Arama yöntemleri

- Bilgiye dayanmayan arama (Blind Strategies)
- Sezgisel arama (Heuristic Strategies)

Bilgisiz veya
KÖR ARAMA- Blind search

Bilgisiz (kör) arama yöntemleri

- Bilgisiz arama yöntemlerinde yalnız sorunun tanımında bulunan bilgiler kullanılabilir
  - Enine arama- Breadth-first search
  - Derinine arama- Depth-first search
  - Sınırlı derinine arama- Depth-limited search
  - Yinelemeli derinine arama- Iterative deepening search
  - Sabit maliyet - Uniform-Cost (UCS)
  - İkiyönlü arama - Bi-directional search

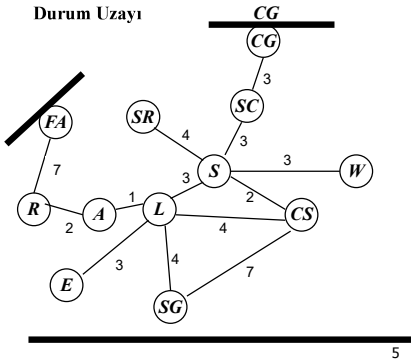
Kör Arama

- Kör arama yöntemlerinde çözüme ulaşmak için hiçbir bilgi verilmez.
- Aramanın her hangi bir adımında çözüme ne kadar yakın (veya uzak) olduğumuz hakkında veya çözümün nasıl bulunabileceği hakkında fikir söylemek mümkün değildir.

Kör arama

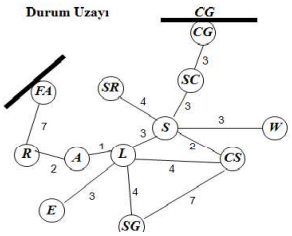
```
başlangıç durumu içeren düğümü (ağacın kökü) seçmeli

listedeki ilk yol amaç durumunda sonlanana dek veya liste boş olana dek aşağıdakileri yapmalı:

    listeden ilk yolu almalı

    İlk yolu, onun uç düğümünün tüm ardıllarına doğru genişletmekle yeni yollar oluşturmalı

    Döngülü tüm yolları gözden çıkarmalı

    Amaç durumu bulunursa aramayı bitirmeli, aksi halde yeni yolları gözden geçirmeli

Tüm yollar gözden geçirildikten sonra amaç durumu bulunamazsa aramayı başarısız kabul etmeli
```

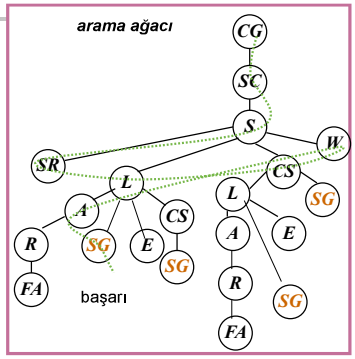
Yol haritası



Enine arama - Breadth-first search

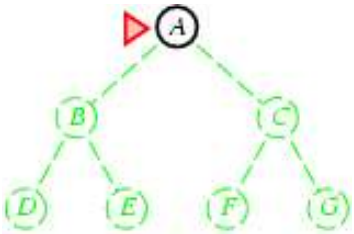
- Enine aramada ağaç soldan sağa, yukarıdan aşağıya doğru taranıyor. Başka bir deyişle, bir seviyedeki tüm düğümler tarandıktan sonra bir sonraki aşağı seviyeye geçilir.





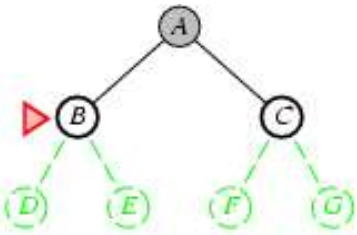
Enine arama - Breadth-first search

- En yüzeyde (en üst seviye) olan genişletilmemiş düğümü genişletmeli
- FIFO yapısı: yeni ardıllar sona eklenecek



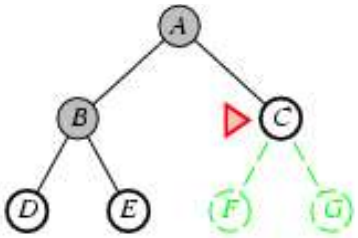
Enine arama - Breadth-first search

- En yüzeyde (en üst seviye) olan genişletilmemiş düğümü genişletmeli
- FIFO yapısı: yeni ardıllar sona eklenecek



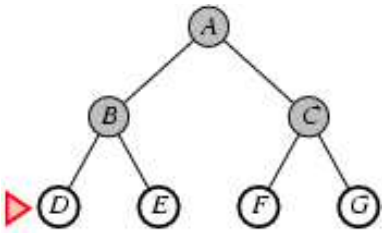
Enine arama - Breadth-first search

- En yüzeyde (en üst seviye) olan genişletilmemiş düğümü genişletmeli
  - FIFO yapısı: yeni ardıllar sona eklenecek



Enine arama - Breadth-first search

- En yüzeyde (en üst seviye) olan genişletilmemiş düğümü genişletmeli
  - FIFO yapısı: yeni ardıllar sona eklenecek



Enine aramanın özellikleri

- Tam? evet (*b* sonlu ise)
- Zaman? $1+b+b^2+b^3+\ldots+b^d$ $O(b^d)$
- Mekan? $O(b^d)$ (her bir düğüm bellekte tutuluyor)
- Optimal? Evet (eğer her adım için değer = 1 ise)
- Mekan sorunu çok önemlidir

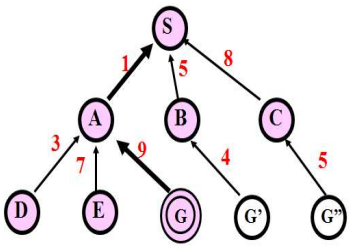
b = dallanma etkeni
d= azami derinlik

Enine arama - Breadth-first search

Gen. düğüm

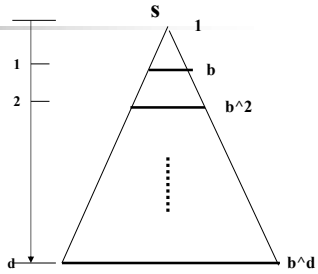
	{ S }
S	{ A B C }
A	{ B C D E G }
B	{ C D E G G' }
C	{ D E G G' G" }
D	{ E G G' G" }
E	{ G G' G" }
G	{ G' G" }

Çözüm yolu- S A G <-- G'nin değeri 10
Genişletilen düğüm sayısı
(amaç düğümle birlikte) = 7



Enine Arama- Örnek

- d derinlikli tam arama ağacı; her bir yaprak olmayan düğümün b oğlu var:
- Toplam: 1 + b + b^2 + ... + b^d



– Örnek: 12 derinlikli tam arama ağacında 0,…,11 derinlikte her düğümün 10 oğlu var. 12.ci derinlikteki düğümlerin oğulları yoktur. Böylelikle, ağaçta 1 + 10 + 100 + 1000 + ... + 10^12 = (10^13 - 1) düğüm var

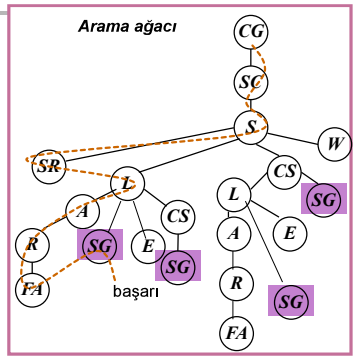
Enine aramada mekan ve zaman değerlendirmesi

Depth	Nodes	Time		Memory	
0	1	1	millisecond	100	bytes
2	111	0.1	second	11	kilobytes
4	11,111	11	seconds	1	megabyte
6	10^6	18	minutes	111	megabytes
8	10^8	31	hours	11	gigabytes
10	10^{10}	128	days	1	terabyte
12	10^{12}	35	years	111	terabytes
14	10^{14}	3500	years	11,111	terabytes

Time and memory requirements for breadth-first search, assuming a branching factor of 10, 100 bytes per node and searching 1000 nodes/second

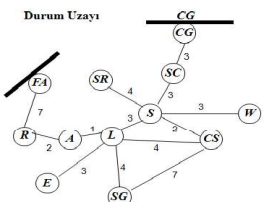
Derine Arama - Depth-first search

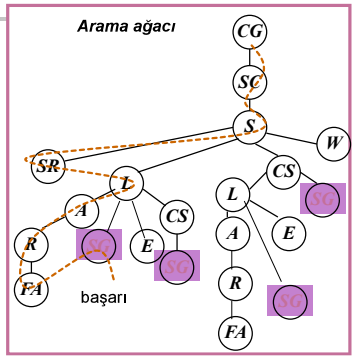
- Derine aramada arama ağacı yukarıdan aşağıya en sol düğümden başlayarak yaprak düğüme ulaşılana dek genişletiliyor.
- Eğer bir yolda çözüm bulunamazsa, arama sonraki en sol ve genişletilmemiş düğümle devam ettirilir.



Derine Arama - Depth-first search

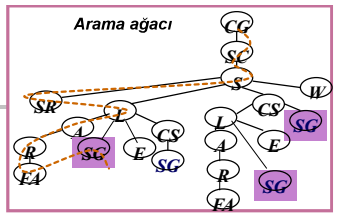
- Derine aramada arama ağacı yukarıdan aşağıya en sol düğümden başlayarak yaprak düğüme ulaşılana dek genişletiliyor.
- Eğer bir yolda çözüm bulunamazsa, arama sonraki en sol ve genişletilmemiş düğümle devam ettirilir.

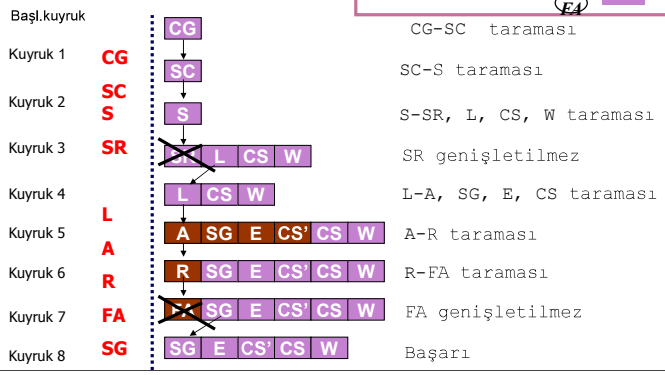




Derinine arama

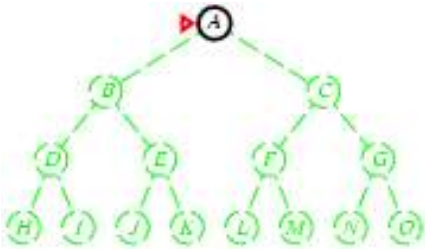
- **Derinine arama işlemleri ardışıklığı**





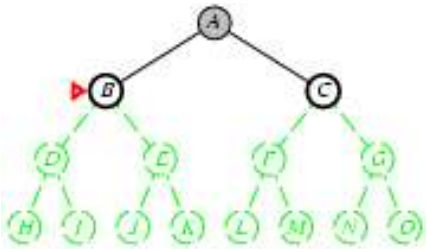
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



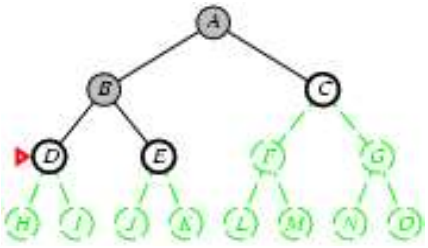
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



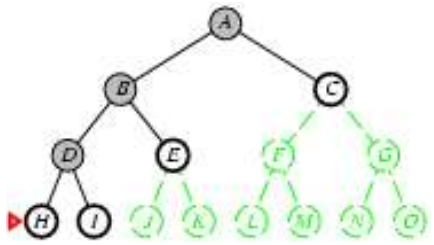
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



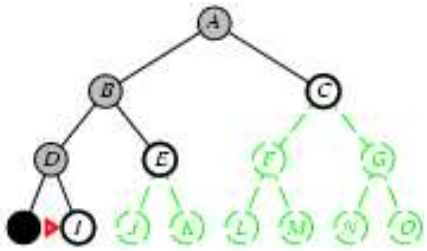
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



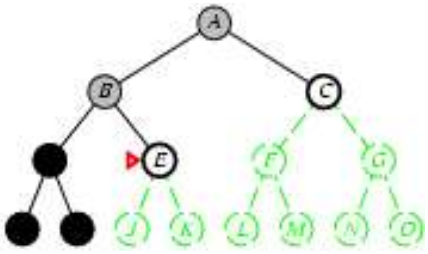
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



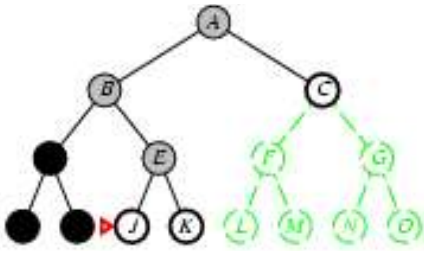
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



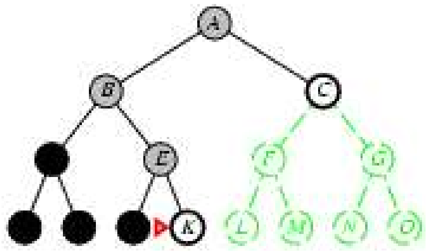
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



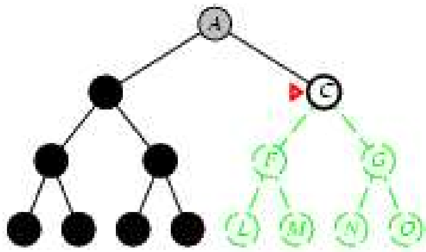
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



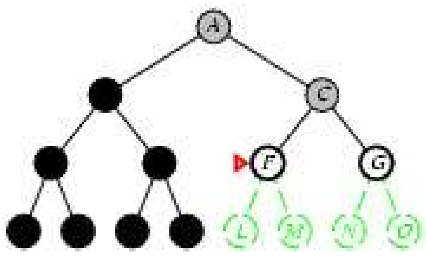
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



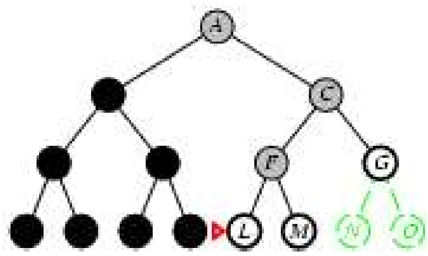
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



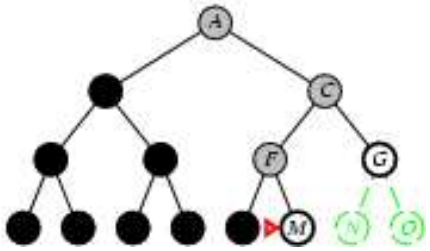
Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak

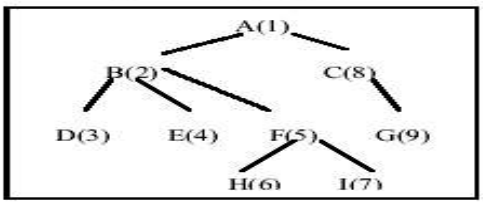


Derinine arama

- En derindeki taranmamış düğümü genişletmeli
- LIFO yapısı, yani ardıllar öne yazılacak



Derinine arama (başka bir örnek)



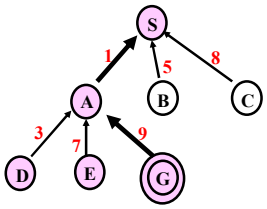
Düğümlerin yanında parantez içinde o düğümün taranma sırası gösteriliyor.

Derinine aramanın özellikleri

- Tam? Değil: sonsuz derinlik, döngülü durumlar olabiliyor
  - Tekrarlanan durumların önlenmesi için algoritmada değişiklik yapılması gerekiyor
    - → Sonlu uzay varsa tamdır
- Zaman? *O(b^m): m , d'den çok büyük ise zaman oldukça büyük olacak; m=yol uzunluğu*
  - çözümler çok ise, enine aramadan daha hızlı olabilir
- Uzay? *O(bm)- doğrusal uzay*
- Optimal? Değil

Derinine Arama algoritması

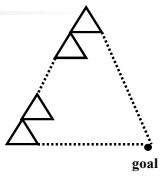
Gen. düğüm	**liste**
	{ S }
S	{ A B C }
A	{ D E G B C}
D	{ E G B C }
E	{ G B C }
G	{ B C }



Çözüm yolu S A G <-- G'nin değeri= 10
Genişletilen düğüm sayısı (amaç düğümle birlikte) = 5

Derinine Arama algoritması

- Algoritmanın esas özellikleri:
  - Genişletme için listeden her zaman en derindeki düğümü seçmeli ve yeni üretilmiş düğümleri listeye yazmalı
  - liste **LIFO** yapılıdır
  - Genişletme için seçilmiş düğüm amaç ise algoritmayı sonlandırmalı



- **Sonlu olmayabilir**
- **Tam değil**
- **Exponensiyel zaman** O(b^d)
- **Doğrusal mekan** O(bd)
- **Bazen** çözüm çok hızlı bulunabilir

Sınırlı derinine arama – Depth-limited search

Derinliğe bir sınır vererek derinine arama

```
function DEPTH-LIMITED-SEARCH( problem, limit) returns soln/fail/cutoff
  RECURSIVE-DLS(MAKE-NODE(INITIAL-STATE[problem]), problem, limit)
function RECURSIVE-DLS(node, problem, limit) returns soln/fail/cutoff
  cutoff-occurred? ← false
  if GOAL-TEST[problem](STATE[node]) then return SOLUTION(node)
  else if DEPTH[node] = limit then return cutoff
  else for each successor in EXPAND(node, problem) do
    result ← RECURSIVE-DLS(successor, problem, limit)
    if result = cutoff then cutoff-occurred? ← true
    else if result ≠ failure then return result
  if cutoff-occurred? then return cutoff else return failure
```

Sınırlı derinine arama – Depth-limited search

- Bu arama yönteminde, derinine aramada olası sonsuz (kısır döngü) arama işlemini önlemek için aramanın belirli bir seviyeye kadar yapılması düşünülmektedir.
- Örneğin, 12 şehirli bir Yol haritasında hiçbir çözüm 11'den fazla adım gerektirmemeli. Çünkü, yalnızca 12 şehir vardır. Bu nedenle sınır olarak 11 kullanabilir. Kısır döngülerin kesin var olduğunu kabul etmiyoruz, sadece varsayıyoruz ki, sorun sonlu derinlik seviyesinde çözülebilsin.

Sınırlı Derinine arama

- **Eğer gereken çözüm L+1 derinlikte ise, o hiçbir zaman bulunamayacak. (L-sınır derinliği)**
- **Karmaşıklık bakımından yöntem sıradan derinine aramaya benzer (azami derinliği ifade eden derinlik sınırını göz önüne alarak)**

Zaman karmaşıklığı	Uzay karmaşıklığı	Tam?	Optimal?
$O(b^l)$	O(bl)	Evet(çözüm <=l derinlikte ise)	hayır

Yinelemeli derinine arama
Iterative deepening search

- **Sınırlı derinine arama yönteminde en iyi arama sınırını bulmak her zaman kolay olmuyor.**
- **Arama uzayı büyük ve çözüm derinliği belli olmayan durumlarda yinelemeli derinine arama tercih edilen yöntemdir**
- **Başarıya ulaşana dek derinlik sınırı her defa 1 arttırılıyor**

Yinelemeli derinine arama
Iterative deepening search

- Zaman sınırı var.
- Her arama için ne kadar zaman kullanılacağı verilmelidir.

```
function ITERATIVE-DEEPENING-SEARCH( problem) returns a solution, or failure
    inputs: problem, a problem
    for depth ← 0 to ∞ do
        result ← DEPTH-LIMITED-SEARCH( problem, depth)
        if result ≠ cutoff then return result
```

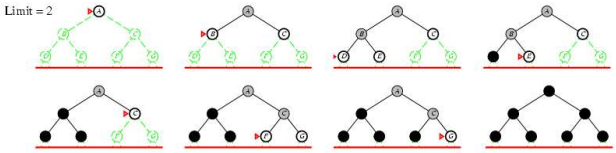
Yinelemeli derinine arama
Iterative deepening search

- Satranç turnuvalarında oyunlar kesin zaman sınırı içinde oynanıyor. Satranç programı her hamle için ne kadar zaman kullanacağına karar vermelidir. Pek çok satranç programı arama işlemini yinelemeli derinine arama ile yapıyor.
- **Yani, program önce 2 seviyede, sonra 3 , sonra 4... seviyede arama yapıyor. Bu, arama için ayrılan süre dolana dek devam ediyor.**
- **Bundan sonra program, bulunan hamleler içinden en iyisini çözüm olarak kabul ediyor**

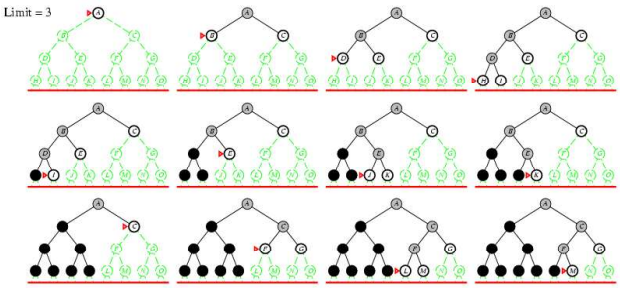
Yinelemeli derinine arama /=0

Yinelemeli derinine arama /=1

Yinelemeli derinine arama /=2



Yinelemeli derinine arama /=3



Yinelemeli derinine arama

- Sınırlı derinine arama yönteminde üretilen düğümler sayısı:
  $N_{DLS} = b^0 + b^1 + b^2 + \ldots + b^{d-2} + b^{d-1} + b^d$
- Yinelemeli derinine aramada üretilen düğümler sayısı:
  $N_{IDS} = (d+1)b^0 + d\ b\text{^}1 + (d-1)b\text{^}2 + \ldots + 3b^{d-2} + 2b^{d-1} + 1b^d$
- Örnek: *b = 10*, *d = 5*,
  - $N_{DLS} = 1 + 10 + 100 + 1{,}000 + 10{,}000 + 100{,}000 = 111{,}111$
  - $N_{IDS} = 6 + 50 + 400 + 3{,}000 + 20{,}000 + 100{,}000 = 123{,}456$
- Yineleme ve sınırlı arama arasındaki fark:
  (123,456 - 111,111)/111,111 = 11%

Yinelemeli derinine arama yönteminin özellikleri:

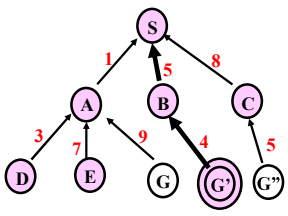
- Tam? Evet
- Zaman? $(d+1)b^0 + d\,b^1 + (d-1)b^2 + \ldots + b^d = O(b^d)$
- Mekan? $O(bd)$
- Optimal? Evet, eğer adım değeri=1

Sabit maliyet - Uniform-Cost (UCS)

- g(n) = başlangıç düğümden açık n düğümüne kadar yolun değeri (maliyeti)
- Algoritma:
  - her zaman en küçük g(n) değerli düğümü seçmeli; tüm yeni üretilmiş düğümleri listeye kaydetmeli
  - Listedeki düğümleri g(n) 'nin artması ardışıklığı ile sıralamalı
  - Açılmak için seçilmiş düğüm amaç ise algoritmayı sonlandırmalı

Sabit maliyet Araması

Açılan düğüm	**düğümler listesi**
	{S(0)}
S	{A(1) B(5) C(8)}
A	{D(4) B(5) C(8) E(8) G(10)}
D	{B(5) C(8) E(8) G(10)}
B	{C(8) E(8) G'(9) G(10)}
C	{E(8) G'(9) G(10) G"(13)}
E	{G'(9) G(10) G"(13) }
G'	{G(10) G"(13) }



çözüm yolu S B G <-- G'nin değeri 10 değil, 9'dur

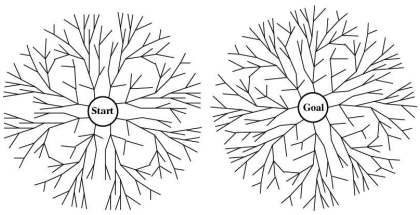
Açılan düğüm sayısı (amaç düğümle birlikte) = 7

Sabit maliyet yönteminin özellikleri

- **Tam** (her bir adımın değeri sonsuz değilse)
  - g(n) <= g(amaç) koşulu ile durum uzayında düğüm sayısı n sonludur
- **Optimal/Uygun**
  - amaç denemesine bağlıdır
  - Çoklu çözüm yolları
  - Açık n düğümünden üretilen her çözüm yolunun değeri >= g(n)
  - Genişletme için açılan ve denemeden geçen birinci düğümün yol değeri listedeki her bir açık düğümün değerinden küçük veya eşittir
- **Eksponensiyel zaman ve mekan karmaşıklığı** (b^d) ;

d- en küçük değerli çözüm için çözüm yolunun derinliğidir

İkiyönlü arama - Bi-directional search



- Başlangıç durumdan amaca ve amaç durumundan başlangıca doğru aynı zamanda arama
- Yollar kesiştiğinde durmalı
- Tek bir başlangıç ve amaç durumu olduğunda ve hareketler değiştirilebilir olduğunda iyidir
- Çözüme daha hızlı ulaşmak mümkün olabilir

Arama yöntemlerinin karşılaştırılması

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal?	Yes	Yes	No	No	Yes	Yes
Complete?	Yes *	Yes	No	Yes, if $l \geq d$	Yes *	Yes

*- if step costs are all identical

Arama Yöntemlerinin özeti

- Yapay Zeka'da kullanılan arama teknikleri, bizi verilen başlangıç durumdan amaç durumuna (durumlarına) doğru götüren adımlar ardışıklığının bulunmasına dayanmaktadır.
- Enine ve derinine arama algoritmaları sonlu arama ağacında tüm düğümlerin bakılmasını gerektirebilir.
- Hangi algoritmanın seçileceği çözülecek soruna bağlıdır.
- Kısmi yolların makul derinlikten sonra ölü sona veya başarılı sona ulaşacağına inanılıyorsa , derine arama yöntemini kullanmak mantıklıdır.
- Yinelemeli Derinine arama küçük bellek alanı ister (derinine arama gibi) ve en kısa yolu önce bulur (enine arama gibi)

Arama Yöntemlerinin özeti

- En kısa yolu bulmak istiyorsanız en iyisi enine arama yöntemini kullanmaktır
- Daha az bellek alanı kullanmak gerekiyorsa derinine arama kullanmak daha etkilidir
- **Sabit Maliyet araması**:
  - Hareketlerin değerleri farklıdır
  - En az değerli çözüm gerekiyor

  Sabit maliyet aramasında yalnızca yol değeri dikkate alınıyor
- Çözümü daha çabuk bulmak gerekiyorsa o zaman daha karmaşık algoritmalar kullanılmalıdır!
- Çözüm durumlarına götüren pek çok yol varsa derinine arama hızlıdır, fakat yollar çok uzun olabilir.
- Hedefe götüren yalnız bir kısa yol varsa enine arama daha hızlıdır. Fakat arama uzayı geniş ve derindir.