LAPORAN INDIVIDU BUBBLE SORT

KELOMPOK 6

Nama : Putri Dwinatryska A.R.F

Kelas/ Nim :A / 21091397075

1. Program C++ Bubble Sort

```
20  int main(){
21      //fungsi void untuk input array
22    int array[100], n, i, j;
23    //menampilkan kalimat
24    cout<<"PROGRAM BILANGAN BUBBLE SORT \n\n";
25    //elemen input
26    cout << "Masukkan jumlah bilangan: ";
27    cin >> n;
28    cout << "Masukkan bilangan: ";
29    //fungsi void mencetak array yang dimasukan
30    for (i = 0; i < n; i++){
31      cin >> array[i];
32    }
33    bubbleSort(array, n);
34    //fungsi void untuk menampilkan hasil sorting array
35    cout << "Hasil array yang sudah disorting :\n";
36    for (i = 0; i < n; i++){
37      cout <<"["<< array[i] << "]";
38    }
39    cout << "\n";
40    //jejak develop
41    cout << "develop_putri dwinatryska_21091397075_kel 6";
42  }
```

Gambar 1.1 program bubble sort

```
20  int main(){
21      //fungsi void untuk input array
22    int array[100], n, i, j;
23    //menampilkan kalimat
24    cout<<"PROGRAM BILANGAN BUBBLE SORT \n\n";
25    //elemen input
26    cout << "Masukkan jumlah bilangan: ";
27    cin >> n;
28    cout << "Masukkan bilangan: ";
29    //fungsi void mencetak array yang dimasukan
30    for (i = 0; i < n; i++){
31      cin >> array[i];
32    }
33    bubbleSort(array, n);
34    //fungsi void untuk menampilkan hasil sorting array
35    cout << "Hasil array yang sudah disorting :\n";
36    for (i = 0; i < n; i++){
37      cout <<"["<< array[i] << "]";
38    }
39    cout << "\n";
40    //jejak develop
41    cout << "develop_putri dwinatryska_21091397075_kel 6";
42  }
```

Gambar 1.2 program Bubble sort

A. Hasil Output Program

```
C:\Users\Mybook 14G\Downloads\bubble_sort.exe
PROGRAM BILANGAN BUBBLE SORT

Masukkan jumlah bilangan: 5
Masukkan bilangan: 4 20 3 9 13
Hasil array yang sudah disorting :
[3][4][9][13][20]
develop_putri dwinatryska_21091397075_kel 6
--------------------------------
Process exited after 8.546 seconds with return value 0
Press any key to continue . . .
```

Gambar 1. 3 Hasil Output Program

2. Bubble short

A. Pengertian Konsep Dasar Bubble Short

Bubble Sort adalah salah satu algoritma untuk sorting data, atau kata lainnya mengurutkan data dari yang terbesar ke yang terkecil atau sebaliknya (Ascending atau Descending)

Bubble sort menggunakan metode/algoritma pengurutan dengan dengan cara melakukan penukaran data secara terus menerus sampai sampai tidak ada lagi perubahan. Jika tidak ada perubahan berarti data sudah terurut.

Metode pengurutan gelembung (Bubble Sort) diinspirasikan oleh gelembung sabun yang berada dipermukaan 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 menukarnya 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 kita mempunyai sebuah array dengan. Elemen-elemen “4 2 5 3 9”. Proses yang akan terjadi apabila digunakan algoritma bubblesort adalah sebagai berikut.

Pass pertama

(4 2 5 3 9) menjadi (2 4 5 3 9)

(2 4 5 3 9) menjadi (2 4 5 3 9)

(2 4 5 3 9) menjadi (2 4 3 5 9)

(2 4 3 5 9) menjadi (2 4 3 5 9)

Pass kedua

(2 4 3 5 9) menjadi (2 4 3 5 9)

(2 4 3 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

Pass ketiga

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

(2 3 4 5 9) menjadi (2 3 4 5 9)

Dapat dilihat pada proses di atas, sebenarnya pada pass kedua, langkah kedua, array telah terurut. Namun algoritma tetap dilanjutkan hingga pass kedua berakhir. Pass ketiga dilakukan karena definisi terurut dalam algoritma bubblesort adalah tidak ada satupun penukaran pada suatu pass, sehingga pass ketiga dibutuhkan untuk memverifikasi array tersebut.

B. Kompleksitas Bubble Sort

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

 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.

 Kondisi Worst-Case

Dalam kasus ini, data terkecil berada pada ujung array. 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(n2). Dengan kata lain, pada kondisi worst-case, algoritma Bubble Sort termasuk dalam kategori algoritma kuadratik.

 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 = x2+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.

C. Kelebihan dan Kekurangan Bubble Sort Kelebihan Bubble Sort :

1. Proses penghitungan Bubble sort merupakan metode yang paling sederhana
2. Algoritma Bubble Sort mudah dipahami
3. Langkah atau tahapan dalam pengurutan data sangat sederhana.

Kekurangan Bubbe Sort :

1. 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
2. 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).
3. Jumlah pengulangan akan tetap sama sampai ke data yang terakhir, walaupun sebagian data yang ada telah terurut.