Algoritma Analizi

Ders 4: Böl ve Yönet Yaklaşımı

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

Böl ve Yönet

- Önceki bölümlerde Birleştirmeli Sıralama algoritmasının böl-ve-yönet metodu ile sıralama problemini nasıl çözüldüğünü görmüştük.
- Buna göre böl-ve-yönet metoduyla problemler aşağıdaki üç aşama uygulanarak çözülürler:
  - Böl: Problem daha küçük alt problemlere bölünür.
  - Yönet: Alt problemler recursive olarak çözülür. Problem yeteri kadar küçüldüğünde problem direk olarak çözülebilir.
  - Birleştir: Alt problemlere bulunan çözümler birleştirilerek asıl problem çözülür.
- Alt problemler yeteri kadar büyük olduğunda bu problemler recursive şekilde daha küçük problemlere ayrılır.
- Problemler yeteri kadar küçüldüğünde başlangıç adımı (İng: base case) oluşur.
  - Bu durumda problem direk olarak çözülür.
  - Daha sonra ufak çözümler birleşere asıl problem çözülür.

Özyineleme

- Özyinelemeler (İng: recurrences) böl-ve-yönet metodunun en önemli parçasıdır.
- Bir özyineleme fonksiyonu, bir fonksiyonun kendisinin daha düşük girdiler ile tanımlanmasını sağlayan eşitlik veya eşitsizlik olarak tanımlanır.
  - Birleştirmeli sıralama algoritmasına bakarsak, bu algoritmanın en kötü çalışma zamanının şu şekilde tanımlandığını görürüz:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 , \end{cases}$$

- Yinelemeler başka şekillerde tanımlanabilirler.
  - $T(n) = T(2n/3) + T(n/3) + \Theta(n)$
  - $T(n) = T(n-1) + \Theta(1)$

Özyineleme

- Bu bölümde özyineleme yöntemi sayesinde çalışan algoritmaların çalışma sürelerini hesaplamak, yani Θ ve O asimtotik notasyonu ile çalışma süresinin sınırlarını bulmak için kullanılan üç farklı yöntem öğreneceğiz:
  - Substitution (değiştirme) metodu: Öncelikle bir sınır tahmin edip matematiksel tümevarım ile bu sınırın doğru olup olmadığını kontrol ediyoruz.
  - Yineleme ağacı metodu: Yinelemeleri bir ağaç şeklinde gösterip her yineleme seviyesinde ne kadar süre geçtiğini hesaplıyoruz. Sonrasında toplam formülleri ile sınırı buluyoruz.
  - Master metodu: Bu yöntem ile $T(n) = aT(n/b) + f(n), a \geq 1, b > 1$ şeklinde yazılabilen yinelemeler için sınır bulunabilir.

Özyineleme

- Bazı yinelemeler eşitsizlik şeklindedir. Örneğin:
  - $T(n) \leq 2T(n/2) + \Theta(n)$
    - Bu tür yinelemeler sadece üst sınır belirttiği için O notasyonu ile üst sınır bulacağız.
  - $T(n) \geq 2T(n/2) + \Theta(n)$
    - Bu tür yinelemeler sadece alt sınır belirttiği için Ω notasyonu ile alt sınır belirleyeceğiz.
- Yineleme tanımı yaparken bazı detayları görmezden gelebiliyoruz.
  - Örneğin Birleştirmeli Sıralama algoritmasının en kötü çalışma zamanı aslında şu şekilde tanımlanmalıdır:

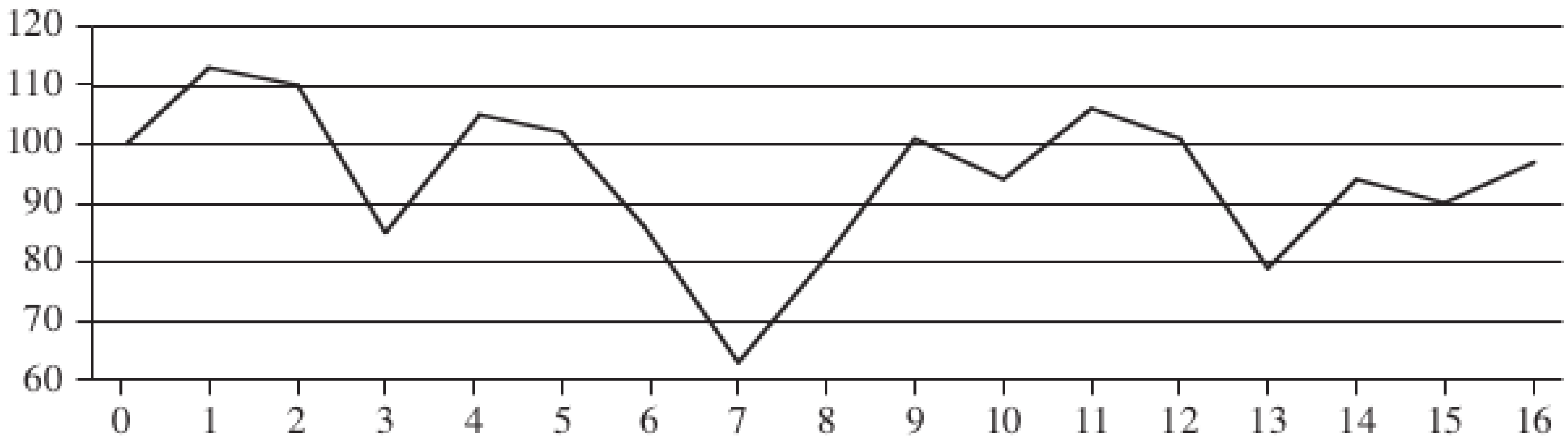
$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \, , \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{if } n > 1 \, . \end{cases}$$

Özyineleme

- Yineleme tanımı yaparken bazı detayları görmezden gelebiliyoruz
  - Sınırdaki değerleri sabit olarak kabul ediyoruz.
    - Örneğin, $T(n) = 2(T(n/2) + \Theta(n))$ yazıyoruz.
    - Bunun nedeni $T(1)$ değerini değiştirdiğimizde toplam süre değişmekle birlikte, toplam süre sabit bir faktör kadar değişmesi ve toplam sürenin büyüme hızının değişmemesidir.
- Bu detayları görmezden gelsek bile analizimizi yaparken taban, tavan durumlarının veya sınır değerlerinin sonucu etkileyip etkilemediğini kontrol edeceğiz.

Maksimum altdizi problemi

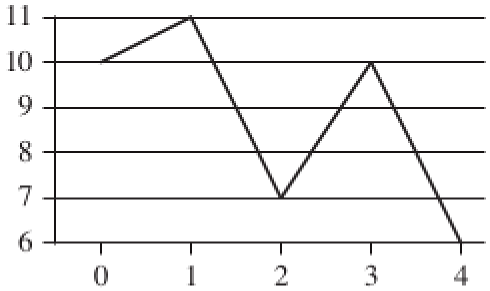
- Kimyasal ürün alıp satan bir firmadan çalıştığını düşünelim.
- Şirket belli zamanlarda belli ürünleri alıp başka bir zamanda satıyor.
- Alınan ve satılan ürünlerin değeri yıl içerisinde değişiyor.
- Senin görevin bu alım satım işinden en yüksek karı elde etmek.



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	−3	−25	20	−3	−16	−23	18	20	−7	12	−5	−22	15	−4	7

Maksimum altdizi problemi

- En düşük fiyat olan günde alıp en yüksek fiyat olan günde satmak en iyi çözüm olur.
  - Ancak bu her zaman mümkün mü?



Day	0	1	2	3	4
Price	10	11	7	10	6
Change		1	−4	3	−4

Maksimum altdizi problemi

- Kaba kuvvet ile çözülmeye çalışıldığında: $\binom{n}{2}$ farklı kombinasyon mevcut.
- Bu durumda $\Theta(n^2)$ zamanda her kombinasyonu denememiz gerekir.
- Sonuç olarak $\Omega(n^2)$ zamanda bu işi tamamlarız.
- Daha hızlı bir çözüm olabilir mi?

Maksimum altdizi problemi

- o(n^2) çalışma süresi olan bir algoritma bulmak istiyoruz.
- Amacımız başlangıcından sonuna fiyat farkının en fazla arttığı altdiziyi bulmak.
- Bu sebeple günlük fiyatlar yerine günlük fiyat değişimine bakalım.
  - Bu değerleri bir dizi olarak düşünürsek, birbirini takip eden değerlerden oluşan ve toplamları en fazla olan alt diziyi bulmalıyız.
  - Bu altdiziyi maksimum altdizi olarak adlandırıyoruz.
    - Örneğin verilen problemdeki A[1..16] dizisinin maksimum altdizisi A[8..11].

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	13	–3	–25	20	–3	–16	–23	18	20	–7	12	–5	–22	15	–4	7

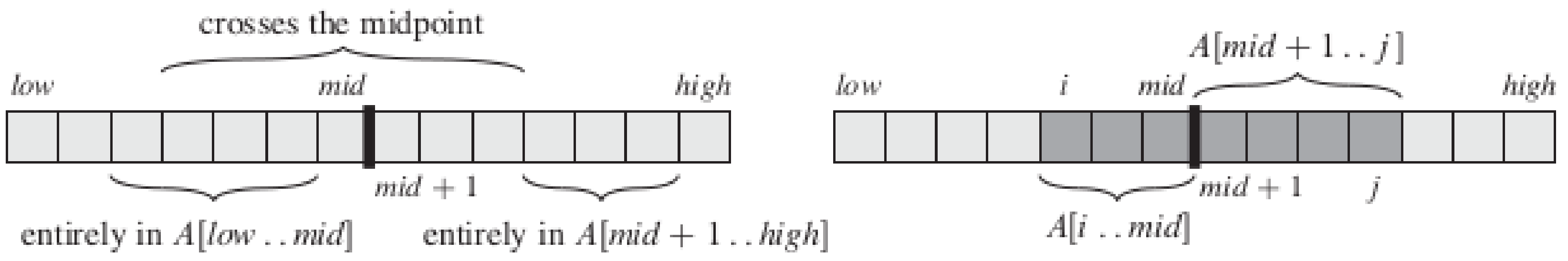
maximum subarray

Maksimum altdizi problemi

- Kaba kuvvet ile çözümler pek hızlı çalışmıyor.
- Alternatif olarak böl-ve-yönet yaklaşımını kullanalım.
- Diyelim ki fiyattaki değişim oranlarını veren dizimiz A[low..high].
  - Bu dizideki maksimum altdiziyi bulmak istiyoruz.
- Böl-ve-yönet yaklaşımına göre problemi iki farklı parçaya ayırıyoruz.
  - Öncelikle orta noktayı buluyoruz. Orta noktayı mid olarak adlandıralım.
  - Daha sonra diziyi iki altdiziye ayrıyoruz: A[low..mid], A[mid+1,high].
  - Bir sonraki slaytta görüleceği üzere bu dizinin her altdizi A[i..j], aşağıdaki durumlardan birinde olmalıdır.
    - Tamamı A[low..mid] içerisinde.
    - Tamamı A[mid+1..high] içerisinde
    - Bir kısmı A[low..mid] bir kısmı A[mid+1..high] içerisinde

Maksimum altdizi problemi

- A[low..high] dizisinin maksimum altdizisi A[low..mid], A[mid+1,high] dizilerinin veya bu iki altdizinin arasında kalan dizilerden en fazla toplama sahip olan dizi olmalıdır.
- A[low..mid] ve A[mid+1..high] dizilerinin en fazla toplama sahip dizilerini yinemeli yöntem kullanarak bulmak kolay, çünkü bu diziler baştaki problemin küçük parçaları.
- O zaman yapmamız gereken arada kalan dizilerin en büyük toplama sahip olanını bulmak ve bu üç değeri karşılaştırmak.



Maksimum altdizi problemi

- İki parçanın arasında kalan maksimum altdiziyi şu şekilde bulabiliriz:
  - Şekilde görüldüğü bu diziler iki altdiziden oluşuyor. Bu diziler A[i..mid] ve A[mid+1..j], $low \le i \le mid$ ve $mid < j \le high$.
  - Bu sebeple maksimum toplama sahip A[i..mid] ve A[mid+1..j] dizilerini bulup birleştirmeliyiz.
  - Bu işlem bir sonraki slaytta görülen şekilde yapılabilir.

Maksimum altdizi problemi

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = -∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = -∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

Maksimum altdizi problemi

- Önceki slayttaki işlem $\Theta(n)$ süre alacaktır.
  - 3-7 satırlarındaki for döngüsü mid-low+1 kere çalışacaktır.
  - 10-14 satırlarındaki for döngüsü high-mid kere çalışacaktır.
  - Bu durumda toplam döngü sayısı
    - (mid – low + 1) + (high – mid) = high – low + 1 = n

Maksimum altdizi problemi

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low])            // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

Maksimum altdizi problemi

- Görüleceği üzere FIND_MAX_CROSSING_SUBARRAY fonksiyonuna benzer şekilde FIND_MAXIMUM_SUBARRAY yinelemeli fonksiyonu verilen dizideki maksimum altdizinin başlangıç ve bitiş noktaları ile bu altdizinin toplam değerini vermektedir.
- Algoritmanın analizi:
  - Birleştirmeli Sıralama algoritmasında olduğu gibi girdinin 2^n şeklinde gösterilebildiğini farzedelim.
  - Sınır durumunda yani $n = 1$ olduğunda (satır 2) yapılan işlem sabit zaman alacaktır. Yani $T(1) = \Theta(1)$
  - Problemin iki parçaya ayrılması sabit zamanda yapılıyor.
  - Her yinelenen çağrıda problem 2 parçaya ayrılıyor ve bu problemlerin büyüklüğü önceki problemin yarısı, yani $2T(n/2)$
  - FIND_MAX_CROSSING_SUBARRAY fonksiyonu $\Theta(n)$ zaman alıyor.

Maksimum altdizi problemi

- Algoritmanın analizi

$$
\begin{aligned}
T(n) &= \Theta(1) + 2T(n/2) + \Theta(n) + \Theta(1) \\
&= 2T(n/2) + \Theta(n) .
\end{aligned}
$$

$$
T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 . \end{cases}
$$

- Görüldüğü üzere hesaplanan çalışma süresi Merge sort ile aynı.
  - Θ(n lg n)

Değiştirme metodu

- Subsititution (Değiştirme) metodu
  - Bu metodun iki aşaması vardır:
    - Doğru çözümü tahmin et.
    - Matematiksel tümevarım yöntemiyle gerekli sabitleri bularak çözümün çalıştığını göster.
  - Örnek
    - $T(n)=2T(\lfloor n/2 \rfloor)+n$
    - Tahminimiz bu fonksiyon için T(n) = O(n lg n).
    - Değiştirme metodu ile bu çözümün doğru olduğunu gösterebilmemiz için uygun bir c > 0 sabiti ile T(n) ≤ cn lg n hesaplandığını göstermemiz lazım.
    - Ispatın geri kalanı tahtada gösterilecektir.

Değiştirme metodu

- Değiştirme metodunun doğru çalışabilmesi için doğru tahmin yapmamız gerekmektedir.
  - Bu amaçla bazı akıllı tahmin metotları kullanılabilir.
  - Ayrıca biraz sonra göreceğimiz üzere yenileme ağacı metodu ile tahmin oluşturulabilir.
  - Daha önce görmüş olduğunuza benzer bir formül görürseniz, tahmin etmek kolaylaşır.
  - $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$
  - 17 sayısı nedeniyle önceki denklemden farklı gözükmekte.
  - Ancak n sayısı arttığında 17 sayısının etkisi azalacaktır.
  - Bu sebeple bu çalışma süresi için O(n lg n) mantıklı bir tahmin olacaktır.

Değiştirme metodu

- Bir başka örnek:
  - $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$
  - Görüntü itibariyle O(n) olabilir gibi gözüküyor.
  - Ancak direk olarak yerine koyduğumuzda ispat olmuyor.
    - Tahtada gösterilecek.
  - Peki nasıl ispatlayabiliriz.
    - Tahtada gösterilecek.
    - Bazı durumlarda düşük seviyeli bir terim ekleyerek veya çıkartarak ispatımızı yapabiliriz.