ALGORITMA PENGURUTAN DAN PENCARIAN

siti.maesyaroh@uniku.ac.id

Pengurutan

- Dalam menyimpan atau menampilkan data perlu dengan urutan tertentu yaitu naik/ascending atau turun/descending

  Contoh : perangkingan
- pengurutan dapat dilakukan secara internal atau eksternal
- Data yang urut juga dapat berpengaruh pada kinerja suatu algoritma : bisa menjadi cepat atau menjadi benar

siti.maesyaroh@uniku.ac.id

Pengurutan

- Pengurutan internal

  Dilakukan terhadap sekumpulan data yang disimpan dalam media internal computer yang dapat diakses secara langsung pada setiap elemennya.

- Pengurutan eksternal

  Dilakukan terhadap data yang disimpan dalam memori sekunder dan digunakan pada data yang cukup besar

Swap

- Misalkan terdapat dua data yang akan ditukar yaitu :

```
data[1] = 2;
data[2] = 4;
```

- Cara yang salah :

```
data[1] = data[2];
data[2] = data[1];
```

- Cara yang benar :

```
Swap = data[1];
data[1] = data[2];
data[2] = swap;
```

Metode Pengurutan

- Perbandingan (comparison-based sorting)
  -> Bubble sort, exchange sort
- Prioritas (priority queue sorting)
  -> Selecton sort, heap sort (menggunakan tree)
- Penyisipan dan penjagaan terurut (insert & keep sorted)
  -> Insertion sort, tree sort
- Pembagian dan penguasaan (devide and conquer)
  -> Quick sort, merge sort
- Berkurang menurun (diminishing increment sort)
  -> shell sort (Pengembangan insertion)

siti.maesyaroh@uniku.ac.id

Bubble Sort

- Pengurutan dengan cara menarik nilai tertinggi atau terendah ke indeks paling ujung dan dilakukan sebanyak n-1 langkah, dimana n adalah ukuran larik

- Misal ada 5 elemen dalam larik bernama A adalah [10,8,3,5,4], elemen-elemen larik A tersebut akan diurutkan secara ascending (nilai terendah ke tertinggi), maka yang dilakukan yaitu :

Bubble Sort

1. Mengambil indeks 0 sebagai acuan A[0]=10, bandingkan A[0] dengan A[1], A[2], A[3], A[4]
   - jika A[0] > A[1] maka tukar nilai elemen, hasilnya A[0] menjadi 8, dan larik A menjadi [8,10,3,5,4]
   - Jika A[0] > A[2] maka tukar nilai elemen, hasilnya A[0] menjadi 3, dan larik A menjadi [3,10,8,5,4]
   - Jika A[0] > A[3] maka tukar nilai elemen, hasilnya tidak terjadi, dan larik A menjadi [3,10,8,5,4]
   - Jika A[0] > A[4] maka tukar nilai elemen, hasilnya tidak terjadi, dan larik A menjadi [3,10,8,5,4]

Bubble Sort

2. Mengambil indeks 1 sebagai acuan A[1]=10, bandingkan A[1] dengan A[2], A[3], A[4]
   - jika A[1] > A[2] maka tukar nilai elemen, hasilnya A[1] menjadi 8, dan larik A menjadi [3,8,10,5,4]
   - Jika A[1] > A[3] maka tukar nilai elemen, hasilnya A[1] menjadi 5, dan larik A menjadi [3,5,10,8,4]
   - Jika A[1] > A[4] maka tukar nilai elemen, hasilnya A[1] menjadi 4, dan larik A menjadi [3,4,10,8,5]

Bubble Sort

3. Mengambil indeks 2 sebagai acuan A[2]=10, bandingkan A[2] dengan A[3], A[4]
   - jika A[2] > A[3] maka tukar nilai elemen, hasilnya A[2] menjadi 8, dan larik A menjadi [3,4,8,10,5]
   - Jika A[2] > A[4] maka tukar nilai elemen, hasilnya A[2] menjadi 5, dan larik A menjadi [3,4,5,10,8]

Bubble Sort

4. Mengambil indeks 3 sebagai acuan A[3]=10, bandingkan A[2] dengan A[3] dengan A[4]
   - jika A[3] > A[4] maka tukar nilai elemen, hasilnya A[3] menjadi 8, dan larik A menjadi [3,4,5,8,10]

   Jadi hasil pengurutanyya adalah 3,4,5,8,10

Bubble Sort

Algoritma bubble sort

```
Deklarasi
        integer a[] ← {10,8,3,5,4}
        integer i,j, temp
Deskripsi
        for (i ← 0, i<4, i←i+1)
        for (j ← (i+1), j<5, j←j+1)
                if (A[i] > A[j])
                        temp←A[i]
                        A[i] ← A[j]
                        A[j] ← temp
                endif
        endfor
Endfor
{tampilkan hasil}
For (x←0, x<6, x←x+1)
        cetak B[x]
endfor
```

siti.maesyaroh@unikl.ac.id

Contoh

```
#include <iostream>
using namespace std;
int main () {
int A[] = {10,8,3,5,4};
int temp;
for(int j=0; j<5; j++)
    cout<<A[j]<<" ";
cout<<""<<endl;
for (int i=0; i<4; i++) {
    for(int k = (i+1); k<5; k++)
        if (A[i]>A[k]){
        temp = A[i];
        A[i] = A[k];
        A[k] = temp;
    }
}
for(int n=0; n<5; n++)
    cout<<A[n]<<"";
}
```

siti.maesyaroh@uniku.ac.id

Insertion Sort

- Pengurutan dengan cara mengambil satu elemen berurut dari posisi awal satu persatu kemudian menyisipkan elemen larik tersebut pada posisi tepat

- Misal terdapat larik B dengan elemen-elemennya {25,27,10,8,76,21}, agar dapat tersusun urut mulai dari terkecil ke besar maka langkah yang dilakukan yaitu :

Insertion Sort

Algoritma insertion sort

```
Deklarasi
    Int B[] ← {25,27,10,8,76,21}
    Int I, temp, x, y
Deskripsi
    for (i←1, i<6, i←i+1)
        temp ← B[i]
        x ← 0
        { cari posisi }
        while (temp>B[x] && x<i)
            x←x+1
        endwhile
```

Insertion Sort

```
{ sisipkan pada posisi yang tepat }
            if (temp <= B[x])
                  { geser ke kanan }
                  for (y←I, y<x, y←y+1)
                        B[y] ← B[y-1]
            endfor
            B[x]←temp
      endif
endfor
{ tampilkan hasil}
for (x←0, x<6, x←x+1)
      cetak B[x]
endfor
```

siti.maesyaroh@uniku.ac.id

Contoh

```
#include <iostream>
using namespace std;
int main () {
int B[]={25,27,10,8,76,21};
int i, temp, a,b;
for(i=2; i<6; i++) {
     temp = B[i];
     a=0;
     while(temp>B[a]&&a<1) {
          a=a+1;
     }
     if (temp <= B[a]){
          for(b=i; b>a; b--) {
               B[b]=B[b-1];
          } B[a]=temp;
     }
}
for (int c=0; c<6; c++)
     cout<<B[c]<<" ";
     cout<<endl;
}
```

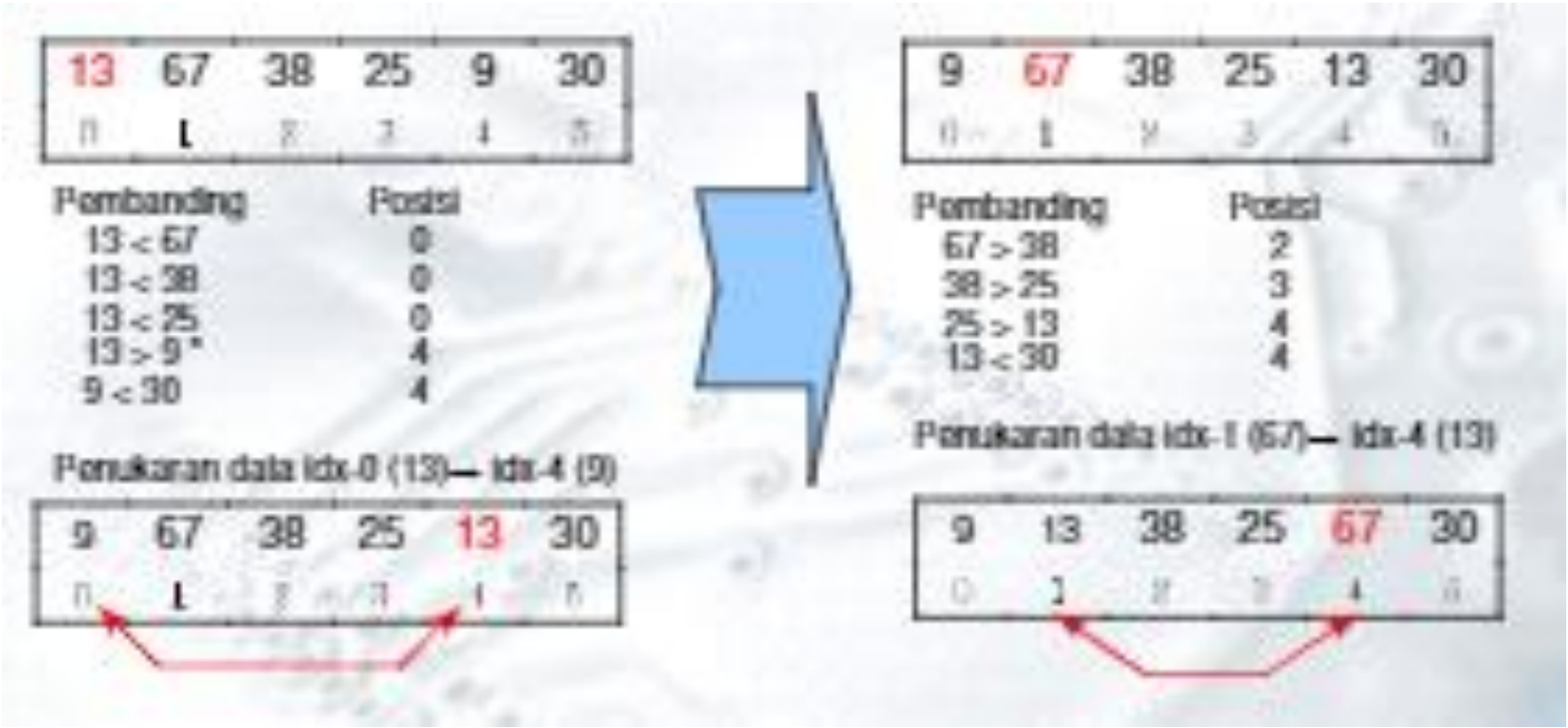
siti.maesyaroh@uniku.ac.id

Selection Sort

- Selection sort meruapakan suatu metode pengurutan yang membandingkan elemen sekarang dengan elemen berikutnya sampai ke elemen yang terakhir.
- Jika ditemukan elemen lain yang lebih kecil dari elemen sekarang maka dicatat posisinya dan langsung ditukar.

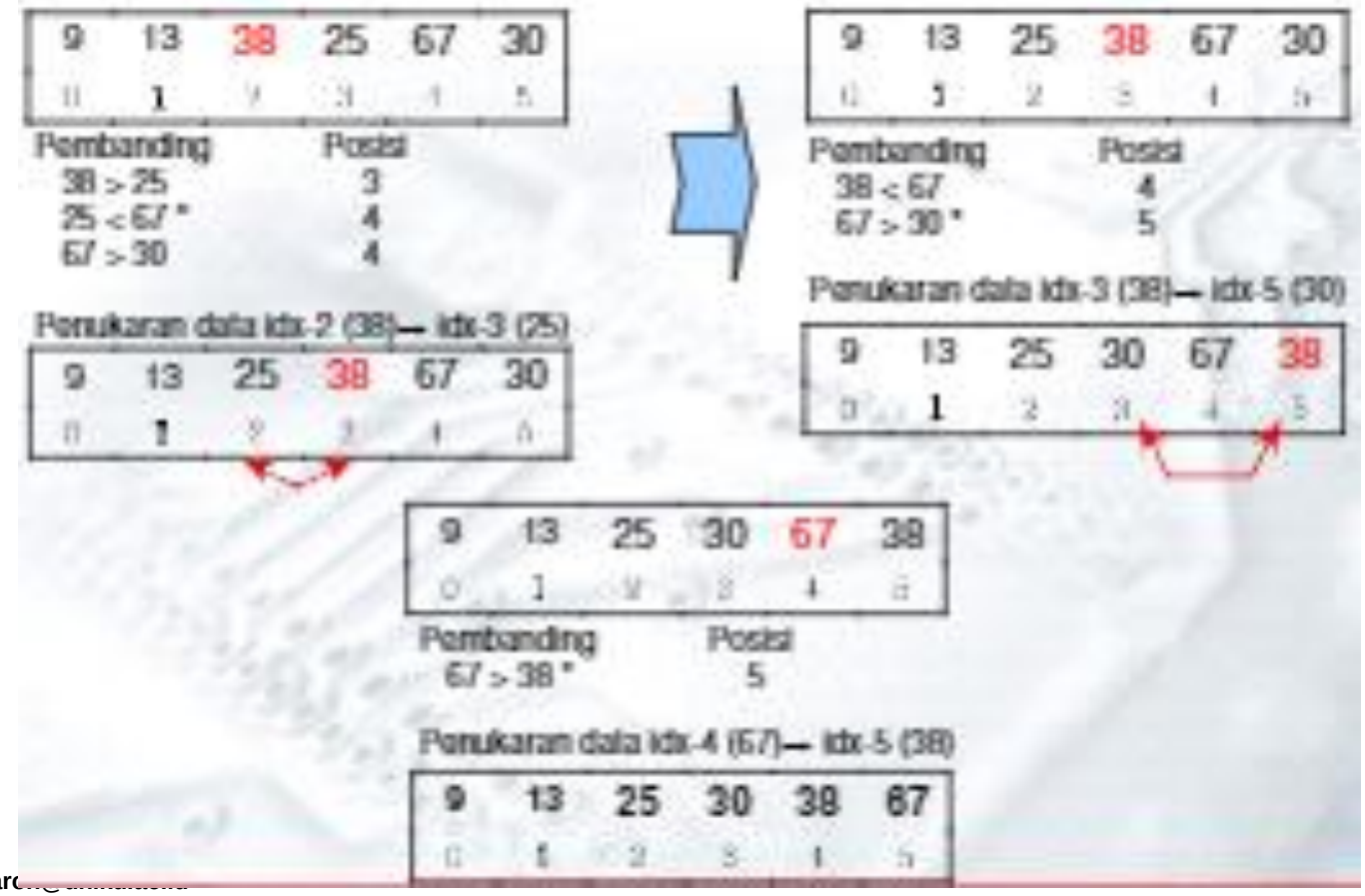
siti.maesyaroh@uniku.ac.id

Selection Sort



siti.maesyaroh@uniku.ac.id

Selection Sort



Contoh Selection Sort

```
#include <iostream>
using namespace std;
int main() {
 int x, tukar, data[10];
    cout << "Masukkan jumlah data: ";
    cin >> x;
    for (int i = 0; i < x; i++) {
        cout << "Data ke-" << i + 1 << " = " ;
        cin >> data[i];
        cout << endl;
    }
    for (int i = 0; i < x-1; i++) {
        tukar = i;
        int temp;
        for(int j = i+1; j < x; j++){
            if(data[j] < data[tukar]){
                tukar = j;
            }
        }
        temp = data[tukar];
        data[tukar] = data[i];
        data[i] = temp;
    }
    for(int i = 0; i < x; i++){
        cout << data[i] << " ";
    }
    cin.get();
    return 0;
}
```

Fungsi Sorting

- Diperlukan untuk mempercepat pencarian suatu target dalam suatu daftar (list).
- Metode pengurutan :
  - Ascending

    Pengurutan yang dilakukan mulai dari nilai terkecil menuju nilai terbesar
  - Descending

    Pengurutan yang dilakukan mulai dari nilai terbesar menuju nilai terkecil

siti.maesyaroh@uniku.ac.id

Pencarian

- Pencarian/searching → cara untuk mendapatkan data yang diinginkan dengan cara menelusuri data-data tersebut
- Proses pencarian merupakan menemukan nilai (data) tertentu di dalam sekumpulan data yang bertipe sama (tipe data dasar atau tipe data bentukan)

Metode Pencarian

1. **Linear / Sequential Searching**
   - Melakukan pengujian setiap item
   - Digunakan pada data yang masih acak dan berurut
2. **Binary Searching**
   - Membagi dua data
   - Digunakan pada data yang berurut

siti.maesyaroh@uniku.ac.id

Pencarian Pada Data Array

- Tempat pencarian data dapat berupa array dalam memori atau file pada external storage

- Terdapat dua jenis searching :

1. **Internal Searching** → Algoritma pencarian yang dilakukan dalam memori komputer
2. **External Searching** → algoritma pencarian yang melibatkan external media dan menambahkan ke memori utama

siti.maesyaroh@uniku.ac.id

Pencarian Sequential

proses pencarian dilakukan dengan membandingkan setiap elemen larik satu per satu secara beruntun, mulai dari elemen pertama sampai elemen yang dicari ditemukan atau seluruh elemen sudah diperiksa serta dapat dilakukan dalam kondisi data apapun.

siti.maesyaroh@uniku.ac.id

Pencarian Sequential

52	87	93	28	76	21
1	2	3	4	5	6

Nilai yang dicari : 28

Elemen yang diperiksa : 52 87 93 28

Index : 3

Nilai yang dicari : 52

Elemen yang diperiksa : 52

Index : 0

siti.maesyaroh@uniku.ac.id

Algoritma Pencarian Sequential

Algoritma Pencarian Sequential

```
Deklarasi
        int A [] ← {52,87,93,28,76,21}
        integer i, posisi, cari
Deskripsi
        cari ← 28
        for (i←0, i<6, i←i+1)
                if(A[i] = cari)
                        posisi ← I
                        print posisi
                        break
                endif

endfor
```

siti.maesyaroh@uniku.ac.id

Contoh

```
#include <iostream>
using namespace std;
main() {
    int A[]={52,87,93,28,76,21};
    int i, posisi, cari;
    cari = 28;
    for(i=0; i<6; i++) {
        if (A[i]==cari) {
            posisi = i;
            cout<<"Nilai berada di indeks ke "<<posisi;
            break;
        }
    }
}
```

siti.maesyaroh@uniku.ac.id

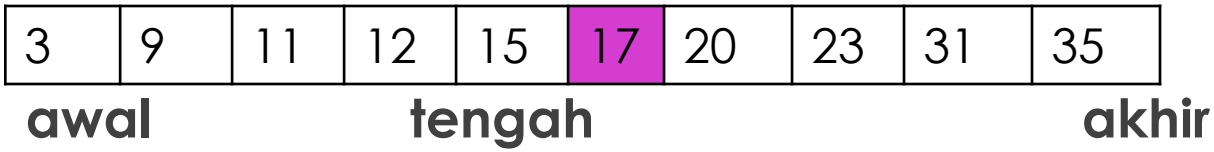
Pencarian Binary

- Memperkecil jumlah operasi pembagian yang harus dilakukan antara data yang dicari dengan data yang ada di dalam table, khususnya untuk jumlah data yang sangat besar ukurannya.
- Prinsip dasarnya yaitu melakukan proses pembagian ruang pencarian secara berulang-ulang sampai data ditemukan atau sampai ruang pencarian tidak dapat dibagi lagi (ada kemungkinan data tidak ditemukan)
- Syarat utama untuk pencarian biner yaitu data di dalam table harus sduah terurut.

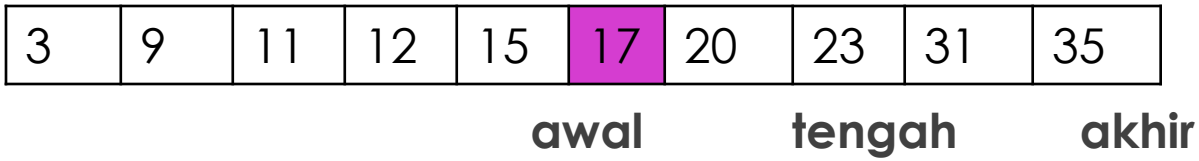
siti.maesyaroh@uniku.ac.id

Pencarian Binary

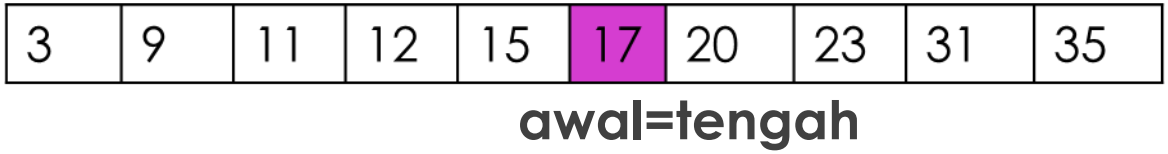
- Misalnya data yang dicari 17



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



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



- Karena 17 = 17 (data tengah, maka KETEMU

siti.maesyaroh@uniku.ac.id

Contoh

```
#include <iostream>

using namespace std;
main(){
    const int arraySize=5;
    int target;
    int array[arraySize]={1,2,3,4,5};
    int awal, tengah, akhir;

    cout<<"Masukan angka yang dicari: ";cin>>target;
    awal=0; //Initialize first and last variables.
    akhir=2;

    while(awal<=akhir)
    {
        tengah=(awal+akhir)/2;
        if(target>array[tengah])
        {
            awal=tengah+1;
        }else if(target<array[tengah])
        {
            akhir=tengah+1;
        }else{
            awal=akhir+1;
        }
    }
    if(target==array[tengah])
    {
        cout<<"Angka ditemukan."<<endl;
    }else{
        cout<<"Angka tidak ditemukan."<<endl
    }
}
```