Soru(20 puan) Aşağıdaki rekürens denklemleri çözünüz.

a) $T(n)=16T(n/2)+n^3\lg n$ **her** $n>1$ **için**
$T(1)=\theta(1)$

Çözüm. a=16,b=2, $\log_2 16=4$ buradan $n^3\lg n=O(n^{\log_2 16-\varepsilon})$ yani master teoremin 1. durumu vardır. Yani $T(n)=\theta(n^4)$

b) $T(n)=4T(n/2)+n^2\lg n$ **her** $n>1$ **için**
$T(1)=\theta(1)$

Çözüm. a=4, b=2, $\log_2 4=2$ buradan da master teoremin 2. durumu vardır yani $T(n)=\theta(n^2\lg^2 n)$

c) $T(n)=9T(n/3)+n^3\lg n$ **her** $n>1$ **için**
$T(1)=\theta(1)$

Çözüm. a=9, b=3, $\log_3 9=2$ buradan da $n^3\lg n=\Omega(n^{\log_3 9+\varepsilon})$ master teoremin 3. durumu vardır. Düzgünlük koşulundan $\left(\frac{n}{3}\right)^3\lg\frac{n}{3}\le\delta n^3\lg n$. Buradan da $\lg n-\lg 3\le 3\delta\lg n$ Yani $(1-3\delta)\lg n\le\lg 3$. $\frac{1}{3}\le\delta<1$ için son eşitsizlik sağlanır.

Buradan da $T(n)=\theta(n^3\lg n)$

d) $T(n)=T(n/2)+T(n/3)+n$ **her** $n>1$ **için**
$T(1)=\theta(1)$

Çözüm.1. $T(n)=\theta(n)$ olduğunu kanıtlayalım. $T(n)>n$ olduğundan $T(n)=\Omega(n)$ dir. Şimdi de $T(n)=O(n)$ olduğunu tümevarımla gösterelim. k<n için doğru olsun. Yani $T(k)<ck$. Buradan da $T(n)=T(n/2)+T(n/3)+n<cn/2+cn/3+n=\frac{5c}{6}n+n<cn$

Son eşitsizlik c>6 için doğrudur.

Çözüm2. Soru ağaç çizilerek ve aynı seviyedeki değerleri topladığımızda da geometrik seri toplamı kullanılarak yapılabilir.

e) $T(n)=5\sqrt[5]{n^4}T(\sqrt[5]{n})+n\log_5 n$
$T(1)=\theta(1)$

Çözüm. $\frac{T(n)}{n}=5\frac{T(\sqrt[5]{n})}{\sqrt[5]{n}}+\log_5 n$. $S(n)=\frac{T(n)}{n}$ olsun. $S(n)=5S(\sqrt[5]{n})+\log_5 n$

$n=5^m$ **olsun.** $S(5^m)=5S(5^{\frac{m}{5}})+m$. $L(m)=S(5^m)$ olsun. $L(m)=5L(m/5)+m$.

Master teoremden $L(m)=\theta(m\lg m)$. $m=\log_5 n$ Yani $S(5^m)=\theta(m\lg m)$ buradan da $S(n)=\theta(\log_5 n\lg(\log_5 n))$ yani $T(n)=\theta(n\log_5 n\lg(\log_5 n))$.

Soru. Aşağıdaki önermelerin doğruluğunu kanıtlayınız veya yanlış olduğunu gösteriniz. (Verilen fonksiyonların hepsi asimptotik pozitif fonksiyonlardır)

a) $f(3n) = O(f(5n))$

b) $n^2 + 24n + 3 = \theta(n^2)$

c) $\sum_{k=1}^{n} \theta(k) = \theta(n^2)$

d) $f(n) = o(g(n))$ **ve** $g(n) = \Omega(h(n))$ **ise** $f(n) = O(h(n))$

e) $f(n) = o(g(n))$ **ve** $g(n) = O(h(n))$ **ise** $f(n) = o(h(n))$

Çözüm. a) $f(n) = 2^{-n}$ olsun. $f(3n) = O(f(5n))$ olması için $2^{-3n} \le c.2^{-5n}$ koşulunu sağlayan bir c>0 sabiti bulunabilmelidir. Yani $2^{2n} \le c$ olmalıdır ki sol taraf artan bir fonksiyon olduğundan böyle bir sabit bulunamaz.

b) $c_1 n^2 \le n^2 + 24n + 3 \le c_2 n^2$ olacak biçimde pozitif c_1, c_2 sabitlerinin varlığını gösterelim. $c_1 = 1$ için sol taraf her n için sağlanır. $c_2 = 5$ alırsak sağ taraf $(2n-6)^2 \ge 39$ eşitsizliğine denktir bu da n>6 için sağlanır.

c) $c_{1,k} \le \theta(k) \le c_{2,k}$ olsun. $\sum_{k=1}^{n} c_{1,k} k \le \sum_{k=1}^{n} \theta(k) \le \sum_{k=1}^{n} c_{2,k} k$ olur. $\min c_{1,k} = c_1$ ve $\max c_{2,k} = c_2$ olsun. Buradan da $\sum_{k=1}^{n} c_1 k \le \sum_{k=1}^{n} \theta(k) \le \sum_{k=1}^{n} c_2 k$ yani $c_1 \frac{n(n+1)}{2} \le \sum_{k=1}^{n} \theta(k) \le c_2 \frac{n(n+1)}{2}$

d) $f(n) = n^4, g(n) = n^5, h(n) = n$ sorunun koşullarını sağlar ama $f(n) = O(h(n))$ olmaz.

e) $\forall c_1 > 0$ için $\exists n_1 > 0$ vardır ki $\forall n > n_1$ için $f(n) < c_1 g(n)$ olduğu veriliyor. Ayrıca $\exists c_2 > 0$ ve $\exists n_2 > 0$ vardır ki $\forall n > n_2$ için $g(n) \le c_2 h(n)$ olduğu da veriliyor.

$\forall c_3 > 0$ alalım. $c_1 = \frac{c_3}{c_2}$ alalım ve $n_3 = \max\{n_1, n_2\}$ alalım. Buradan da $\forall n > n_3$ için

$f(n) < c_1 g(n) = \frac{c_3}{c_2} g(n) \le \frac{c_3}{c_2} c_2 h(n) = c_3 h(n)$ eşitsizliği sağlanır. Yani $f(n) = o(h(n))$ dir.

Soru. Aşağıdaki rekürensleri master teoreminin yardımıyla çözünüz. (Tüm rekürenslerde T(1)=O(1) dir.)

a)(10 puan) $T(n)=9T(n/3)+n^2\lg^3 n$
b)(10 puan) $T(n)=4T(n/3)+n\lg n$
c)(10 puan) $T(n)=4T(n/2)+n^3$

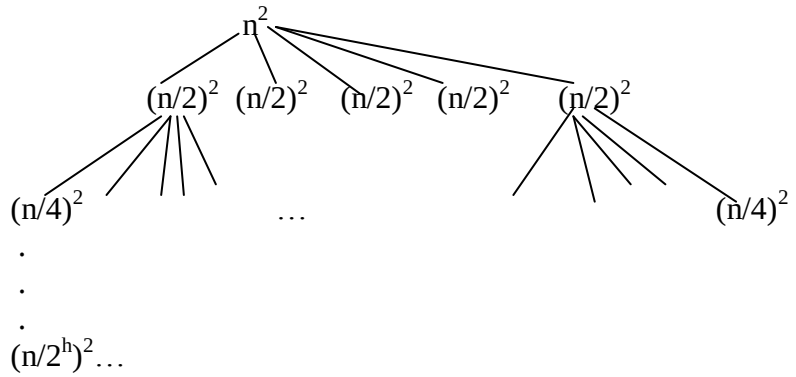
Çözüm.

a)a=9, b=3, $f(n)=n^2\lg^3 n$ olduğundan $n^{\log_3 9}=n^2$ Yani k=3 ile 2. Durum var. Buradan da $T(n)=\theta(n^2\lg^4 n)$
b)a=4, b=3, f(n)=nlgn ve $n^{\log_3 4}>n$ olduğundan 1. durumdan $T(n)=\theta(n^{\log_3 4})$
c)a=4, b=2, $f(n)=n^3$ ve $n^3 > n^{\log_2 4}$ olduğundan 3. Durum var. Düzgünlük koşulunun sağlanıp sağlanmadığını kontrol edelim. $4(n/2)^3<cn^3$, yani $n^3/2<cn^3$ eşitsizliği $1/2<c<1$ sabit sayısı için sağlanır. Sonuç olarak $T(n)=\theta(n^3)$ bulunur.

Soru Aşağıdaki rekürensi özyinelemeli ağaç yöntemiyle çözünüz. (T(1)=O(1) dir)

$T(n)=5T(n/2)+n^2$

Çözüm.



$(n/2^h)=1$ buradan da $h=\log_2 n=\lg n$

$T(n)=n^2+5(n/2)^2+5^2(n/2^2)^2+\ldots 5^h(n/2^h)^2=n^2+5n^2/4(1+5/4+\ldots(5/4)^{h-1})=$
$n^2+5n^2((5/4)^h-1)=n^2+5.5^{\lg n}-5n^2=\theta(n^{\lg 5})$

Soru (**10 puan**) T(n)=T(n/2)+T(n/3)+n rekürensi için T(n)=O(n) tahmininin doğru olduğunu gösteriniz. (T(1)=O(1) dir)

Çözüm. $T(n)\leq cn$ olduğunu tümevarımla gösterelim. n den küçük tüm değerler için doğru olsun. Bu durumda
$T(n/2) \leq cn/2$
$T(n/3) \leq cn/3$
eşitsizlikleri doğrudur. Buradan da
$T(n) \leq cn/2+cn/3+n\leq cn$ (*) olduğunu gösterirsek biter. (T(1)=O(1) olduğundan n=1 için doğrudur) Buradan da $1\leq c/6$ yani $c\geq 6$ seçilirse eşitsizliği her $n\geq 1$ için doğru olur.

Soru (20 puan) Eklemeli sıralama (insertion sort) algoritması özyinelemeli(recursive) olarak aşağıdaki gibi yapılabilir: A[1…n] dizisini sıralamak için A[1…n-1] dizisini sırala ve A[n] elemanını sıralanmış A[1…n-1] dizisine ekle.

- **a)** Özyinelemeli eklemeli sıralama (resursive insertion sort) algoritmasının sözde kodunu(pseudocode) yazınız
- **b)** Özyinelemeli eklemeli sıralama (resursive insertion sort) algoritmasının işlem zamanı için uygun rekürensi yazınız ve çözünüz.

Çözüm. a)

```
RECURSIVE-INSERTION-SORT (A,n)
1  if  n>1
2       RECURSIVE-INSERTION-SORT (A,n-1)
3       INSERT(A,n)

INSERT(A,k)
```

```
// Insert A[k] to the sorted A[1..k-1]
1     key=A[k]
2     i=k-1
3     while i>0 and A[i]>key
4           A[i+1] = A[i]
5           i=i-1
6     A[i+1] = key
```

b)

$T(1) = \theta(1)$

$T(n) = T(n-1) + \theta(n)$

Buradan da

$T(n) = T(n-1) + \theta(n) = T(n-1) + cn = T(n-2) + cn + c(n-1) = ...c(1+2+..n) = cn(n+1)/2 = \theta(n^2)$

Soru (20 puan) Aşağıdaki algoritmada F fonksiyonunun 2 giriş değişkeni vardır. S, elemanları birbirinden farklı olan tamsayılar dizisi ve x bir tamsayıdır.

F(S,x)

1. S'nin elemanlarını artan sıra ile sırala ve sonucu S'ye yaz
2. x sayısından S'nin elemanlarını teker teker çıkart ve sonucu bir A dizisine yaz
3. A dizisinin elemanlarını artan sıra ile sırala.
4. MERGE algoritmasını S ve A dizilerine uygula ve sonucu A dizisine yaz
5. A dizisinde değerleri birbirine eşit olan ardışık ikililerin sayısını hesapla ve k'ya yaz
6. k'yı geri gönder

a) F(S,x) algoritması hangi soruyu çözüyor, yani algoritmanın bulduğu k sayısının S ve x için anlamı nedir?
b) En kötü durum için F(S,x) algoritmasının her adımının en iyi işlem zamanını yazınız ve S' nin uzunluğu n olduğunda algoritma için T(n) işlem zamanını bulunuz.

Çözüm. a) S dizisinde toplamları x'e eşit olan sıralı ikililerin sayısını buluyor. S dizisinin sıralanmış hali $a_1, a_2, ...a_n$ olsun. Bu durumda A dizisinin sıralanmış hali $x-a_n, x-a_{n-1}, ...x-a_1$ olur. Eğer MERGE ettiğimizde ardışık 2 sayı eşit oluyorsa bunun anlamı $x-a_k = a_t$ olacak biçimde k ve t indislerinin varlığıdır yani $x = a_k + a_t$.

b)En kötü durum S dizisinin azalan sıra ile verilmesidir.

1. adım işlem zamanı en iyi olan sıralama algoritması seçilebilir. $\theta(n \lg n)$ olur.
2. adım $\theta(n)$
3. adım A dizisinin azalan bir dizi olduğunu kesin bildiğimizden bu diziyi artan sıra ile sıralamak için $\theta(n)$ işlem zamanı gerekir.
4. adım $\theta(n)$
5. adım $\theta(n)$
6. adım $\theta(1)$

Toplam işlem zamanı $\theta(n \lg n)$ olur.

Soru. n elemanlı artan sıralı A ve n elemanlı azalan sıralı B dizileri veriliyor.
a) (**10 puan**) Bu dizilerin toplam 2n sayıda elemanı arasında bir birinden farklı elemanların sayısını O(n) işlem zamanında bulabilen algoritma tasarlayınız.
b) (**10 puan**) Algoritmanızın sözde kodunu (pseudocode) yazınız.

Çözüm. a)Bu soruyu Merge algoritmasına benzer bir algoritma ile çözebiliriz. Başlangıçta birbirine eşit ikililerin sayısını tutmak için tanımladığımız counter isimli sayacımızın değerini sıfır olarak atarız. A dizisini baştan sona taramak için ilk değeri 1 olan bir k değişkeni, B dizisini sondan başa taramak için ilk değeri n olan bir m değişkeni tanımlarız. Sonra bir while döngüsü içinde (while koşulu $k \leq n$ and $m \geq 1$) A(k) ile B(m) i karşılaştırırız. Eğer A(k) büyük çıkarsa m i bir azaltırız, B(m) büyük çıkarsa k yı 1 artırırız, eşit çıkarlarsa counteri 2 artırırız. While döngüsünden çıktığımızda 2n-counter sayısını yazdırırız.

b)

```
counter←0
k←1
m←n
while (k≤n and m≥1)
  if A(k)>B(m)
        m←m-1
  else if A(k)<B(m)
        k←k+1
      else
        counter←counter+2
return (2n-counter)
```

Not: Bu çözümde A dizisinin elemanlarının kendi aralarında ve B dizisinin elemanlarının kendi aralarında farklı oldukları (A dizisi artan, B dizisi azalan olduklarından) dikkate alınmıştır.

Soru. n elemanlı bir A dizisi veriliyor. Bu dizinin elemanları arasında öyle x ve y ikilisini bulmak istiyoruz ki |x-y| ifadesi en büyük olsun.
a)(10 puan) Bu problemi $\boldsymbol{\theta(n^2)}$ işlem zamanında çözebilen algoritmanın sözde kodunu yazınız.
b)(**10 puan**) Bu problemi O(n) işlem zamanında çözebilen algoritma tasarlayınız.
c) (**10 puan**) Bu problemi O(n) işlem zamanında çözebilen algoritmanın sözde kodunu yazınız.

Çözüm.
a)Burada 2 döngü içinde dizinin elamanları arasında tüm mümkün ikililerin farklarının mutlak değerinin en büyüğünü buluruz.

```
Max←|A(1)-A(2)|
k←1
m←2
for i←1 to n-1
    for j←i+1 to n
      if  Max<|A(i)-A(j)|
            Max←|A(i)-A(j)|
            k←i
            m←j
return A(k) and A(m)
```

c)Bu soruda bulunması istenen aslında dizinin maksimum elemanı ile minimum elemanıdır. Bu elemanları O(n) işlem zamanında farklı yollarla bulabiliriz.

1.yol Önce n-1 karşılaştırma yaparak maksimum elemanı buluruz, sonra da kalan n-1 sayı arasında n-2 karşılaştırma yaparak minimum elemanı buluruz. Toplam 2n-3 karşılaştırma yapmış oluruz.

2.yol A dizisinin ilk 3 elemanını birbiriyle karşılaştırır, ortanca elemanı eleriz.(Yani en küçük ve en büyük elemanı sırasıyla Min ve Max isimli 2 değişkende tutarız). Sonra A dizisini 4. elemandan başlayarak tarar, sıradaki her elemana Min ve Max sayıları ile birlikte bakarız ve A dizisinin ilk 3 elemanı için yaptığımız işlemi bu 3 sayı için tekrarlarız. Tarama işlemi bittikten sonra Max ve Min sayılarını yazdırırız. Bu yolda ilk 3 sayı için 3 karşılaştırma, sonraki n-3 sayı için 2(n-3) karşılaştırma yapmış oluruz, yani toplamda 3+2(n-3)=2n-3 karşılaştırma ile buluruz.

3.yol A dizisinin elemanları arasında eleme usulü bir turnuva düzenleriz. Önce ikişerli karşılaştırırız. Sonraki her aşamada büyüklerle büyükleri, küçüklerle küçükleri karşılaştırır, bir sonraki aşamaya büyüklerin büyüğünü ve küçüklerin küçüğünü taşırız ve böyle devam ederek en büyük ve en küçük elemanı aynı zamanda buluruz. Örneğin, sayılar 3,9,4,12,6,5,11,10 olsun. İlk aşamada 3 ile 9, 4 ile 12, 6 ile 5, 11 ile 10 karşılaştırılır. 2. Aşamada 3 ile 4 (küçükler karşılaştırılması), 9 ile 12(büyükler karşılaştırılması), 6 ile 11(büyükler karşılaştırılması) ve 5 ile 10 (küçükler karşılaştırılması) karşılaştırılır ve bir sonraki aşamaya büyükler olarak 12 ile 11 ve küçükler olarak 3 ile 5 taşınır. Son aşamada da 12 ile 11 ve 3 ile 5 karşılaştırılır, 12 maksimum, 3 minimum eleman olarak bulunur.
Bu yolda $n=2^k$ olduğunu düşünürsek ilk aşamada 2^{k-1}, sonraki aşamalarda $2(2^{k-2}+2^{k-3}+\ldots 1)$ karşılaştırma olacaktır, yani toplam $2^{k-1}+2(2^{k-1}-1)=3.2^{k-1}-2=3.2^k/2-2=3n/2-2$ karşılaştırma yapılır.

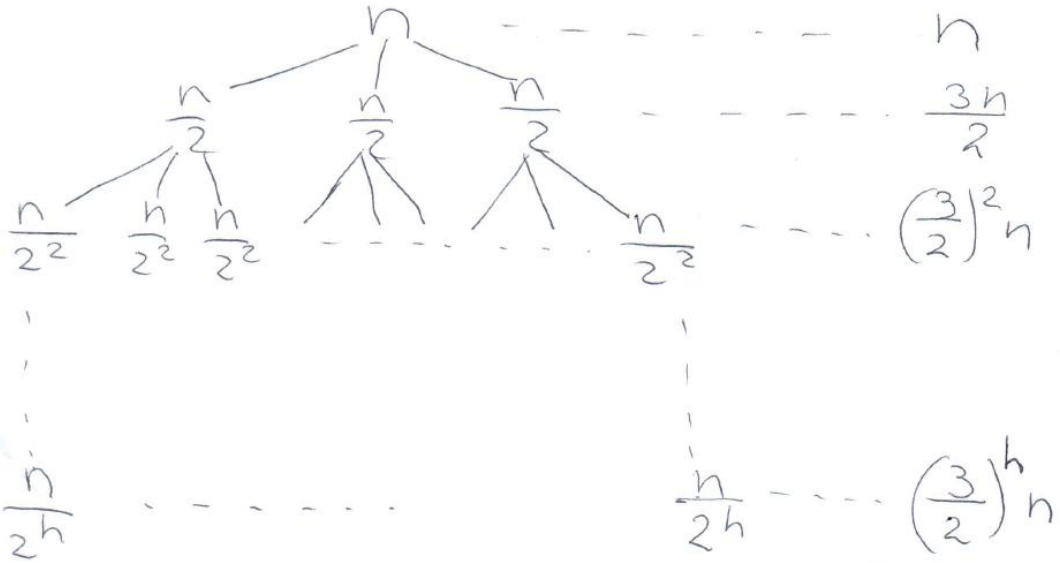
Soru

a) T(n)=3T(n/2)+n, n>1için
T(1)=1
rekürensini özyinemeli ağaç yöntemiyle çözünüz.

Soru 1. Çözüm

a) $T(n) = 3T(n/2) + n$, $n > 1$ için

$T(1) = 1$



$$T(n) = n\left(1 + \frac{3}{2} + \cdots \left(\frac{3}{2}\right)^h\right) = n \cdot \frac{\left(\frac{3}{2}\right)^{h+1} - 1}{\frac{3}{2} - 1}$$

$$= 2n \frac{3^h \cdot 3}{2^h \cdot 2} - 2n$$

$$\frac{n}{2^h} = 1 \Rightarrow 2^h = n \Rightarrow h = \log_2 n$$

$$\Rightarrow T(n) = \cancel{2n} \cdot \frac{3^{\log_2 n} \cdot 3}{\cancel{n \cdot 2}} - 2n = 3 \cdot 3^{\log_2 n} - 2n$$

$$= 3 \cdot n^{\log_2 3} - 2n = \Theta\left(n^{\log_2 3}\right)$$

b) $T(n)=9T(n/3)+n^2 \lg^3 n$, $n>1$için
$T(1)=1$
rekürensini master yöntemle çözünüz.

b) $T(n)=9T(n/3)+n^2\lg^3 n$, $n>1$ için
$T(1)=1$
$a=9$, $b=3$, $f(n)=n^2\lg^3 n$
$n^{\log_b a}=n^{\log_3 9}=n^2$ 2. durum master teor
$\Rightarrow T(n)=\Theta(n^2\lg^4 n)$

Soru(20=10+10 puan)

Aşağıdaki önermelerin doğru veya yanlış olduklarını bulunuz. Cevabınızı açıklayınız.

a)Asimptotik pozitif her f fonksiyonu için f(n/5)=O(f(5n))

a) Bu önerme yanlıştır. Örneğin, $a>1$ ve $f(n)=a^n$ olsun. $f(n)$ pozitif fonksiyondur.
$f(\frac{n}{5})=O(f(5n))$ koşulu doğuyse
$a^{n/5}\le c.a^{5n}$
eşitsizliği bir $c>0$ sabiti ve $\forall n>n_0$ için doğru olmalıdır. (n_0 bir sab.)
ama
$a^{5n-\frac{n}{5}}\le c$
$a^{\frac{24n}{5}}\le c$.
$a>1$ olduğundan $a^{\frac{24n}{5}}$ artan bir fonksiyon dur ve $\lim_{n\to\infty} a^{\frac{24n}{5}}=+\infty$, yani hiç bir sabitle üstten sınırlanamaz

b) f(n)=Ω(g(n)), g(n)=Ω(h(n)) ve h(n)=Ω(f(n)) ise f(n)=Θ(h(n)) dir. Burada f, g ve h fonksiyonları asimptotik pozitif fonksiyonlardır.

b) $f(n)=\Omega(g(n)) \Rightarrow \exists c_1>0, n_1>0$ vardır ki,

$\forall n \geq n_1$ için

$f(n) \geq c_1 g(n)$ (1)

$g(n)=\Omega(h(n)) \Rightarrow \exists c_2>0, n_2>0$ vardır ki, $\forall n \geq n_2$ için

$g(n) \geq c_2 h(n)$ (2)

$h(n)=\Omega(f(n)) \Rightarrow \exists c_3>0, n_3>0$ vardır ki, $\forall n \geq n_3$ için

$h(n) \geq c_3 f(n)$ (3)

(1) ve (2) den $\forall n \geq \max\{n_1, n_2\}$ için

$f(n) \geq c_1 g(n) \geq c_1 c_2 h(n)$

$c_1 c_2 = c_4$ alırsak yani $f(n) \geq c_4 h(n)$ (4)

---

(3)'ten $f(n) \leq \frac{1}{c_3} h(n)$ $\forall n \geq n_3$ için

$\frac{1}{c_3} = c_5$ olsun

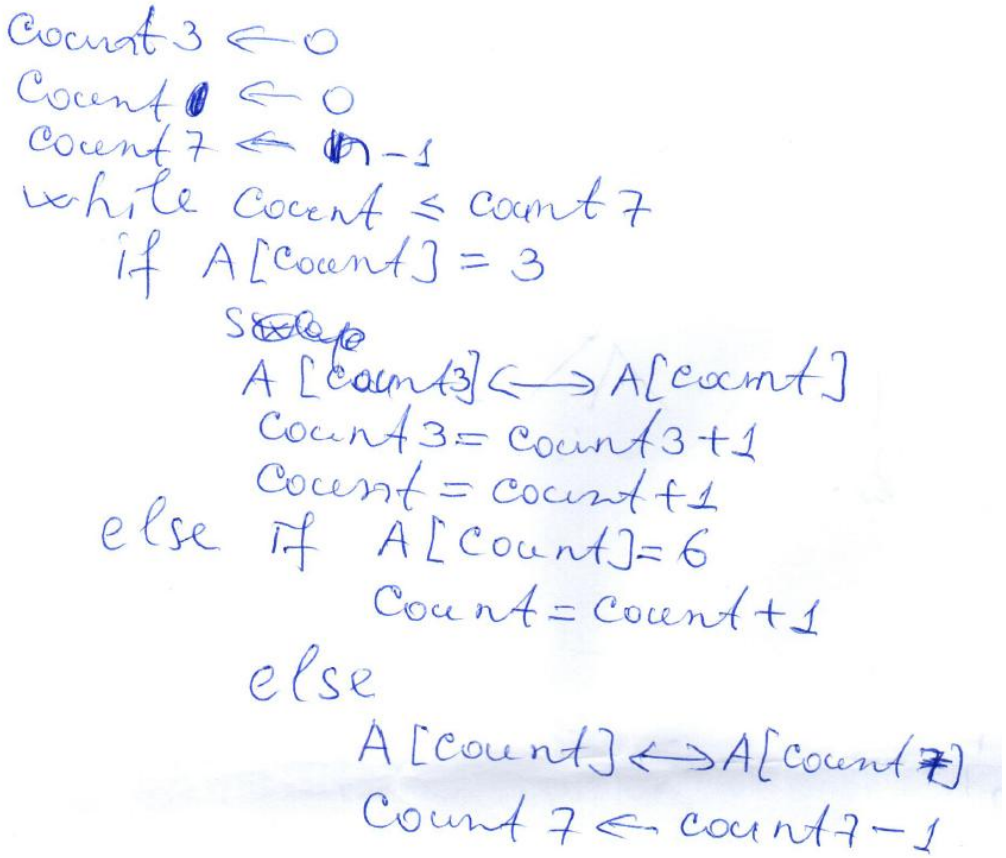
$\Rightarrow \forall n \geq \max\{n_1, n_2, n_3\}$ için

$c_4 h(n) \leq f(n) \leq c_5 h(n)$

Yani $f(n) = \Theta(h(n))$

Soru

Her terimi 3,6 veya 7 sayılarından biri olan n elemanlı bir diziyi doğrusal zamanda ve başka bir dizi kullanmadan sıralayabilen bir algoritma tasarlayınız.



Soru 4.

Her terimi bir rakam olan n elemanlı bir dizinin, 1 ile başlayan ve 9 ile biten tüm altdizilerinin sayısını bulma sorusunu ele alalım. (Örneğin, 3,1,9,1,1,5,9,3,1 dizisi için bu tür 4 altdizi vardır: 1,9 ve 1,9,1,1,5,9 ve 1,1,5,9 ve 1,5,9 altdizileri)

a)Bu soruyu çözmek için kaba kuvvet algoritması tasarlayınız. Algoritmanızın işlem zamanını hesaplayınız.

Çözüm a) i. ($0 \leq i < n-1$) pozisyonda bulunan 1 ile başlayan ve 9 ile biten istenen altdizilerin sayısı, bu 1 den sonra gelen 9'ların sayısına eşittir. Buna göre aşağıdaki algoritmayı yazabiliriz

```
count=0
for i=0 to n-2
    if A[i]==1
        for j=i+1 to n-1
            if A[j]==9
                count=count+1
```

b)a şıkkındaki çözümünüzden daha kısa zamanda çalışabilen bir algoritma tasarlayınız.

b) i. ($0 \le i < n-1$) pozisyonda bulunan 9 ile biten ve 1 ile başlayan istenen alt dizilerin sayısı, bu 9'un solunda bulunan 1'lerin sayısına eşittir. Buna göre aşağıdaki algoritmayı yazabiliriz:

```
Count = 0
Count1 = 0
for i = 0 to n-1
    if A[i] == 1
        Count1 = Count1 + 1
    if A[i] == 9
        Count = Count + Count1
```

Zaman analizi

a) İç içe 2 döngü var $O(n^2)$

b) Sadece 1 döngü var $O(n)$

Soru

Her birinde bir kilit olan n tane kapı veriliyor. Bazı kapıların kilitleri aynıdır. Bir kilit kapıların yarısından çoğunda bulunuyorsa bu kilide **esas kilit** denir. Her hangi 2 kapının kilitlerinin aynı olup olmadığını test eden bir cihazımız vardır. (Örnegin, 2 kapının kilidinin aynı olup olmadığını anlamak için cihazı önce ilk kapının kilidi üzerinde, sonra ise diğer kapının kilidi üzerinde tutuyoruz, kilitler aynı ise cihazın yeşil ışığı yanıyor, değilse kırmızı ışığı yanıyor. Cihaz aynı anda sadece 2 kapının kilidini belirlemek içindir.)

a) Esas kilidi (eğer varsa) bulmak için O(nlogn) işlem zamanında çalışan bir algoritma tasarlayınız.
b) Algoritmanızı önce AABCD dizisine sonra ise AACCCAAACCBCDCCC dizisine uygulayarak anlatınız. (Burada A, B, C , D gibi 4 farklı kilit vardır)
c) Algoritmanızın çalışma süresinin gerçekten O(nlgn) olduğunu kanıtlayınız.

Çözüm

a) MERGE-SORT algoritmasına benzer bir böl-yönet algoritması oluşturacağız.

Önce özyinelemeli olarak n tane kapıyı $\lfloor n/2 \rfloor$ ve $\lceil n/2 \rceil$ sayıda 2 parçaya ayırıyoruz ve bu işleme her parçada tek kapı kalana kadar devam ediyoruz. Bakılan kapı sayısı 1 ise bu kapının kilidi bu altdizi için esas kilittir. Şimdi 2 altdiziyi nasıl birleştireceği anlatalım. (MERGE işlemi)

4 durum vardır:

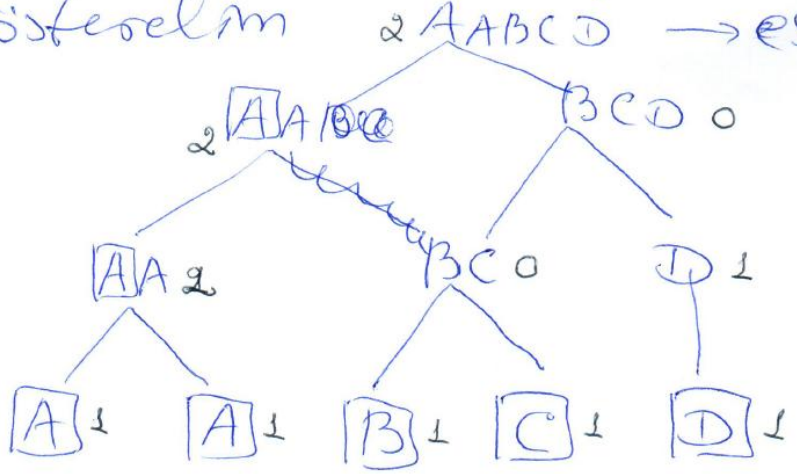
1. durum. Sol ve sağ parçada esas kilit yoktur. Bu durumda sol ve sağ parçanın birleştirilmesinden oluşan parçada da esas kilit olamaz

2. durum. Sol parçada esas kilit vardır, sağ parçada yoktur. Bu durumda sol parçadaki esas kilidi, sağdaki elemanlarla tek tek karşılaştırırız ve bu kilidin birleştirilmiş dizide de esas kilit olup olmadığını anlarız

3. durum Sol parçada esas kilit yoktur, sağ parçada vardır. Bu durum, 2. durumun simetriğidir, benzer yolla sağdaki esas kilidin birleştirilmiş dizide esas kilit olup olmadığını anlarız

4. durum. Sağ ve sol parçanın her birinde esas kilit vardır. Bu durumda soldaki esas kilidi sağ parçanın elemanları ile teker teker karşılaştırırız. Bu eleman esas kilit değilse bu defa sağdaki esas kilidi soldaki elemanlarla teker teker karşılaştırırız.

b) Algoritmayı önce AABCD dizisine uygulayalım. Her adımda esas kilidi dikdörtgen içinde gösterelim



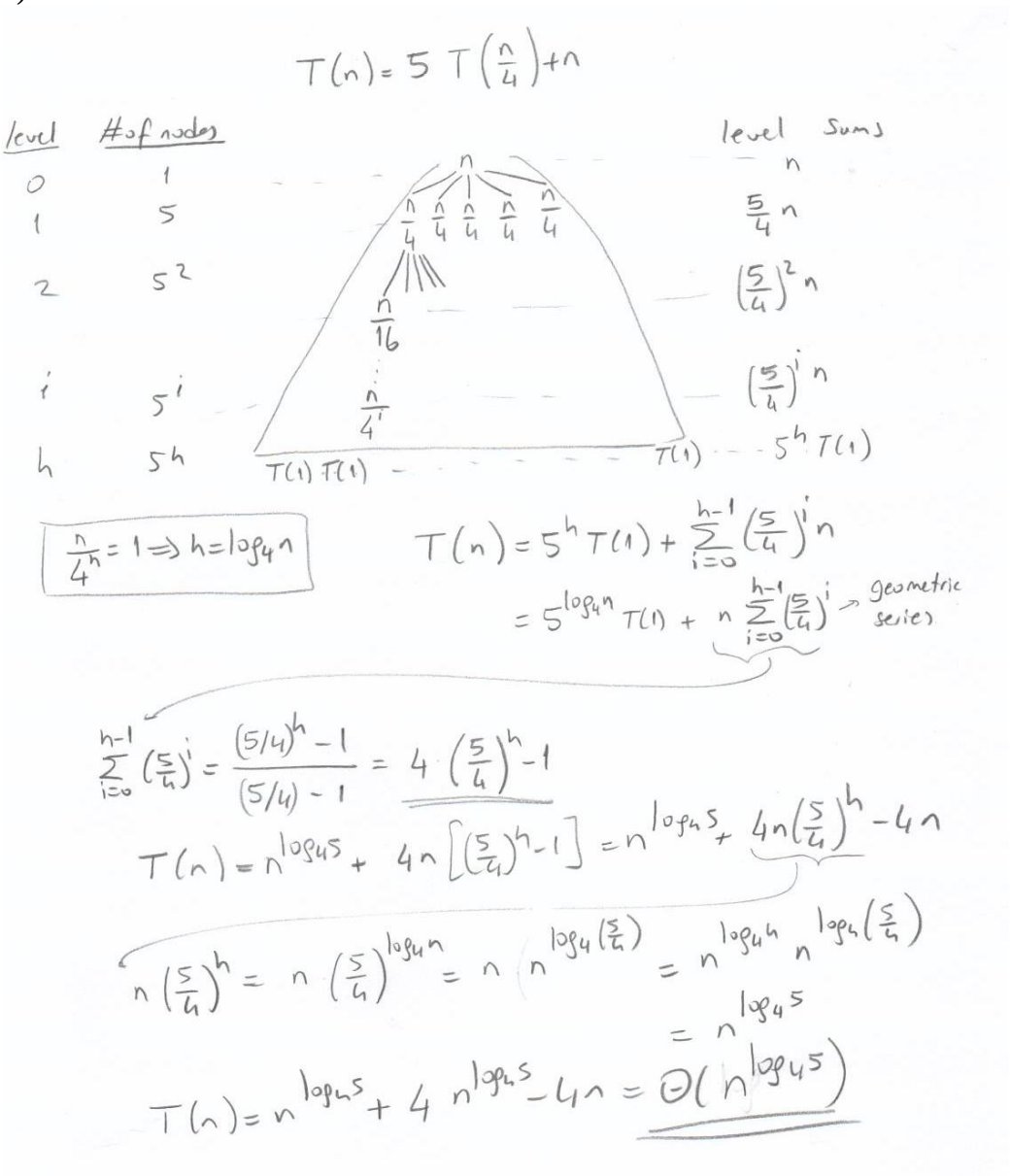
Q1. Solve the recurrence $T(n) = 5T(n/4) + n$

a) Using Master theorem [5 points]

b) Recurrence tree method. [10 points]

a) a=5, b=4, f(n)=n, $n^{\log_b a} = n^{\log_4 5} > n$. $\Theta(n^{\log_4 5})$.

b)



Q2. Consider ***ternary search***—the following algorithm for searching in a sorted array $A[0..n-1]$.

If $n = 1$, simply compare the search key K with the single element of the array; otherwise, search recursively by comparing K with $A[\lfloor n/3 \rfloor]$, and if K is larger, compare it with $A[\lfloor 2n/3 \rfloor]$ to determine in which third of the array to continue the search.

a) Set up a recurrence for the number of key comparisons in the worst case. Solve the recurrence for $n = 3^k$. [10 points]

b) Compare this algorithm's efficiency with that of binary search by solving the recurrence of binary search for $n=2^k$. Determine which algorithm has fewer comparisons. [10 points]

a. $C(n) = 2 + C(n/3)$ for $n = 3^k$ $(k > 0)$, $C(1) = 1$.

$$C(3^k) = 2 + C(3^{k-1}) \quad [\text{sub. } C(3^{k-1}) = 2 + C(3^{k-2})]$$
$$= 2 + [2 + C(3^{k-2})] = 2 \cdot 2 + C(3^{k-2}) = \quad [\text{sub. } C(3^{k-2}) = 2 + C(3^{k-3})]$$
$$= 2 \cdot 2 + [2 + C(3^{k-3})] = 2 \cdot 3 + C(3^{k-3}) = \ldots = 2i + C(3^{k-i}) = \ldots =$$
$$2k + C(3^{k-k}) = 2\log_3 n + 1.$$

b. We have to compare this formula with the worst-case number of key comparisons in the binary search, which is about $\log_2 n + 1$. Since

$$2\log_3 n + 1 = 2\frac{\log_2 n}{\log_2 3} + 1 = \frac{2}{\log_2 3}\log_2 n + 1$$

and $2/\log_2 3 > 1$, binary search has a smaller multiplicative constant and hence is more efficient (by about the factor of $2/\log_2 3$) in the worst case, although both algorithms belong to the same logarithmic class.

Q3. Let $A[0..n-1]$ be an array of n real numbers. A pair $(A[i], A[j])$ is said to be an ***inversion*** if these numbers are out of order, i.e., $i < j$ but $A[i] > A[j]$. Design an algorithm for counting the number of inversions. For your algorithm, find the total number of comparisons made on the elements of the given array. [15 points]

```
counter =0

for i = 0 to n - 1

    for j = i+1 to n
        if( A[i] > A[j] )
            then counter=counter+1

return counter
```

Total comparisons: (n-1)+(n-2)+…+2+1=n(n-1)/2

Q4. You are given **3 different algorithms,** each of which can be used to solve the same problem. After analyzing the time efficiencies of these three algorithms, indicate which one you would choose.

Algorithm A : It solves the problem *recursively* by dividing the problem into *5 sub-problems*. The input sizes of the sub-problems are equal to *the half of the main problem*. Algorithm A combines the solutions of sub-problems in *linear time* to determine main solution.

Algorithm B : It solves the problem *recursively* by dividing the problem into *2 sub-problems*. The input sizes of the sub-problems are *one less than the main problem*. Algorithm B combines the solutions of sub-problems in *constant time* to determine main solution.

Algorithm C : It solves the problem *recursively* by dividing the problem into *9 sub-problems*. The input sizes of the sub-problems are equal to *the one third of the main problem*. Algorithm C combines the solutions of sub-problems in *quadratic time* to determine main solution.

Solution: The recurrence relations for the three algorithms are,

Alg1

$$T(n) = 5T(\frac{n}{2}) + n$$

Using Master Theorem, $a = 5, b = 2, c = 1$ and $\log_2^5 > 1$. Hence the running time is $\Theta(n^{\log_2^5})$.

Alg2

$$T(n) = 2T(n-1) + 1$$

Expanding the recurrence,

$$T(n) = \begin{matrix} 1 + 2 + 2^2 + 2^3 T(n-4) \dots \\ 1 + 2 + 2^2 + 2^3 + \dots + 2^{n-1} T(0) \end{matrix}$$

Hence the running time is $O(2^n)$.

Alg3

$$T(n) = 9T(\frac{n}{3}) + n^2$$

Using Master Theorem, $a = 9, b = 3, c = 2$ and $\log_3^9 = 2$. Hence the running time is $\Theta(n^2 \log n)$.

So from above three algorithm, we can see that time complexity of third algorithm is best. So we will choose algorithm C.

Soru (20=4+4+4+4+4 puan) Bu soruda verilen fonksiyonlar asimptotik pozitif fonksiyonlardır. Aşağıdaki iddiaları ispatlayınız veya yanlış olduğunu gösteriniz:

a) Her $f(n)$ ve $g(n)$ için $f(n) = O(g(n))$ veya $g(n) = O(f(n))$ dır.
b) Her $f(n)$ ve $g(n)$ için
$O(f(n) + g(n)) = f(n) + O(g(n))$ dir.
c) Eğer $f(n) = O(g(n))$ ise $\log f(n) = O(\log g(n))$ dir.
d) Eğer $f(n) = \theta(g(n))$ ve $h(n) = O(f(n))$ ise $h(n) = O(g(n))$
e) Eğer $f(n) = \theta(g(n))$ ve $h(n) = O(g(n))$ ise $h(n) = O(f(n))$

Çözüm.

a) Bu önerme yanlıştır. Örneğin $f(n) = \begin{cases} 1, & n \text{ çift ise} \\ 0, & n \text{ tek ise} \end{cases}$ ve $g(n) = \begin{cases} 1, & n \text{ tek ise} \\ 0, & n \text{ çift ise} \end{cases}$ fonksiyonları için doğru değil

b) Bu önerme yanlıştır. Örneğin, $f(n)=n^2$, $g(n)=n$ ve $O(f(n)+g(n))=2n^2+3n$

c) Bu önerme yanlıştır. Örneğin, $f(n)=3^{\frac{n+1}{n}}$ ve $g(n)=2^{\frac{1}{n}}$ olsun. Önce $f(n)=O(g(n))$ olduğunu gösterelim. $f(n)=3^{\frac{n+1}{n}} \le c.2^{\frac{1}{n}}$ eşitsizliğini sağlamaya çalışalım. Buradan $3.3^{\frac{1}{n}} \le c.2^{\frac{1}{n}}$ sağlanmalıdır. $3.\left(\frac{3}{2}\right)^{\frac{1}{n}} \le c$ sağlanmalıdır. $\left(\frac{3}{2}\right)^{\frac{1}{n}} \le \frac{3}{2}$ olduğundan $\frac{9}{2} \le c$ alınırsa son eşitsizlik sağlanır. $\log f(n)=\frac{n+1}{n}\log 3 \le c_1 \frac{1}{n}\log 2 = c_1 \log g(n)$ olabilmesi için $(n+1)\frac{\log 3}{\log 2} \le c_1$ sağlanmalıdır ama bu da imkansızdır. (Sağ taraf sabit, sol taraf sonsuza yaklaşmaktadır)

d) $f(n)=\theta(g(n))$ ve $h(n)=O(f(n))$ olduğundan öyle c_1, c_2, c_3 pozitif tamsayıları vardır ki belirli bir n_0 sayısından sonraki her n için

$c_1 g(n) \le f(n) \le c_2 g(n)$ ve $h(n) \le c_3 f(n)$ eşitsizlikleri sağlanmaktadır. Buradan

$h(n) \le c_3 f(n) \le c_3 c_2 g(n)$

e) $f(n)=\theta(g(n))$ ve $h(n)=O(g(n))$ olduğundan öyle c_1, c_2, c_3 pozitif reel sayıları vardır ki belirli bir n_0 sayısından sonraki her n için

$c_1 g(n) \le f(n) \le c_2 g(n)$ ve $h(n) \le c_3 g(n)$ eşitsizlikleri sağlanmaktadır. Buradan da

$h(n) \le c_3 g(n) \le \frac{c_3}{c_1} f(n)$

Soru (15=5+5+5 puan) Aşağıdaki bağıntıları sağlayan asimptotik pozitif $f(n)$ ve $g(n)$ fonksiyonlarını bulunuz veya olamayacağını gösteriniz.

a) $f(n)=o(g(n))$ ve $f(n) \ne \theta(g(n))$

b) $f(n)=\theta(g(n))$ ve $f(n)=o(g(n))$

c) $f(n)=\theta(g(n))$ ve $f(n) \ne O(g(n))$

Çözüm. a) $f(n)=n, g(n)=n^2$

b) $f(n)=\theta(g(n))$ ve $f(n)=o(g(n))$ olduğundan öyle c_1, c_2 pozitif reel sayıları vardır ki belirli bir n_0 sayısından sonraki her n için

$c_1 g(n) \le f(n) \le c_2 g(n)$ ve her $c>0$ için öyle bir n_1 pozitif tam sayısı vardır ki her $n > n_1$ için $f(n) < cg(n)$ eşitsizlikleri sağlanmaktadır. $0 < c_3 < c_1$ koşuluna uyan c_3 alalım. Buradan her $n > \max(n_1, n_0)$ için

$c_3 g(n) < c_1 g(n) \le f(n) < c_3 g(n)$ çelişki yani verilen 2 bağıntı aynı zamanda sağlanamaz

c) $f(n)=\theta(g(n))$ olduğundan öyle c_1, c_2 pozitif reel sayıları vardır ki belirli bir n_0 sayısından sonraki her n için

$c_1 g(n) \le f(n) \le c_2 g(n)$

Yani $f(n)=O(g(n))$ dolayısıyla her 2 koşul aynı zamanda sağlanamaz

Soru (15=5+10 puan) Aşağıdaki bağıntıları
$T(1)=1,$ ve tüm $n \geq 2$ için $T(n)=2T(n/2)+3n+1$

a) Master teoremin yardımıyla
b) Kesin olarak (Bu durumda n sayısının 2 nin kuvveti olduğunu düşünebilirsiniz.) çözünüz.

Çözüm.a) a=2,b=2, $\log_b a=1,$ $3n+1=\theta(n)$ Master teoreminden $T(n)=\theta(n\log n)$

b) $T(n)=2T(n/2)+3n+1=2(2T(n/4)+3n/2+1)+3n+1=4T(n/4)+3n+2+3n+1=$
$4(2T(n/8)+3n/4+1)+3n+2+3n+1=8T(n/8)+3n+4+3n+2+3n+1$

i adımdan sonra

$T(n)=2^iT(n/2^i)+3in+\sum_{j=0}^{i-1}2^j$ bulunur.

i=logn yazalım. $T(n)=nT(1)+3n\log n+\sum_{j=0}^{i-1}2^j=n+3n\log n+2^{\log n}-1=2n+3n\log n-1$

Soru (20=5+15 puan) Aşağıdaki algoritma $S[1...n]$ dizisinin bir değerini buluyor. Ana programda Mystery$(1,n)$ kullanımı vardır.

```
Mystery ( x, y )
if y − x ≤ 1 and S[ x ] ≤ S[ y ]
        return S[ y ]
if y − x ≤ 1 and S[ x ] > S[ y ]
        return S[ x ]
a1=Mystery( x, ⌊(x + y) / 2⌋)
a2=Mystery(⌊(x + y) / 2⌋+1, y )
if a1 ≤ a2
        return a2
return a1
```

a) Bu algoritma dizinin hangi değerini buluyor?
b) Mystery$(1,n)$ fonksiyonunda karşılaştırmaların sayısı için indirgemeli bir bağıntı (recurrence) yazınız.. Bu bağıntıyı master teoremin yardımıyla ve kesin olarak çözünüz (n sayısının 2 nin kuvveti olduğunu düşünebilirsiniz)

Çözüm. a)Maksimum değerini buluyor.
b) $T(1)=0$ ve her n>1 için $T(n)=2T(n/2)+1$ dir. Master teoremden $T(n)=\theta(n)$.
Kesin çözüm T(n)=n-1 dir.

Soru. Aşağıdaki algoritmanın işlem süresi analizini yapınız.

```
for i←1 to n-1
for j← i+1 to n
   if A[i] > A[j] then
      A[i]↔A[j]
```

Çözüm.
i=1 için n-1, i=2 için n-2, … , i=n-1 için 1 karşılaştırma yapılır.
$n-1+n-2+n-3+\ldots 1=(n-1)n/2=\theta(n^2)$

Soru. Aşağıdaki algoritmanın en kötü durumda işlem süresi analizini yapınız.

```
for i←1 to n-1
   Switch←False
   for j←1 to n-i
      if A[j] > A[j+1] then
         A[j] ↔A[j+1]
         Switch = True
    if Not Switch then break
```

Çözüm. En kötü durumda i=1 için n-1, i=2 için n-2, … , i=n-1 için 1 karşılaştırma yapılır.
$n-1+n-2+n-3+\ldots 1=(n-1)n/2=\theta(n^2)$

Soru. Aşağıdaki algoritmanın işlem süresi analizini yapınız:

```
i←n
count←0
while(i>0)
   for j←0 to i-1
      count←count+1
   i←floor(i/2)
```

Çözüm.
Hesaplamayı kolay yapabilmek $n=2^k$ olduğunu varsayalım. i=n için for döngüsü n defa, i=n/2 için for döngüsü n/2 defa, i=n/4 için for döngüsü n/4 defa,…i=1 için for döngüsü 1 defa çalışacaktır. Yani
$T(n)=2^k+2^{k-1}+\ldots 1=2^{k+1}-1=2n-1=\Theta(n)$

Soru 4. Aşağıdaki algoritmayı ele alalım

```
Algoritma(n)
for i ← 1 to n
    if (n mod i = 0) then
                    for j ← 1 to n
                        print('*')
```

a) Ahmet bu algoritmanın işlem süresinin $T(n) = \Theta(n^2)$ olduğunu iddia ediyor. Ahmet haklı mıdır? Yanıtınızı kanıtlayınız.

Çözüm Yani Ahmet

$$c_1 n^2 \leq T(n) \leq c_2 n^2 \quad \forall n \geq n_0$$

olacak biçimde $c_1, c_2 > 0$ ve n_0 sabitlerinin olduğunu iddia ediyor.

Ancak $n = p$ asal sayı olursa

if(n mod i = 0) koşulu sadece $i = 1$ ve $i = n$ için doğru olacaktır.

Yani içerideki for $j \leftarrow 1$ to n döngüsü sadece $i = 1$ ve $i = n$ için çalışacaktır.

$\Rightarrow$ Asal n'ler için algoritmanın çalışma süresi $\Theta(2n) = \Theta(n)$ olacaktır. Bu durumda n'leri asal seçersek

$$c_1 n^2 \leq T(n)$$ eşitsizliği sağlanamaz

$\Rightarrow$ Ahmet haklı değildir.

b) Mehmet bu algoritmanın işlem süresinin $T(n)=\Theta(n)$ olduğunu iddia ediyor. Mehmet haklı mıdır? Yanıtınızı kanıtlayınız.

Çözüm: n sayısı asal sayıdan farklı ise içteki for $j \leftarrow 1$ to n döngüsü n'in sayılarının sayısı kadar çalışacaktır. Örneğin, $n=2^k$ alırsak bu döngü $\log_2 n$ kadar çalışacaktır ve bu durumda işlem süresi $O(n \log_2 n)$ olacaktır. Yani Mehmet haklı değildir.

c) Ali bu algoritmanın işlem süresinin $T(n)=O(n^2)$ olduğunu iddia ediyor. Ali haklı mıdır? Yanıtınızı kanıtlayınız.

Çözüm Ali haklıdır, sonuçta 2 tane iç içe n boyutlu döngü var ve koşul her zaman doğru olsa bile işlem süresi $O(n^2)$ yi aşamaz.

Soru6. (10 puan) Aşağıdaki algoritmanın işlem zamanının asimptotik değerini n cinsinden bulunuz.

```
Algorithm(n)
    i=1
    s=1
    while (s<=n)
        do
            i=i+1
            s=s+i
            print('*')
```

Çözüm. Algoritmanın işlem zamanı while döngüsünün kaç defa çalışacağına bağlıdır. Bu döngünün koşuluna göre i nin alabileceği son değer k olsun. Bu durumda algoritma s=2+3+…k=k(k+1)/2-1=(k^2+k-2)/2 değerini hesaplar. Döngünün durma koşulu s<=n yani k^2+k-2<=n buradan da k^2+k-2-n<=0 olur. Bu eşitsizliğin doğru olabilmesi için k sayısının k^2+k-2-n=0 denkleminin kökleri arasında olması gerekmektedir. Yani k sayısı bu denklemin büyük kökünden küçüktür. Bu denklemin büyük kökü $\frac{-1+\sqrt{9+4n}}{2} = O(\sqrt{n})$ dir. Yani $k <= O(\sqrt{n})$ Bu da işlem zamanının $O(\sqrt{n})$ olması demektir.