Laporan Individu Bubble Sort
Kelompok 5

Nama : Muhammad Zidan Alif Oktavian
Kelas / NIM : MI21A / 21091397045

1) Program C++ Bubble Sort

a) Program Coding

```cpp
//Program C/C++ untuk implementasi//
//dari Bubble Sort//
#include <iostream>
#include <conio.h>

//Fungsi untuk mengimplementasikan bubble sort//
using namespace std;
    int data[30], data2[30];
    int Jml_Data;

//Base case//
int tukar (int a,int b){
    int t;
    t=data[b];
    data[b]=data[a];
    data[a]=t;
}

//Berapa Jumlah data yg akan dimasukkan//
int input(){
    cout<<"Masukan Jumlah Data = ";
cin>>Jml_Data;

cout<<endl;

//Untuk memasukkan data yg akan di sort//
    for (int i=0;i<Jml_Data;i++){
        cout<<"Masukan Data Ke-"<<i+1<<" = ";
        cin>>data[i];

        data2[i]=data[i];
    }
    cout<<endl;
}
```

```cpp
//Fungsi untuk mencetak array//
int tampil(){
    for (int i=0;i<Jml_Data;i++){
        cout<<"["<<data[i]<<"] ";
    }
    cout<<endl;
}
    //Bubble sort Process//
//One pass of bubble sort. After//
//this pass, the largest element//
//is moved (or bubbled) to end.//
int bubble_sort(){
    for (int i=1; i<Jml_Data;i++){
        for (int j=Jml_Data-1; j>=i;j--){
            if (data[j]<data[j-1]){
                tukar(j,j-1);
            }
        }
        tampil();
    }
    cout<<endl;
}

//Program driver untuk menguji fungsi di atas//
int main()
  {
    cout<<"ALGORITMA BUBBLE SORT"<<endl;
    cout<<"----------------------"<<endl;
    input();
    cout<<"Proses Bubble Sort"<<endl;
    tampil();
    bubble_sort();
  getch();
  }
```

b) Hasil Output

```
ALGORITMA BUBBLE SORT
----------------------
Masukan Jumlah Data = 5

Masukan Data Ke-1 = 42
Masukan Data Ke-2 = 35
Masukan Data Ke-3 = 12
Masukan Data Ke-4 = 77
Masukan Data Ke-5 = 5

Proses Bubble Sort
[42] [35] [12] [77] [5]
[5] [42] [35] [12] [77]
[5] [12] [42] [35] [77]
[5] [12] [35] [42] [77]
[5] [12] [35] [42] [77]
```

2) Bubble Sort

a) Pengertian

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. Pengurutan Ascending dilakukan jika elemen sekarang lebih besar dari elemen berikutnya maka kedua elemen tersebut ditukar. Pengurutan Descending dilakukan jika elemen sekarang lebih kecil dari elemen berikutnya, maka kedua elemen tersebut ditukar.Ketika satu proses telah selesai, maka bubble sort akan mengulangi proses, demikian seterusnya sampai dengan iterasi sebanyak n-1. Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan. Proses Bubble Sort Data paling akhir dibandingkan dengan data di depannya, jika ternyata lebih kecil atau besar maka tukar sesuai dengan ketentuan (descending atau ascending). Dan pengecekan yang sama dilakukan terhadap data yang selanjutnya sampai dengan data yang paling awal. Di bawah ini adalah gambar proses Bubble sort.

Algoritma bubble dapat dituliskan sebagai berikut (ascending):1. i = 02. selama (i < N-1) kerjakan baris 3 sampai 73. j = N 14. Selama (j >= i) kerjakan baris 5 sampai 75. Jika (Data[j-1] > Data[j]) maka tukar Data[j-1] dengan Data[j]6. j = j 17. i = i + 1 Algoritma dan Procedure Bubble Prosedur yang menggunakan metode gelembung:

void BubbleSort(){ int i, j; for(i=1; i=i; j--) if(Data[j-1] > Data[j]) Tukar(&Data[j-1], &Data[j]);} Dengan prosedur tersebut, data terurut naik (ascending).

b) Kompleksitas Bubble Sort

Kompleksitas Algoritma Bubble Sort dapat dilihat dari beberapa jenis kasus, yaitu worst-case, average-case, dan best-case.

i) Kondisi Best Case

Dalam kasus ini, data yang akan disorting telah terurut sebelumnya, sehingga proses perbandingan hanya dilakukan sebanyak (n-1) kali, dengan satu kali pass. Proses perbandingan dilakukan hanya untuk memverifikasi keurutan data. Contoh Best-Case dapat dilihat pada pengurutan data “1 2 3 4” di bawah ini.

Pass Pertama

(**1 2** 3 4) menjadi (**1 2** 3 4)
(1 **2 3** 4) menjadi (1 **2 3** 4)
(1 2 **3 4**) menjadi (1 2 **3 4**)

Dari proses di atas, dapat dilihat bahwa tidak terjadi penukaran posisi satu kalipun, sehingga tidak dilakukan pass selanjutnya. Perbandingan elemen dilakukan sebanyak tiga kali.

Proses perbandingan pada kondisi ini hanya dilakukan sebanyak (n-1) kali. Persamaan Big-O yang diperoleh dari proses ini adalah O(n). Dengan kata lain, pada kondisi Best-Case algoritma Bubble Sort termasuk pada algoritma lanjar.

ii) Kondisi Worst Case

Contoh Worst-Case dapat dilihat pada pengurutan data “4 3 2 1” di bawah ini.

Pass Pertama

(**4 3** 2 1) menjadi (**3 4** 2 1)
(3 **4 2** 1) menjadi (3 **2 4** 1)
(3 2 **4 1**) menjadi (3 2 **1 4**)

Pass Kedua

(**3 2** 1 4) menjadi (**2 3** 1 4)
(2 **3 1** 4) menjadi (2 **1 3** 4)
(2 1 **3 4**) menjadi (2 1 **3 4**)

Pass Ketiga

(**2 1** 3 4) menjadi (**1 2** 3 4)
(1 **2 3** 4) menjadi (1 **2 3** 4)
(1 2 **3 4**) menjadi (1 2 **3 4**)

Pass Keempat

(**1 2** 3 4) menjadi (**1 2** 3 4)
(1 **2 3** 4) menjadi (1 **2 3** 4)
(1 2 **3 4**) menjadi (1 2 **3 4**)

Dari langkah pengurutan di atas, terlihat bahwa setiap kali melakukan satu pass, data terkecil akan bergeser ke arah awal sebanyak satu step. Dengan kata lain, untuk menggeser data terkecil dari urutan keempat menuju urutan pertama, dibutuhkan pass sebanyak tiga kali, ditambah satu kali pass untuk memverifikasi. Sehingga jumlah proses pada kondisi best case dapat dirumuskan sebagai berikut.

Jumlah proses = n2+n (3)

Dalam persamaan (3) di atas, n adalah jumlah elemen yang akan diurutkan. Sehingga notasi Big-O yang didapat adalah $O(n^2)$. Dengan kata lain, pada kondisi worst-case, algoritma Bubble Sort termasuk dalam kategori algoritma kuadratik.

iii) Kondisi Average Case

Pada kondisi average-case, jumlah pass ditentukan dari elemen mana yang mengalami penggeseran ke kiri paling banyak. Hal ini dapat ditunjukkan oleh proses pengurutan suatu array, misalkan saja (1 8 6 2). Dari (1 8 6 2), dapat dilihat bahwa yang akan mengalami proses penggeseran paling banyak adalah elemen 2, yaitu sebanyak dua kali.

Pass Pertama

(**1 8** 6 2) menjadi (**1 8** 6 2)
(1 **8 6** 2) menjadi (1 **6 8** 2)
(1 6 **8 2**) menjadi (1 6 **2 8**)

Pass Kedua

(**1 6** 2 8) menjadi (**1 6** 2 8)
(1 **6 2** 8) menjadi (1 **2 6** 8)

(1 2 **6 8**) menjadi (1 2 **6 8**)

Pass Ketiga

(**1 2** 6 8) menjadi (**1 2** 6 8)

(1 **2 6** 8) menjadi (1 **2 6** 8)

(1 2 **6 8**) menjadi (1 2 **6 8**)

Dari proses pengurutan di atas, dapat dilihat bahwa untuk mengurutkan diperlukan dua buah passing, ditambah satu buah passing untuk memverifikasi. Dengan kata lain, jumlah proses perbandingan dapat dihitung sebagai berikut.

Jumlah proses = x^2+x (4)

Dalam persamaan (4) di atas, x adalah jumlah penggeseran terbanyak. Dalam hal ini, x tidak pernah lebih besar dari n, sehingga x dapat dirumuskan sebagai

Dari persamaan (4) dan (5) di atas, dapat disimpulkan bahwa notasi big-O nya adalah O(n2). Dengan kata lain, pada kondisi average case algoritma Bubble Sort termasuk dalam algoritma kuadratik.

3) **Kelebihan dan Kekurangan Bubble Sort**

a) Kelebihan

- Proses penghitungan Bubble sort merupakan metode yang paling sederhana
- Algoritma Bubble Sort mudah dipahami
- Langkah atau tahapan dalam pengurutan data sangat sederhana.
- Cocok untuk pengurutan data dengan elemen kecil telah terurut.

b) Kekurangan

- Proses penghitungan Bubble Sort menggunakan metode pengurutan termasuk paling tidak efisien walaupun dianggap sederhana. Karena proses pengurutan data dilakukan dengan tahapana satu - satu, mulai dari data paling awal sebelah kiri, sampai data terakhir.
- Ketika data yang kita punya banyak atau dalam jumlah yang besar, maka proses penghitungan akan semakin lama dan lambat. Karena proses pengurutan data secara tunggal (satu - satu).
- Jumlah pengulangan akan tetap sama sampai ke data yang terakhir, walaupun sebagian data yang ada telah terurut.