PRAKTIKUM 7

Sorting dan Searching

Tujuan Praktikum :

1. Memperkenalkan kepada mahasiswa metoda pengurutan (sorting)
2. Memperkenalkan kepada mahasiswa metoda pencarian (searching)
3. Mempraktekkan pemakaian metoda bubblesort dan selection sort
4. Mempraktekkan pemakaian metoda sequential search dan binary search

Pengurutan (Sorting)

Pengurutan (sorting) data dalam struktur data sangat penting terutama untuk data yang beripe data numerik ataupun karakter. Pengurutan dapat dilakukan secara **ascending** (urut naik) dan **descending** (urut turun). Pengurutan (Sorting) adalah proses pengurutan data yang sebelumnya disusun secara acak sehingga tersusun secara teratur menurut aturan tertentu.

Contoh:

Data Acak	5 6 8 1 3 25 10
Ascending	1 3 5 6 8 10 25
Descending	25 10 8 6 5 3 1

Pada algoritma **sorting** umumnya menggunakan fungsi tukar 2 buah data:

```
void tukar( int a, int b) {
    int tmp;
    tmp = data[a];
    data[a] = data[b];
    data[b] = tmp;
}
```

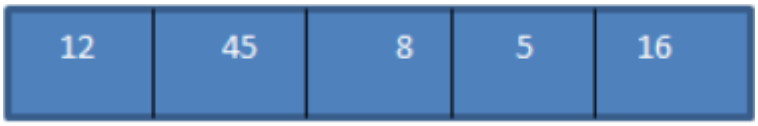
Percobaan 7.1: Bubble Sort

Merupakan metode sorting termudah, diberi nama “Bubble” karena proses pengurutan secara berangsur- angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda. Bubble Sort mengurutkan data dengan cara **membandingkan elemen sekarang dengan elemen berikutnya**. Jika elemen sekarang lebih besar dari elemen berikutnya maka kedua elemen tersebut ditukar, jika pengurutan ascending. Jika elemen sekarang lebih kecil dari elemen berikutnya, maka kedua elemen tersebut ditukar, jika pengurutan descending.

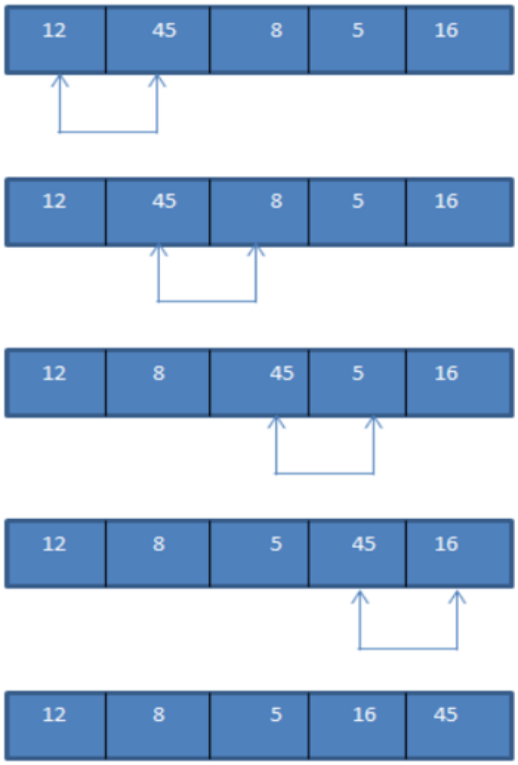
Algoritma ini seolah-olah menggeser satu per satu elemen dari kanan ke kiri atau kiri ke kanan, tergantung jenis pengurutannya. **Ketika satu proses telah selesai, maka bubble sort akan mengulangi proses, demikian seterusnya**. Kapan berhentinya? Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan.

Contoh:

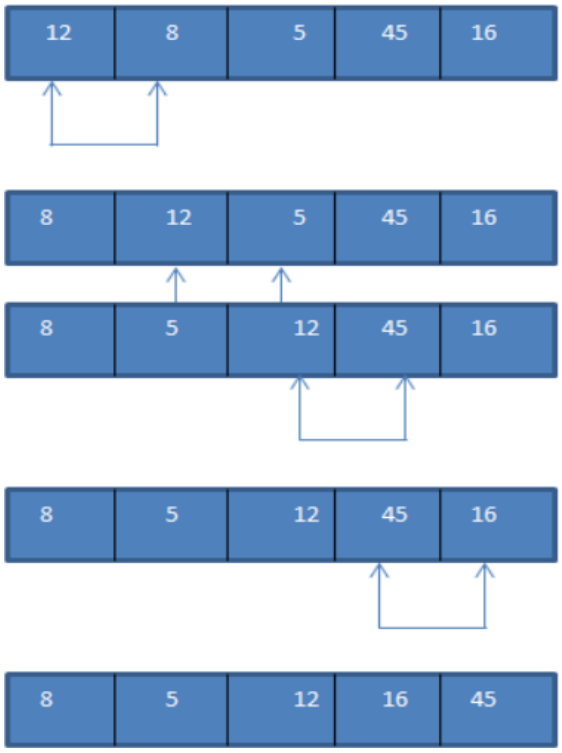
Array yang akan di urutkan



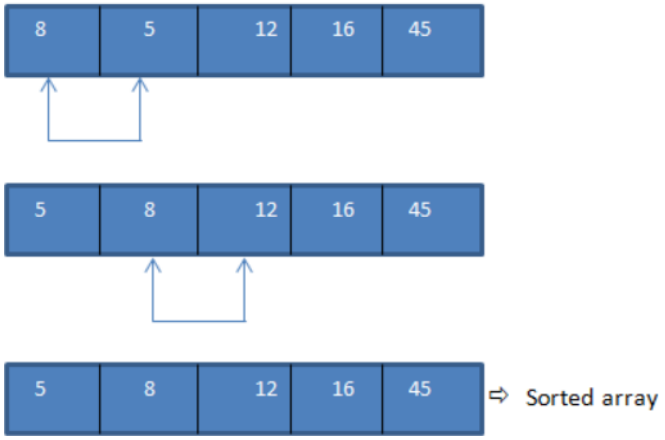
Pass 1:



Pass 2:



Pass 3:

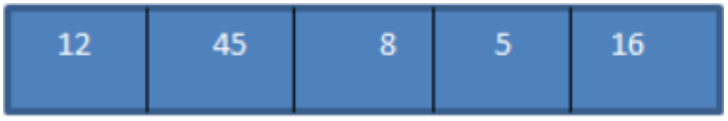


```
#include<iostream>
using namespace std;

main () {
       int i, j,temp;
       int a[5] = {10,2,0,43,12};
       cout <<"List Array :\n";
       for(i = 0; i<5; i++) {
              cout <<a[i]<<"\t";
       }

       cout<<endl;
       for(i = 0; i<5; i++) {
          for(j = i+1; j<5; j++) {
             if(a[j] < a[i]) {
                temp = a[i];
                a[i] = a[j];
                a[j] = temp;
             }
          }
       }

       cout <<"Sorted Array :\n";
       for(i = 0; i<5; i++) {
          cout <<a[i]<<"\t";
       }
}
```

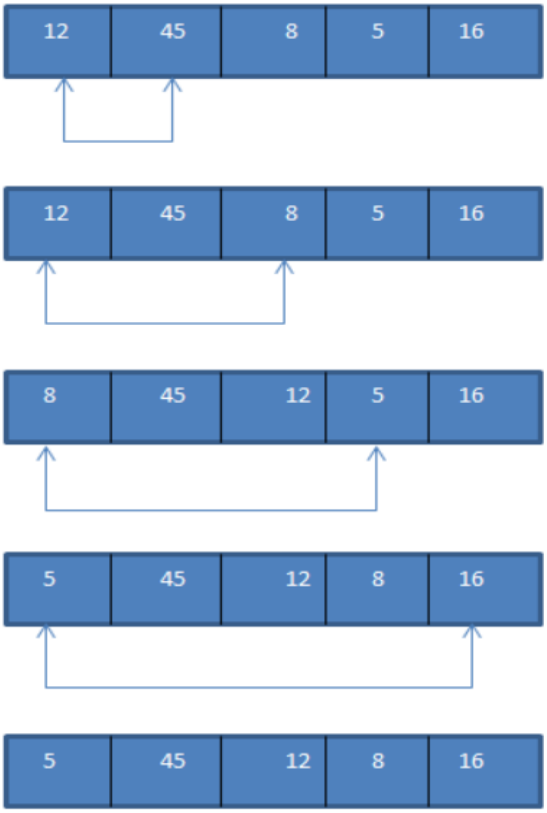
Percobaan 7.2: Selection Sort

Merupakan kombinasi antara sorting dan searching. Untuk setiap proses, akan dicari elemen-elemen yang belum diurutkan yang memiliki nilai terkecil atau terbesar akan dipertukarkan ke posisi yang tepat di dalam array. Misalnya **untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan di indeks terkecil (data[0]), pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua (data[1]).** Selama proses, pembandingan dan pengubahan hanya dilakukan pada indeks pembanding saja, pertukaran data secara fisik terjadi pada akhir proses.

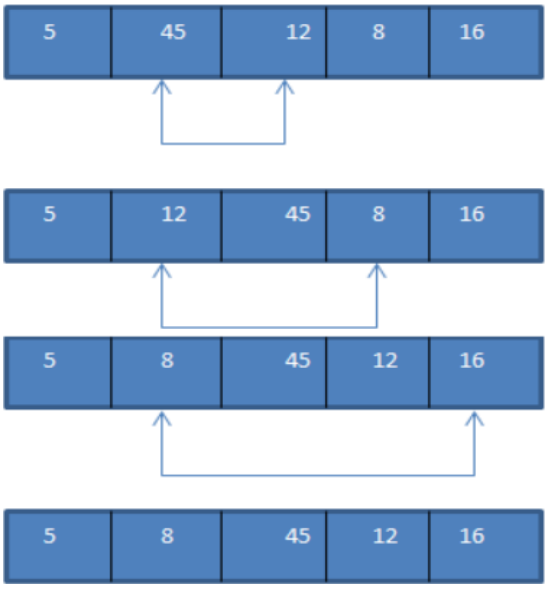
Contoh:
Array yang akan di urutkan



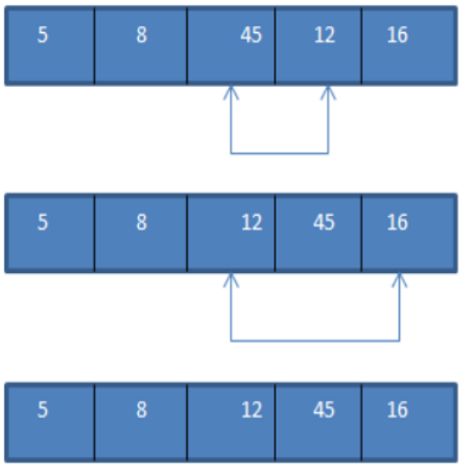
Pass 1:



Pass 2:



Pass 3:

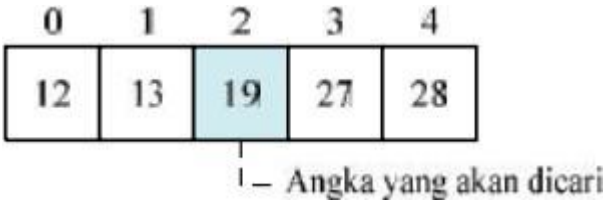


Pass 4:

```
#include<iostream>
using namespace std;

int findSmallest (int[],int);

main ()
{
   int myarray[5] = {12,45,8,15,33};
   int pos,temp;
   cout<<"\n list element array\n";
   for(int i=0;i<5;i++) {
      cout<<myarray[i]<<"\t";
   }
   for(int i=0;i<5;i++) {
      pos = findSmallest (myarray,i);
      temp = myarray[i];
      myarray[i]=myarray[pos];
      myarray[pos] = temp;
   }
   cout<<"\n Sorted array\n";
   for(int i=0;i<5;i++) {
      cout<<myarray[i]<<"\t";
   }
}

int findSmallest(int myarray[],int i)
{
   int ele_small,position,j;
   ele_small = myarray[i];
   position = i;
   for(j=i+1;j<5;j++) {
         if(myarray[j]<ele_small) {
         ele_small = myarray[j];
         position=j;
      }
   }
   return position;
}
```

Percobaan 7.3: Sequential Search

Teknik pencarian data dari array yang paling mudah adalah dengan cara sequential search, dimana data dalam **array dibaca 1 demi satu**, diurutkan dari index terkecil ke index terbesar, maupun sebaliknya.



Misalkan, dari data diatas angka yang akan dicari adalah angka 19 dalam array A, maka proses yang akan terjadi pada proses pencarian adalah sebagai berikut:

- Pencarian dimulai pada index ke-0 yaitu angka 12, kemudian dicocokan dengan angka yang akan dicari, jika tidak sama makapencarian akan dilanjutkan ke index selanjutnya.
- Pada index ke-1, yaitu angka 13, juga bukan angka yang dicari, maka pencarian juga akan dilanjutkan pada index selanjutnya
- Pada index ke-2, yaitu angka 19, ternyata angka 19 merupakan angka yang dicari. Pencarian angka telah ditemukan, maka pencarian akan dihentikan dan keluar dari looping pencarian.

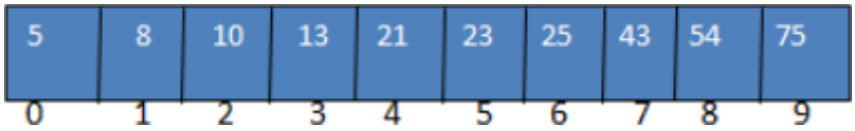
```
#include <iostream>
using namespace std;

main()
{
    int myarray[10] = {21,43,23,54,75,13,5,8,25,10};
    int key,loc;
    cout<<"List array:"<<endl;
    for(int i=0;i<10;i++){
        cout<<myarray[i]<<" ";
    }
    cout<<endl;

    cout<<"Masukkan angka yang dicari : "; cin>>key;
    for (int i = 0; i< 10; i++)
    {
        if(myarray[i] == key) {
            loc = i+1;
            break;
        }
        else {
                    loc = 0;
              }
    }
    if(loc != 0) {
        cout<<"Angka ditemukan pada posisi ke "<<loc<<" pada array";
    }
    else {
        cout<<"Angka tidak ditemukan pada array";
    }
}
```

Percobaan 7.4: Binary Search

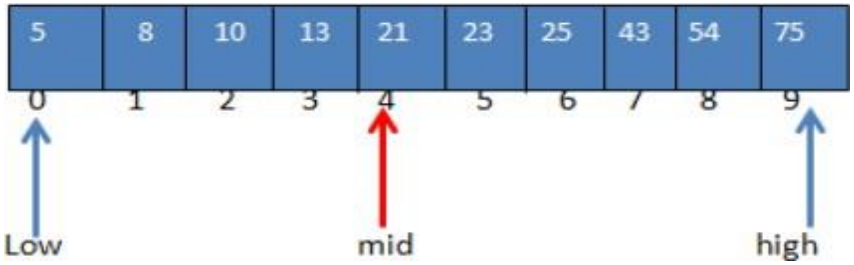
Metode pencarian yang kedua adalah binary search, pada metode pencarian ini, **data harus diurutkan terlebih dahulu**. Pada metode pencarian ini, **data dibagi menjadi dua bagian (secara logika)**, untuk setiap tahap pencarian. Algoritma binary search :

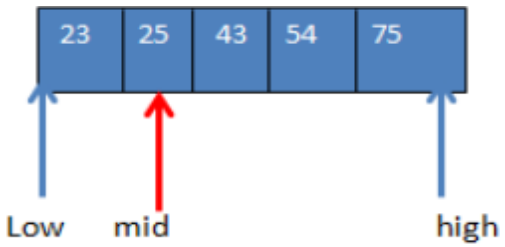
a. Data diambil dari posisi 1 sampai posisi akhir N
b. Kemudian cari posisi data tengah dengan rumus: (posisi awal + posisi akhir) / 2
c. Kemudian data yang dicari dibandingkan dengan data yang di tengah, apakah sama atau lebih kecil, atau lebih besar?
d. Jika lebih besar, maka proses pencarian dicari dengan posisi awal adalah posisi tengah+1
e. Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir adalah posisi tengah–1
f. Jika data sama, berarti ketemu.

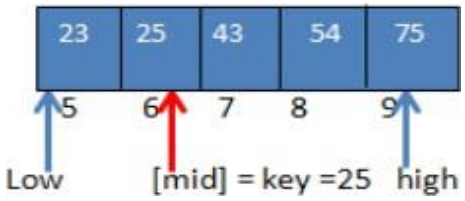
Contoh:



Data yang akan dicari adalah 25
Cari posisi tengah array:
Mid = (0 + 9) / 2 = 4







```
#include <iostream>
using namespace std;

binarySearch(int myarray[], int beg, int end, int key)
{
    int mid;
    if(end >= beg) {
        mid = (beg + end)/2;
        if(myarray[mid] == key) {
            return mid+1;
        } else if(myarray[mid] < key) {
            return binarySearch(myarray,mid+1,end,key);
        } else {
            return  binarySearch(myarray,beg,mid-1,key);
        }
    }
    return -1;
}

main ()
{
    int myarray[10] = {5,8,10,13,21,23,25,43,54,75};
    int key, location=-1;
    cout<<"List array :"<<endl;
    for(int i=0;i<10;i++ ){
        cout<<myarray[i]<<" ";
    }
    cout<<endl;
    cout<<"Masukkan angka yang akan dicari:"; cin>>key;
    location = binarySearch(myarray, 0, 9, key);
    if(location != -1)  {
        cout<<"Angka " << key << " ditemukan pada posisi "<<location;
    }
    else    {
        cout<<"Angka tidak ditemukan";
    }
}
```

Sumber/Referensi:

(1) Munir, R. (2012). Algoritma dan Pemrograman. Jilid 1 Bandung: Penerbit Informatika.

(2) Wirth, N. (1990). Algorithms + Data Structures = Programs. India: Prentice-Hall Of India Pvt. Limited.

(3) Hubbard, J.R. (1996). Programming With C++, Schaum's outlines Series. USA: McGraw Hill

(4) Suarga, M.Math., (2012). Algoritma Dan Pemrograman (Edisi 2). Yogyakarta: Penerbit Andi.

(5) Shalahuddin, M dan AS. Rosa. (2007) Belajar Bahasa Pemrograman dengan C++ dan Java: Penerbit Informatika.

(6) Joel Adams-Sanford Leestma-Larry Nyhoof, (1995). C++ An Introduction To Computing: Prentice-Hall, Inc

(7) James P. Cohoon-Jack W.Davidson, McGraw-Hill, (1997). C++ Programming Design.

(8) Budi Raharjo, (2004). Mengungkap Rahasia Pemrograman Dalam C++: Penerbit Informatika.

(9) Abdul Kadir, (2003). Pemrograman C++, Andi, Yogyakarta

(10) Yosua Onesimus Suheru, (2004). Trik Memecahkan Masalah Dengan Tiga bahasa Pemrograman – C++, Pascal dan Visual Basic, Gava Media, Yogyakarta