SAKARYA
ÜNİVERSİTESİ

BSM 310
YAPAY ZEKA

CEMİL ÖZ, İSMAİL ÖZTEL

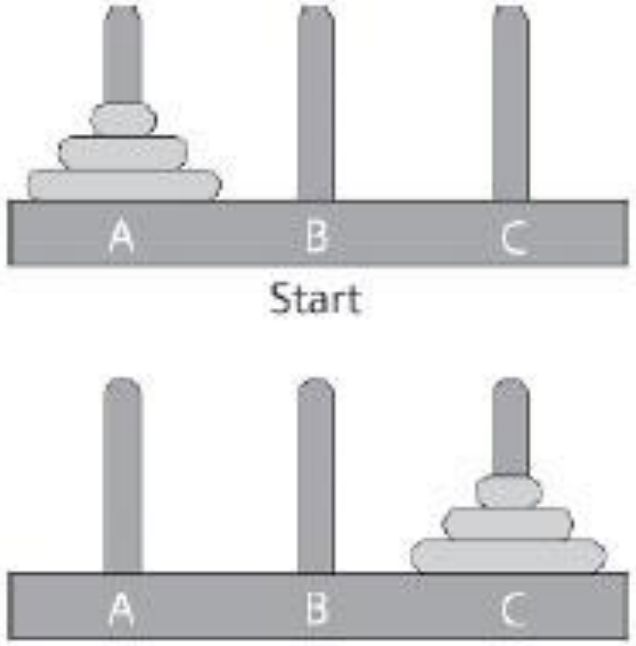
~ SEZGİSEL PROBLEM ÇÖZME YAKLAŞIMI – II ~

KONULAR

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

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.

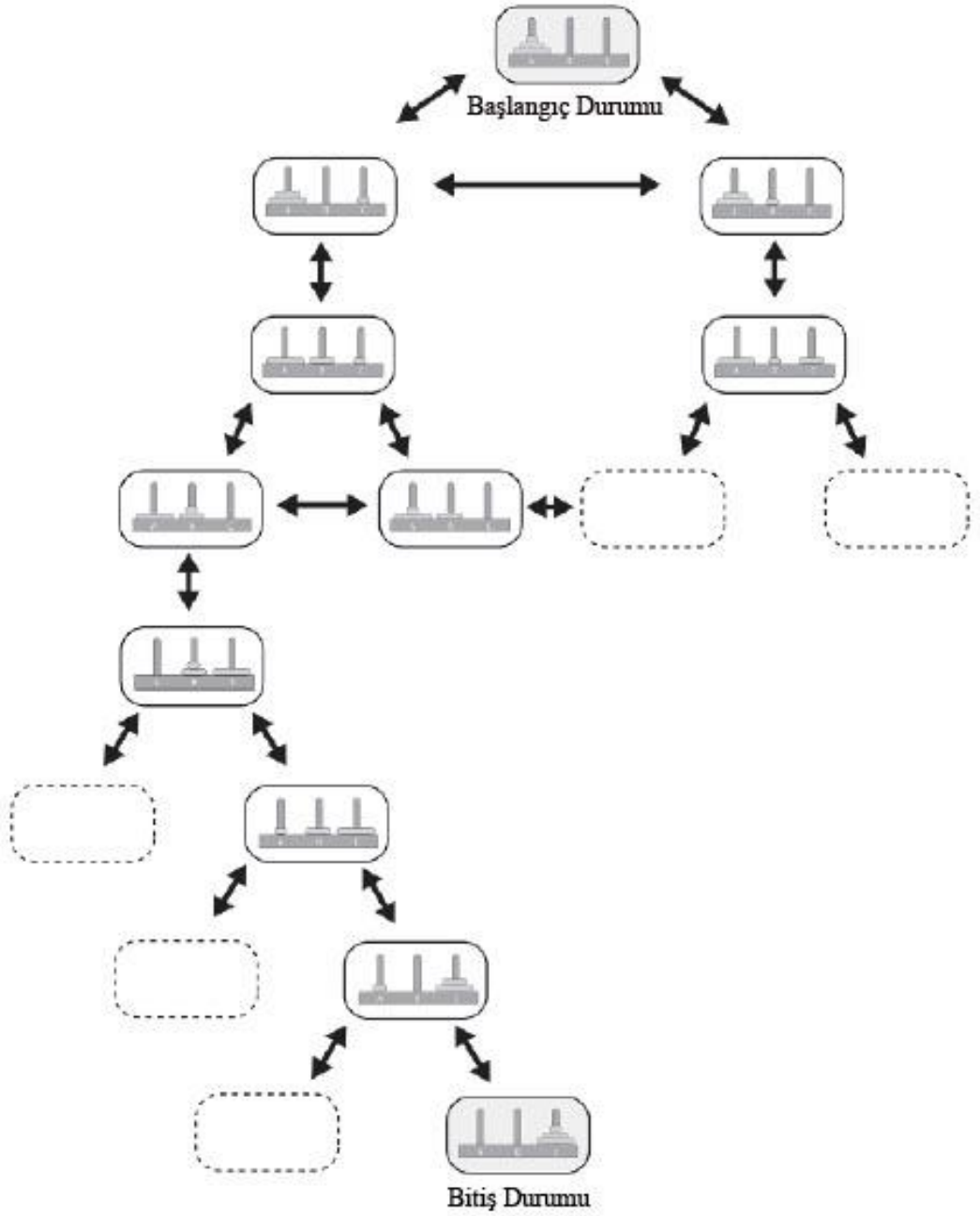
- Graf kenarları; durumlar arasındaki geçişleri gösterir. Eğer bir durumdan diğerine direkt olarak geçiş mümkünse, bu düğümleri birbirine bağlayan bir kenar mevcut olacaktır, aksi takdirde böyle bir bağlantı olmayacaktır.

- 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.

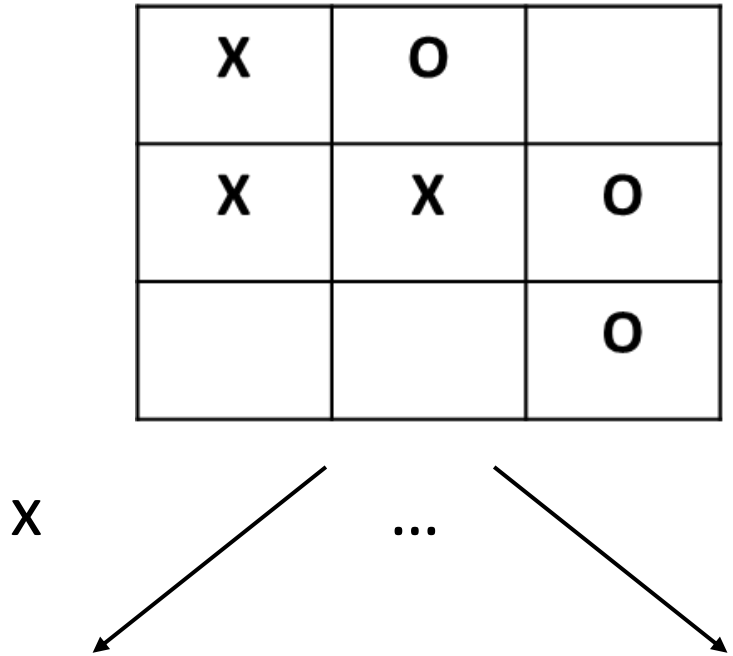
- Her durumun bir önceki durumu ebeveyn durum ve oluşturulan yeni durum çocuk durum olarak adlandırılır.

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 grafı 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.



Alıştırma: tic-tac-toe



Durum grafları

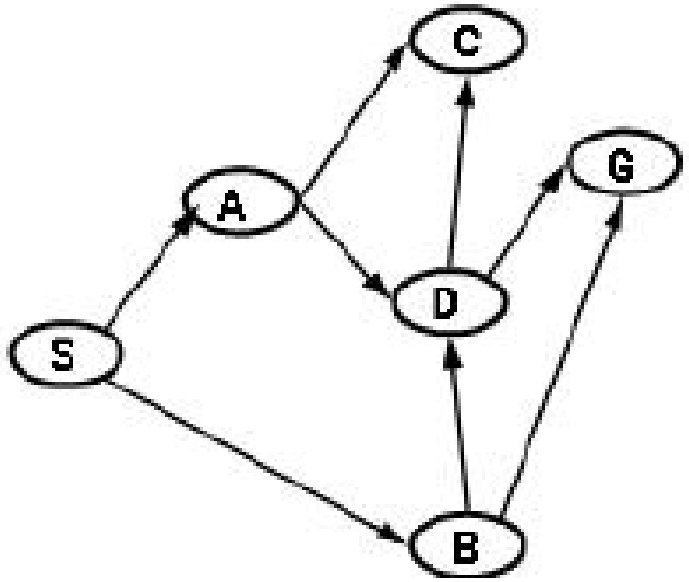
- Bir durum grafı; sistemin içinde bulunabileceği bütün durumların gösterimidir ve bu durumlar arasındaki geçişlerden oluşur.
- Bu sistemin bütün potansiyel durumlarının toplamı, durum uzayı olarak bilinir.
- Bu tip bir graf, özel bir durum mümkünse bunu görmek; ya da belli bir durum için en etkili yönü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

Durum grafları

- Her ebeveyn düğümden 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 düğümden 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.

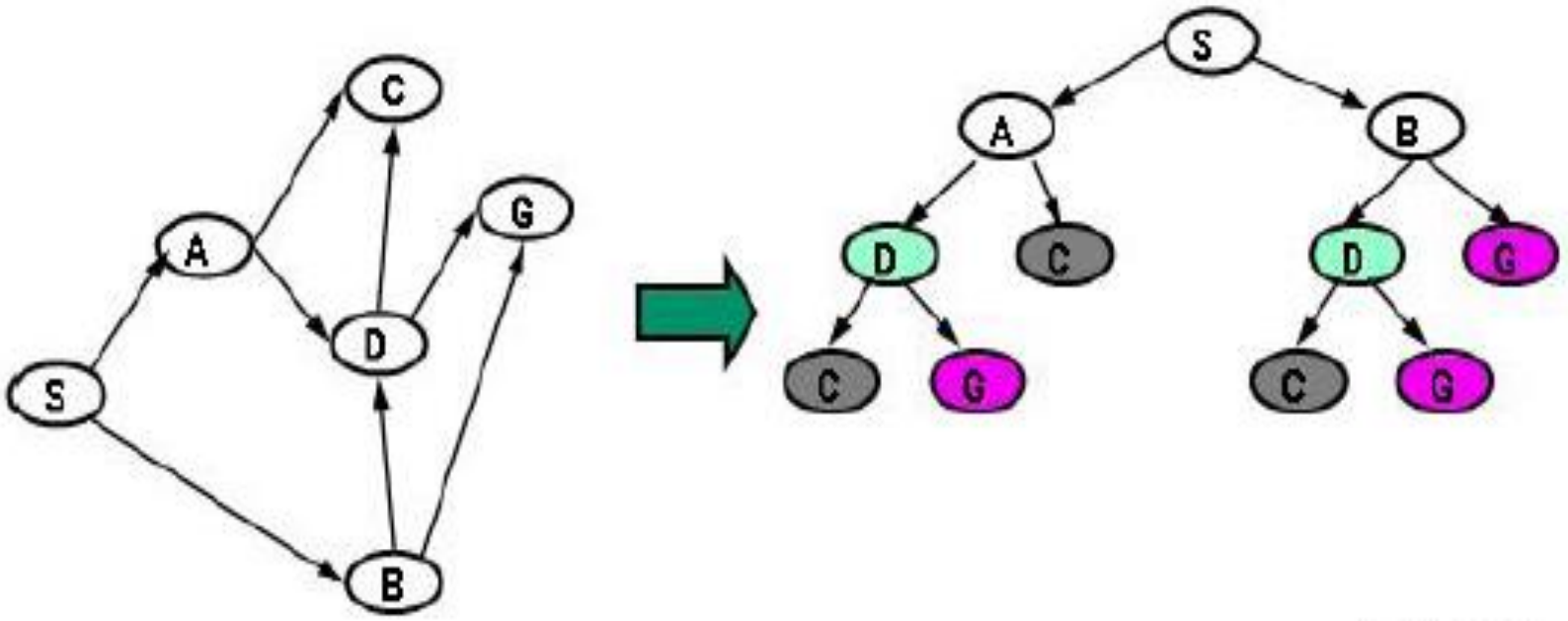
Ağaçlarda arama işlemlerinin graflara uygulanması

- 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:

```
   D0NALD
+  GERALD
-----------
   R0BERT
```

- 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

```
   D0NALD
+  GERALD
-----------
   R0BERT
```

- 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.

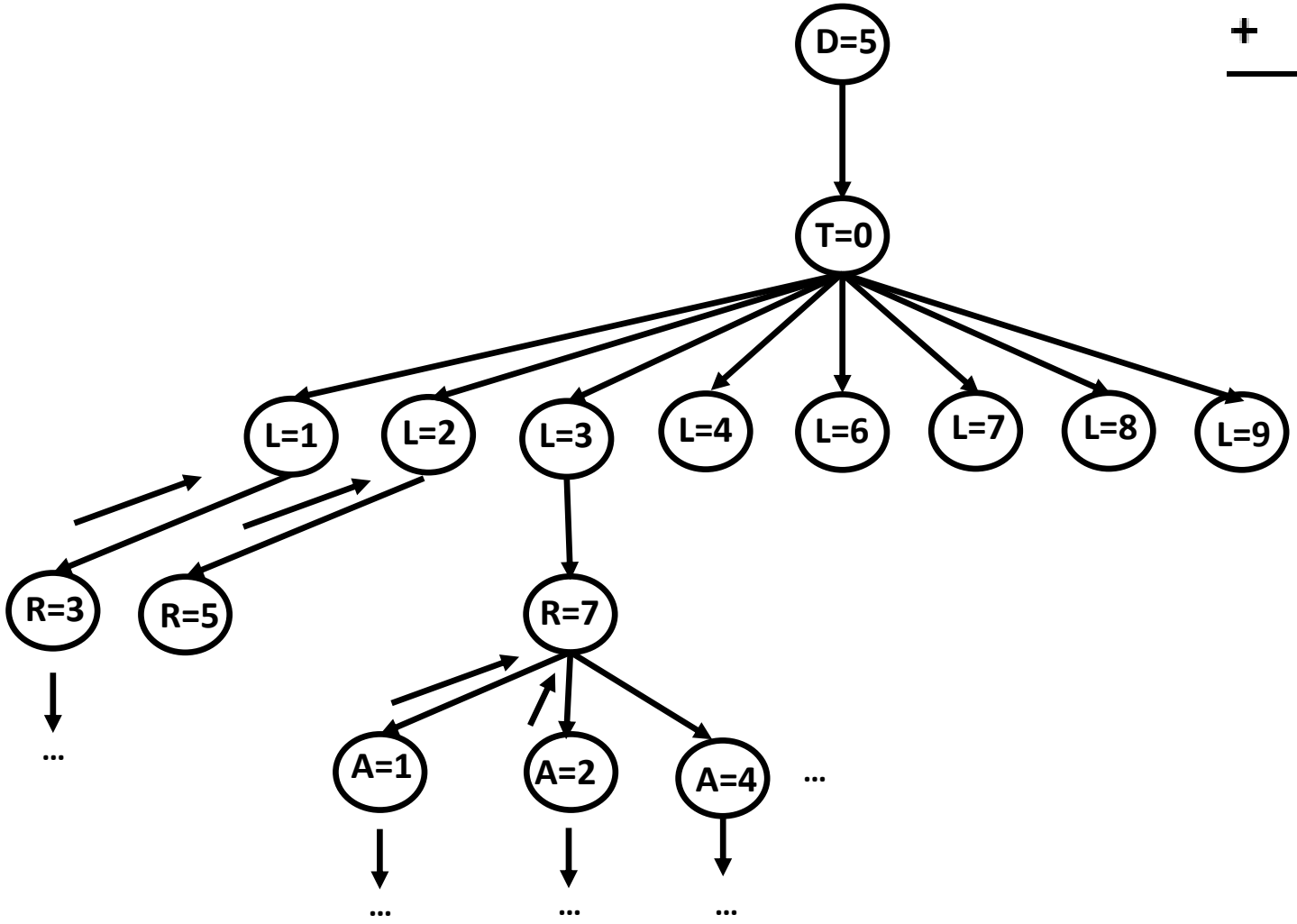
Şifreli hesaplama problemi

```
  DONALD
+ GERALD
--------
  ROBERT
```

- Problem bilgisayar yardımı ile de benzer şekilde çözülür.
- Olasılıkların yer aldığı çözüm ağacı oluşturulur.
- D=5 → 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 → R>5 olmalı
- …
- D=5, T=0, L=8, R=7, A=4, E=9, N=6, O=2, G=1, B=3

Şifreli hesaplama problemi

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

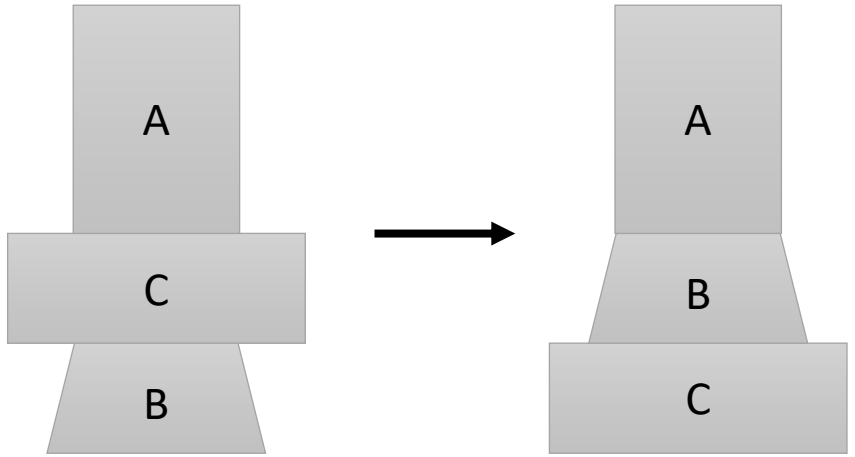


Durum uzayı

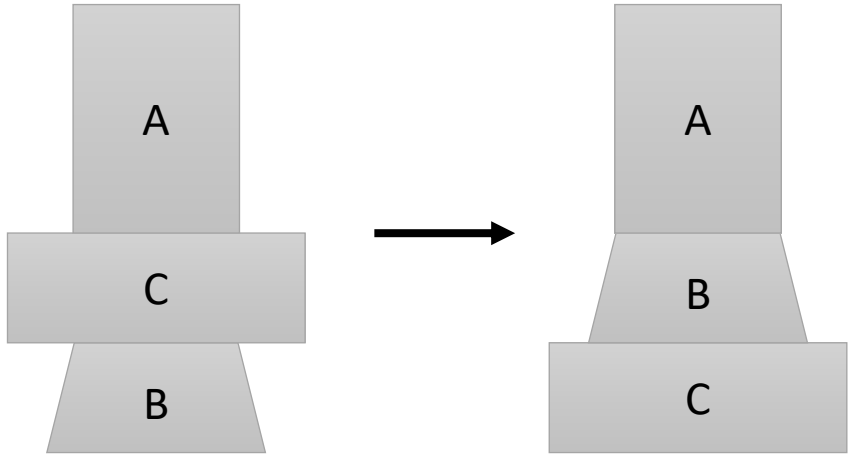
- Bir yapay zeka problemi için izinli eylemleri barındıran sonlu durumlar kümesi mevcuttur.
- Kümedeki ilişkili elemanlar graf olarak ifade edilir.
- İzinli tüm durumları içeren bu graf problemin durum uzayıdır.
- Problemin çözümü ilgili graftaki minimum yolun bulunması ile sağlanır.

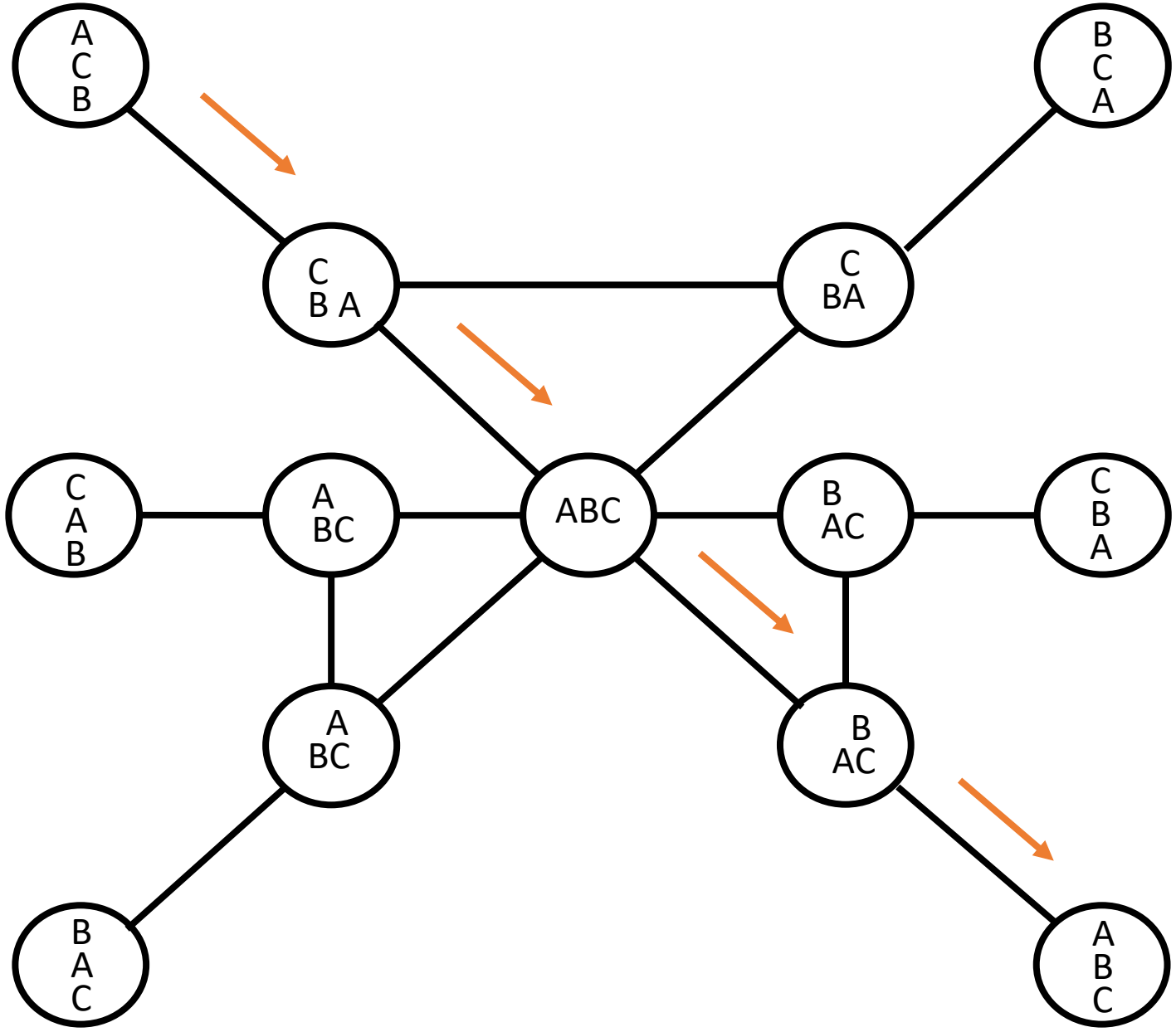
Durum uzayı - Örnek

- 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



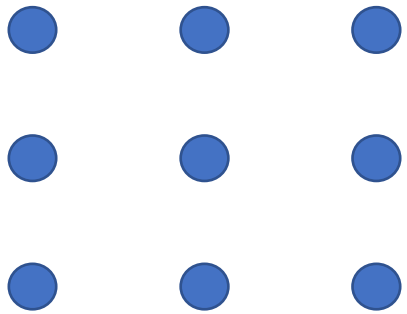


Durum uzayı

- 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 olabilir.
- Çünkü insan beyni ’’otomatik bir minimizasyon mekanizması’’ içerir.

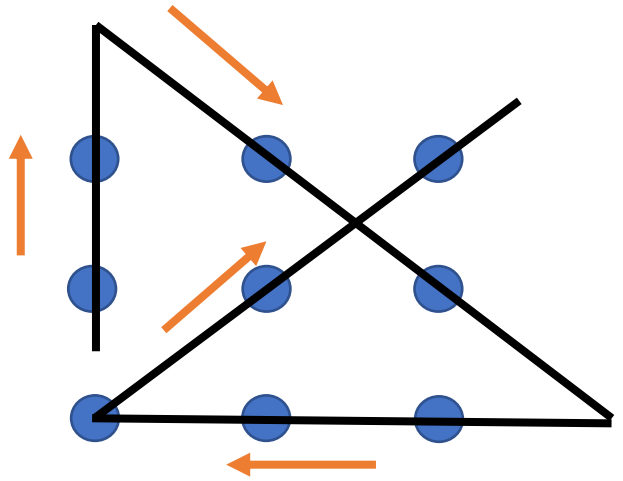
Durum uzayı – Örnek 2

- 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 birleştirilebilir mi?



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 birleştirilebilir mi?



Durum uzayı

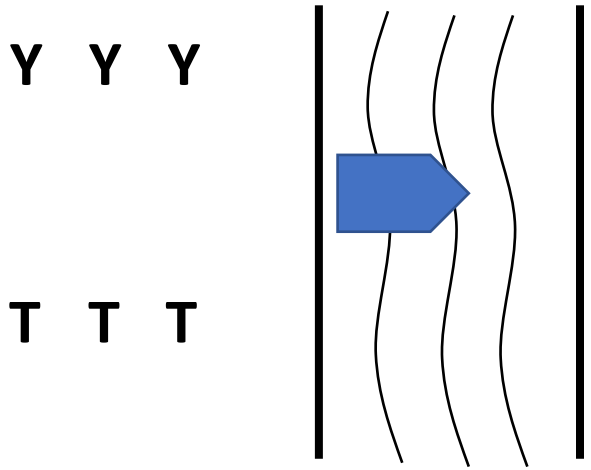
- Yapay zeka yöntemlerini bilmeyen bir programcı problemleri çözerken deneme/yanılma yöntemi ile çözüme gitmeye çalışabilir.
- Bu yaklaşım durum uzayı geniş olmayan problemler için çalışabilir.
- Durum uzayı büyüdükçe bilgisayar kaynakları yetersiz kalabilmektedir.
- Örneğin 15 taş oyunun durum uzayı 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

6	5	3
1	7	8
2	4	

Durum uzayı – Turistler ve yamyamlar problemi



- Tekne en fazla 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.

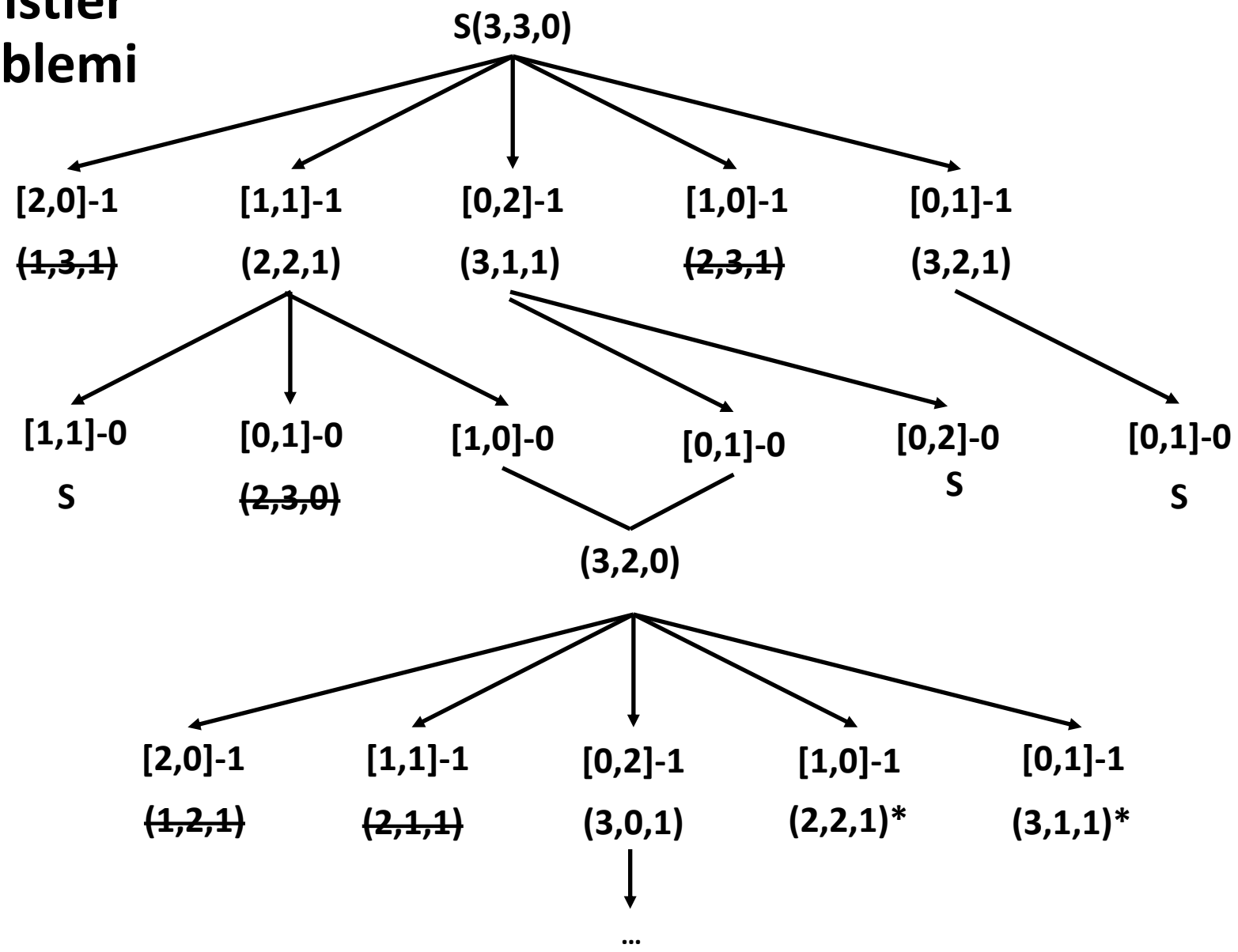
Durum uzayı – Turistler ve yamyamlar problemi

- (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.

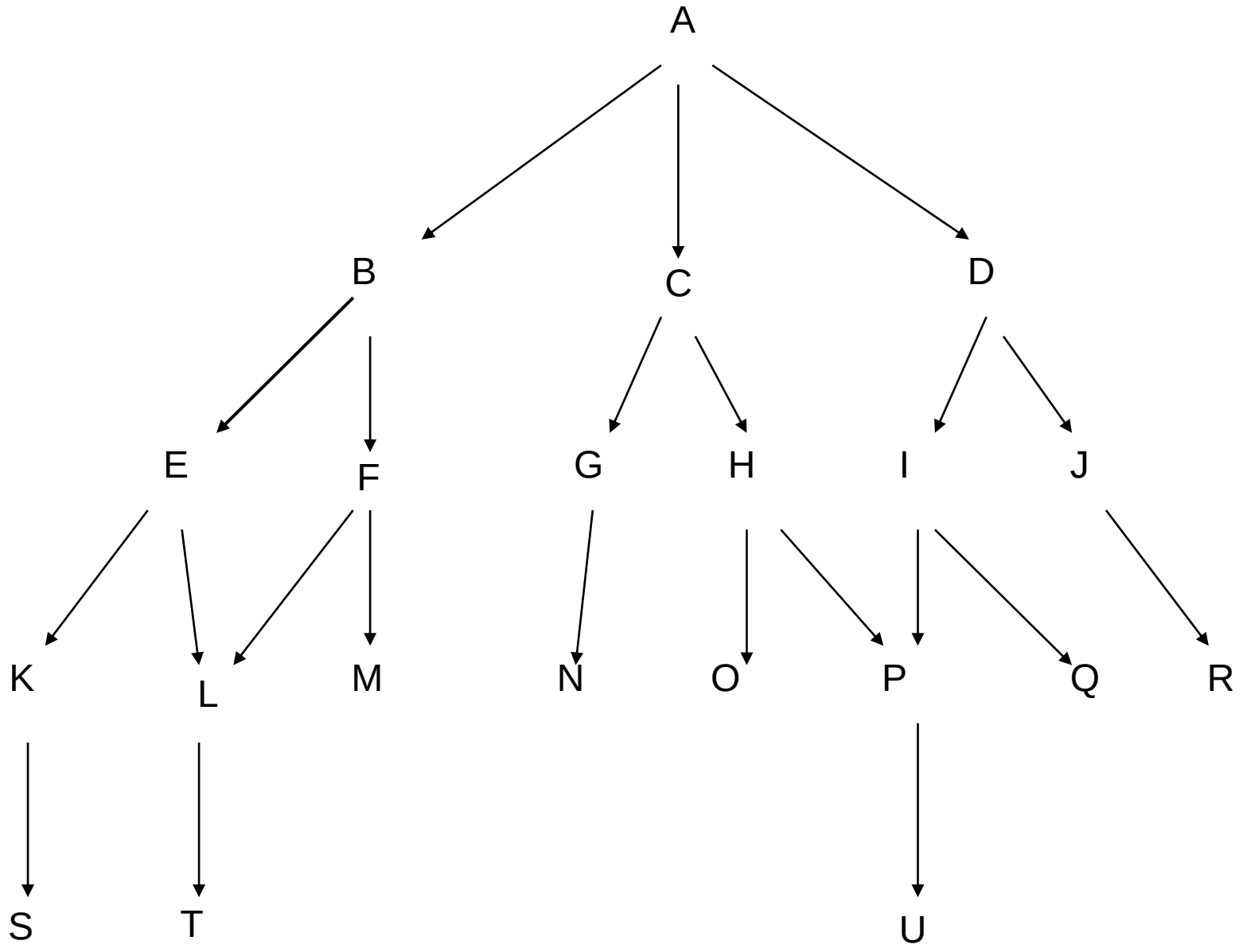
Durum uzayı – Turistler ve yamyamlar problemi



Depth-first search - Derinlik öncelikli arama

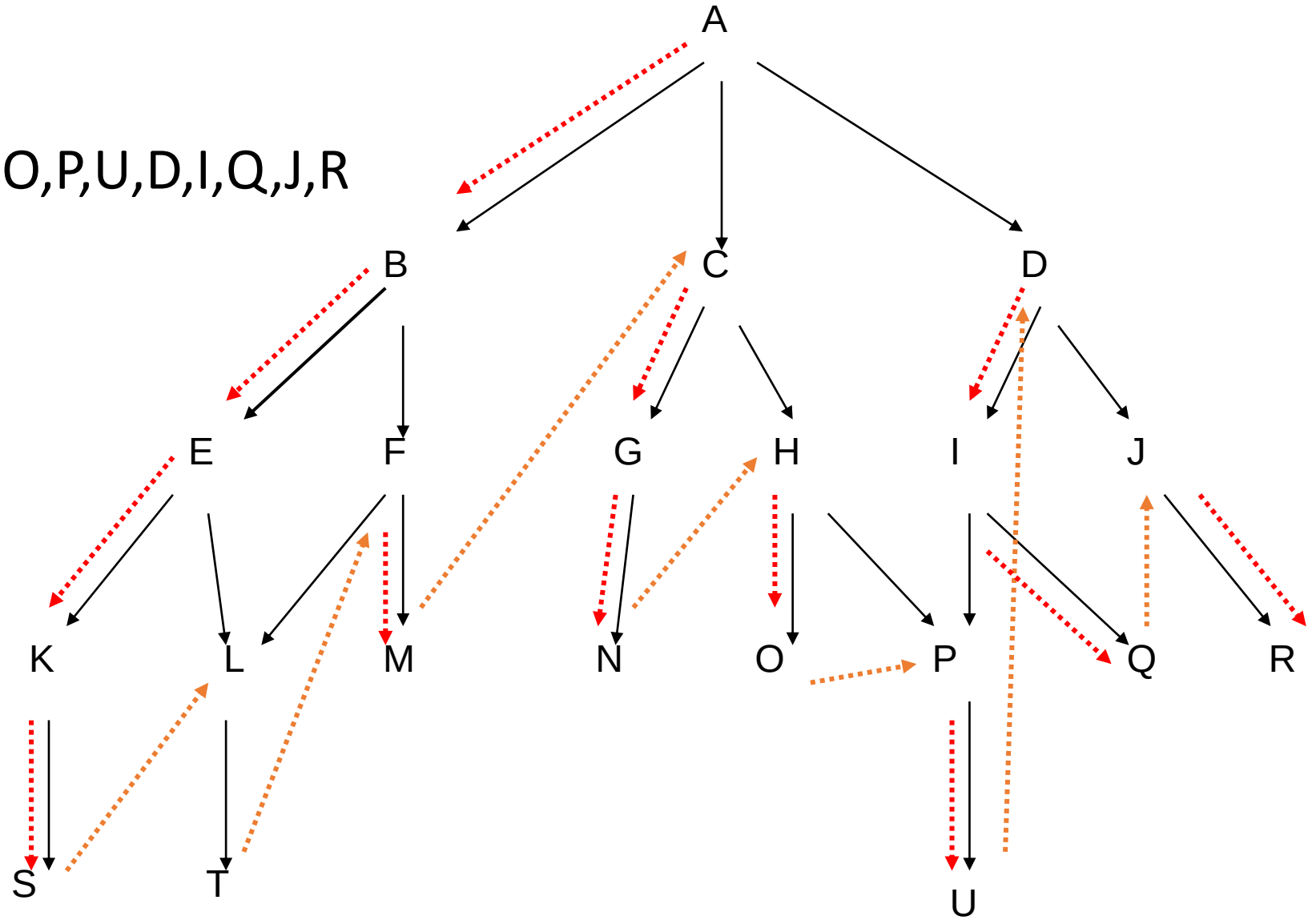
- Derinlik Öncelikli Aramada, bir durum(düğüm) incelendiğinde, yan duruma geçmeden önce incelenen düğümün en uç düğümüne kadar inilir.
- Derinine Öncelikli Arama, mümkün olan en derine kadar gider.
- Artık gidecek alt düğüm olmadığında, yan düğüme dallanılır.

Depth-first search

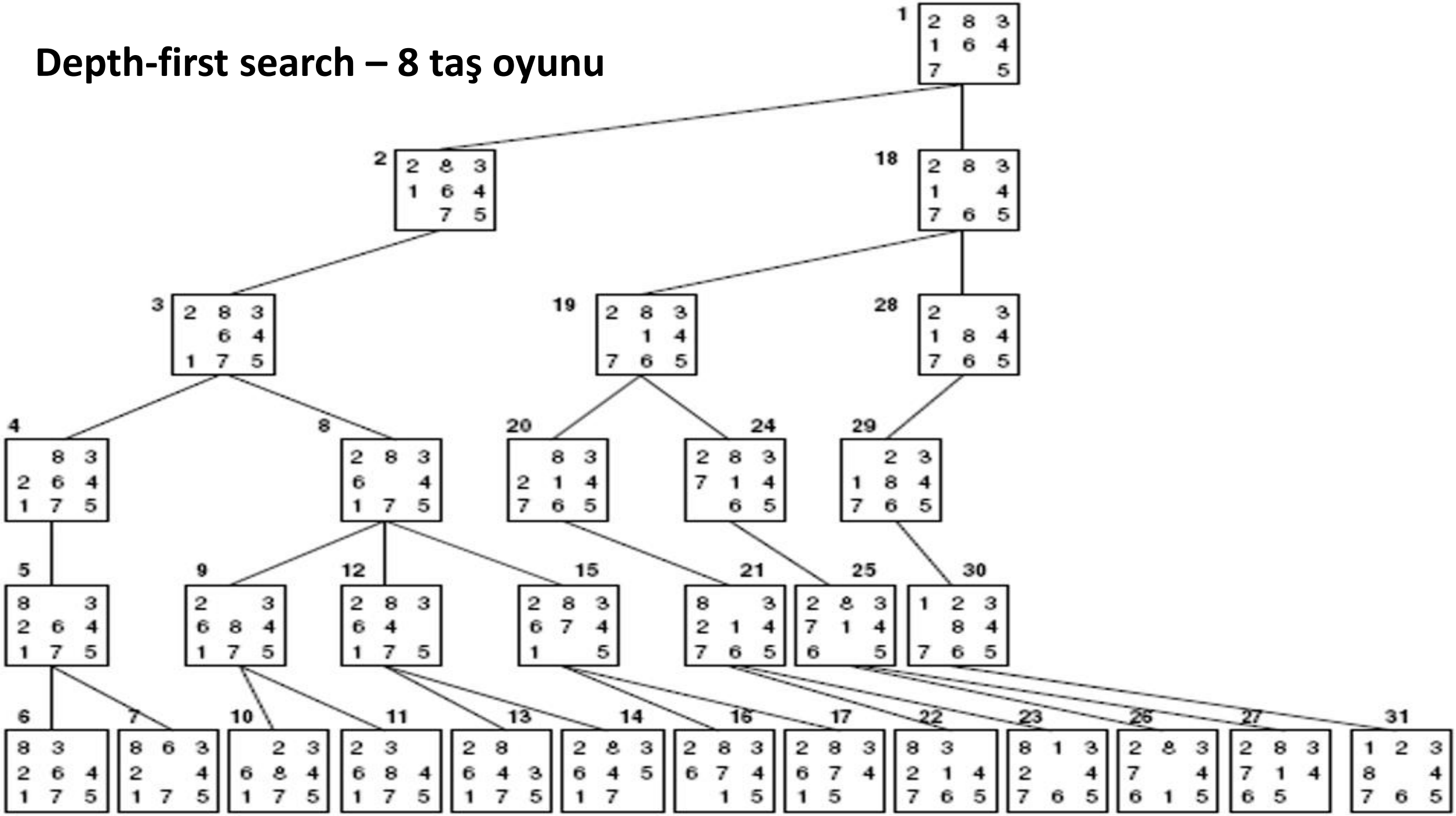


Depth-first search

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



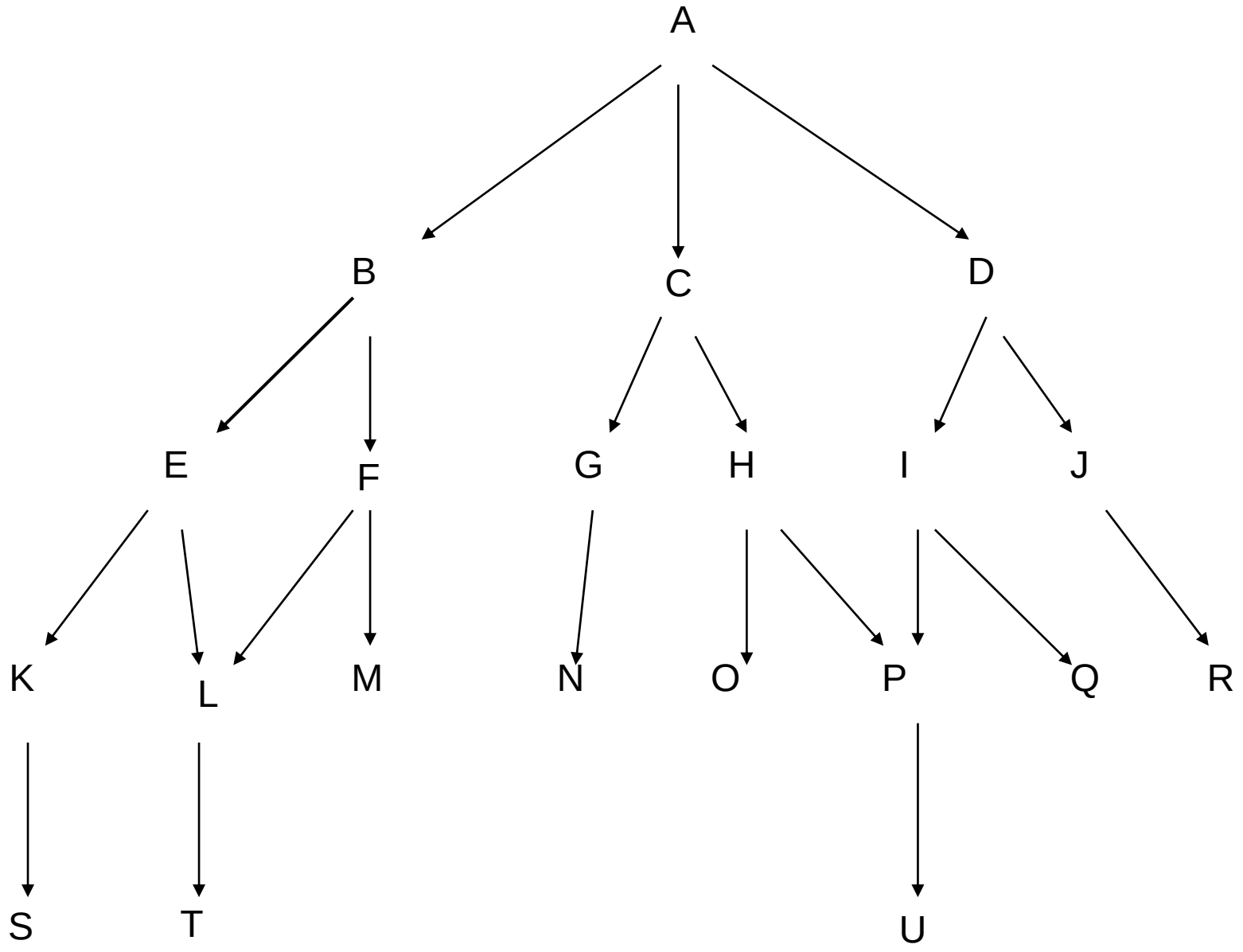
Depth-first search – algoritma karmaşıklığı

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

Breadth-first search - genişlik öncelikli arama

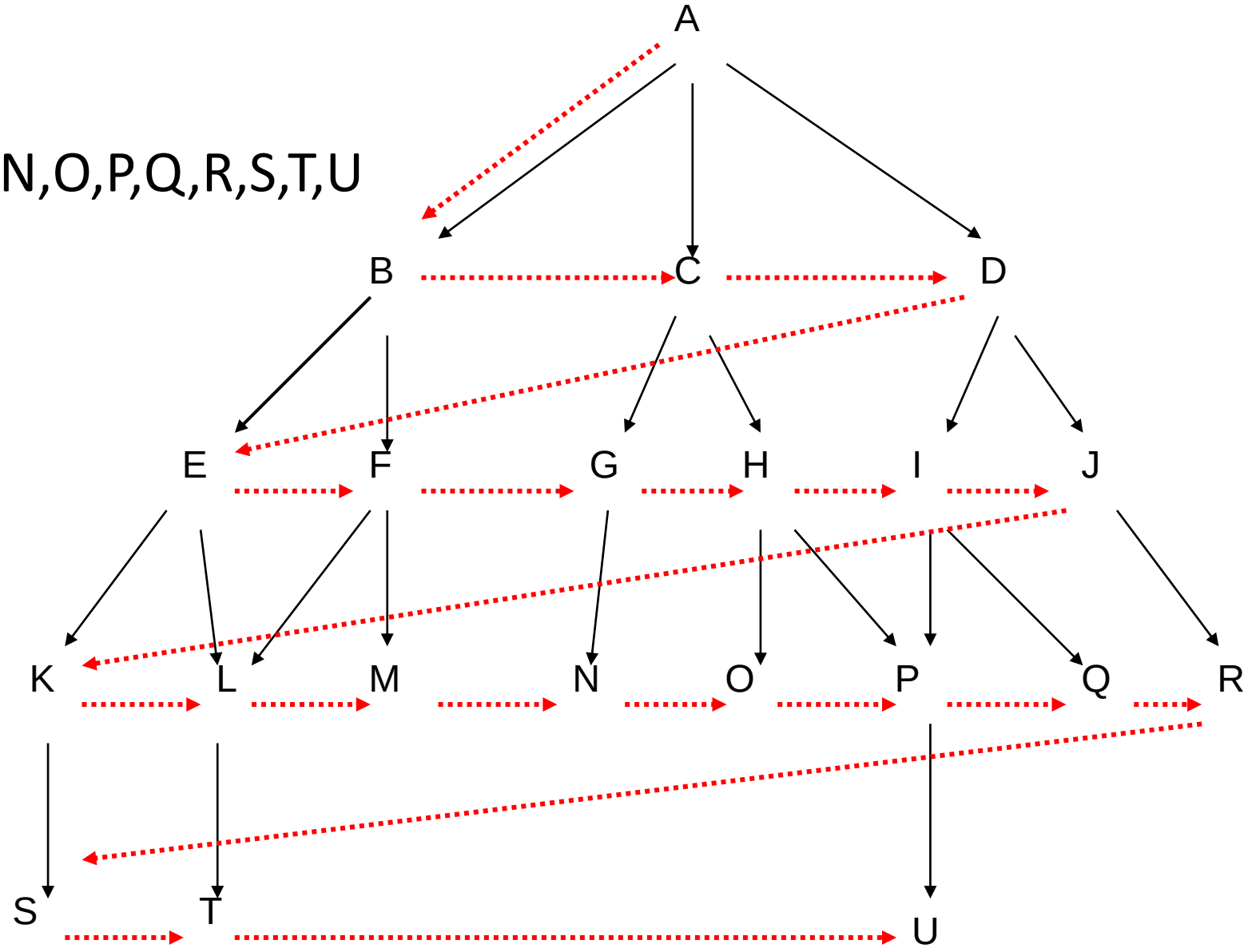
- Genişlik Öncelikli Arama, mümkün olan ilgili seviyedeki tüm düğümleri dolaşır.
- Artık gidecek düğüm olmadığında, alt düğüme dallanılır.

Breadth-first search

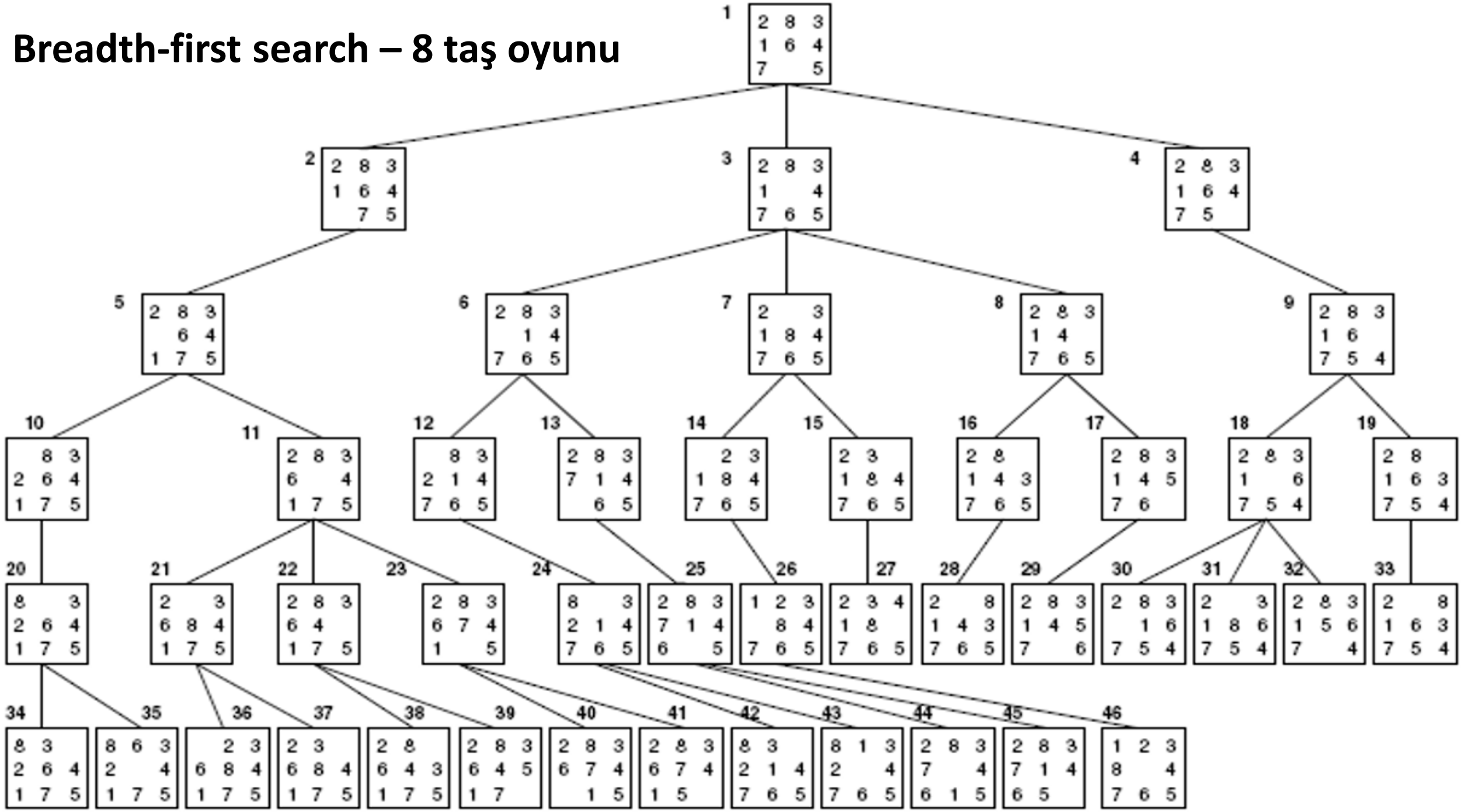


Breadth-first search

- 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+…+ 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

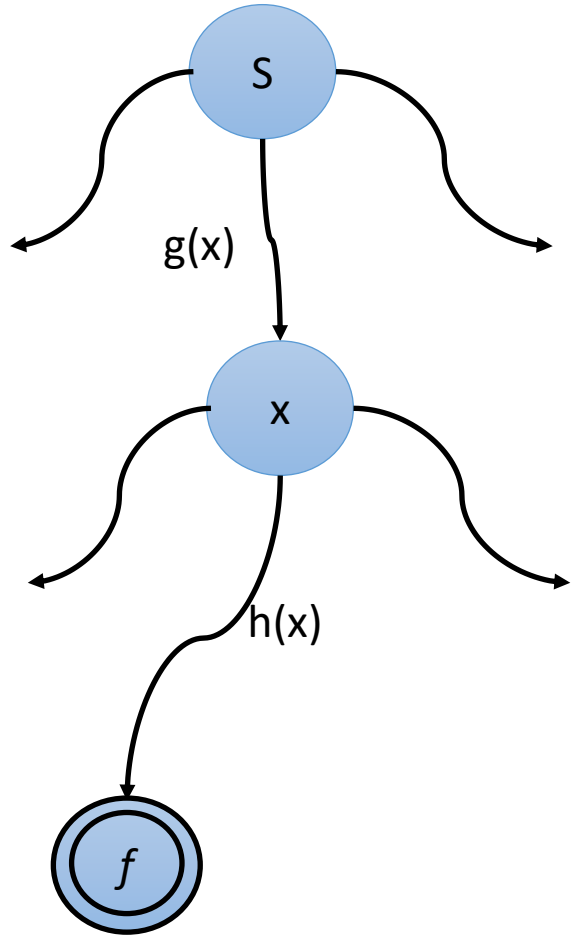
- Durum uzayında çözüme giden düğümlerin seçilmesini sağlayan bir *f* fonksiyonu olduğunu düşünelim.
- Herhangi bir *x* düğümü için fonksiyonun değeri *f(x)* olsun.
- Bu fonksiyon çözüme giden yollar içinde minimum olanının bulunmasına yardımcı olmaktadır.
- Açılma sırası gelen düğümler *f* fonksiyonunun 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 ulaşılamamışsa 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.

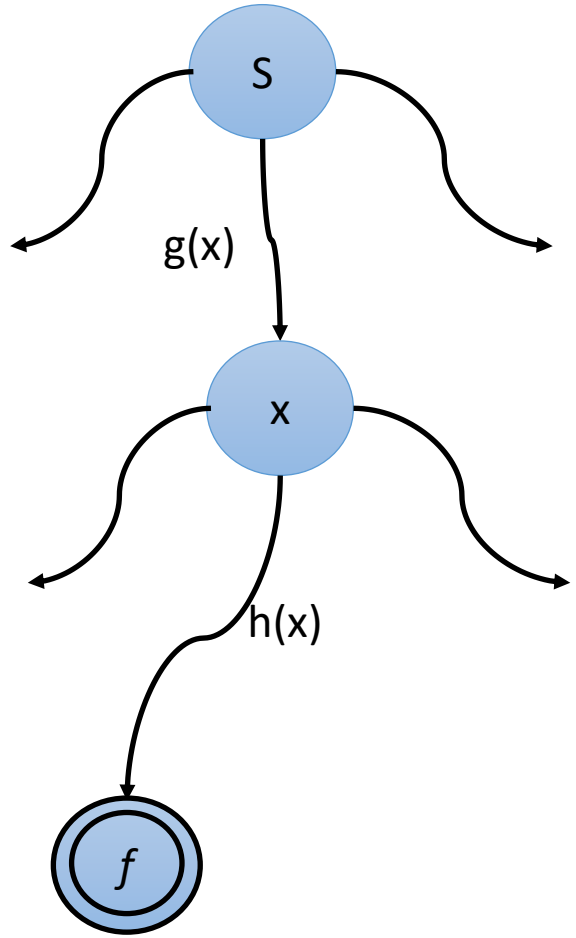
A* algoritması - değer fonksiyonu

- $f(x) = g(x) + h(x)$
- *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.



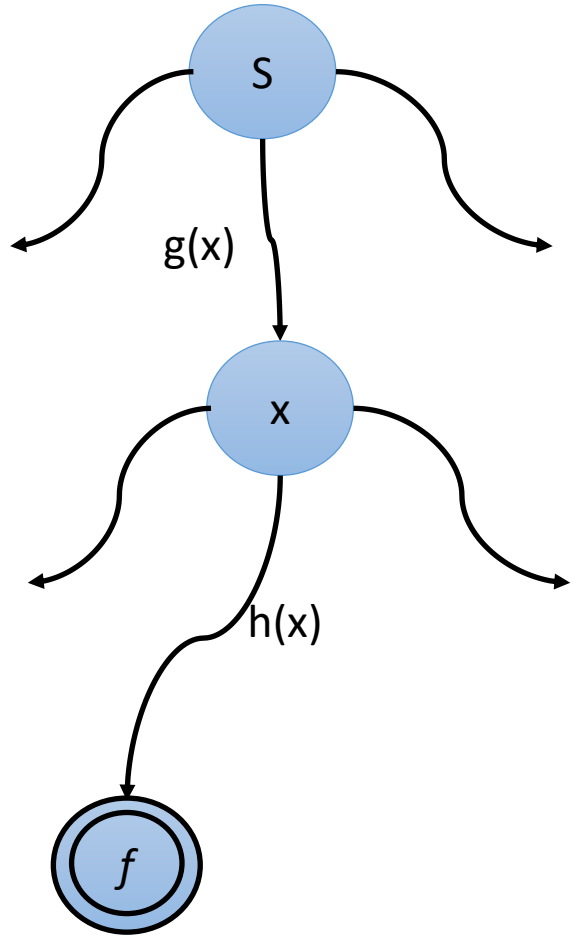
A* algoritması - değer fonksiyonu

- *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

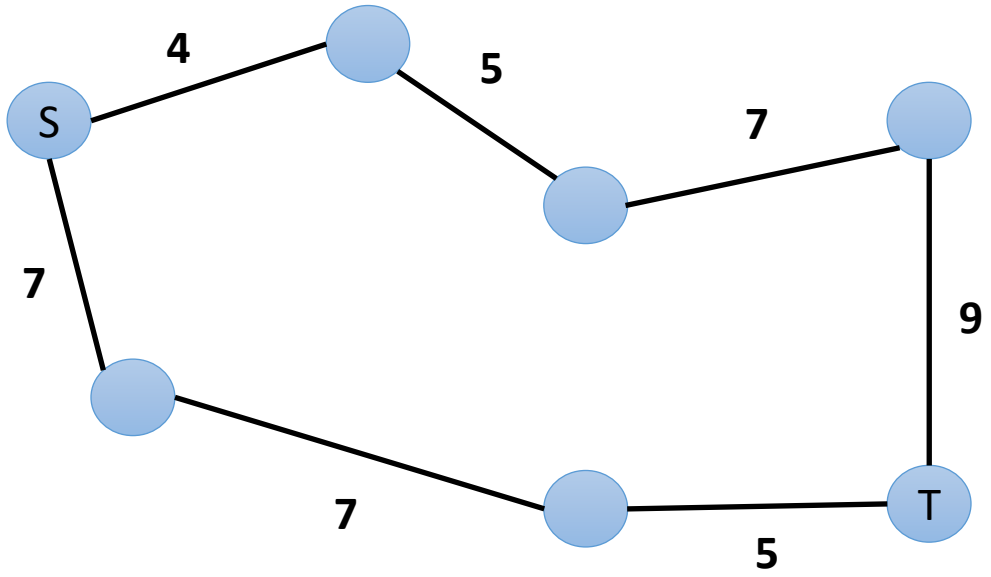
- Fonksiyonun son hali aşağıdaki şekilde tanımlanmaktadır.
- $f(x) = \alpha\, g(x) + (1- \alpha)\, h(x),\ \ 0 <= \alpha <= 1$
- α' nın büyük değerleri için daha çok enine, küçük değerleri için daha çok derinine arama yapılır.
- α değeri problemin belirli aşamalarında aramayı yönlendirmek için kullanılabilir.



A* algoritması – örnek

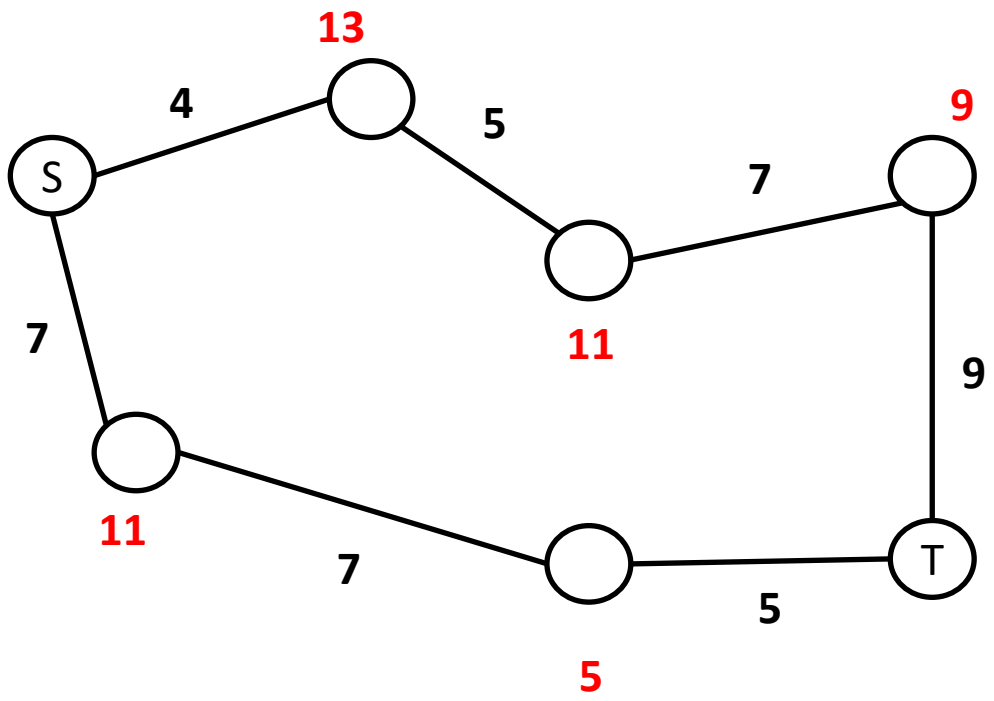
- S: başlangıç düğümü
- T: hedef düğüm
- Maliyeti en düşük yol?

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



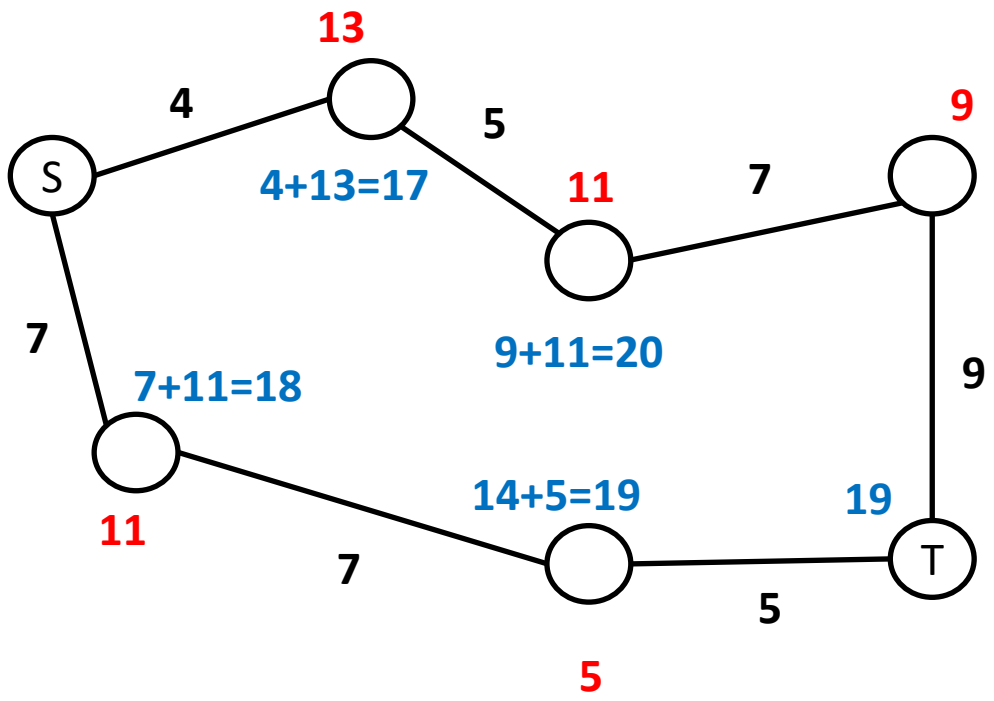
A* algoritması – örnek

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

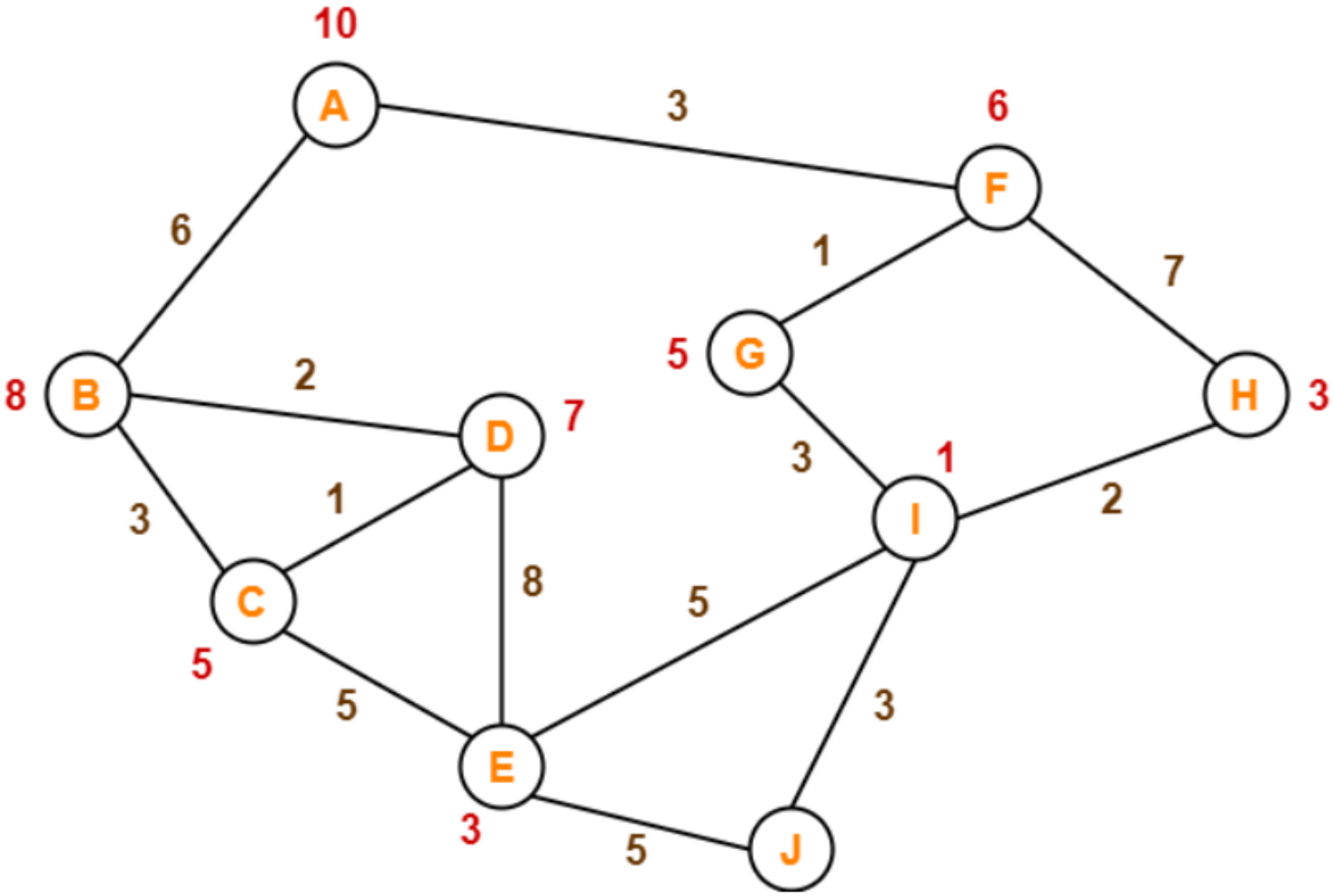


A* algoritması – örnek

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



Alıştırma:



https://www.gatevidyalay.com/a-algorithm-a-algorithm-example-in-ai/