Farmer	Wolf	Farmer and Cabbage to the right
5	Goat	Cabbage, Wolf, Farmer	Farmer to the left
6	Goat, Farmer	Cabbage, Wolf	Farmer and Goat to the right
7		Cabbage, Wolf, Goat, Farmer	

Problemin Alt Problemlere Ayrılması

- Karmaşık problemler alt problemlere parçalanarak çözülür.
- Her alt problem farklı bir yöntemle çözülebilir.
- **Alt problemler**
 - Çözümleri kesin olan ve parçalanamayan durumları ifade eder.
- Bu tür çözümler ağaç yapısına sahiptir
 - AND/OR ağıacı
- Verilmiş bir P probleminin
 - A,B,C problemlerinin çözümüyle veya
 - D,E,F problemleri yardımıyla yada
 - G ve K problemleri ile çözülebileceğini düşünelim.



Durum Uzayı Örnek: Hanoi Kulesi

- Amaç: Sol sütundaki diskleri, kurallara uyarak sağdaki sütuna geçirmek.

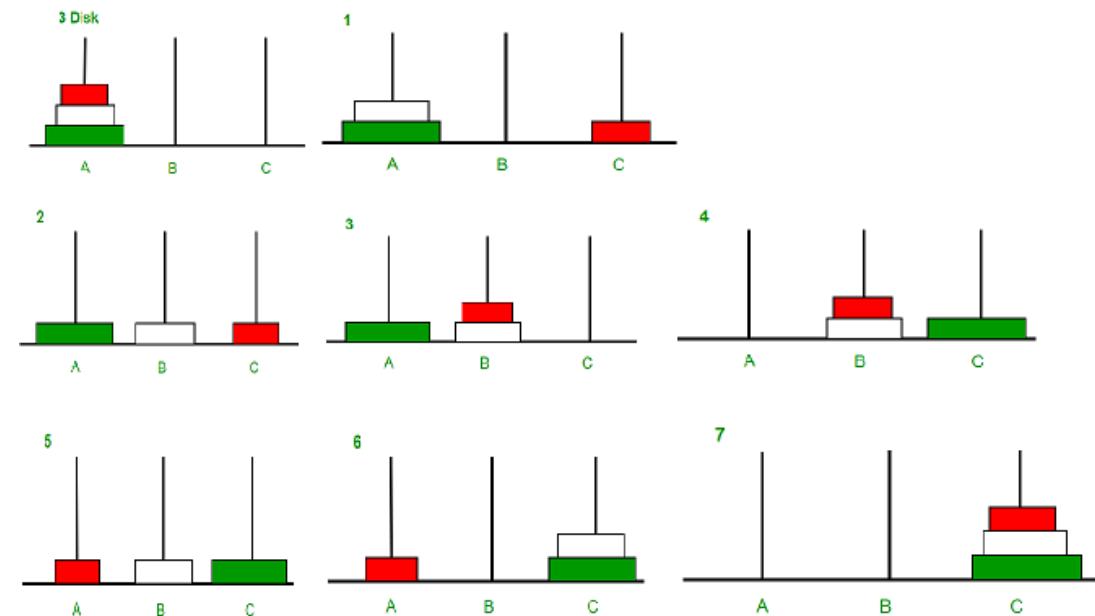
- Kurallar

- Her hamlede bir sütundaki en üstteki disk alınıp başka bir sütuna geçirilebilir.
- Her hamlede yalnızca bir disk taşınabilir.
- Küçük bir disk asla büyük bir diskin altına konamaz.
- Üç sütun istenilen şekilde kullanılabilir.

- 3 disk için toplam 27 durum var.

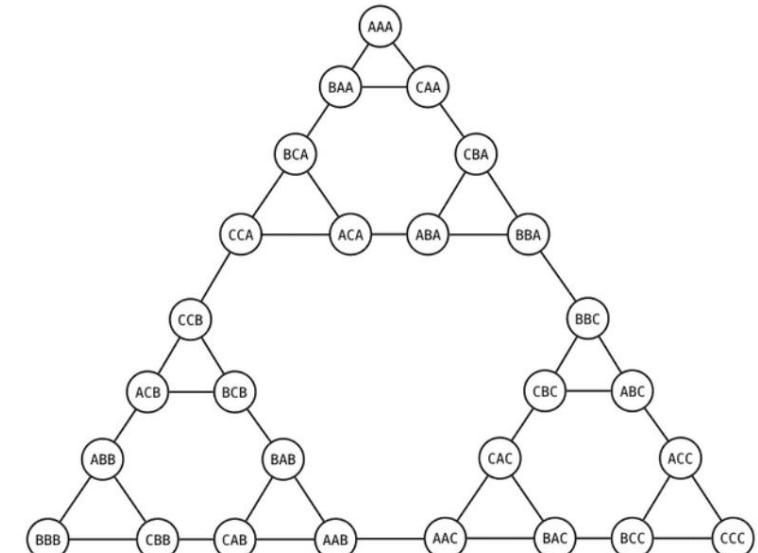
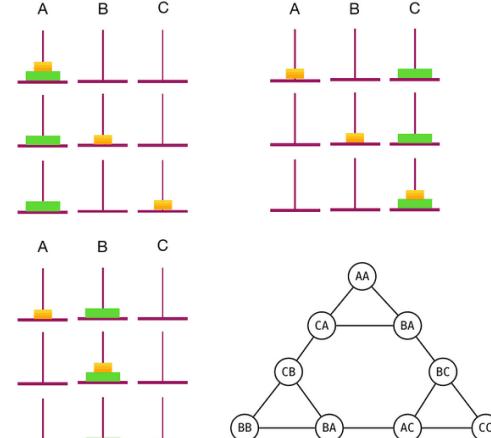
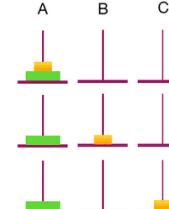
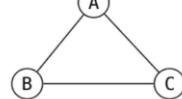
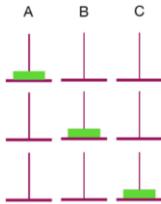
- En kısa çözümler:

- 3 disk = 7 hareket = 2^3-1
- 4 disk = 15 hareket = 2^4-1
- 5 disk = 31 hareket = 2^5-1
- 6 disk = 63 hareket = 2^6-1
- 7 disk = 127 hareket = 2^7-1
- 8 disk = 255 hareket == 2^8-1

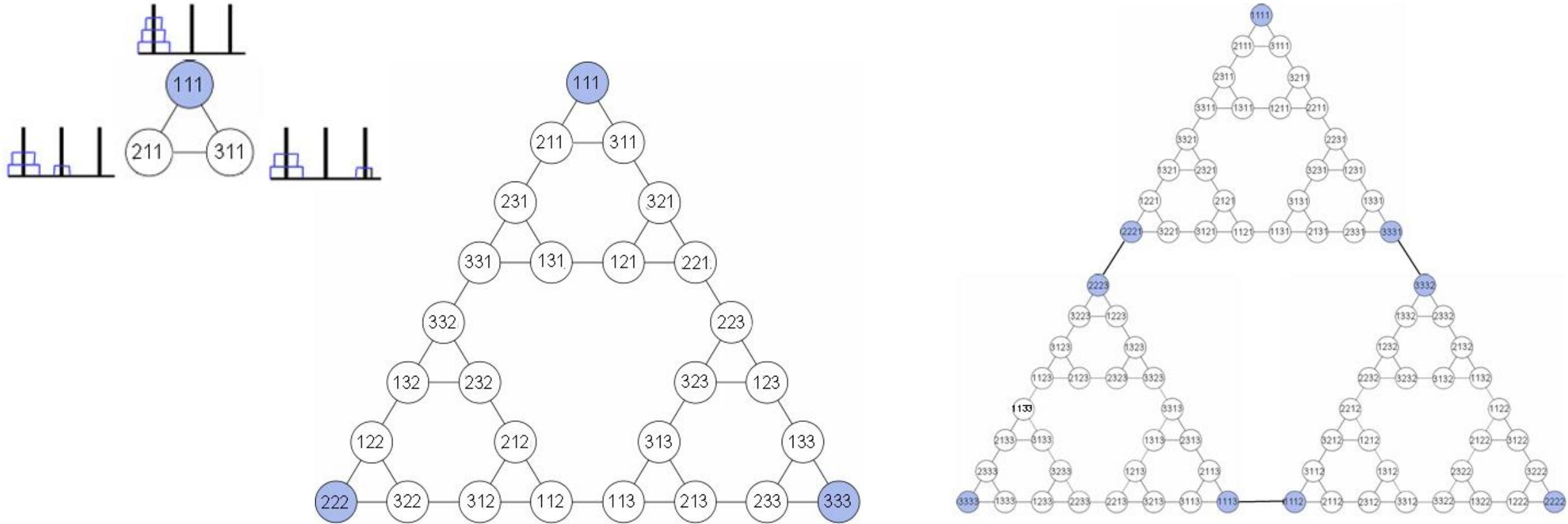


Durum Uzayı Örnek: Hanoi Kulesi

- Problem yönsüz bir grafla temsil edilebilir
 - Düğümler disklerin dağılımını temsil eder
 - Kenarlar hareketleri temsil eder
 - Bir disk için grafik bir üçgendir:
 - Daha büyük diski temsil etmek için ikinci bir harf eklenir.
 - Durum uzayında disklerin konumları (A,B,C) şeklinde gösterilmiştir

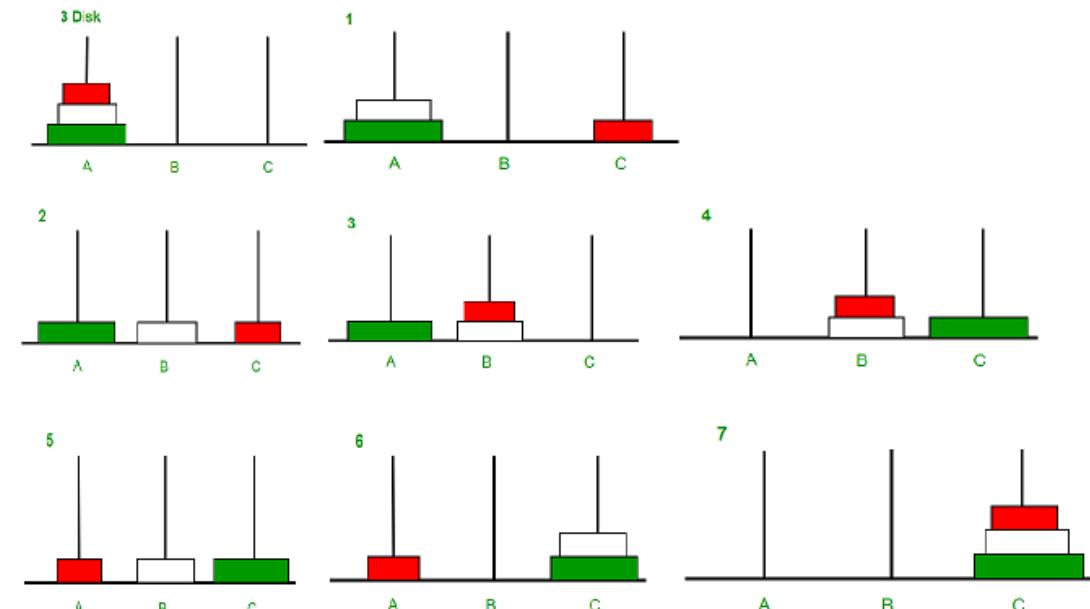


Durum Uzayı Örnek: Hanoi Kulesi



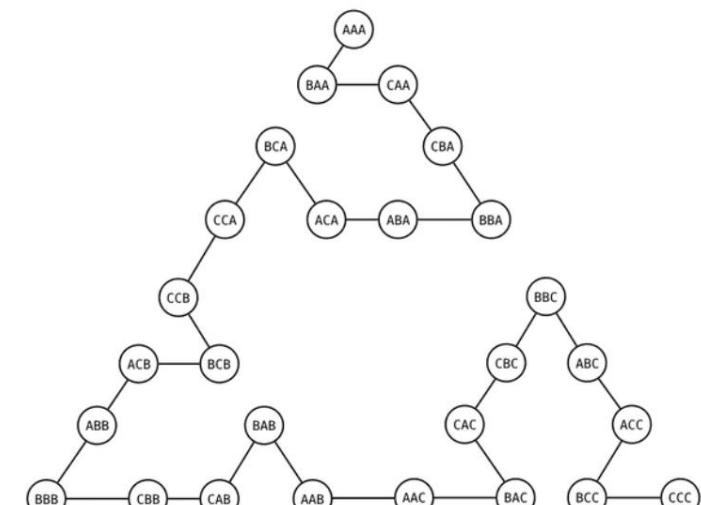
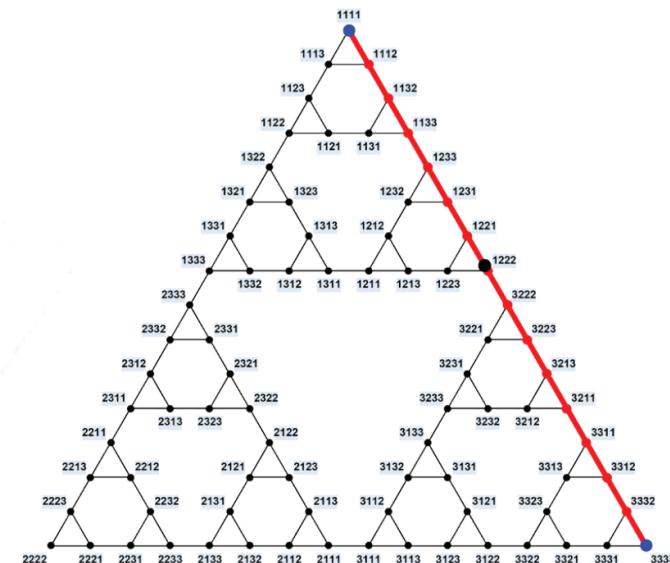
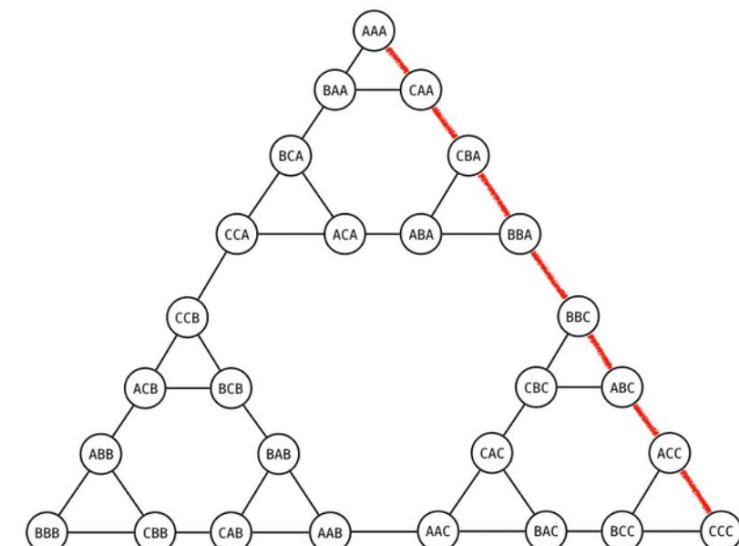
Durum Uzayı Örnek: Hanoi Kulesi

- Problemi alt problemlere parçalayalım:
- Başlangıçta çözüm 3 amaç içerir:
 - 1. Kırmızı diskin 3. çubuğa konması
 - 2. Beyaz diskin 3. çubuğa konması
 - 3. Yeşil diskin 3. çubuğa konması
- İlk önce en zor problemi çözelim (alt problemlerden daha çok sınırlama içeren daha zordur) o halde 3. seçenek daha zordur.
- Problem 3 yeni alt problemlere bölünsün:
 - 1. Kırmızı ve Beyaz disklerin 2. çubuğa konması
 - 2. Yeşil diskin direk 3. çubuğa taşınması
 - 3. Kırmızı ve Beyaz disklerin 3. çubuğa taşınması ve hedefe ulaşılması
- Bu yeni alt problemlerde değişken sayıları ve sınırlamalar azaldı.
- 1. Kırmızı ve Beyaz disklerin 2. çubuğa konması
 - 1.1. Kırmızı diskin 3. çubuğa konması
 - 1.2. Beyaz diskin 2. çubuğa konması
 - 1.3. Kırmızı diskin 2. çubuğa konması
- 2. Yeşil diskin direk 3. çubuğa taşınması
- 3. Kırmızı ve Beyaz disklerin 3. çubuğa taşınması ve hedefe ulaşılması
 - 3.1. Kırmızı diskin 1. çubuğa konması
 - 3.2. Beyaz diskin 3. çubuğa konması
 - 3.3. Kırmızı diskin 3. çubuğa konması



Durum Uzayı Örnek: Hanoi Kulesi

- Buradan yola çıkarak n sayıda disk olduğunda genel çözüm şu şekildedir:
 - Tek disk direk 3. çubuğa konur
 - N sayıda disk 3 adımda koyulur
 - Adım 1: ($N-1$) disk orta çubuğa taşınır
 - Adım 2: En alttaki disk direk sağa konur
 - Adım 3: ($N-1$) disk sağa taşınır

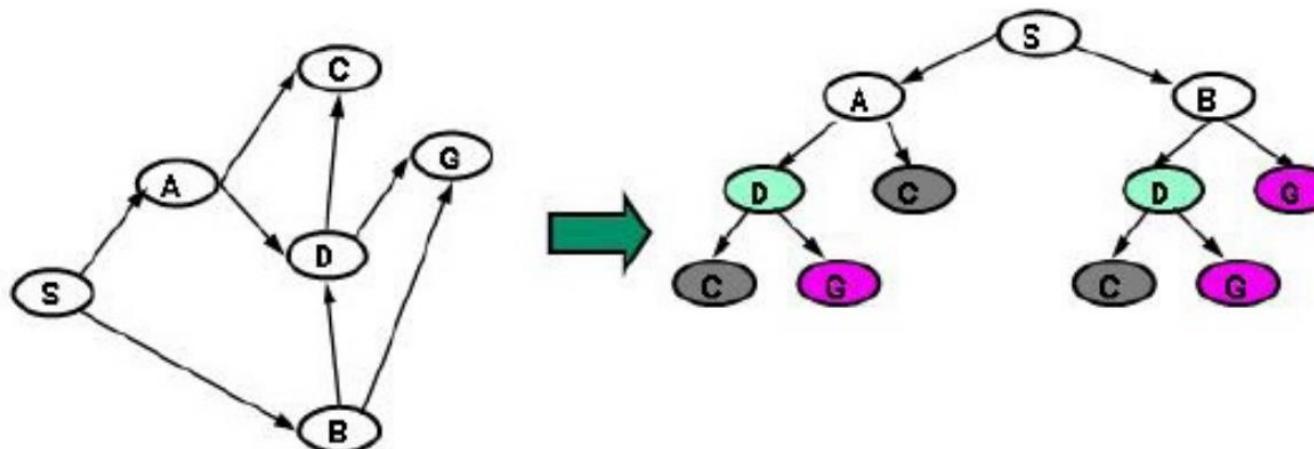


Durum Uzayında Arama

- Bilgisiz Arama (Un-informed Strategies)
 - Genişlik Öncelikli Arama
 - Breadth-first search
 - Sabit Maliyetli Arama
 - Uniform-cost search
 - Derinlik Öncelikli Arama
 - Depth-first search
 - Derinlik Sınırlı Arama
 - Depth-limited search
 - Yinelemeli Derinleşem Derinlik Öncelikli Arama
 - Iterative deepening depth-first search
 - İki Yönlü Arama
 - Bidirectional Search
- Bilgili Arama Stratejileri (Informed (Heuristic) Strategies)
 - Gelecek Hafta

Bilgisiz Arama

- Blind: problemde tanımlananın ötesinde hiçbir ek bilgi içermezler.
- Tüm yaptıkları successor (amaç) fonksiyon yaratmak ve amaçsız(nongoal) bir durumdan amaç(goal) durumu ayırmaktır.
- Durum uzayı graf ile temsil edilebilir.
- **Ağaç:** çocuk düğüm için yalnızca 1 parent düğüm
- Problemin başlangıç durumu ağacın köküne yerleştirilir. Ve hedef düğüme doğru en kısa yol için farklı çözümler düşünülebilir.
- S şehrinden G şehrine minimum yolun bulunması ve durum uzayı:

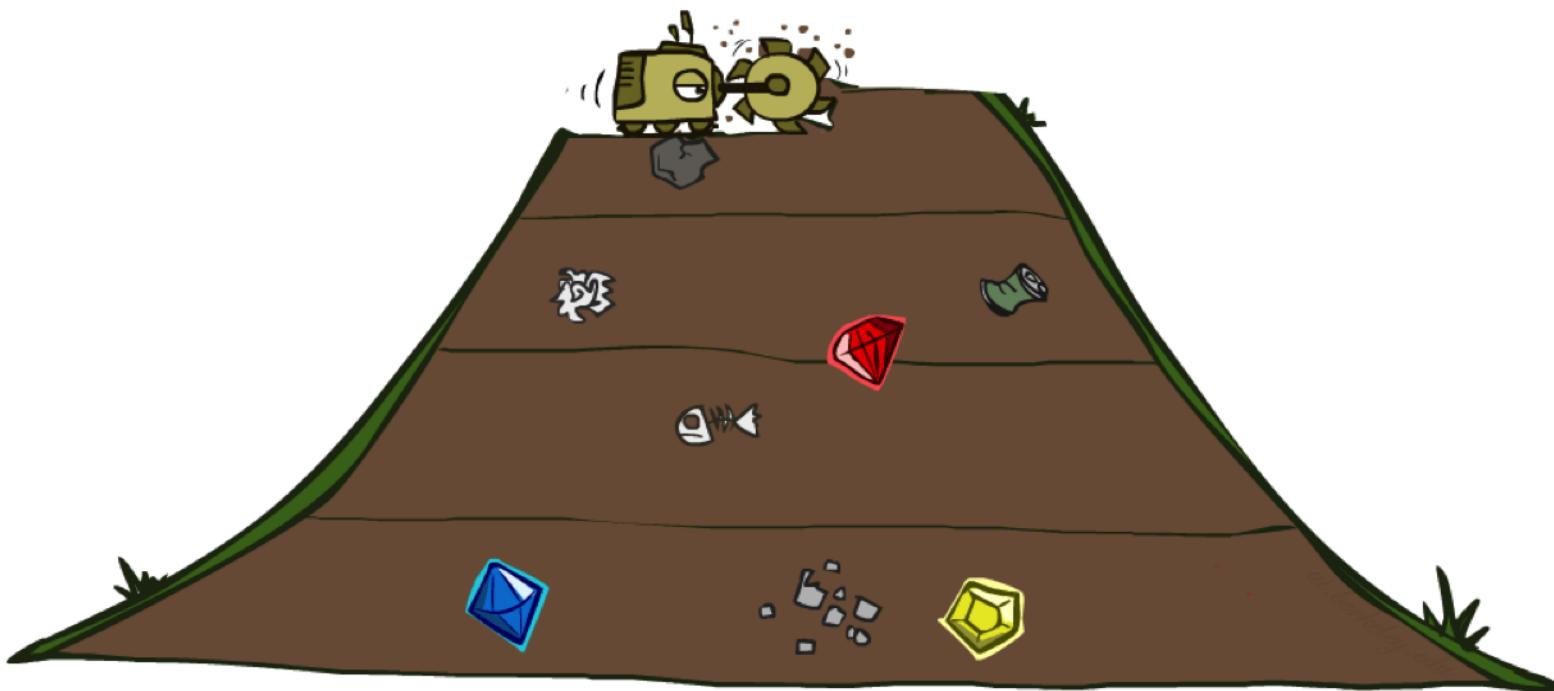


Ağaçta Arama Algoritmaları

```
function TREE-SEARCH( problem, strategy) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
    end
```

- Önemli Noktalar:
 - Uç düğümler
 - Genişleme
 - Genişleme Stratejisi
- Ana Problem: Hangi uç düğümler keşfedilecek?

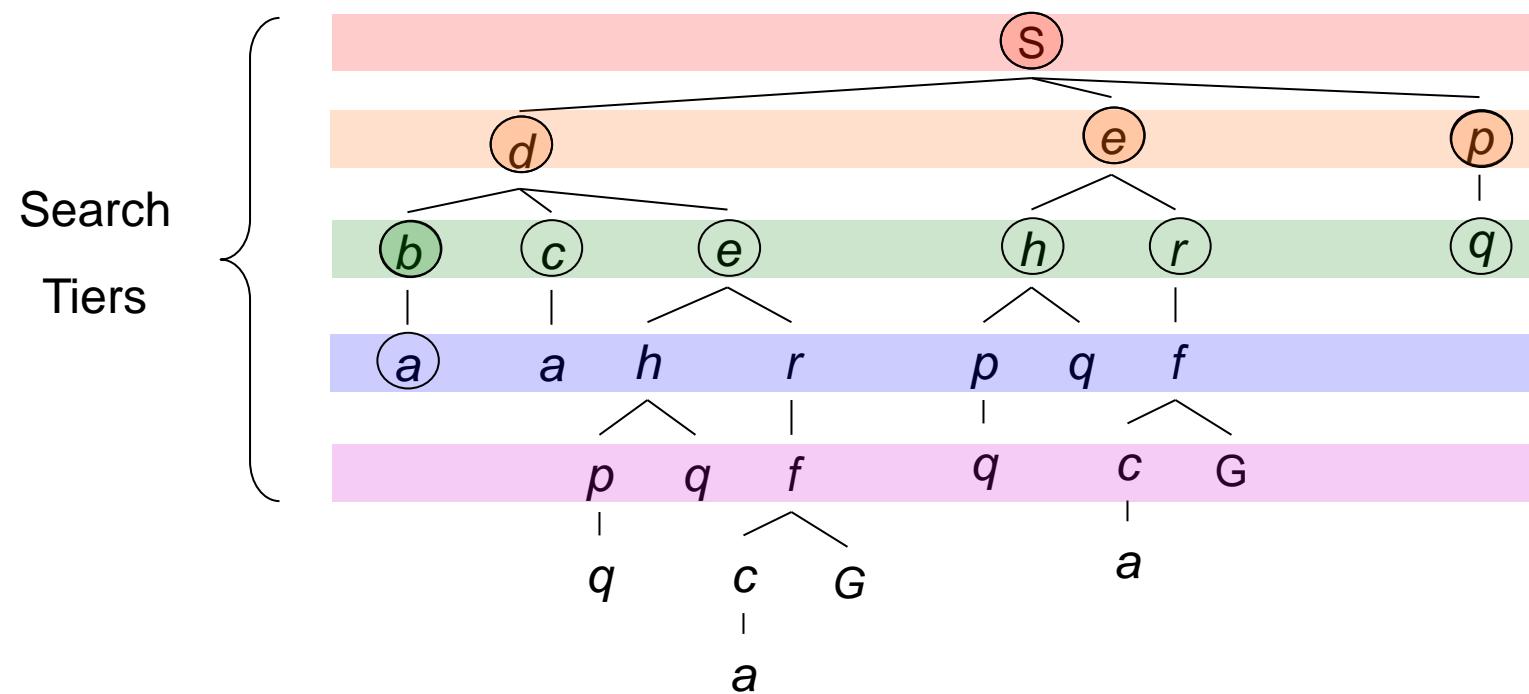
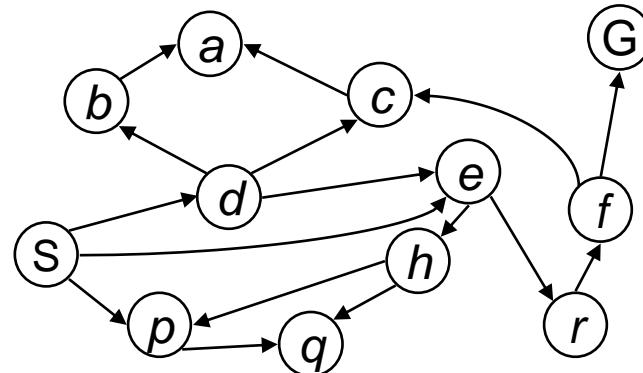
Genişlik Öncelikli Arama (BFS)



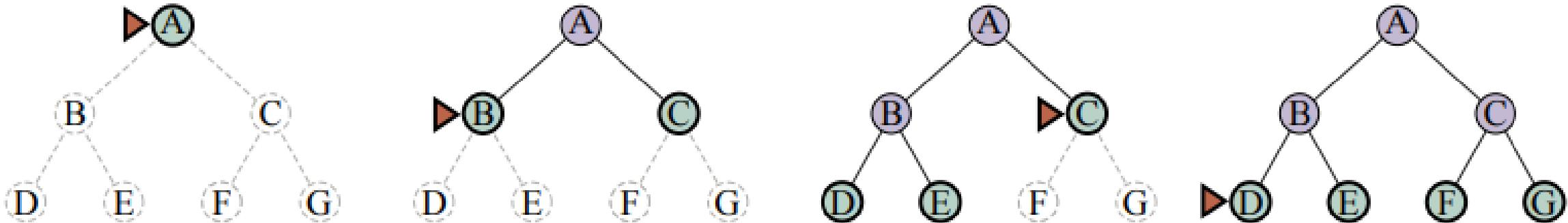
Genişlik Öncelikli Arama

Strateji: İlk olarak en sıçradıktır.

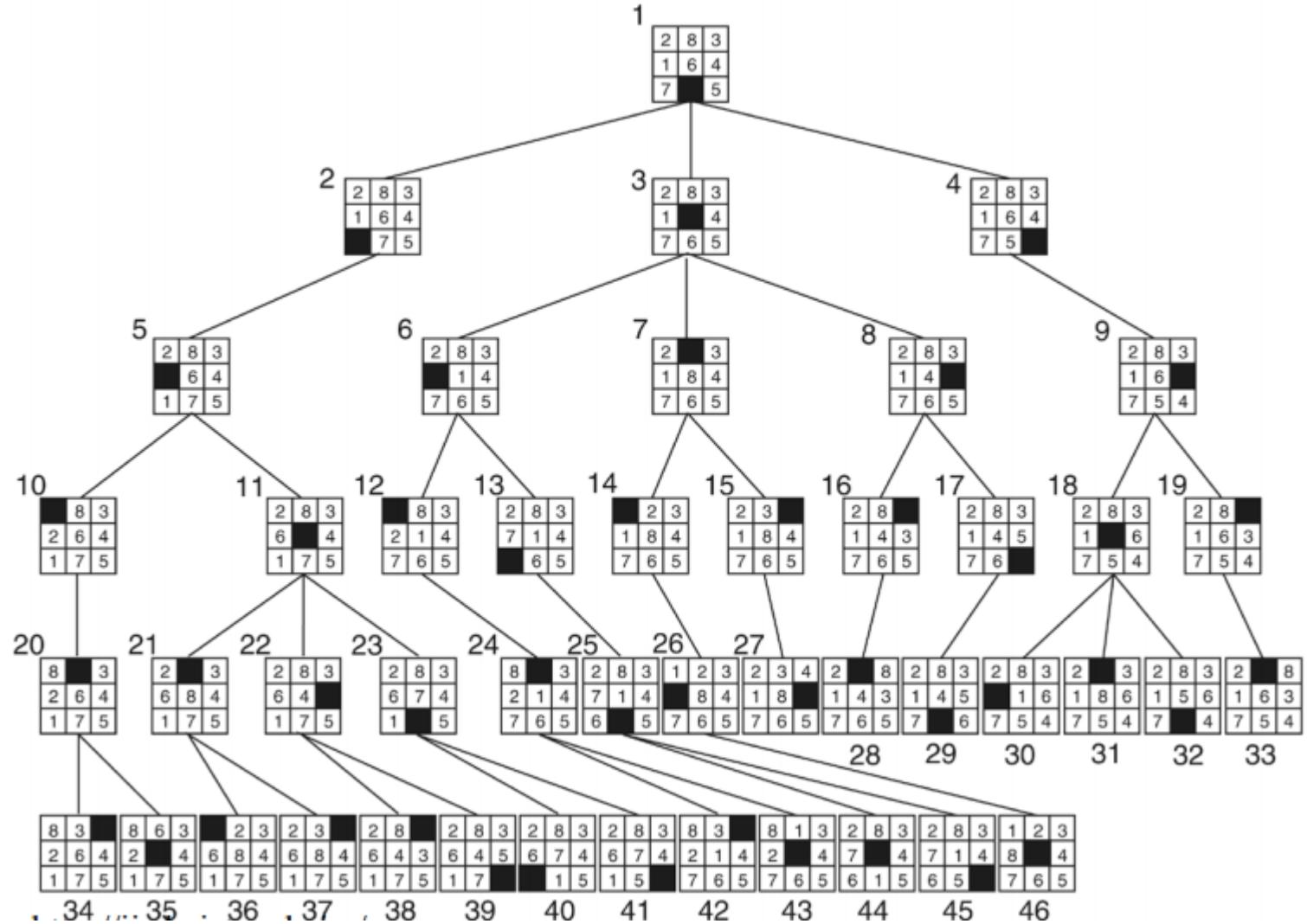
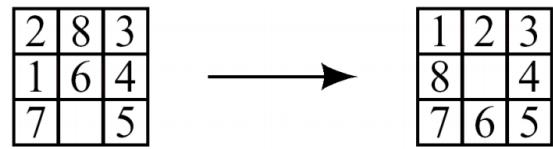
Uygulama: FIFO Kuyruğu



BFS – İkili Ağaç (Binary Tree)

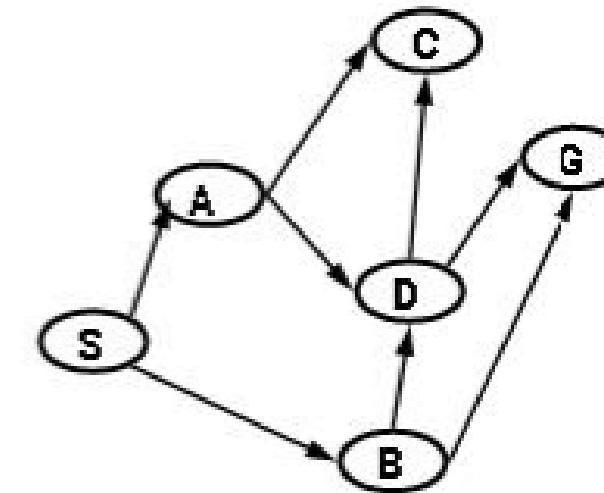


BFS – 8 Puzzle



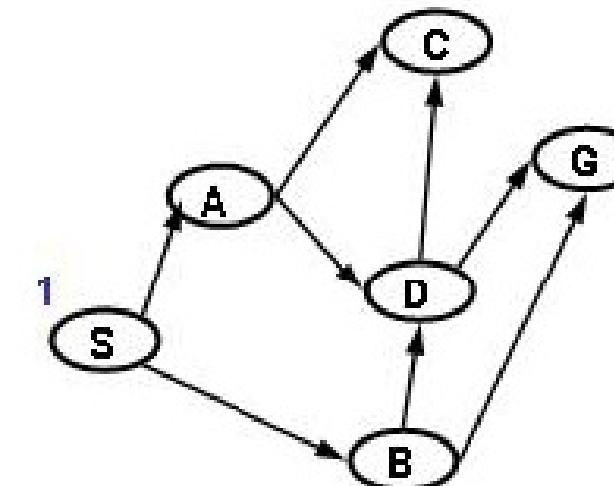
BFS – Örnek

	Q	Visited
1	(S)	S
2		
3		
4		
5		
6		



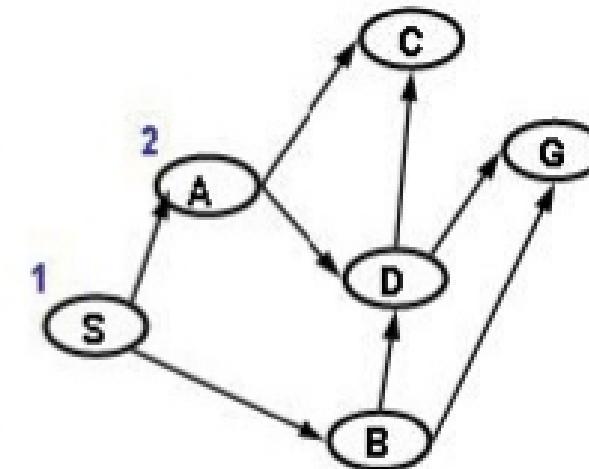
BFS – Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A,B,S
3		
4		
5		
6		



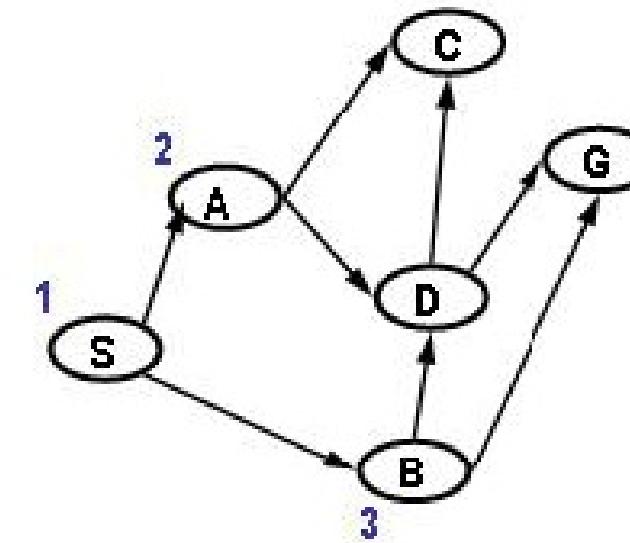
BFS – Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A,B,S
3	(B S) (C A S) (D A S)	C,D,B,A,S
4		
5		
6		



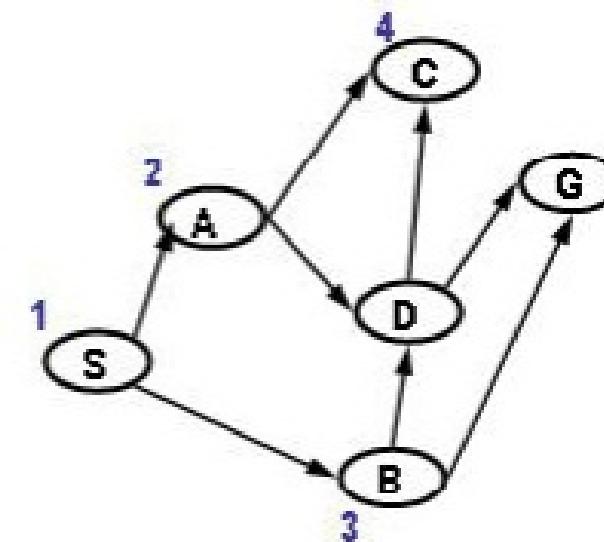
BFS – Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A,B,S
3	(B S) (C A S) (D A S)	C,D,B,A,S
4	(C A S) (D A S) (G B S)*	G,C,D,B,A,S
5		
6		



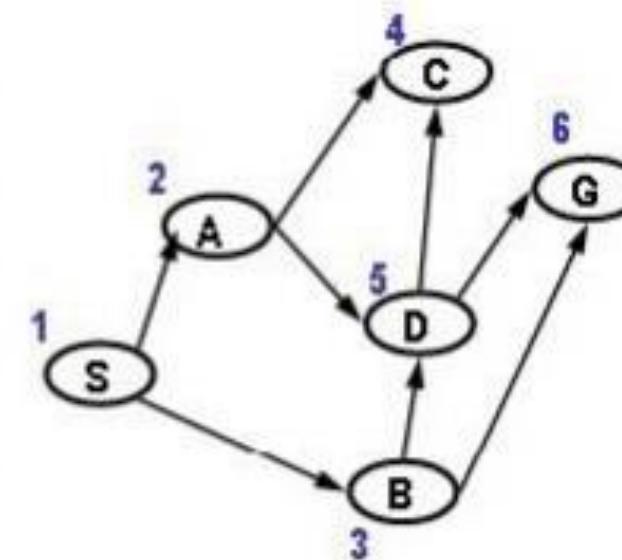
BFS – Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A,B,S
3	(B S) (C A S) (D A S)	C,D,B,A,S
4	(C A S) (D A S) (G B S)*	G,C,D,B,A,S
5	(D A S) (G B S)	G,C,D,B,A,S
6		



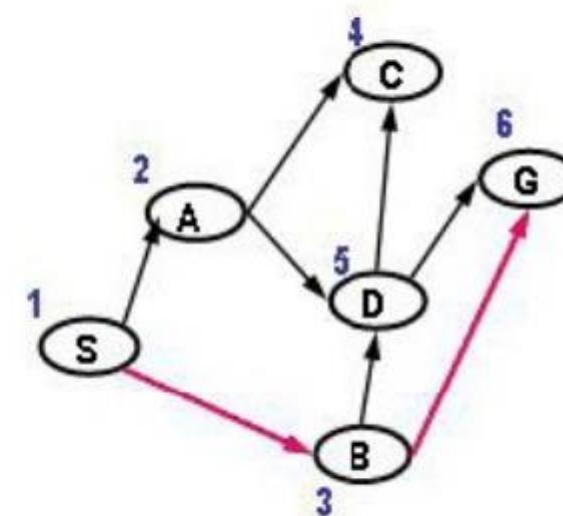
BFS – Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A,B,S
3	(B S) (C A S) (D A S)	C,D,B,A,S
4	(C A S) (D A S) (G B S)*	G,C,D,B,A,S
5	(D A S) (G B S)	G,C,D,B,A,S
6	(G B S)	G,C,D,B,A,S



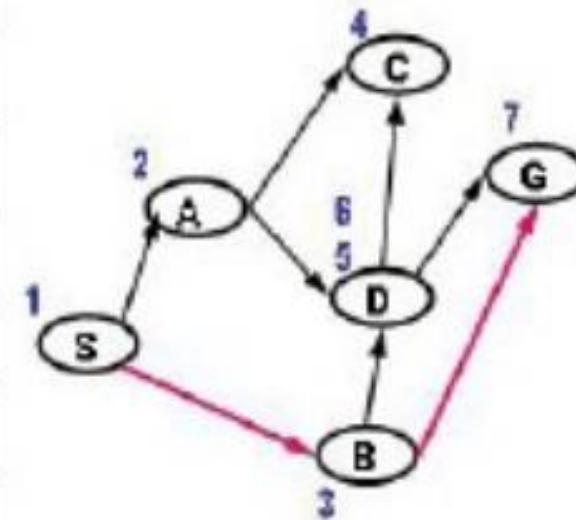
BFS – Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A,B,S
3	(B S) (C A S) (D A S)	C,D,B,A,S
4	(C A S) (D A S) (G B S)*	G,C,D,B,A,S
5	(D A S) (G B S)	G,C,D,B,A,S
6	(G B S)	G,C,D,B,A,S



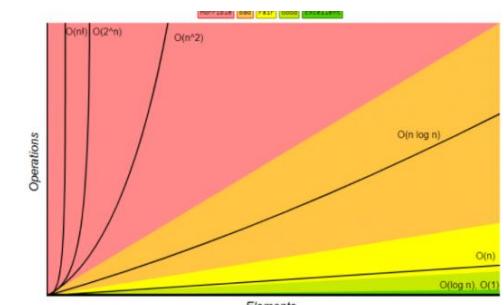
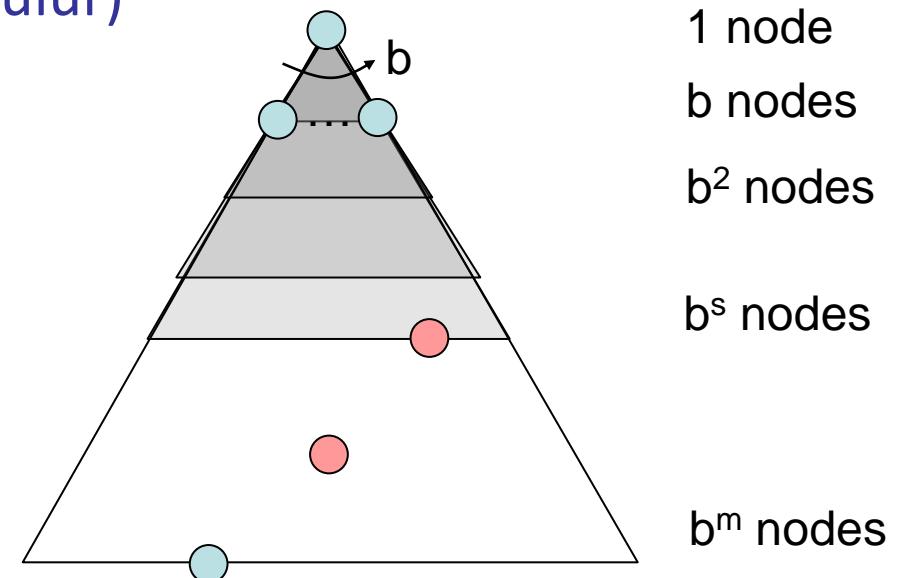
BFS (Visited List Olmadan)– Örnek

	Q
1	(S)
2	(A S) (B S)
3	(B S) (C A S) (D A S)
4	(C A S) (D A S) (D B S) (G B S)*
5	(D A S) (D B S) (G B S)
6	(D B S) (G B S) (C D A S) (G D A S)
7	(G B S) (C D A S) (G D A S) (C D B S) (G D B S)



BFS Özellikleri

- Dallanma faktörü b sabit ile hedef düğümü d seviyesinde varsayıyalım:
 - Zaman karmaşıklığı (oluşturulan düğüm sayısı ile ölçülür)
 - $1 + b + b^2 + \dots + b^d + b(b^d - 1) = O(b^{d+1})$
 - İlgili seviyede en sağa gitmeyi hedefler.
 - Uzay karmaşıklığı:
 - $O(b^{d+1})$
 - Tamdır
 - Optimal mi?
 - Tüm maliyetler 1 ise: Evet



BFS Özellikleri

Depth	Nodes	Time	Memory
2	110	.11 milliseconds	107 kilobytes
4	11,110	11 milliseconds	10.6 megabytes
6	10^6	1.1 seconds	1 gigabyte
8	10^8	2 minutes	103 gigabytes
10	10^{10}	3 hours	10 terabytes
12	10^{12}	13 days	1 petabyte
14	10^{14}	3.5 years	99 petabytes
16	10^{16}	350 years	10 exabytes

Figure 3.13 Time and memory requirements for breadth-first search. The numbers shown assume branching factor $b = 10$; 1 million nodes/second; 1000 bytes/node.

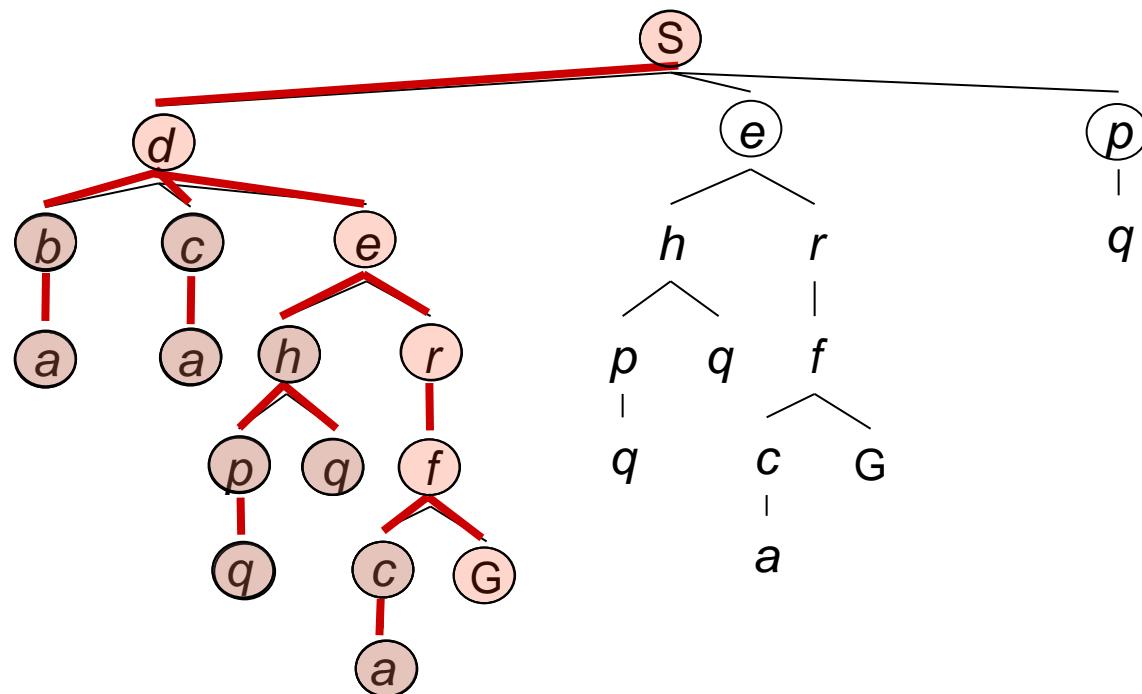
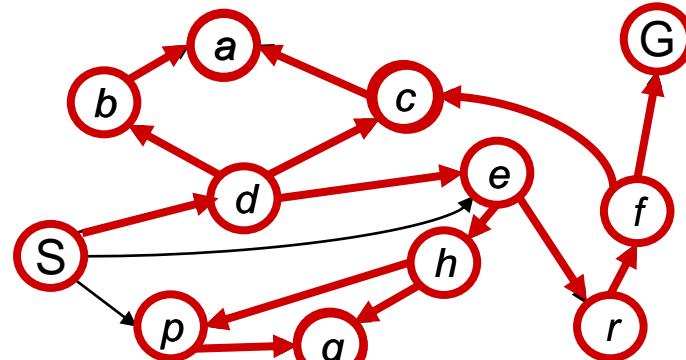
Derinlik Öncelikli Arama (DFS)



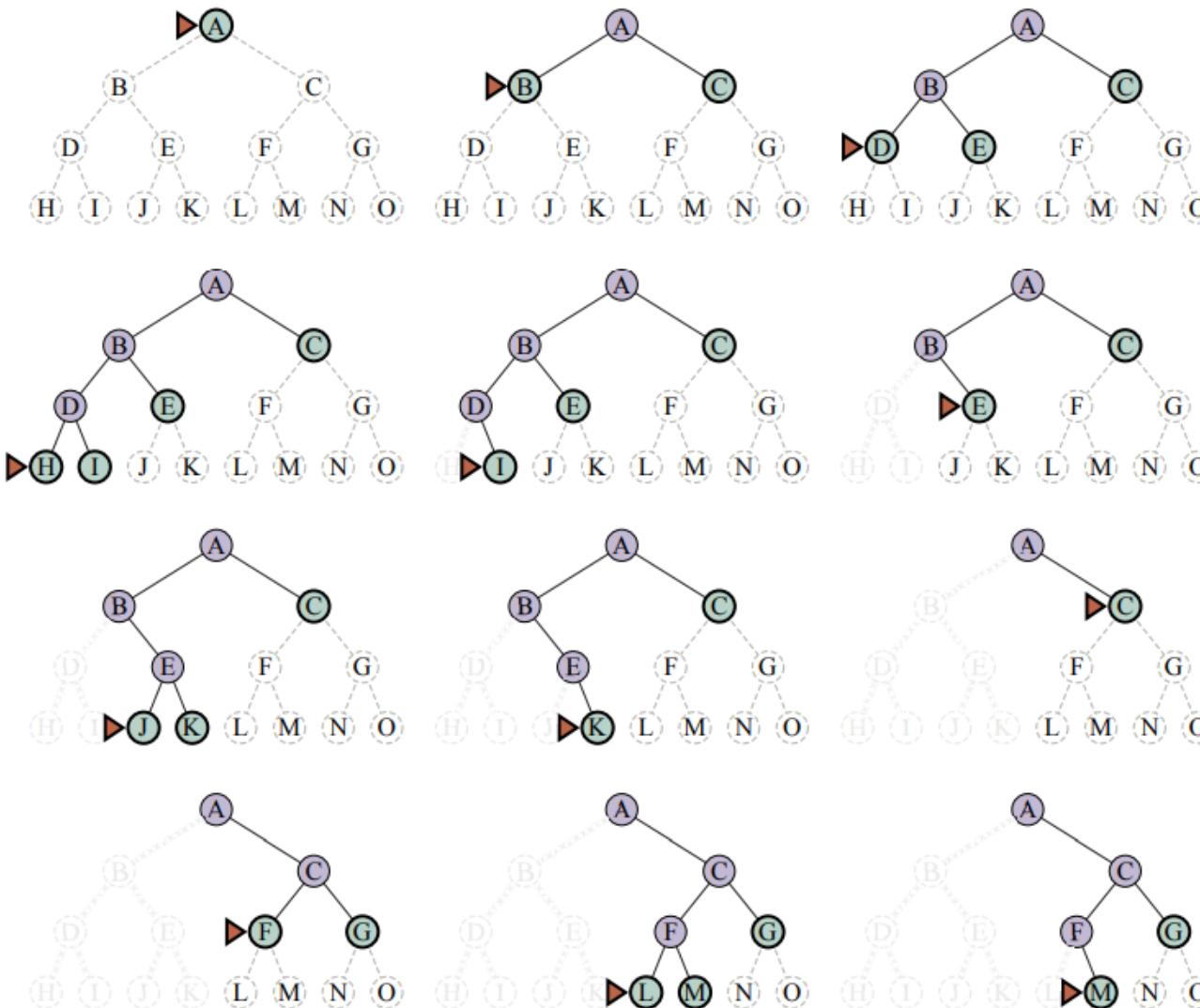
Derinlik Öncelikli Arama

Strateji: Önce en derin
düğümü genişlet

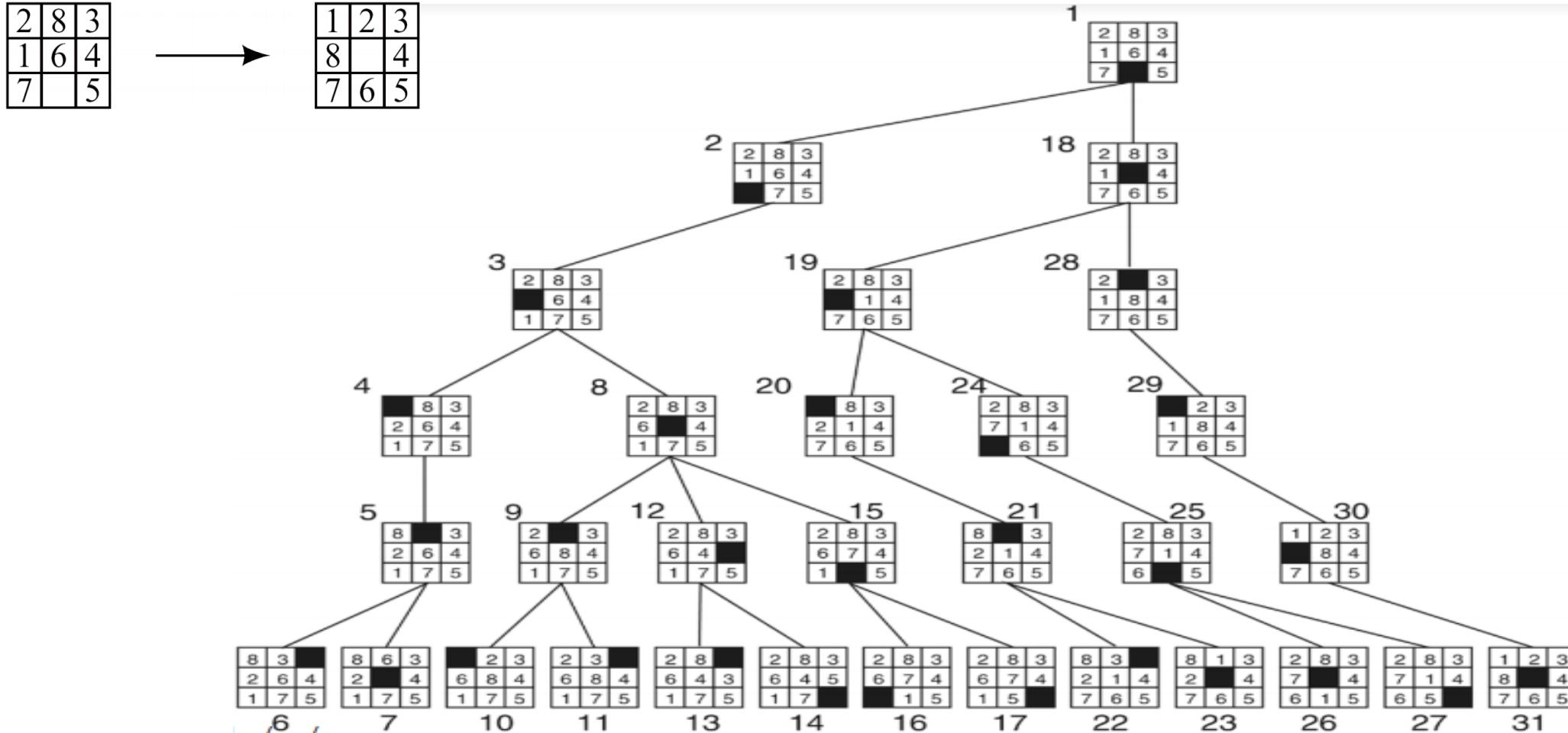
Uygulama: LIFO Kuyrugu



DFS – İkili Ağaç

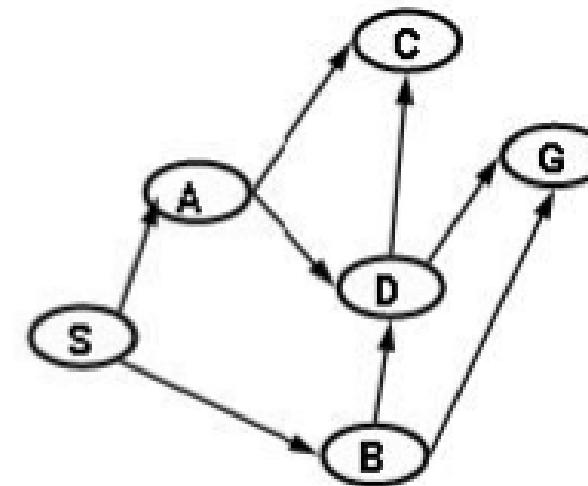


DFS – 8 Puzzle



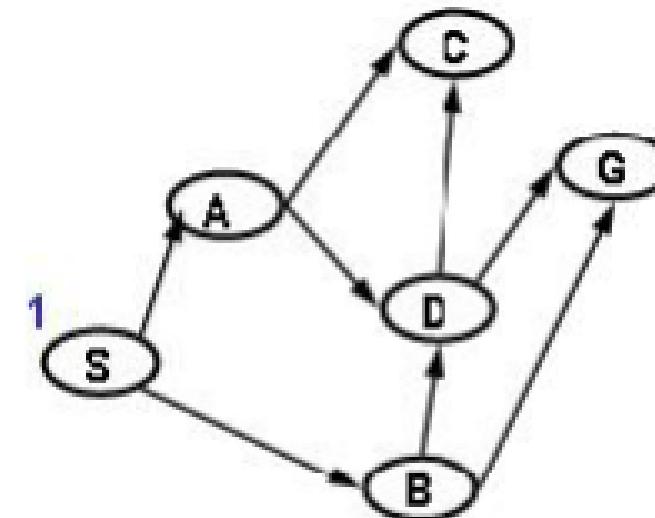
DFS - Örnek

	Q	Visited
1		
2		
3		
4		
5		



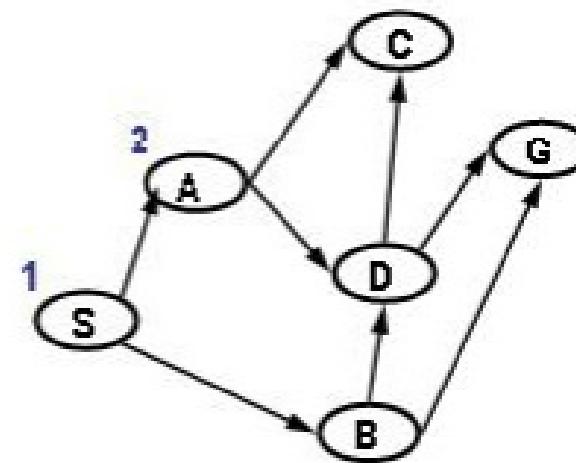
DFS - Örnek

	Q	Visited
1	(S)	S
2		
3		
4		
5		



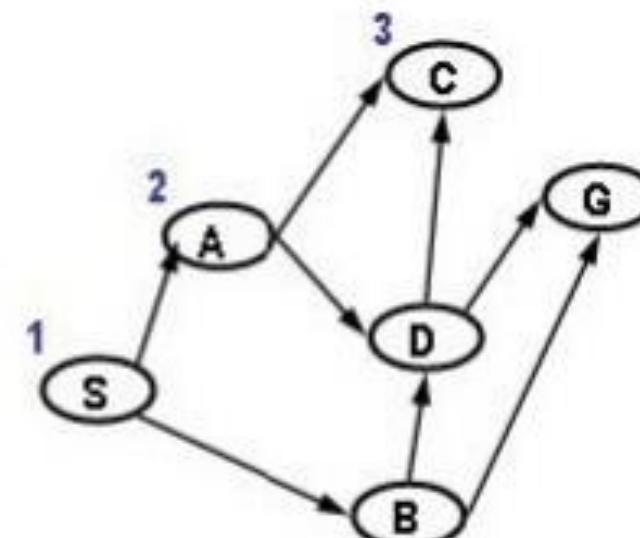
DFS - Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A, B, S
3		
4		
5		



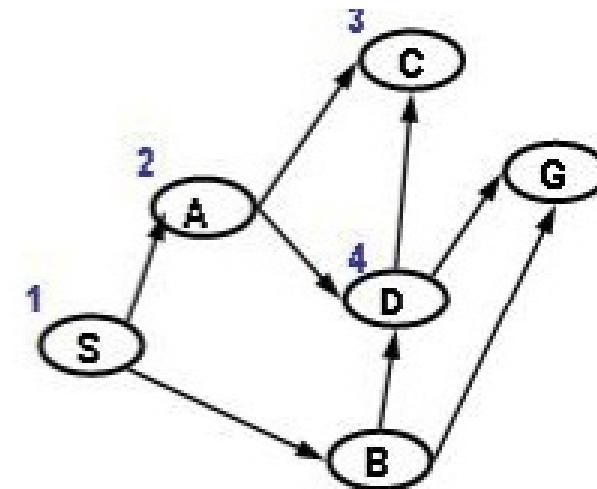
DFS - Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A, B, S
3	(C A S) (D A S) (B S)	C,D,B,A,S
4		
5		



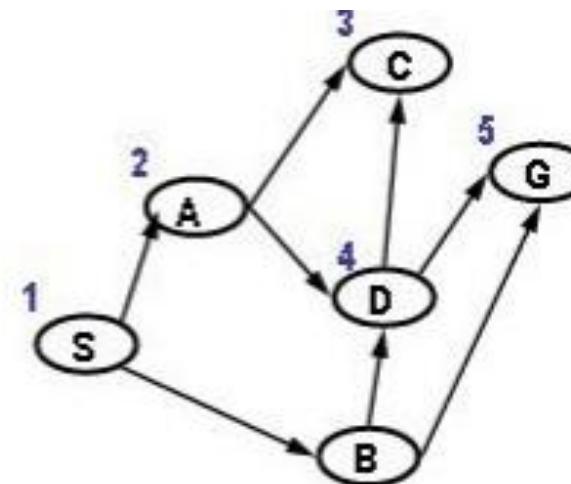
DFS - Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A, B, S
3	(C A S) (D A S) (B S)	C,D,B,A,S
4	(D A S) (B S)	C,D,B,A,S
5		



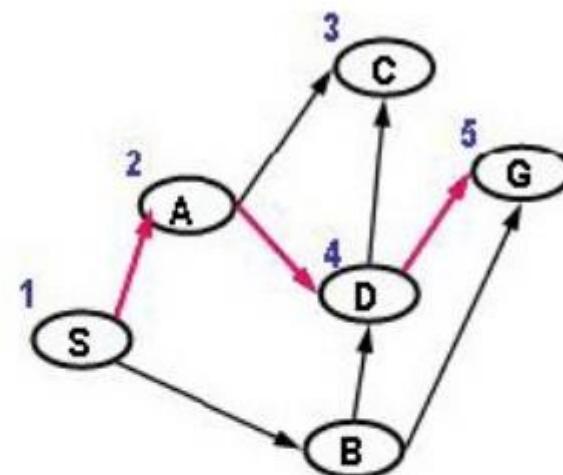
DFS - Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A, B, S
3	(C A S) (D A S) (B S)	C,D,B,A,S
4	(D A S) (B S)	C,D,B,A,S
5	(G D A S) (B S)	G,C,D,B,A,S



DFS - Örnek

	Q	Visited
1	(S)	S
2	(A S) (B S)	A, B, S
3	(C A S) (D A S) (B S)	C,D,B,A,S
4	(D A S) (B S)	C,D,B,A,S
5	(G D A S) (B S)	G,C,D,B,A,S



DFS Özellikleri

- Zaman karmaşıklığı

- En kötü durumda, tüm alanı arayın
- Hedef d seviyesinde olabilir, ancak ağaç m seviyesine devam edebilir, $m \geq d$
- $O(b^m)$
- Özellikle ağaç sonsuz derinlikte ise kötü

- Uzay karmaşıklığı

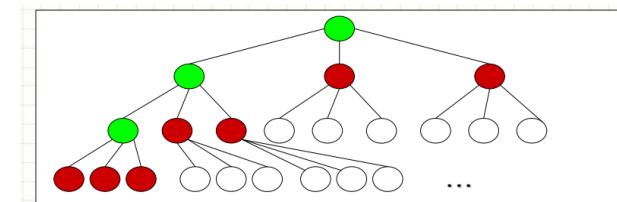
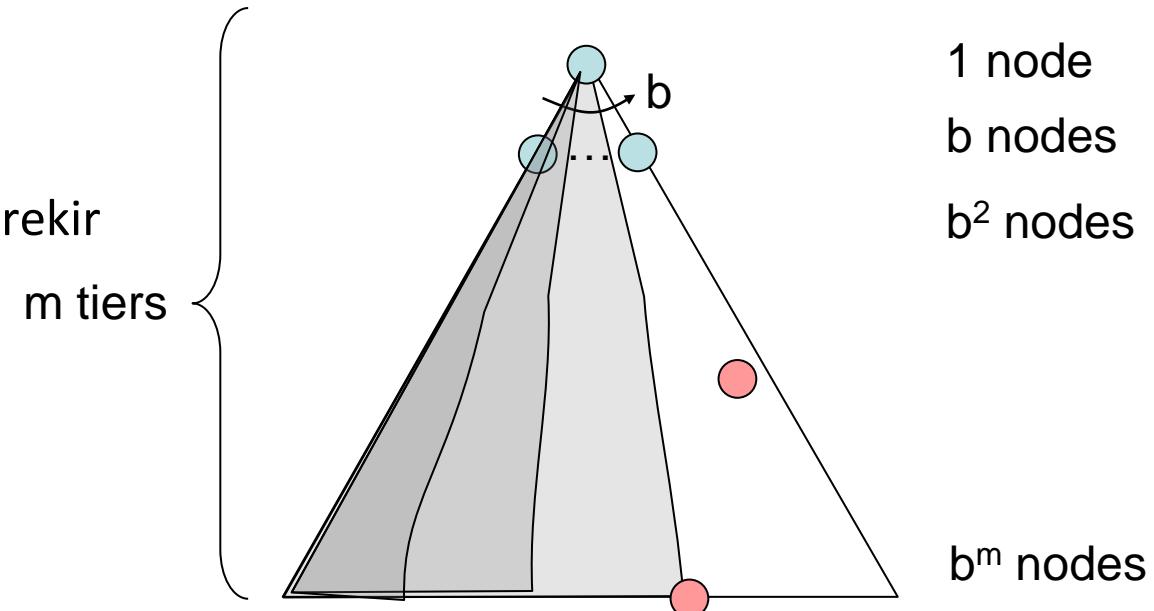
- Her seviyede sadece bir grup çocuğu kaydetmeniz gereklidir
- $1 + b + b + \dots + b$ (toplam m seviye) = $O(bm)$

- Tamlık?

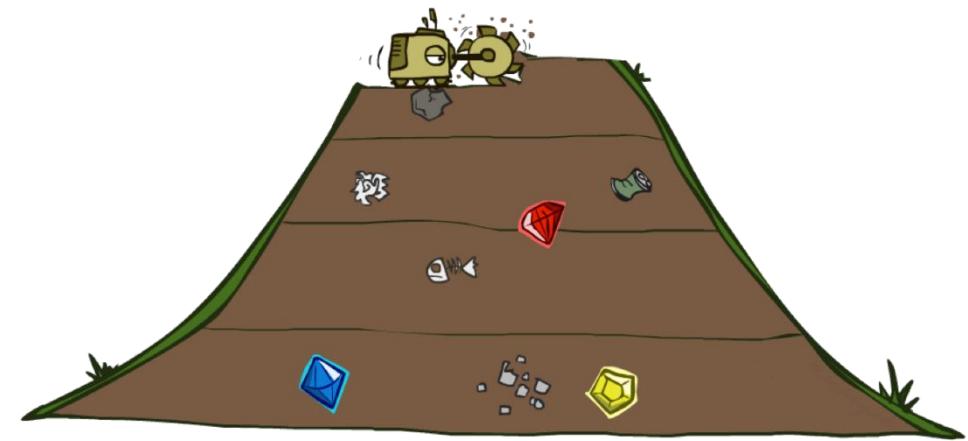
- Her zaman çözüm bulmayı bilir
- Tam değildir

- Optimal?

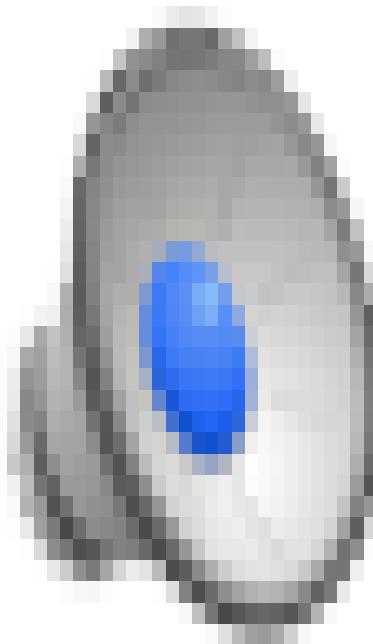
- Çözüm mutlaka en kısa veya en düşük maliyetli değildir
- Optimal değildir!



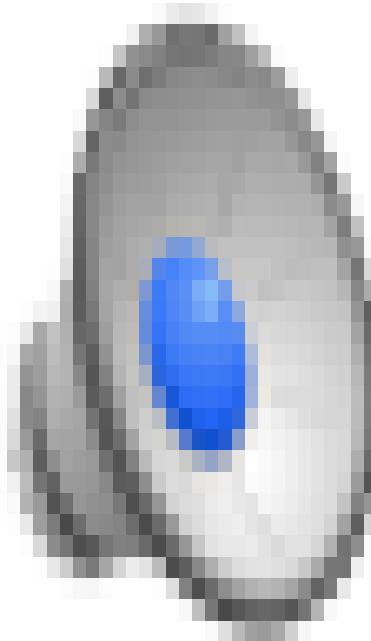
DFS vs BFS



Video of Demo Maze Water DFS/BFS (part 1)



Video of Demo Maze Water DFS/BFS (part 2)



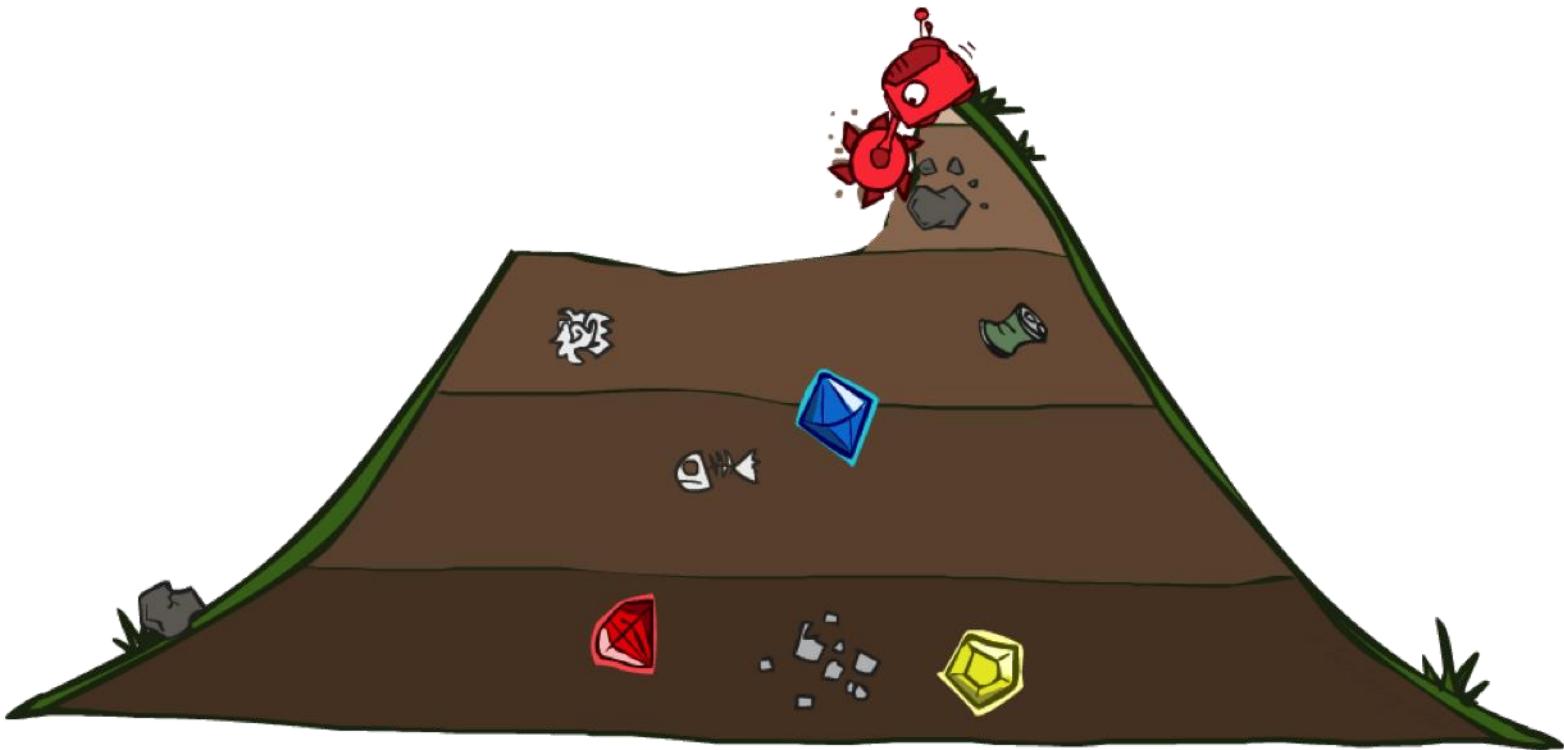
DFS vs BFS

- BFS ne zaman DFS'den daha iyi performans gösterir?
- DFS ne zaman BFS'den daha iyi performans gösterir?

Derinlik Sınırlı Arama

- Sonsuz durum uzaylarında derinlik öncelikli arama başarısızlığa düşüyor.
- Bu durumun etkisini hafifletmek için bir derinlik sınırı belirlenir.
- Derinlik öncelikli arama: derinlik sınırı « $|$ »
 - Depth first search 'de $|=$ sonsuz
- $|$. derinlikteki düğümlerin önemi yok varsayıyoruz.
- Çözüm d . derinlikte ve biz $| < d$ seçtik çözüme ulaşamayız
 - Genellikle d bilinmediğinde ortaya çıkar.
- $| > d$ seçersek bu sefer de optimal olmama sorunu karşımıza çıkar.
- Zaman karmaşıklığı $O(b^{|})$, uzay karmaşıklığı $O(bl)$ 'dir.

Sabit Maliyet Araması (UCS)



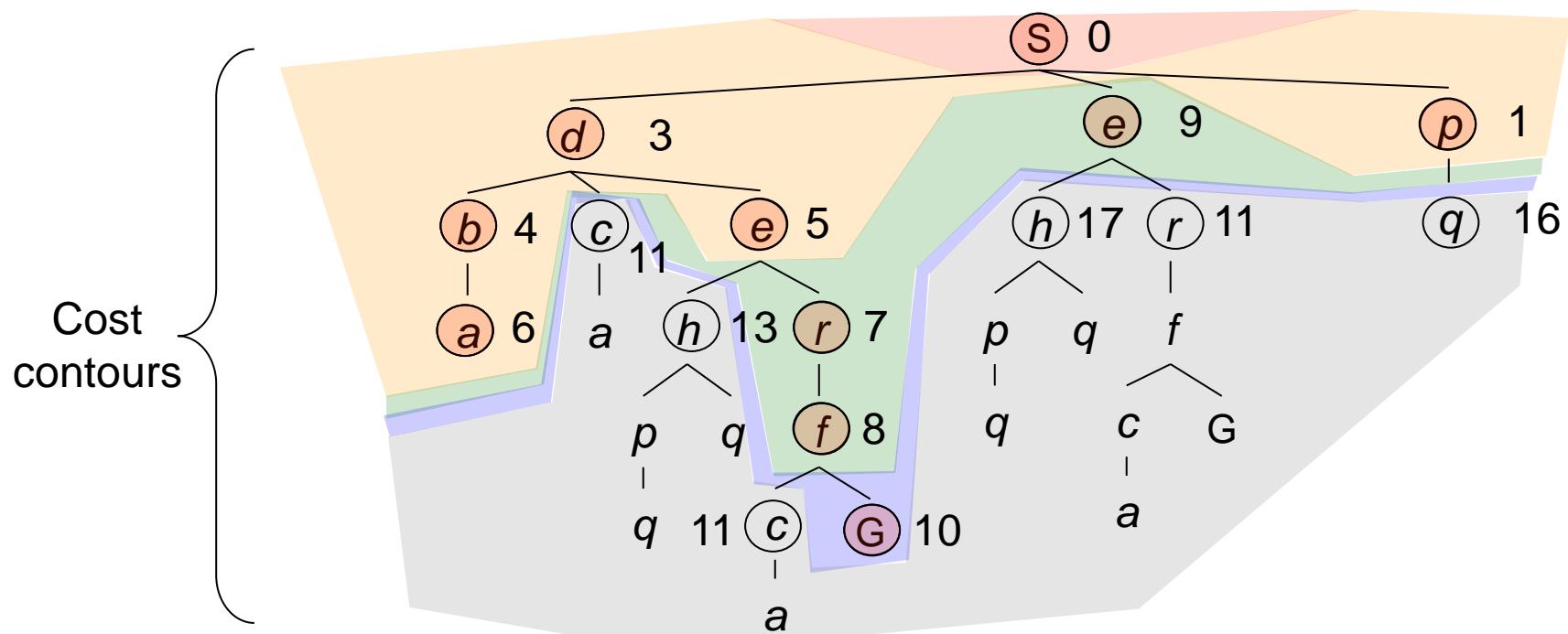
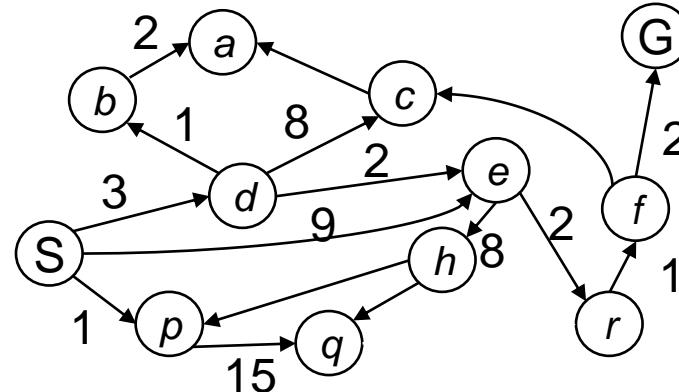
Sabit Maliyet Araması

Strateji:

Önce en ucuz düğümü genişlet

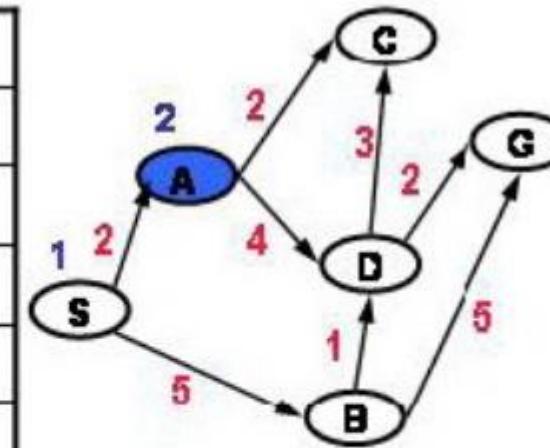
Uygulama:

Öncelik Kuyruğu



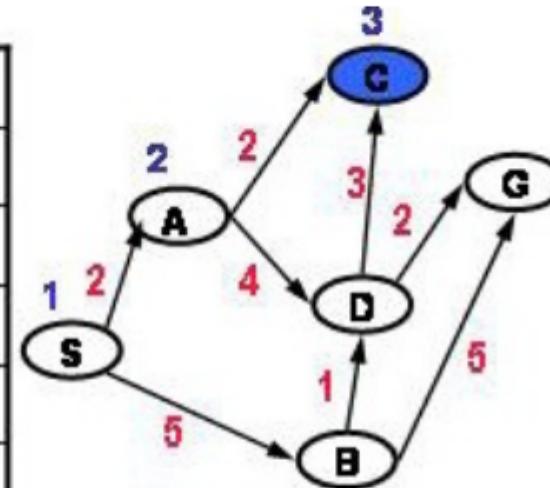
Sabit Maliyet Araması

	Q
1	(0 S)
2	(2 A S) (5 B S)



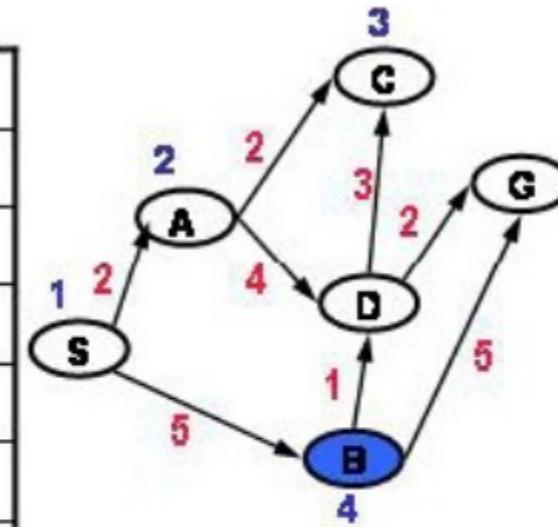
Sabit Maliyet Araması

	Q
1	(0 S)
2	(2 A S) (5 B S)
3	(4 C A S) (6 D A S) (5 B S)



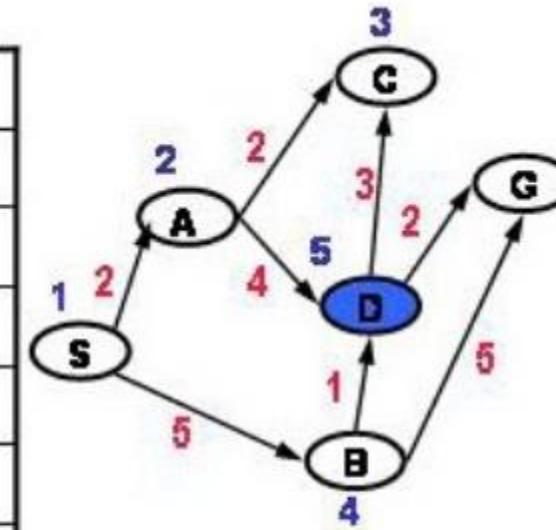
Sabit Maliyet Araması

	Q
1	(0 S)
2	(2 A S) (5 B S)
3	(4 C A S) (6 D A S) (5 B S)
4	(6 D A S) (5 B S)



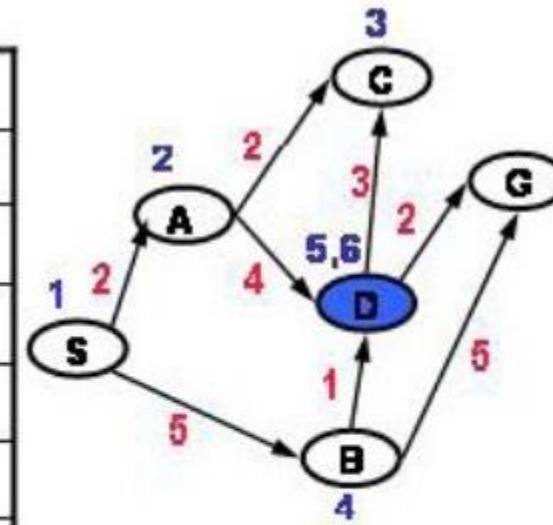
Sabit Maliyet Araması

	Q
1	(0 S)
2	(2 A S) (5 B S)
3	(4 C A S) (6 D A S) (5 B S)
4	(6 D A S) (5 B S)
5	(6 D B S) (10 G B S) (6 D A S)



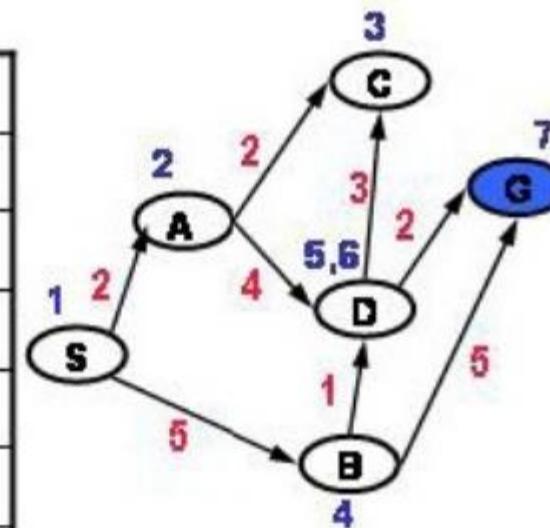
Sabit Maliyet Araması

	Q
1	(0 S)
2	(2 A S) (5 B S)
3	(4 C A S) (6 D A S) (5 B S)
4	(6 D A S) (5 B S)
5	(6 D B S) (10 G B S) (6 D A S)
6	(8 G D B S) (9 C D B S) (10 G B S) (6 D A S)



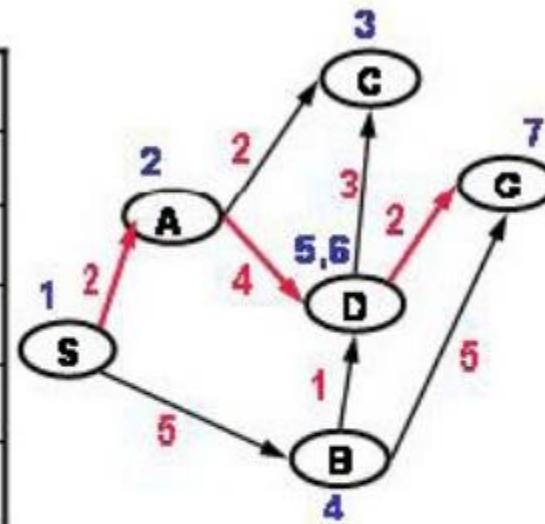
Sabit Maliyet Araması

	Q
1	(0 S)
2	(2 A S) (5 B S)
3	(4 C A S) (6 D A S) (5 B S)
4	(6 D A S) (5 B S)
5	(6 D B S) (10 G B S) (6 D A S)
6	(8 G D B S) (9 C D B S) (10 G B S) (6 D A S)
7	(8 G D A S) (9 C D A S) (8 G D B S) (9 C D B S) (10 G B S)



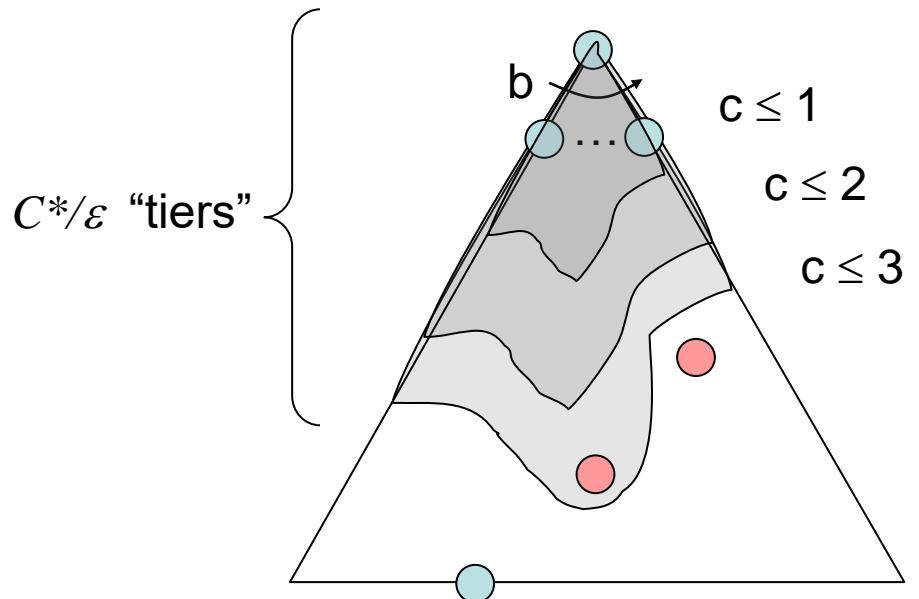
Sabit Maliyet Araması

	Q
1	(0 S)
2	(2 A S) (5 B S)
3	(4 C A S) (6 D A S) (5 B S)
4	(6 D A S) (5 B S)
5	(6 D B S) (10 G B S) (6 D A S)
6	(8 G D B S) (9 C D B S) (10 G B S) (6 D A S)
7	(8 G D A S) (9 C D A S) (8 G D B S) (9 C D B S) (10 G B S)



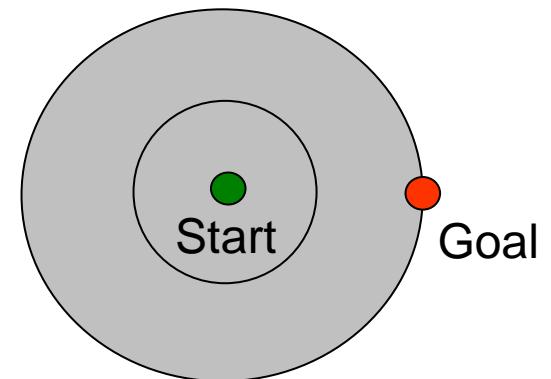
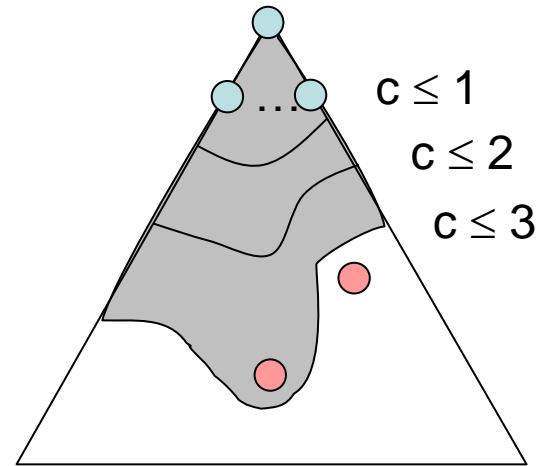
UCS Özellikleri

- C^* : optimal çözümün maliyeti
- ε : en düşük adım maliyeti
- Zaman Karmaşıklığı?
 - $O(b^{C^*/\varepsilon})$
 - Her adımın maliyeti aynı ise $O(bd)$
 - Çünkü genişlik-öncelikliye dönüşür
- Yer Karmaşıklığı?
 - C^*/ε katman var ise: $O(b^{C^*/\varepsilon})$
- Tamlık?
 - Evet
- Optimal?
 - Her adım maliyeti $\geq \varepsilon > 0$ ise evet
 - Adım maliyetleri sınırlı ise evet optimaldir.

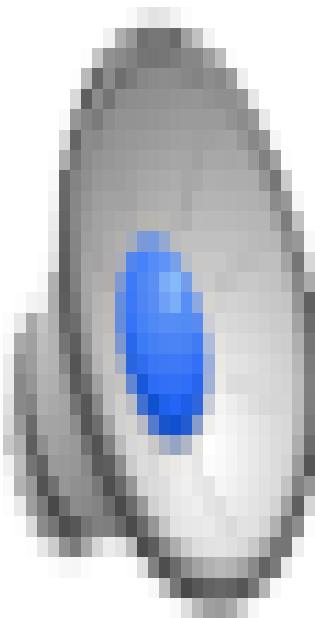


UCS Analiz

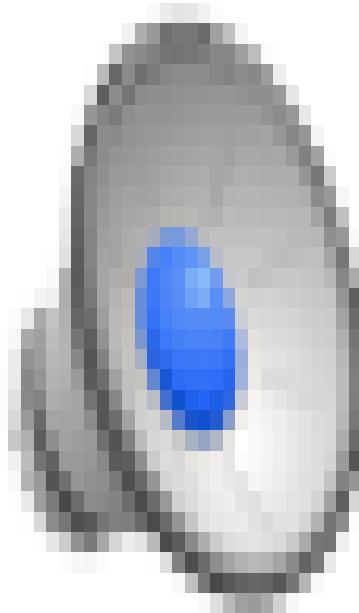
- İyi Tarafı:
 - UCS tam ve optimaldir!
- Kötü Tarafı:
 - Hedef konumu hakkında bilgi yoktur.
 - Her yöndeki seçenekleri araştırır.



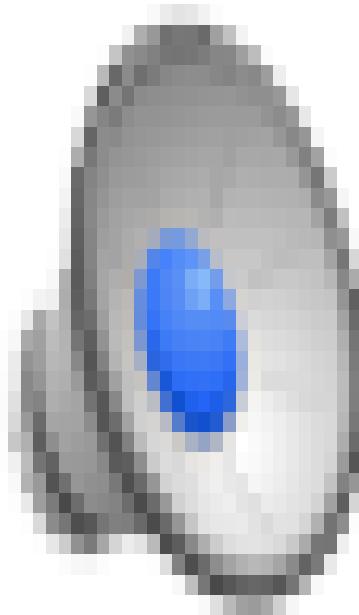
Video of Demo Empty UCS



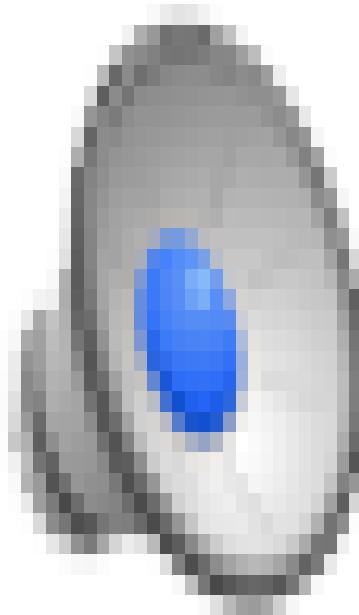
Video of Demo Maze with Deep/Shallow Water --- DFS, BFS, or UCS? (part 1)



Video of Demo Maze with Deep/Shallow Water --- DFS, BFS, or UCS? (part 2)



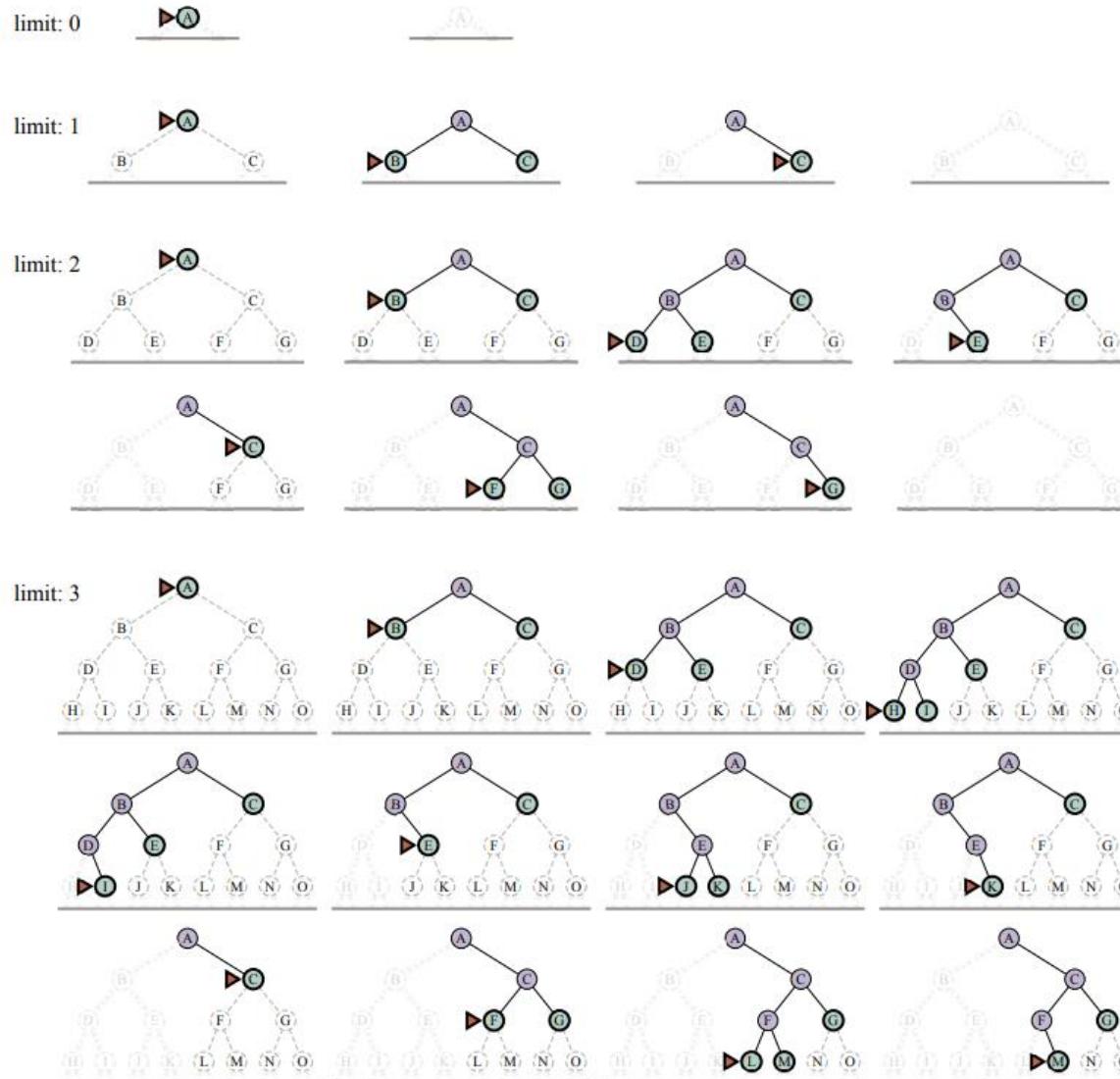
Video of Demo Maze with Deep/Shallow Water --- DFS, BFS, or UCS? (part 3)



Yinelemeli Derinleşen Arama

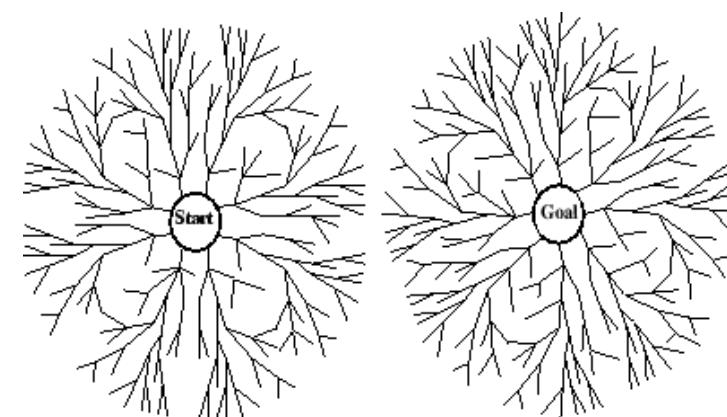
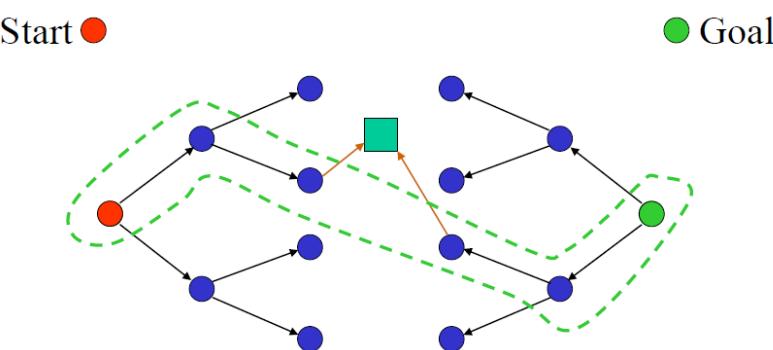
- DFS ve BFS yöntemlerinin iyi yönlerini birleştirir.
 - BFS gibi optimal ve tamdır!
 - DFS gibi az bellek gerektirir!
-
- Dallanma faktörü b ile derinlik d'ye kadar derinlik sınırlı aramada oluşturulan düğüm sayısı:
 - $N_{DLS} = b^0 + b^1 + b^2 + \cdots + b^{d-2} + b^{d-1} + b^d$
 - Dallanma faktörü b ile derinlik d'ye kadar yinelemeli derinleştirme aramasında oluşturulan düğüm sayısı:
 - $N_{IDS} = (d+1)b^0 + db^1 + (d-1)b^2 + \cdots + 3b^{d-2} + 2b^{d-1} + 1b^d$
 - Zaman karmaşıklığı:
 - $O(b^d)$
 - Yer karmaşıklığı:
 - $O(bd)$

Yinelemeli Derinleşen Arama



İki Yönlü Arama

- Temel düşünce: Es zamanlı arama işletmek
 - Birincisi başlangıç durumundan ileriye doğru
 - İkincisi hedef durumdan geriye doğru
- İki arama kesişince algoritma sonlanır.
- $b^{d/2} + b^{d/2} \ll b^d$
- Zaman ve Yer Karmaşıklığı
 - $O(b^{d/2})$
- Arama uzayının çok geniş olduğu uygulamalarda zaman karmaşıklığını azalttığından çözüm olabilir (sosyal paylaşım siteleri gibi)



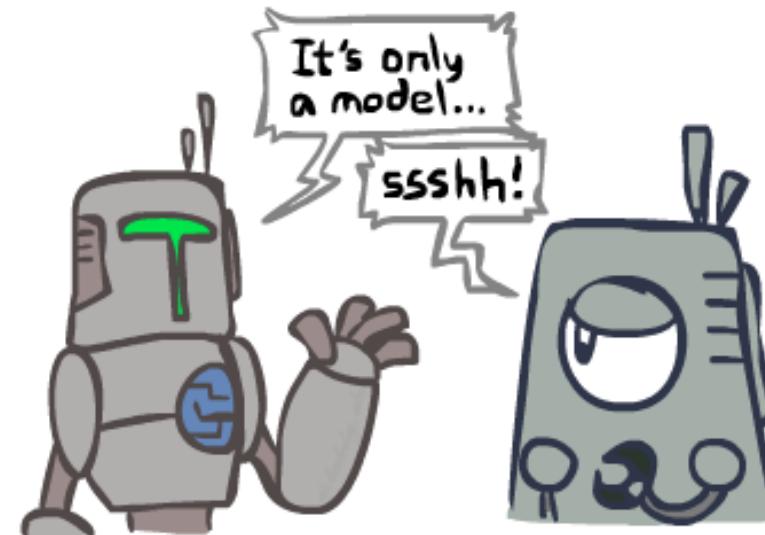
Arama Algoritmaları Karşılaştırması

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes ¹	Yes ^{1,2}	No	No	Yes ¹	Yes ^{1,4}
Optimal cost?	Yes ³	Yes	No	No	Yes ³	Yes ^{3,4}
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$

Figure 3.15 Evaluation of search algorithms. b is the branching factor; m is the maximum depth of the search tree; d is the depth of the shallowest solution, or is m when there is no solution; ℓ is the depth limit. Superscript caveats are as follows: ¹ complete if b is finite, and the state space either has a solution or is finite. ² complete if all action costs are $\geq \epsilon > 0$; ³ cost-optimal if action costs are all identical; ⁴ if both directions are breadth-first or uniform-cost.

Arama ve Modeller

- Arama dünyadaki modeller üzerinde çalışır
 - Ajanlar gerçek dünyadaki tüm planları denemezler!
 - Planlama teoride(simülasyonda) kalır !!!
 - Model ne kadar iyiysse arama da o kadar iyi olacaktır...



Referanslar

- Artificial Intelligence A Modern Approach, Stuart Russell and Peter Norvig, Prentice Hall Series in Artificial Intelligence.
- Yapay Zeka, Vasif Vagifoğlu Nabihev, Seçkin Yayıncılık
- <http://www.cs.bilkent.edu.tr/~duygulu/Courses/CS461/Notes/Search.pdf>
- http://aytugonan.cbu.edu.tr/YZM3217/LectureNotes/YZM3217_lecture4.pdf