

Hatırlatma

- Bilgisiz Arama Yöntemleri
 - Genişlik-öncelikli (Breadth-first)
 - Eşit-maliyetli (Uniform-cost)
 - Derinlik-öncelikli (Depth-first)
 - Derinlik-sınırlı (Depth-limited)
 - Yinelemeli Derinleşen (Iterative deepening)
 - İki Yönlü (Bi-directional)

Bilgili Arama

- Bu yöntemler, problem hakkındaki bilgiden de yararlanarak arama yaparlar
- Bunun için hedefe olan tahmini uzaklık (maliyet) kullanılır
- Aramayı yönetmek için sezgiseller kullanılır
 - Bu tahmini gerçekleyen fonksiyona Sezgisel (Heuristic) fonksiyonu ($h(n)$) denilir

Bilgili Arama Stratejileri

- En iyi öncelikli (Best-first)
- Açı gözlü (Greedy)
- A*
- IDA* (Iterative Deepening A*)
- SMA* (Simplified Memory Bounded A*)

En iyi öncelikli (Best-first) arama

- Genel yaklaşım: Her durum için, o durumun istenebilirliğini (**desirability**) tarif edecek bir tahmin fonksiyonu $f(n)$ kullanılır
- En çok istenen ilerletilmemiş durum ilerletilir

Uygulama:

- Durumları (düğümleri) istenebilirlikleri azalacak şekilde sırala: Priority Queue

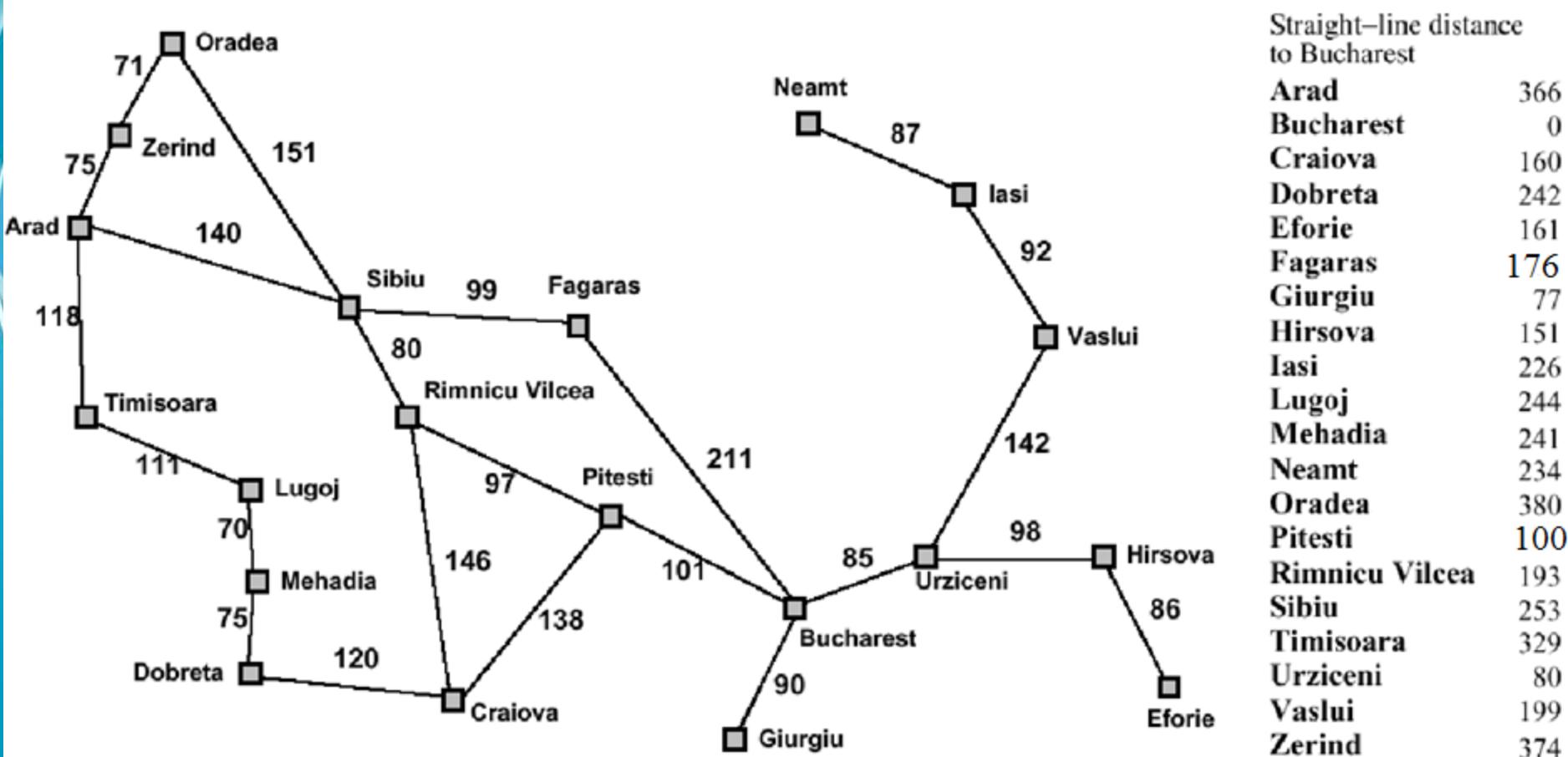
Özel durumlar:

- Açı gözlü arama (greedy search)
- A*arama

Aç Gözülü (Greedy) Arama

- Hedef durumuna en yakın olduğu tahmin edilen/düşünülen durumu ilerletir
- Tahmin fonksiyonu
 - $f(n) = h(n) = n$ durumundan hedef duruma kadar olan, sezgisele göre hesaplanmış masraf
- Örnek: $h_{SLD}(n) = n$ şehrinden İzmir'e kadar olan kuş uçuşu mesafe
 - *SLD: Straight Line Distance*
 - $h(n) = 0$, n hedef ise

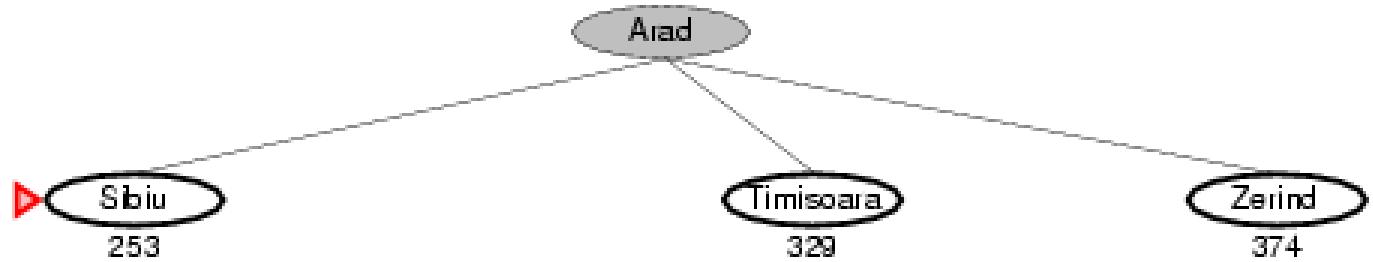
Örnek: Arad'dan Bükreş'e ulaşmak



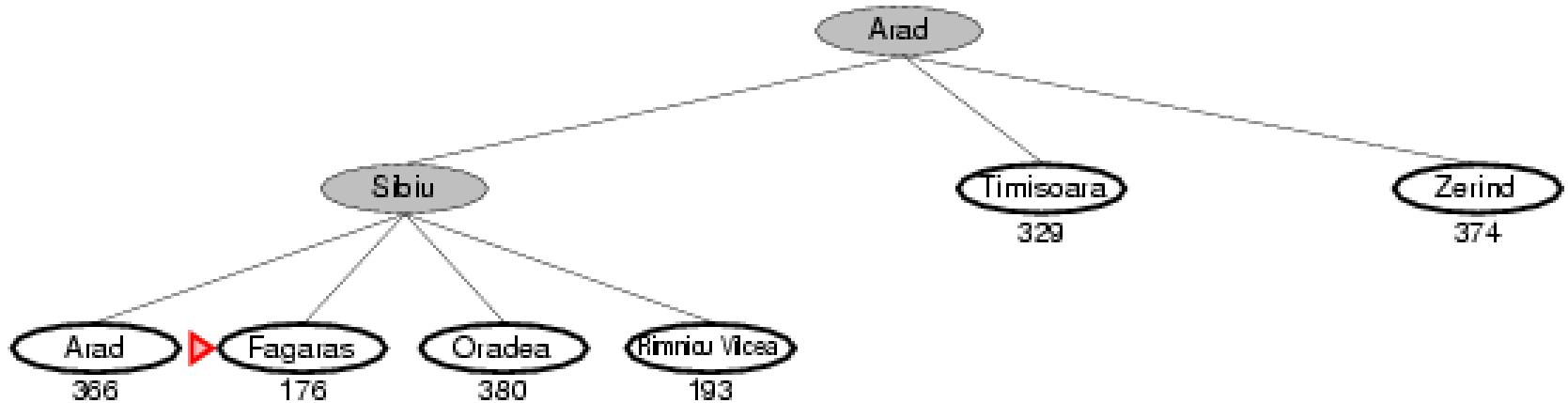
Greedy Çözümü



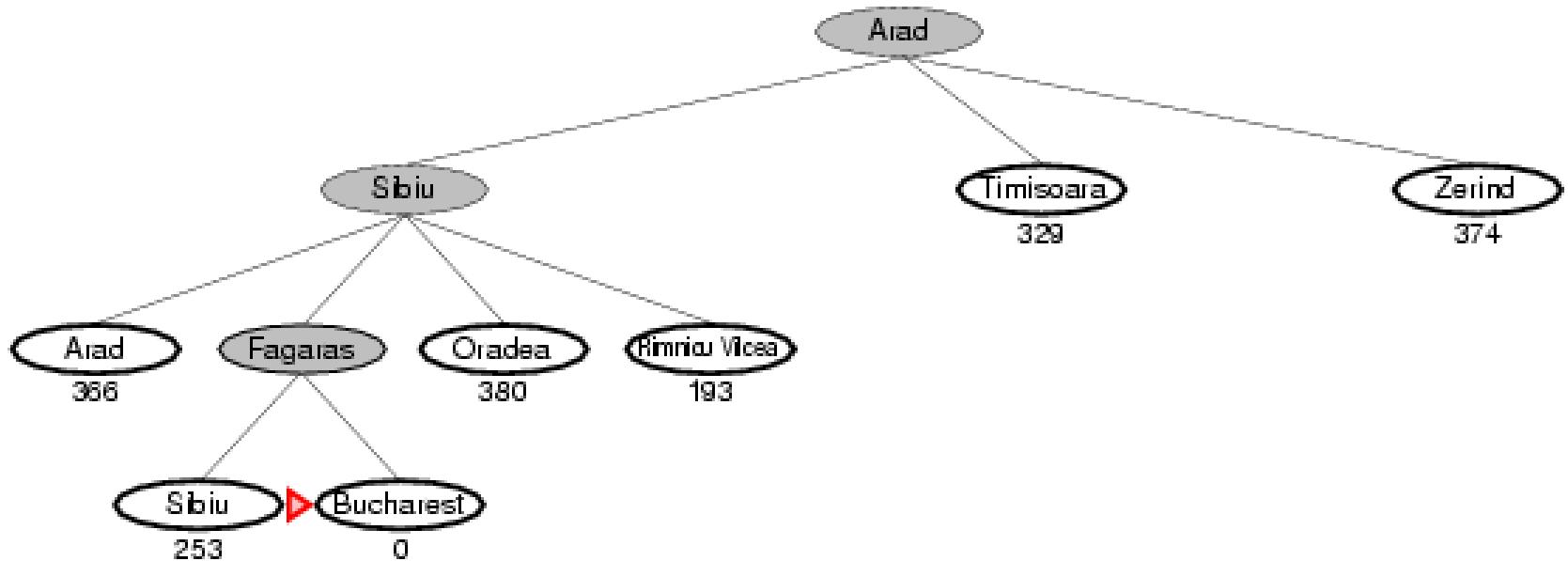
Greedy Çözümü



Greedy Çözümü

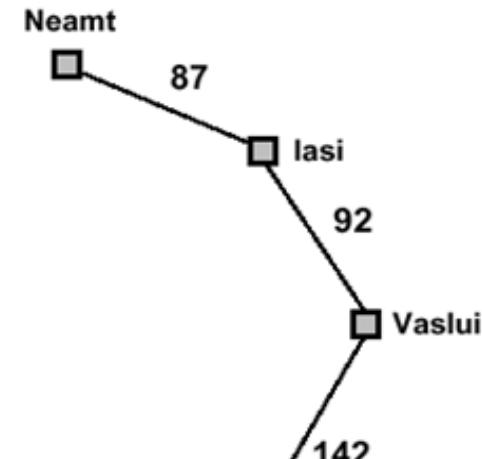


Greedy Çözümü



Aç Gözülü: Özellikleri

- Completeness: Hayır
 - Döngülerde takılabilir:
 - Iasi > Neamt > Iasi > Neamt > ...
- Time complexity: $O(b^m)$
 - Fakat iyi bir heuristic bu zamanı çok azaltabilir
- Space complexity: $O(b^m)$
 - Değerleri karşılaştırmak için bütün düğümleri bellekte tutar
- Optimality: Hayır



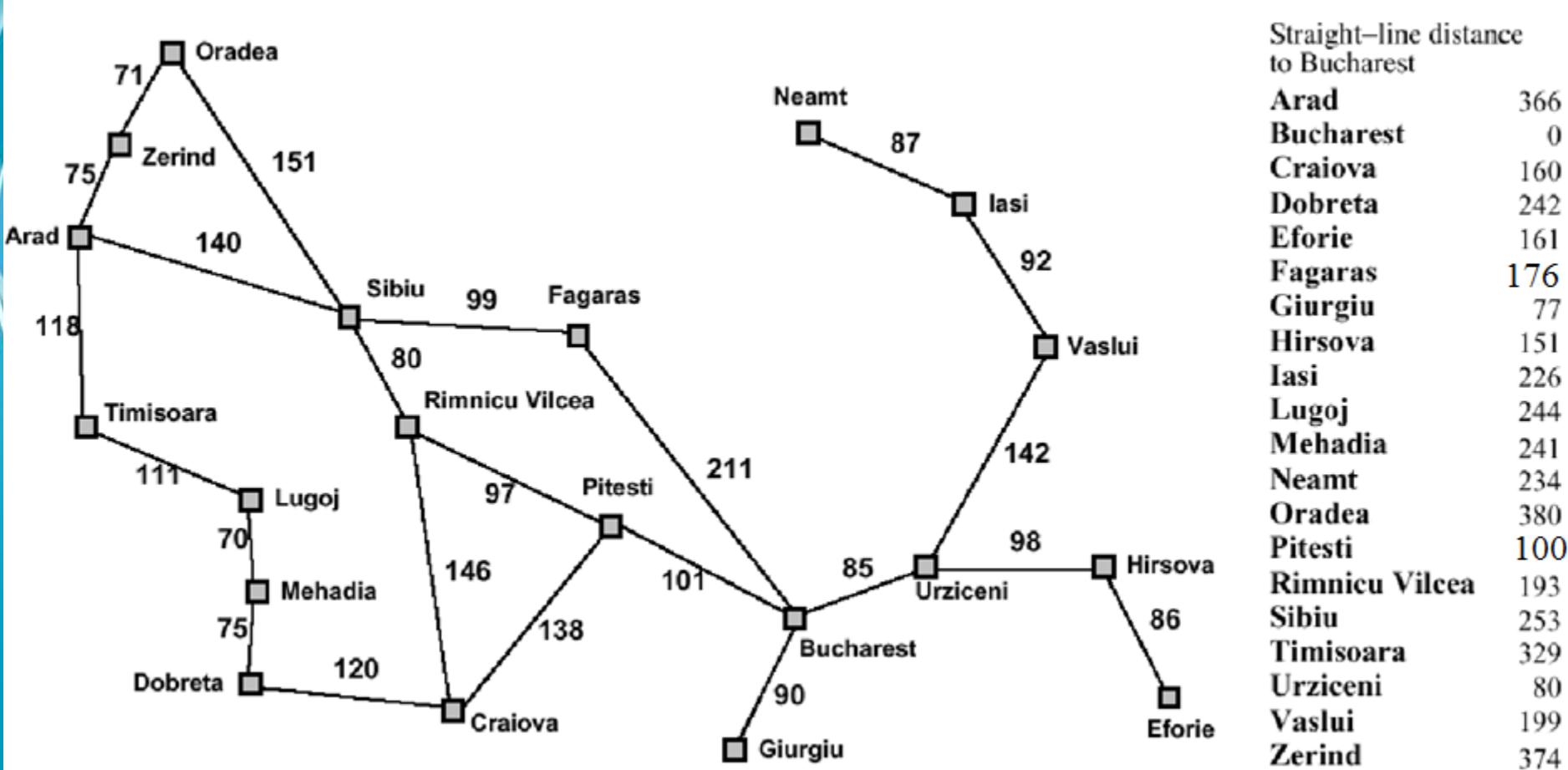
b: dallanma faktörü

m: arama ağacının max derinliği

A* Arama

- Ana Fikir: Masraflı olan yolları ilerletmekten kaçın
- Değerlendirme fonksiyonu: $f(n) = g(n) + h(n)$
 - $g(n)$: n 'e ulaşmak için şimdije kadarki maliyet
 - $h(n)$: n 'den hedefe olan tahmini maliyet
 - $f(n)$: n 'den hedefe olan yolun tahmini toplam maliyeti
- Teorem: A* arama optimaldır

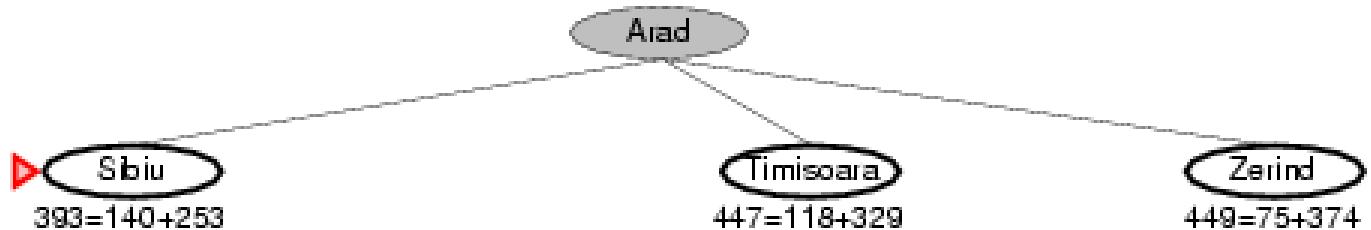
Örnek: Arad'dan Bükreş'e ulaşmak



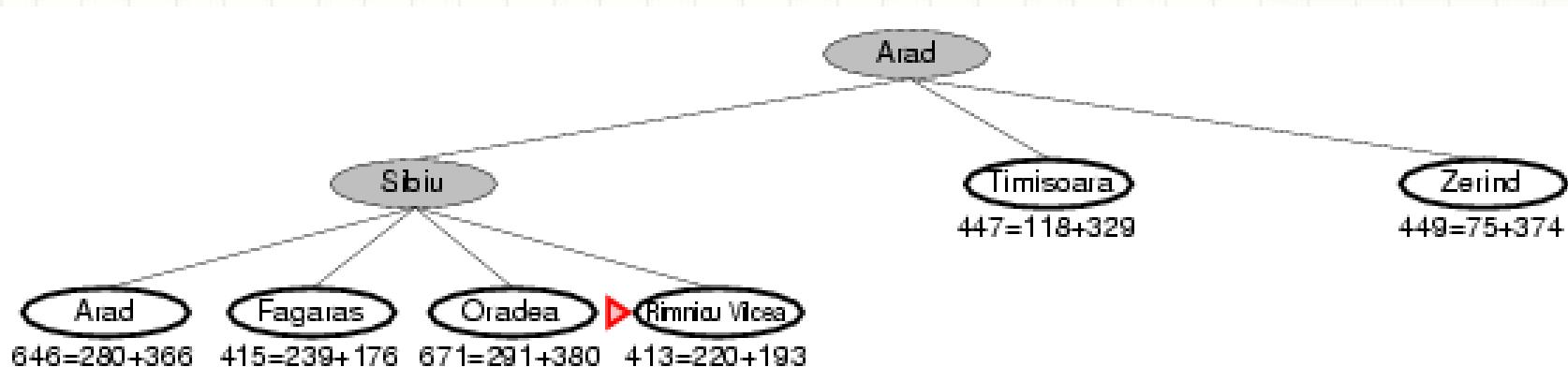
A* Çözümü

► Aiad
 $366=0+366$

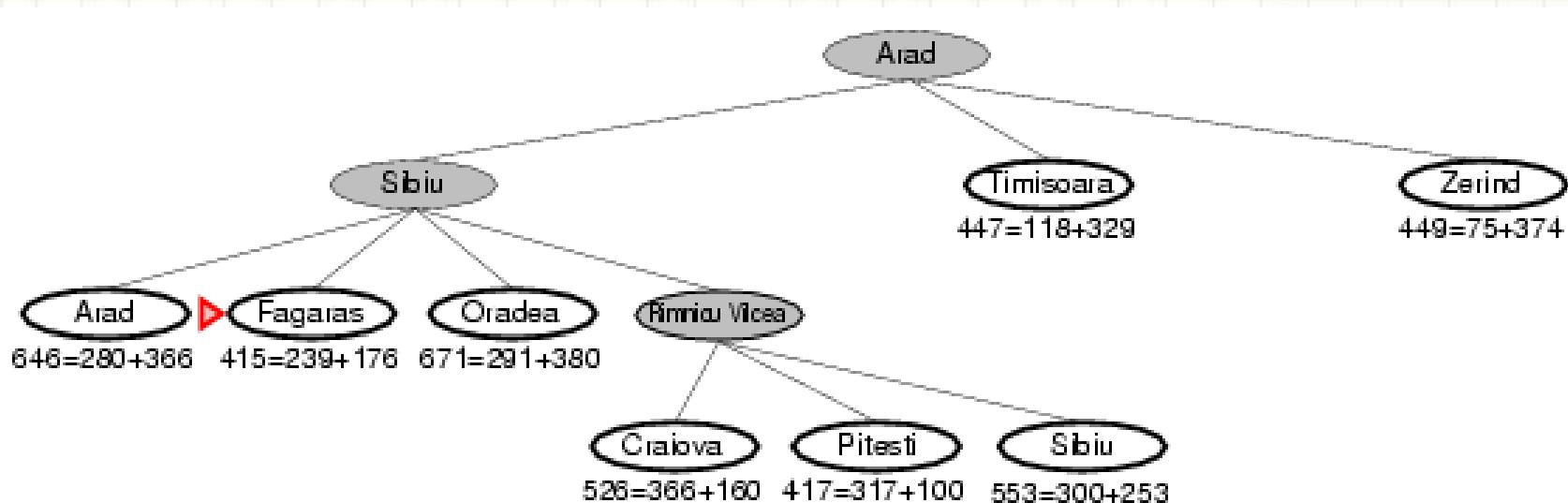
A* Çözümü



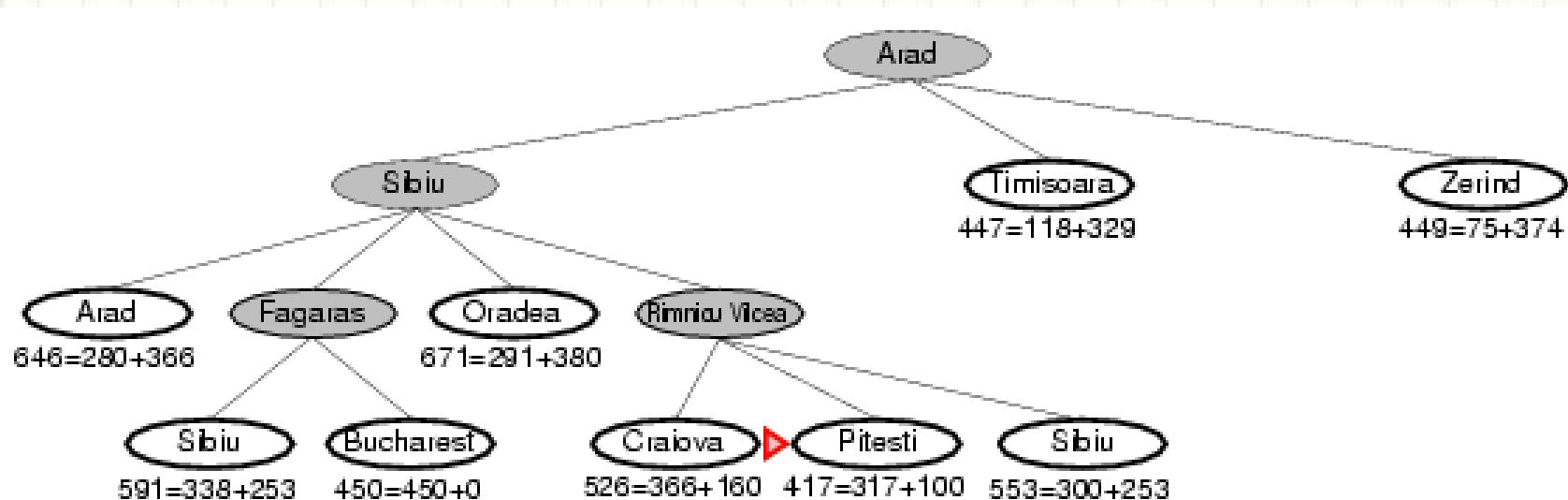
A* Çözümü



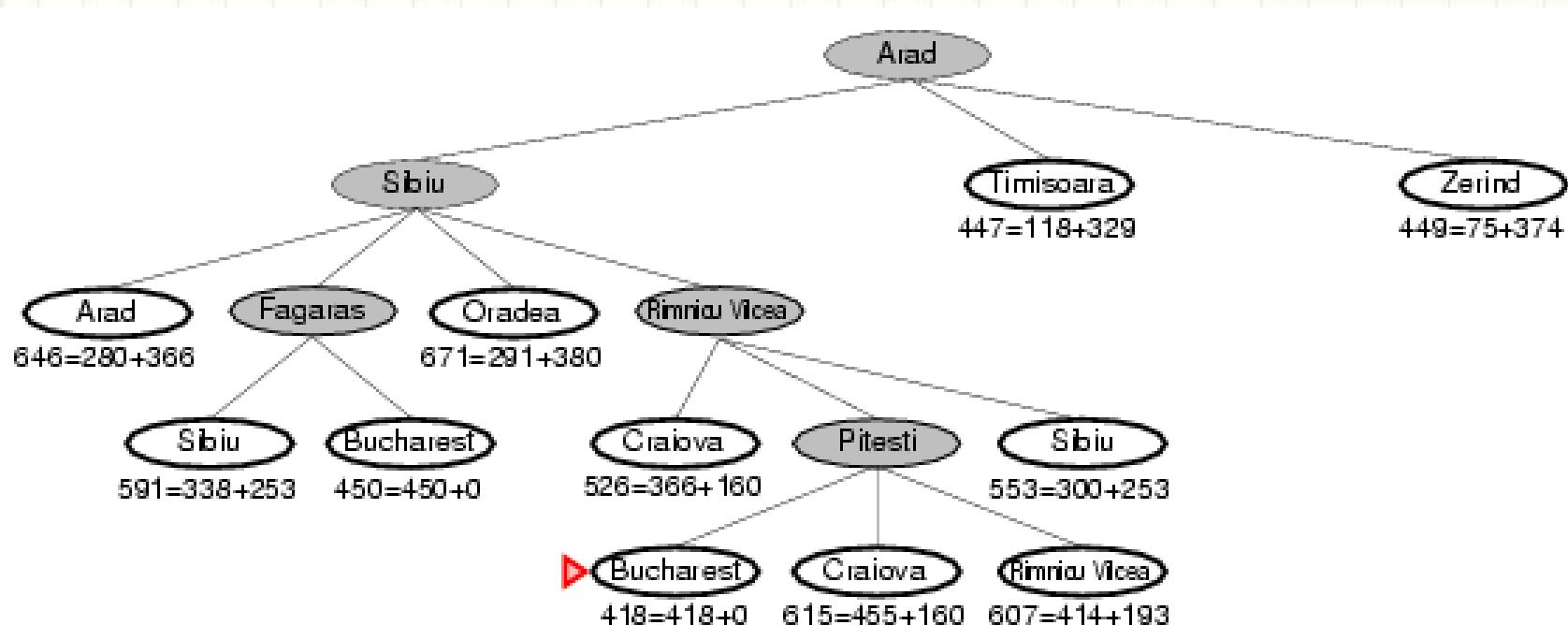
A* Çözümü



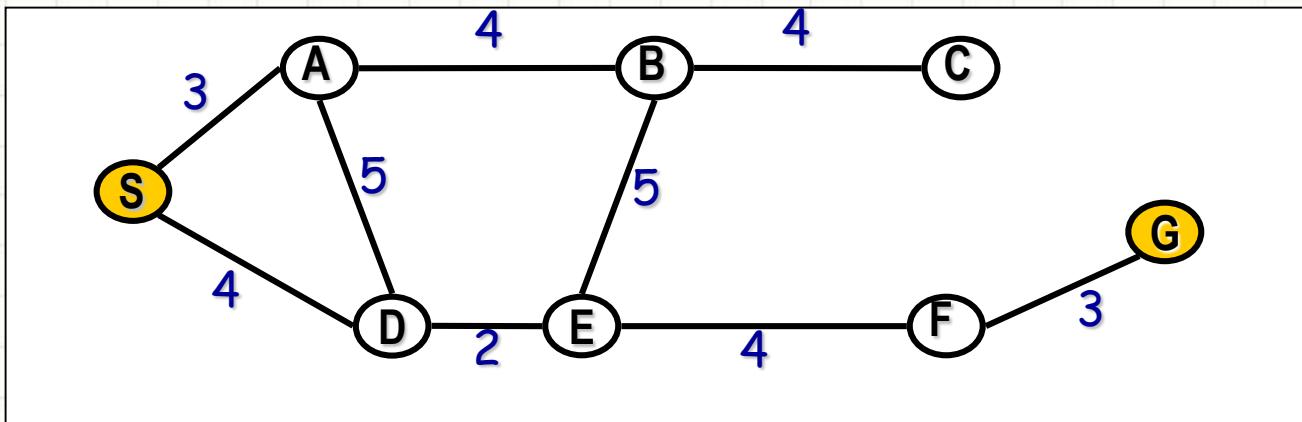
A* Çözümü



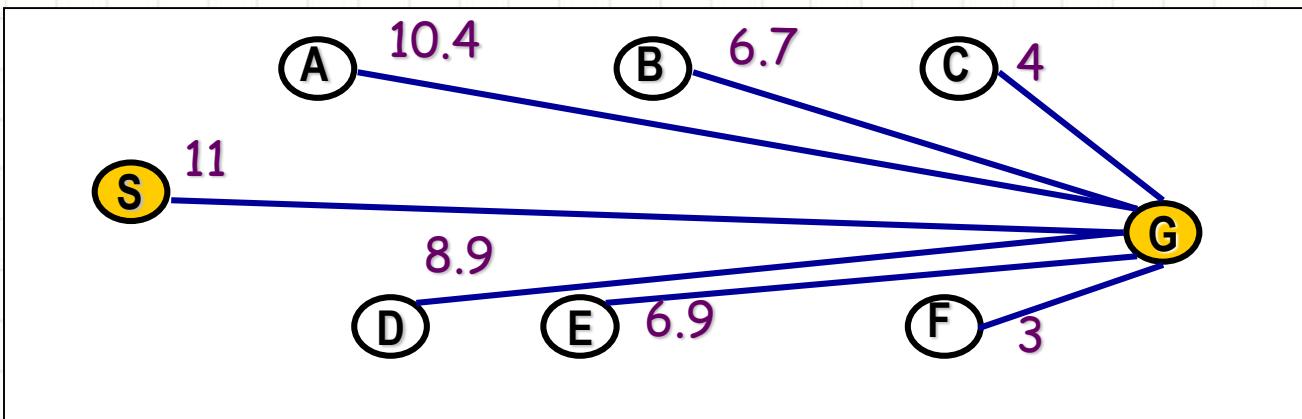
A* Çözümü

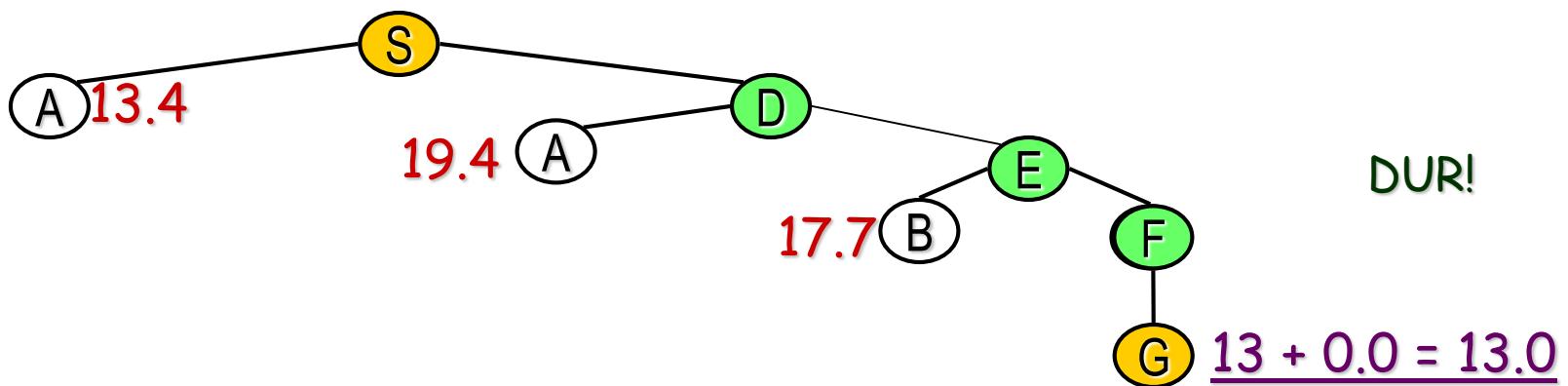
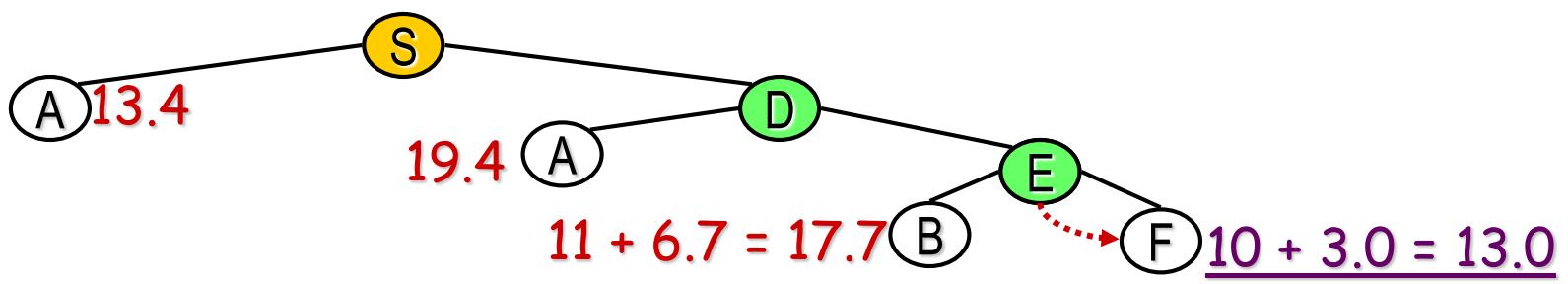
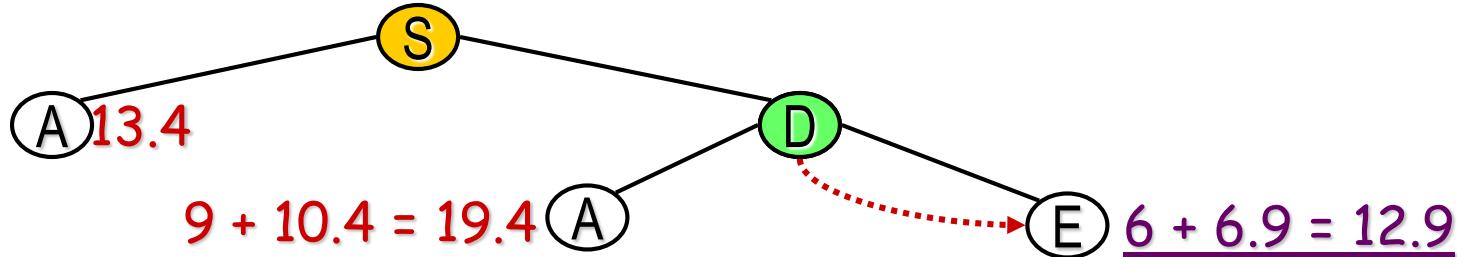
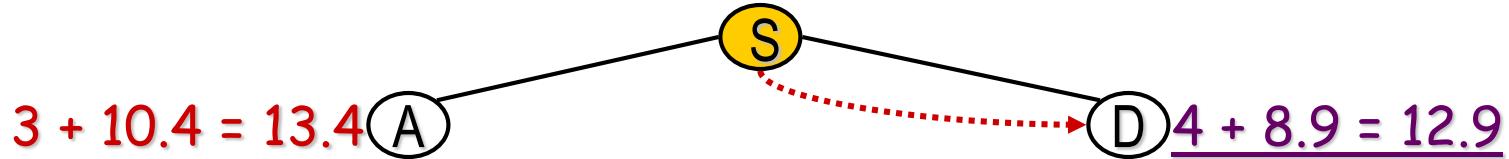


Örnek-2



- $h(T) = T'$ den G' ye düz yol uzaklığı





A* Özellikleri

- Completeness: Evet
 - Döngülere girmediği için
- Time complexity: Kullanılan sezgisele göre değişebilir
 - En kötü durum (worst case): Çözüm yolunun uzunluğunda eksponansiyel
- Space complexity: $O(b^m)$
 - Bütün düğümleri bellekte tutar
- Optimality: Evet

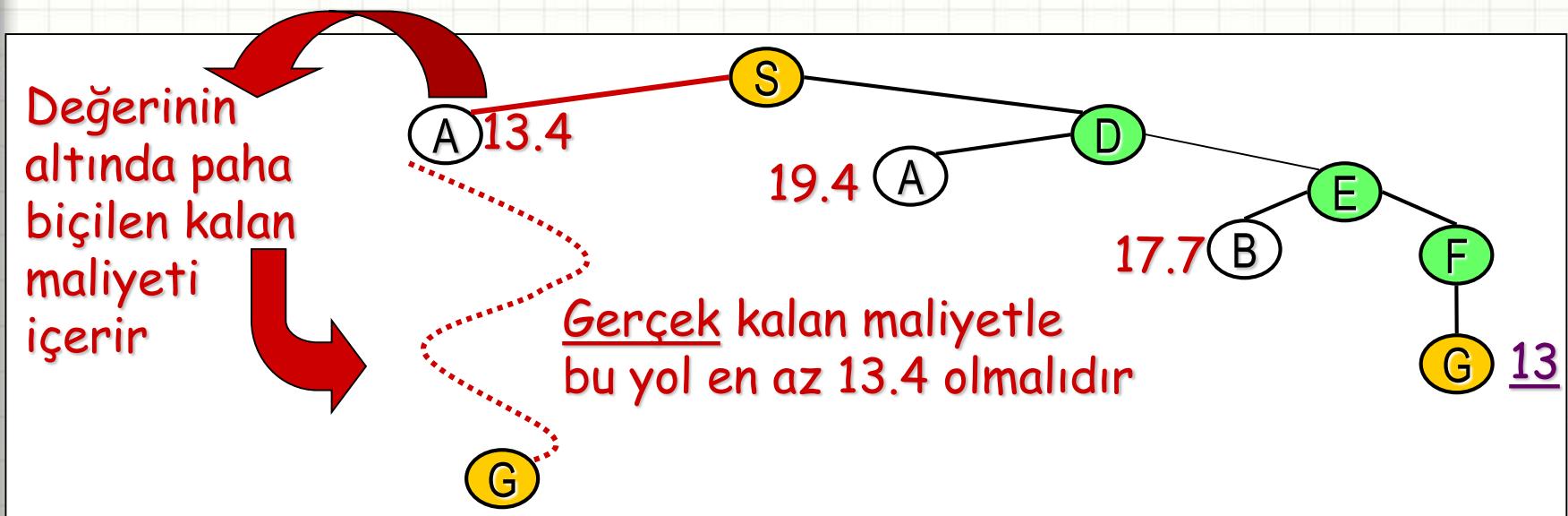
b: dallanma faktörü

m: arama ağacının max derinliği

Optimallik

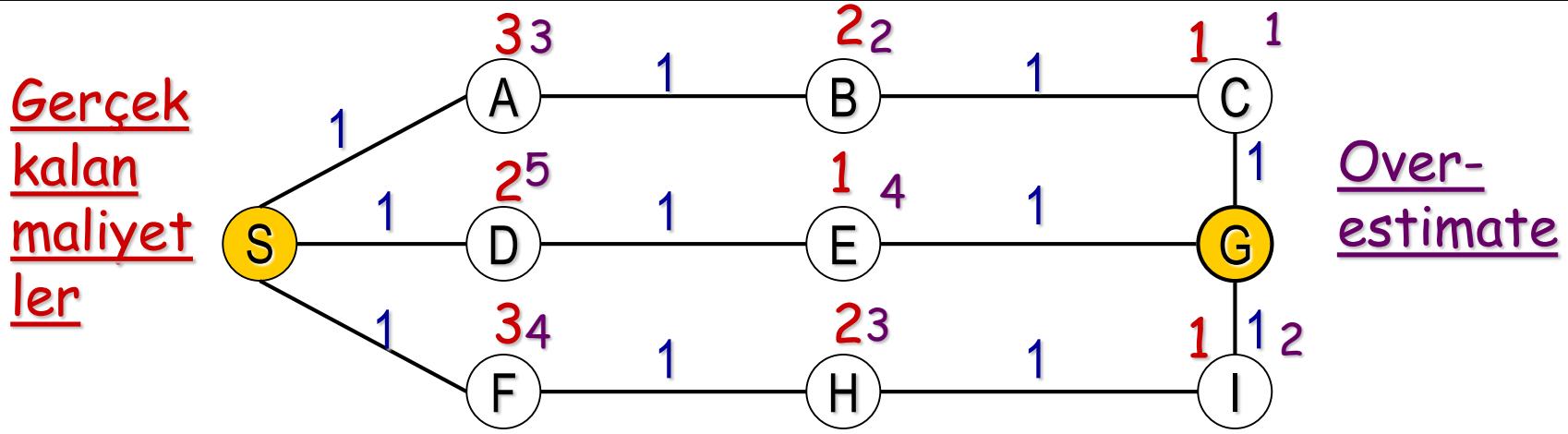
IF for tüm T : $h(T)$ ile hedef düşüme kalan maliyetin altında paha biçilmişse (**UNDERestimate**)
THEN A* optimal dir.

- $h_{SLD}(n)$: her zaman gerçek uzaklıktan küçük olduğundan **kabul edilebilir** sezgiseldir

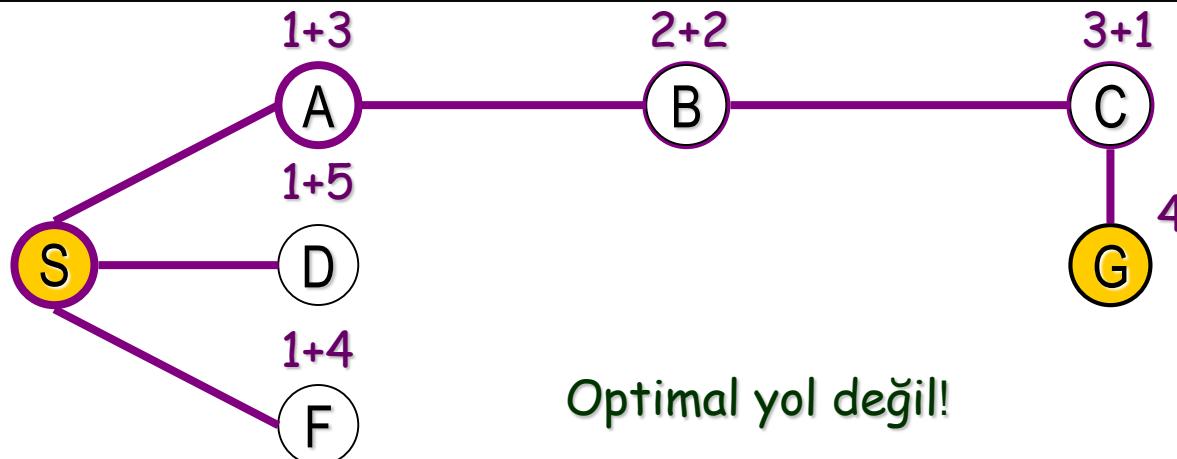


Underestimate ile ilgili

Örnek:

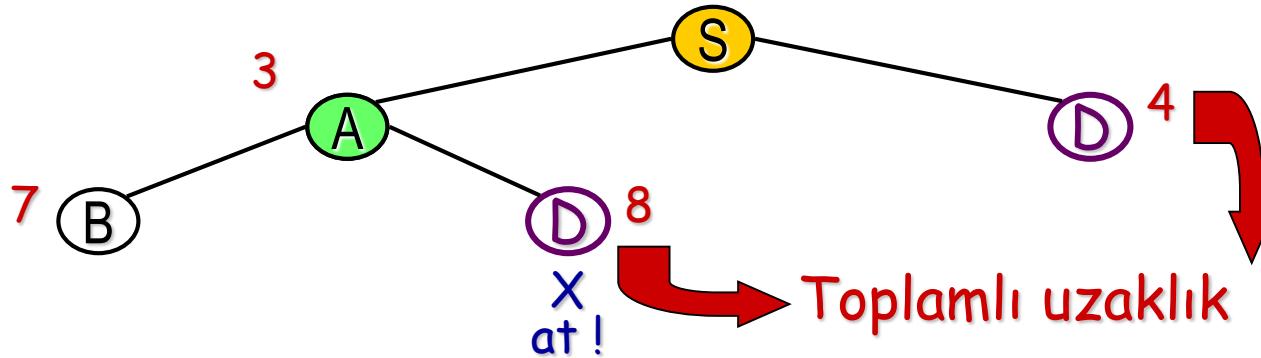


- h bir underestimate **DEĞİLSE**



İyileştirme: Yol silme

- Artık yolları at:

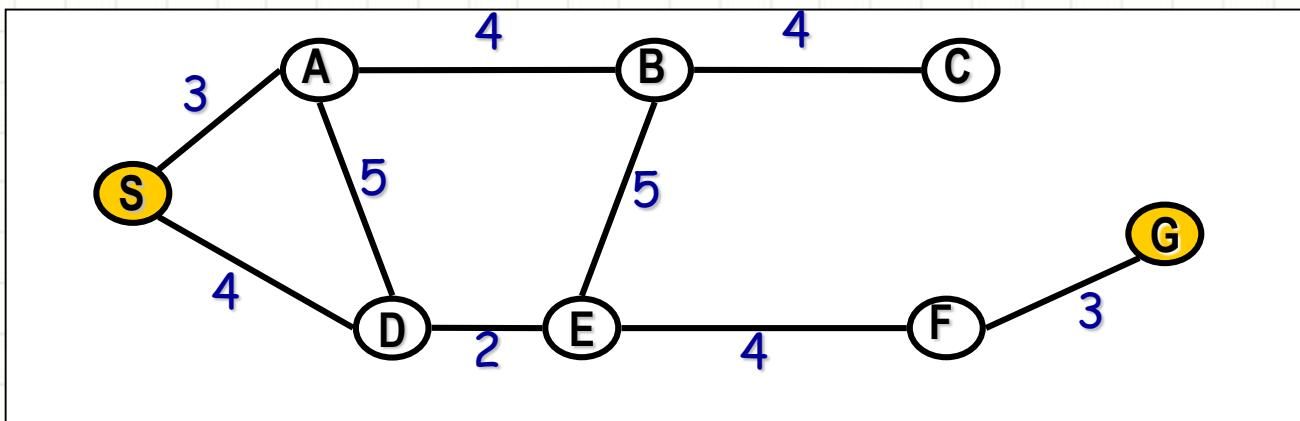


- İlke:

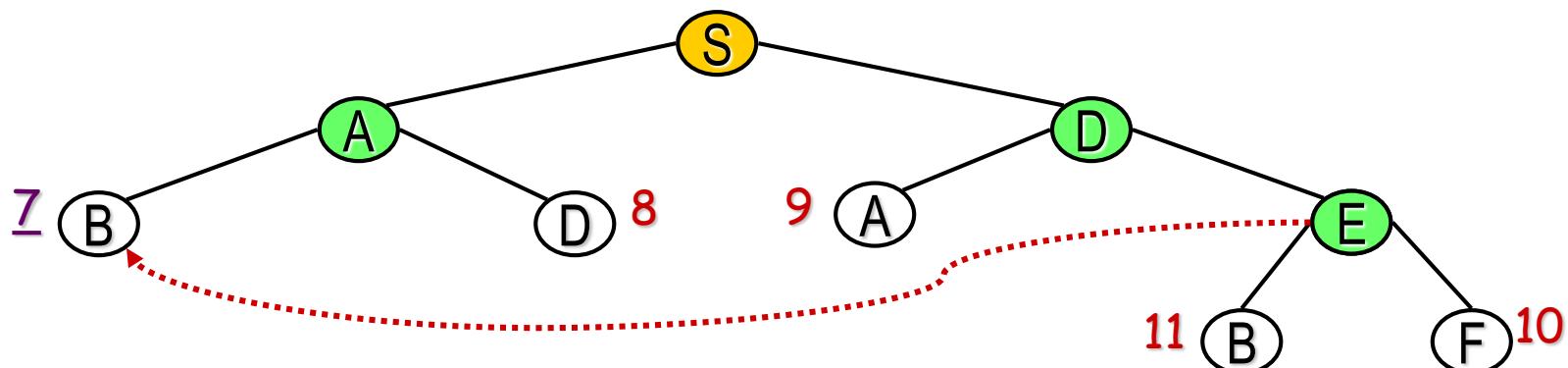
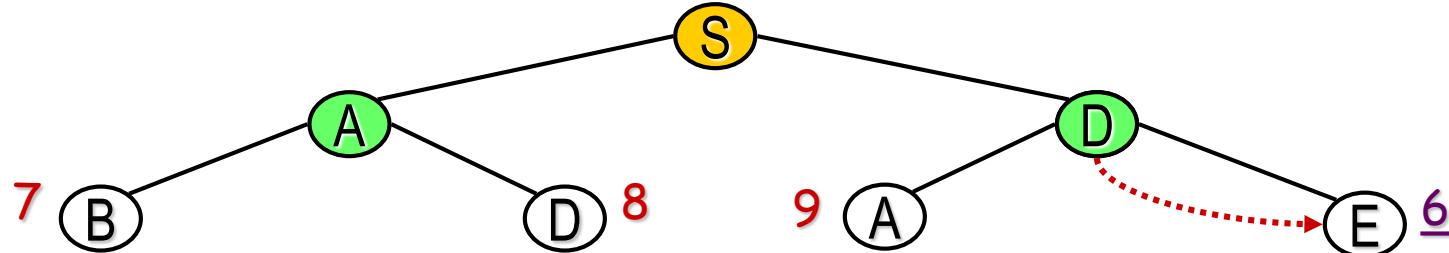
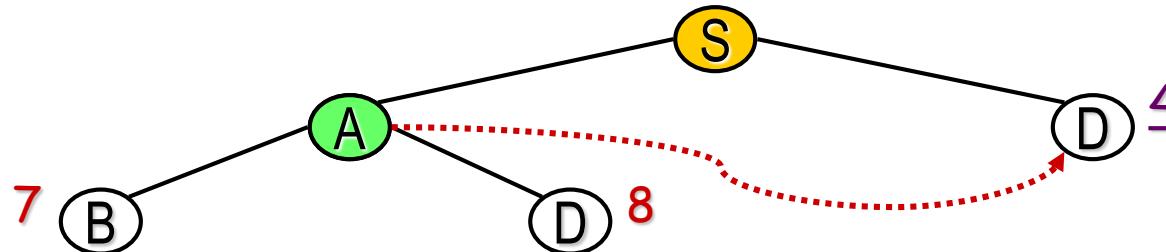
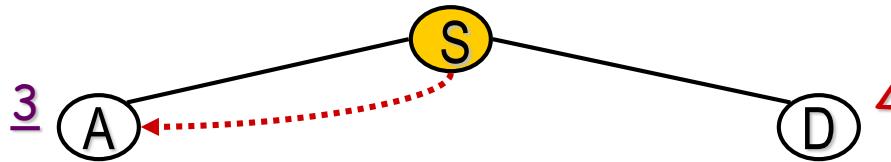
S' ten G' ye I' dan geçerek minimum maliyet =
(S' ten I' ya minimum maliyet)
+ (I' dan G' ye minimum maliyet)

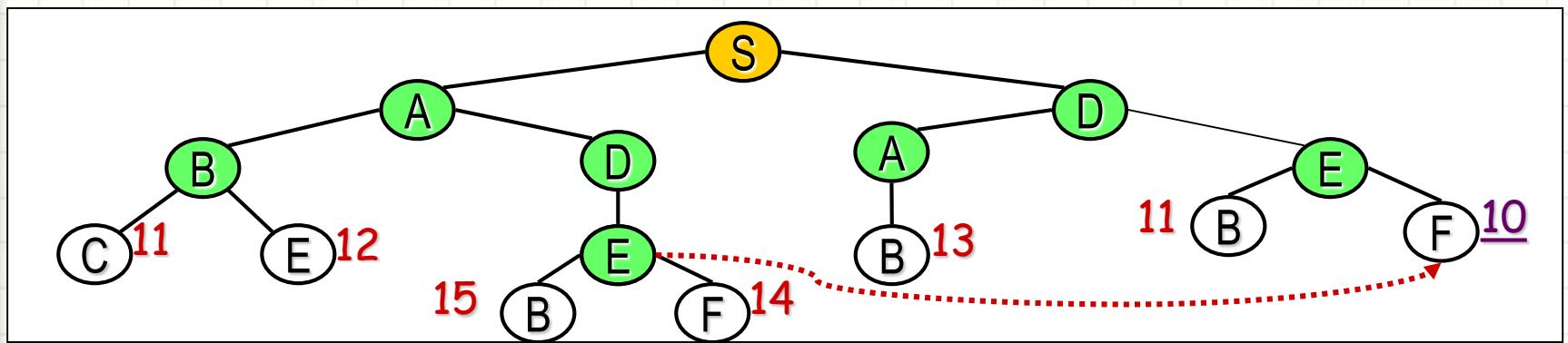
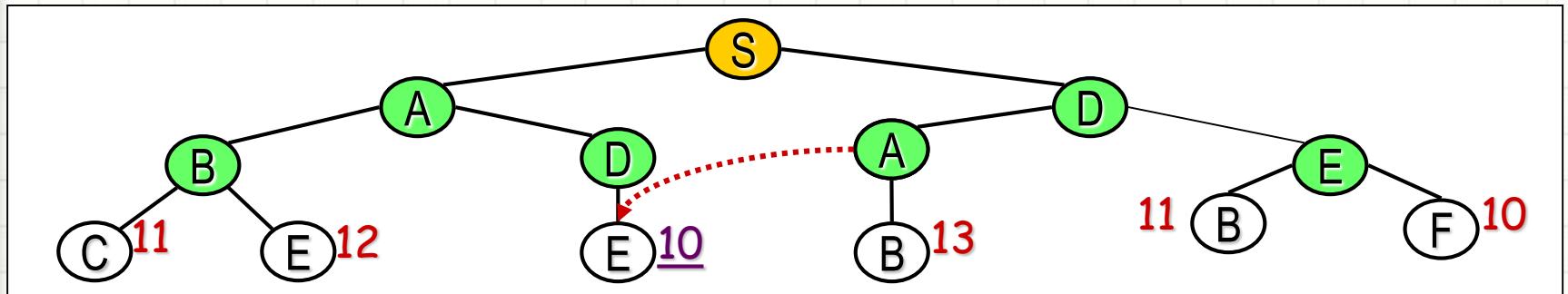
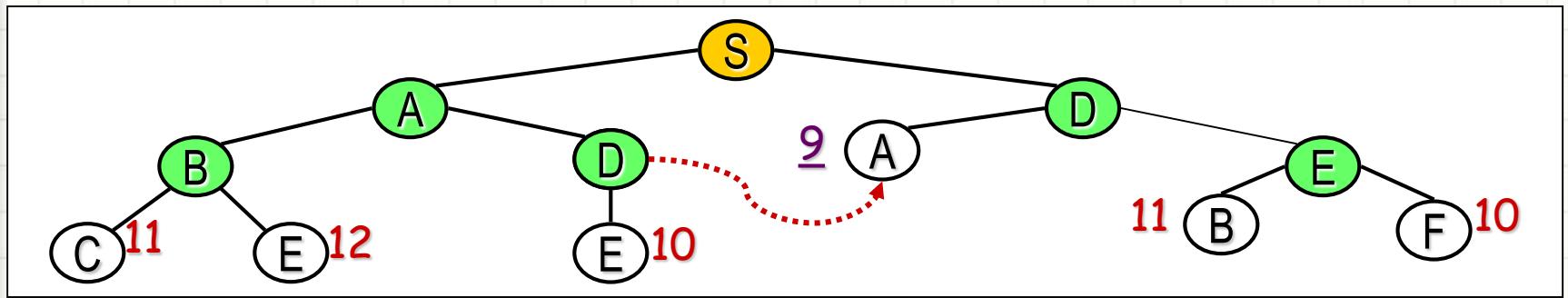
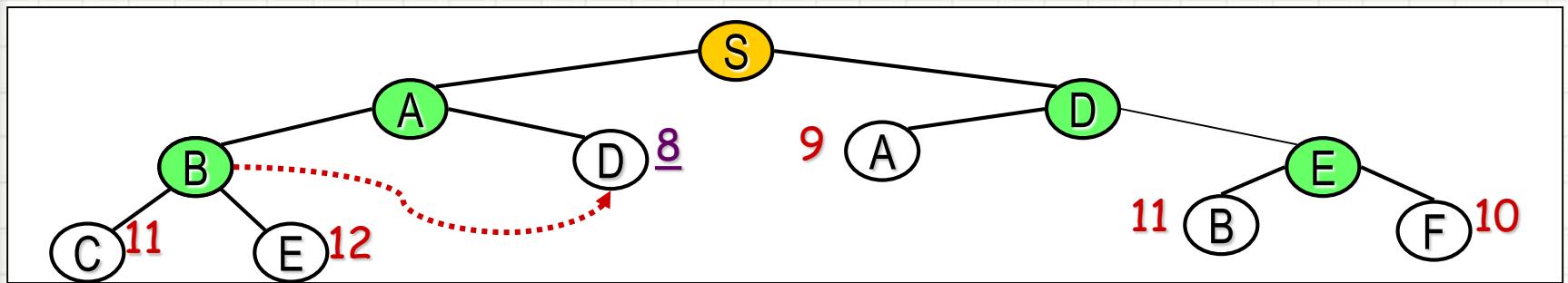
Eşit Maliyetli (Uniform Cost) için

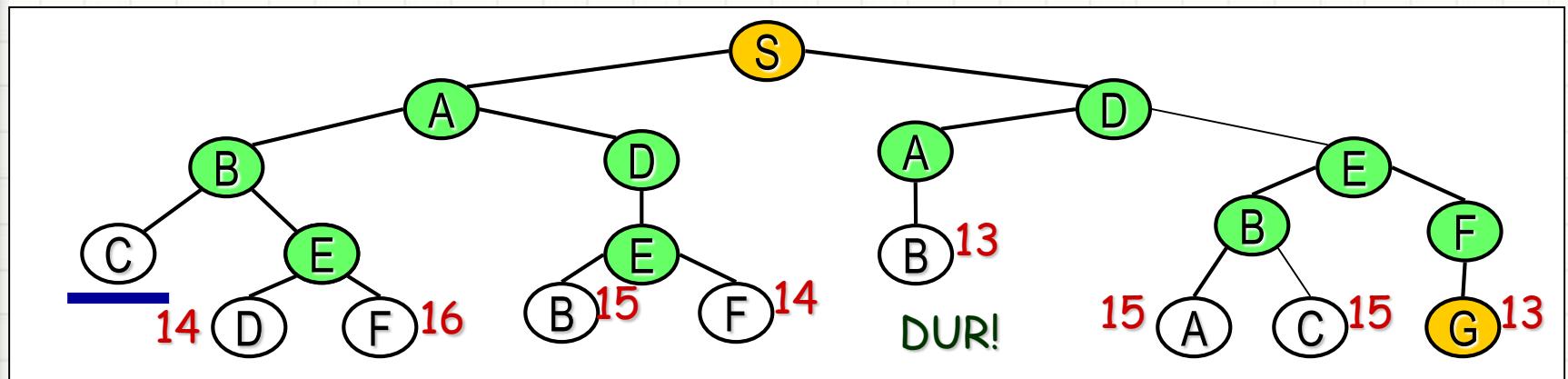
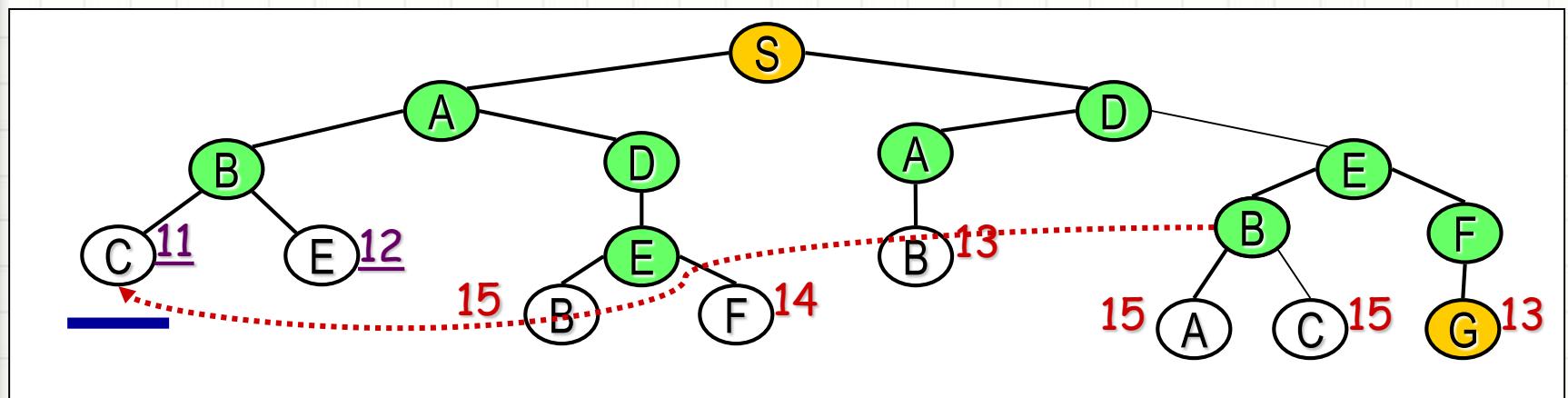
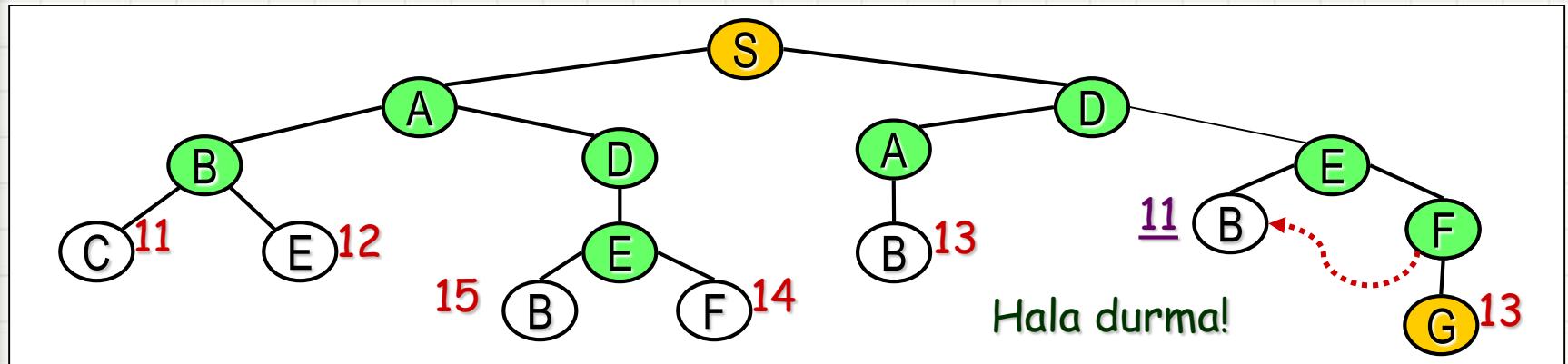
- Hatırlatma:
 - Şimdiye kadarki minimum maliyetli düğümü genişlet
- İki Örnek:**
- Yol silmesiz
 - Yol silmeli



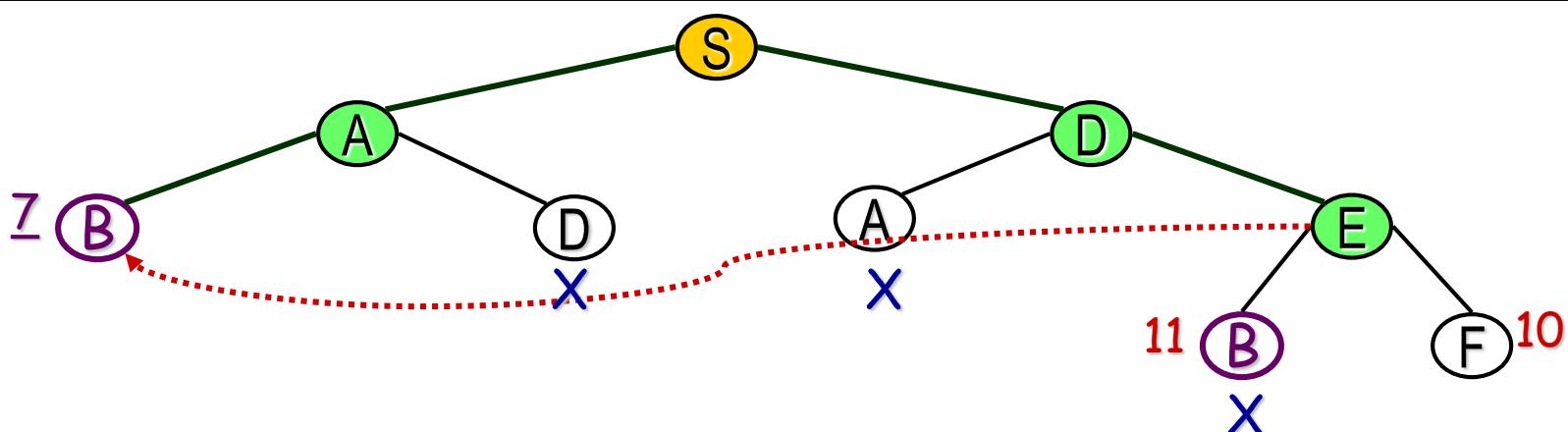
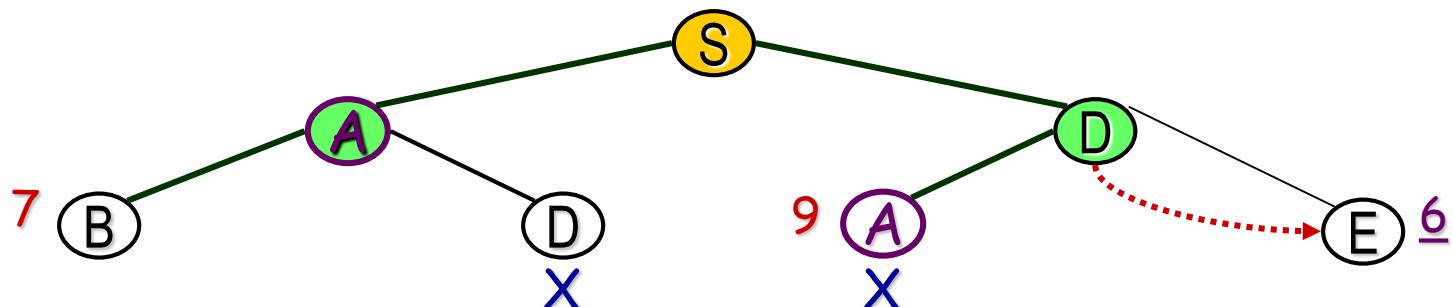
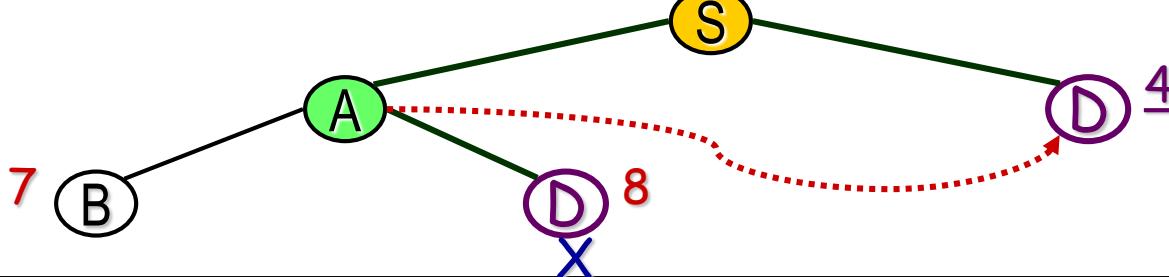
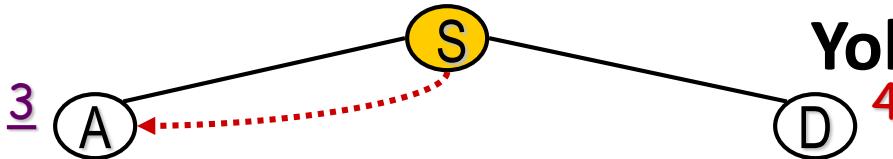
Eşit maliyetli arama (yol silmesiz)

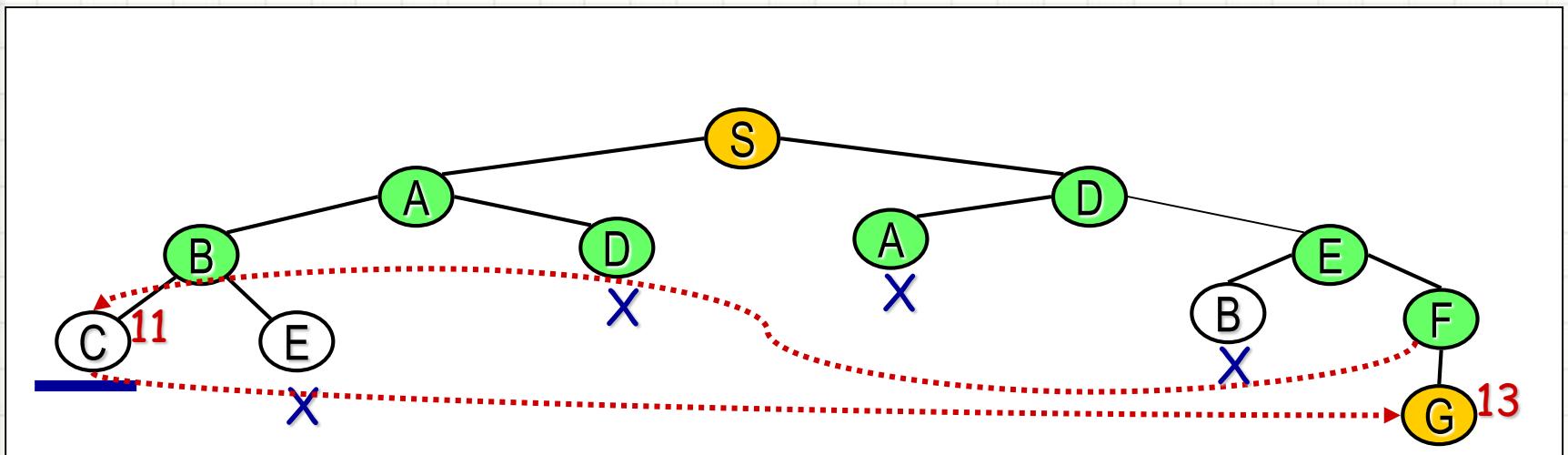
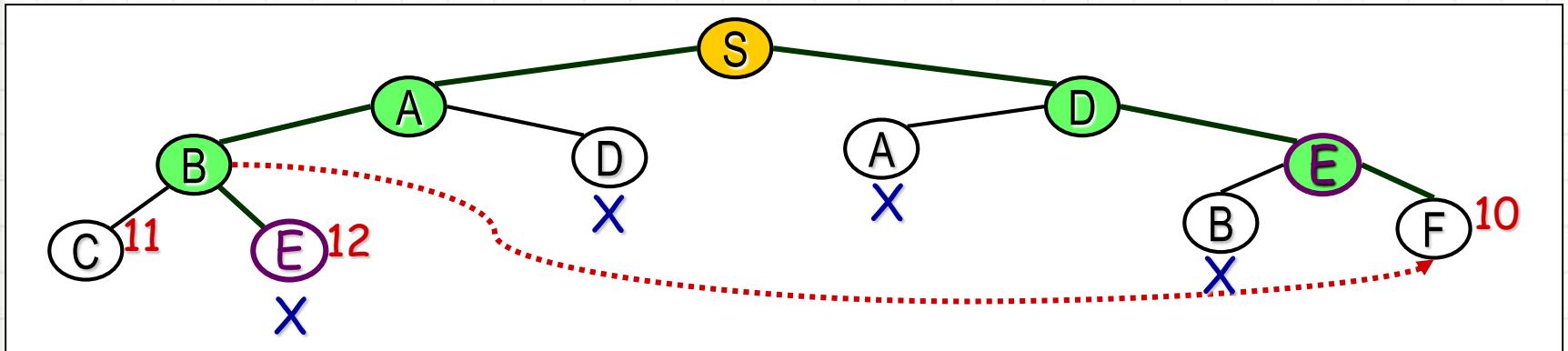






Yol silmeli





- Genişletilen düğüm sayısı **büyük ölçüde** azaldı:
 - 5 genişletme daha az
- => Bellek gereksinimi azalır

Yol silmeli A*

$$3 + 10.4 = 13.4 \quad \text{A} \quad \text{S} \quad \text{D} \quad 4 + 8.9 = 12.9$$

A 13.4

$$9 + 10.4 = 19.4 \quad \text{A} \quad \text{X} \quad \text{D} \quad \text{E} \quad 6 + 6.9 = 12.9$$

A 13.4

$$11 + 6.7 = 17.7 \quad \text{A} \quad \text{X} \quad \text{D}$$

$$10 + 3.0 = 13.0 \quad \text{E} \quad \text{F}$$

A 13.4

17.7

DUR!

$$13 + 0.0 = 13.0 \quad \text{G}$$

Sezgisel Fonk. Örnek: 8-puzzle

- $h_1(n)$ = yerinde bulunan taşların sayısı

1	3	2
8		4
5	6	7

$$h_1(n) = 4$$

Hedef

1	2	3
8		4
7	6	5

- $h_2(n)$ = yerinde bulunmayan taşların sayısı

1	3	2
8		4
5	6	7

$$h_2(n) = 4$$

Sezgisel Fonk. Örnek: 8-puzzle

- $h_3(n)$ = taşların hedefteki yerlerine uzaklıklarını toplamı (yatay ve dikey hane toplamları)
- Bu uzaklıklara *Manhattan uzaklığı* da denir

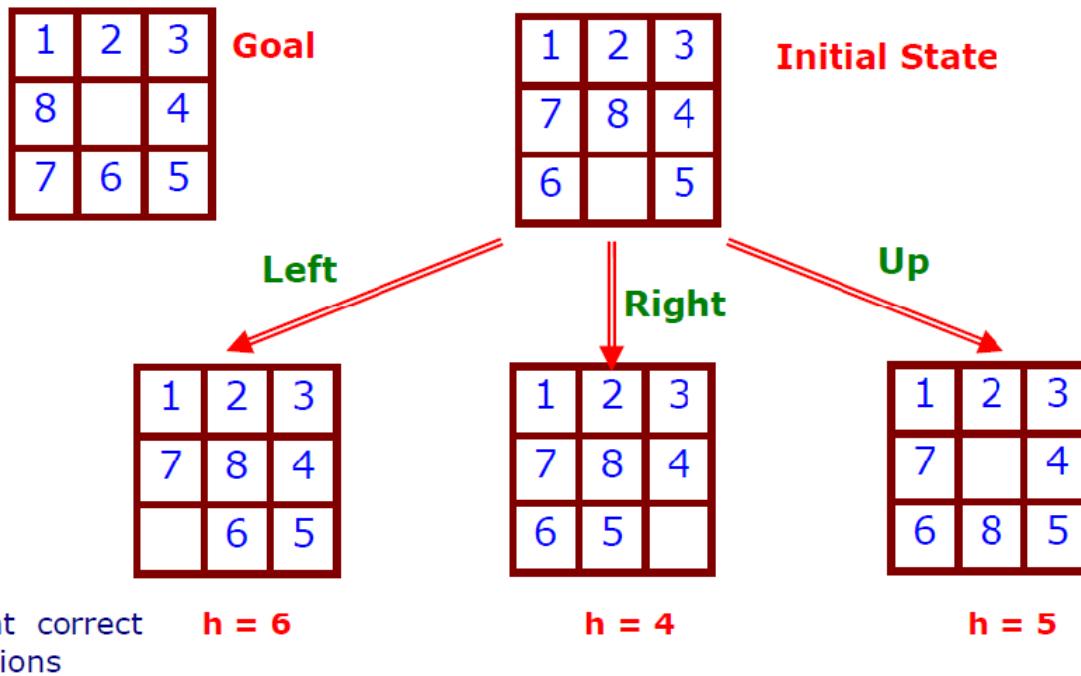
1	3	2
8		4
5	6	7

$$h_3(n) = 1 + 1 + 2 + 2 = 6$$

1	2	3
8		4
7	6	5

Kabul edilebilir sezgiseller

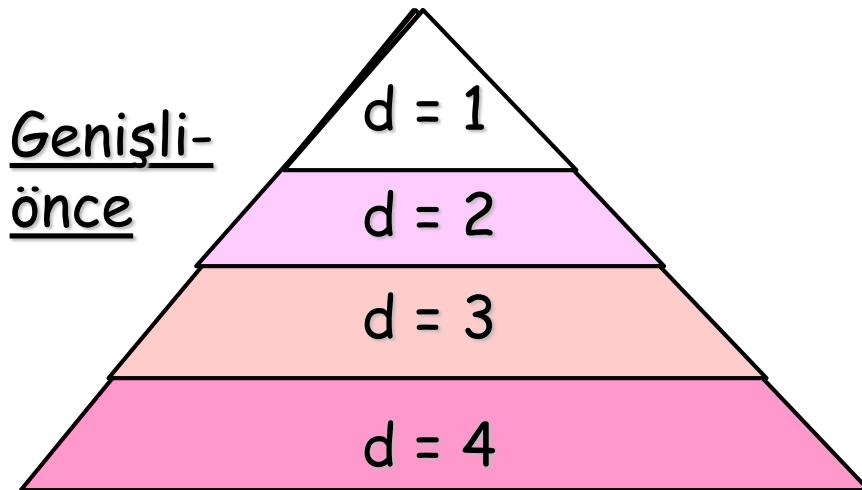
8 Puzzle Örneği



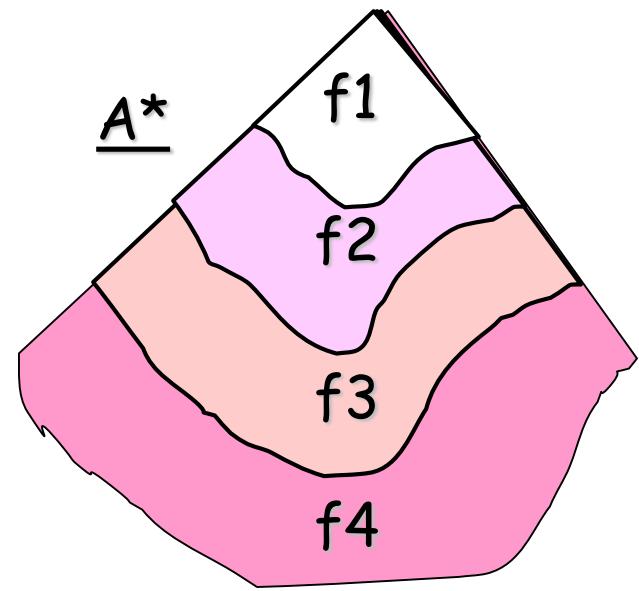
- Sayılar sağa, sola, aşağı veya yukarı hareket ettirilebilir.
- Her harekette doğru ve yanlış yerlerde bulunan sayıların kaç tane olduğu ve her bir sayı amaç durumda olması gereken yerden ne kadar uzaklıktadır?

İlerleyen Derinleştirmeli A* (IDA*)

- A* genişlik önceye benzer :



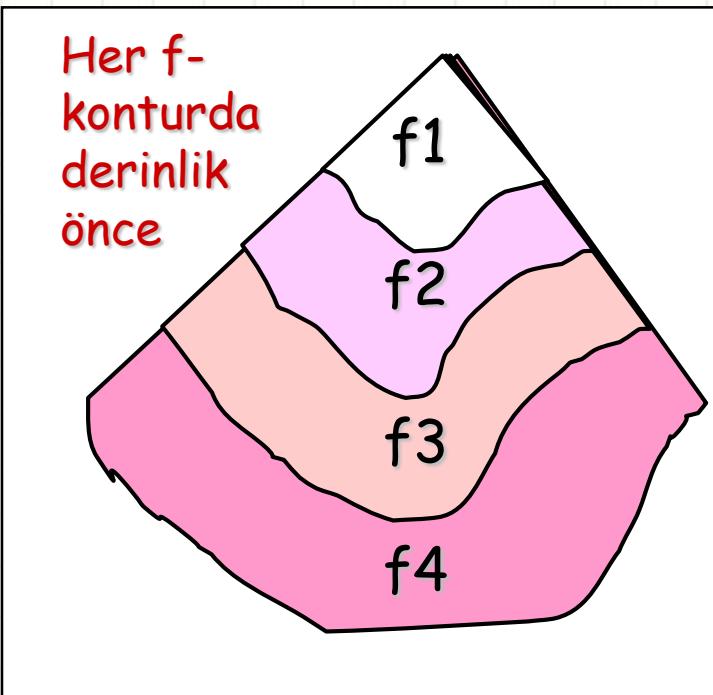
Derinlik-katmanları ile açar



F-konturlarıla açar

- IDA* : Yinelemeli Derinleştirmeye benzer bir fikri kullanır

IDA*



- F-sınırlına limitlenmiş derinlik önce aramasını yürüt.
- Hedef bulunmuşsa: TAMAM.
Değilse: f-sınırlını arttır ve tekrar başla.

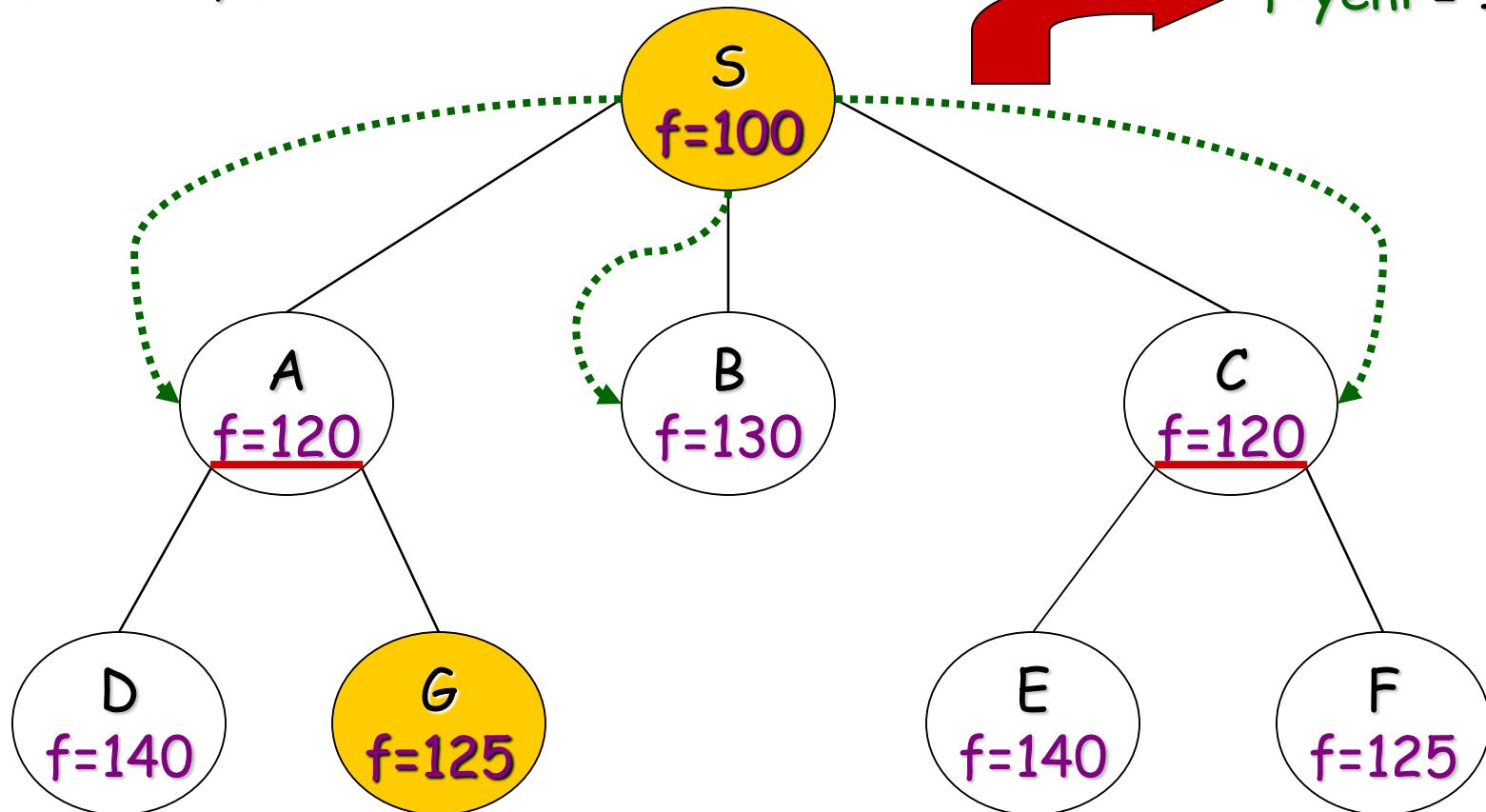
F-sınırları nasıl belirlenmeli?

- başlangıçta: $f(S)$ sezgisele göre belirle
 1. tüm ardılları (succ) üret
 2. minimal $f(\text{succ}) > f(S)$ ise kaydet
 3. $f(S)$ ' in yerine minimal $f(\text{succ})$ ile devam et (**1'e dön**)

Örnek

f-limitli, f-sınır = 100

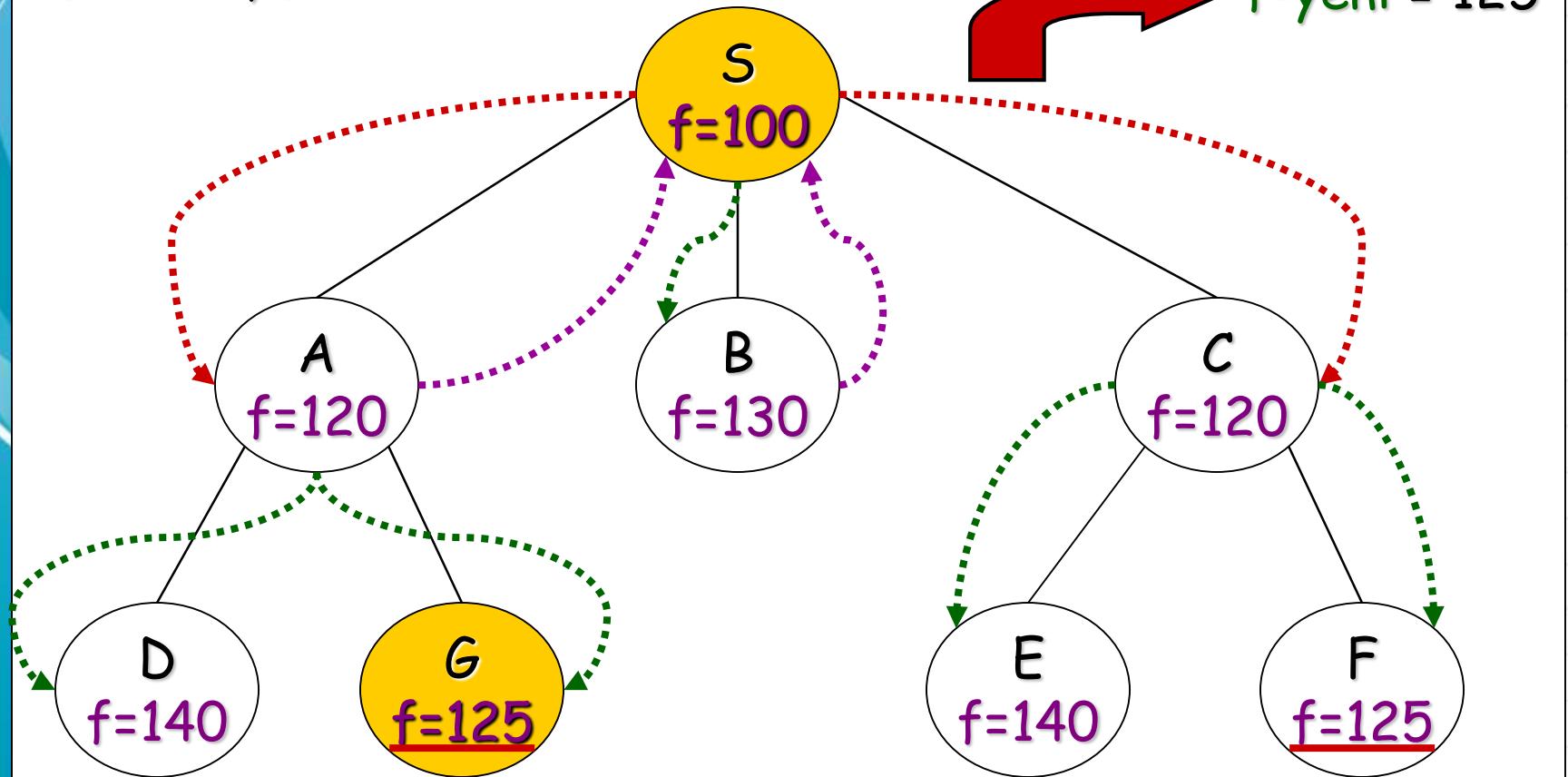
f-yeni = 120



Örnek

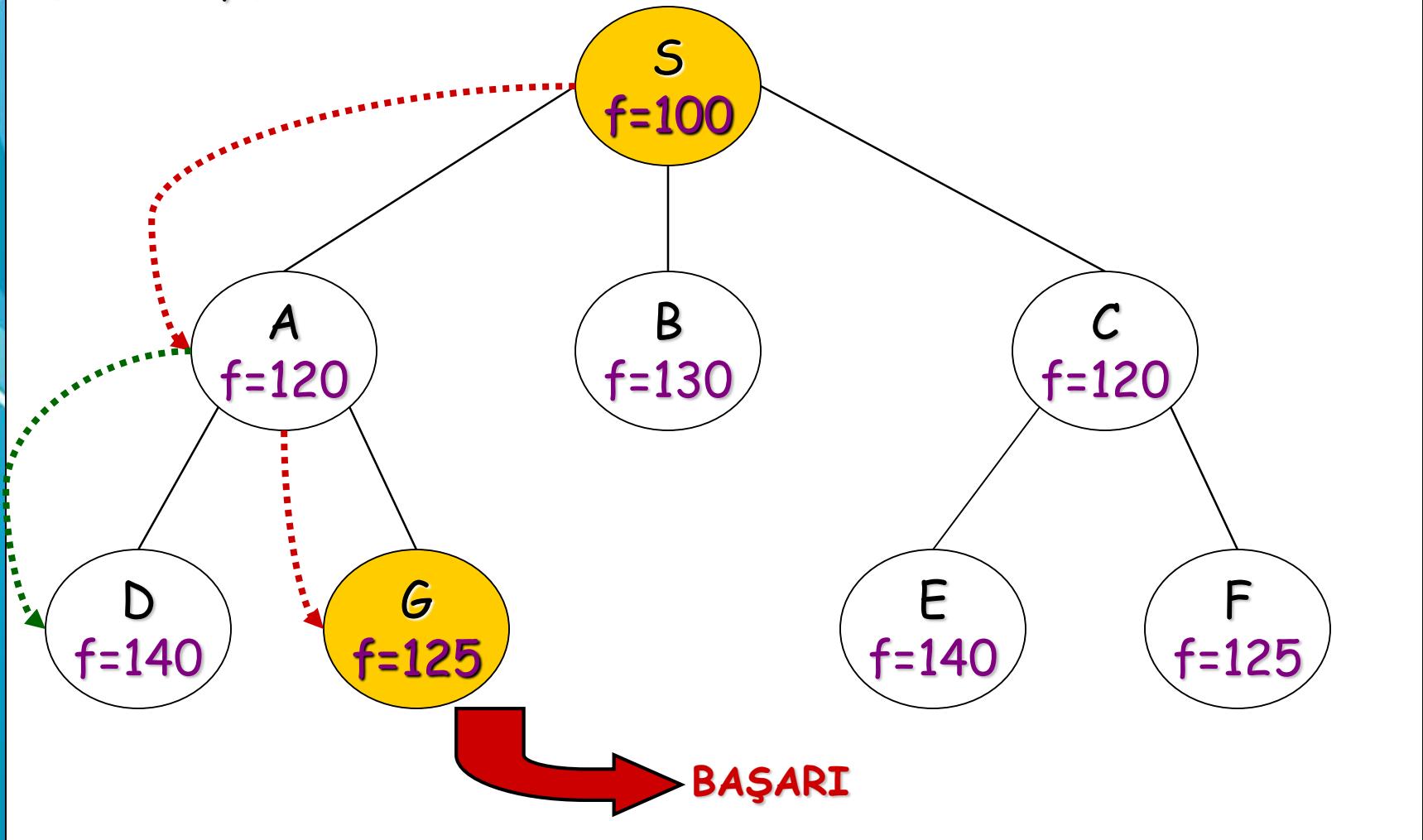
f-limitli, f-sınır = 120

f-yeni = 125



Örnek

f-limitli, f-sınır = 125



IDA* Özellikleri

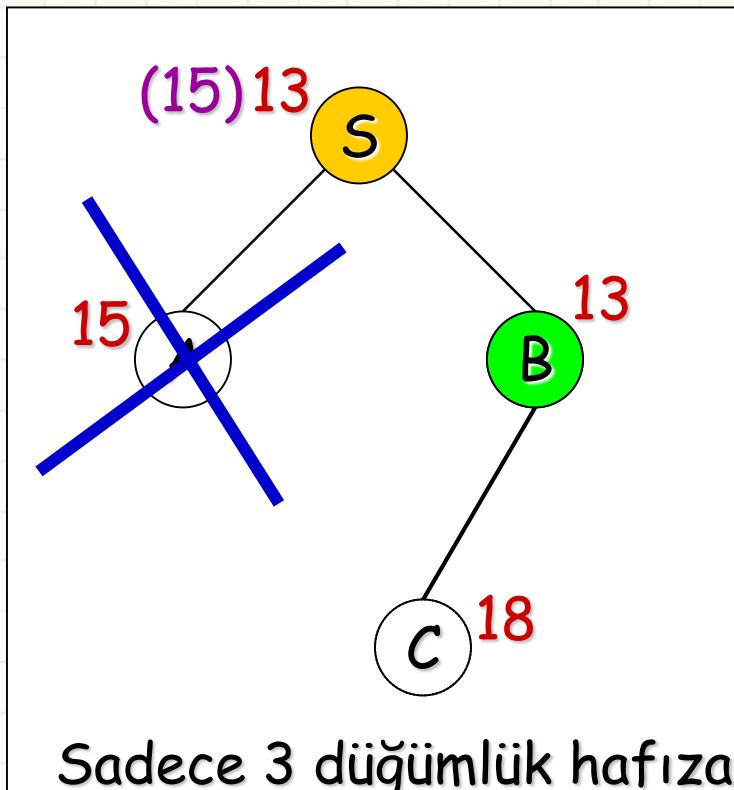
- Yinelemeli derinleşen ile aynı mantığa dayanır
 - Tam ve Optimal
- Completeness: Evet
- Optimality: Evet
- Time complexity: $O(b^d)$
- Space complexity: $O(b d)$

b: dallanma faktörü

d: en düşük maliyetli çözümün derinliği

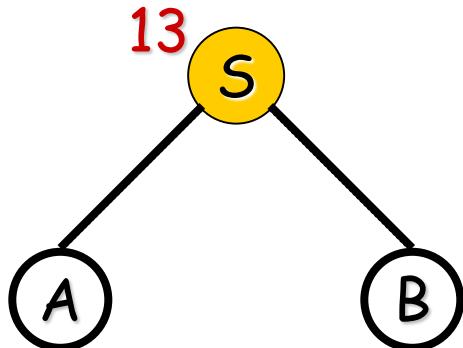
Basitleştirilmiş Hafıza-sınırlı A*

- SMA*: Simplified Memory Bounded A*



- Eğer hafıza dolu ise ve yeni bir ekstra düğüm (C) üretmek zorundaysak :
En yüksek f-değerlikli yaprağı Kuyruktan kaldır(A).
Her ata düğümünde en iyi "**unutulan**" çocuğun f-değerini hatırla (S'te15).

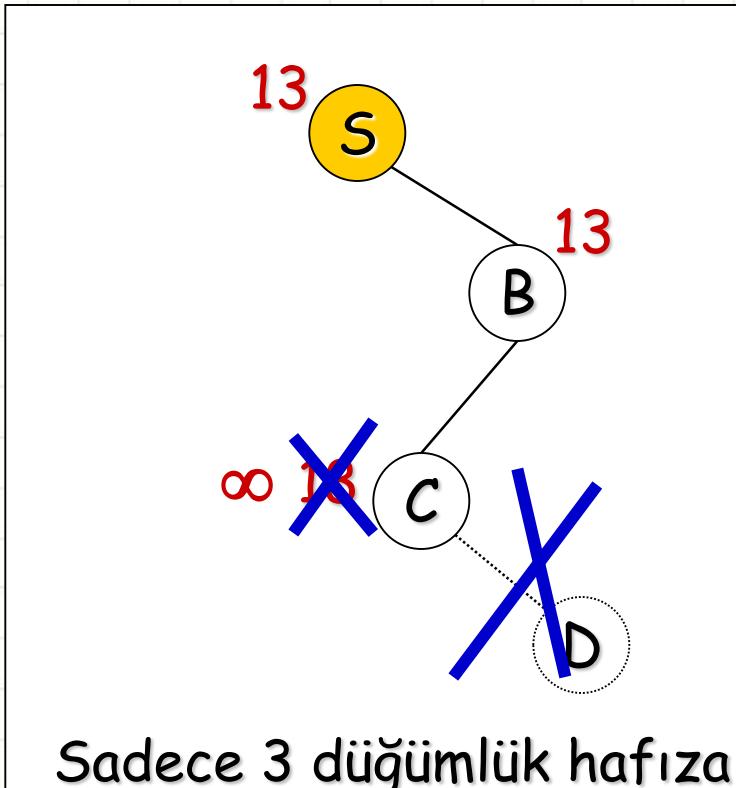
Tek tek çocukları üretme



Önce A sonra B'yi ekle

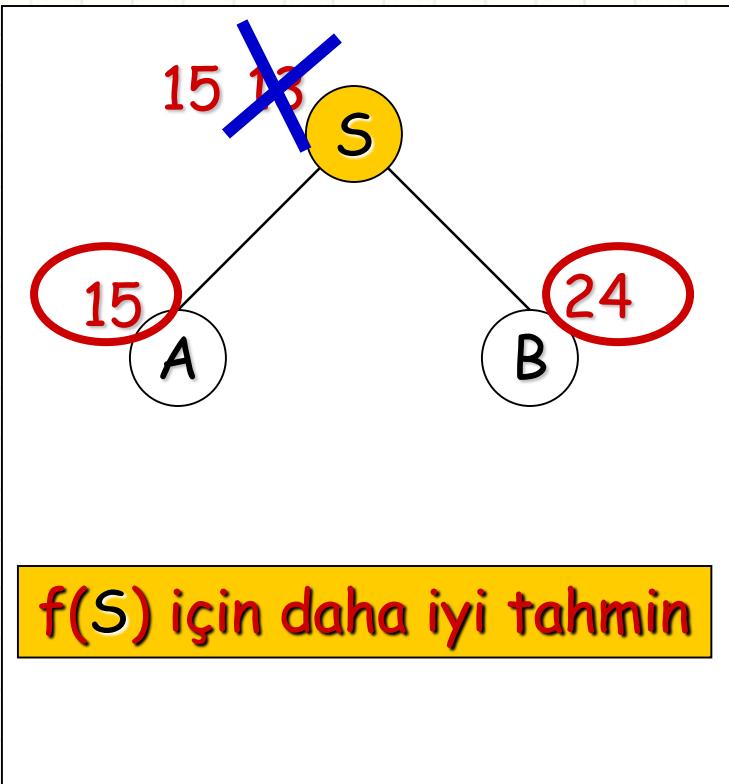
- Bir düğümü açarken (S) herhangi bir zamanda sadece bir çocuğunu kuyruğa ekle.
 - Soldan-sağayı kullanırız
- Hafıza taşmasından kaçın ve başka bir düğümü silip silmeyeceğimizi gözle

Çok uzun yol: vazgeç



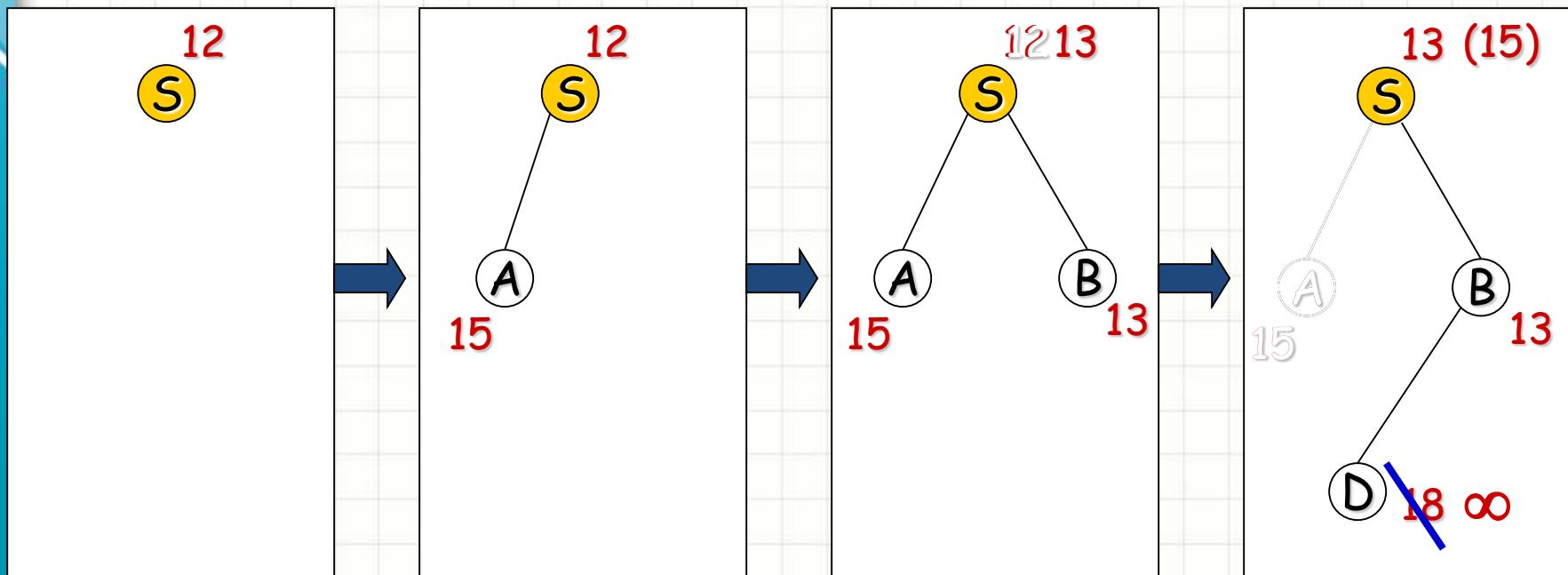
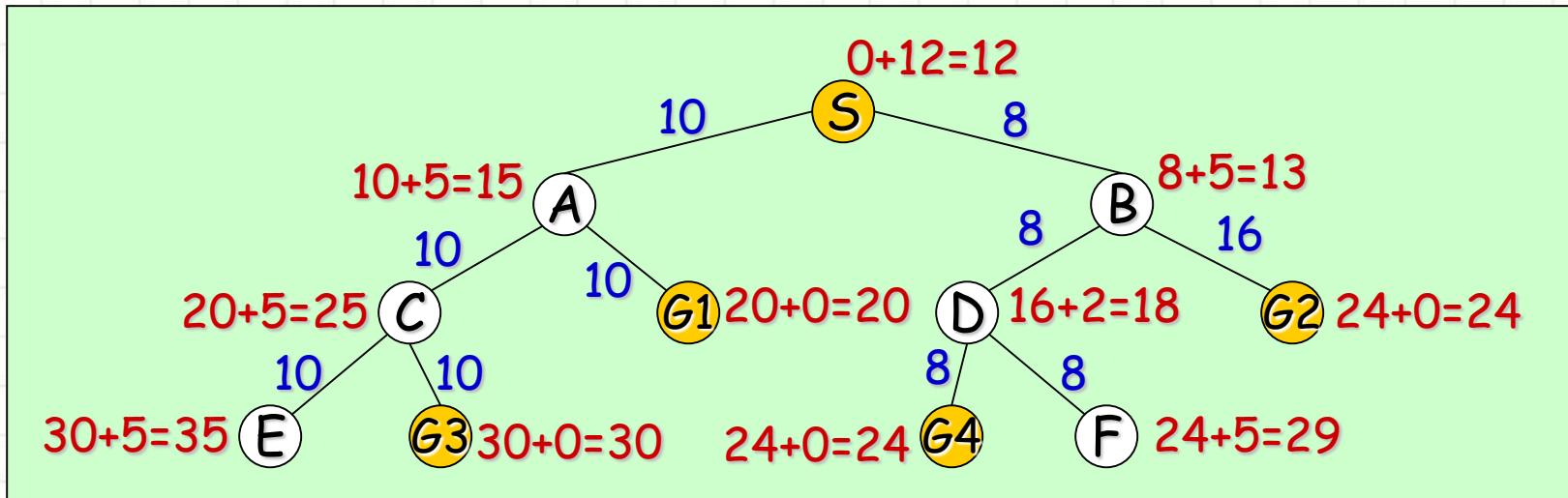
- Genişletilen bir düğüm hafızadan daha uzun bir yol üretiyorsa :
 - Bu yoldan vazgeç (C).
Cünkü daha az maliyetli bir opsiyon mevcut
 - Düğümün (C) f-değerini ∞ yap
 - (burdan bir yol bulamayacağımızı hatırlamak için)

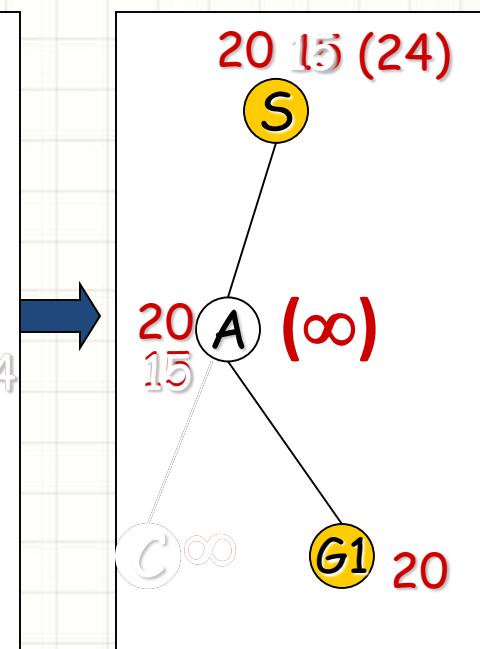
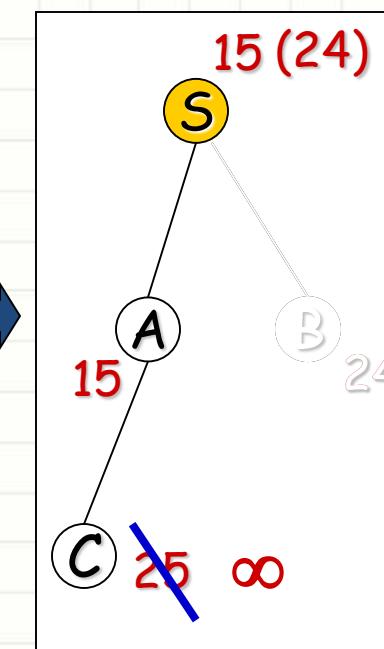
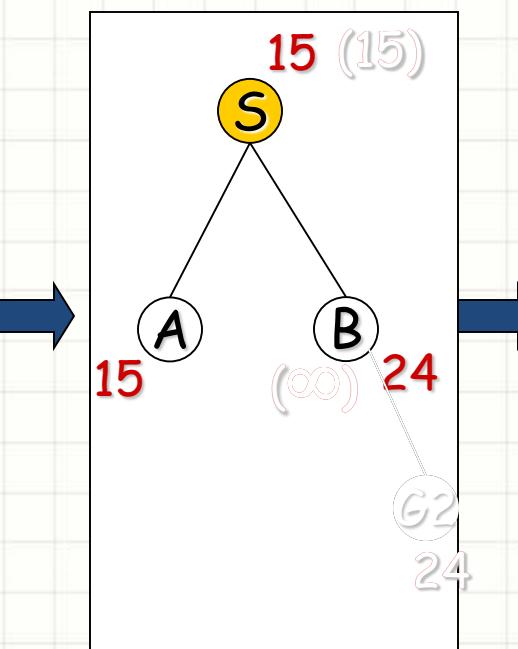
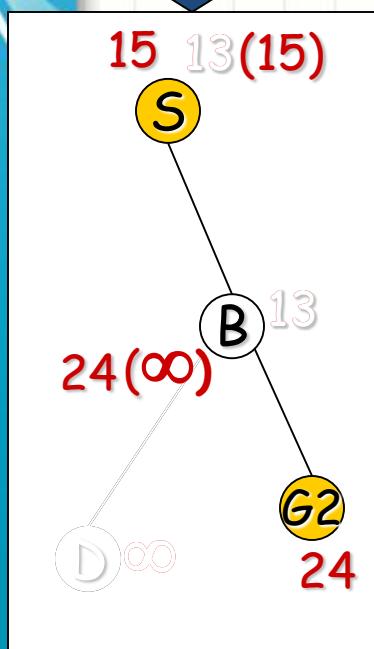
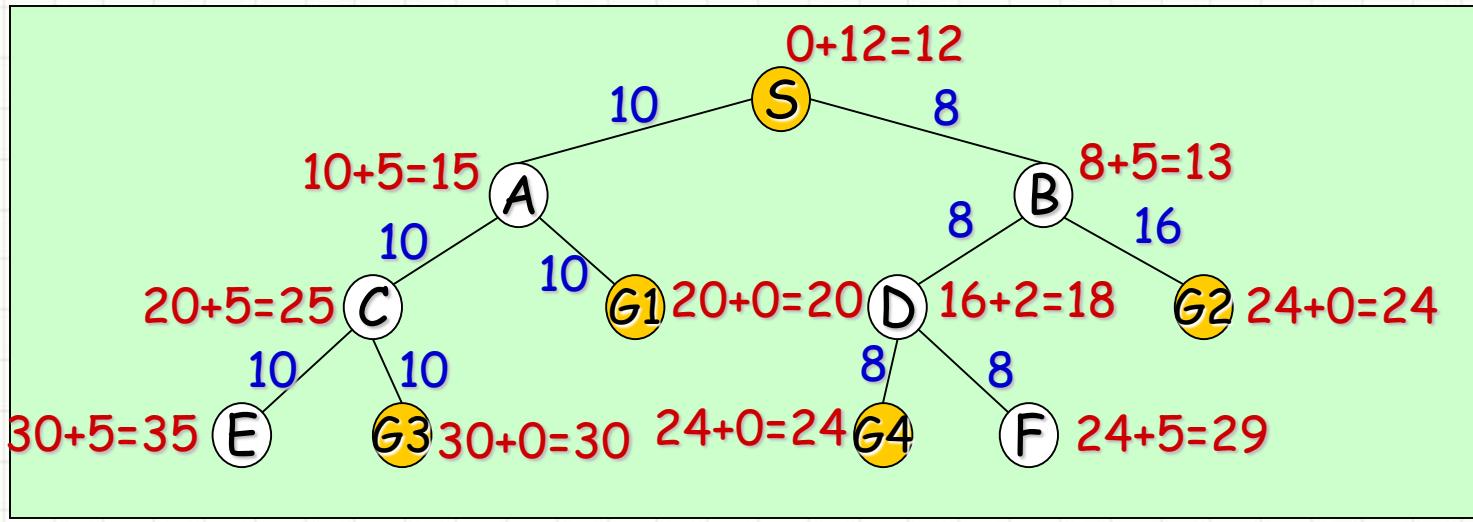
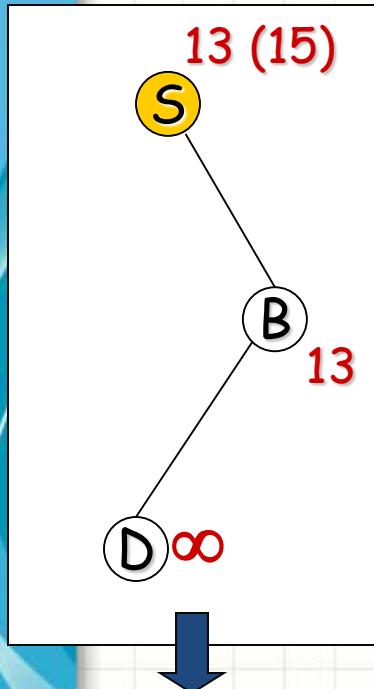
F-değerlerini ayarla



- Bir n düğümünün tüm m çocukları gezilmişse ve tüm m için :
 - $f(m) > f(n)$
- o zaman güncelle:
 - $f(n) = \min \{ f(m) \}$
- Yol boyunca n'in çocuklarından birisinin üzerinden gitmesi gereği için!

SMA*: bir örnek





SMA* Özellikleri

- Completeness: Evet
 - En kısa yolu depolamaya izin verecek yeterli hafıza varsa
- Optimality: Evet
 - En iyi yolu depolamaya yetecek hafıza varsa
 - Değilse: hafızaya uyan en iyi yolu geri gönderir
- Time complexity: Tüm ağaç depolayacak yeterli hafıza varsa A* daki gibidir
- Space complexity: Varolan tüm hafızayı kullanır

b: dallanma faktörü

d: en düşük maliyetli çözümün derinliği

SONUÇ: Bilgisiz ve Bilgili Arama

- **Bilgisiz** arama algoritmaları, düğümlerin amaca uzaklığı hakkında bir bilgiye sahip değildirler. Eğer belirli sayıda düğüm varsa, uygun çözüm bulunabilir
- **Bilgili** arama yaklaşımlarında, amaca olan tahmini uzaklık kullanılır. Amaca yakın olan düğümler ilk önce açılır. Çözümün bulunması garanti değildir. Bu yöntemlerin başarılı olmasında doğru değerlendirme fonksiyonunun seçilmesi önemli rol oynar