



SAKARYA
ÜNİVERSİTESİ

BSM 310 YAPAY ZEKA

CEMİL ÖZ, İSMAİL ÖZTEL

~ SEZGİSEL PROBLEM ÇÖZME YAKLAŞIMI ~

KONULAR

- Sezgisellik
- Graflar
- Durum Uzayı
- Arama Yaklaşımları
- A* algoritması
- Örnek Problemler

Sezgisellik (Heuristic)

- İnsanlar gündelik yaşamlarında bir çok sezgisel yaklaşım sergiler.
- İnsanlarda sezgisel olarak çözülebilen problemler; problemi çözen zeka ve içinde bulunulan durum gibi parametrelere göre farklılık gösterir.
- Algoritma, içinde belirsizlik barındırmaz ama yapay zeka problemlerinde belirsizlikler mevcuttur.
- Bu sebeple yapay zeka için sezgisellik, algoritma kavramına eşdeğerdir.
- Yapay zeka problemlerinin çözümünde sezgisel yaklaşımlar kullanılır.

Sezgisellik (Heuristic)

- Sezgisel yaklaşımlar, problem için en iyi çözümü garanti etmezler.
- Bu yaklaşımlar, problemlere kabul edilebilir bir zaman içinde bir çözüm bulacağını garanti ederler.
- Genellikle bulunan çözüm en iyiye yakın bir çözümdür.
- Sezgisel yaklaşımların her problemi çözebileceği söylenemez.
- Bir problem için sonuç üretemeyen bir yaklaşım, bir başka problem çözümü için uygun olabilir.

Sezgisellik (Heuristic)

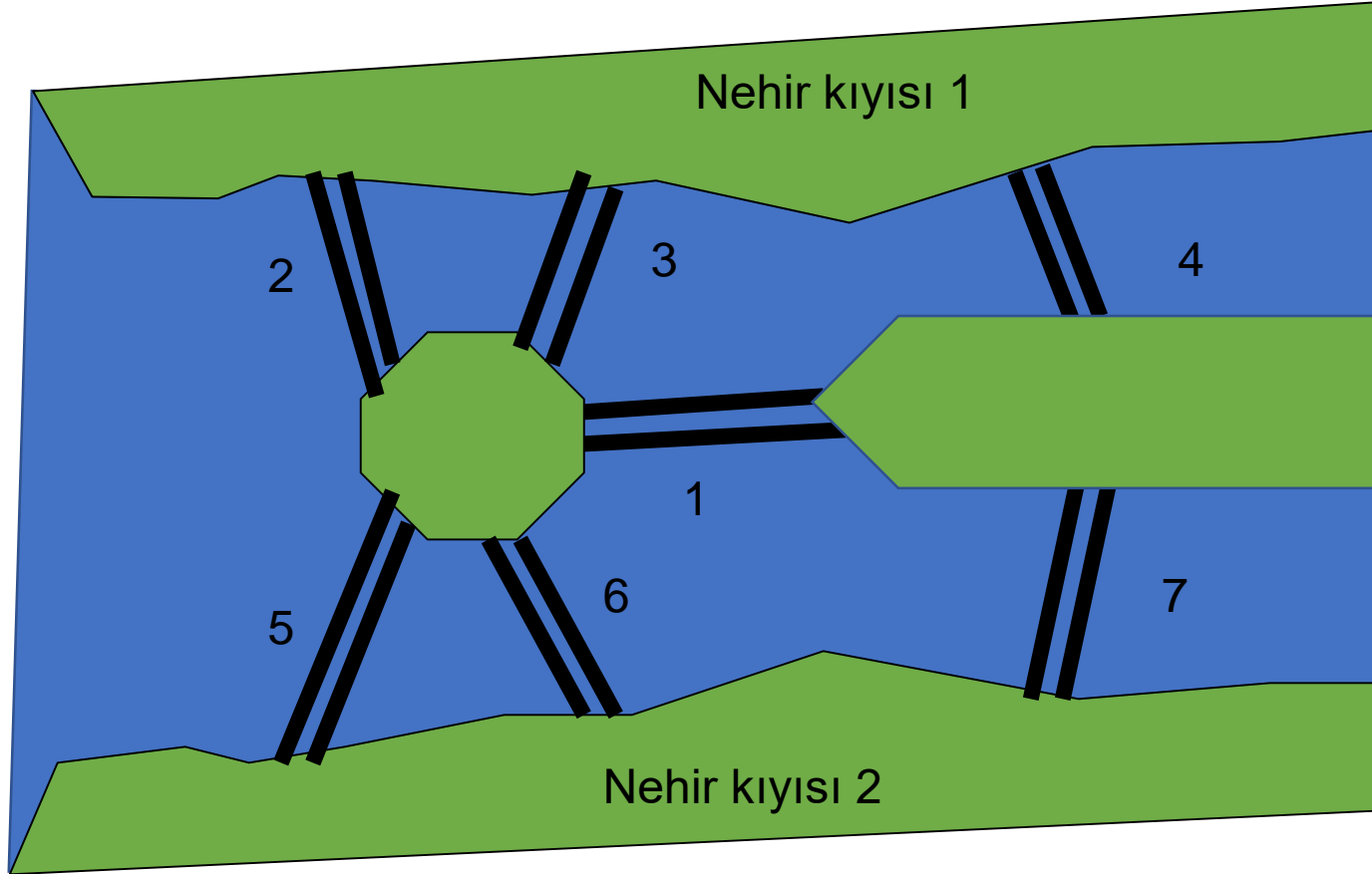
- Problemler genel olarak iki sınıfa ayrılabilir:
 - İyi biçimlendirilmiş problemler: Algoritmik yöntemler ile problemin çözümünün mümkün olduğu problemlerdir.
 - Ör: Teoremlerin ispatı
 - Kötü biçimlendirilmiş problemler: Algoritmik yöntemler ile çözülemeyen problemlerdir.
 - Gündelik hayattaki problemlerin çoğu kötü biçimlendirilmiştir.
 - Ör: Satranç oyununda bir hamle yapılması
 - Ör: Hücumda futbolcunun pas vermesi mi şut çekmesi mi problemi
 - Ör: Astronot Gagarin'in İngiltere kraliçesi tarafından yemeğe davet edilmiştir. Masada 5'er adet çatal, kaşık ve bıçak bulunmaktadır. Bu konuda deneyimi olmayan Gagarin bunları rastgele kullanmıştır. Son olarak çay servisi esnasında en büyük kaşık kalmıştır.
 - Genel olarak bu tür problemlerde amaç en iyi çözümü üretmek değil, en kısa zamanda probleme çözüm üretmektir.

Sezgisellik (Heuristic)

- Sezgisel yöntemlerle çözüm ağacındaki tüm düğümler değil, yalnızca en avantajlı düğümler incelenir.
- Bu durum zamandan tasarruf sağlasa da en kötü durumda ağacın tüm düğümlerini dolanmak gerekebilir.

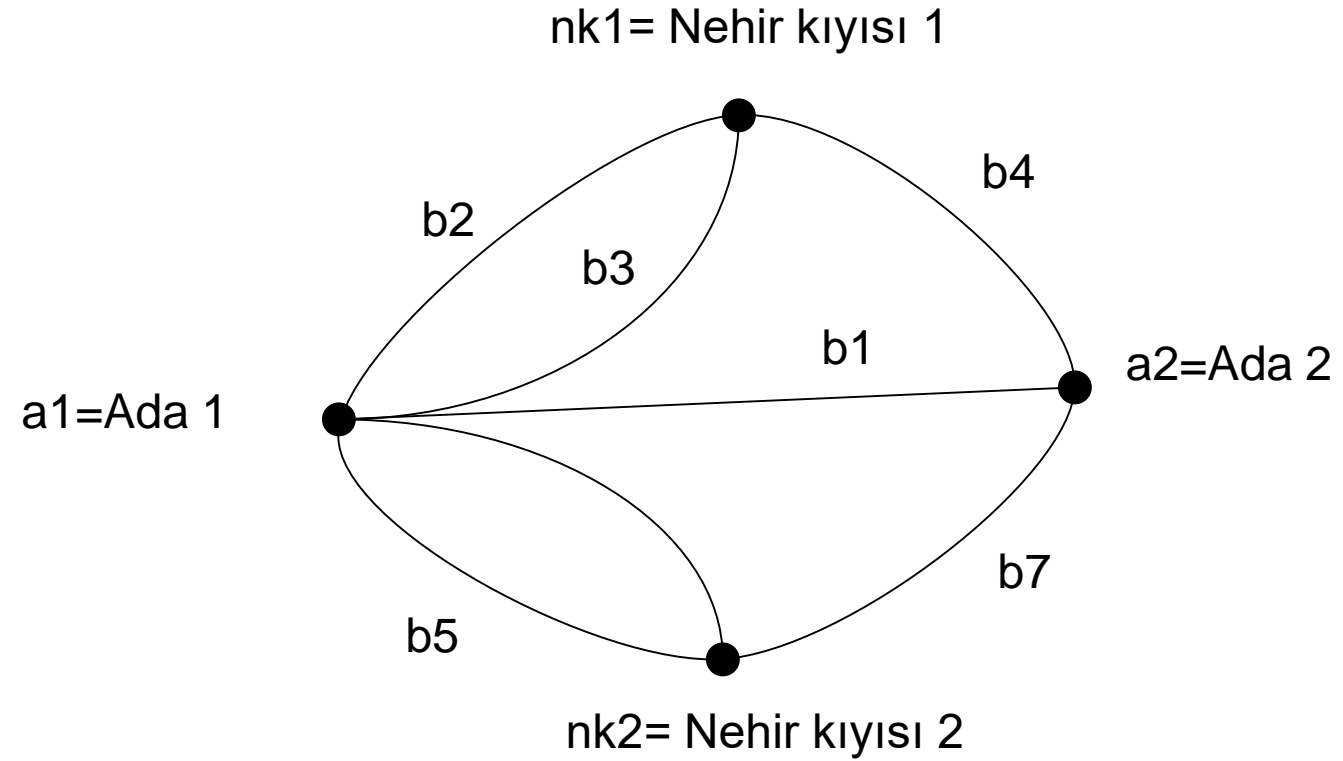
Graflar

- Königsberg / Kaliningrad



Graflar

- Königsberg / Kaliningrad



Graflar

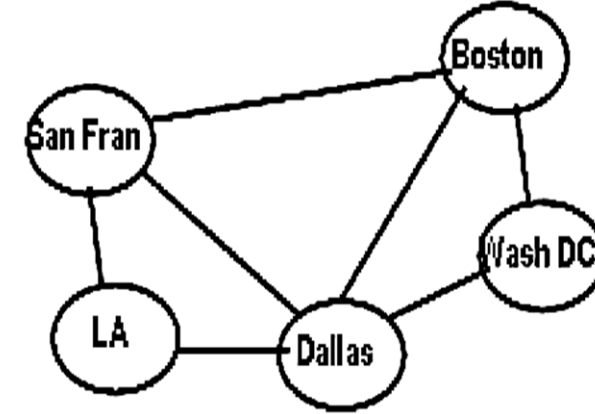
- Graf teorisi 18. yy ilk yıllarında Avusturyalı matematikçi Leonhard Euler tarafından ortaya atılmıştır.
- Bir düğümler kümesi ve düğümleri birleştiren kenarlar topluluğuna graf denir.
- Düğümleri birbirine bağlayan kenarların başlangıcı ve sonu var ise graf "yönlü graf" olarak isimlendirilir.
- Bir graftaki iki ayrı düğüm tek bir kenar ile birbirine bağlanıyorsa buna yalın graf denir.
- Bir graftaki iki ayrı düğüm birden çok kenar ile bağlanıyorsa buna bağlantılı graf denir.

Graflar

- Bir düğümden çıkan yolların sayısı, o düğümün derecesini verir.
- Eğer bağlantılı graf içerisinde süreklilik yok ise bu tür graflara ağaç denilmektedir.
- Diğer bir tanımla ağaç, bir kök düğüm, sonlu sayıda düğümleri ve onları birbirine bağlayan dalları olan bir yapıdır.
- Ağaçlar köklere sahiptir ve kök en üste yer alır.
- Yapay zeka problemlerinde durum uzaylarının ya da çözüm ağaçlarının gösteriminde kullanılabilir.

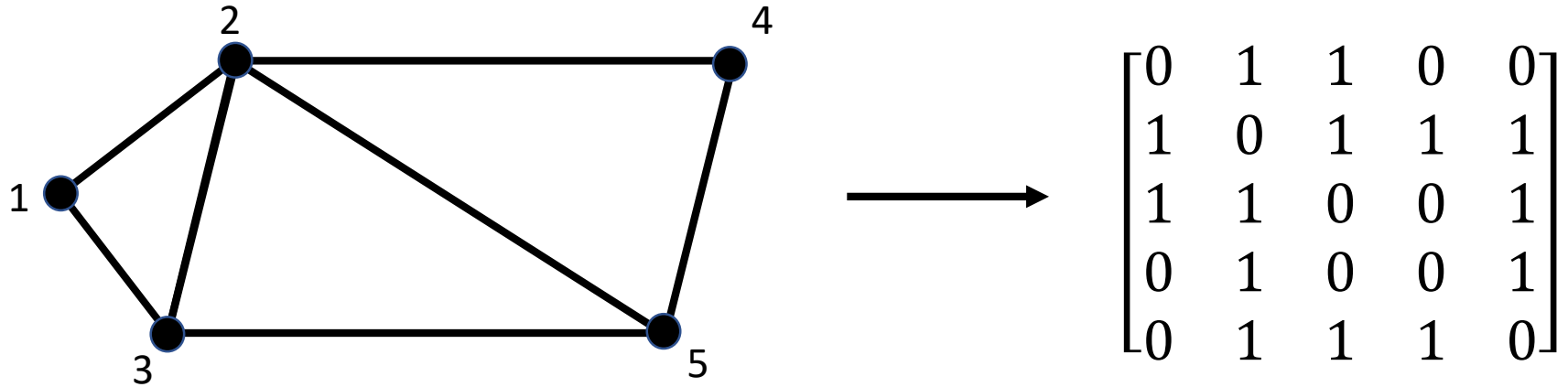
Grafların kullanım alanlarına örnekler

- Otoyol, demir, hava ve deniz yolu ağı
- Bilgisayar Ağları
- Çöp kamyonları yol planlaması
- Elektrik ve elektronik devre çözümleri
- vb.



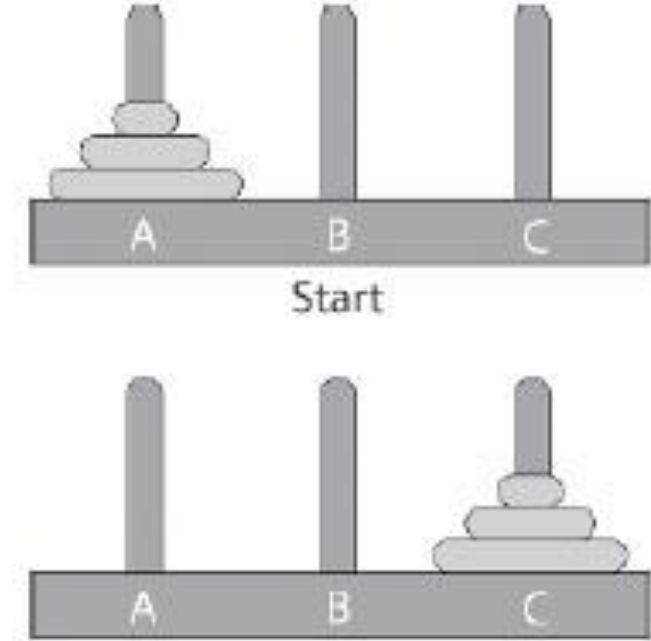
Graflar

- Grafların matrisel gösterimi:



Hanoi kuleleri

- Hanoi'nin kulesinin bu örneğinde, 3 tane çubuk vardır: A,B ve C. Bu çubukları saran 3 farklı büyüklükte halka vardır.
- Bu halkalar başlangıçta A'nın üzerine sırayla yerleştirilmiştir.
- Problemin amacı, C'nin üzerine bütün halkalar yerleştirilene kadar, sıralı olacak şekilde halkaları hareket ettirmektir.
- Bir defada sadece bir halka taşınabilir. Boş bir çubuğa bir halka yerleştirilebilir ya da kendisinden daha büyük olan bir halkanın üstüne yerleştirilebilir.

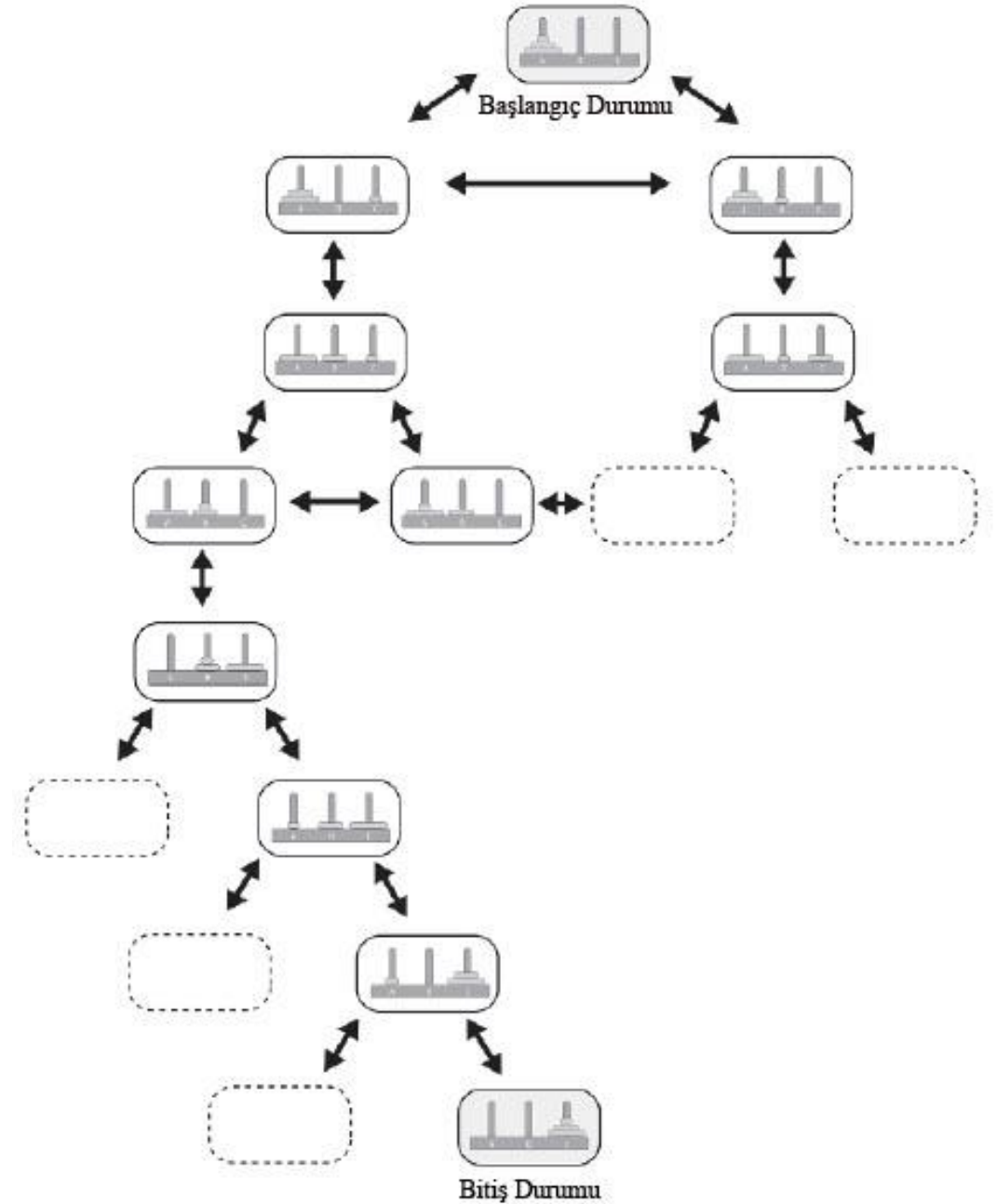


Hanoi kuleleri

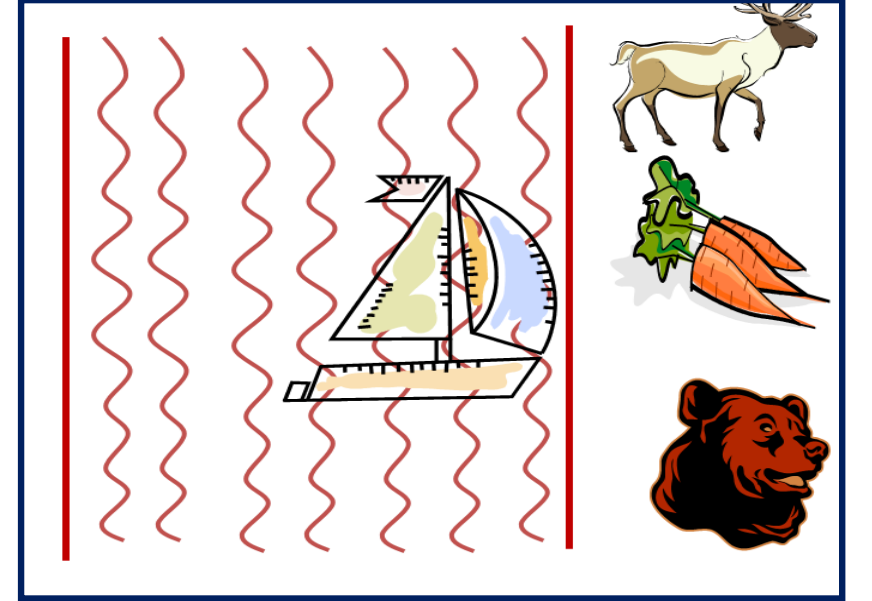
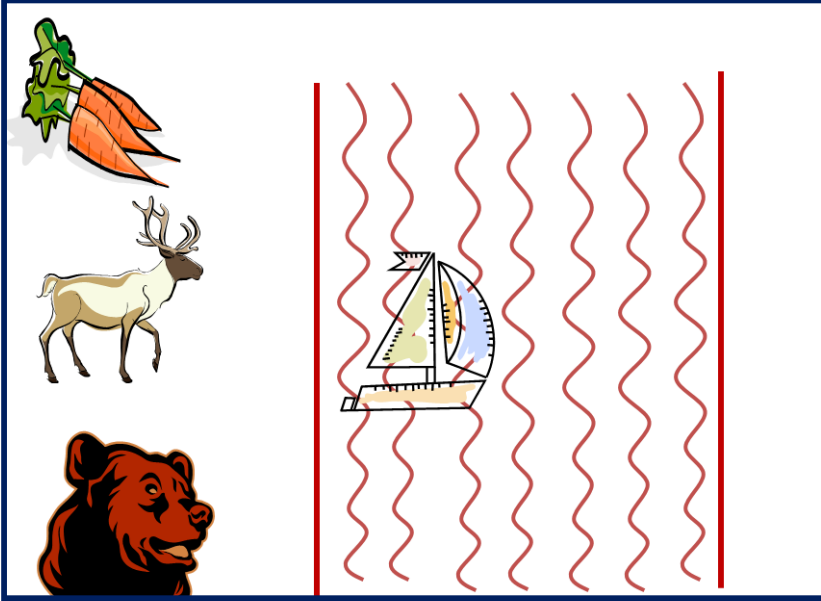
- Problemin işgal ettiği her düğümün mümkün durumlardan birini gösterdiği yerde bir graf kullanarak problemin durum uzayını gösterebiliriz.
- Graf kenarları; durumlar arasındaki geçişleri gösterir. Eğer bir durumdan diğerine direkt olarak geçiş mümkünse, bu düğümleri birbirine bağlayan bir kenar mevcut olacaktır, aksi takdirde böyle bir bağlantı olmayacaktır.
- Problemin başlangıç durumunu içeren bir düğümü başlangıçta oluşturarak, graf meydana getirilir. Bu kök düğüm olarak bilinir. Kök düğüm, graf üzerinde kendisinden erişilebilen bütün düğümlerin eklenmesiyle genişletilebilir ve bütün düğümler ve geçişler grafa eklenmiş olana kadar bu ekleme işlemi devam eder.
- Her durumun bir önceki durumu ebeveyn durum ve oluşturulan yeni durum çocuk durum olarak adlandırılır.

Hanoi kuleleri

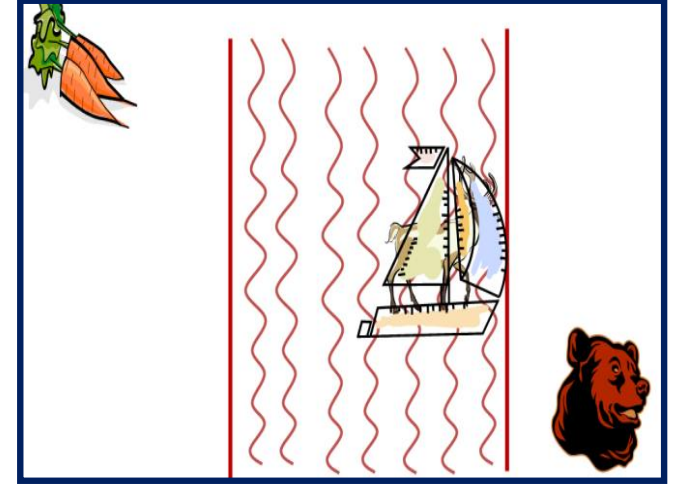
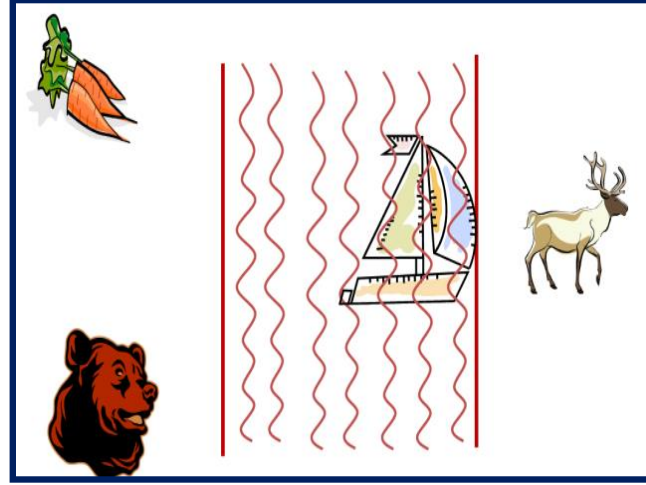
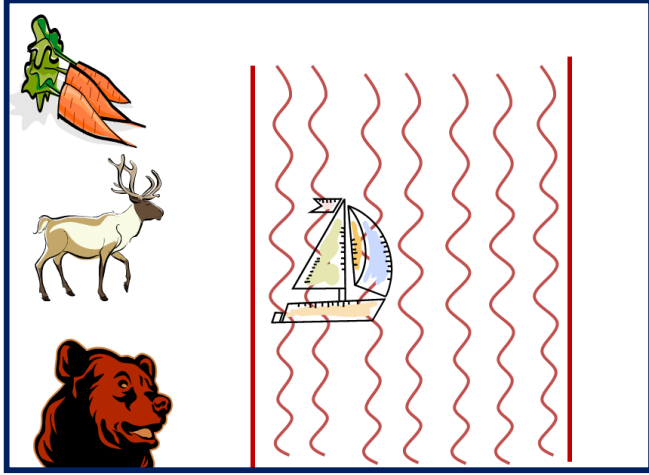
- Her ok, bir durumdan bir diskin kaydırılmasıyla oluşan yeni duruma geçişi gösterir.
- Graf kısa sürede karmaşık hale gelir, bu yüzden çözüme giden yollardan birini görmeyi kolaylaştırmak için birçok muhtemel durumu çıkarabiliriz.
- Hedef durumu bulmak için bir durum grafi kolay bir şekilde aranabilir.
- Bu örnekte hedef durum, C çubuğunun üzerine bütün parçaların doğru sırada yerleştirildiği yerdir.
- Durum uzayı arama ile amaç sadece basit bir çözüm bulmak değil, her mümkün çözümü; ya da en az hareket gerektiren çözümü bulmak olabilir.



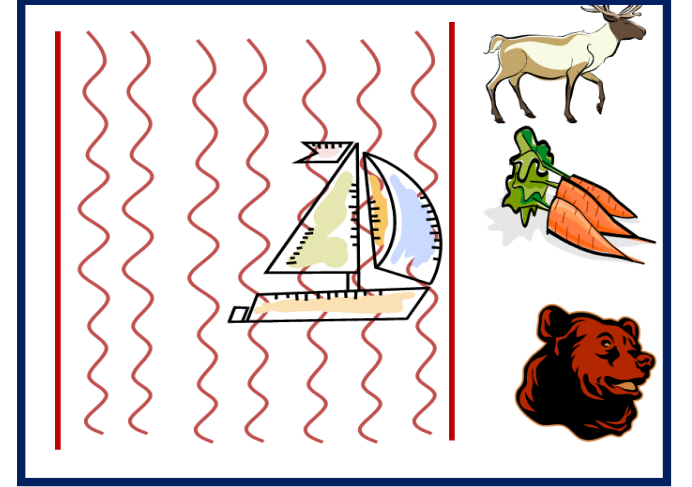
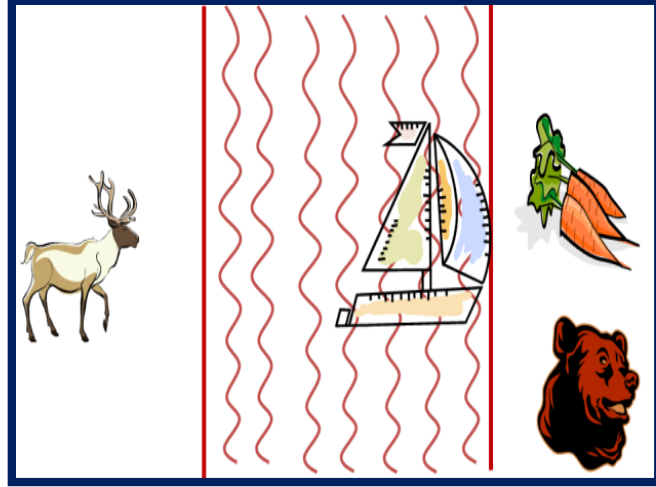
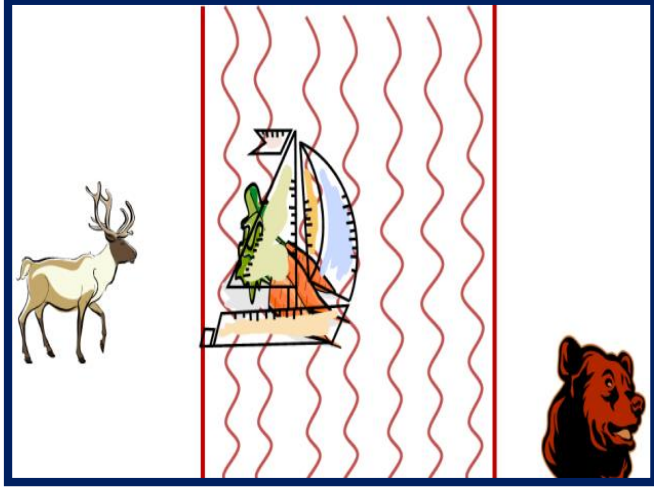
Nehir problemi



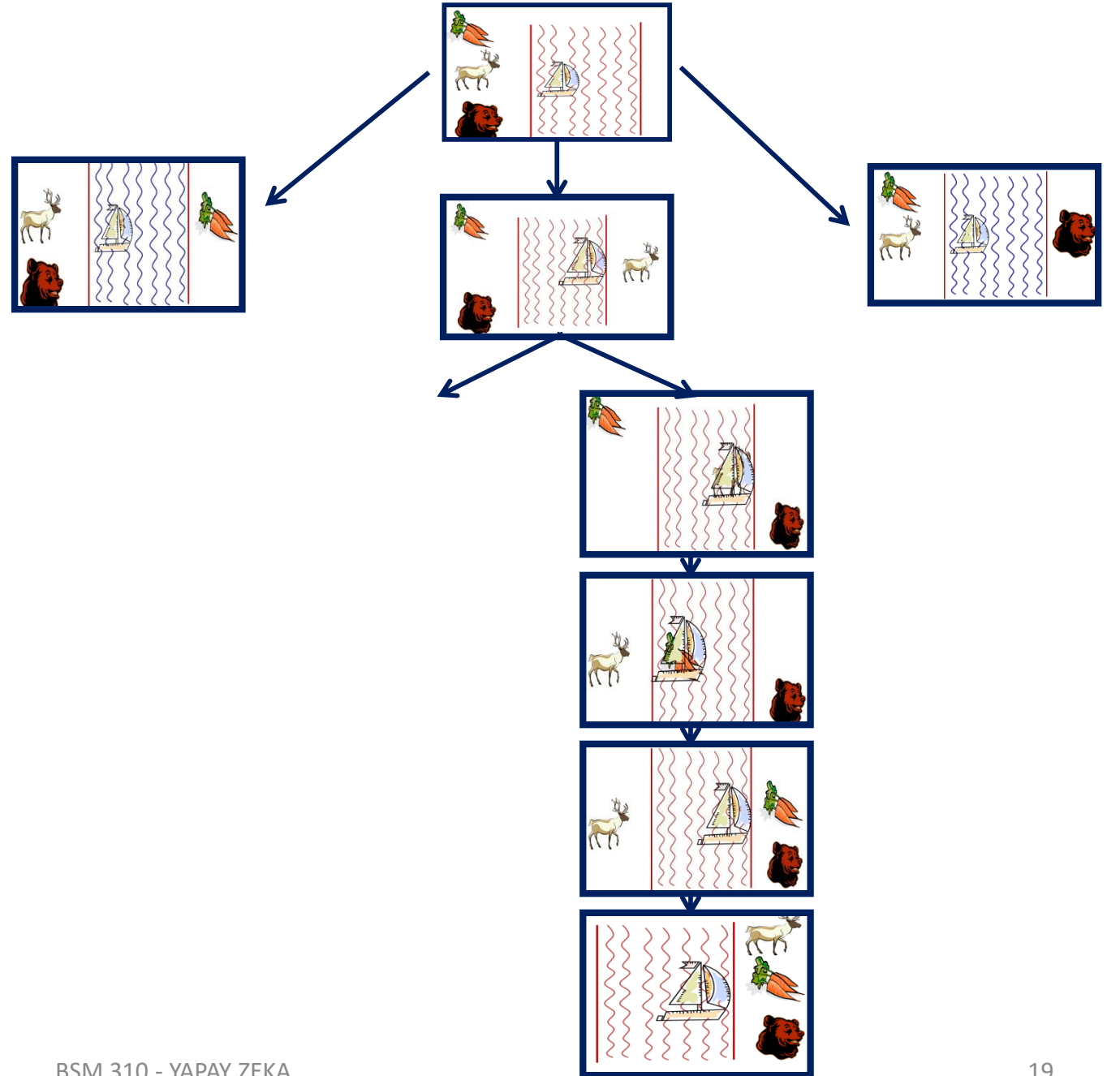
Nehir problemi



Nehir problemi



Nehir problemi



Durum grafları

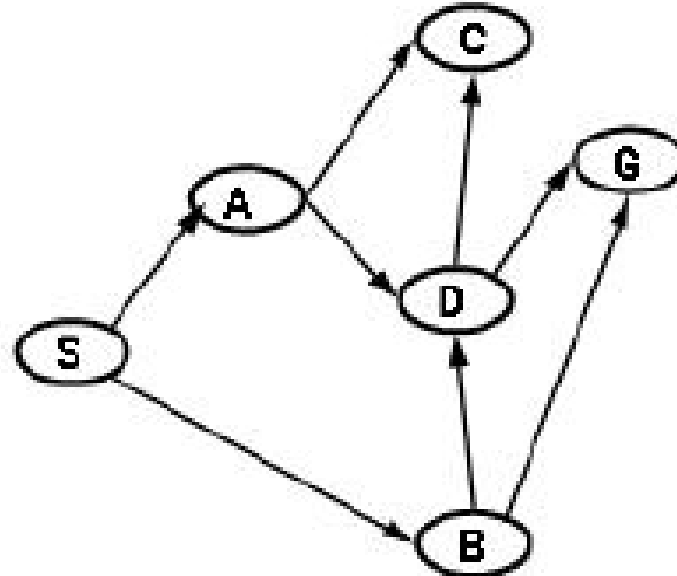
- Bir durum grafi; sistemin içinde bulunabileceği bütün durumların gösterimidir ve bu durumlar arasındaki geçişlerden oluşur.
- Bu sistemin bütün potansiyel durumlarının toplamı, durum uzayı olarak bilinir.
- Bu tip bir graf, özel bir durum mümkünse bunu görmek; ya da belli bir durum için en etkili yönü bulmak için kullanılır.

Durum grafları

- Her ebeveyn düğümünden yayılan ortalama çocuk düğüm sayısı dallanma faktörü olarak bilinir.
- Birkaç problem için burada bahsi geçen örnekler gibi dallanma faktörü düşüktür, düğüm başına üç dal şeklinde, bütün durum uzayını bilgisayarın hafızasında bir grafla göstermeyi mümkün kılar.
- Çoğu alanlar için dallanma faktörü çok fazla ve mevcut potansiyel durum sayısı kök düğümünden uzaklaşıp artarak büyümektedir (graf derinliği).
- Bu tip sistemlerde bütün durum uzayını göstermek mümkün değildir; çünkü bu en güçlü bilgisayarın bile hafıza kapasitesinin çok hızlı bir şekilde aşılmasına neden olabilir.
- Graf sonlandığında bir aramayı tamamlamak oldukça uzun süre alabilir.

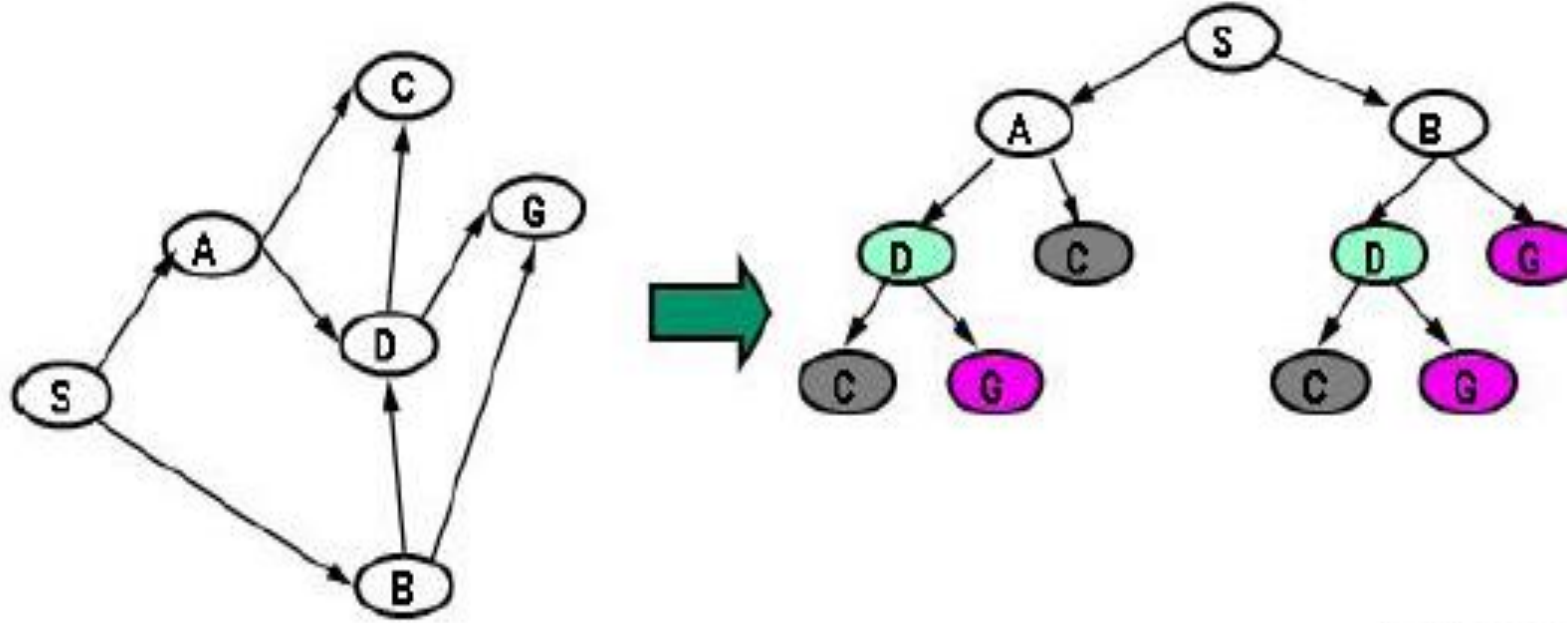
Ağaçlarda arama işlemlerinin graflara uygulanması

- Graf arama problemini ağaç arama problemine dönüştürebiliriz. Bunun için dikkat edilmesi gerekenler:
 - Yönsüz bağlantılar iki yönlü bağlantı şekline döndürülmeli
 - Yolda döngülerden kaçınılmalı



Ağaçlarda arama işlemlerinin graflara uygulanması

- Graf arama problemini ağaç arama problemine dönüştürebiliriz. Bunun için dikkat edilmesi gerekenler:
 - Yönsüz bağlantılar iki yönlü bağlantı şekline döndürülmeli
 - Yolda döngülerden kaçınılmalı



Şifreli hesaplama problemi

- Problemde farklı her harf için farklı bir rakam değeri vardır. Ör:

$$\begin{array}{r} \text{DONALD} \\ + \text{GERALD} \\ \hline \text{ROBERT} \end{array}$$

- Her harfin rakamsal karşılığı olduğu bu problem tek çözüme sahiptir.
- D=5 olduğu bir durumda bu problemi nasıl çözersiniz?
- Bilgisayara nasıl çözdürürsünüz?

Şifreli hesaplama problemi

$$\begin{array}{r} \text{DONALD} \\ + \text{GERALD} \\ \hline \text{ROBERT} \end{array}$$

- Durum uzayını tamamen incelemeye gerek yoktur, ipucundan yola çıkılarak çözüme ulaşılır.
- Kesin değerlerden yararlanarak olası değerler belirtilir, seçilen bir değer kurallara uymaz ise geriye kalan olası değerler değerlendirilir.

Şifreli hesaplama problemi

$$\begin{array}{r} \text{DONALD} \\ + \text{GERALD} \\ \hline \text{ROBERT} \end{array}$$

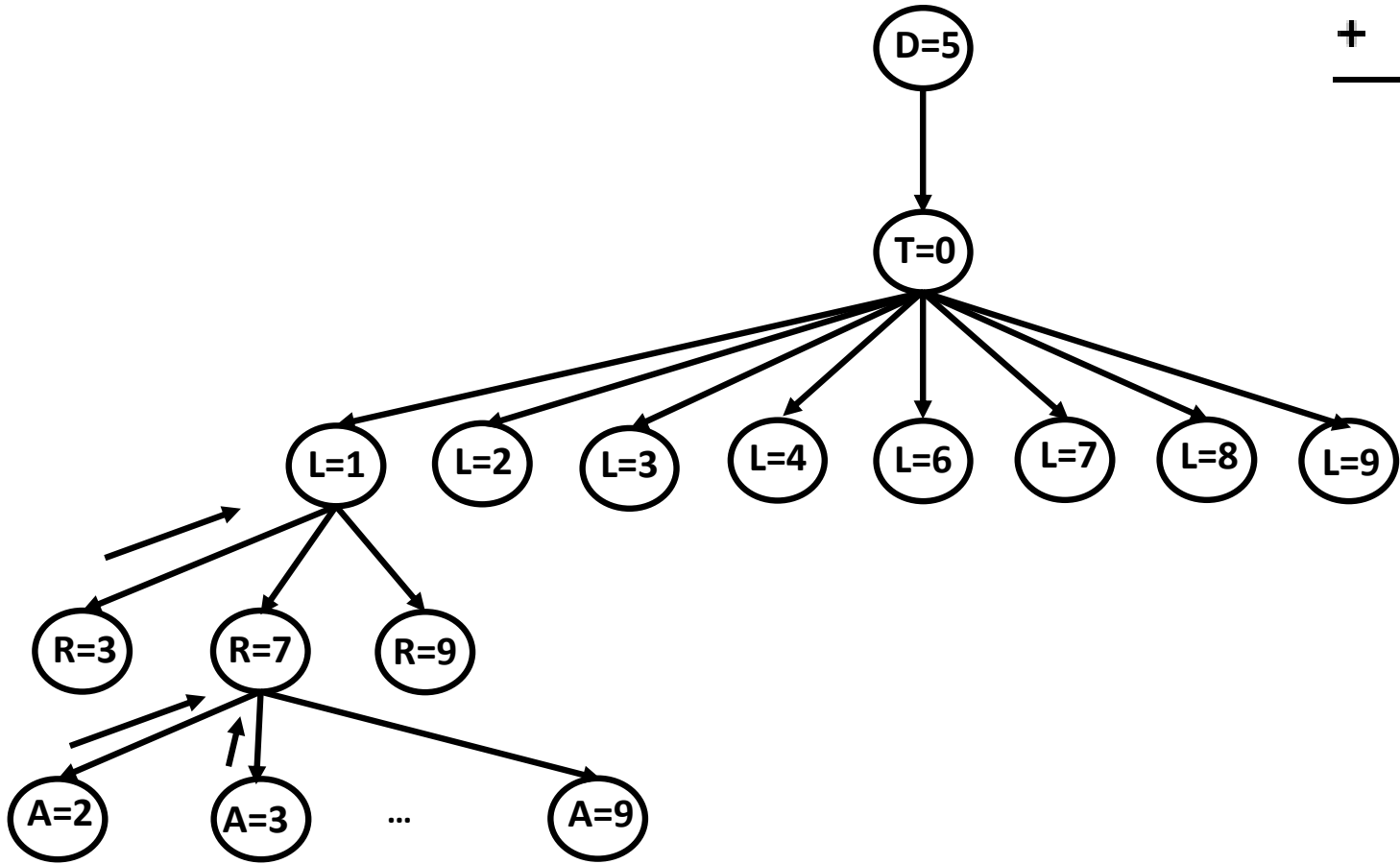
- Problem bilgisayar yardımı ile de benzer şekilde çözülür.
- Olasılıkların yer aldığı çözüm ağacı oluşturulur.
- $D=5 \rightarrow T=0$
- $L+L$ mutlaka çifttir. $L+L+1$ tek olur. Öyle ise R tek ve 5'ten farklıdır. ($R=1/3/7/9$)
- $D+G=R \rightarrow R>5$ olmalı
- $R=7$ olsa $L = 3 / 8$ olabilir...
- ...
- $D=5, T=0, L=8, R=7, A=4, E=9, N=6, O=2, G=1, B=3$

Şifreli hesaplama problemi

DONALD

+ GERALD

ROBERT

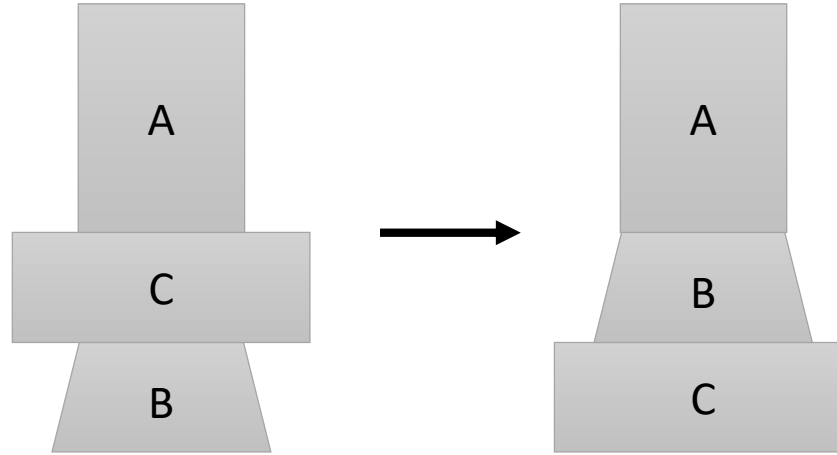


Durum uzayı

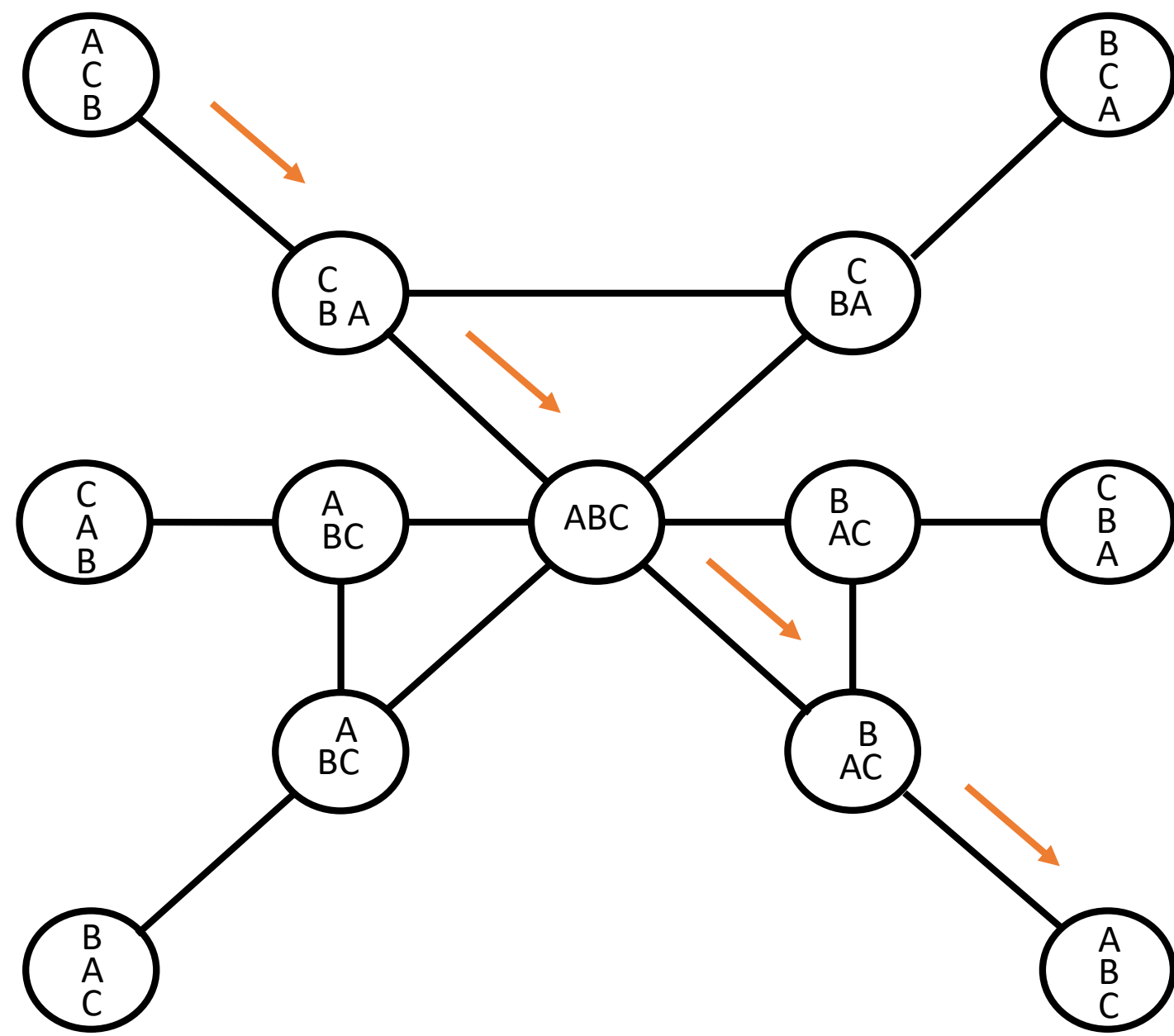
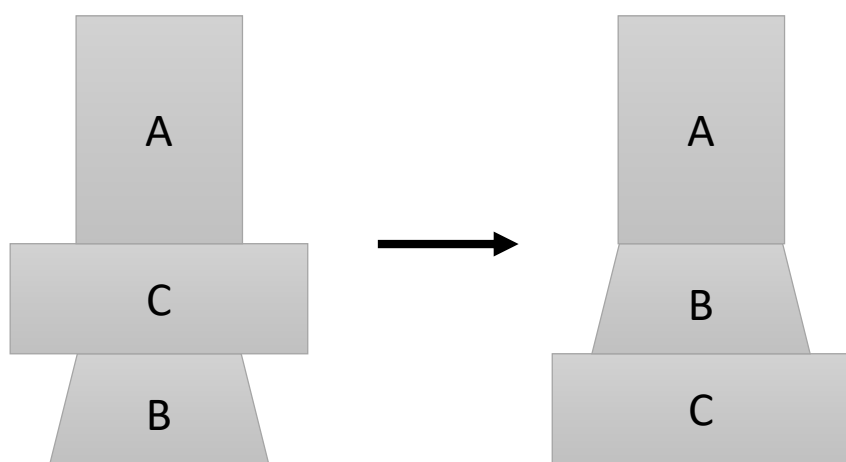
- Bir yapay zeka problemi için izinli eylemleri barındıran sonlu durumlar kümesi mevcuttur.
- Kümedeki ilişkili elemanlar graf olarak ifade edilir.
- İzinli tüm durumları içeren bu graf problemin durum uzayıdır.
- Problemin çözümü ilgili graftaki minimum yolun bulunması ile sağlanır.

Durum uzayı - Örnek

- A, B ve C ürünleri hareketli bir bant üzerindedir.
- Bir robot kolu bu ürünleri sıralayacaktır.
- Robot kolu ürünü bandın ya da diğer bir ürünün üzerine koyabilir.
- Başlangıç şartı değişkendir, graf yönsüzdür.



Durum uzayı - Örnek

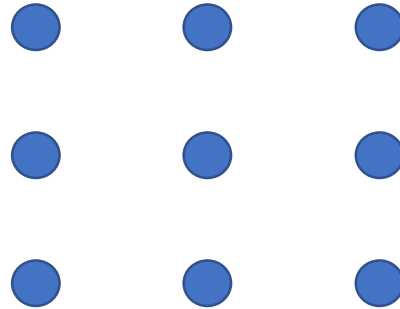


Durum uzayı

- Herhangi bir problem için durum uzayının sınırları kesin ve açık bir şekilde tespit edilmelidir.
- Tam olarak anlaşılamayan girdiler, insan beyni için gereksiz ilişkiler kurulmasına ve gereksiz kısıtlamalar yapılmasına sebep olur.
- Çünkü insan beyni "otomatik bir minimizasyon mekanizması" içerir.

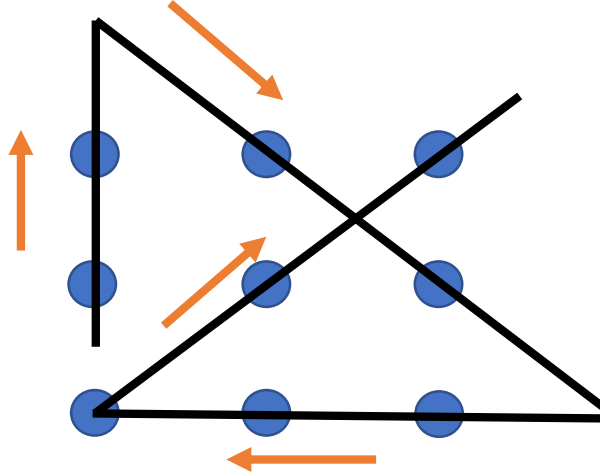
Durum uzayı – Örnek 2

- Problemde dikey ve yatay olmak üzere her sırada 3 tane olmak üzere toplamda 9 nokta vardır.
- Bu 9 nokta, kalemi kaldırmadan 4 doğru parçası ile nasıl birleştirilir.



Durum uzayı – Örnek 2

- Problemde dikey ve yatay olmak üzere her sırada 3 tane olmak üzere toplamda 9 nokta vardır.
- Bu 9 nokta, kalemi kaldırmadan 4 doğru parçası ile nasıl birleştirilir.



Durum uzayı

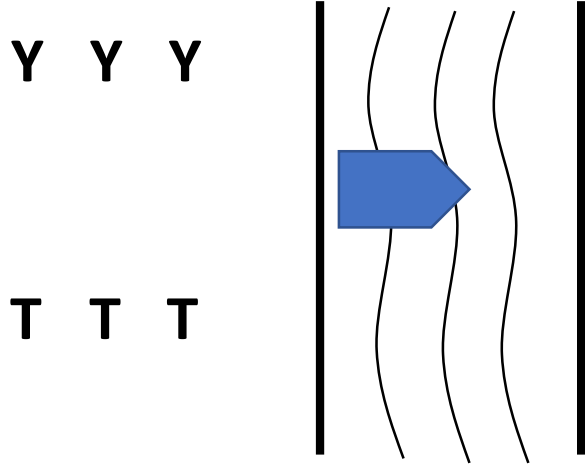
- Yapay zeka yöntemlerini bilmeyen bir programcı problemleri çözerken deneme/yanılma yöntemi ile çözüme gitmeye çalışabilir.
- Bu yaklaşım durum uzayı geniş olmayan problemler için çalışabilir.
- Durum uzayı büyüdükçe bilgisayar kaynakları yetersiz kalabilmektedir.
- Örneğin 15 taş oyununun durum uzayı 21.000.000.000.000 durum içerir.
- Bu durumda durum uzayı sistemli bir şekilde aranmalıdır.
- Bunun için öncelikle operatör kavramını bilmek gerekecektir.
- Durumlar arasında geçiş yapabilen yapılara operatör denir.

Durum uzayı

- 15 taş oyunun basit versiyonu olan 8 taş probleminde amaç başlangıç durumundan hedefe ulaşmak için boşluğun yapması gereken en kısa hareket dizisini bulmaktır.
- Burada 4 operatör söz konusudur.
 - Eğer üst sınıra ulaşılmadı ise boşluğu yukarı kaydır
 - Eğer alt sınıra ulaşılmadı ise boşluğu aşağı kaydır
 - Eğer sağ sınıra ulaşılmadı ise boşluğu sağa kaydır
 - Eğer sol sınıra ulaşılmadı ise boşluğu sola kaydır

9	5	3
1	7	8
2	4	

Durum uzayı – Turistler ve yamyamlar problemi



- Tekne yalnız iki kişi alabiliyor.
- Teknenin hareket edebilmesi için kayıkta en az bir kişi olmalıdır.
- İki kıyıda da turist sayısı yamyam sayısından az olmamalıdır.
- Toplam sayılar korunarak iki grup da karşıya taşınmalıdır.

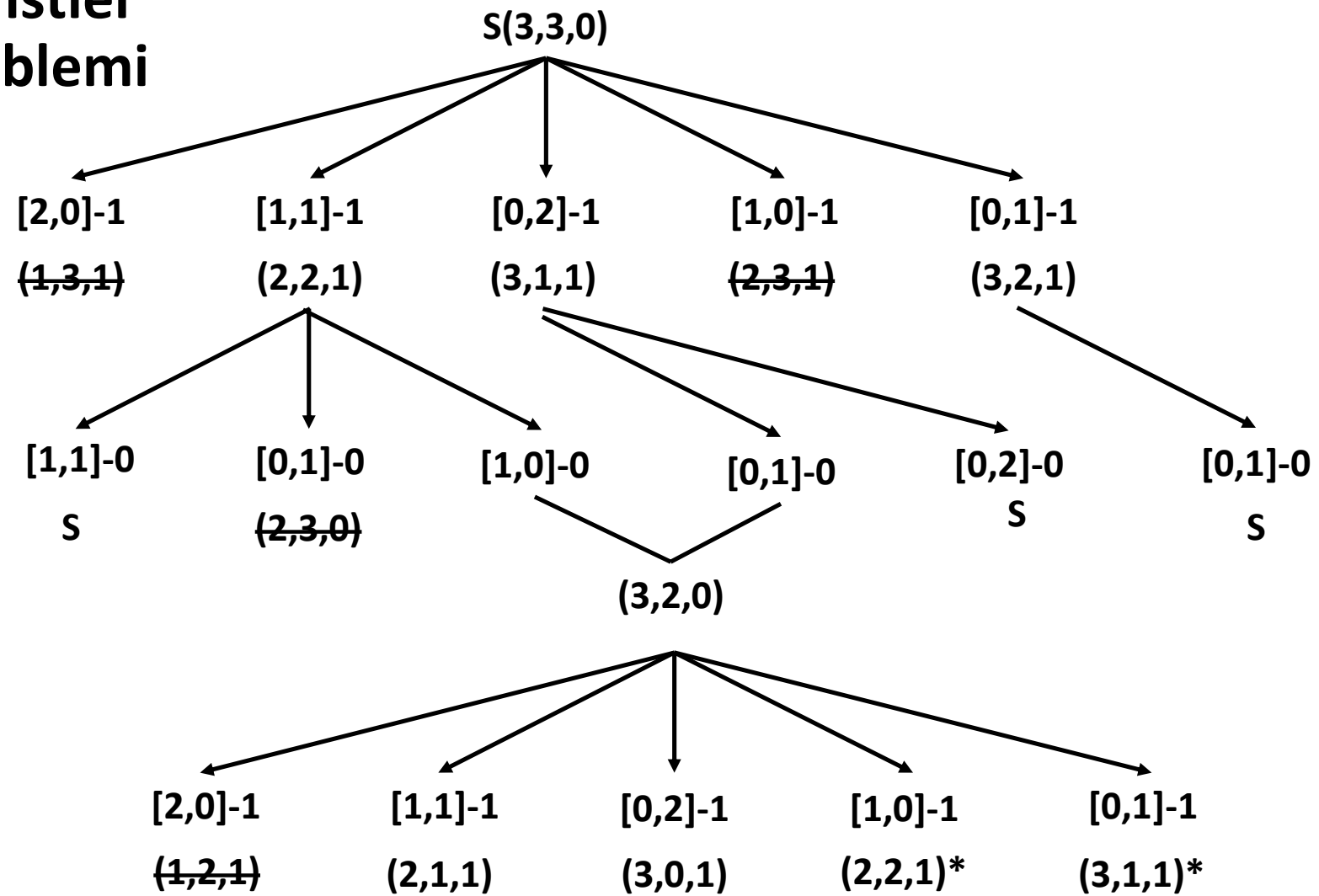
Durum uzayı – Turistler ve yamyamlar problemi

- (x,y,z)
 - x : sol kıyıdaki turist sayısı
 - y : sol kıyıdaki yamyam sayısı
 - z : teknenin sol (0) ya da sağ (1) kıyıda olması durumu
- Başlangıç durumu: $(3,3,0)$
- Hedef durum: $(0,0,1)$
- Operatör: $[u,v]-w$
 - u : teknedeki turist sayısı
 - v : teknedeki yamyam sayısı
 - w : *teknenin hareket yönü (sağ: 1, sol:0)*

Durum uzayı – Turistler ve yamyamlar problemi

- Problemin çözümü için farklı ifadelendirmeler de yapılabilirdi
- Durum uzayında çatışmalı bir duruma rastlanırsa geriye dönülür.
- Ebeveyn (tekrarlanan) bir durumla karşılaşır ise geriye dönülür.
- Problemin çözümüne ait ağacın birkaç seviye derinlikli hali aşağıdaki gibidir.

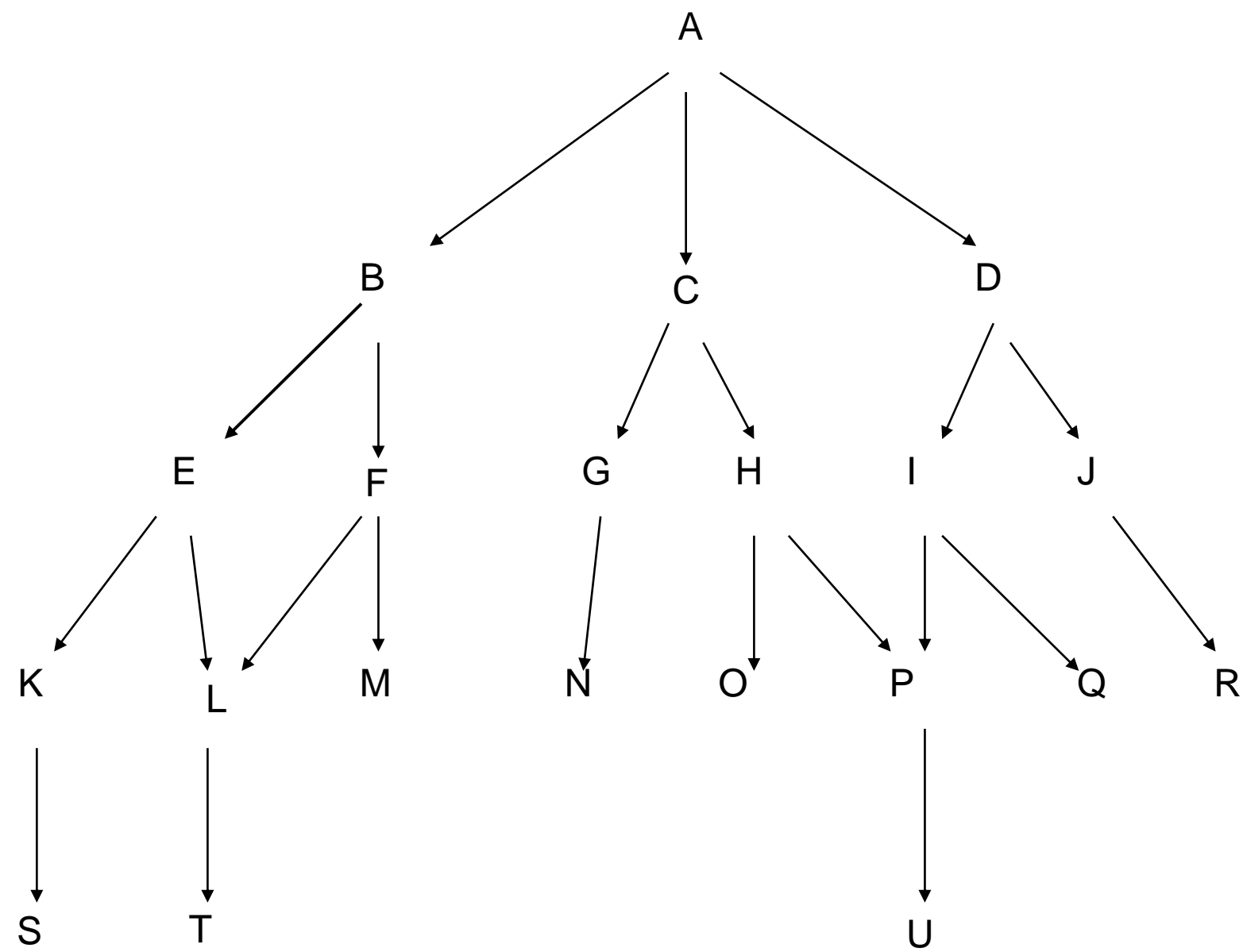
Durum uzayı – Turistler ve yamyamlar problemi



Depth-first search - Derinlik öncelikli arama

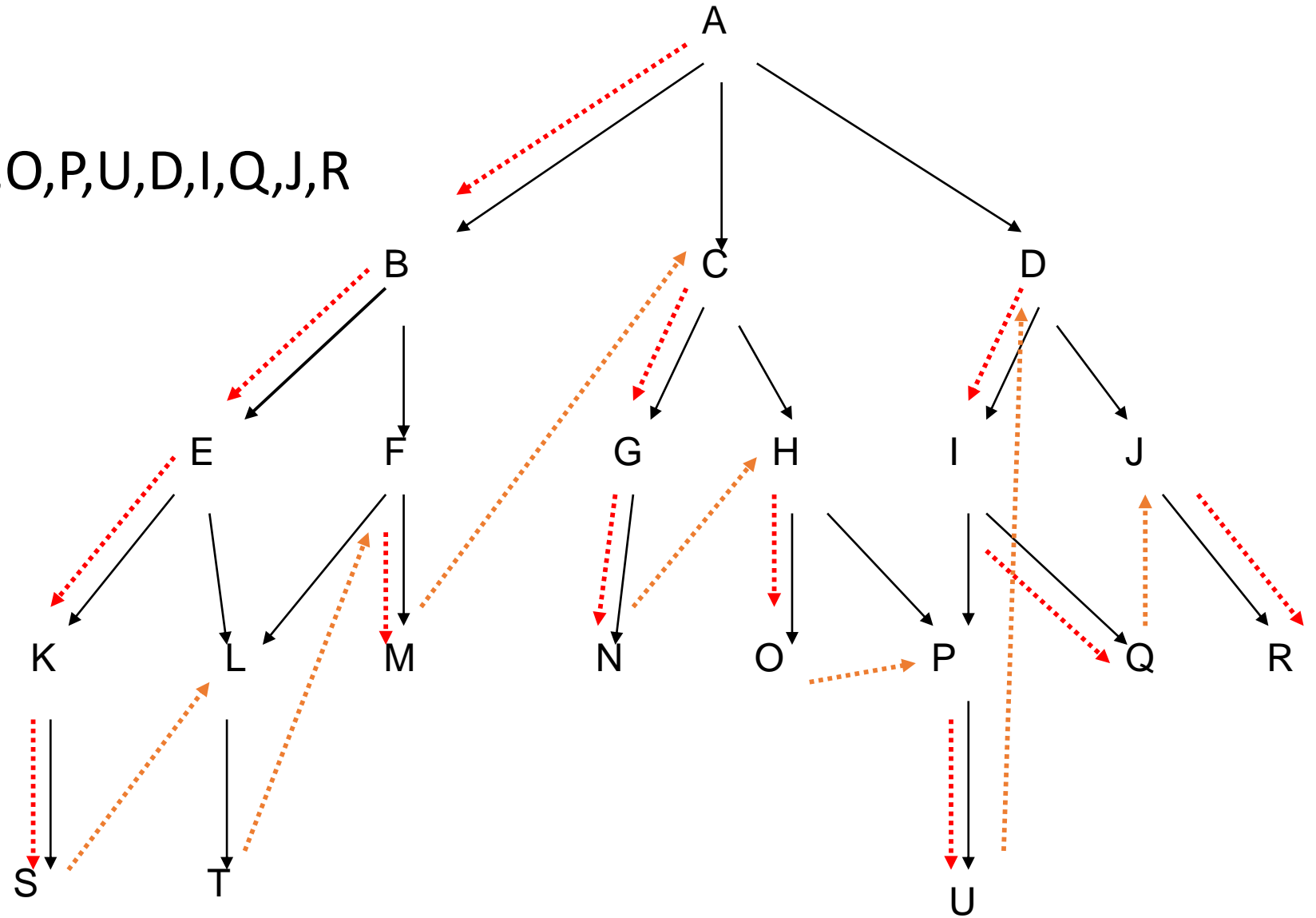
- Derinlik Öncelikli Aramada, bir durum(düğüm) incelendiğinde, yan duruma geçmeden önce incelenen düğümün en uç düğümüne kadar inilir.
- Derinine Öncelikli Arama, mümkün olan en derine kadar gider.
- Artık gidecek alt düğüm olmadığında, yan düğüme dallanılır.

Depth-first search

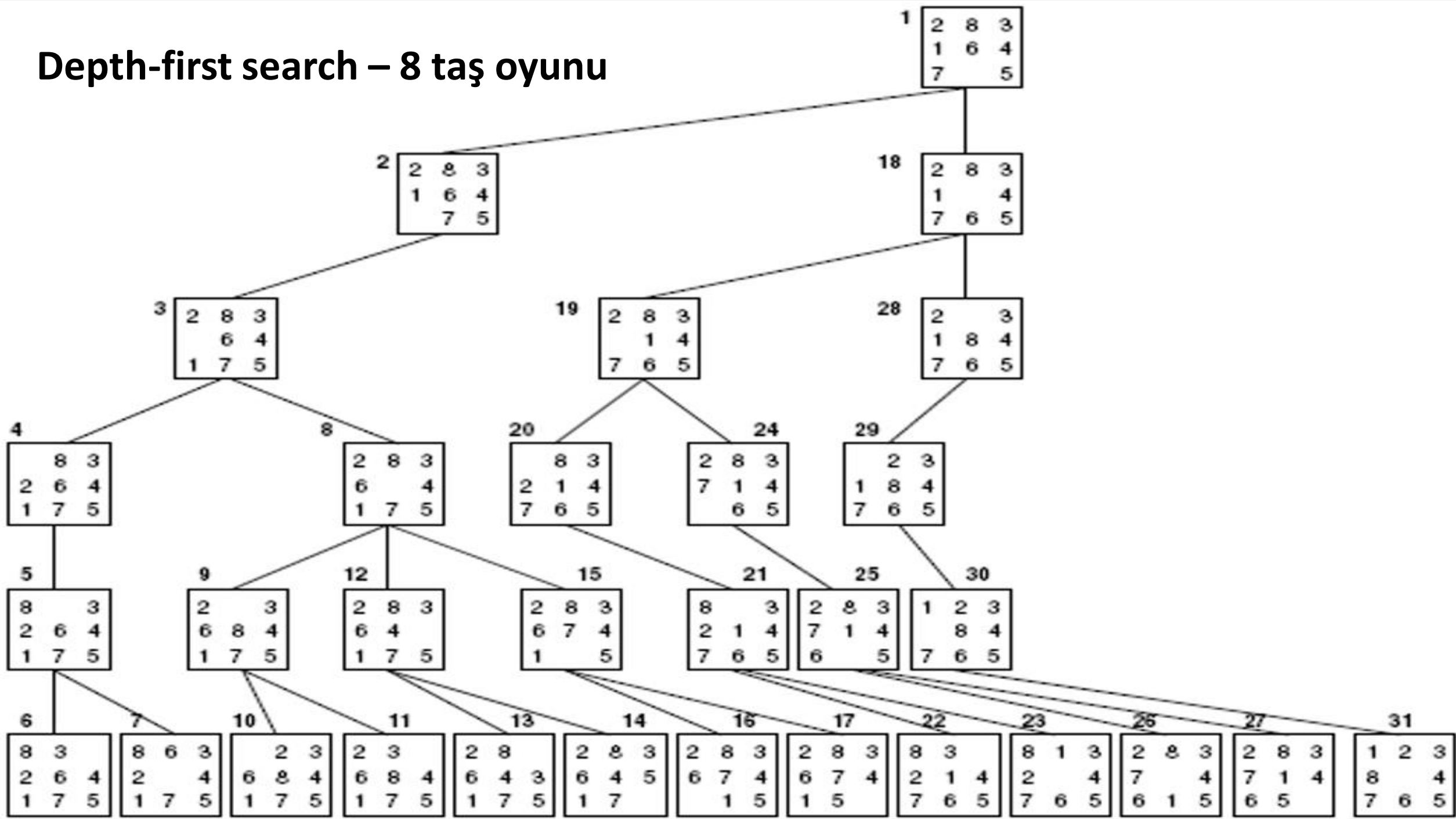


Depth-first search

A,B,E,K,S,L,T,F,M,C,G,N,H,O,P,U,D,I,Q,J,R



Depth-first search – 8 taş oyunu



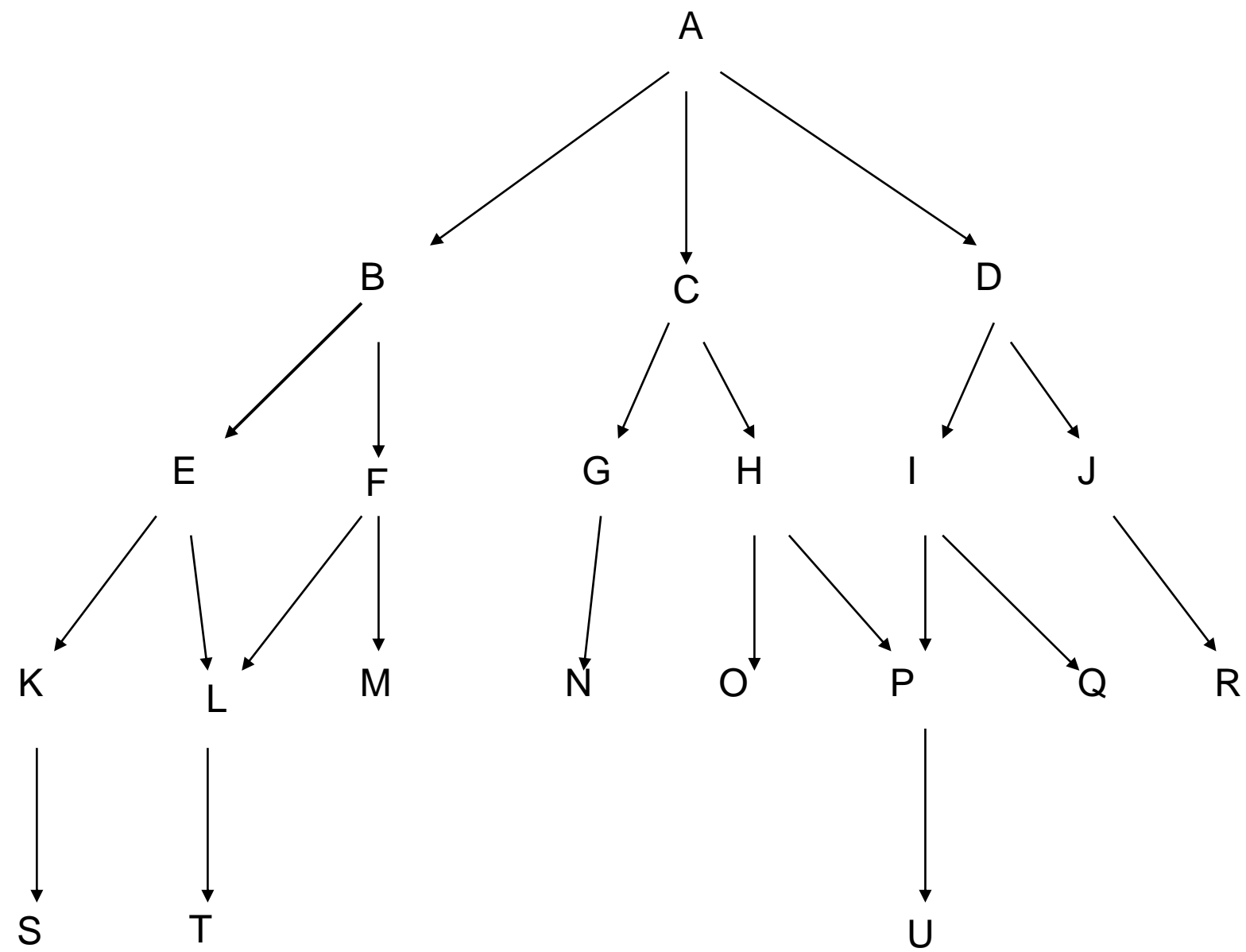
Depth-first search – algoritma karmaşıklığı

- b : ağaçtaki düğümlerin çocuk sayısı
- k : ağaçtaki maksimum derinlik
 - Depolama karmaşıklığı: $O(bk)$
 - Zaman karmaşıklığı: $O(b^k)$

Breadth-first search - genişlik öncelikli arama

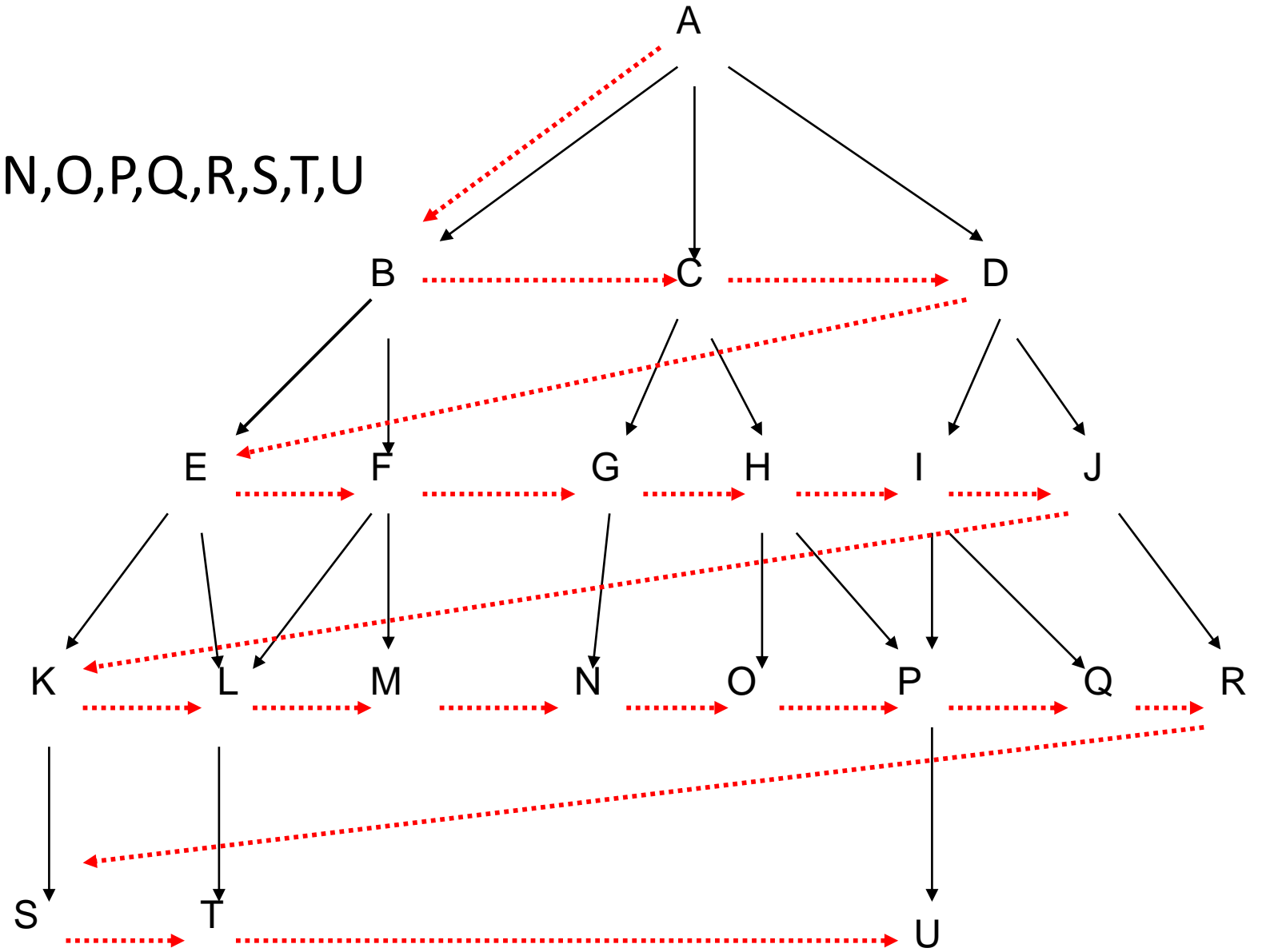
- Genişlik Öncelikli Arama, mümkün olan ilgili seviyedeki tüm düğümleri dolaşır.
- Artık gidecek düğüm olmadığında, alt düğüme dallanılır.

Breadth-first search

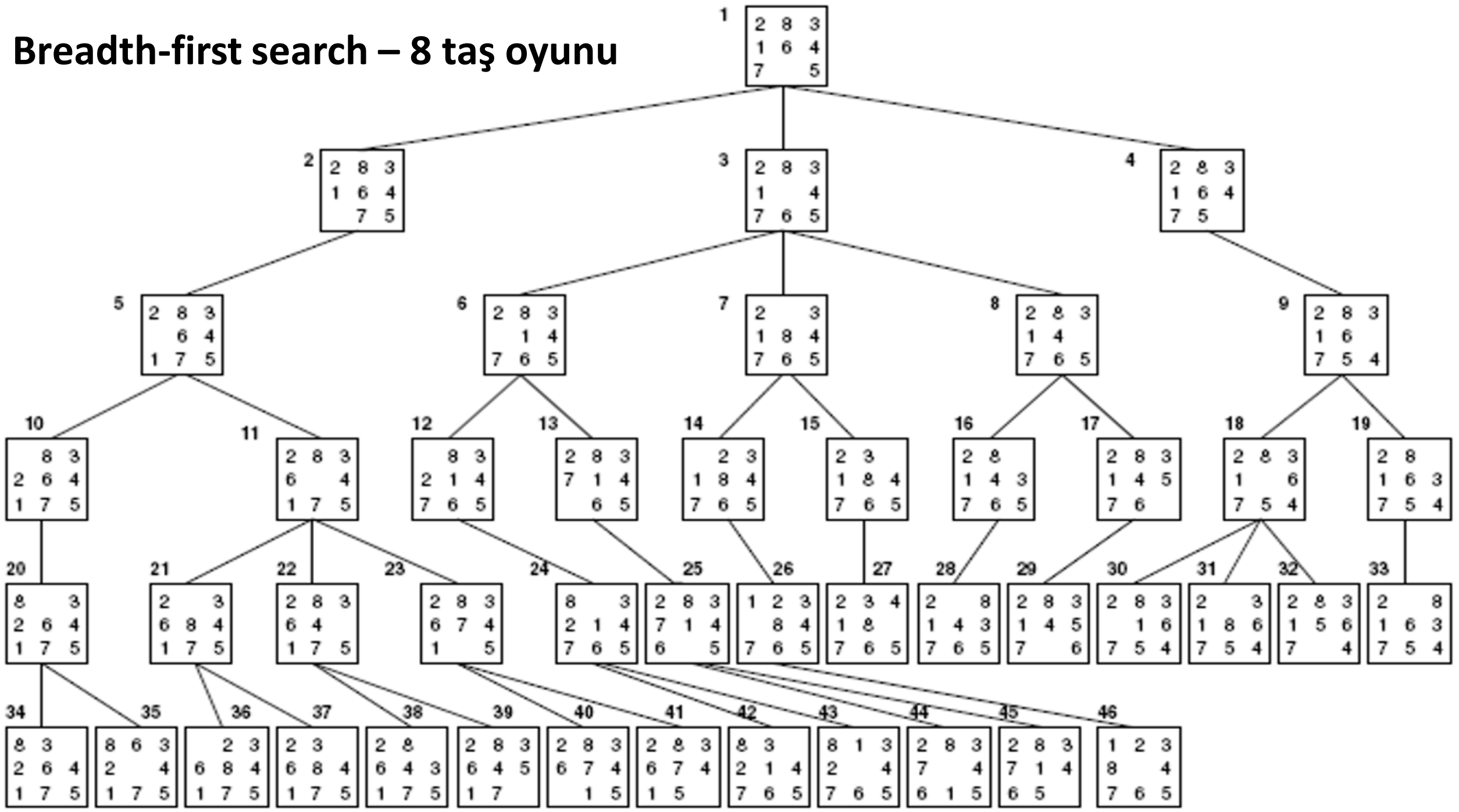


Breadth-first search

- A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U



Breadth-first search – 8 taş oyunu



Breadth-first search – algoritma karmaşıklığı

- b : ağaçtaki düğümlerin çocuk sayısı
- d : ağaçtaki maksimum derinlik

- Depolama karmaşıklığı: $O(b^d)$
- Zaman karmaşıklığı: $O(b^d)$

Derinlik	Toplam düğüm sayısı	Zaman	Bellek
0	1	1 milisaniye	100 byte
2	111	0.1 saniye	11 kb
10	10^{10}	128 gün	1 T
14	10^{14}	3500 yıl	11.111 T

- *Saniyede 1000 düğüm*
- *Her düğüm 100 byte*
- *Dal etmeni (b) = 10*
- *Toplam düğüm sayısı = $1 + b^1 + b^2 + \dots + b^k$ k : derinlik seviyesi*

Breadth-first search

- Enine arama her zaman köke en yakın çözümü verir.
- Mümkün olan tüm yolların bulunması için tüm ağacın taranması gerekir.
- Problem çözümü daha derin seviyelerde ise bu yöntemin etkinliği azalmaktadır.
- Doğal dil işleme, bulmacalar, oyunlar, görüntü yorumlama, vb. yapay zeka problemlerinde kullanılabilmektedir.

A* algoritması – değer fonksiyonu

- Sezgisel algoritmalarda değer fonksiyonunun belirlenmesi büyük önem taşımaktadır.
- Değer fonksiyonu, durum uzayında hedefe giden minimum yolu bulmaya (daha az tarama yapabilmek için) yardımcı olacak bir nitelikte olmalıdır.
- Değer fonksiyonlarının belirlenmesinde çeşitli yöntemler vardır, ör:
 - Düğümün çözümün içinde olmasının olasılık değerlendirilmesi
 - Verilen düğümden hedefe olan uzaklık

A* algoritması – değer fonksiyonu

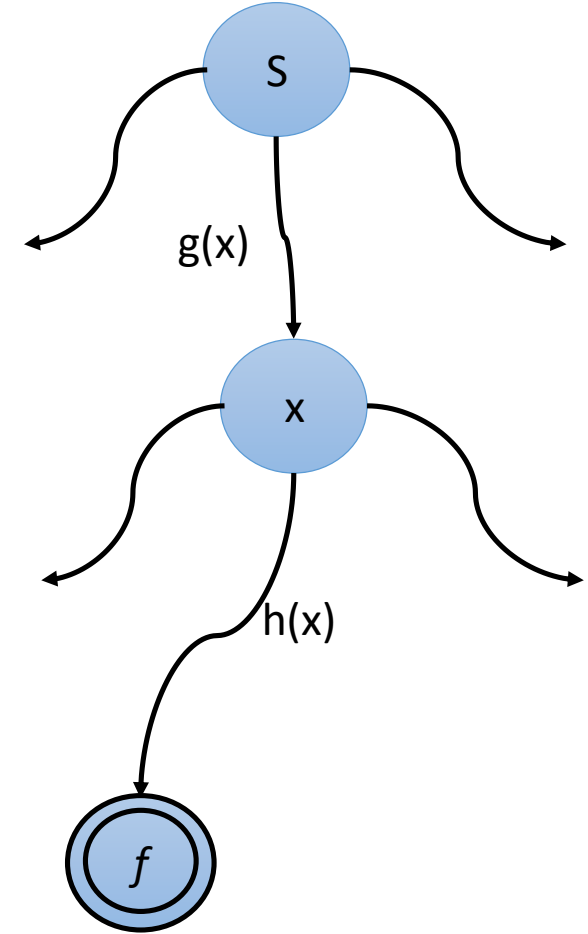
- Durum uzayında çözüme giden düğümlerin seçilmesini sağlayan bir f fonksiyonu olduğunu düşünelim.
- Herhangi bir x düğümü için fonksiyonun değeri $f(x)$ olsun.
- Bu fonksiyon çözüme giden yollar içinde minimum olanının bulunmasına yardımcı olmaktadır.
- Açılma sırası gelen düğümler f fonksiyonunun artışına göre sıralanırlar, bu da sıralı arama algoritmaları olarak bilinmektedir.

A* algoritması - sıralı arama algoritması

- Sıralı arama algoritması:
 - Başlangıç düğümü ağacın kökü olsun ve operatörler bu durumdan mümkün olan çocuk düğümleri elde etsin.
 - Elde edilen çocuk düğümlerde f fonksiyonu sayesinde minimum değere sahip düğüm bulunur (eşit değerli durumlarda herhangi biri seçilebilir).
 - Açılan yeni düğümün çocukları da benzer şekilde f fonksiyonu tarafından değerlendirilir. Eğer mevcut düğümden çocuk düğüm elde edilemiyorsa (örneğin daha önce açılmış bir durum denk geliyorsa) ve henüz hedefe ulaşamamışsa bir üst seviyeye geri dönülerek sıradaki minimum değerli düğüm üzerinden devam edilir.
 - İşlemler hedef düğüm bulununcaya kadar devam eder, bulununca başlangıç ve hedef arası yol çözümü verir.

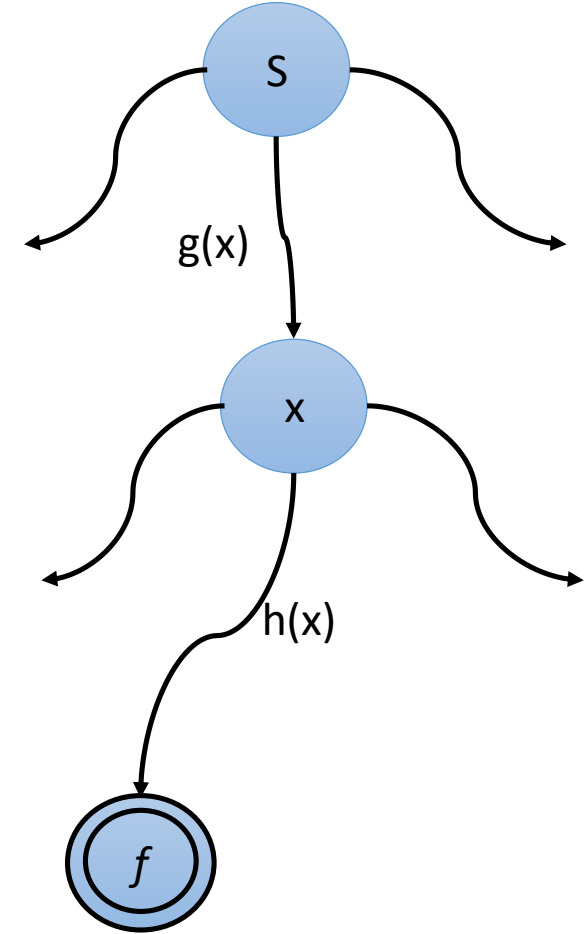
A* algoritması - değer fonksiyonu

- $f(x) = g(x) + h(x)$
- $g(x)$: başlangıç durumundan bulunulan duruma kadar olan bileşen
- $h(x)$: bulunulan durumdan hedefe kadar tahmin edilen bileşen
- Burada $h(x)$ değeri bilinmemekte ve yerine sezgisel $h'(x)$ kullanılmaktadır.



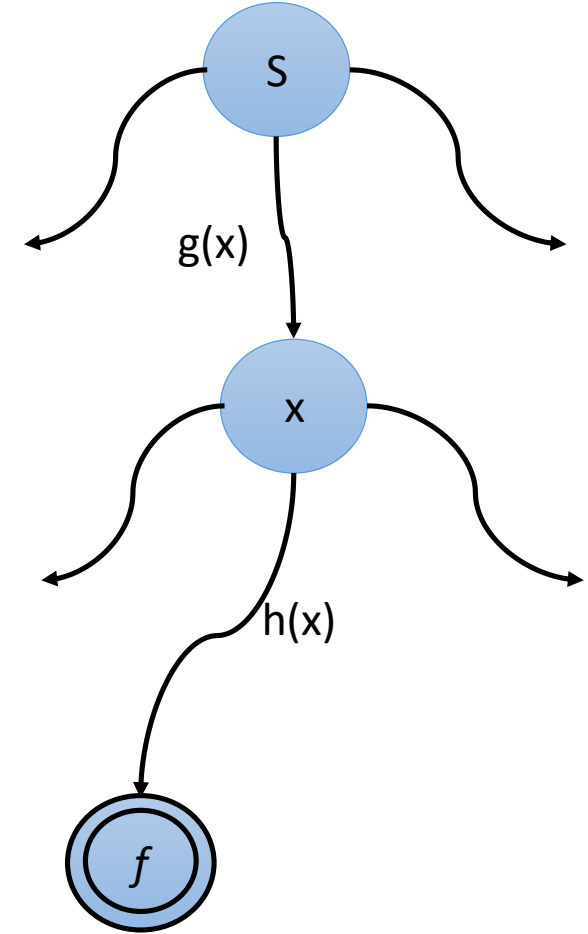
A* algoritması - değer fonksiyonu

- $h'(x)$ fonksiyon tasarımı probleme bağlı olarak yapılır.
- $h'(x)$ hedeften uzaklaştıkça büyük, hedefe yaklaştıkça küçük değerler almalıdır.
- $h'(x)$ fonksiyonunun temel görevi hedefe hızlı bir şekilde ulaşmayı sağlamaktır.
- $h'(x)$ çok iyi seçilmez ise de hedefe ulaşılır fakat daha fazla tarama yapılır.



A* algoritması - değer fonksiyonu

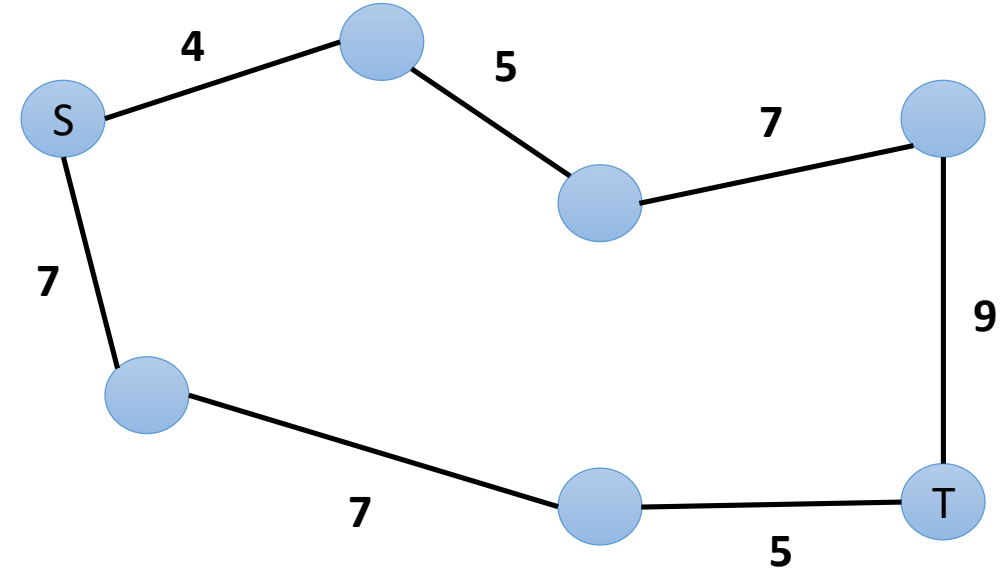
- Fonksiyonun son hali aşağıdaki şekilde tanımlanmaktadır.
- $f(x) = \alpha g(x) + (1 - \alpha) h(x)$, $0 \leq \alpha \leq 1$
- α 'nın büyük değerleri için daha çok enine, küçük değerleri için daha çok derinine arama yapılır.
- α değeri problemin belirli aşamalarında aramayı yönlendirmek için kullanılabilir.



A* algoritması – örnek

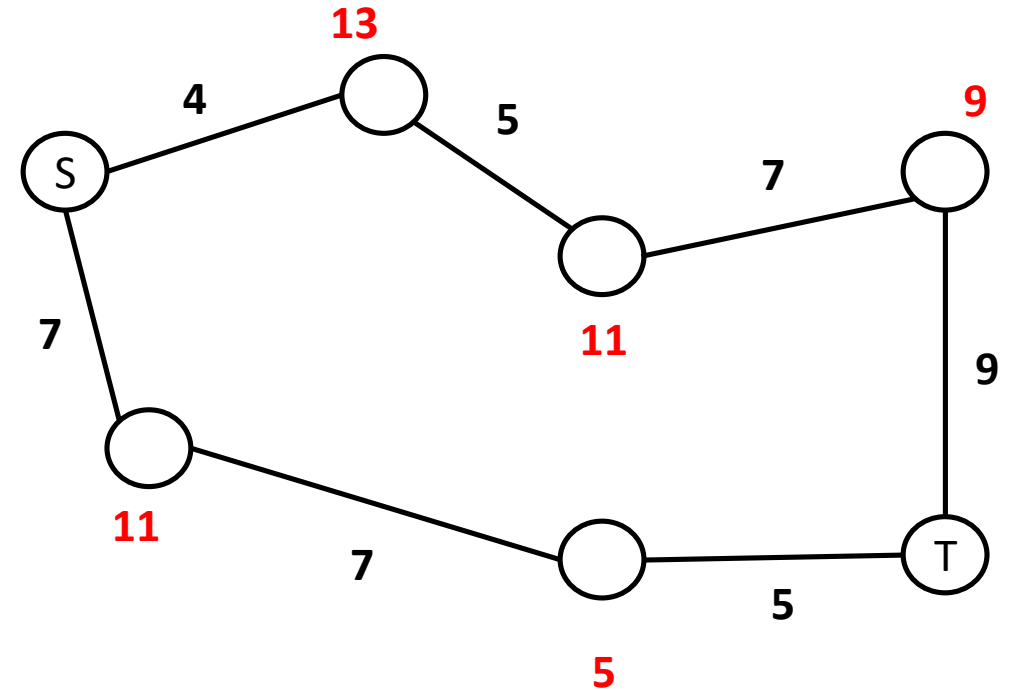
- S: başlangıç düğümü
- T: hedef düğüm
- Maliyeti en düşük yol?

- $f(x) = g(x) + h(x)$
- $g(x)$: mesafe değeri



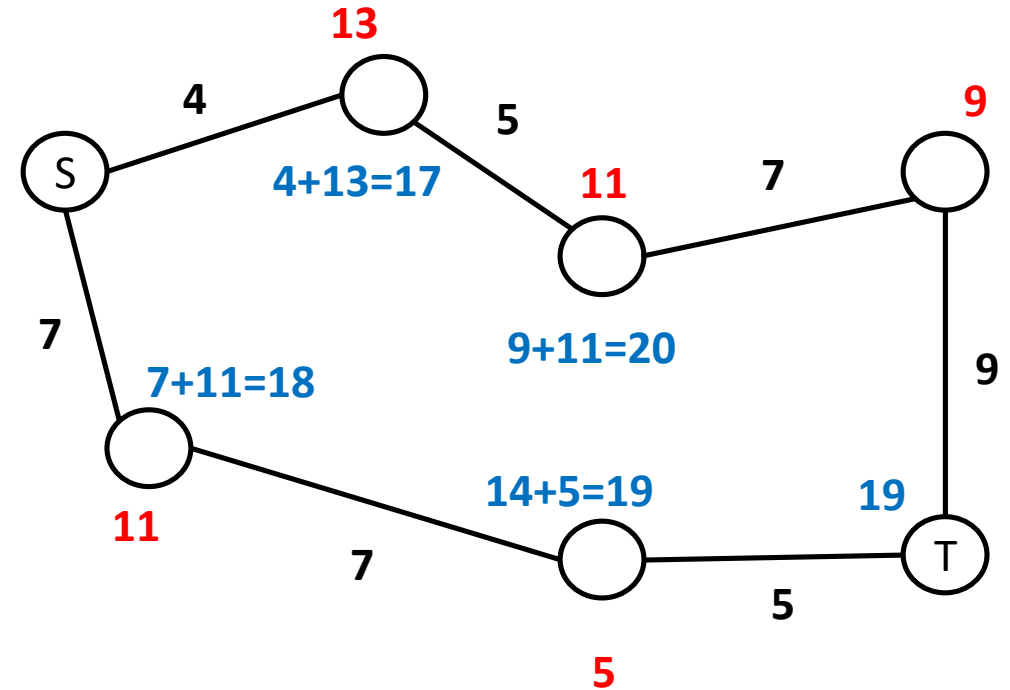
A* algoritması – örnek

- $f(x) = g(x) + h'(x)$
- $g(x)$: mesafe değeri
- $h'(x)$: kuş uçuşu mesafe



A* algoritması – örnek

- $f(x) = g(x) + h(x)$
- $g(x)$: mesafe değeri
- $h'(x)$: kuş uçuşu mesafe



A* algoritması – 8 taş oyunu

4	1	3
2	8	5
7		6

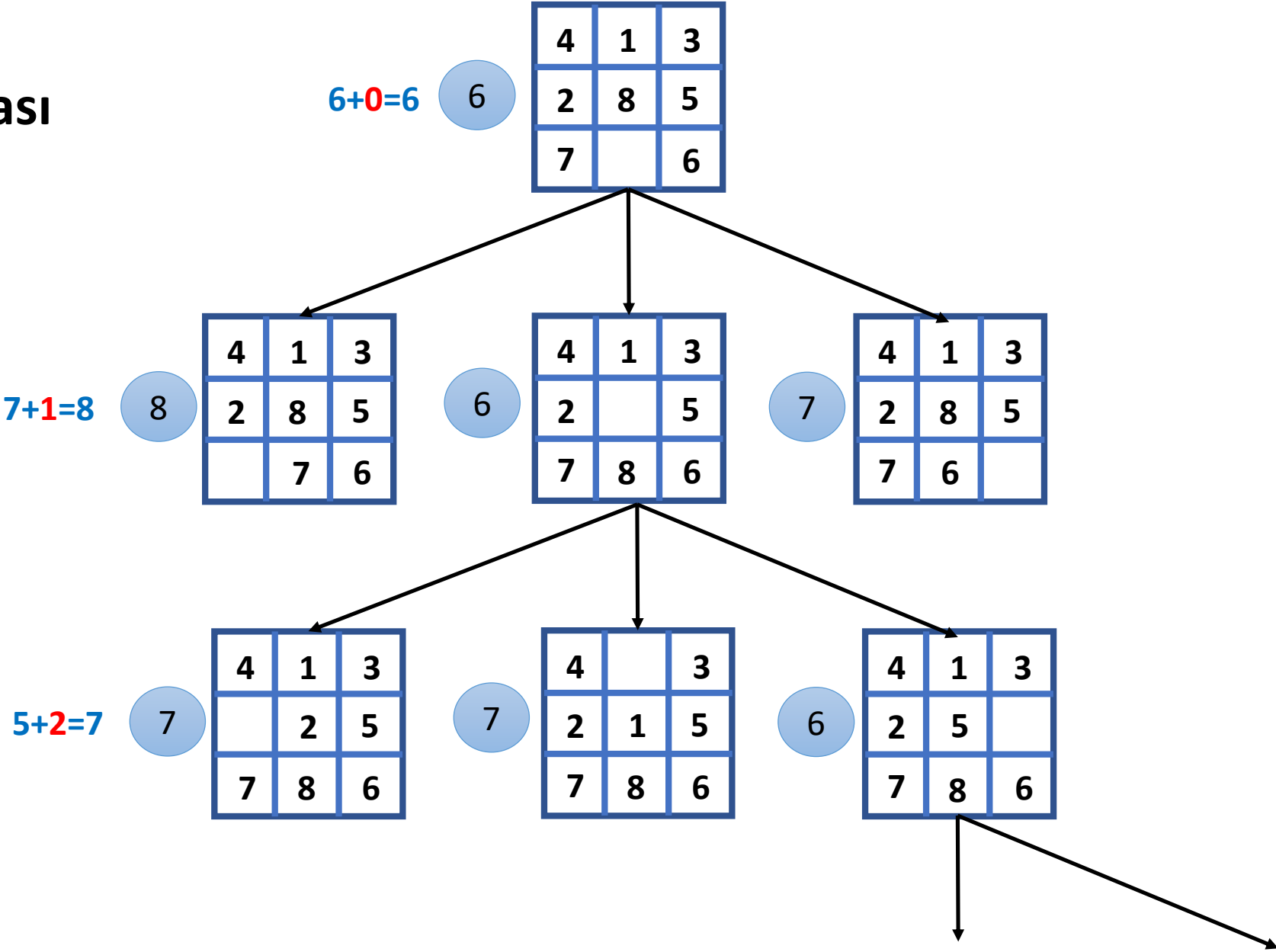
- Amaç hedef durum için boşluğun en kısa hareket dizisini bulmak
- Boşluğun hareketi için her durumda en fazla 3 operatör vardır.
- Minimum n hareketle hedef duruma ulaşıyor ise algoritma karmaşıklığı: $O(3^n)$.
 - Örneğin, 10 adımda çözüme gidiliyor ise $3^{10} = 59049$ düğümü açmak gerekir. Sezgisel bir yaklaşım olmadan çözüm çok uzun zaman alır.

A* algoritması – 8 taş oyunu

- Çözüm için $g(x)$ başlangıçtan bulunulan düğüme kadar olan derinlik
- $h'(x)$ için ise kendi yerlerinde olmayan taşların sayısı olarak belirlenebilir.
- Bu belirlemelerin ardından çözüm yolu için minimum $f(x)$ değerine sahip düğüm seçilir, operatörler yardımı ile çocuk düğümler oluşturulur ve yeni $f(x)$ değerleri hesaplanır.

4	1	3
2	8	5
7		6

A* algoritması
8 taş oyunu

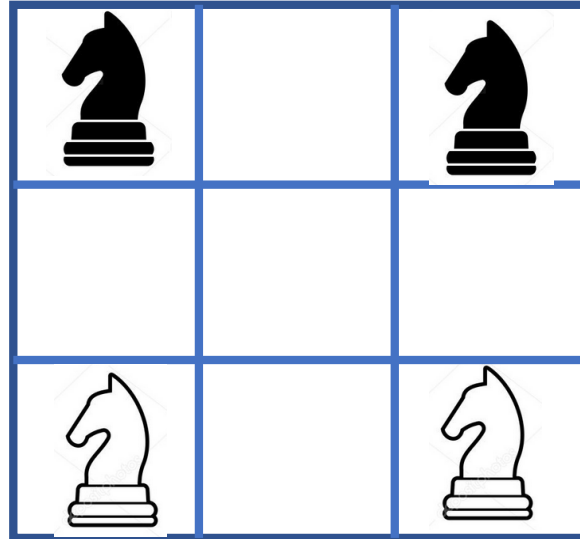


A* algoritması – 8 taş oyunu

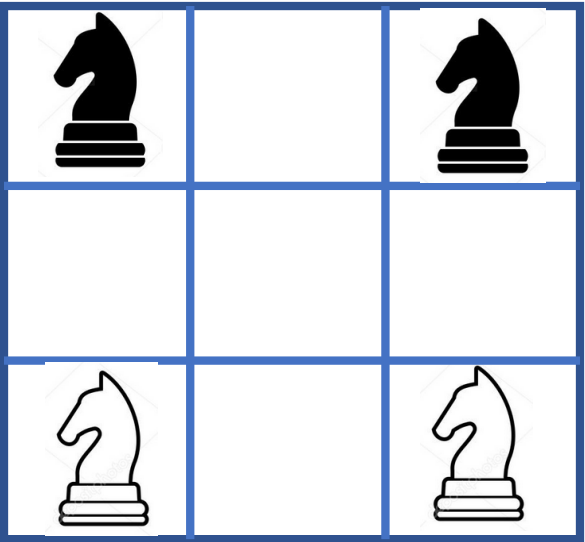
- $h'(x)$ fonksiyonu farklı şekillerde de tasarlanabilirdi:
 - $h'(x)_1$ bulunulan durum ile hedef durum karşılaştırıldığında yerinde olmayan taşların sayısı
 - $h'(x)_2$ taşların olması gereken yere olan uzaklıkları
- Başlangıç durum: 2 3 6 1 7 5 4 8 Hedef durum: 1 2 3 4 5 6 7 8
 - $h'(x)_1 = 7$
 - $h'(x)_2 = 8$

Sezgisel problem örnekleri: Dört at problemi

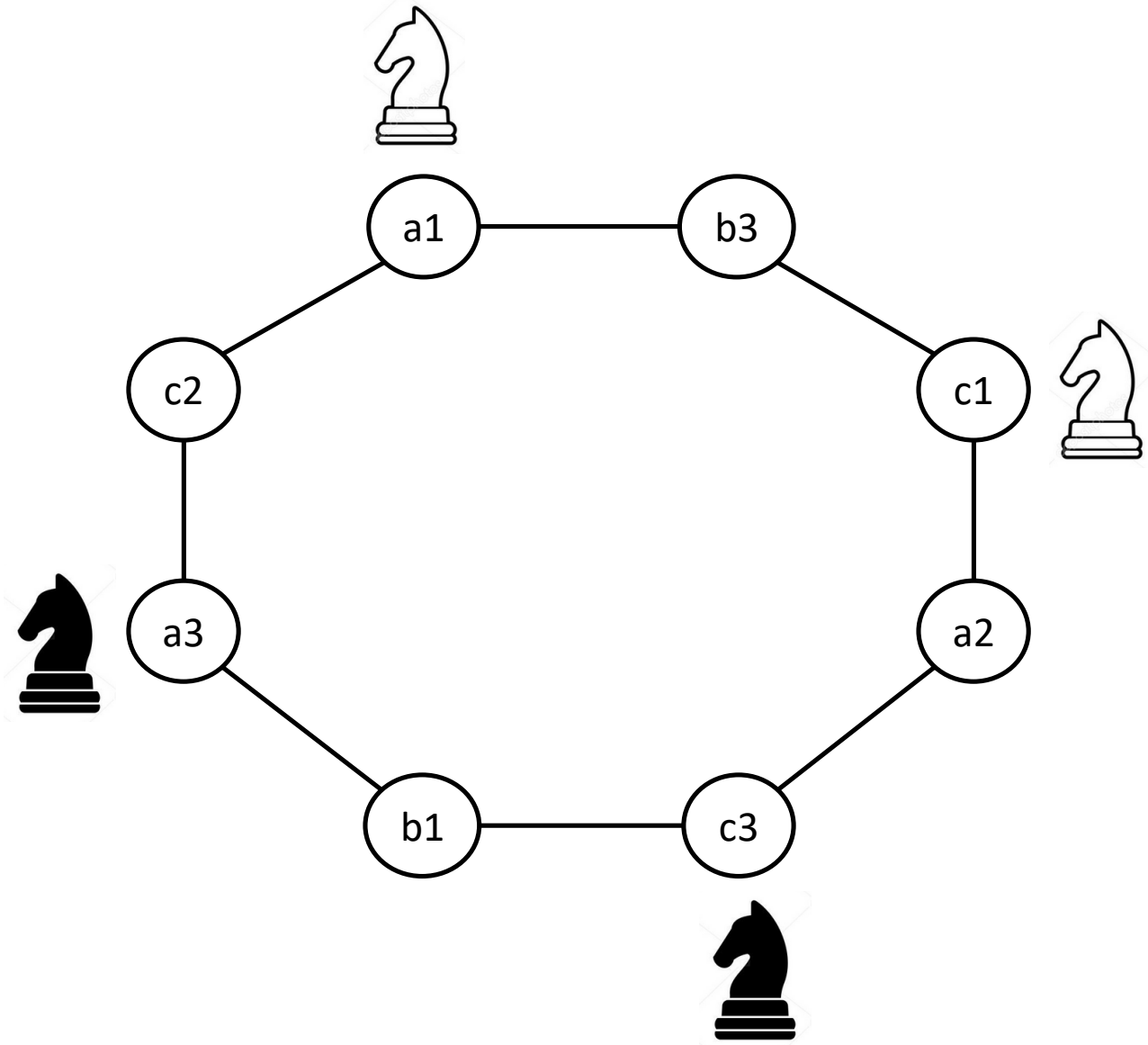
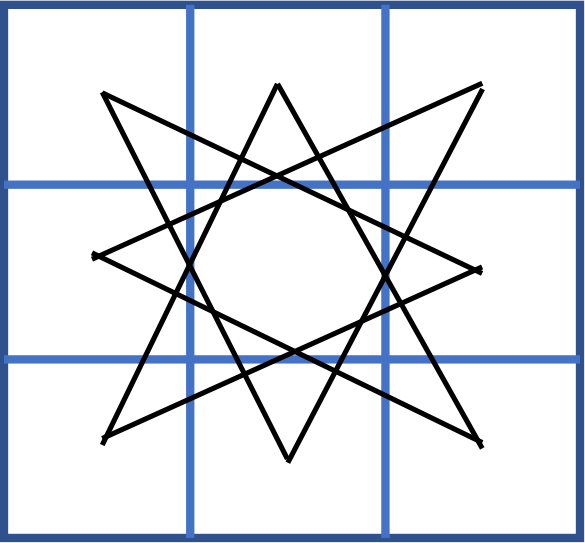
- 3*3 boyutlu mini satranç tahtası
- Tahtanın köşelerinde iki beyaz iki siyah at
- Minimum sayıda gidişle siyah ve beyaz atların yer değiştirmesi?



4 at problemi

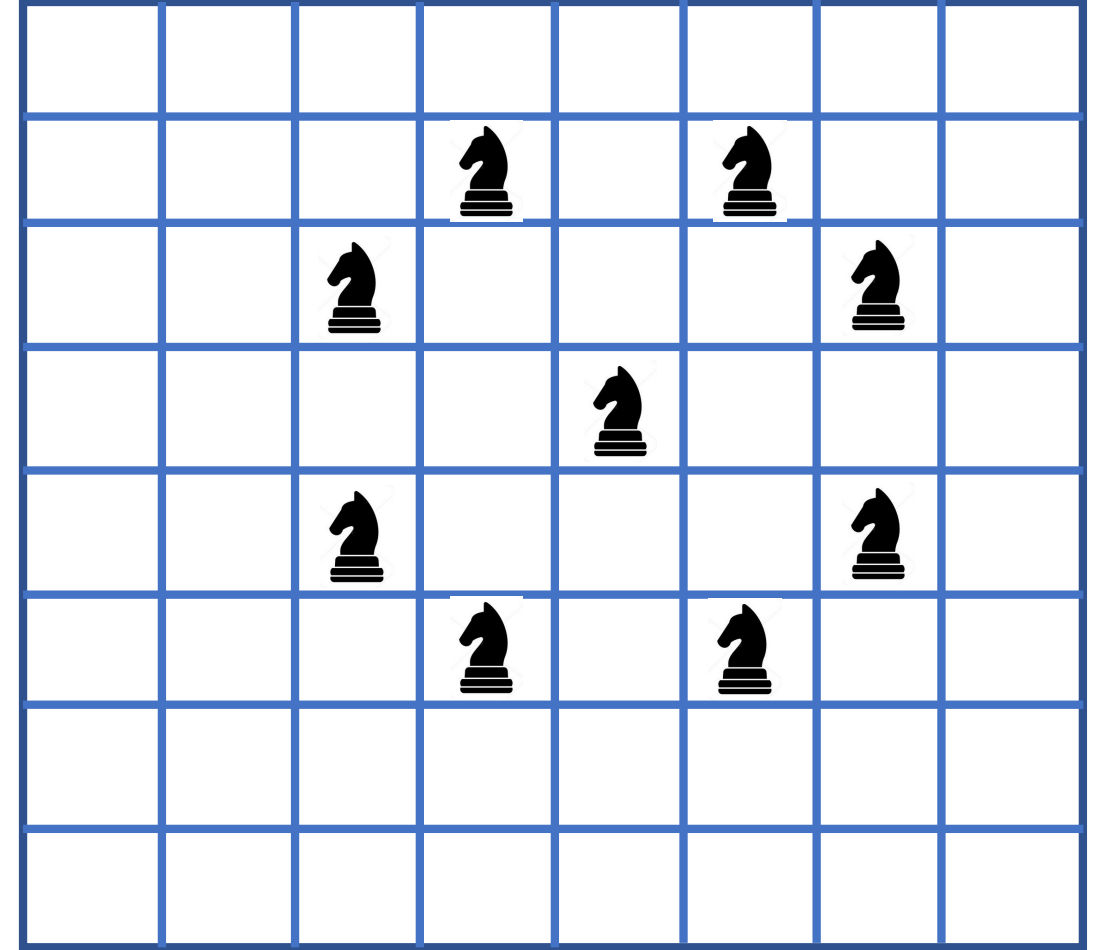


a b c



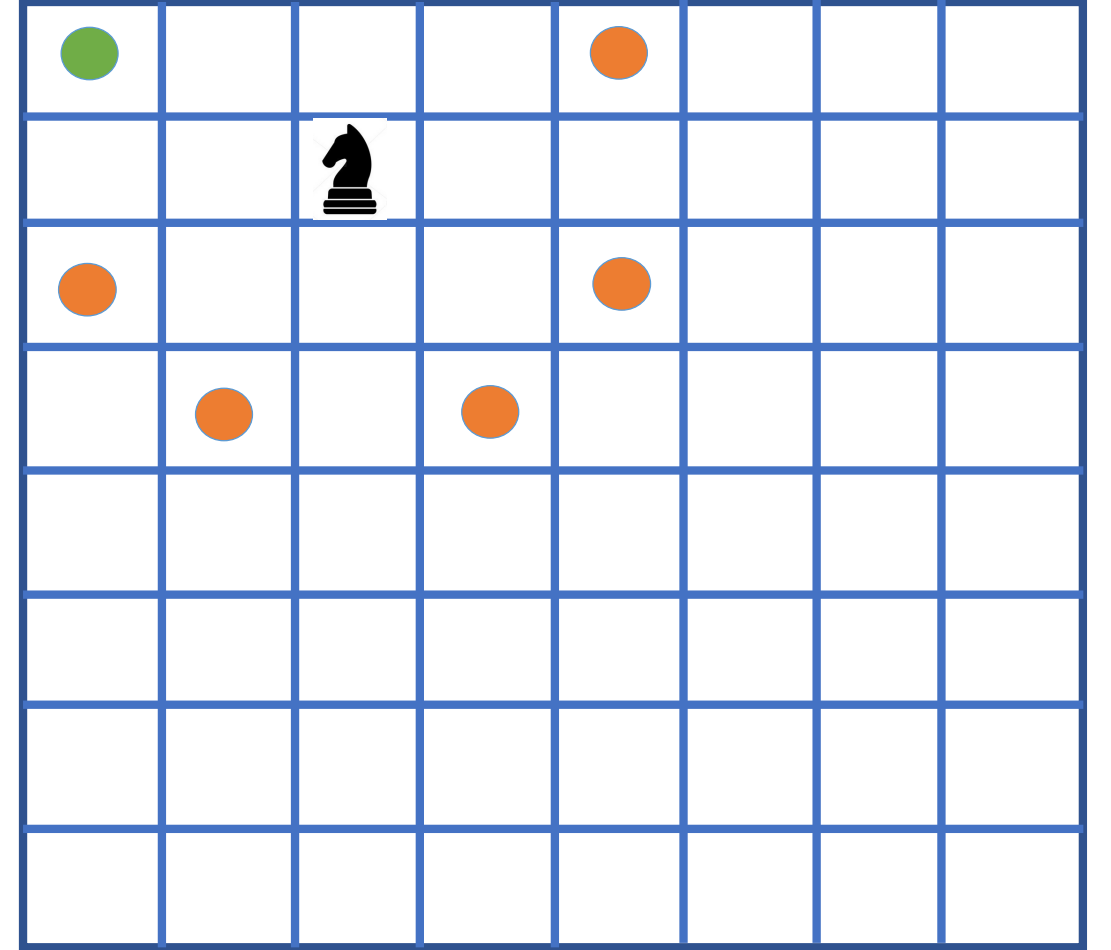
64 at problemi

- Bir at, tahtanın herhangi bir hanesinden başlamak ve her haneyle yalnızca bir kez ziyaret etmek şartı ile tüm haneleri gezmelidir.
- Bir at, tahta üzerindeki herhangi bir konumda iken en fazla 8 gidiş seçeneği vardır.
- Bu sayı ortalama olarak 4 kabul edilse çözümün aranması 4^{63} düğüm



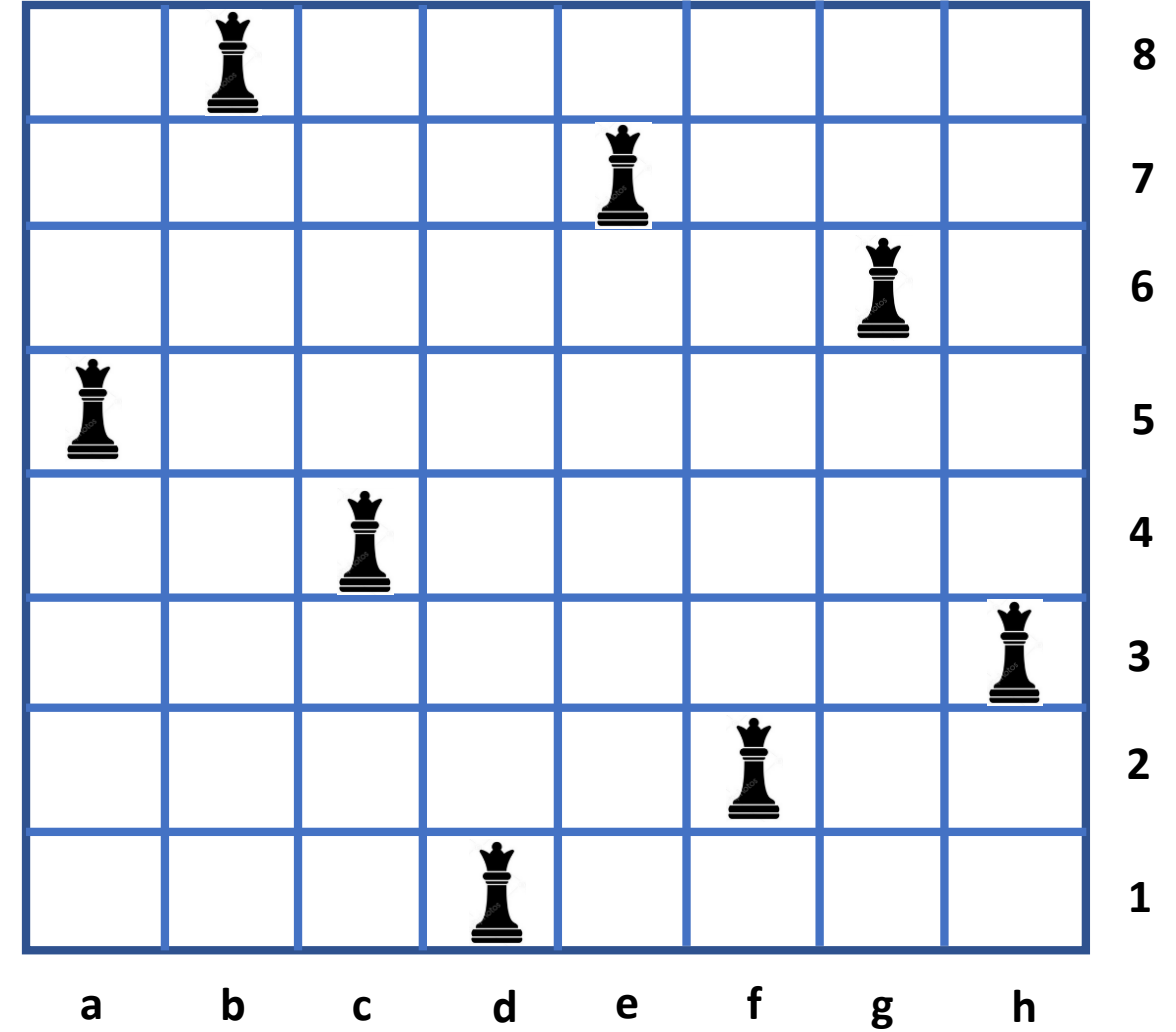
64 at problemi

- Çözüm olarak, atın mevcut durumdan gidebileceği olası durumlardan en az çıkışı olanın seçimi yaklaşımı önerilmiştir.

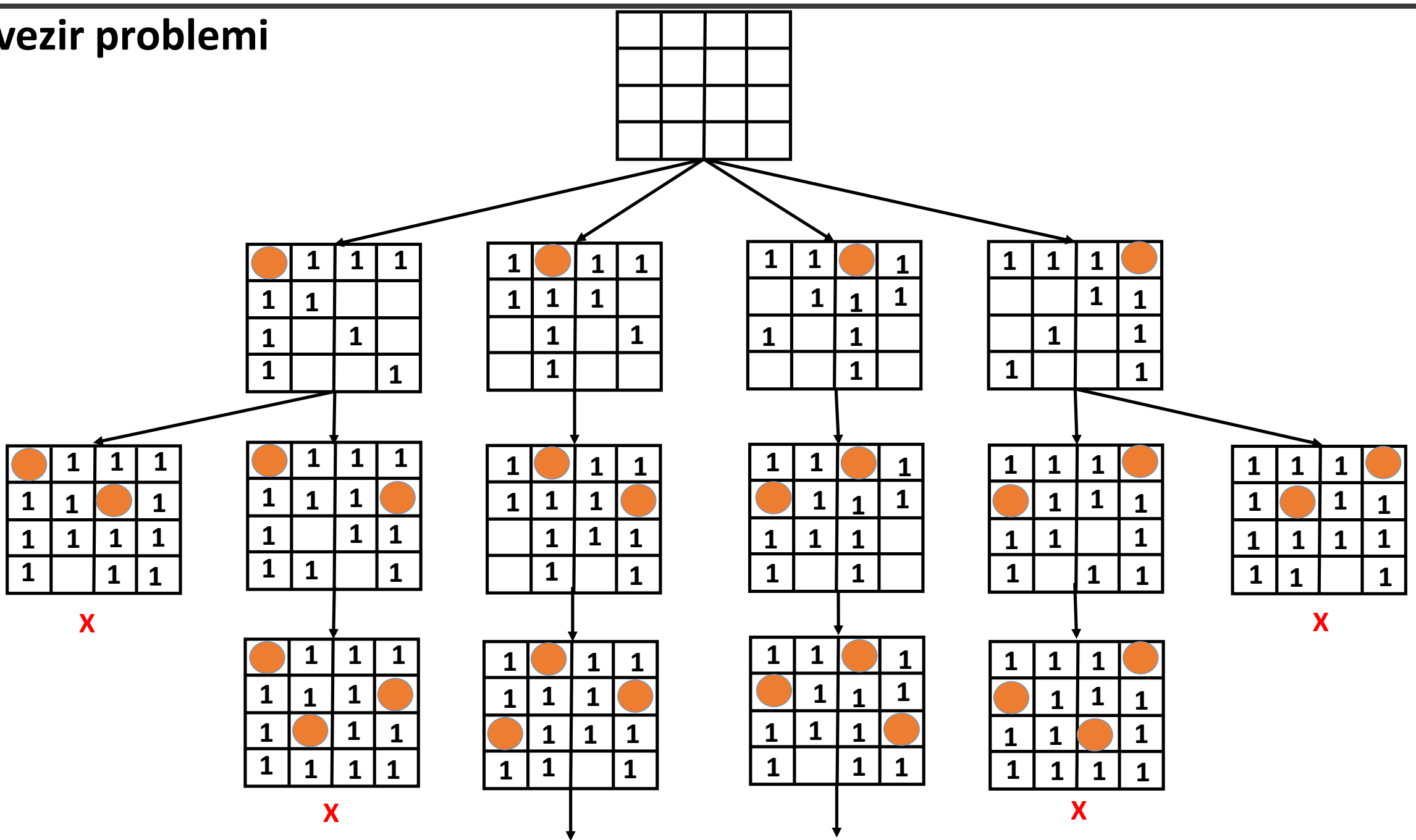


8 vezir problemi

- $n*n$ 'lik bir satranç tahtası üzerinde birbirini görmeyecek şekilde en çok sayıda vezir yerleştirme
- $8*8$ 'lik bir satranç tahtası ise problem *8 vezir problemi* ismini alır.
- $4*4$ 'lük tahta üzerinde problemin çözümünü inceleyelim.



8 vezir problemi





8 vezir problemi

- Sezgisel bir değerlendirme yapılmaz ise durum uzayını bünyesinde barındıran ağacın tüm dallarında çözüm aranır.
- Durum uzayı büyüdükçe ağacın derinliği de artacaktır.
- Tüm dalların gezilmesi için açılacak düğüm sayısı n^n
- 2016 yılı dahil tüm çözümleri bilinen vezir sayısı 26'dır.

$n*n$	Çözüm sayısı	Tek çözümler
1*1	1	1
2*2	0	0
3*3	0	0
4*4	2	1
...
26*26	22.317.699.616.364.044	2.789.712.466.510.289




8 vezir problemi: 1. yaklaşım

- Başlangıç durumu: $4c, 3f$
- Hamle altında olabilecek hücreler ilgili vezirin numarasını alır.
- Sezgisel parametre: yatay yöndeki en küçük serbestlik derecesi
- Serbestlik derecesi: satırlar ya da sütunlar boyunca mümkün olan yerleştirme sayısı

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

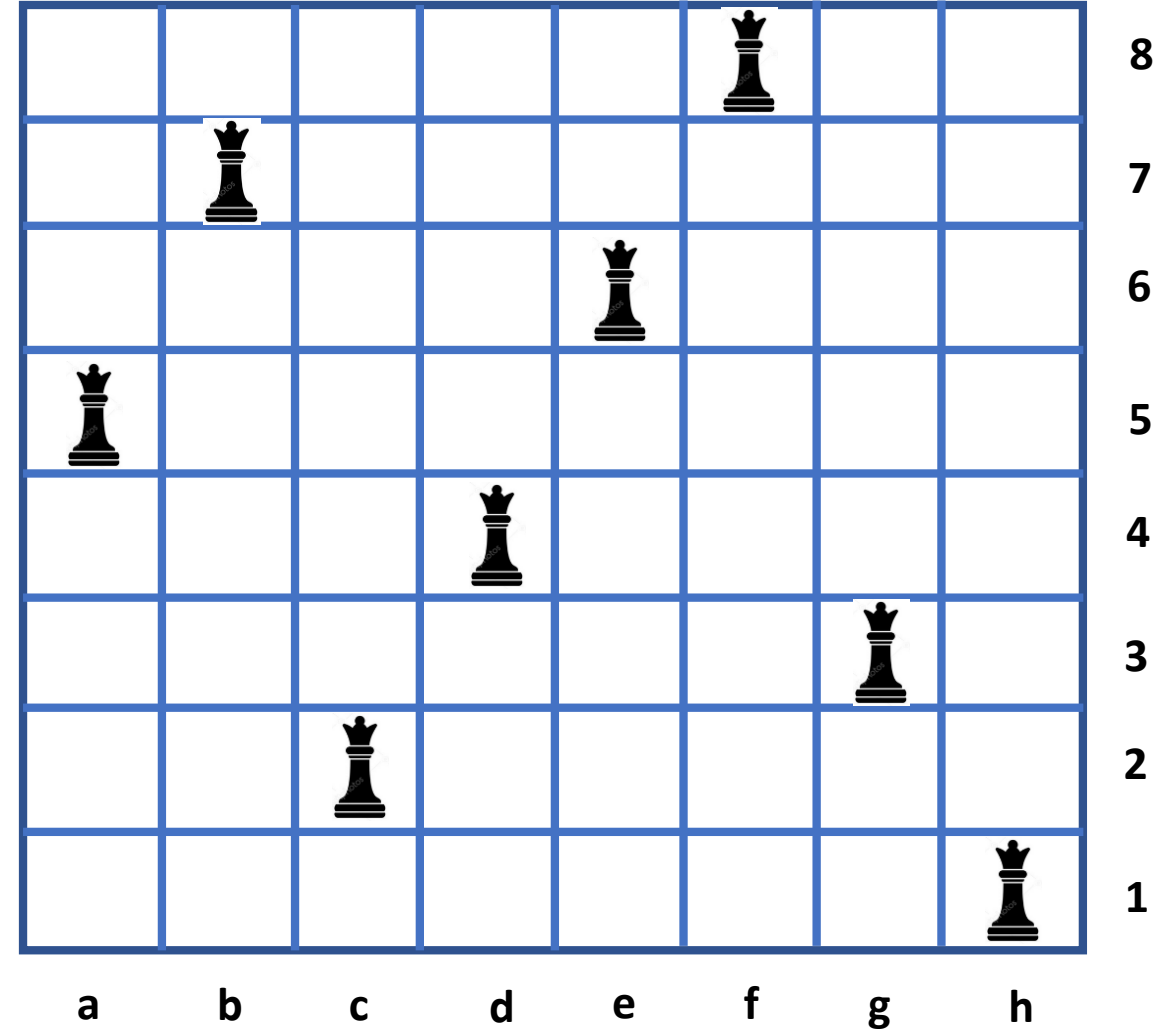
8 vezir problemi: 1. yaklaşım

- En düşük serbestlik derecesi: 2. ve 5. satır (5. satırı seçelim)
- 5. satır için sütun dereceleri incelendiğinde a sütunu seçilir.
- ...
- *a5, b2, c4, d6, e8, f3, g1, h7*

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

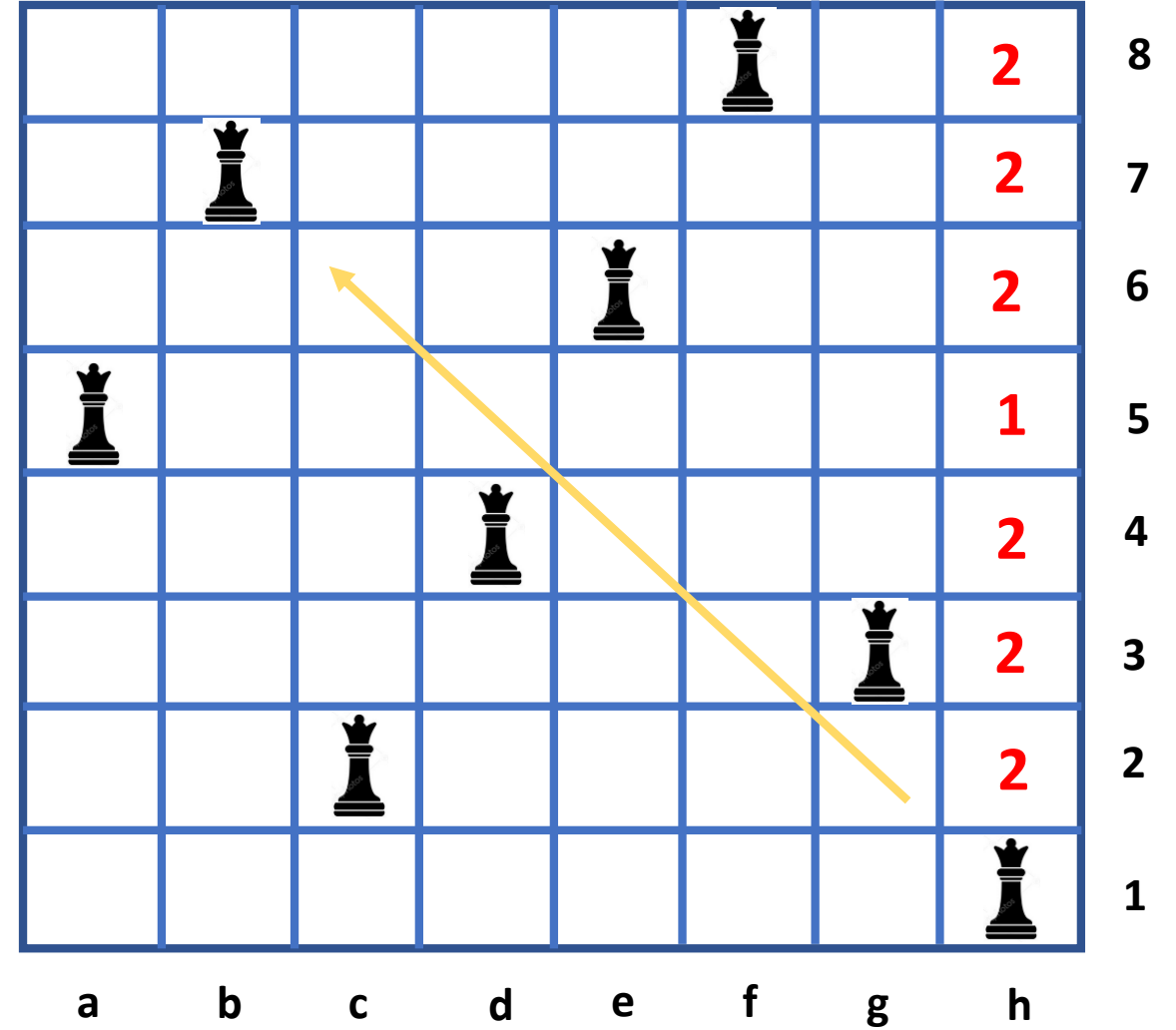
8 vezir problemi: 2. yaklaşım (sezgisel onarım)

- Sezgisel yöntem ile problem hızlı bir şekilde çözülebiliyor.
- Fakat sezgisel fonksiyon iyi karakterize edilmeli.
- "Sezgisel onarım" yönteminde var olan bir durumda çatışan durumlar belirlenir ve bu durumlar minimuma çekilir.
- Vezirler şekildeki gibi rastgele yerleştirilmiş olsun.

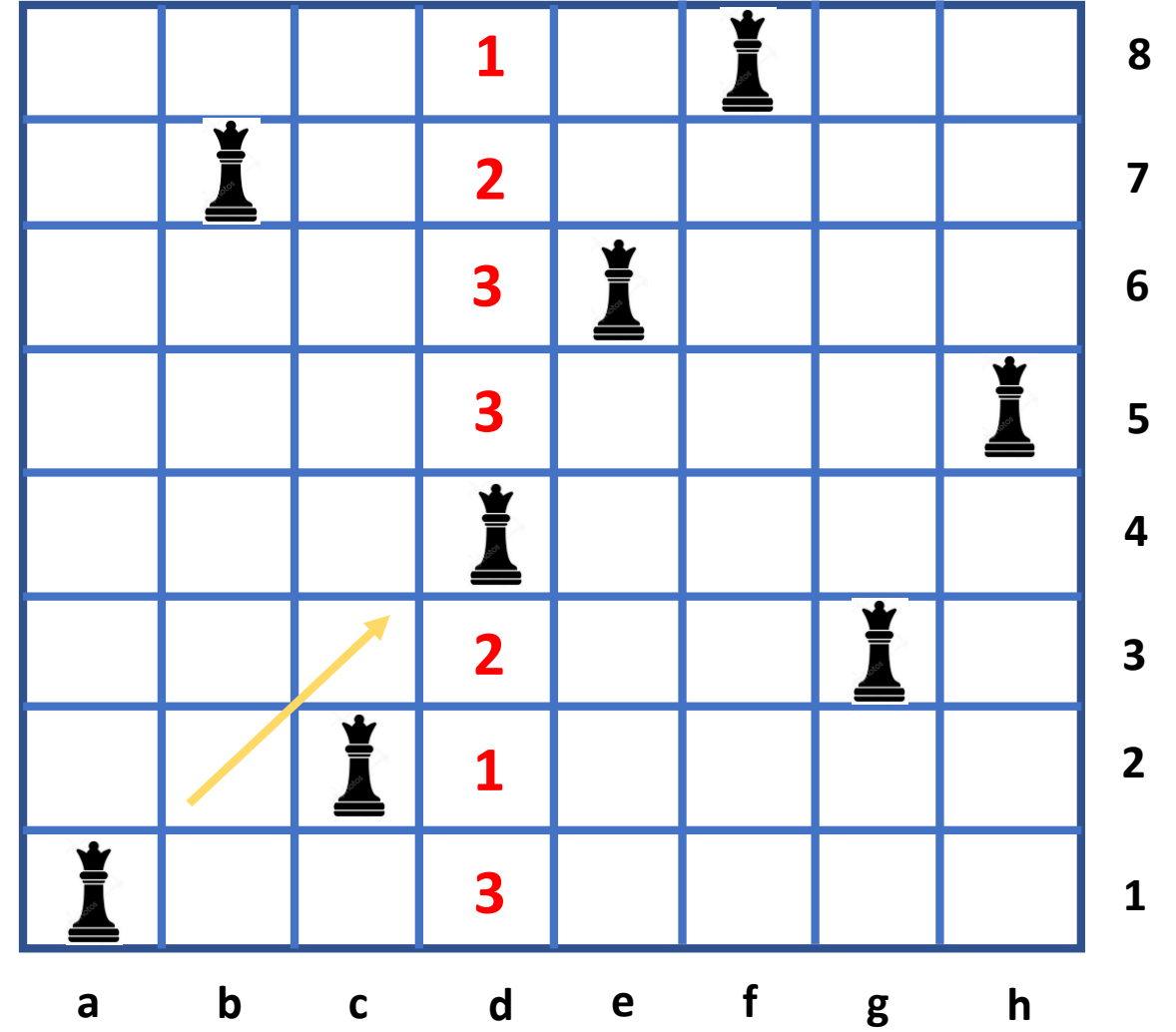
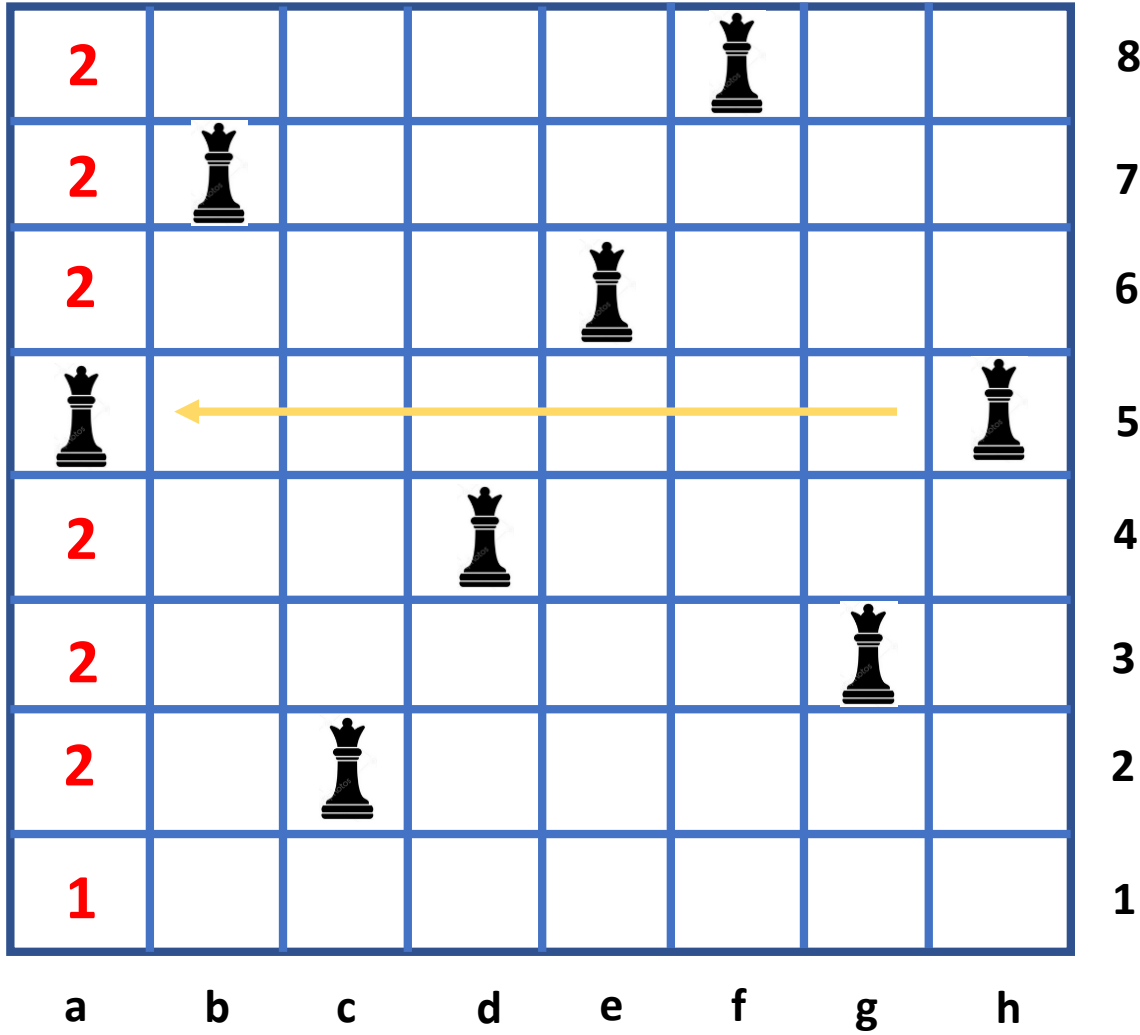


8 vezir problemi: 2. yaklaşım (sezgisel onarım)

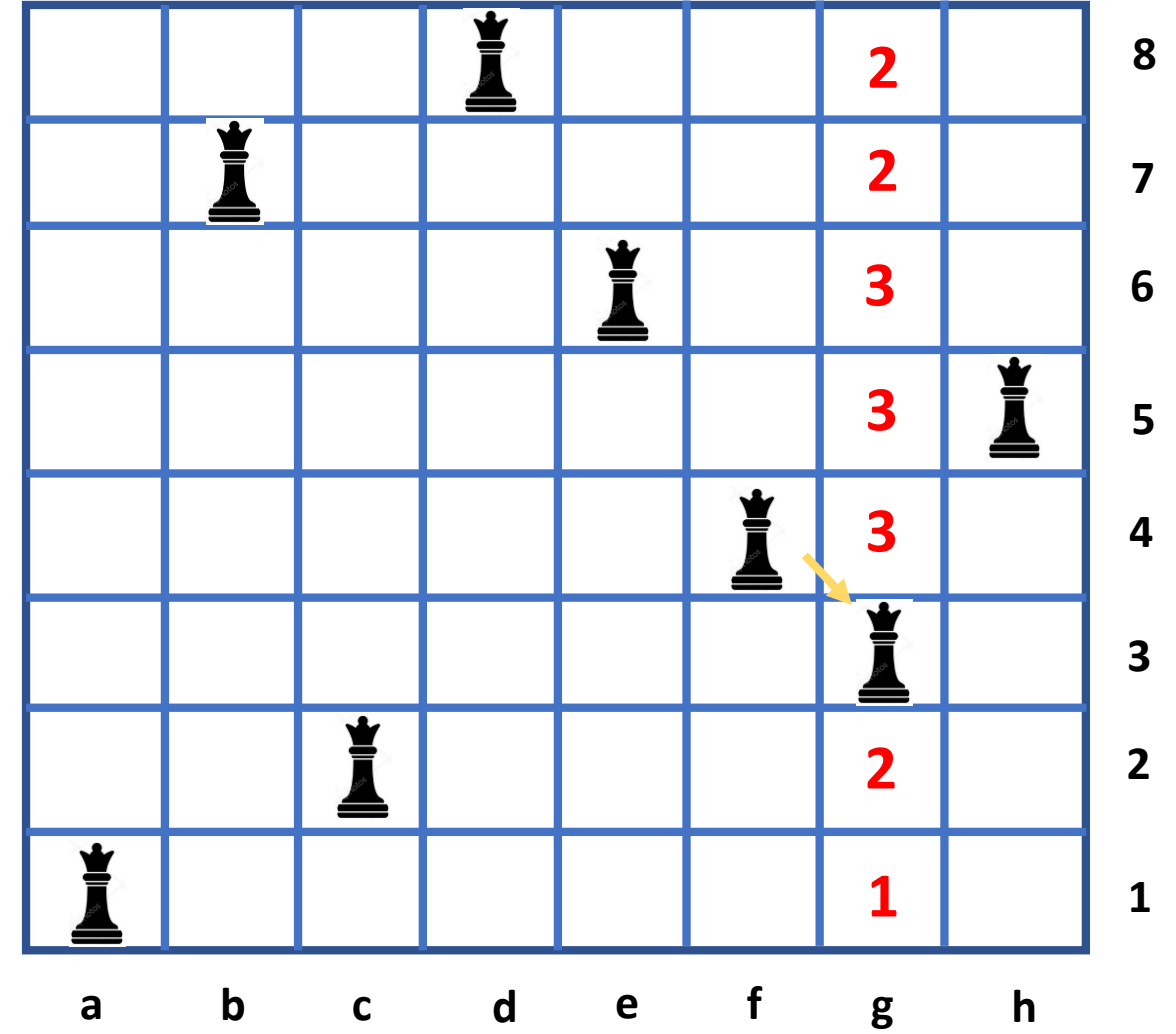
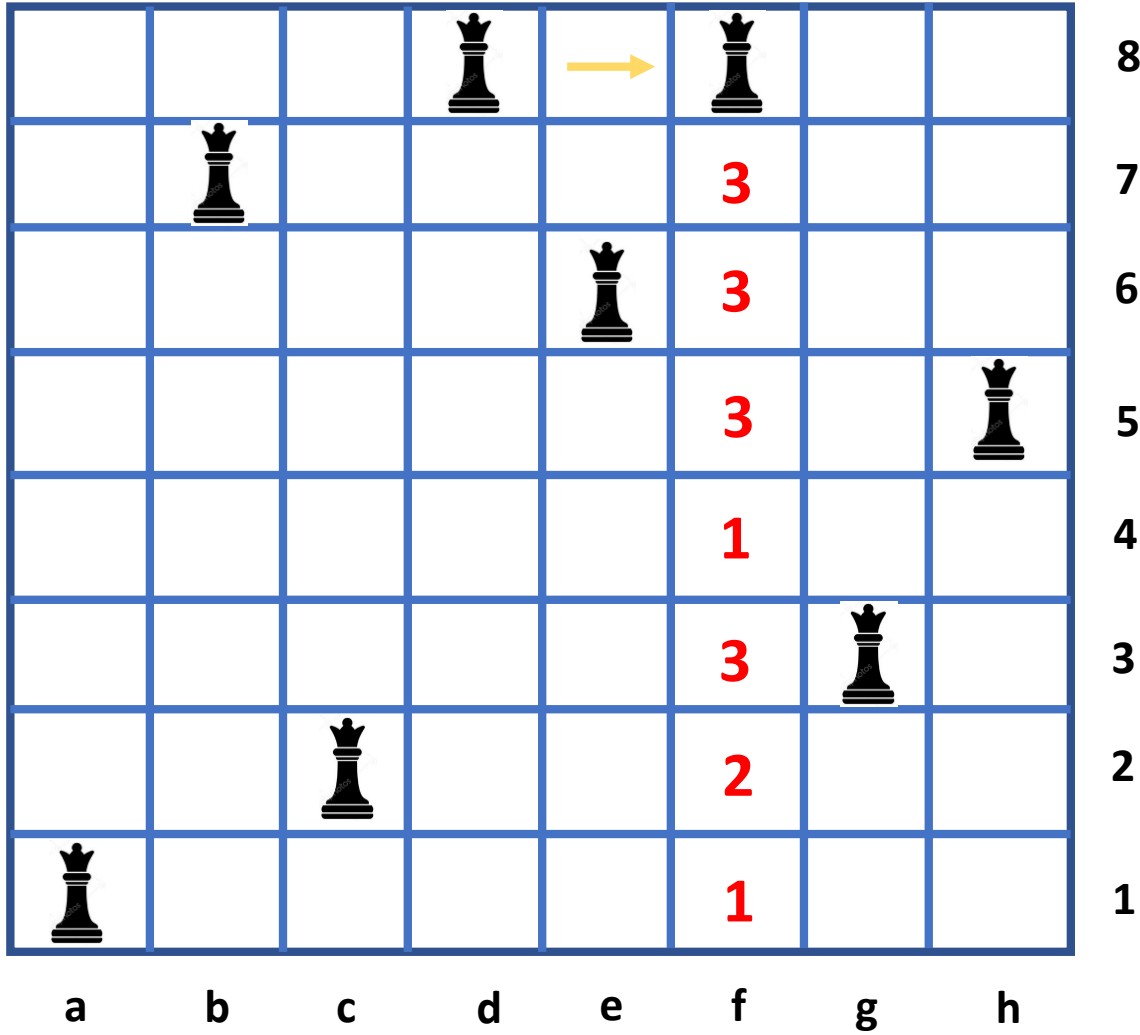
- a5 tehdit altında değil
- b7 h1'in tehdidi altında
- Çatışma durumundaki h1 sütununun satır değerleri her hücre için kaç vezirin tehdidi altında olduğu sayıdır.
- h1 hücresindeki vezir h5'e çekilir.



8 vezir problemi: 2. yaklaşım (sezgisel onarım)



8 vezir problemi: 2. yaklaşım (sezgisel onarım)



8 vezir problemi: 2. yaklaşım (sezgisel onarım)

