BAB 1

PENGURUTAN (SORTING)

1.1. TUJUAN PEMBELAJARAN

Tujuan dari pembelajaran pada pertemuan ini adalah mahasiswa diharapkan mampu:

1. Memahami dan dapat mengimplementasikan *Bubble-sort*
2. Memahami dan dapat mengimplementasikan *Insert-sort*
3. Memahami dan dapat mengimplementasikan *Selection-sort*

1.2. DASAR TEORI

Sorting merupakan sebuah konsep atau proses yang mengurutkan elemen-elemen array yang acak mulai dari yang terkecil ke terbesar atau sebaliknya. Sorting dari yang terkecil ke terbesar disebut *Ascending*, sedangkan dari yang terbesar ke terkecil disebut *Descending*. Fungsi Algoritma Penyortiran, yaitu untuk mengatur ulang array atau elemen daftar tertentu sesuai dengan operator perbandingan pada elemen. Operator perbandingan biasanya digunakan untuk memutuskan urutan elemen baru dalam struktur data masing-masing. Tujuan dari sorting adalah untuk mengatur elemen data agar lebih mudah ditemukan pada saat proses pencarian.

Contoh implementasi dari konsep sorting, diantaranya pada suatu perpustakaan setiap buku disusun berdasarkan indeks, menyortir kartu pada saat bermain kartu, pada online shop terdapat filter pencarian berdasarkan harga dan lokasi terdekat untuk mengurutkan barang yang dicari. Karena sorting sering diterapkan dikehidupan sehari – hari, maka pada bab ini akan membahas algoritma untuk menyortir satu set N item. Memahami algoritma penyortiran adalah langkah pertama menuju analisis algoritma pemahaman. Banyak algoritma pengurutan dikembangkan dan dianalisis.

Untuk mengurutkan dataset, algoritma pengurutan terbagi menjadi 2. Untuk set input kecil dapat menggunakan Bubble-Sort, insert-Sort dan Selection-Sort. Sedangkan untuk data set besar dapat menggunakan Merge-Sort, Quick-Sort dan Heap-Sort.

Sebagai contoh dari penyortiran algoritma, pahami illustrasi berikut. Daftar karakter di bawah ini diurutkan dalam meningkatkan urutan nilai ASCII mereka. Artinya, karakter dengan nilai ASCII yang lebih rendah akan ditempatkan pertama daripada karakter dengan nilai ASCII yang lebih tinggi.

KESATUAN =======> AAEKNSTU

Input output

1.2.1. Algoritma Bubble Sort

Bubble Sort adalah algoritma penyortiran paling sederhana yang bekerja dengan teknik pengurutan data dengan cara menukar dua data yang bersebelahan jika urutan dari data tersebut salah.

Contoh:

Pass Pertama:

(**5 1** 4 2 8) → (**1 5** 4 2 8), Di sini, algoritma membandingkan dua elemen pertama, dan swap sejak 5 > 1.

(1 **5 4** 2 8) → (1 **4 5** 2 8), Swap sejak 5 > 4

(1 4 **5 2** 8) → (1 4 **2 5** 8), Swap sejak 5 > 2

(1 4 2 **5 8**) → (1 4 2 **5 8**), Sekarang, karena elemen-elemen ini sudah dalam rangka (8 > 5), algoritma tidak menukarnya.

Pass Kedua:

(1 4 2 5 8) → (1 4 2 5 8)

(1 4 2 5 8) → (1 2 4 5 8), Swap sejak 4 > 2

(1 2 4 5 8) → (1 2 4 5 8)

(1 2 4 5 8) → (1 2 4 5 8)

Setelah, array diurutkan, algoritma membutuhkan satu parameter dimana seluruh elemen lulus tanpa swap untuk mengetahui itu diurutkan untuk mengetahui apakah pegurutan sudah selesai dilakukan.

Pass Ketiga**:**

(**1 2** 4 5 8) → (**1 2** 4 5 8)

(1 **2 4** 5 8) → (1 **2 4** 5 8)

(1 2 **4 5** 8) → (1 2 **4 5** 8)

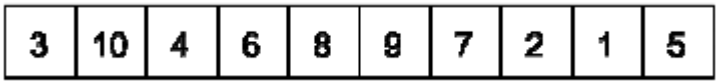
(1 2 4 **5 8**) → (1 2 4 **5 8**)

1.2.2. Algoritma Insertion Sort

Insertion sort merupakan salah satu algoritma sorting yang paling sederhana selain *bubble sort*. Pada dasarnya algoritma *insertion sort* dimulai dari memilah data yang akan urutkan menjadi 2 bagian (yang belum diurutkan dan yang sudah diurutkan). Elemen pertama diambil dari bagian array yang belum diurutkan dan kemudian diletakkan sesuai posisinya pada bagian lain dari array yang telah diurutkan. Langkah ini dilakukan secara berulang hingga tak ada lagi elemen tersisa pada bagian array yang belum diurutkan. Hal ini berkaitan dengan prinsip kerja atau metode *insertion sort* dimana terdapat sebuah metode yang mengurutkan bilangan-bilangan yang telah terbaca, dan berikutnya secara berulang akan menyisipkan bilangan-bilangan dalam array yang belum terbaca ke sisi kiri array yang telah terurut.

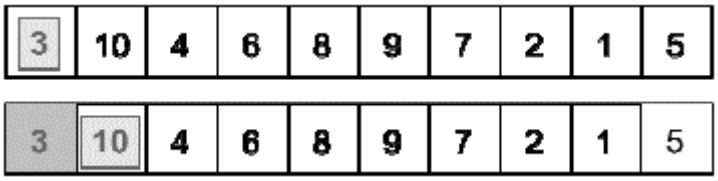
Berikut adalah illustrasi dari *insertion sort* yang digunakan untuk mengurutkan bilangan 1-10 yang diacak secara *ascending.*

1. Langkah pertama ambil bilangan yang paling kiri. Bilangan tersebut dikatakan urut terhadap dirinya sendiri karena bilangan yang di bandingkan baru 1.



Gambar 1.1 Langkah 1 *insertion sort*

2. Selanjutnya, perika bilangan ke 2 (10) apakah lebih kecil dari bilangan yang ke 1(3). Apabila lebih kecil maka ditukar. Tapi kali ini bilangan ke 1 lebih kecil dari bilangan ke 2, maka tidak ditukar.

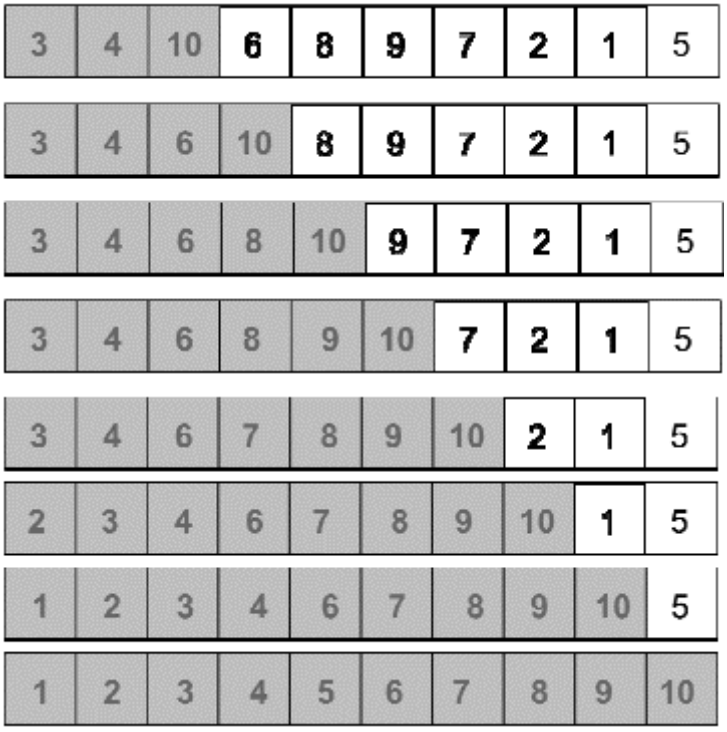


Gambar 1.2 Langkah 2 *insertion sort*

3. Pada kotak warna abu - abu sudah dalam keadaan terurut. Kemudian membandingkan lagi pada bilangan selanjutnya yaitu bilangan ke 3 (4). Bandingkan dengan bilangan yang ada di sebelah kirinya. Pada kasus ini bilangan ke 2 bergeser dan digantikan bilangan ke-3.

Gambar 1.3 Langkah 3 *insertion sort*

4. Lakukan langkah seperti di atas pada bilangan selanjutnya. 4 bilangan pertama sudah dalam keadaan terurut relatif. Ulangi proses tersebut sampai bilangan terakhir disisipkan.



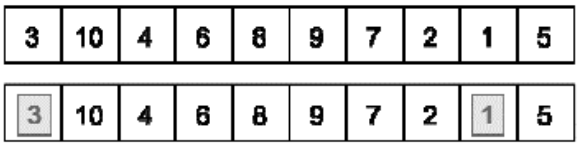
Gambar 1.4 Langkah 4 *insertion sort*

1.2.3. Algoritma Selection Sort

Selection sort merupakan sebuah teknik pengurutan dengan cara mencari nilai tertinggi / terendah di dalam array kemudian menempatkan nilai tersebut di tempat semestinya. Algoritma ini memiliki kompleksitas waktu yang sama dengan algoritma *bubble short*, namun metode ini dianggap lebih baik karena dalam *selection sort* array diurutkan dibuat mundur. Sederhananya, dapat dikatakan jika algoritma selection sort adalah teknik memilih elemen dengan nilai paling rendah dan menukar elemen yang terpilih dengan elemen ke-i. Nilai dari i dimulai dari 1 ke n, dimana n adalah jumlah total elemen dikurangi 1.

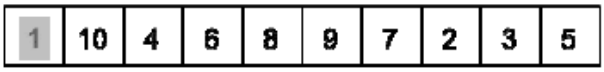
Berikut adalah illustrasi dari *selection sort* yang digunakan untuk mengurutkan bilangan 1-10 yang diacak secara *ascending.*

1. Langkah pertama, periksalah seluruh array dan cari array yang mempunyai nilai terkecil → index 8 (1). Setelah ketemu tukar dengan array yang berada di pojok kiri (3).



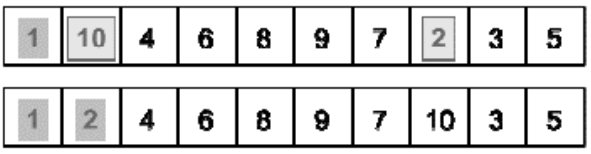
Gambar 1.5 Langkah 1 *selection sort*

2. Selanjutnya, setelah di tukar terdapat bagian yang berwarna abu-abu yang merupakan index yang telah terurutkan.



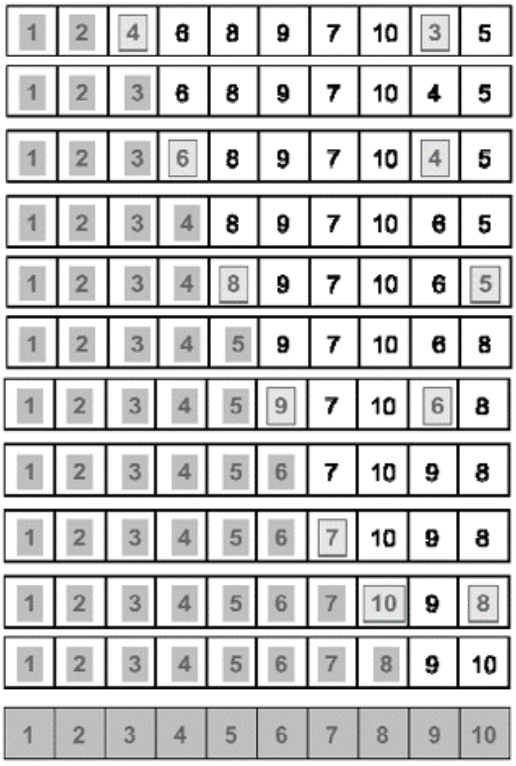
Gambar 1.6 Langkah 2 *selection sort*

3. Lalu, cari bilangan terkecil selanjutnya (selain di kotak abu-abu) yaitu bilangan 2 dan tukar dengan sebelah array yang telah terurutkan.



Gambar 1.7 Langkah 3 *selection sort*

4. Terakhir, Dua array sudah terurutkan. Kemudian ulangi langkah di atas dan lakukan langkah yang sama yaitu pilih terkecil dan tukar dengan sebelah array yang sudah terurutkan.



Gambar 1.8 Langkah 4 *selection sort*

1.3. PERCOBAAN

1. Buatlah workspace menggunakan Replit.
2. Buatlah project baru SORTING yang berisi file C++ source untuk metode *bubble sort*, *insertion sort* dan *selection sort*.
3. Cobalah untuk masing-masing percobaan di bawah.

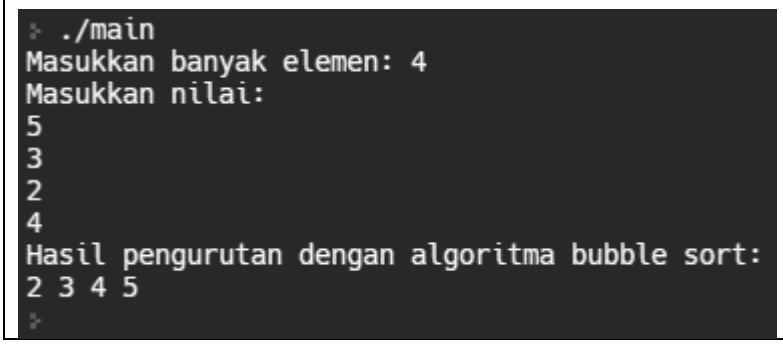
Percobaan 1 : Implementasi *bubble sort* secara *ascending*

```
#include <iostream>

using namespace std;

void bubbleSort(int arr[], int n){
  int i, j, tmp;
  for (i = 0; i < n; i++){
    for (j = 0; j < n - i - 1; j++){
      if (arr[j] > arr[j + 1]){
        tmp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = tmp;
      }
    }
  }
}

int main(){
  int array[100], n, i, j;
  cout << "Masukkan banyak elemen: ";
  cin >> n;
  cout << "Masukkan nilai: \n";
  for (i = 0; i < n; i++){
    cin >> array[i];
  }
  bubbleSort(array, n);
```

```
  cout << "Hasil pengurutan dengan algoritma bubble
sort:\n";
  for (i = 0; i < n; i++){
    cout << array[i] << " ";
  }
  cout << "\n";
}
```

OUTPUT

Masukkan banyak elemen: 4
Masukkan nilai:
5
3
2
4
Hasil pengurutan dengan algoritma bubble sort:
2 3 4 5



Percobaan 2 : Implementasi *insertion sort* secara *ascending*

```
#include <iostream>

using namespace std;



int main(){

     int y;

     cout<<"masukan banyak array:";

     cin>>y;
```

```
int x[y];
for(int i=0; i<y; i++){
     cout<<"masukan angka ke "<<i<<" :";
     cin>>x[i];
     cout<<endl;
}
for(int i=1; i<y; i++){
     int key = x[i];
     int j = i-1;
     while(j>=0 && x[j] > key){
          x[j+1] = x[j];
          j--;
     }
     x[j+1] = key;
     cout<<"proses sorting"<<endl;
     for(int m=0;m<y;m++){
     cout<<x[m]<<" ";
    }
    cout<<endl;
}
cout<<"hasil akhir"<<endl;
```

```
      for(int m=0;m<y;m++){

            cout<<x[m]<<" ";

      }

}
```

OUTPUT

masukan banyak array:6
masukan angka ke 0 :1

masukan angka ke 1 :4

masukan angka ke 2 :6

masukan angka ke 3 :2

masukan angka ke 4 :5

masukan angka ke 5 :3

proses sorting
1 4 6 2 5 3
proses sorting
1 4 6 2 5 3
proses sorting
1 2 4 6 5 3
proses sorting
1 2 4 5 6 3
proses sorting
1 2 3 4 5 6
hasil akhir
1 2 3 4 5 6

```
masukan banyak array:6
masukan angka ke 0 :1

masukan angka ke 1 :4

masukan angka ke 2 :6

masukan angka ke 3 :2

masukan angka ke 4 :5

masukan angka ke 5 :3

proses sorting
1 4 6 2 5 3
proses sorting
1 4 6 2 5 3
proses sorting
1 2 4 6 5 3
proses sorting
1 2 4 5 6 3
proses sorting
1 2 3 4 5 6
hasil akhir
1 2 3 4 5 6
```

Percobaan 3 : Implementasi *selection sort* secara *ascending*

```
#include <iostream>

using namespace std;



void selectionSort(int arr[], int n) {

  int i, j, minIndex, temp;

  for (i = 0; i < n-1; i++) {

    minIndex = i;

    for (j = i+1; j < n; j++) {

      if (arr[j] < arr[minIndex]) {

        minIndex = j;
```

```
      }
    }
    temp = arr[minIndex];
    arr[minIndex] = arr[i];
    arr[i] = temp;
    cout << "Iterasi ke-" << i+1 << ": ";
    for (int k = 0; k < n; k++) {
      cout << arr[k] << " ";
    }
    cout << endl;
  }
}

int main() {
  int n, i;
  cout << "Masukan jumlah elemen: ";
  cin >> n;
  int arr[n];
  cout << "Masukan nilai elemen: ";
  for (i = 0; i < n; i++) {
    cin >> arr[i];
```

```
  }
  cout << "Data sebelum sorting: ";
  for (i = 0; i < n; i++) {
    cout << arr[i] << " ";
  }
  cout << endl;
  selectionSort(arr, n);
  cout << "Data setelah sorting: ";
  for (i = 0; i < n; i++) {
    cout << arr[i] << " ";
  }
  return 0;
}
```

OUTPUT

Masukan jumlah elemen: 6

Masukan nilai elemen: 3

6

5

2

4

1

Data sebelum sorting: 3 6 5 2 4 1

Iterasi ke-1: 1 6 5 2 4 3

Iterasi ke-2: 1 2 5 6 4 3

Iterasi ke-3: 1 2 3 6 4 5

Iterasi ke-4: 1 2 3 4 6 5

Iterasi ke-5: 1 2 3 4 5 6

Data setelah sorting: 1 2 3 4 5 6

```
./selectionsort
Masukan jumlah elemen: 6
Masukan nilai elemen: 3
6
5
2
4
1
Data sebelum sorting: 3 6 5 2 4 1
Iterasi ke-1: 1 6 5 2 4 3
Iterasi ke-2: 1 2 5 6 4 3
Iterasi ke-3: 1 2 3 6 4 5
Iterasi ke-4: 1 2 3 4 6 5
Iterasi ke-5: 1 2 3 4 5 6
Data setelah sorting: 1 2 3 4 5 6
```

1.4. TUGAS DAN LATIHAN

1. Ubahlah ketiga program yang terdapat pada sub bab C “PERCOBAAN” menjadi pengurutan secara decending (dari besar ke kecil)
2. Seorang programer diberi tugas untuk membuat program perpustakaan sederhana untuk mempermudah proses pencarian buku. Oleh karena itu, sebagai langkah pertama yang harus dilakukan, programer tersebut perlu menentukan metode apa yang tepat digunakan, jika ingin membuat program yang dapat menginput judul buku, kemudian mengurutkannya secara ascending berdasarkan alfabet. Kemudian, buatlah program tersebut!