LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA

Judul: Analisis Asimtotik

DISUSUN OLEH

Ilham Nur Romdoni M0520038

PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SEBELAS MARET

2021

1. **algoritma-1.cpp**

```cpp
#include<iostream>
#include<time.h>
#include<chrono>
#include <array>
using namespace std;
using namespace std::chrono;
void swap(int * xp, int * yp){
    int temp = *xp;
    *xp = *yp;
    *yp = temp;
}
void STAY(int arr[], int n){
    int i, j;
    for(i=0; i<n-1; i++){
        for(j=0; j<n-i-1; j++){
            if(arr[j]>arr[j+1]) swap(&arr[j], &arr[j+1]);
        }
    }
}
```

algoritma-1.cpp menggunakan fungsi `swap` yang digunakan untuk menukarkan dua buah nilai. Pada tipe data integer, dibuat sebuah *pointer*. Pointer ini hanya bisa menunjuk ke variabel yang dideklarasikan. Terdapat juga fungsi `STAY` yang merupakan algoritma *bubble sort* untuk mengurutkan angka-angka dengan menggunakan fungsi `swap`. Algoritma *bubble sort* diaplikasikan dengan cara membandingkan elemen ke satu dengan elemen selanjutnya.

a. Big-O

```cpp
    for(i=0; i<n-1; i++){
        for(j=0; j<n-i-1; j++){
            if(arr[j]>arr[j+1]) swap(&arr[j], &arr[j+1]);
        }
    }
}
```

Loop pada baris ke 13 dan 14 masing-masing memiliki Big O = O(n) dan operasi lainnya adalah O(1).. Jadi, Big O-nya adalah O(n.n)= $O(n^2)$

b. Analisa Test Case

```cpp
int main () {
    int n=10000;
    int best[n], worst[n], random[n];

    for (int i=0; i<n; i++) best[i] = i;
    for (int i=0; i<n; i++) worst[i] = n - 1;
    for (int i=0; i<n; i++) random[i] = rand() % n;
```

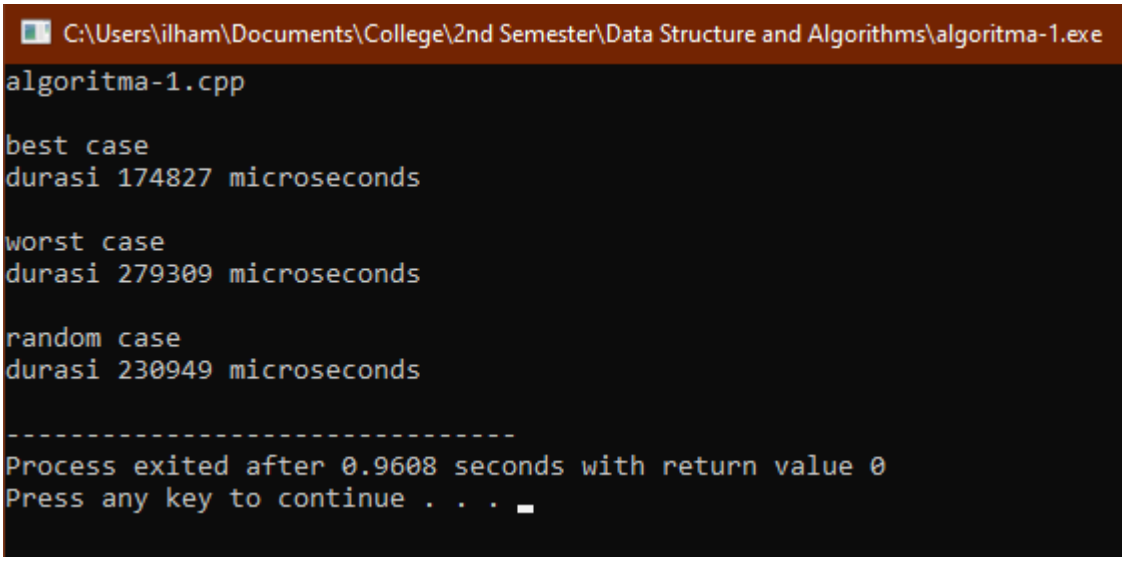
Pada baris ke-21, merupakan fungsi main yang akan dijalankan pertama kali saat mendeklarasikan *variable* dan *array*, kemudian untuk *best case* yang akan digunakan data dengan urutan asli, untuk *worst case* menggunakan data yang urutannya terbalik, sedangkan *random case* menggunakan data acak untuk membandingkan.

```
29      cout << "algoritma-1.cpp\n";
30      {
31          cout << "\nbest case\n";
32          auto t0 = high_resolution_clock::now();
33          STAY(best, n);
34          auto t1 = high_resolution_clock::now();
35          auto dt = t1 - t0;
36          long long dtms = duration_cast<microseconds>(dt).count();
37          cout << "durasi " << dtms << " microseconds\n";
38      }
39      {
40          cout << "\nworst case\n";
41          auto t0 = high_resolution_clock::now();
42          STAY(worst, n);
43          auto t1 = high_resolution_clock::now();
44          auto dt = t1 - t0;
45          long long dtms = duration_cast<microseconds>(dt).count();
46          cout << "durasi " << dtms << " microseconds\n";
47      }
48      {
49          cout << "\nrandom case\n";
50          auto t0 = high_resolution_clock::now();
51          STAY(random, n);
52          auto t1 = high_resolution_clock::now();
53          auto dt = t1 - t0;
54          long long dtms = duration_cast<microseconds>(dt).count();
55          cout << "durasi " << dtms << " microseconds\n";
56      }
```

```
// bahasa C++11 (ringkas)
#include <chrono>
using namespace std::chrono;
auto t0 = high_resolution_clock::now();
// ... (kode yang diukur)
auto t1 = high_resolution_clock::now();
auto dt = t1 - t0;
long long dtms = duration_cast<microseconds>(dt).count(); // mikrodetik
```

Baris ke-30 merupakan perintah pengukuran waktu yang memilik format di atas. Kode dijalankan dengan cara mencari selisih waktu selesai dari waktu mulai dan

waktu selesai, sehingga didapatkan durasi waktu berjalannya program tersebut dalam satuan *microsecond*.



Compiler pemrograman menampilkan hasil sesuai dengan jalannya perintah yang dioperasikan dalam algoritma dan komponen masukan pada kode yang diukur.

1) *Best Case*: O(n)
2) *Worst Case*: O(n^2)
3) *Random Case*: O(n^2)

2. algoritma-2.cpp

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

void HOME(int arr[], int n){
    int i, key, j;
    for(i=1; i<n; i++){
        key = arr[i];
        j = i - 1;
        while(j>=0 && arr[j]>key){
            arr[j+1] = arr[j];
            j--;
        }
        arr[j+1] = key;
    }
}
```

algoritma-2.cpp menggunakan fungsi `HOME` yang merupakan algoritma *insertion sort* untuk mengurutkan angka-angka dengan akan membandingkan dua elemen data pertama, mengurutkannya, kemudian mengecek elemen data berikutnya satu persatu dan membandingkannya dengan elemen data yang telah diurutkan.

a. Big-O

```
7     for(i=1; i<n; i++){
8         key = arr[i];
9         j = i - 1;
10        while(j>=0 && arr[j]>key){
11            arr[j+1] = arr[j];
12            j--;
13        }
14        arr[j+1] = key;
15    }
```

Loop pada baris ke-7 dan 10 masing-masing memiliki Big O = O(n) dan operasi lainnya adalah O(1). Jadi, Big O-nya adalah O(n.n)= O(n^2)

b. Analisa Test Case

```
19  int main () {
20      int n=10000;
21      int best[n], worst[n], random[n];
22
23      for (int i=0; i<n; i++) best[i] = i;
24      for (int i=0; i<n; i++) worst[i] = n - i;
25      for (int i=0; i<n; i++) random[i] = rand() % n;
```

Pada baris ke-19, merupakan fungsi main yang akan dijalankan pertama kali saat mendeklarasikan *variable* dan *array*, kemudian untuk *best case* yang akan digunakan data dengan urutan asli, untuk *worst case* menggunakan data yang urutannya terbalik, sedangkan *random case* menggunakan data acak untuk membandingkan.

```
// bahasa C++11 (ringkas)
#include <chrono>
using namespace std::chrono;
auto t0 = high_resolution_clock::now();
// ... (kode yang diukur)
auto t1 = high_resolution_clock::now();
auto dt = t1 - t0;
long long dtms = duration_cast<microseconds>(dt).count(); // mikrodetik
```

Baris ke-27 merupakan perintah pengukuran waktu yang memilik format di atas. Kode dijalankan dengan cara mencari selisih waktu selesai dari waktu mulai dan waktu selesai, sehingga didapatkan durasi waktu berjalannya program tersebut dalam satuan *microsecond*.

```
27      cout << "algoritma-2.cpp\n";
28      {
29          cout << "\nbest case\n";
30          auto t0 = high_resolution_clock::now();
31          HOME(best, n);
32          auto t1 = high_resolution_clock::now();
33          auto dt = t1 - t0;
34          long long dtms = duration_cast<microseconds>(dt).count();
35          cout << "durasi " << dtms << " microseconds\n";
36      }
37      {
38          cout << "\nworst case\n";
39          auto t0 = high_resolution_clock::now();
40          HOME(worst, n);
41          auto t1 = high_resolution_clock::now();
42          auto dt = t1 - t0;
43          long long dtms = duration_cast<microseconds>(dt).count();
44          cout << "durasi " << dtms << " microseconds\n";
45      }
46      {
47          cout << "\nrandom case\n";
48          auto t0 = high_resolution_clock::now();
49          HOME(random, n);
50          auto t1 = high_resolution_clock::now();
51          auto dt = t1 - t0;
52          long long dtms = duration_cast<microseconds>(dt).count();
53          cout << "durasi " << dtms << " microseconds\n";
54      }
```

```
C:\Users\ilham\Documents\College\2nd Semester\Data Structure and Algorithms\algoritma-2.exe
algoritma-2.cpp

best case
durasi 0 microseconds

worst case
durasi 157648 microseconds

random case
durasi 73529 microseconds

--------------------------------
Process exited after 0.4767 seconds with return value 0
Press any key to continue . . .
```

Compiler pemrograman menampilkan hasil sesuai dengan jalannya perintah yang dioperasikan dalam algoritma dan komponen masukan pada kode yang diukur.

1) *Best Case*: O(n)
2) *Worst Case*: $O(n^{2)}$
3) *Random Case*: $O(n^2)$

3. **algoritma-3.cpp**

```cpp
#include<iostream>
#include<chrono>
using namespace std;
using namespace std::chrono;
int eraseAT(string str) {
    string acc;
    bool a, b;
    int x = 0;
    if (str.length() == 0)
        return 0;
    for (int i = 0; i < str.length(); i++) {
        if (str[i] == '[' || str[i] == ':' || str[i] == '|') {
            if (str[i] == '[') {
                acc.push_back(str[i]);
                a = true;
            }
            else if (str[i] == ':' && a) {
                if (str[i] == ':' && acc[acc.length() - 1] == ':') {
                    acc.pop_back();
                    b = false;
                }
                else {
                    acc.push_back(str[i]);
                    b = true;
                }
            }
            else if (str[i] == '|' && b) {}
            else x++;
        }
        else if (str[i] == ']') {
            if (acc.length() == 0)
                x++;
            else {
                acc.pop_back();
                a = false;
            }
        }
        else x++;
    }
    if (acc.length() == 0) return x;
    else return -1;
    for (int i = 0; i < str.length(); i++)
        for (int j = 0; j < str.length(); j++)
            for (int k = 0; k < str.length(); k++)
                str[i] = str[k];
    return 0;
}
```

algoritma-3.cpp menggunakan `eraseAT` merupakan pernyataan deklarasi *function* yang memberi tahu *compiler* akan berapa tipe data yang dibutuhkan untuk parameter, dan tipe data pada *return value* dari fungsi tersebut.

Fungsi `if` akan mengeksekusi jika suatu kondisi terpenuhi jika tidak maka disinilah fungsi dari `else` tersebut. Fungsi `else` akan melakukan perintah *looping*.

a. Big-O

```
11      for (int i = 0; i < str.length(); i++) {
12          if (str[i] == '[' || str[i] == ':' || str[i] == '|') {
13              if (str[i] == '[') {
14                  acc.push_back(str[i]);
15                  a = true;
16              }
17              else if (str[i] == ':' && a) {
18                  if (str[i] == ':' && acc[acc.length() - 1] == ':') {
19                      acc.pop_back();
20                      b = false;
21                  }
22                  else {
23                      acc.push_back(str[i]);
24                      b = true;
25                  }
26              }
27              else if (str[i] == '|' && b) {}
28              else x++;
29          }
30          else if (str[i] == ']') {
31              if (acc.length() == 0)
32                  x++;
33              else {
34                  acc.pop_back();
35                  a = false;
36              }
37          }
38          else x++;
39      }
40      if (acc.length() == 0) return x;
41      else return -1;
42      for (int i = 0; i < str.length(); i++)
43          for (int j = 0; j < str.length(); j++)
44              for (int k = 0; k < str.length(); k++)
45                  str[i] = str[k];
```

O(n).O(1).O(1)= O(n)

b. Analisa Test Case

```
49  int main () {
50      int n=100000;
51      string best = "", worst = "", random = "";
52
53      for (int i=0; i<n; i++) {
54          best.push_back('p');
55      }
56      for (int i=0; i<n; i++) {
57          char symbol[] = {'[', ':'};
58          worst.push_back(symbol[rand() % 2]);
59      }
60      for (int i=0; i<n; i++) {
61          char symbol[] = {'['};
62          random.push_back(symbol[rand() % 5]);
63      }
```

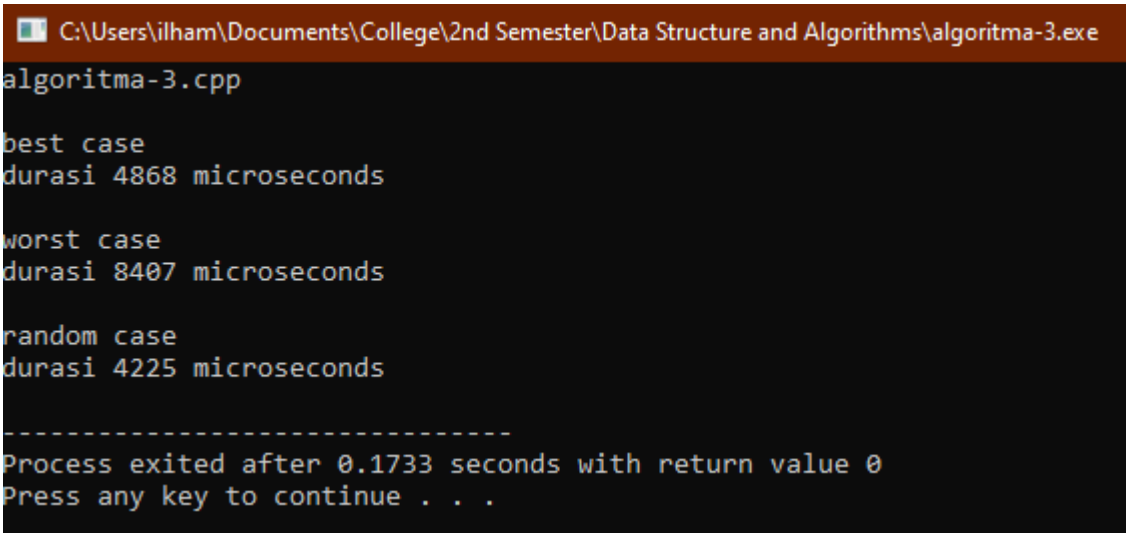
Pada baris ke-49, merupakan fungsi main yang akan dijalankan pertama kali saat mendeklarasikan *variable*. Untuk *best case* yang akan digunakan tipe data *string*

yang tidak berisi salah satu dari ‘[‘, ‘:’, ‘|’, atau ‘]’, untuk *worst case* menggunakan tipe data *string* yang hanya berisi salah satu dari ‘[‘ atau ‘:’, sedangkan random case menggunakan tipe data *string* yang hanya berisi salah satu dari dari ‘[‘, ‘:’, ‘|’, ‘]’, atau huruf. Setiap kondisi akan melakukan fungsi *loop*.

```
65      cout << "algoritma-3.cpp\n";
66      {
67          cout << "\nbest case\n";
68          auto t0 = high_resolution_clock::now();
69          eraseAT(best);
70          auto t1 = high_resolution_clock::now();
71          auto dt = t1 - t0;
72          long long dtms = duration_cast<microseconds>(dt).count();
73          cout << "durasi " << dtms << " microseconds\n";
74      }
75      {
76          cout << "\nworst case\n";
77          auto t0 = high_resolution_clock::now();
78          eraseAT(worst);
79          auto t1 = high_resolution_clock::now();
80          auto dt = t1 - t0;
81          long long dtms = duration_cast<microseconds>(dt).count();
82          cout << "durasi " << dtms << " microseconds\n";
83      }
84      {
85          cout << "\nrandom case\n";
86          auto t0 = high_resolution_clock::now();
87          eraseAT(random);
88          auto t1 = high_resolution_clock::now();
89          auto dt = t1 - t0;
90          long long dtms = duration_cast<microseconds>(dt).count();
91          cout << "durasi " << dtms << " microseconds\n";
92      }
93
94      return 0;
95  }
```

```
// bahasa C++11 (ringkas)
#include <chrono>
using namespace std::chrono;
auto t0 = high_resolution_clock::now();
// ... (kode yang diukur)
auto t1 = high_resolution_clock::now();
auto dt = t1 - t0;
long long dtms = duration_cast<microseconds>(dt).count(); // mikrodetik
```

Baris ke-65 merupakan perintah pengukuran waktu yang memilik format di atas. Kode dijalankan dengan cara mencari selisih waktu selesai dari waktu mulai dan waktu selesai, sehingga didapatkan durasi waktu berjalannya program tersebut dalam satuan *microsecond*.



Compiler pemrograman menampilkan hasil sesuai dengan jalannya perintah yang dioperasikan dalam algoritma dan komponen masukan pada kode yang diukur.

1) *Best Case*: O(n)
2) *Worst Case*: O(n)
3) *Random Case*: O(n)