* Yapay Zeka’nın Temel Konuları :
  + Bilgi Gösterimi (Knowledge Representation)
  + Çıkarsama (Inference)
  + Öğrenme (Learning)

Alt Çalışma Alanları

* Mantıksal YZ (Logical AI )
* Arama (Search)
* Örüntü Tanıma (Pattern Recognition)
* Temsil, Gösterim (Representation)
* Çıkarsama (Inference)
* Akıl Yürütme (Reasoning)
* Öğrenme (Learning From Experience)
* Varlıkbilim (Ontology)
* Sezgi (Heuristics)

Alt Çalışma Alanları

* İnsanların zorlandıkları alanlarda işlem yapabilecek sistemler üretmek.
* İnsan beyninin fonksiyonlarını, bilgisayarda modellemeye çalışmak.
* İnsanın öğrenme ve buluş yapma gibi zihinsel yeteneklerini araştırmak.
* Bilgisayarda öğrenmeyi gerçekleştirebilmek
* İnsan bilgisayar iletişimini kolaylaştıran kullanıcı arabirimleri geliştirmek.
* Yapay uzman sistemler oluşturmak.
* Yapay zekaya sahip robotlar geliştirmek

**Örnek AI Sistemi**

* Satranç Oynama (Deep Blue (IBM))
  + Algılama : Satranç tahtasının ileri özellikleri
  + Eylemler : Hareket seçme
  + Çıkarsama : Tahta konumlarını değerlendirme sezgisi ve arama.
* Tıbbi Teşhis (Pathfinder (D. Heckerman, Microsoft Research))
  + Algılama : Belirtiler, test sonuçları
  + Eylemler : Test önerme, teşhis etme
  + Çıkarsama : “Bayesian Inference”, makine öğrenmesi
* Kendi Başına Giden Araba (ALVINN (D. Pomerleau, CMU))
  + Algılama :Yolun sayısal kamera görüntüsü
  + Eylemler : 64 farklı direksiyon açısı
  + Çıkarsama : “Back propagation” eğitimli yapay sinir ağı.

Agents

* Agent, çevresini sensörler ile algılayan ve kumandalar (actuators) ile eylemde bulunan bir varlıktır.
* Kısaca bir ajan, eski algılardan eylemlere bir fonksiyondur.

Aranan İyi Davranış:Rasyonel(Mantıklı) Ajan

* Bir ajanı iyi veya kötü yapan nedir?
  + Verilen şart ve işler için, en iyi performansı veren etmen aranır.
* Bilgisayar sınırlamalarından dolayı ideal duruma ulaşılamaz.
* Verilen makine kaynakları ile yapılabilecek en iyi programın tasarımı
* Figür 2.3 deki mantıklı bir ajan mıdır?
  + Öncelikle performans ölçüsünün ne olduğunu söylemeliyiz.
  + Bunun için de çevresel özellikleri, sensörleri ve ajanlara ait kumandaları(actuators) bilmeliyiz.
* Ajanlar, mantıklı davranmak zorundadırlar.
* **Mantıklı hareket :** Verilen bilgiler doğrultusunda, performans ölçüsünün beklenen değerini maksimize eden eylem.
* Beklenen değer olmasının nedeni
  + ortamların deterministik olmaması
  + Belirsizlik ve beklenmeyen, istenmeyen durumlar.

Nature of Enviroments

* Mantıksal ajanlar yapılandırmak için**:**
* Öncelikle task enviroments seçenekleri düşünülür. Bunlar genelde rasyonel ajanların çözüm olduğu problemlerdir.
* Tanımlamamız gereken task enviroments **PEAS (**Performance, Environment, Actuators, Sensors)olarak adlandırılacak.
* Ajan tasarımında, öncelikle PEAS seçenekleri olabildiğinde tamamlanmış şekilde oluşturulmalıdır.

Agent Tasarımı

* Task environment 4 temel eleman içerir(
  + Percepts : Agent’ın aldığı bilgiler
  + Actions : Agent’ın yaptıkları
  + Goals : Agent’ın ulaşmaya çalıştığı
    - İstenen Durum
  + Environment : Agent’ın hareket ettiği yer, ortam

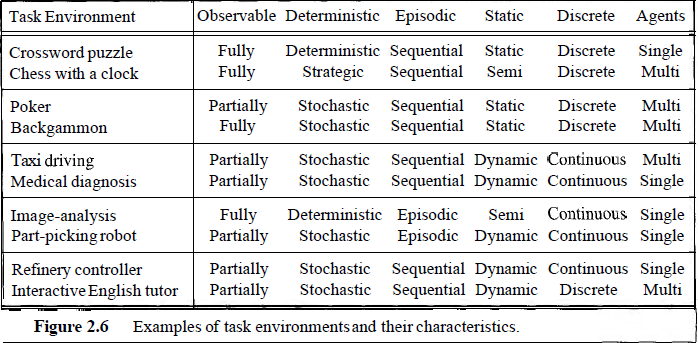
Örnek software agent (or software robot or softbot)

* Sensors: fuctions
* Actuators:functions
* İnternet haber kaynaklarını tarayan ve ilginç olanları gösteren bir softbot tasarımı.
* Doğal dil işleme işlemlerini yapabilme yeteneğine ihtiyaç var.
* Her müşterinin ilgi duyduğu alanları öğrenmesine ihtiyaç duyulacak.
* Ve planları dinamik olarak değiştirebilen bir softbot olacak, örneğin bir haber kaynağı için olan bağlantı sorunlu ise dinamik olarak başka kaynağa yönlenecek yada çözüm üretecek.

Properties of task environments

* **Accessible vs Inaccessible :** Agent ortamın durumunu görebiliyor mu?
* **Deterministic vs. Stochastic :** Aynı koşullarda yapılan aynı hareket hep aynı sonucu veriyor mu?
* **Episodic vs. Infinite :** İş, bilinen zaman aralığında bitirilebiliyor mu?
* **Static vs. Dynamic :** Agent düşünürken, ortamda değişiklik oluyor mu?
* **Discrete vs. Continuous :**
  + discrete-state environment: satranç oyunu gibi. Sabit sayıda farklı durumları vardır. Satranç aynı zamanda farklı percepts ve actions kümesine sahiptir.
  + Continuous-state enviroment: Taxi driving. Taksinin hızı ve yeri zaman içinde sürekli değişir.

Examples of Task Enviroments



AI Neden Zordur?

* Gerçek ortamlar genelde, “inaccessible”, “stochastic”, “dynamic” ve “continuous” ’dur.
* “Accessible”, “deterministic”, “static”, “discrete” ortamlar da olasılıkları çok fazladır. (Satranç örneği)
* Sınırlı kaynaklar (bellek, zaman, yetersiz bilgi)
* Agent tasarımında “environment” türünün büyük önemi vardır.
* Gerçek ortamlar genelde, "ulaşılamaz", "stokastik", "dinamik" ve "sürekli" 'dur.
* "Erişilebilir", "deterministik", "statik", "ayrık" ortamlar da olasılıkları çok fazladır. (Satranç örneği)
* Sınırlı kaynaklar (bellek, zaman, yetersiz bilgi)
* Ajan tasarımında "çevre" türünün büyük önemini yaşıyoruz.

YZ Görevi

* YZ Görevi: **agent program dizayn etmektir.**
  + ***Agent program:*** *agent function mapping percepts to actions.*
* Agent program bir mimari üzerinde çalışacaktır, bu mimari:
  + fiziksel sensörler ve actuator lerden oluşur.
* **architecture:**
  + agent = architecture +program

YZ Görevi: ajans programı dizayn etmektir.

Ajan programı: Ajan işlevleri haritalama işlemleri algılar.

Ajan programı bir mimari üzerinde çalışmaya, bu mimari:

Düzenleyiciler.

mimari:

Aracı = mimari + program

Structure Of Agents

* **Table-driven agents**
  + bir percept sequence kullanır/ bir sonraki action bulmak için bellekte action table saklar. Çok geniş lookup tablolar ile uygulanırlar.
* **Simple reflex agents**
  + condition- action kurallara dayanır ve bir kural tabanı sistemi ile uygulanır.
* **Agents with memory**
  + Geçmiş hamlelere bakarak yeni durum alırlar.
* **Agents with goals**
  + Hedef durumları tanımlayan bir amaç bilgisine sahiptirler.
* **Utility-based agents**
  + Mantıklı davranmak için kararları fayda teorisine (utility-theory) bağlı olarak alırlar.

///

* Masa başı ajanlar
* Bir algı dizisi kullanır / bir sonraki eylem. Çok geniş arşiv tablolarıyla uygulanırlar.
* Basit refleks ajanları
* Koşul eylemi tabanı sistemiyle birlikte uygulanır.
* Hafızalı Ajanlar
* Yeni durum okundu.
* Hedefleri olan ajanlar
* Bilgece kazandırmak bir yere hitab etmek.
* Hizmetlere dayalı ajanlar
* Mantıklı davranmak için kararları fayda teorisine (yarar-teori) bağlı olarak alırlar.

Skeleton agent

**function** SKELETON-AGENT (*percept*) **returns** action

static: *memory*, the agent’s memory of the world

*memory* 🡨 UPDATE-MEMORY(*memory,percept*)

*action* 🡨 CHOOSE-BEST-ACTION(*memory*)

*memory* 🡨 UPDATE-MEMORY(*memory, action*)

**return** *action*

On each invocation, the agent’s memory is updated to reflect the new percept, the best action is chosen, and the fact that the action was taken is also stored in the memory. The memory persists from one invocation to the next.

Input = Percept, not history

İskelet ajan

Işlevi SKELETON-AGENT (algılama) eylemi döndürür

    Statik: bellek, ajanın dünyadaki anısı

    Bellek ß GÜNCELLEME-BELLEK (bellek, algılama)

    Eylem ß CHOOSE-BEST-ACTION (bellek)

    Bellek ß GÜNCELLEME-BELLEK (bellek, eylem)

    Geri dönüş eylemi

Her bir çağrı üzerine, aracının belleği yeni algıyı yansıtacak şekilde güncellenir, en iyi eylem seçilir ve işlemin gerçekleştirildiği gerçeği de belleğe kaydedilir. Bellek bir başlatmadan diğerine çağrılır.

Giriş = Geçmiş değil, algılama

Summary

* Bir **ajan** bulunduğu çevreyi algılar ve harekete geçer, bir ajan program ile uygulanır.
* Bir **ideal ajan** daima beklenen performansı maksimum yapacak aksiyonları seçer.
* Bir **otonom ajan** tasarımcı tarafından belirlenen çevre özelliklerini kullanmaktansa kendi tecrübelerini kullanır.
* Bir **ajan program** algıdan (percept) 🡪aksiyona (action) eşleştirme yapar ve kendi durumunu günceller.
  + **Reflex agents** oluşan algılara anında cevap verir.
  + **Goal-based agents** amaçlara ulaşmak için belirli sırada hareket eder.
  + **Utility-based** kendi utility function(fayda fonksiyonu) değerini maksimum yapar.
* **Bilgi temsili** başarılı bir ajan dizayn için önemlidir.
* Bazı çevre (**environments)** seçenekleri diğerlerine göre ajanlar için daha zordur. En zorlu çevreler inaccessible, nondeterministic, dynamic, ve continuous özellikli olanlardır.

**Measuring problem-solving performance**

* Algoritmaların performansı 4 yoldan değerlendirilir:
* **Completeness:** algoritma çözümün olduğu durumda bir çözümü bulmayı garanti ediyor mu?
* **Optimality:** Strateji optimal çözümü buluyor mu?
* **Time complexity:** çözüme ulaşmak ne kadar sürüyor?
* **Space complexity:** arama için ne kadar bellek alanına ihtiyaç var?

Well defined problems and solutions

* Bir **problem** formal olarak 5 bileşen ile tanımlanabilir:
* **initial state:** ajanın başlangıç durumu. Örneğin haritada Arad da ilk durumda bulunmak
* **Actions:** ajanın gerçekleştirebileceği olasık aksiyonların tanımı. Verilen bir s durumuna göre s’de yerine getirilen aksiyonlar kümesi.örneğin arad başlangıç durumunda yerine getirilebilecek aksiyonlar {Go(Sibiu), Go(Timisoara), Go(Zerind)}.
* Her aksiyonun ne yaptığı tanımı; **transition model**, bir function RESULT(s, a) ile tanımlanır. S durumunda iken a aksiyonunu alırsa sonuçları döndürür.
* Örneğin:  
   RESULT(In(Arad), Go(Zerind)) = In(Zerind) .
* **Initial state+actions+transition model: state space model**
* **Goal test:** verilen bir durumun amaç durum olup olmadığını test eder. Örönek amaç durum: {In(Bucharest)}.
* **Path cost:** her yola bir sayısal cost değeri atar. Problem-solving agent kendi persormansını ölçmek için bir cost fonksiyonu seçer.
* **Step cost:** bir adımdan diğerine geçerken cost değeri.   
    
    
    
    
    
    
    
  Infrastructure for search algorithms
* Arama algoritmalarında arama ağacında n. düğümü temsil etmek üzere kurulan veri yapısı şunları içerir:
* n.STATE: adım uzayında düğüme ait durum bilgisi
* n.PARENT-NODE: düğümün parent düğümü
* n.ACTION: düğümü yaratmak için parent düğüme uygulanan aksiyon
* n.PATH-COST: cost *g(n) ile temsil edilir,başlangıç durumundan düğüme olan yol bilgisi*
* n.DEPTH: başlangıç durumdan düğüme olan adımların sayısı

Frontier / Closed List

Genişletilen düğümlerin hatırlanması gerekir, yeniden genişletmenin önüne geçmek için . Bunun için, **explored set** adı verilen bir veri yapısı tanımlanır. (**closed list** olarak da adlandırılır) ve genişletilen her düğümü içerir.

Graflar

* YZ problemlerinin çoğunda durum uzayının ve çözüm ağacının gösterilmesinde graflar (1822, J.J.Sylvester) kullanılır.
* Graf:
  + Yalın graf: iki ayrı düğüm tek bir kenar ile birbirine bağlanır
  + Bağlantılı graf: iki ayrı düğüm birden çok kenar bağlantısı içerisindedir.
* Komşu düğüm: birbiri ile kenar bağlantısı olan düğümlerdir. Grafın herhangi düğüm komşuları sayısı onun derecesini belirler.
* P(v0,vk) yolu ilk v0 ve son vk düğümleri arasındaki yol olsun. Kapalı bir yol varsa buna döngü denir.
* Döngü içermeyen bağlantılı graflar ağaç olarak adlandırılır.
* YZ de graflar: durum uzayını, min. Yol gibi nesneler arası ilişkileri, doğal dil işleme çözümlerini vb. ifade etmek için kullanılır.

Problem Çözümü

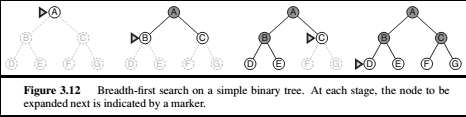
* **1. Problemi Anlama:** girişlerin ve değişkenlerin belirlenmesi, problemin genel resminin çizilmesi ve küçük parçalara ayrılması
* **2. Çözümü planlama:** benzer problemlerin aranması, değişkenler, veriler ve girişler arasındaki ilişkilerin belirlenmesi
* **3. Seçilen planı uygulama:** çözümün her aşamasının uygulanması ve optimum çözümün aranması
* **4. Sonuçların değerlendirilmesi**

Durum Uzayı

* YZ problemleri sonlu durumlar kümesinden oluşur.
* Durum uzayı kümesinin elemanları graf yapısı ile temsil edilir.
* Durumlar: düğümleri,
* Durumlardan durumlara izinli geçişler: yolları ifade eder.
* Çözüm: graftaki minimum yolun bulunması problemidir.

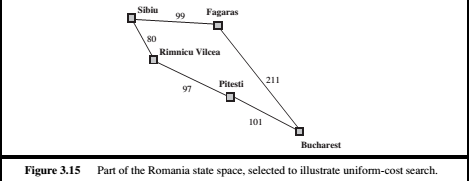
Breadth-first Search

* aranan nokta: J
* **ağaca göre tüm noktaları gezmemiz gerekmektedir.**
* Her zaman köke en yakın çözümü verir.
* Her seviyede b düğüm genişletilirse ve d derinliğe kadar arama devam ederse karmaşıklığı:



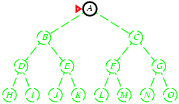
Uniform Cost Search

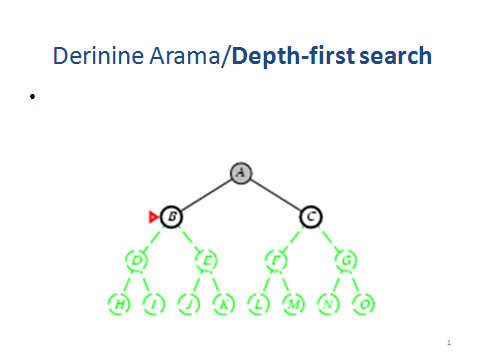
* Root a en yakın (en sığ) düğümü genişletmek yerine Uniform-cost arama, en düşük path cost g(n)’e sahip n düğümünü genişletir.
* Bu frontier kuyruğunu g fonksiyonuna göre sıralanmış şekilde tutarak gerçekleştirilir (priority queue ordered by g).
* Örnek problem: sibiu’dan Bucrahest’e gitmek
* The least-cost node, Rimnicu Vilcea, is expanded next, adding Pitesti with cost 80 + 97 = 177. The least-cost node is now Fagaras, so it is expanded, adding Bucharest with cost 99 + 211 = 310. Now a goal node has been generated, but uniform-cost search keeps going, choosing Pitesti for expansion and adding a second path to Bucharest with cost 80+97+101 = 278. Now the algorithm checks to see if this new path is better than the old one; it is, so the old one is discarded. Bucharest, now with g-cost 278, is selected for expansion and the solution is returned.
* Uniform-cost search does not care about the *number* of steps a path has, but only about their total cost.



Depth-first search

* En derindeki genişlenmemiş düğümü genişletmeli
* Breadth first frontier listesine FİFO olarak erişirken, Depth first search LİFO olarak erişiyor





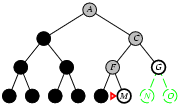




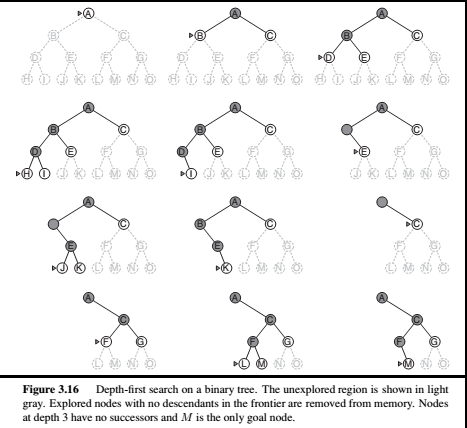








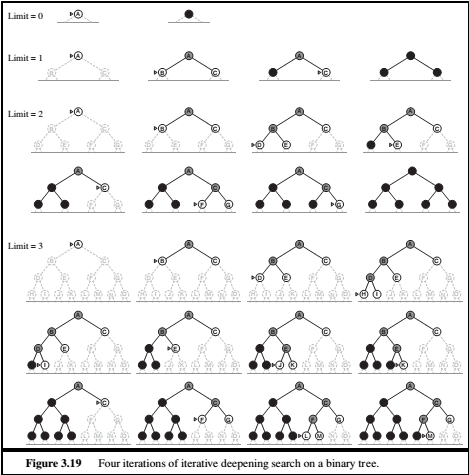
Depth First Search

* depth-first search yönteminin breadth-first search’ e göre görünen çok büyük avantajı yok, O halde neden bu aramaya ihtiyaç var?
* Sebebi: space complexity. Yaprak düğüme kadar taranmış alt ağaç uzaydan eksiltilebilir.
* For a state space with branching factor b and maximum depth m, depth-first search requires storage of only O(bm) nodes.
* 

Depth Limited Search

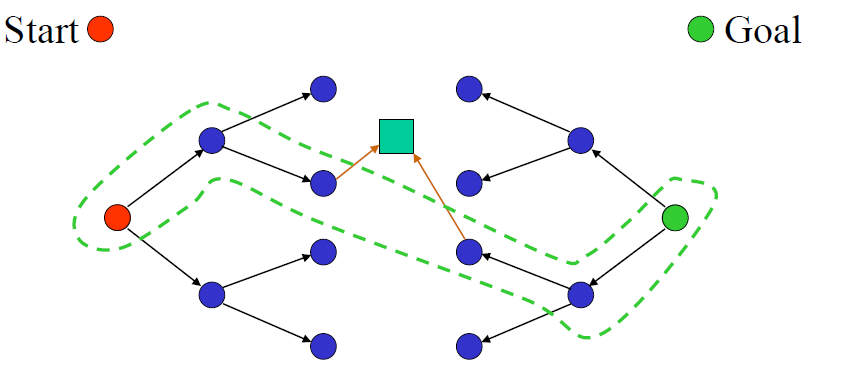
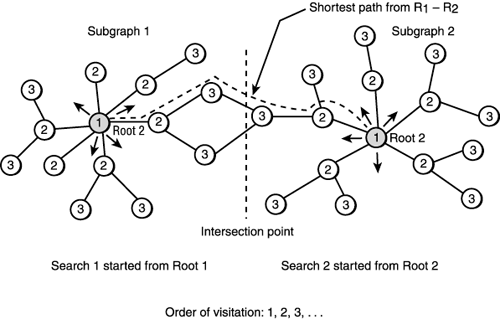
* l<d derinliğinde kesilen DFS’dir.
* Zaman karmaşıklığı: O(bI) ve yer karmaşıklığı O(bl)
* Depth first search ‘de l=sonsuz
* Üç olası sonuç vardır :
  + Çözüme ulaşılır (Solution)
  + Çözüme ulaşılamaz (Failure)
  + Cutoff (derinlik sınırları içinde sonuca ulaşılamaz)

Iterative Deepening Search



Bidirectional search

* Temel düşünce:Eş zamanlı arama işletmek
  + Birincisi başlangıç durumundan ileriye doğru
  + İkincisi hedef durumdan geriye doğru
* İki arama kesişince algoritma sonlanır.
* The motivation is that bd/2 + bd/2 is much less than bd
* Arama uzayının çok geniş olduğu uygulamalarda zaman karmaşıklığını azalttığından çözüm olabilir (sosyal paylaşım siteleri gibi)



* Arama Yöntemleri: Her aşamada hangi düğümün açılacağını belirler.
  + Blind Strategies (Bilgiye dayanmayan arama): Durum bilgisinden yararlanmaz.
  + Heuristic Strategies (Sezgisel arama): Durum bilgisinden yararlanır. Daha umut verici hareketi tercih eder.

Karşılaştırma: Sezgisel olmayan aramalar

* BFS, tamdır, optimaldir ancak yer karmaşıklığı yüksektir.
* DFS, yer karmaşıklığı etkindir, ancak tam da değildir, optimal de değildir.
* Iterative Deepening Search, yaklaşık olarak optimaldir.

Heuristic Search  
(Sezgisel Arama)

* Sezgisel olmayan arama yöntemlerinde düğümlerin açılmasında kullanılan yöntem, başlangıç düğümünden olan uzaklık bilgisine dayanıyordu.
* Hedefe olan uzaklık kestirilirse ne olur?
* Gerçek uzaklık tam olarak bilinmese de kestirilebilir. Bu tahmine, **Heuristic (Sezgi)** yani **h(n)** denilir.
* YZ da sezgisellik: insanın sezgisel davranışları makinelere nasıl aktarılabilir ve problem çözümünde nasıl kullanılabilir?
* Sezgisellik: problem karmaşıklık içerdiğinde çözüm için yolun bulunmasındaki yardımcı anahtardır.
* İyi seçilmiş anahtar ile çözüm kolayca bulunurken kötü seçilmiş anahtar çözüme ulaşmayı güçleştirir veya geciktirir.

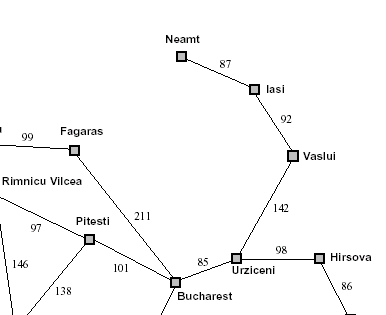
Admissibility of a Heuristic

* c(n): n düğümünden amaç düğümüne optimum yol maliyetini temsil etsin.
* h(n) <= c(n) (tüm düğümler için) ise h(n) heuristic fonksiyonu kabul edilebilirdir.
* Örnek: routing networklerde öklit uzaklığı

Sıralı Arama

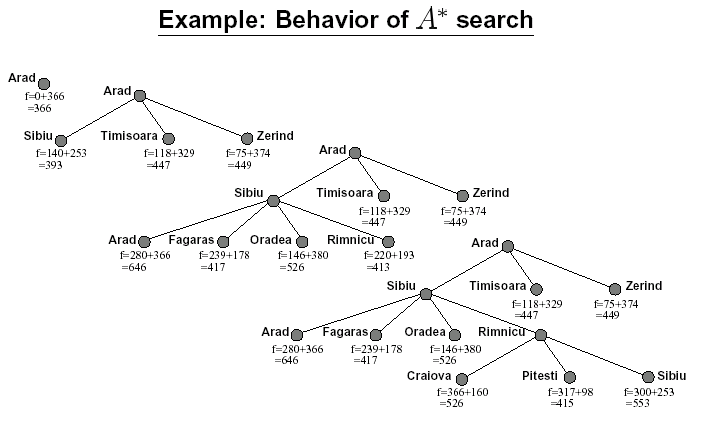
* Algoritma adımları genel olarak:
  + Başlangıç durum çözüm ağacının köküne yerleştirilir ve geçerli gidişlerler (operatörlerle) bu durumdan mümkün olan yeni durumlar (çocuklar) elde edilir.
  + Çocuk düğümler için f(x) fonksiyon değerlendirmesi yapılarak minimum değerli düğüm seçilir. Aynı değere sahip düğümlerden herhangi biri seçilir
  + Açılan min değerli düğümün benzer şekilde çocuk düğümleri değerlendirilir. O anki düğümden çocuk düğüm elde edilemiyorsa ve henüz hedefe ulaşılmamışsa üst seviyeye geri dönülerek arama yönü minimum olan fonksiyon yönüne doğru değiştirilir.
  + İşlemler hedefe ulaşılıncaya kadar tekrarlanır. Hedef bulununca başlangıç düğümünden hedefe olan yol saklanır.
* Greedy Yöntemleri:
  + uzun vadeli sonuçları önemsemeden, kısa vadeli avantajları en iyi duruma getirir.
  + Tam değildir
  + Optimal değildir.
  + Zaman ve Yer Karmaşıklıkları kötüdür (üstel)

Greedy Best First Search

* **Greedy best-first search amaca en yakın düğümü genişletmeye çalışır.** Düğümleri bir heuristic function ile değerlendirir, f(n) = h(n).
* Problem haritada Arada dan Bucharest e yol bulma problemi.
* Sezgisel fonksiyon: **straight** **line distance** heuristic
* Solution Path: Arada-Sibiu-Fagaras-Bucharest.
* Her adımda amaç durumuna en çok yaklaşılacak yeni bir durum aradığı için Aç Gözlü arama olarak adlandırılır.
* Optimum çözümü aramaz.
* Çözüm tam değildir. Örneğin problem : Iasi den Fagaras a gitmek olsaydı çözüm bulunamazdı. Neamt düğümü genişletildiğinde Frontier list: Iasi şeklinde olacaktı. Iasi yeniden ziyaret edilecek ve sonra tekrar Neamt a bir loop oluşacaktı.  
  

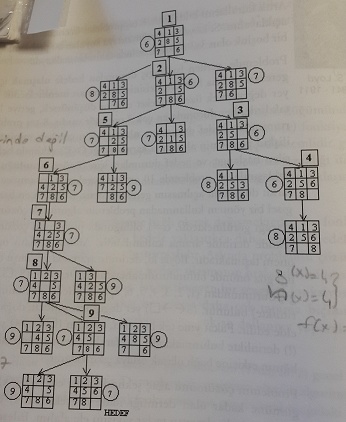
A\* Search

* Best-first Search’ün en büyük dezavantajı, uzak görüşlü olmamasıdır.
* Başlangıç düğümü: s düğümü
* S düğümünden X düğümüne kadar yol:
  + g(x): şu anki duruma kadar maliyet fonksiyonu
* X düğümünden hedef f düğümüne kadar yol:
  + h(x): uygun sezgi fonksiyonu olsun.
* g(x): x durumunun gerçek olan o anki değeri
* h(x): x düğümünden çözüme olan gidişlerin sezgisel değeri
* Uygun olması, iyimser olması yani hedefe götüren maliyeti hiçbir zaman gerçek değerinden daha yüksek değerde kestirememesi demektir.
* A\* Algoritması:
  + - f(x)=g(x)+h(x)
* Hem optimal hem de tamdır.



A\* Search 8-taş problemi için

* H(x) sezgisel fonksiyonun tasarımı probleme göre değişmektedir.
* **Örnekte 2. sırada** açılan durum için derinlik: g(x)=1
* Hedef duruma göre yerinde olmayan taşların sayısı:h(x)=5
* O halde f(x)=g(x)+h(x)=1+5=6
* **7. durum için** g(x)=4, h(x)=3 ve f(x)=7
* Karelerin üzerindeki sayılar sıra ile açılan düğümleri,
* Yan tarafındaki sayılar sezgisel skorları ifade eder.
* Şekildeki gibi h(x)=0 olduğunda algoritma durursa çok fazla derinine arama yapılabilir.
* Bu nedenle A\* algoritmasının 8-taş problemine uygulanmasında farklı sezgisel h’(x) fonksiyonları tanımlanabilir.
* 1- h’(x): ele alınan durumla hedef durum karşılaştırıldığında yerinde olmayan taşların sayısı
* 2-h’(x):taşların hedefteki yerlerine uzaklıkları toplamı



4-15