



Struktur Data  
Pendidikan Teknik Informatika  
Universitas Negeri Padang  
TIM DOSEN ©2022

# JOBSHEET (JS-10)

## Sorting (1)

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.2317	4 x 50 Menit

### **TUJUAN PRAKTIKUM**

1. Mahasiswa mampu mengaplikasikan konsep Bubble Sort dalam menyelesaikan kasus.
2. Mahasiswa mampu mengaplikasikan konsep Insertion Sort dalam menyelesaikan kasus.
3. Mahasiswa mampu mengaplikasikan konsep Selection Sort dalam menyelesaikan kasus.

### **HARDWARE & SOFTWARE**

1. Personal Computer
2. IDE: Dev C++

---

### **TEORI SINGKAT**

#### **A. Bubble Sort**

*Bubble sort* (metode gelembung) adalah metode/algoritma pengurutan dengan dengan cara melakukan pertukaran data dengan tepat disebelahnya secara terus menerus sampai bisa dipastikan dalam satu iterasi tertentu tidak ada lagi perubahan. Jika tidak ada perubahan berarti data sudah terurut. Disebut pengurutan gelembung karena masing-masing kunci akan dengan lambat menggelembung ke posisinya yang tepat.

Metode pengurutan gelembung (Bubble Sort) diinspirasikan oleh gelembung sabun yang berada di permukaan air. Karena berat jenis gelembung sabun lebih ringan daripada berat jenis air, maka gelembung sabun selalu terapung ke atas permukaan. Prinsip di atas

dipakai pada pengurutan gelembung.

Algoritma bubble sort adalah salah satu algoritma pengurutan yang paling simple, baik dalam hal pengertian maupun penerapannya. Ide dari algoritma ini adalah mengulang proses pembandingan antara tiap-tiap elemen array dan menukaranya apabila urutannya salah. Pembandingan elemen-elemen ini akan terus diulang hingga tidak perlu dilakukan penukaran lagi. Algoritma ini termasuk dalam golongan algoritma comparison sort, karena menggunakan perbandingan dalam operasi antar elemennya. Berikut ini adalah gambaran dari algoritma bubble sort.

Misalkan elemen array adalah :

13	32	26	35	10
----	----	----	----	----

### Pass Pertama

Penyortiran akan dimulai dari dua elemen awal. Mari bandingkan mereka untuk memeriksa mana yang lebih besar.

13	32	26	35	10
----	----	----	----	----

Di sini, 32 lebih besar dari 13 ( $32 > 13$ ), jadi sudah diurutkan. Sekarang, bandingkan 32 dengan 26.

13	32	26	35	10
----	----	----	----	----

Di sini, 26 lebih kecil dari 32. Jadi, swapping diperlukan. Setelah menukar array baru akan terlihat seperti -

13	26	32	35	10
----	----	----	----	----

Sekarang, bandingkan 32 dan 35.

13	26	32	35	10
----	----	----	----	----

Di sini, 35 lebih besar dari 32. Jadi, tidak diperlukan swapping karena sudah diurutkan.

Sekarang, perbandingannya akan berada di antara 35 dan 10.

13	26	32	35	10
----	----	----	----	----

Di sini, 10 lebih kecil dari 35 yang tidak diurutkan. Jadi, swapping diperlukan. Sekarang, kita sampai di akhir array. Setelah pass pertama, array akan menjadi -

13	26	32	10	35
----	----	----	----	----

Sekarang, pindah ke iterasi kedua.

### Pass Kedua

Proses yang sama akan diikuti untuk iterasi kedua.

13	26	32	10	35
----	----	----	----	----

13	26	32	10	35
----	----	----	----	----

13	26	32	10	35
----	----	----	----	----

Di sini, 10 lebih kecil dari 32. Jadi, swapping diperlukan. Setelah bertukar, array akan menjadi -

13	26	10	32	35
----	----	----	----	----

13	26	10	32	35
----	----	----	----	----

Sekarang, pindah ke iterasi ketiga.

### Pass Ketiga

Proses yang sama akan diikuti untuk iterasi ketiga.

13	26	10	32	35
----	----	----	----	----

13	26	10	32	35
----	----	----	----	----

Di sini, 10 lebih kecil dari 26. Jadi, swapping diperlukan. Setelah bertukar, array akan menjadi -

13	10	26	32	35
----	----	----	----	----

13	10	26	32	35
----	----	----	----	----

13	10	26	32	35
----	----	----	----	----

Sekarang, pindah ke iterasi keempat.

### Pass Keempat

Demikian pula, setelah iterasi keempat, array akan menjadi -

10	13	26	32	35
----	----	----	----	----

Oleh karena itu, tidak diperlukan swapping, sehingga array sepenuhnya diurutkan.

## B. Insertion Sort

Insertion sort bekerja mirip dengan penyortiran kartu remi di tangan. Diasumsikan bahwa kartu pertama sudah diurutkan dalam permainan kartu, dan kemudian kami memilih kartu yang tidak disortir. Jika kartu yang tidak disortir lebih besar dari kartu pertama, itu akan ditempatkan di sisi kanan; jika tidak, itu akan ditempatkan di sisi kiri. Demikian pula, semua kartu yang tidak disortir diambil dan diletakkan di tempat yang tepat.

Pendekatan yang sama diterapkan dalam insertion sort. Gagasan di balik jenis penyisipan adalah pertama-tama ambil satu elemen, ulangi melalui array yang diurutkan. Langkah-langkah sederhana untuk mencapai jenis penyisipan tercantum sebagai berikut :

1. **Langkah 1** - Jika elemen adalah elemen pertama, anggap sudah diurutkan. Kembali 1.
2. **Langkah 2** - Pilih elemen berikutnya, dan simpan secara terpisah di sebuah **kunci**.
3. **Langkah 3** - Sekarang, bandingkan **kunci** dengan semua elemen dalam array yang diurutkan.

4. **Langkah 4** - Jika elemen dalam array yang diurutkan lebih kecil dari elemen saat ini, maka pindah ke elemen berikutnya. Jika tidak, geser elemen yang lebih besar dalam larik ke arah kanan.
5. **Langkah 5** - Masukkan nilainya.
6. **Langkah 6** - Ulangi sampai array diurutkan.

Untuk memahami cara kerja algoritma insertion sort, mari kita ambil array yang tidak disortir. Akan lebih mudah untuk memahami jenis penyisipan melalui contoh.

Misalkan elemen array adalah :

12	31	25	8	32	17
----	----	----	---	----	----

Awalnya, dua elemen pertama dibandingkan dalam jenis penyisipan.

12	31	25	8	32	17
----	----	----	---	----	----

Di sini, 31 lebih besar dari 12. Itu berarti kedua elemen sudah dalam urutan menaik. Jadi, untuk saat ini, 12 disimpan dalam sub-array yang diurutkan.

12	31	25	8	32	17
----	----	----	---	----	----

Sekarang, pindah ke dua elemen berikutnya dan bandingkan.

12	31	25	8	32	17
----	----	----	---	----	----

12	31	25	8	32	17
----	----	----	---	----	----

Di sini, 25 lebih kecil dari 31. Jadi, 31 tidak pada posisi yang benar. Sekarang, tukar 31 dengan 25. Bersamaan dengan swapping, insertion sort juga akan memeriksanya dengan semua elemen dalam array yang diurutkan.

Untuk saat ini, array yang diurutkan hanya memiliki satu elemen, yaitu 12. Jadi, 25 lebih besar dari 12. Oleh karena itu, array yang diurutkan tetap diurutkan setelah bertukar.

12	25	31	8	32	17
----	----	----	---	----	----

Sekarang, dua elemen dalam array yang diurutkan adalah 12 dan 25. Maju ke elemen berikutnya yaitu 31 dan 8.

12	25	31	8	32	17
----	----	----	---	----	----

12	25	31	8	32	17
----	----	----	---	----	----

Baik 31 dan 8 tidak diurutkan. Jadi, tukar mereka.

12	25	8	31	32	17
----	----	---	----	----	----

Setelah bertukar, elemen 25 dan 8 tidak disortir.

12	25	8	31	32	17
----	----	---	----	----	----

Jadi, tukar mereka.

12	8	25	31	32	17
----	---	----	----	----	----

Sekarang, elemen 12 dan 8 tidak disortir.

12	8	25	31	32	17
----	---	----	----	----	----

Jadi, tukar mereka juga.

8	12	25	31	32	17
---	----	----	----	----	----

Sekarang, array yang diurutkan memiliki tiga item yaitu 8, 12 dan 25. Pindah ke item berikutnya yaitu 31 dan 32.

8	12	25	31	32	17
---	----	----	----	----	----

Oleh karena itu, mereka sudah diurutkan. Sekarang, array yang diurutkan mencakup 8, 12, 25 dan 31.

8	12	25	31	32	17
---	----	----	----	----	----

Pindah ke elemen berikutnya yaitu 32 dan 17.

8	12	25	31	32	17
---	----	----	----	----	----

17 lebih kecil dari 32. Jadi, tukarkan.

8	12	25	31	17	32
---	----	----	----	----	----

8	12	25	31	17	32
---	----	----	----	----	----

Bertukar membuat 31 dan 17 tidak disortir. Jadi, tukar mereka juga.

8	12	25	17	31	32
---	----	----	----	----	----

8	12	25	17	31	32
---	----	----	----	----	----

Sekarang, swapping membuat 25 dan 17 tidak disortir. Jadi, lakukan swapping lagi.

8	12	17	25	31	32
---	----	----	----	----	----

Sekarang, array benar-benar diurutkan.

### C. Selection Sort

Selection sort, nilai terkecil di antara elemen larik yang tidak diurutkan dipilih di setiap lintasan dan dimasukkan ke posisi yang sesuai ke dalam larik. Ini juga merupakan algoritma yang paling sederhana. Ini adalah algoritma pengurutan perbandingan di tempat. Dalam algoritma ini, array dibagi menjadi dua bagian, pertama adalah bagian yang diurutkan, dan yang lainnya adalah bagian yang tidak diurutkan. Awalnya, bagian array yang diurutkan kosong, dan bagian yang tidak disortir adalah array yang diberikan. Bagian yang diurutkan ditempatkan di sebelah kiri, sedangkan bagian yang tidak diurutkan ditempatkan di sebelah kanan.

Dalam selection sort, elemen terkecil pertama dipilih dari array yang tidak disortir dan ditempatkan pada posisi pertama. Setelah itu elemen terkecil kedua dipilih dan ditempatkan di posisi kedua. Proses berlanjut hingga array terurut seluruhnya.

Untuk memahami cara kerja algoritma Selection sort, mari kita ambil array yang tidak disortir. Akan lebih mudah untuk memahami pengurutan Seleksi melalui sebuah contoh.

Misalkan elemen array adalah :

12	29	25	8	32	17	40
----	----	----	---	----	----	----

Sekarang, untuk posisi pertama dalam larik terurut, seluruh larik harus dipindai secara berurutan.

Saat ini, **12** disimpan di posisi pertama, setelah mencari seluruh array, ditemukan bahwa **8** adalah nilai terkecil.

12	29	25	8	32	17	40
----	----	----	---	----	----	----

Jadi, tukar 12 dengan 8. Setelah iterasi pertama, 8 akan muncul di posisi pertama dalam array yang diurutkan.

8	29	25	12	32	17	40
---	----	----	----	----	----	----

Untuk posisi kedua, di mana 29 disimpan saat ini, kami kembali secara berurutan memindai sisa item dari array yang tidak disortir. Setelah memindai, kami menemukan bahwa 12 adalah elemen terendah kedua dalam array yang seharusnya muncul di posisi kedua.

8	29	25	12	32	17	40
---	----	----	----	----	----	----

Sekarang, tukar 29 dengan 12. Setelah iterasi kedua, 12 akan muncul di posisi kedua dalam array yang diurutkan. Jadi, setelah dua iterasi, dua nilai terkecil ditempatkan di awal dengan cara yang diurutkan.

8	12	25	29	32	17	40
---	----	----	----	----	----	----

Proses yang sama diterapkan ke elemen array lainnya. Sekarang, kami menunjukkan representasi bergambar dari seluruh proses penyortiran.

8	12	25	29	32	17	40
---	----	----	----	----	----	----

8	12	25	29	32	17	40
---	----	----	----	----	----	----

8	12	17	29	32	25	40
---	----	----	----	----	----	----

8	12	17	29	32	25	40
---	----	----	----	----	----	----

8	12	17	29	32	25	40
---	----	----	----	----	----	----

8	12	17	25	32	29	40
---	----	----	----	----	----	----

8	12	17	25	32	29	40
---	----	----	----	----	----	----

8	12	17	25	32	29	40
---	----	----	----	----	----	----

8	12	17	25	29	32	40
---	----	----	----	----	----	----

8	12	17	25	29	32	40
---	----	----	----	----	----	----

Sekarang, array benar-benar diurutkan.

## LATIHAN

### 1. Bubble Sort secara ascending

```

/* Nama File  : bubble sort ascending
Pembuat      : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

// Bubble Sort secara ascending

#include<stdio.h>
void print(int a[], int n) //function to print array
elements
{
    int i;
    for(i = 0; i < n; i++)
    {
        printf("%d ",a[i]);
    }
}

void bubble(int a[], int n) // function to implement bubble
sort
{
    int i, j, temp;
    for(i = 0; i < n; i++)

```

```
{  
    for(j = i+1; j < n; j++)  
    {  
        if(a[j] < a[i])  
        {  
            temp = a[i];  
            a[i] = a[j];  
            a[j] = temp;  
        }  
    }  
}  
  
int main ()  
{  
    int i, j,temp;  
    int a[5] = { 10, 35, 32, 13, 26};  
    int n = sizeof(a)/sizeof(a[0]);  
    printf("Before sorting array elements are - \n");  
    print(a, n);  
    bubble(a, n);  
    printf("\nAfter sorting array elements are - \n");  
    print(a, n);  
}
```

## 2. Insertion Sort secara ascending

```
/* Nama File : insertion sort ascending  
Pembuat : tuliskan nama dan NIM anda  
Tgl pembuatan : tuliskan tanggal hari ini*/  
  
// Insertion Sort secara ascending  
  
#include <stdio.h>  
  
void insert(int a[], int n) /* function to sort an array  
with insertion sort */
```

```

{
    int i, j, temp;
    for (i = 1; i < n; i++) {
        temp = a[i];
        j = i - 1;

        while(j>=0 && temp <= a[j]) /* Move the elements
greater than temp to one position ahead from their current
position*/
        {
            a[j+1] = a[j];
            j = j-1;
        }
        a[j+1] = temp;
    }
}

void printArr(int a[], int n) /* function to print the
array */
{
    int i;
    for (i = 0; i < n; i++)
        printf("%d ", a[i]);
}

int main()
{
    int a[] = { 12, 31, 25, 8, 32, 17 };
    int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting array elements are - \n");
    printArr(a, n);
    insert(a, n);
    printf("\nAfter sorting array elements are - \n");
    printArr(a, n);

    return 0;
}

```

```
}
```

```
}
```

### 3. Selection Sort secara ascending

```
/* Nama File : selection sort ascending
Pembuat : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/



// Selection Sort secara ascending

#include <stdio.h>

void selection(int arr[], int n)
{
    int i, j, small;

    for (i = 0; i < n-1; i++)      // One by one move boundary
of unsorted subarray
    {
        small = i; //minimum element in unsorted array

        for (j = i+1; j < n; j++)
        if (arr[j] < arr[small])
            small = j;

        // Swap the minimum element with the first element
        int temp = arr[small];
        arr[small] = arr[i];
        arr[i] = temp;
    }
}

void printArr(int a[], int n) /* function to print the array */
{
    int i;
    for (i = 0; i < n; i++)
```

```

        printf("%d ", a[i]);
    }

int main()
{
    int a[] = { 12, 31, 25, 8, 32, 17 };
    int n = sizeof(a) / sizeof(a[0]);
    printf("Before sorting array elements are - \n");
    printArr(a, n);
    selection(a, n);
    printf("\nAfter sorting array elements are - \n");
    printArr(a, n);
    return 0;
}

```

## TUGAS

1. Pahami dan pelajari program Bubble Sort pada Latihan, setelah itu editlah program tersebut sehingga menjadi program **Bubble Sort secara descending**.
2. Pahami dan pelajari program Insertion Sort pada Latihan, setelah itu editlah program tersebut sehingga menjadi program **Insertion Sort secara descending**.
3. Pahami dan pelajari program Selection Sort pada Latihan, setelah itu editlah program tersebut sehingga menjadi program **Selection Sort secara descending**.

## DAFTAR PUSTAKA

Kelley, Al and Pohl, Ira. 2003. *C by Dissection: The Essentials of C Programming*. Addison-Weasley.

Khannedy, Eko Kurniawan. 2007. *Diktat Pemrograman C*. Bandung : Unikom.

Liem, Ingriani. 2003. *Catatan Singkat Bahasa C*. Jurusan Teknik Informatika. Institut Teknologi Bandung.

Reema Thareja. 2014. *Data Structures Using C*. India : Oxford University Press.

Solichin, Achmad. 2003. *Pemrograman Bahasa C dengan Turbo C*. Kuliah Berseri ilmukomputer.com