Yapay Zeka

Ders 5 – Bölüm 1

Doç. Dr. Mehmet Dinçer Erbaş
Bolu Abant İzzet Baysal Üniversitesi
Mühendislik Fakültesi
Bilgisayar Mühendisliği Bölümü

Bilgisiz arama

- Enine arama (breadth-first search)
- Sabit maliyetli arama (uniform cost search)
- Derinlik öncelikli arama (depth-first search)
- Derinlik limitli arama (depth limited search)
- Yinelemeli derinleşen derinlik öncelikli arama (iterative deepening depth-first search)
- Çift yönlü arama (bidirectional search)

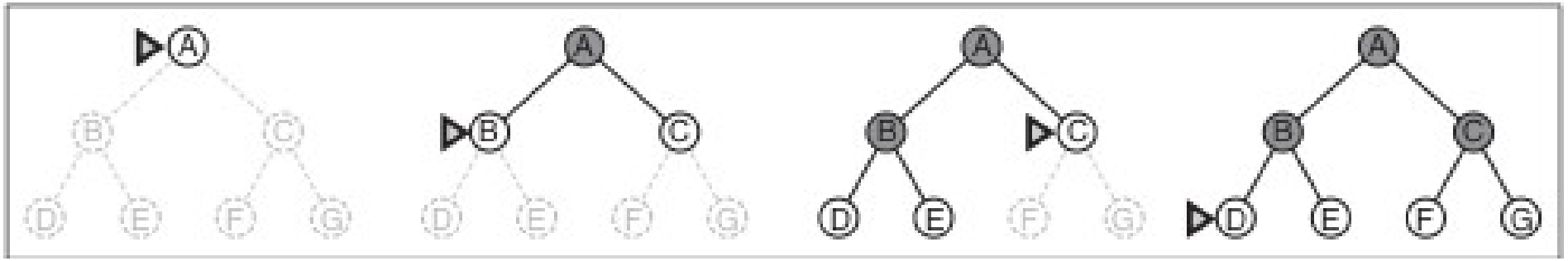
Bilgisiz arama

- Enine arama
  - Enine arama metodu kullanıldığında graftaki açılmamış düğümlerden en sığ (köke en yakın) olanı açılmak üzere seçilir.

```
fonksiyon ENİNE-ARAMA (problem) dönüş bir çözüm veya çözümsüz
  düğüm <== DURUM = problem.BAŞLANGIÇ-DURUMU olan bir düğüm
  yol-maliyeti = 0
  if problem.HEDEF-TESTİ(düğüm.DURUM) then return ÇÖZÜM(düğüm)
  sınır <== sadece düğümü içeren bir FIFO sıra
  keşfedilmiş <== boş küme
  loop do
     if BOŞ?(sınır) then return çözümsüz
     düğüm <== POP(sınır) /* sınırdaki en sığ düğümü seçer */
     Ekle düğüm.STATE'i keşfedilmişe
     for each aksiyon in problem.ASKİYONLAR(düğüm.DURUM) do
          alt <== ALT-DÜĞÜM(problem, düğüm, aksiyon)
          if alt.STATE keşfedilmiş veya sınır içinde değilse then
               if problem.HEDEF-TESTİ(alt.DURUM) then return ÇÖZÜM(alt)
               sınır <== EKLE(alt,sınır)
```

Bilgisiz arama

- Enine arama
  - Bir FIFO sırası (ilk eklenen ilk işleme alınır) kullanılarak bu özellik sağlanabilir.
  - Böylece yeni eklenen düğümler sıranın sonuna gider ve daha önce eklenmiş daha sığ olan düğümler önce işleme alınır.



Bilgisiz arama

- Enine arama
  - Bütün müdür?
    - Çözüm d dernliğinde olsun. Enine arama daha sığ olan bütün düğümleri açtıktan sonra bu düğümü bulacaktır => Evet.
  - Optimal midir?
    - Bulunan en sığ çözüm optimal çözüm olmayailir.
    - Enine arama, yol maliyetleri düğüm derinliğine bağlı azalmayan bir fonksiyon ile hesaplandığında optimal olur.
      - Her aksiyon aynı maliyete sahip olduğunda bu durum olur.

Bilgisiz arama

- Enine arama
  - Ağacın her düğümünün b tane alt düğümü olduğunu farzedelim.
  - Aradığımız çözüm d derinlikte olsun
  - Bu durumda toplam açılacak düğüm sayısı
    - $b + b^2 + b^3 + \ldots + b^d = O(b^d)$
  - Yer açısından düşündüğümüzde açılan her düğüm keşfedilmiş kısmında saklanıyor. Ayrıca sınırda yeni açılanlar saklanıyor.
    - Yer karaşıklığı $O(b^d)$
  - Hem zaman hem de yer karmaşıklığı oldukça hızlı şekilde artıyor.
  - Örnek: b= 10 olsun, saniyede 1 milyon düğüm oluşturulabilsin ve her düğüm 1000 byte yer gerektirsin.

Bilgisiz arama

- Enine arama

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

- Üstel karmaşıklığa sahip arama problemleri bilgisiz arama metotları ile ufak çaplı örnekler dışında çözülemez.

Bilgisiz arama

- Sabit maliyet arama
  - Eğer her adım maliyeti eşit olursa, enine arama optimal hale gelir.
    - Çünkü her zaman en sığ açılmamış düğüm açılır.
  - Basit bir değişiklik ile bu algoritmayı her tür adım maliyeti ile optimal hale getirebiliriz.
  - En sığ düğümü açmak yerine, sabit maliyet arama düğümleri adım maliyetine göre sıralar ve en düşük adım maliyetine sahip düğümü açar.
    - Sınırdaki düğümler adım maliyetlerine göre bir öncelikle sırada tutalarak bu işlem gerçekleştirilir.
  - Enine arama ile karşılaştırdığımızda iki yenilik var.
    - Hedef testi, düğüm oluşturulduğunda değil, düğüm açılmak üzere seçildiğinde yapılıyor.
    - Sınırdaki bir düğüm için daha iyi bir yol bulunması durumu için bir test eklenmiş.

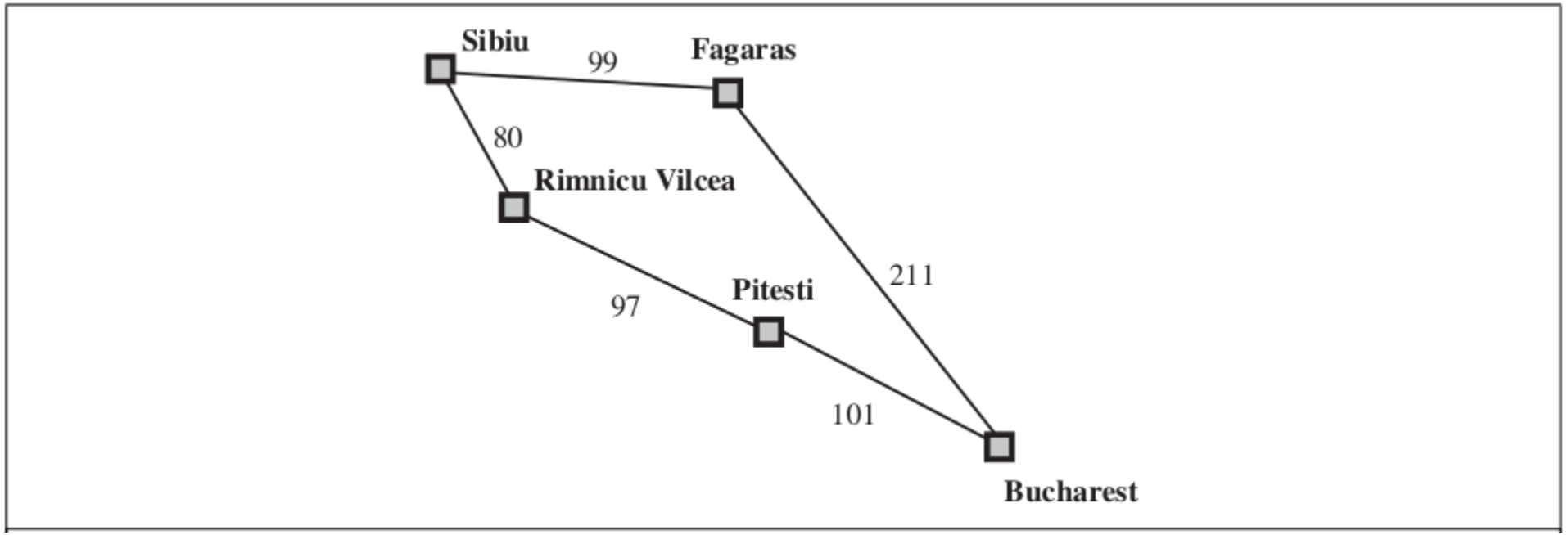
Bilgisiz arama

- Sabit maliyet arama

```
fonksiyon SABİT-MALİYET-ARAMA (problem) dönüş bir çözüm veya çözümsüz
  düğüm <== DURUM = problem.BAŞLANGIÇ-DURUMU olan bir düğüm
  yol-maliyeti = 0
  sınır <== sadece düğümü içeren bir YOL-MALİYETİNE göre sıralanmış
öncelikli sıra
  keşfedilmiş <== boş küme
  loop do
     if EMPTY?(sınır) then return çözümsüz
     düğüm <== POP(sınır) /* sınırdaki en düşük maliyetli düğümü seçer */
     if problem.HEDEF-TESTİ(düğüm.DURUM) then return ÇÖZÜM(düğüm)
     Ekle düğüm.STATE'i keşfedilmişe
     for each aksiyon in problem.ASKİYONLAR(düğüm.DURUM) do
          alt <== ALT-DÜĞÜM(problem, düğüm, aksiyon)
          if (alt.STATE keşfedilmiş veya sınır içinde değilse then
                sınır <== EKLE(alt,sınır)
          else if alt.DURUM sınırda ve YOL-MALİYETİ daha yüksek ise then
                sınırdaki belirtilen düğümü alt ile değiştir
```

Bilgisiz arama

- Sabit maliyet arama



Bilgisiz arama

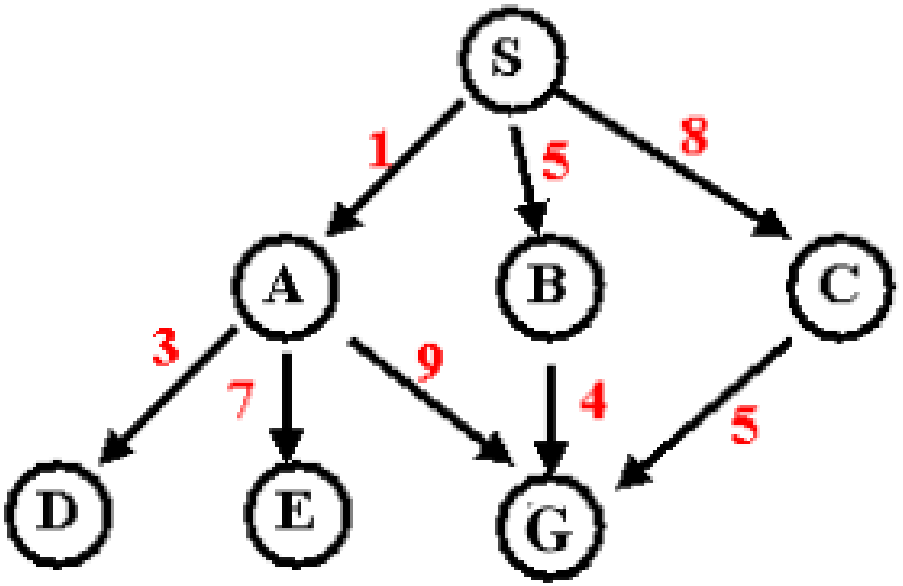
- Sabit maliyet arama
  - Sabit maliyetli arama genelde optimaldir diyebiliriz.
    - Bir düğüm açılmak için seçildiğinde, bu düğüme optimal yol bulunmuştur.
    - Ayrıca hiçbir adım maliyeti negatif olmadığı için yollar kısalmaz.
      - Böylece sabit maliyetli arama düğümleri optimal yol maliyetine göre açar.
  - Sabit maliyetli arama atılan adım sayısı ile ilgilenmez, sadece adımların maliyetlerine bakar.
    - Bu sebeple 0-maliyetli sınırsız bir aksiyon var ise takılı kalır.
    - Bütün olabilmesi için her adımın maliyetinin en az ε pozitif sabitinden büyük olması gerekir.

Bilgisiz arama

- Sabit maliyet arama
  - Sabit maliyet arama düğüm derinliği ile değil yol maliyetleri üzerinden çalışır.
    - Bu sebeple b ve d üzerinden karmaşıklığını hesaplamak zordur.
  - C* optimal çözümün maliyeti olsun, her aksiyonun maliyeti en az Ɛ olsun.
  - Bu durumda algoritmanın en kötü zaman ve yer karmaşıklığı $O(b^{1+C^*/\varepsilon})$ olur ve bu b^d'den büyük olabilir.
    - Bunun sebebi sabit maliyetli aramanın büyük bir arama ağacındaki ufak adımları keşfederken, büyük ve belki de yararlı adımları keşfetmekte gecikmesidir.
  - Her adım maliyeti eşit ise $b^{1+C^*/\varepsilon}$ b^{1+d} olur.
  - Adım maliyetleri eşit olduğunda sabit maliyetli arama enine aramaya benzer.
    - Ancak enine arama çözümü bulur bulmaz dururken, sabit maliyetli arama hedefin bulunduğu derinlikteki bütün düğümleri daha düşük maliyetli bir yol olup olmadığını bulabilmek için açar.

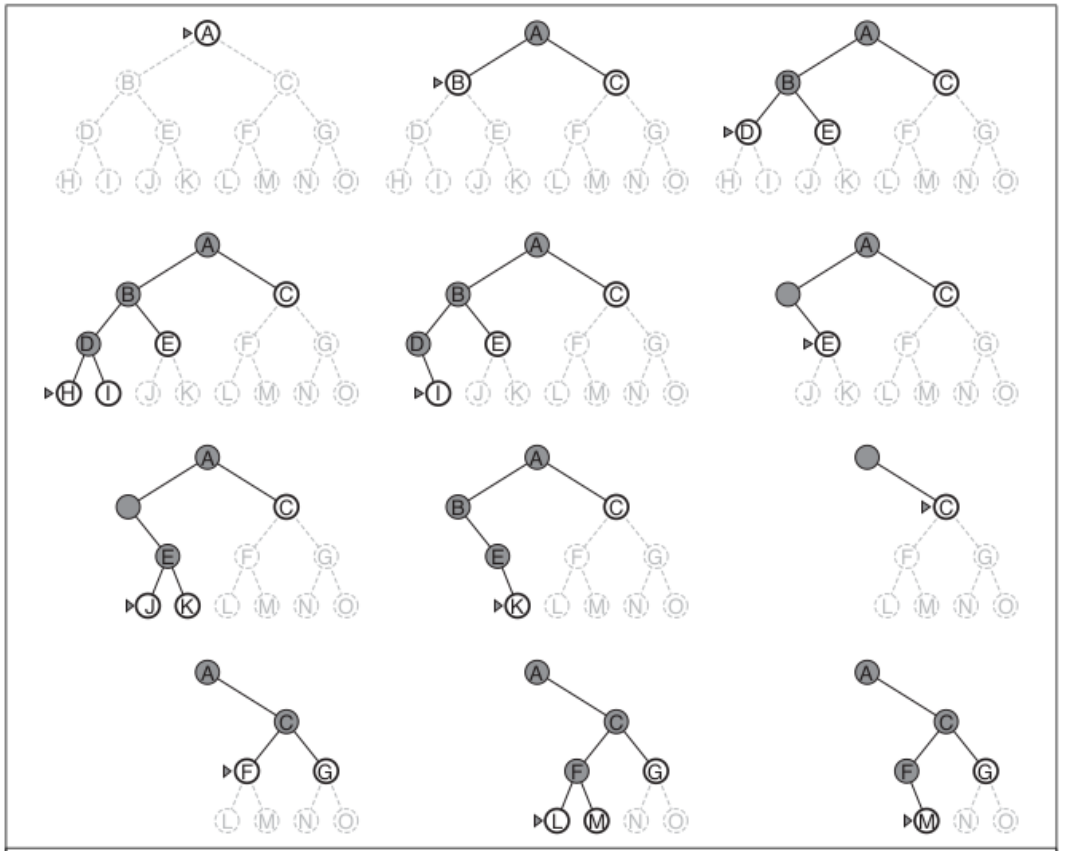
Bilgisiz arama

- Örnek



Bilgisiz arama

- Derinlik öncelikli arama (ing: depth first search)
  - Derinlik öncelikli arama daima sınırdaki düğümler arasında en yüksek derinliğa sahip düğümü açar



Bilgisiz arama

- Derinlik öncelikli arama
  - Bu algoritmada arama anında arama ağacının alt düğümleri olmayan seviyesine ilerler.
  - Alt düğümü olmayan düğümler açıldıktan sonra sınırdan silinirler
    - Sonrasında arama geri dönerek en yüksek derinliğe sahip açılmamış bir sonraki düğüme geçer.
    - Derinlik öncelikli arama enine aramaya benzer
      - Ancak enine arama FIFO sırası kullanırken derinlik öncelikli arama LIFO sırası kullanır.
      - LIFO sırası ile en son oluşturulan düğüm açılmak üzere seçilecektir.
      - Bu düğüm en derindeki açılmamış düğümdür, çünkü bu düğüm üst düğümünden bir seviye derindedir – üst düğüm de açıldığında en derindeki açılmamış düğüm olmalıdır.

Bilgisiz arama

- Derinlik öncelikli arama
  - Derinlik öncelikli arama algoritmasının özellikleri graf-arama veya ağaç-arama versiyonlarından hangisinin kullanıldığına göre değişkenlik gösterir.
    - Graf-arama versiyonu tekrar edilen durumları ve gereksiz yolları engellediği için sınırlı durum uzayında bütündür.
      - Çünkü sonuçta bütün düğümleri açacaktır.
    - Ağaç-arama versiyonu ise bütün değildir.
      - 24 numaralı slayttaki örnekte bulunan ağaçta döngüye girer.
    - Ağaç-arama versiyonu değiştirilerek yol üzerindeki düğümlerin tekrar açılmaması sağlanabilir.
      - Ancak bu durum gereksiz yollardan kurtulmamızı sağlamaz.
    - Sınırsız durum uzayında, her iki versiyon sınırsız hedef-olmayan bir yol var ise başarısız olur.

Bilgisiz arama

- Derinlik öncelikli arama
  - Benzer sebeplerle her iki versiyon optimal değildir.
    - 47 numaralı slaytta gösterilen örnekte C düğümü hedef olsaydı, Bu düğümün bulunabilmesi için sol tarafta alt-ağacın tamamının açılması gerekiyordu.
    - j düğümü bir başka hedef olsaydı, bu durumda derinlik öncelikli arama bu düğümü dönecekti – Halbuki daha düşük maliyetli bir çözüm vardı.
    - Öyleyse derinlik öncelikli arama optimal değildir.

Bilgisiz arama

- Derinlik öncelikli arama
  - Derinlik öncelikli arama algoritmasının zaman karmaşıklığı durum uzayının büyüklüğü ile sınırlıdır (durum uzayı sınırsız olabilir).
  - Derinlik öncelikli ağaç arama, arama ağacındaki bütün $O(b^m)$ düğümü oluşturabilir
    - Burada m en derindeki düğümün derinliğidir.
    - Açılan düğümlerin sayısı durum uzayından çok büyük olabilir.
  - Ayrıca m değeri d değerinden (en sığ çözümün derinliği) çok büyük olabilir.
    - Ağaç sınırsız ise m sonsuz olabilir.

Bilgisiz arama

- Derinlik öncelikli arama
  - Derinlik öncelikli aramanın enine aramaya üstün bir yanı var mıdır?
    - Evet! => yer karmaşıklığı
  - Graf-arama metodunda farklılık olmamakla birlikte, ağaç-arama yapıldığında derinlik öncelikli aramanın kök düğümden bir yaprak düğüme sadece bir yol ile birlikte yol üzerindeki açılmamış alt düğümleri saklaması yeterlidir.
  - Bir düğüm açıldıktan sonra, alt düğümlerinin her biri keşfedildiğinde hafızadan silinebilir.
  - Dallanma faktörü b, maksimum derinlik m ise, derinlik öncelikli arama sadece O(bm) düğümü saklamalıdır.
  - 40 numaralı slayttaki örneğe bakarsak d = 16 olduğunda 10 exabyte yerine 156 Kb yer yeterli olacaktır.
  - Yer karmaşıklığı konusundaki bu güzel özelliği nedeniyle birçok YZ alanında derinlik öncelikli arama kullanılır.
    - Örneğin kısıtlama sağlama problmeleri, önerme sağlayabilirlik,
    - mantık programalama gibi.

Bilgisiz arama

- Derinlik limitli arama
  - Önceki slaytlarda görüdüğümüz üzere derinlik öncelikli arama sonsuz durum uzayında başarılı olamıyor.
  - Bu sorunu çözebilmek için derinlik öncelikli aramaya baştan tanımlı bir derinlik limiti l sağlayabiliriz.
    - l derinlikteki düğümler alt düğüme sahip olamaz.
  - Bu yaklaşım derinlik limitli arama olarak adlandırılır.
  - Derinlik limiti sonsuz yol problemini çözer.
  - Ancak $l < d$ ise, bütün olamaz.
  - $l > d$ seçsek bile optimal değildir.
  - Zaman karmaşıklığı $O(b^l)$, yer karmaşıklığı ise $O(bl)$ olacaktır.
  - Derinlik öncelikli arama, $l = \infty$ olarak belirlenen derinlik limitli aramanın özel bir durumudur.

Bilgisiz arama

- Derinlik limitli arama
  - `l` değerini seçerken problem hakkında sahip olduğumuz bilgi kullanılabilir.
    - Örneğin Romanya tatili probleminde haritada toplam 20 şehir var.
    - `l` = 20 seçilebilir.
    - Haritayı incelediğimizde herhangi bir şehirden diğerine en fazla 9 adımda gidilebildiğini görüyoruz.
    - Öyleyse `l` = 9 seçilebilir.
  - Derinlik limitli arama yinelemeli fonksiyonlar kullanılarak yapılabilir.

Bilgisiz arama

- Derinlik limitli arama

```
fonksiyon DERINLIK-LIMITLI-ARAMA (problem, limit) dönüş bir çözüm veya çözümsüz / kesme
  return YINELEMELI-DLA(DÜĞÜM-OLUŞTUR(problem.BASLANGIC-DURUMU),problem, limit)

fonksiyon YINELEMELİ-DLA(düğüm, problem, limit) dönüş bir çözüm veya çözümsüz / kesme
  if problem.HEDEF-TESTİ(düğüm.DURUM) then return ÇÖZÜM(düğüm)
  else if limit = 0 then return kesme
  else
    kesme_oldu? <== false
    for each aksiyon in problem.AKSIYONLAR(düğüm.DURUM) do
      alt <== ALT-DÜĞÜM(problem, düğüm, aksiyon)
      sonuç <== YINELEMELI-DLA(alt,problem,limit – 1)
      if sonuç = kesme then kesme_oldu <== true
      else if sonuç ≠ çözümsüz then return sonuç
    if kesme_oldu? then return kesme else return çözümsüz
```

Bilgisiz arama

- Yinelemeli derinleşen derinlik öncelikli arama
  - Yinelemeli derinleşen derinlik öncelikli arama, derinlik öncelikli arama ile birlikte kullanılan bir genel arama stratejisidir.
  - Düzenli olarak limiti artırarak çalışır.
    - Limit başta 0, sonra 1, sonra 2, … hedefe ulaşıncaya kadar devam eder.
  - En sığ olan hedef d derinliğinde ise, limit d oluncaya kadar limit artırımı devam eder.

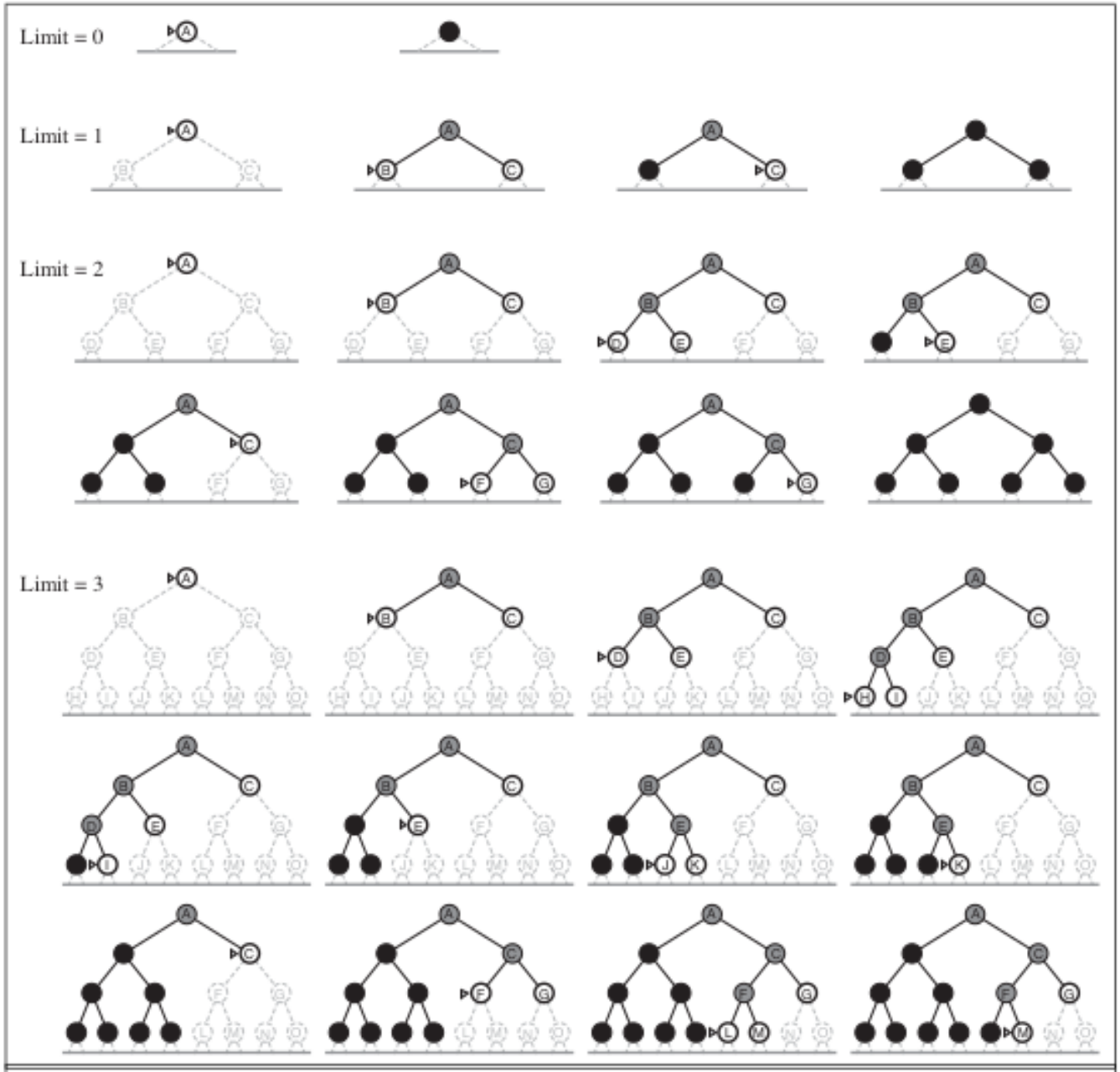
Bilgisiz arama

- Yinelemeli derinleşen derinlik öncelikli arama

```
fonksiyon YINELEMELI-DERINLESEN ARAMA (problem) dönüş bir çözüm veya çözümsüz
  for derinlik = 0 to ∞ do
    sonuç = DERINLIK-LIMITLI-ARAMA(problem, derinlik)
    if sonuc ≠ kesme then return sonuç
```

- Yinelemeli derinleşen arama derinlik öncelikli arama ve enine arama algoritmalarının iyi yanlarını birleştirir.
  - Derinlik öncelikli aramada olduğu gibi yer karmaşıklığı O(bd) olur.
  - Enine aramada olduğu gibi dallanma faktörü sınırlı ise bütün olur.
  - Enine aramada olduğu gibi yol maliyetleri azalmayan fonksiyon ise optimal olur

Bilgisiz arama



Bilgisiz arama

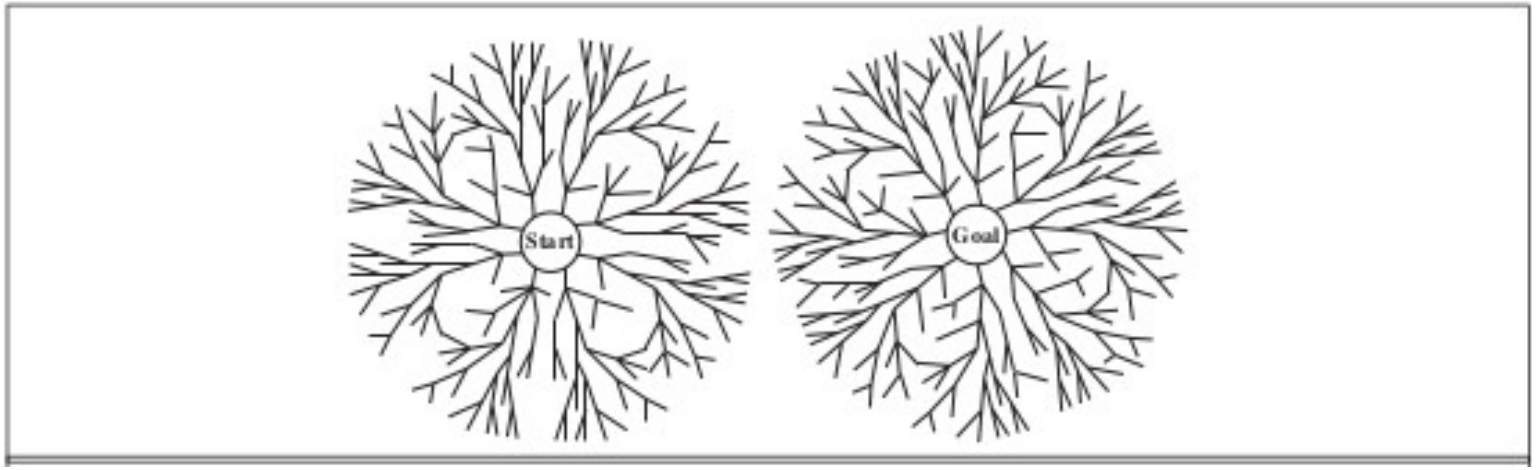
- Yinelemeli derinleşen derinlik öncelikli arama
  - Yinelemeli derinleşen arama, düğümler birden fazla kez oluşturulduğu için savurgan olarak görülebilir.
  - Ancak bu durum aslında maliyeti fazla artırmamaktadır.
    - Bunun sebebi, sabit bir dallanma faktörüne sahip bir arama ağacında, düğümlerin büyük kısmı en altta seviyede oluşmaktadır. Bu sebeple üstteki seviyelerin birden fazla oluşturulması fazla önem arzetmemektedir.
  - Yinelemeli derinleşen aramada, en alt seviyedeki düğümler (derinlik d) bir kere, bir üst seviyedeki düğümler iki kere … kök düğüm ise d kere oluşturulur.
  - Öyleyse en kötü durumda oluşturulan düğüm sayısı
    - $N(\text{düğüm}) = (d)\ b + (d - 1)\ b^2 + \ldots + (1)\ b^d$
  - Bu hesap ile $O(b^d)$ asimtotik sınırına erişiriz.
    - Enine arama ile aynı.
    - Sadece üst seviyedeki düğümlerin birden fazla oluşturulmasından kaynaklı ek maliyet var.

Bilgisiz arama

- Yinelemeli derinleşen derinlik öncelikli arama
  - Örnek için b = 10 ve d = 5 ise oluşturulan düğüm sayısı yinelemeli derinleşen arama için
    - N(düğüm) = $5 * 10 + 4 * 10^2 + 3 * 10^3 + 2 * 10^4 + 10^5 =$ 123,450
  - Enine arama için
    - N(düğüm) = 10 + 100 + 1000 + 10000 + 100000 = 111,110
  - Yinelemeli derinleşen arama, arama uzayı geniş olduğu ve çözümün derinliği bilinmediği durumlarda genellikle tercih edilen bilgisiz arama metodudur.

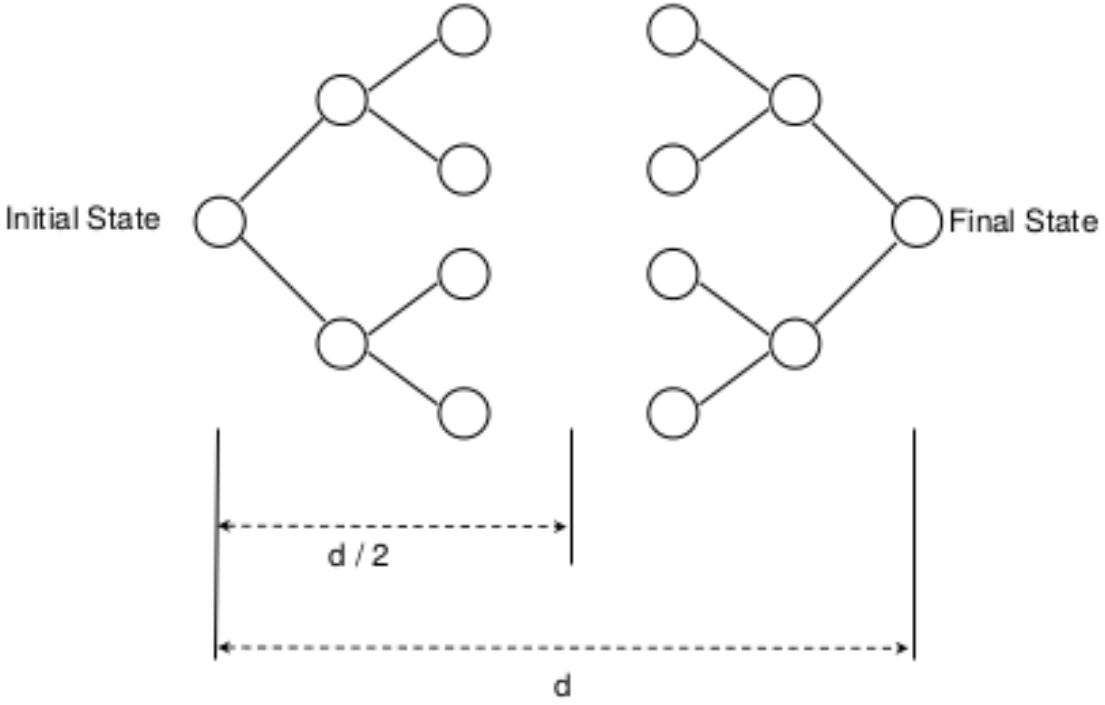
Bilgisiz arama

- Çift yönlü arama (ing: bidirectional search)
  - Bu arama metodunun arkasındaki fikir aynı anda iki farklı arama başlatmak ve bu iki aramayı bir noktada buluşturmaktır
    - Bu aramalardan biri başlangıç durumundan başlarken, diğeri hedef durumundan başlar.
  - $b^{d/2} + b^{d/2}$ ile hesaplanan karmaşıklık b^d olarak hesaplanandan küçük olacaktır.



Bilgisiz arama

- Çift yönlü arama



Bilgisiz arama

- Çift yönlü arama
  - Zaman karmaşıklığını azatlma ihtimali çift yönlü aramayı çekici hale getirir.
  - Ancak sorulması gereken soru: Hedeften geri aramayı nasıl yapacağız?
  - Alt düğümleri oluşturduğumuz gibi üst düğümlerini oluşturan bir metodumuz olmalı
  - Durum uzayındaki her aksiyon geri döndürülebilir olursa, görece yapılabilir.
  - Ancak başka durumlarda bu işlemi yapmak oldukça zor olabilir.
  - Önceki örneklerimizi düşünürsek, harita üzerinde yapmak mümkün olabilirken 8-vezir oyununda uygulamak zordur.

Bilgisiz arama

- Bilgisiz arama algoritmalarının karşılaştırılması

Kriter	Enine	Sabit maliyet	Derinlik öncelikli	Derinlik limitli	Yinelemeli derinleşen	Çift yönlü (uygunsa)
Bütün?	Evet[a]	Evet[a,b]	Hayır	Hayır	Evet[a]	Evet[a,d]
Zaman	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(b^m)$	$O(b^l)$	$O(b^d)$	$O(b^{d/2})$
Yer	$O(b^d)$	$O(b^{1+\lfloor C^*/\varepsilon \rfloor})$	$O(bm)$	$O(bl)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Evet[c]	Evet	Hayır	Hayır	Evet[c]	Evet[c,d]

Dallanma faktörü b, en sığ çözümün derinliği d, arama ağacının maksimum derinliği m, derinlik limiti `l`. [a] => b sınırlı ise bütün; [b] => adım maliyetleri ≥ Ɛ, pozitif Ɛ değeri için, ise bütün, [c] => bütün adım maliyetleri aynı ise optimal, [d] => eğer iki yönde enine arama kullanılırsa