Arama ile problem çözme

Konular

- Arama sorununu çözen vekil
- Sorunun ifade edilmesi
- Sorun örnekleri
- Arama ağacı

Amaca Yönelik Vekillerin Oluşturulması

- Vekilin ulaşmak istediği bir **amaç** var:
  - Arabayı A noktasından B noktasına sürmek
  - 8 veziri satranç tahtası üzerinde biri-birine hamle etmeyecek şekilde yerleştirmek
- Vekilin başlangıç durum, yani amaca doğru nereden yola çıkacağı hakkında bilgisi var.
- Bu başlangıç durumdan çıka bilmesi için vekilin yapa bileceği **hareketler** kümesi var.
- **Hedef**: vekili başlangıç noktadan amaca götüren mümkün hareketler ardışıklığının bulunması

Sorun çözen vekiller-Problem-solving agents

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  static: seq, an action sequence, initially empty
          state, some description of the current world state
          goal, a goal, initially null
          problem, a problem formulation
  state ← UPDATE-STATE(state, percept)
  if seq is empty then do
      goal ← FORMULATE-GOAL(state)
      problem ← FORMULATE-PROBLEM(state, goal)
      seq ← SEARCH(problem)
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

Arama sorunu -Yönbulma

Sorun: Çanakkale'den Ankara'ya varmak

Sorunun ifadesi:

Amaç
- Ankara'ya varmak

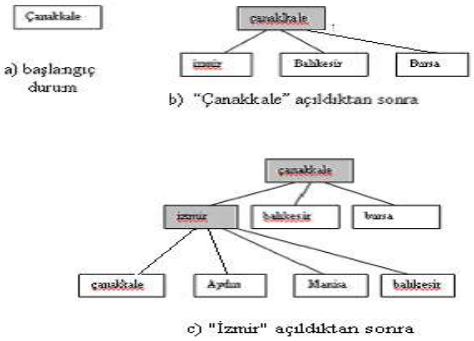
Başlangıç durum
- Çanakkale'de bulunmak

durumlar
- Vekilin bulunduğu şehirler

hareketler
- bir şehirden diğerine hareket

Çözüm
- Çanakkale'den Ankara'ya götürecek şehirler ardışıklığı, örnek: Çanakkale,Bursa,Eskişehir,Ankara

Arama sorunu-Yönbulma (devamı)



sorunun ifade edilmesi

Sorun aşağıdaki kavramlarla tanımlanabilir:

1. Başlangıç durum , "Çanakkale"
2. Hareketler veya ardıl fonksiyonu *S(x)* = hareketler kümesi–durum çiftleri

   S(Çanakkale) = { *<Çanakkale → Bursa, Bursa>, …* }

3. Amaç denemesi

   açık, *x* = "Ankara"

   imalı, mat*(x)*
4. Yol değeri

   gidilen yol, düğümler sayısı, gerçekleştirilen hareketler sayısı…

Çözüm, başlangıç durumdan amaç durumuna götüren hareketler ardışıklığıdır

Durumun tasviri

- Her bir zaman anındaki ortam **durumla** ifade ediliyor.
  - Başlangıç durum –sorunun çözümü için yapılacak ilk hareketin başlandığı durum
  - Hareket, güncel durumu diğer bir durumla değiştiriyor; buna **geçiş durumu** denir.
    - Her bir verilmiş durum için birkaç uygulanabilir hareket ola bilir
  - **Amaç durumu-** sorunun tanımında verilmiş ulaşılması gereken durum-sorunun çözümü
  - **başarısız durum-** hiçbir hareketin uygulana bilemediği ve amaç olmayan durum

Durumun tasviri (devamı)

- **Durum uzayı**:
  - Başlangıç durumdan ve bu durumdan başlayarak hareketler ardışıklığı ile ulaşılabilecek tüm durumlardan oluşan küme
  - Durum uzayı graflarla ifade edilebilir:
    düğümler: uzaydaki durumlar
    kenarlar: hareketler/işlemler
- **Sorunun boyutu** genelde **mümkün durumlar sayısı** ile (veya durum uzayının boyutu ile) ifade ediliyor.
  - Tic-Tac-Toe oyununda yaklaşık 3^9 durum vardır
  - Damada (Checkers) yaklaşık 10^40 durum vardır.
  - Rubik' Cube'da durumlar sayısı 10^19 civarındadır.
  - Satrançta durumlar sayısı yaklaşık 10^120 dir.
  - Go oyununda durumlar sayısı ise çok daha fazladır

Durumlara ait bilgilerin ifade edilmesi

- Durumda nasıl bilgiler olmalıdır:
  - Yamyamlar ve Yolcular sorununun çözümü için kayığın hangi renkte olmasının bir önemi var mı? Borsada hisse senetlerinin değerlendirilmesinde güneşteki lekeler ne kadar etkilidir? Hangi bilgilerin ifade edilmesi sistem tasarımcısının sorumluluğundadır.
- **Ortamın soyutlanma seviyesi** nasıl olmalıdır veya ortam hangi ayrıntılarına dek ifade edilmelidir?
  - Çok ayrıntılarına gidersek "ağaçlar arasında ormanı kaybederiz". Çok kabaca tasvirle ise sorunun çözümü için gereken ayrıntıları gözden kaçırmış ola biliriz
- Durumların sayısı seçtiğimiz ifade yöntemine ve soyutlama seviyesine bağlıdır.

Ulaşılacak amacın tasviri

- Ulaşmak istediğimiz durum, elde etmek istediğimiz özellikler kümesi ile ifade edile bilmelidir
- Amacımızın sağlanıp sağlanmadığını belirlemek için amaç denemesi yapılmalıdır

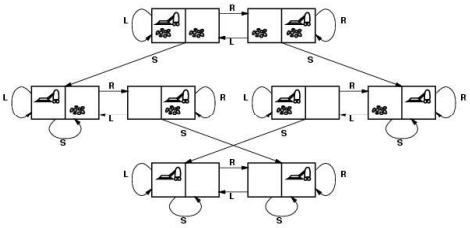
Hareketler

- Hareketlere, her birisi bir zaman diliminde oluşan kesintili olaylar gibi bakmak mümkündür
  - Ortam her hangi bir durumun içindedir; bir hareket gerçekleşir ve ortam yeni duruma geçiyor. Örneğin, "Ali sınıftadır" ve "Eve gitti" hareketleri ardışık olarak gerçekleşmişse sonraki durum "Ali evdedir" olacaktır. (Burada Ali'nin sınıfta ve evde olduğu andaki durumlarından farklı durumlara (örn., Ali eve gidiyor" durumuna) bakmıyoruz)

Hareketler (devamı)

- Meselenin çözümü boyunca tüm değişmeleri ifade etmek için uygun her bir hareket ve olay değerlendirilmelidir
- Önceki hareket verilmişse ve ortamın güncel durumu belli ise sonraki hareket belirlenebilir
  - **Önkoşul:** hareket şu anki ortama uygulanabilir ve yasaldır
  - **Etki:** şu anki ortamda hareket yapıldıktan sonra ortamın kesin durumu nasıl olacak

Süpürge Vekilin Durum uzayı grafı



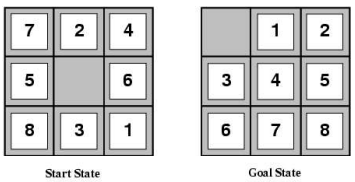
- durumlar? toz ve robotun yeri
- hareketler? *Sol, Sağ, Temizle*
- Amaç denemesi? hiç bir karede toz yoktur
- Yol değeri? Her hareketin değerinin 1 olduğunu kabul ediyoruz

Bazı örnek sorunlar

- Oyun sorunları
  - 8-taş
  - Yolcular ve yamyamlar
  - Kriptoaritmetik
  - 5 kibrit çöpü
  - Gezgin satıcı sorunu (TSP)
- Gerçek dünya sorunları
- http://www.permadi.com/fpcgi/

Örnek: 8-Taş

8 numaralanmış taş 3 x 3 tahtasında rasgele dizilmiştir. Taşları öyle hareket ettirmeli ki, onların arzu olunan amaç dizilişine ulaşılmış olsun



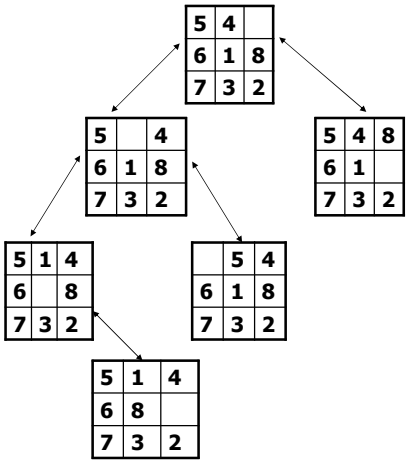
8 taş sorunu (devamı)

- **Durum:** taşların tahta üzerinde 3 x 3 dizi yapısı
- **Hareketler:** BoşYeri Sola,Sağa,Yukarı,Aşağı hareket ettirmek.
  - Her bir 8 taşın 4 mümkün hareketini tasvir etmektense

  "Boş yer"in hareket ettirilmesi işlemlerin tasviri için daha etkili yoldur.
- **Başlangıç durum:** Tahta üzerinde her hangi bir durum
- **Amaç:** Taşların tahta üzerinde arzu olunan dizilişi

- Çözümü O(2^k) adım gerektiriyor; k-çözüm yolunun uzunluğudur.
- 15-taş sorunu (15 taşlı 4 x 4 çerçeve) ve N-taş sorunu (N = n^2-1)

8 taş sorununun durum uzayından bir kesinti

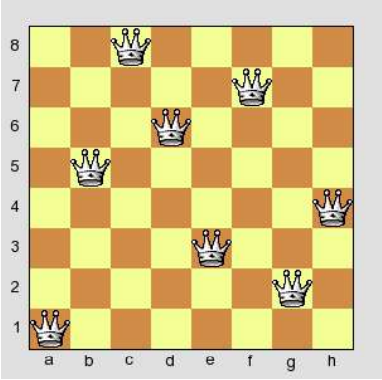


8-vezir sorunu

8 veziri satranç tahtasında öyle yerleştirmeli ki, vezirler birbirine hamle etmesin

Toplam durum sayısı: 4x10^9

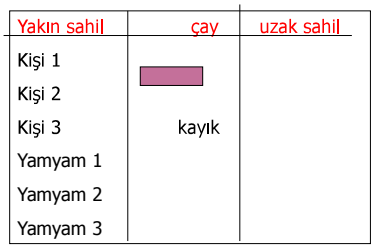
Toplam çözüm sayısı: 12



Yolcular ve Yamyamlar

3 yolcu ve 3 yamyam kayıkla nehrin bir sahilinden karşı sahile geçmek istiyor. Kayığa en fazla 2 kişi binebilir.

- **Amaç**: Tüm yamyamların ve yolcuların nehri geçmesi.
- **Sınırlama:** Yamyamların sayısı çayın her hangi sahilinde yolculardan çok olursa yamyamlar yolcuları yiyecektir
- **Durum:** Nehrin her iki sahilinde ve kayıktaki yamyam ve yolcular
- **Hareketler/İşlemler:** Her iki yönde içinde bir, veya iki kişi ile kayığın hareketi



Bu sorun 11 harekette çözülebilir.

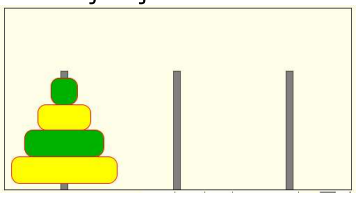
Yamyamlar ve Yolcular Sorunun Çözümü

		Yakın sahil		*Karşı sahil*
0	Başlangıç durum:	MMMCCC	B	–
1	2 yamyam çayı geçti:	MMMC	B	CC
2	Birisi geri döndü:	MMMCC	B	C
3	2 yamyam çayı geçti:	MMM	B	CCC
4	Biri geri döndü:	MMMC	B	CC
5	2 yolcu çayı geçti:	MC	B	MMCC
6	Bir yolcu ve bir yamyam geri döndü:	MMCC	B	MC
7	İki yolcu çayı geçti:	CC	B	MMMC
8	Bir yamyam geri döndü:	CCC	B	MMM
9	İki yamyam çayı geçti:	C	B	MMMCC
10	Bir yamyam geri döndü:	CC	B	MMMC
11	İki yamyam çayı geçti:	–	B	MMMCCC

M- yolcu, C -yamyam

Hanoi kulesi

- Hanoi kuleleri, bir matematik oyunu veya bulmacadır. Üç direk ve farklı boyutlarda disklerden oluşur. Bu diskleri dilediğiniz direğe aktarabilirsiniz. Bulmaca bir direkte en küçük disk yukarıda olacak şekilde, küçükten büyüğe direk üstünde dizilmiş olarak başlar. Böylece konik bir şekil oluşmuş olur.

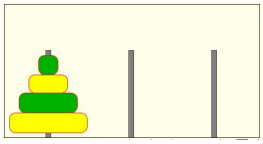


- http://www.cut-the-knot.org/Curriculum/Combinatorics/TowerOfHanoi.shtml

Hanoi kulesi

Kurallar

- Her hamlede sadece bir disk taşınabilir.
- Her hamle en üstteki diski direkten alıp diğer bir direğe taşımaktan oluşur. Diğer direkte daha önceden diskler olabilir.
- Hiçbir disk kendisinden küçük bir diskin üzerine koyulamaz.



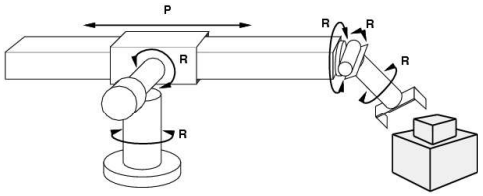
Şifreli hesap-Cryptarithmetic

- Harflerin yerine (0, ..., 9) rakamlarını öyle koymalı ki ifade doğru olsun. Örn: SEND + MORE = MONEY

```
 FORTY      Çözüm:    29786
+  TEN                  850
+  TEN                  850
 -----                -----
 SIXTY                31486
F=2, O=9, R=7, etc.
```

- Bu sorunda çözüm her bir harfe uygun sayı atamasının yapılmış olduğu amaç durumun bulunmasıdır

Örnek: parçaların yığılması

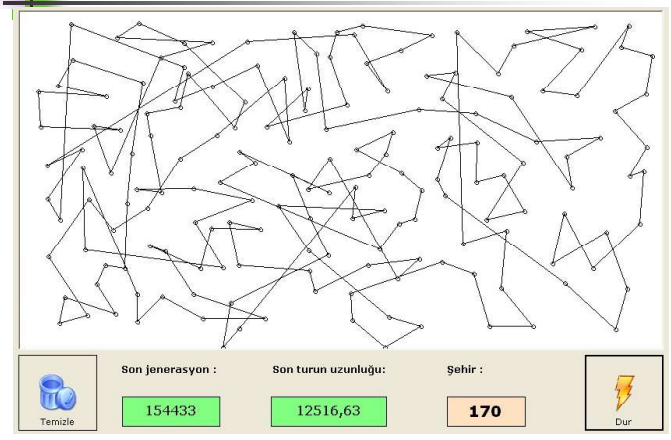


- durumlar?: robot kolunun koordinatlarının yığılacak nesnenin parçasının köşelerine uygunlaştırılması
- hareketler?: robot kolunun hareketi
- Amaç denemesi?: yığımın bitmesi
- Yol değeri?: yığım için gereken zaman

Gezgin satıcı sorunu

- N şehir arasında yol haritası verilmiştir. Her şehirde bir kez bulunmakla tüm şehirlerden geçerek başlangıç şehrine dönüşü gerçekleştiren en kısa yolu bulmalı (*Hamiltonian devresi*)
- Geometrik ifadesi:
  - N düğümlü tam graf.
  - n!/2n mümkün yol
  - Mümkün yollar içinden en kısasının bulunması

Gezgin satıcı sorunu



Durum Uzayında Arama

Durum Uzayında aramanın formal ifadesi

- Durum uzayı -(V, E) grafıdır. V düğümler, E kenarlar kümesidir. Kenar bir düğümden diğerine yönelmiştir.
- **düğüm:** durumu ifade eder
  - Düğümün babası ile ilgili bilgileri, baba düğümden bu düğüme geçmek için gereken işlem hakkında bilgileri ve diğer istatistiksel bilgileri içerir
- **kenar:** uygulanabilir hareketi/işlemi ifade eder
  - Kaynak ve hedef düğümler uygun olarak baba ve oğul düğümlerdir.
  - Her bir kenarın sabit, eksi olmayan değeri vardır. Bu değer hareketin değerini ifade eder

Durum Uzayında aramanın formal ifadesi (devamı)

- Düğümün üretilmesi- önceden belirlenmiş (genişlenmiş) bir düğüm üzerinde işlem yapmakla diğer bir düğümün belirlenmesi
- Düğümün genişlenmesi: belirlenmiş düğüm üzerinde tüm uygulanabilir hareketleri uygulamakla tüm oğul düğümlerin üretilmesi
- Bir veya birkaç düğüm başlangıç düğüm olarak kabul ediliyor
- Amaç denemesi -üzerinde işlem yapılan düğümün durumunun amaç durumu olup-olmadığının belirlenmesi
- Çözüm- başlangıç durumdan amaç duruma doğru yolda yapılan işlemler ardışıklığı
- Çözümün değeri- çözüm yolundaki kenarların değerlerinin toplamı

Durum Uzayında aramanın formal ifadesi (devamı)

- Durum_Uzayında_ Arama- genişlenmemiş durum uzayı grafının, amaç düğümü de içine alan bir kısmının genişlenmesi yolu ile çözümün aranması.
  - Başlangıçta V={S}, S başlangıç düğümdür, S genişledikçe onun ardılları üretiliyor ve bu düğümler V'ye, uygun kenarlar ise E'ye ilave ediliyor. Bu süreç amaç düğüm üretilene (V'ye ilave etme) ve tanımlanana (amaç denemesi) dek devam ediyor
- Arama sürecinde düğüm üç halden birinde olabilir:
  - Henüz üretilmemiş
  - AÇIK: üretilip,ama genişlenmemiş
  - SON: genişlenmiş
  - Aramanın her adımında genişleme için AÇIK düğümün seçilmesi yoluna göre farklı arama stratejileri bulunmaktadır

Durum_Uzayında_Arama Algoritması (devamı)

- Arama sürecinde arama ağacı oluşturuluyor:
  - Ağacın kökü S başlangıç durumudur
  - Yaprak düğümler
    - Henüz genişlenmemişler (OPEN listesinde değiller) veya
    - Ardılları yoktur
- Arama ağacı döngülerden dolayı, hatta durum uzayı küçük olsa bile sonsuz olabilir
- Her düğüm başlangıç düğümden verilmiş düğüme dek kısmı çözüm yolunu (ve bu yolun değerini) ifade ediyor.

Durum_Uzayında_Arama Algoritması

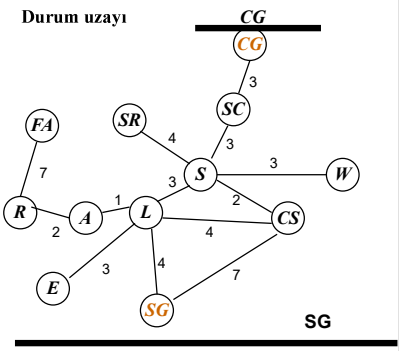
```
open := {S}; closed :={}; /* S- başlangıç durum
repeat
   n := select(open);        /* Açık listesinden açılmak için bir
  düğüm seçmeli  */
  if n amaç düğüm ise
       then exit başarılı sonuç
  expand(n)
          /* n'in tüm oğullarını üretmeli
             yeni üretilmiş düğümleri açık listesine eklemeli
  (tekrarlamaları kontrol etmeli)
           put n'i kapalı listesine eklemeli (tekrarlamaları yoklamalı) */
until open = {};
exit başarısız sonuç
```

Kapalı Dünya yaklaşımı-Closed World Assumption

- Sorun alanı hakkında tüm gereken bilgiler her algılamada erişilebilendir, başka deyişle her bir durum dünyanın tam tasviridir.
- Zamanın hiçbir anında tam olmayan bilgi yoktur.

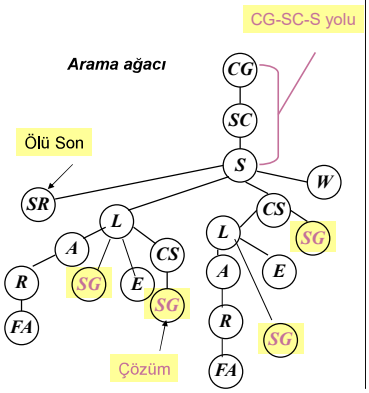
Arama Ağaçları-örnek

- Örnek haritada CG'den SG'ye ulaşılabilecek tüm yollar gösteriliyor. Kenarların üzerinde uygun 2 kent arasındaki yol uzunluğu (yol değeri) gösterilmiştir.



Arama Ağaçları-örnek (devamı)

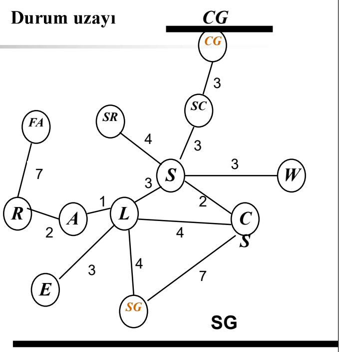
- Eğer biz CG'den çıkan tüm mümkün yolları bulmaya çalışsak, kesin olarak SG'ye giden yolu da bulmuş olacağız. Çünkü böyle bir yol mevcuttur.
- Burada harita grafla,yani arama ağacı ile ifade edilmiştir. Bu ağacın düğümleri arasındaki kenarlar bir yolu (durumu değil) ifade etmektedir.
- Arama yöntemleri, bu ağacın boyutu hakkında hiçbir bilgi olmadan çalışmaya başlar. Önemli olan, seçilen yöntemle daha az yol sayısı bulunan ağacı açmış olsun

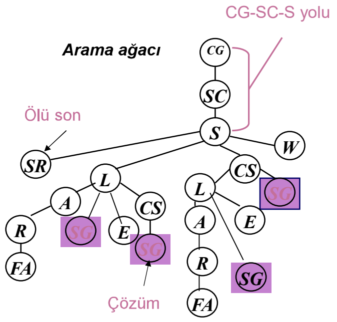


Durum uzayı ve Arama Ağaçları

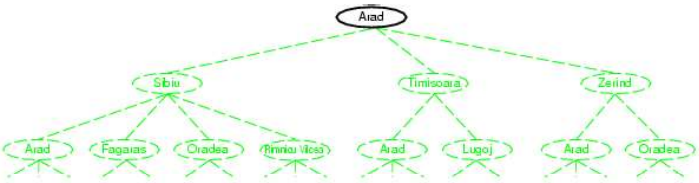
Arama ağacı ve durum uzayı farklı kavramlardır:

Durum uzayında yalnız 12 düğüm vardır. (her kent için bir düğüm). Ama arama ağacında sonsuz sayıda düğüm olabiliyor. (eğer döngülere izin veriliyorsa). Hatta döngüleri gözden çıkarsak dahi ağaçta 20 düğüm olduğunu görürüz.

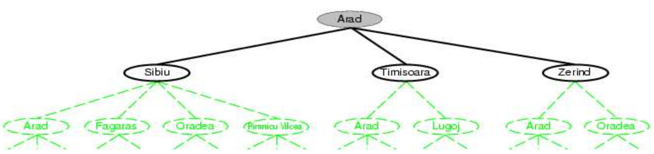




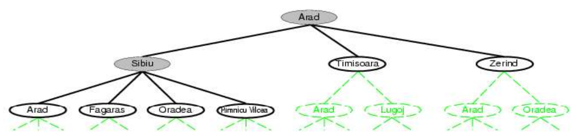
Ağaç üzere arama örneği



Ağaç üzere arama örneği



Ağaç üzere arama örneği



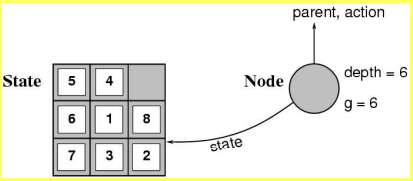
Ağaç üzere arama algoritması

```
function TREE-SEARCH( problem, fringe) returns a solution, or failure
    fringe ← INSERT(MAKE-NODE(INITIAL-STATE[problem]), fringe)
    loop do
        if fringe is empty then return failure
        node ← REMOVE-FRONT(fringe)
        if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
        fringe ← INSERTALL(EXPAND(node, problem), fringe)

function EXPAND( node, problem) returns a set of nodes
    successors ← the empty set
    for each action, result in SUCCESSOR-FN[problem](STATE[node]) do
        s ← a new NODE
        PARENT-NODE[s] ← node;  ACTION[s] ← action;  STATE[s] ← result
        PATH-COST[s] ← PATH-COST[node] + STEP-COST(node, action, s)
        DEPTH[s] ← DEPTH[node] + 1
        add s to successors
    return successors
```

Durum ve düğümün karşılaştırılması

- Durum fiziki bir yapının tasviridir
- Düğüm arama ağacının bir kısmını oluşturan veri yapısıdır ve durum, baba düğüm, hareket, yol değeri *g(x)*, derinlik kavramlarını içeriyor



Arama yöntemleri

- Arama yöntemi, düğümlerin genişleme sırasını belirler
- Yöntemleri, aşağıdaki parametreleri ile nitelendirmek mümkündür:
  - Tamlık-completeness: çözüm (eğer varsa) her zaman bulunabilir mi?
  - Zaman karmaşıklığı-time complexity: incelenen düğümler sayısı
  - Mekan karmaşıklığı-space complexity: bellekteki düğümler sayısı
  - Optimallık-optimality: en az maliyetli çözüm her zaman bulunur mu?
- Zaman ve mekan karmaşıklığı aşağıdaki kavramlarla ölçülüyor:
  - *b:* arama ağacında en fazla dallanma etkeni
  - *d: en az maliyetli çözümün derinliği*
  - *m*: durum uzayının en fazla derinliği (∞ olması mümkündür)

Zaman ve mekan karmaşıklığı

- **Aramanın zaman ve mekan karmaşıklıkları ağacın, incelenen ve bellekte yer tutan düğümlerinin sayısı ile bağlıdır.**
- **Karmaşıklığın değerlendirilmesinde ağacın dallanma etkeni ("b") ve ağacın azami derinliği ("d") etkeni kullanılıyor. Eğer dallanma etkeni b ise bu, " uç olmayan (yaprak olmayan) düğümün b oğlu var" anlamını veriyor.**