Algoritma Searching

Tenia wahyuningrum, S.Kom. MT
dan
Sisilia Thya Safitri, MT


search
search
search
search
search
search
search
search
search
search
search
search

mengapa ?

mengapa ?

mengapa ?

mengapa ?

mengapa ?

mengapa ?

mengapa ?

MENGAPA ?

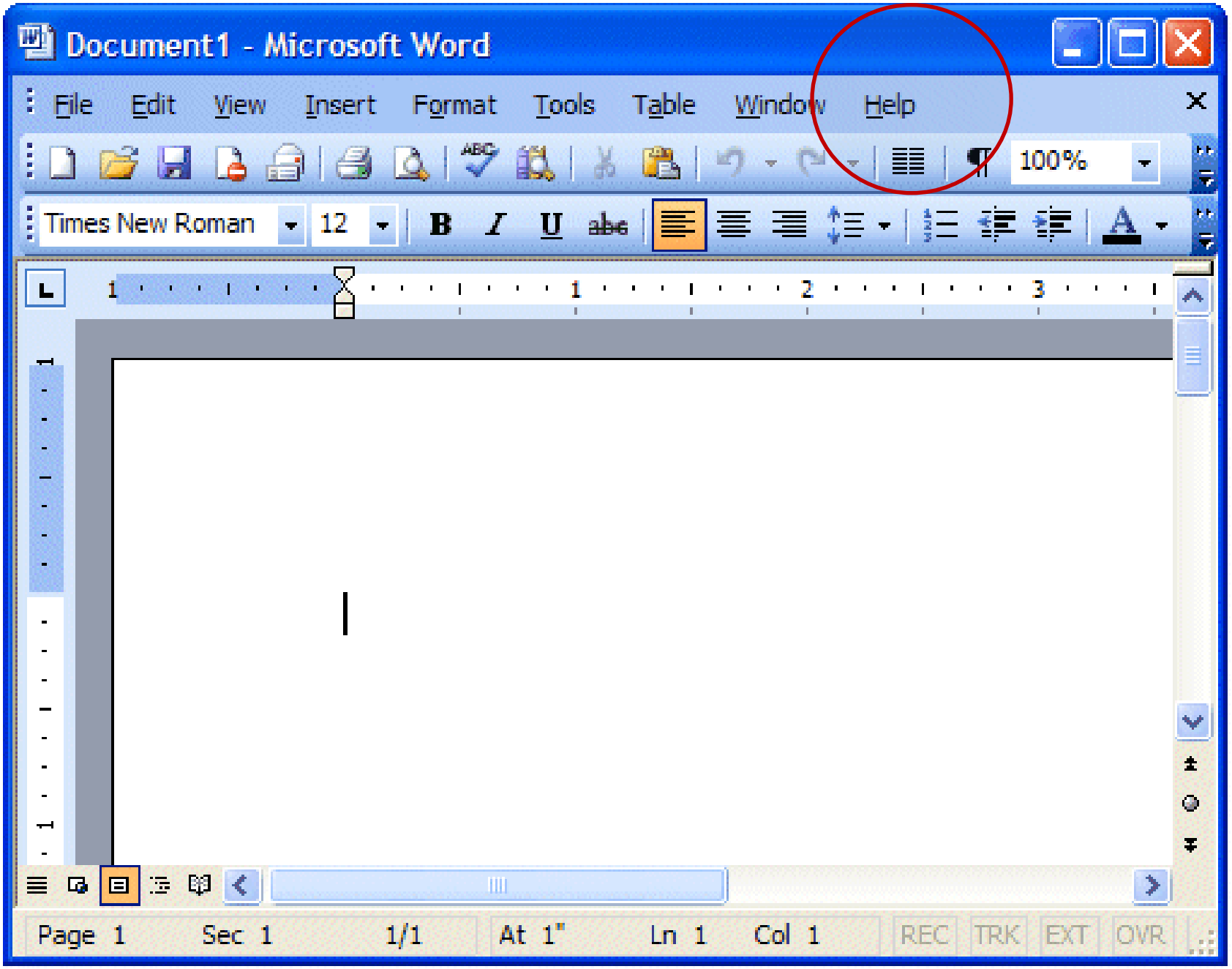

MENU
MUTE
SLEEP
SWAP
PROG
WOOP
AUDIO
POWER
MENU
SLEEP
SYSTEM
MUTE

Mengapa tombol power ada **di atas**?

Mengapa diberi **warna lain**?

untuk **memudahkan**

pencarian


Document1 - Microsoft Word
File Edit View Insert Format Tools Table Window Help
100%
Times New Roman
12
B I U abc
Page 1 Sec 1 1/1 At 1" Ln 1 Col 1
REC TRK EXT OVR

Mengapa menu help paling **kanan**?

untuk
memudahkan
pencarian


Bagaimana cara anda mencari buku tertentu dari sekumpulan buku?

menemukan **nilai (data)** tertentu didalam sekumpulan data yang bertipe sama.

data dapat disimpan secara **temporer dalam memori utama** atau disimpan secara **permanen dalam memori sekunder**.

dalam memori utama data disimpan dalam bentuk array(larik) sedangkan dalam memori sekunder dalam bentuk file(arsip).

Pencarian elemen dalam larik disebut juga pencarian internal, sedangkan pencarian data yang disimpan dalam **memori sekunder disebut juga pencarian eksternal**.

Linier

Binnary

Linier search

"Pencarian dilakukan secara teratur (secara sekuensial) dari awal sampai akhir data (atau bisa juga dari akhir ke awal data)"

Ada 2 macam kemungkinan

" data yang dicari ditemukan (successful) atau tidak ditemukan (unsuccessful)"

int array[5]

id	0	1	2	3	4
array[id]	**5**	**6**	**4**	**2**	**9**

Bagaimana cara mencari angka 4 dalam array?

Algoritma

```
1.  begin
2.  index ← 0
3.  while index <= n
4.     if key(cari) = key(index)
5.        result ← index
6.        goto 11
7.     end if
8.     index ← index + 1
9.  end while
10. hasil ← not found
11. end
```

search =4

```
for (c = 0; c < n; c++)
    {
        if (array[c] == search)
        {
            print<<search<<" is present at
            location "<< c+1;
            break;
        }
    }
```

```
    public static void main(String[] args) {
        // TODO code application logic here
        int[]data = {5,6,4,2,9};
        int search = 14;
        int c = 0;
        boolean ditemukan = false;
        for(c=0; c<data.length; c++){
            if(data[c] == search){
                ditemukan = true;
                break;
            }
        }
        if (ditemukan){
            System.out.println("Data "+search+" ditemukan pada index "+c);
        }
        else
        {
            System.out.println("Data tidak ditemukan");
        }
    }

}
```

Output - kuliahSearching (run-single)

```
init:
deps-jar:
Compiling 1 source file to D:\Dosen\STT\Kuliah\ALPRO\Mandiri\JavaProgram\kuliahSearching\build\classes
compile-single:
run-single:
Data 4 ditemukan pada index 2
BUILD SUCCESSFUL (total time: 0 seconds)
```

Contoh

- Misalnya terdapat array satu dimensi sebagai berikut:

0	1	2	3	4	5	6	7	indeks
8	10	6	-2	11	7	1	100	value

- Kemudian program akan meminta data yang akan dicari, misalnya **6**.
- Iterasi :

  6 = 8 (tidak!)

  6 = 10 (tidak!)

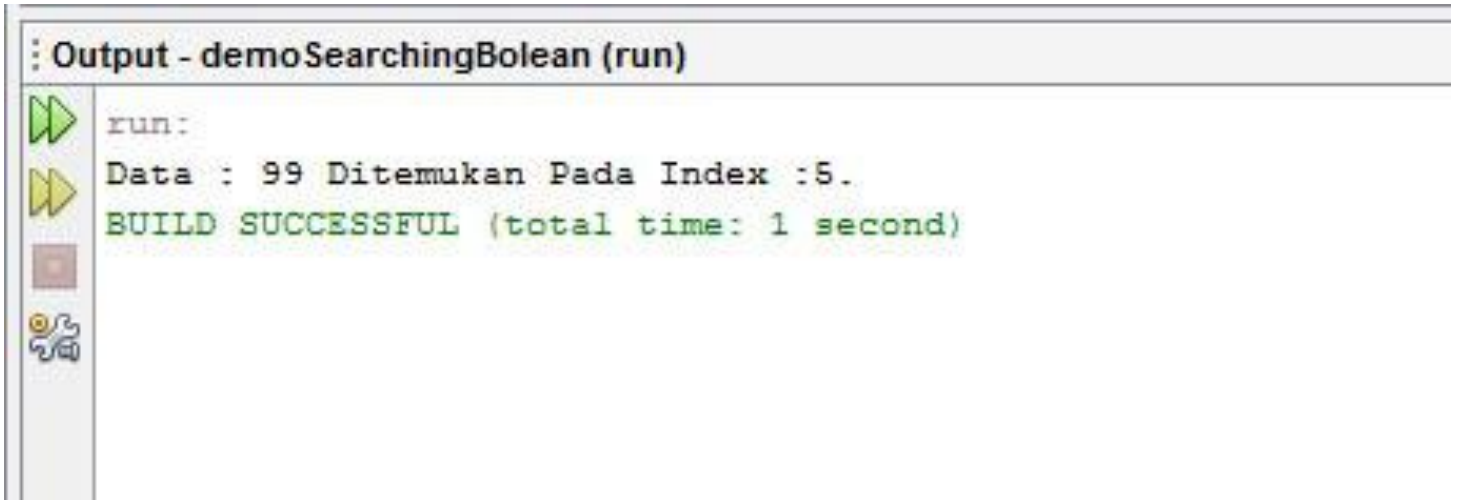
  6 = 6 (Ya!) => output : 2 (index)

Contoh Program Pencarian Data

- Data Array : 10, 8, 11, 20, 27, 99, 21, 5, 41, 17
  elemen yang di cari : 99
  Maka eleman yang di periksa adalah : 10, 8, 11, 20, 27, 99 (Data 99 di temukan)
  Index larik yang di kembalikan : idx : 5
  Pencarian akan berhenti di sini tanpa memeriksa elemen setelah elemen 99.
  Program dapat di lihat sebagai berikut :

Start Page | Main.java | Main.java

```
5
6   package demosearchingbolean;
7   /**
8    *
9    * @author ryan
10   */
11  public class Main {
12      /**
13       * @param args the command line arguments
14       */
15      public static void main(String[] args) {
16          // TODO code application logic here
17          int[]data = {10, 8, 11, 20, 27, 99, 21, 5, 41, 17};//Data Array
18          int cari=99;//Data yang di cari
19          int i =0;
20          boolean ditemukan = false; //Kondisi awal sebelum data di temukan
21
22          for(i=0; i < data.length; i++){
23              if (data[i] == cari){
24                  ditemukan=true;
25                  break;
26              }
27          }
28          if (ditemukan){
29              System.out.println("Data : " + cari + " Ditemukan Pada Index :"+ i +".");
30          }
31          else {
32              System.out.println("Data Tidak di temukan");
33          }
34      }
35
36  }
```


Output - demoSearchingBolean (run)
run:
Data : 99 Ditemukan Pada Index :5.
BUILD SUCCESSFUL (total time: 1 second)

Q & A

- **Problem:** Apakah cara di atas efisien? Jika datanya ada 10000 dan semua data dipastikan unik?
- **Solution**: Untuk meningkatkan efisiensi, seharusnya jika data yang dicari sudah ditemukan maka perulangan harus dihentikan!
  - **Hint**: Gunakan **break**!
- **Question**: Bagaimana cara menghitung ada berapa data dalam array yang tidak unik, yang nilainya sama dengan data yang dicari oleh user?
  - **Hint**: Gunakan variabel counter yang nilainya akan selalu bertambah jika ada data yang ditemukan!

Binnary search

"Sebuah pencarian biner mencari nilai tengah (median), kemudian dibandingkan apakah nilai yang dicari ada sebelum atau sesudahnya, kemudian mencari setengah sisanya dengan cara yang sama"

syarat

data sudah dalam keadaan **terurut**

ilustrasi

Contoh Data:

Misalnya data yang dicari **17**

0	1	2	3	4	5	6	7	8
3	**9**	**11**	**12**	**15**	**17**	**23**	**31**	**35**
A				**B**				**C**

- Karena 17 > 15 (data tengah), maka: awal = tengah + 1

0	1	2	3	4	5	6	7	8
3	**9**	**11**	**12**	**15**	**17**	**23**	**31**	**35**
					A **B**		**C**	

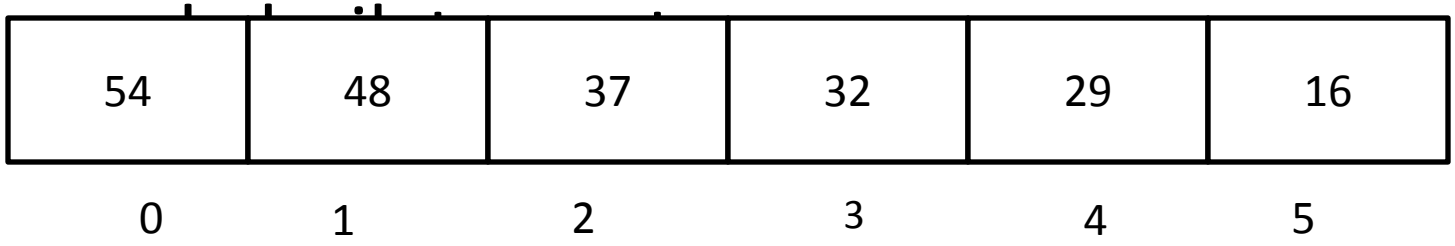
- Karena 17 < 23 (data tengah), maka: akhir = tengah – 1

0	1	2	3	4	5	6	7	8
3	**9**	**11**	**12**	**15**	**17**	**23**	**31**	**35**
					A=B=C			

- Karena 17 = 17 (data tengah), maka KETEMU!

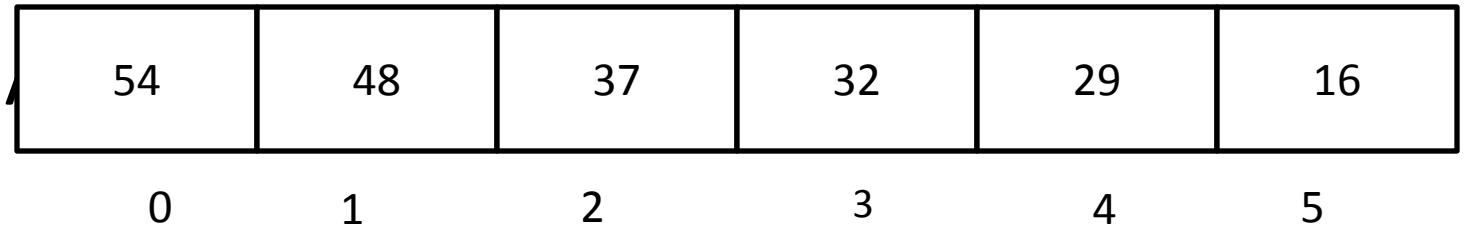
PENCARIAN BAGI DUA

- Pada pencarian indeks dibutuhkan dua buah variabel untuk menyimpan indeks terbesar dan terkecil
- Inti dari pencarian ini adalah membagi urutan data kedalam dua bagian pencarian
- Pada contoh selanjutnya akan dibahas pencarian menggunakan algoritma bagi dua



Pencarian bagi dua

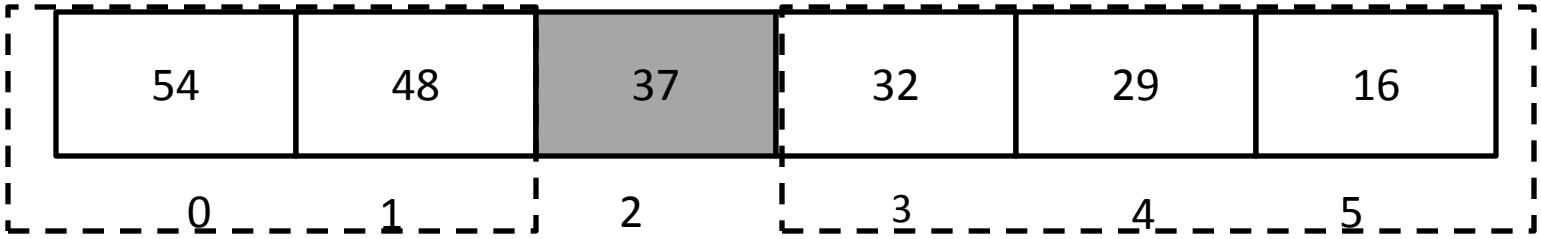
Pencarian ini hanya bisa dilakukan pada data yang sudah diurutkan (kecil → besar, atau besar → kecil)



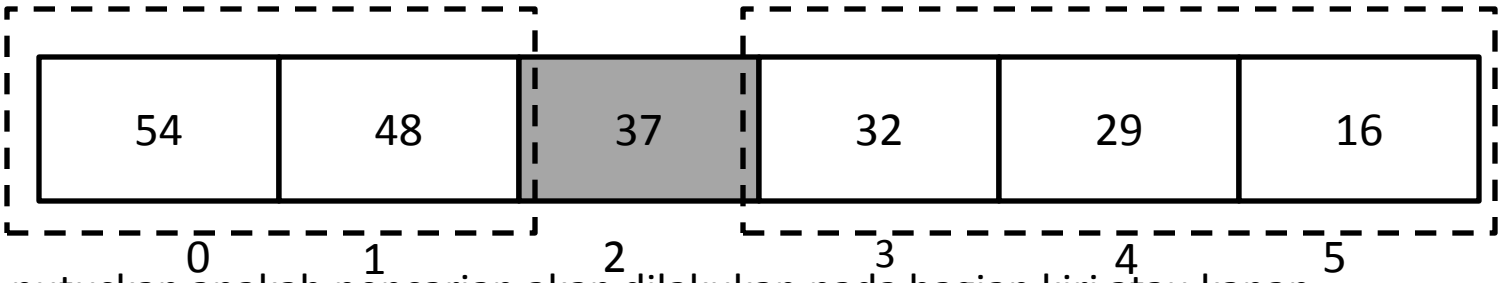
Misal di cari nilai x = 16

Langkah 1 : i = 0 dan j = 5

Di cari indeks tengah (t) = `(i+j) div 2` = (0 + 5) div 2 = **2**



Bila L[2] ≠ 16



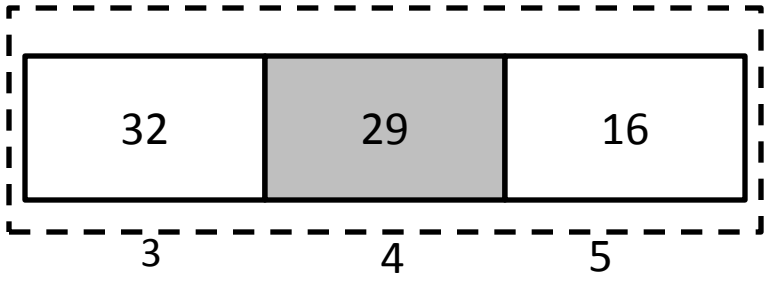
Di putuskan apakah pencarian akan dilakukan pada bagian kiri atau kanan

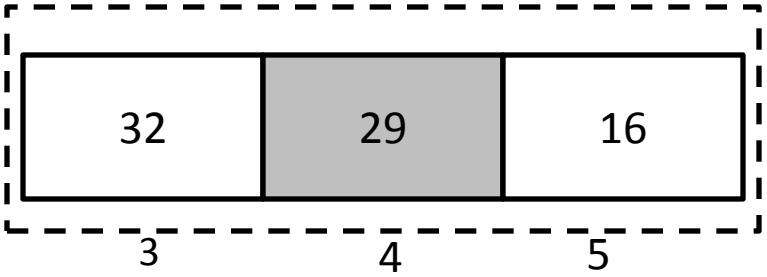
Jika nilai pada indeks saat ini lebih besar dari yang dicari dilakukan pencarian pada bagian kanan, sebaliknya pada bagian kiri.

Jika L[2] > 16 ? → true: 37 > 16

Maka pencarian akan dilakukan pada bagian kanan:

Sehingga : i = t + 1 = 3 dan j = 5 dan nilai t= (3 + 5) div 2 = 4





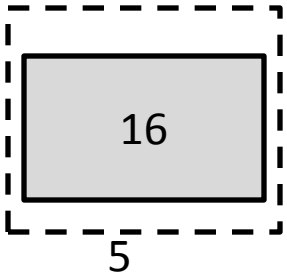
Apakah L[4] == 16 ? → false

Di putuskan apakah pencarian akan dilakukan pada bagian kiri atau kanan

Jika nilai pada indeks saat ini lebih besar dari yang dicari dilakukan pencarian pada bagian kanan, sebaliknya pada bagian kiri.

L[4] > 16 = 29 > 16? → true

Maka pencarian dilakukan pada bagian kanan :



Apakah L[5] == 16 ?? TRUE.. Nilai ditemukan

Algoritma Binary Search

1. Data diambil dari posisi 1 sampai posisi akhir N
2. Kemudian cari posisi data tengah dengan rumus: **(posisi awal + posisi akhir) / 2**
3. Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?
4. Jika data terurut secara DESCENDING (Besar ke Kecil), Jika data di indeks lebih besar MAKA nilai awal adalah **posisi tengah + 1** **atau** Jika data di indeks lebih kecil ,MAKA nilai akhir adalah **posisi tengah – 1**
5. Jika data terurut secara ASCENDING (Kecil ke Besar), Jika data di indeks lebih besar MAKA nilai akhir adalah **posisi tengah - 1** **atau** Jika data di indeks lebih kecil ,MAKA nilai awal adalah **posisi tengah + 1**
6. Jika data sama, berarti ketemu.

77;23;69;12;79;82;6;301;46;92;7

- Urutkan secara ASC, kemudian dengan binary search carilah 7

- Urutkan secara DSC, kemudian dengan binary search carilah 92

tenia@st3telkom.ac.id
http://www.slideshare.net/kuliahtenia