Array ADT

Data

Array Space
Size
length (No:of elements)

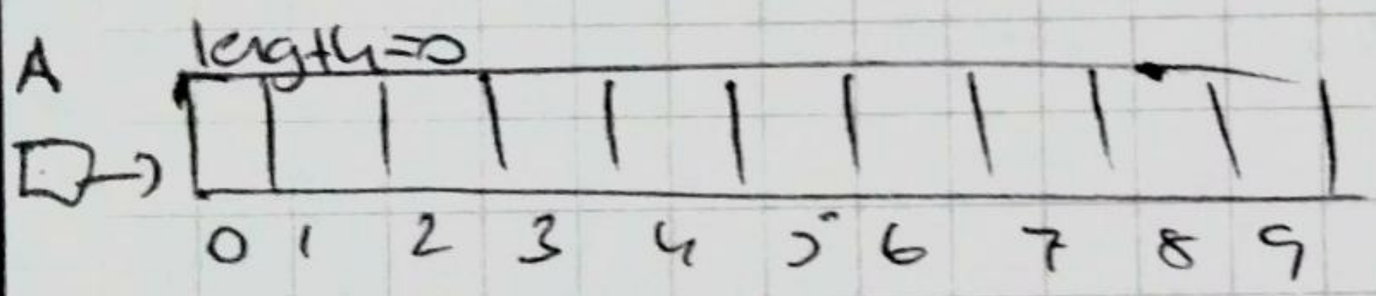
Operations

1- Display()
2- Add(x), Append(x)
3- Insert (index, x)
4- Delete (index)
5- Serch (x)
6- Get (index)
7- Set (index, x)
8- Max(), Min()
9- Reverse()
10- Shift(), Rotate()

① int A[10]; → stack

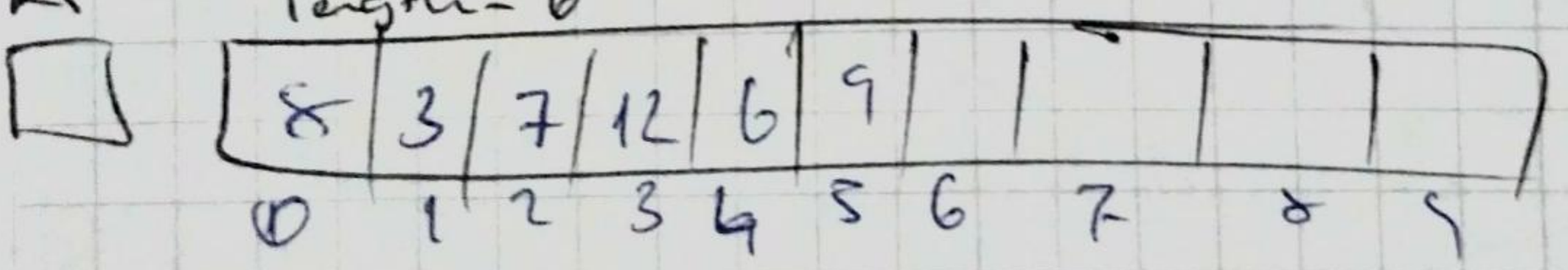
② int * A;
A = new int[size];

size = 10
length = 0



1- Display

size = 10
length = 6



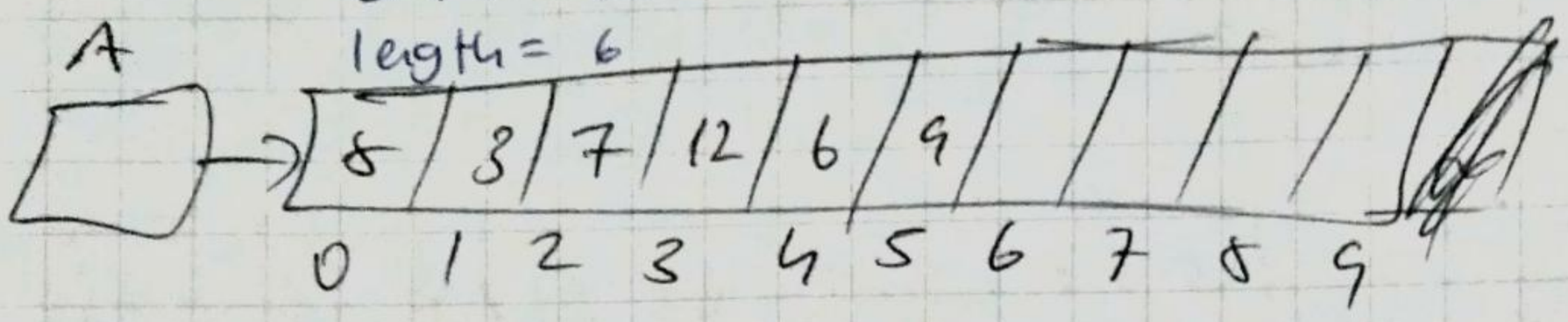
Display

```
for (i=0; i<length; i++)
        print(A[i]);
```

Scanned by TapScanner

2. Add(x) / Append(x)

size = 10
length = 6

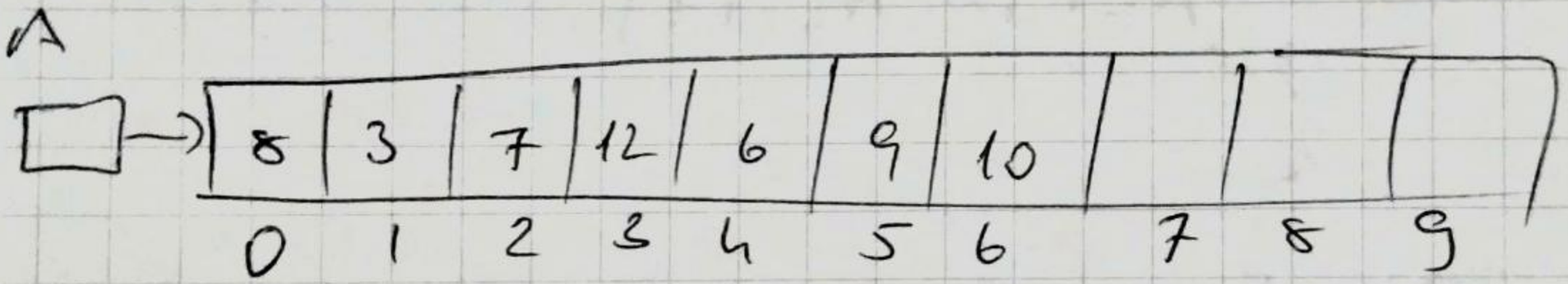


A[length] = x — 1
length ++; — 1

size = 10
length = 7

O(1)

3. Insert(index, x)



Insert(4, 15)

~~(i = length ; i > index~~

Not burda dizide son yer olduğu varsayılır

(i = length ; i > index ; i --) {

} A[i] = A[i-1]; — 0 - n

A[index] = x; — 1

length ++;

min O(1)
max O(n)

bu durumda kaydırma işlemi yoktur direkt boş yere atanır

Scanned by TapScanner

4 - Deleting

biggest = 10
utilized = 8

A

0	1	2	3	4	5	6	7	8	9
8	3	7	12	15	6	9	10		

Delete (3) index

x = A[index] — 1 Dil değişkeni kaç kez değişir

for (i=index ; i < length-1 ; i++) {

A[i] = A[i+1]

}

O(1)
O(n)

length-- ; — 1

ne kadar sayı silinse döngü n kere çalışır.

Son elemen silinirse döngü 1 kere çalışır

Not Dizideki indexler arasında boşluk olmamalı yani

1	2	—	3	4	5

hatalı kullanım

Time / best O(1)
worse O(n)

Scanned by TapScanner

5 - Search

1- linear Search
2- Binary Search

1- linear Search

size 10;
length 10;

A

8	9	4	7	6	3	10	5	14	2
0	1	2	3	4	5	6	7	8	9

successful → Key = 5

unsuccessful → Key = 12

Best - $O(1)$ → 1st element

Worst - $O(n)$ → son element

Avg - $O(n)$ → $\frac{1+2+3+\dots+n}{n}$ → ortalamalı element

$= \frac{\frac{n(n+1)}{2}}{n}$

$= \frac{n+1}{2}$

pscode

```
for( i=0 ; i<length ; i++) {
    if ( Key == A[i] )
        return i;
}
return -1;
```

Scanned by TapScanner

PERA

improve linear search

size = 10
length = 10

A

8	9	4	7	6	3	10	5	14	2
0	1	2	3	4	5	6	7	8	9

1- Transposition

Bir elemanın kullanılma sayısına göre ön alma işlemi. Yani bir eleman aranduğı zaman o eleman dizideki kendinden bir önceki veriyle yer değiştirir.

```
for (i=0; i<length; i++){
    if (key = A[i])
        swap (A[i], A[i-1])
        return i-1
    }
}
```

element yukarı bir şekilde ilerler

bu bir sıralama değildir.

2- Move to front / Head

Burada ise çağrılan elemanı en baştaki elemanla yer değiştirir.

```
for(i=0; i<length; i++) {
    if (key==A[i]) {
        swap (A[i], A[0]
    } return 0;
```

burada ise zaman daha hızlı ilerleyecek eğer sıradaki aranacak, aynı elemanı çağırırsa sabit zamanlı işlem yapacak

Binary Search

Size = 15
Length = 15

A

4	8	10	15	18	21	24	27	29	33	34	37	39	41	43
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
l							m							h

-) Binary Search için dizinin sıralı olması gerekir

-) 3 tane değişkene ihtiyacımız vardır.

l (low), h (high), mid (middle) $= \left[\frac{l+h}{2}\right]$

Key = 18 ✓		
l	h	m
0	14	7
0	6	3
4	6	5
4	4	4 found

Key 31 ✓		
l	h	m
0	14	7
8	14	11
8	10	9
10	10	10 found

Key 25		
l	h	m
0	14	7
8	6	3
4	6	5
6	6	6
7	6	✗ error

Scanned by TapScanner

Binary Search

Size = 15
Length = 15

A

4	8	10	15	18	21	24	27	29	33	34	37	39	41	43
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
l							m							h

-) Binary Search için dizinin sıralı olması gerektir

-) 3 tane değişkene ihtiyacımız vardır.

l (low), h (high), $mid = \left[\frac{l+h}{2}\right]$ (middle)

Key = 18 ✓			Key 31 ✓			Key 25		
l	h	m	l	h	m	l	h	m
0	14	7	0	14	7	0	14	7
0	6	3	8	14	11	8	6	3
4	6	5	8	10	9	4	6	5
4	4	4	10	10	10	6	6	6
		found			found	7	6	✗ error
							✗	

Scanned by TapScanner

PERA

Algorithm BM Search (l, h, key)

```
{
  while(l<=h) {
      mid = [(l+h)/2];
      if( key == A[mid]);
            return mid;
      else if ( key < A[mid]);
            h= mid-1;
      else
            l=mid+1;
  }
      return -1;
}
```

loop

Algorithm RBM Search (l, h, key) {

```
  if(l<=h) {
      mid = [(l+h)/2];
      if( key == A[mid])
            return mid
      else if ( key < A[mid])
            return RBMSearch (l, mid-1, key)
      else
            return RBM search (mid+1, h, key)
  }
  return -1; }
```

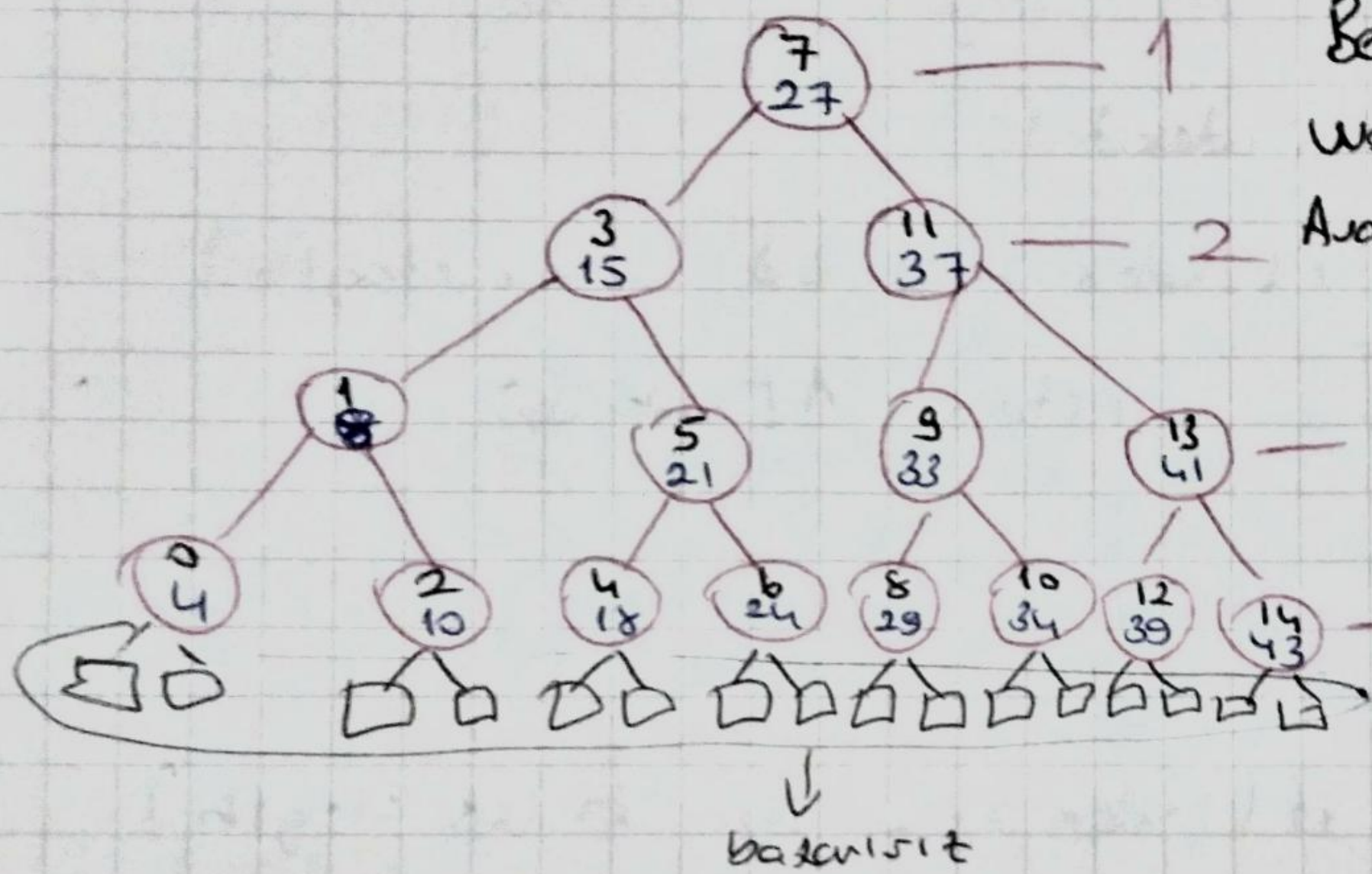
recursive

Analysis of Binary Search

PERA

size = 15
length = 15

A	4	8	10	15	18	21	24	27	29	33	34	37	39	41	43
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14



Successful

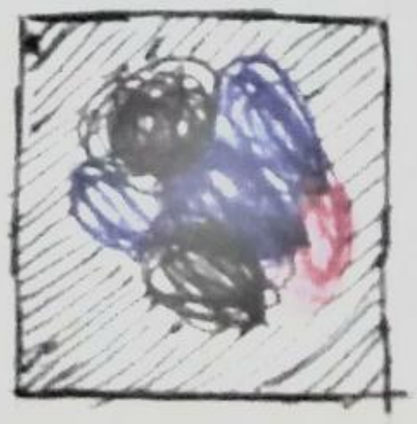
Best - min - $O(1)$

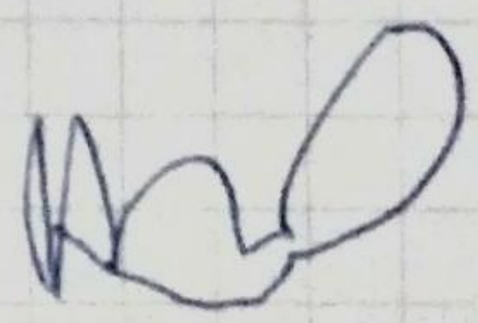
Worst - max - $O(\lg n)$

Average $O(\log n)$

$\log n$

$\left[\log_2(n+1)\right]$ → genel formül





Scanned by TapScanner

PERA

6- Get, Set

size = 15
length = 15

A

4	8	10	15	18	21	24	27	29	35	34	37	39	41	43
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

1.

```
Get (index)
    if(index >=0 && index <length)
        return A[index];
```

2.

```
Set (index, x)
    if(index >=0 && index <length)
        A[index] = x;
```

7- Max, Min

max karışık listeler içinde zaten sıralı listelerin ya başı yada sonu en büyüktür.

```
max( )
    1  —  max = A[0]
    n  —  for (i=1 ; i<length ; i++) {
f(n)=2n+1      n-1  —  if( A[i] > max)
O(n)                       max=A[i]
                    }
        1  —  return max; }
```

Scanned by TapScanner

min ( )

1 — min = A[0]

n — for (i=1 ; i<length ; i++) {

n-1 — if(A[i]<min)

f(n)=2n+1 1 — 1 min = A[i];

O(n) 5

1 — return min ;

8- sum

sum( )

n — total=0

n+1 — for ($\underbrace{i=0}_{1}$; $\underbrace{i<length}_{n+1}$; $\underbrace{i++}_{n}$) {

n — total = total + A[i];

2

1 — return total ;

f(n) = 3n+2

O(n)

Scanned by TapScanner

PERA

$$sum(A,n) = \begin{cases} 0 & n<0 \\ sum(A,n-1)+A[n] & n>=0 \end{cases}$$

```
int sum(A,n)
        {
        if(n<0)
            return 0;
        else
        return Sum(A,n-1)+A[n] ;
        }
```

call Sum(A,length

9- Avg

```
Avg()
  total=0;
  for (i=0 ; i<length ; i++) {
        total = total+ A[i];
  }
  return total/n;
```

Scanned by TapScanner

Reverse and Shift an Array

size=10
length=10

	0	1	2	3	4	5	6	7	8	9
A	8	3	9	15	6	10	7	2	12	4

1- Reverse
2- left shift
3- left rotate
4- right shift
5- right rotate

1. yol

Yardımcı bir dizi oluşturulur. dizi tersten A'yı kopyalar ardından da A = B yapılır.

Reverse

B =

0	1	2	3	4	5	6	7	8	9
4	12	2	7	10	6	15	9	3	8

```
for (i=length-1, j=0 ; i>=0 ; i--, j++
    B[j]=A[i] ;
```

n

```
for (i=0 ; i<length ; i++)
    A[i]=B[i]
```

n

2n O(n)

2- yol indexler yer değiştirilir

```
for (i=0, j-length-1 ; i<j ; i++, j--)
{
    temp = A[i]
    A[i] = B[j]
    A[j] = temp
}
```

Scanned by TapScanner

Check if Array is Sorted

A

0	1	2	3	4	5	6	7	8	9
4	8	13	16	20	25	28	33		

1. Sıralı dizige eleman ekleme
2. Aracın sıralı olduğunu kontrol etme
3. Dizide bulunan pozitif negatif sayıların sağa sola kaydırma

(1)

A

0	1	2	3	4	5	6	7	8	9
4	8	13	16	20	25	28	33		

insert - 18

```
x=18;
i=length-1;

while ( A[i]>x) {

    A[i+1]=A[i];
      i--;
  }

A[i+1] = x;

}
```

Scanned by TapScanner

2-

4	8	13	26	20	25	28	33		
0	1	2	3	4	5	6	7	8	9

8

```
Algoritmo isSorted (A, n)
{
    for (i=0; i<n-1; i++)
    {
        if (A[i] > A[i+1])
            return false;
    }
    return true;
}
```

3-

-6	3	-8	10	5	-7	-9	12	-4	2
0	1	2	3	4	5	6	7	8	9
	i								j

```
i=0;
j= length-1;
    while(i<j) {
        while (A[i]<0) {i++;}
        while (A[j]>=0) {j--;}
        if(i<j) {
            swap (A[i], A[j])
        }
    }
```

Scanned by TapScanner

PERA Merging Arrays

A

3	8	16	20	25
0	1	2	3	4

i m

B

4	10	12	22	23
0	1	2	3	4

n j

Sıralı bir dizide birleştirme

C

3	4	8	10	12	16	20	22	23	25
0	1	2	3	4	5	6	7	8	9

k

$\Theta(m+n)$

i=0
j=0
k=0

```
while (i<m && j<n)
{
    if (A[i] < B[i])
        c[k++]= A[i++];
    else
        c[k++]= B[j++];
}
```

burada dizi elemanı karşılaştırılır ve son bir elemanı alınan dizinin bir sonraki elemanı alınır

```
for (; i<m; i++)
    c[k++]=A[i];
for (; j<n; j++)
    c[k++]=B[j]
}
```

Scanned by TapScanner

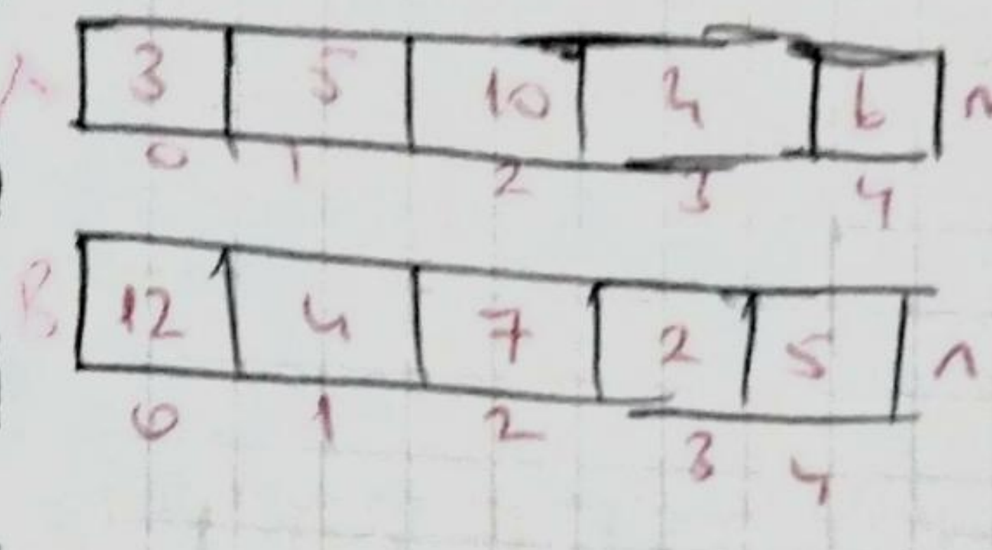
Set operation on Array

- Union , - Intersection - Difference

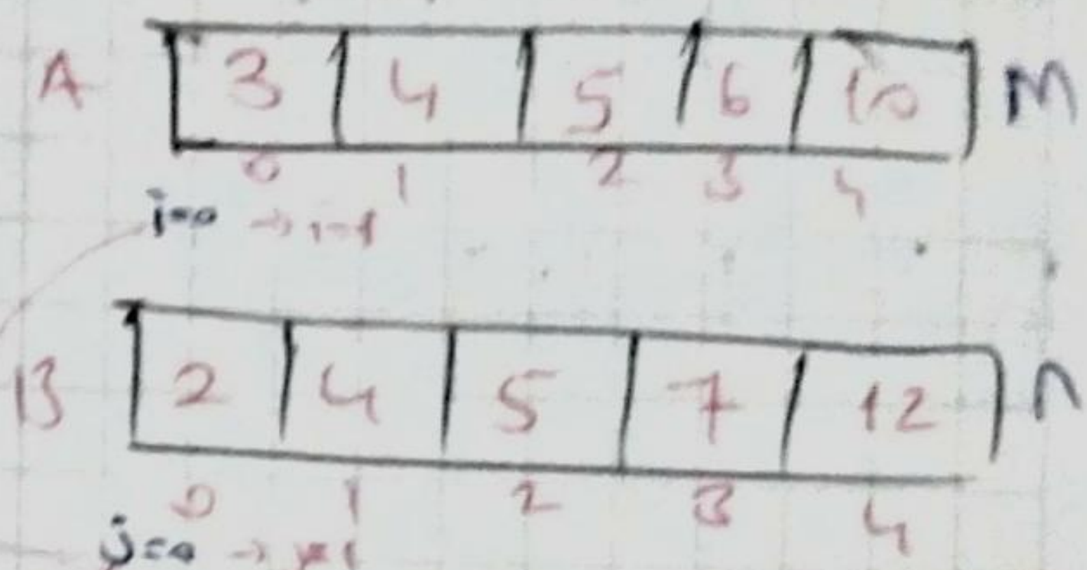
PERA

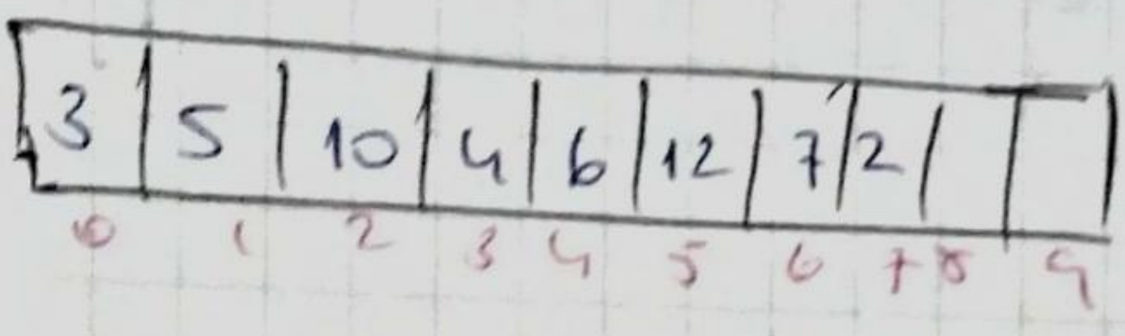
Union

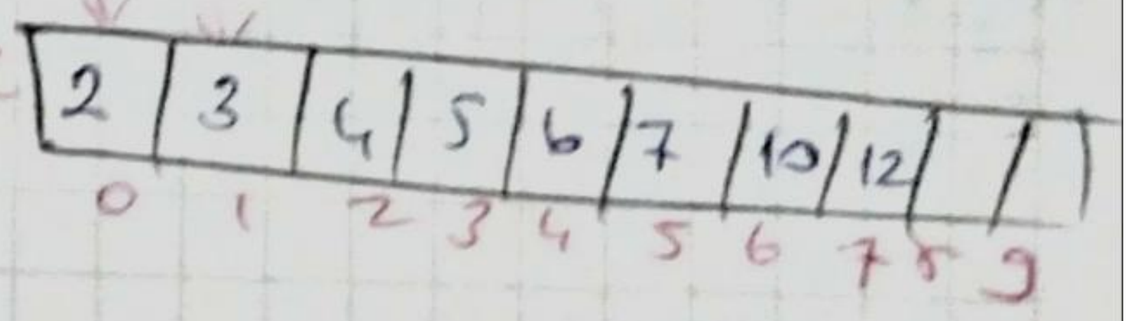
Karmaşık



Sıralı







$m + m*n$

$n + n*n$

$n + n^2 \rightarrow O(n^2)$

Burada ilk önce a->c'ye kopyalanır ondan da b tek tek kontrol edilerek c'ye aktarılır

$O(m+n)$

$O(n+n)$

$O(2n)$

$O(n)$

Burada A ve B ilk indexten başlanarak kendileri ni karşılaştırılır. Küçük olan c'ye aktarılır ve index değeri 1 artırılarak yeni bir karşılaştırma yapılır.

Scanned by TapScanner

PERA

2- Intersection

Sıralı

	0	1	2	3	4	
A	3	5	10	4	6	m

	0	1	2	3	4	
B	12	4	7	2	5	n

	0	1	2	3	4	5	6	7	8	9
C	5	4								

$n * m$

$n * n = O(n^2)$

dizideki her bir eleman B dizisindeki her elemanlar karşılaştırılır.

Karmaşık

	0	1	2	3	4	
A	3	4	5	6	10	m

i

	0	1	2	3	4	
B	2	4	5	7	12	n

j

	0	1	2	3	4	5	6	7	8	9
C	4	5								

k

$O(m+n)$

$O(n)$

Buradaki işlem önkadaki ile aynı sayılır. İndexler bir birl ile kontrol ettirilir eğer aynı indexten küçük olan hangi ise index değeri 1 artırılır. eğer indexler değerler eşit ise c'ye aktarılır k indexi 1 artırılır

Difference

A − B

karışık

	0	1	2	3	4	
A	3	5	10	4	6	m

	0	1	2	3	4	
B	12	4	7	2	5	n

	0	1	2	3	4	5	6	7	8	9
C	3	10	6							

$m \times n$

$n \times n = n^2$

$O(n^2)$

her eleman diğer dizideki tüm elemanlar ile karşılaştırılır.

sıralı

	0	1	2	3	4	
A	3	4	5	6	10	m

	0	1	2	3	4	
B	2	4	5	7	12	n

	0	1	2	3	4	5	6	7	8	9
C	3	6	10							

$O(m+n)$

$O(n+n)$

$O(n)$

burada her iki dizi ilk index ten bir birleri ile karşılaştırılır. küçük olanın indexi bir artırılır. Not sadece A ya adtarlır.

Scanned by TapScanner