

**LAPORAN PRAKTIKUM  
STRUKTUR DATA DAN ALGORITMA**

**ANALISIS ASIMTOTIK**



**DISUSUN OLEH**  
Latifah Hukma Shobiyya (M0520044)

Asisten:

1. Nuha Lina Atqia (M0519070)
2. Rizal Aji Purbadinata (M0519073)

**PROGRAM STUDI S-1 INFORMATIKA  
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM  
UNIVERSITAS SEBELAS MARET  
2021**

## 1. Algoritma-1

Source code swap function dan STAY function

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

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]);
        }
    }
}
```

## Source code main function

```
int main (){

    cout << "Best case Algoritma-1\n";
    for(int n = 1; n <= 10000; n+=1000){
        int best[n];
        cout << "n = " << n << ", " ;
        for(int i = 0; i < n; i++){
            best[i] = i;
        }
        auto t0 = chrono::high_resolution_clock::now();
        STAY(best, n);
        auto t1 = chrono::high_resolution_clock::now();
        auto dt = t1 - t0;
        long long dtms = chrono::duration_cast<chrono::microseconds>(dt).count();
        cout << "duration: " << dtms << " microseconds" << endl;
    }

    cout << "\nWorst case Algoritma-1\n";
    for(int n = 1; n <= 10000; n+=1000){
        int worst[n];
        cout << "n = " << n << ", " ;
        for(int i = 0; i < n; i++){
            worst[i] = n-i;
        }
        auto t0 = chrono::high_resolution_clock::now();
        STAY(worst, n);
        auto t1 = chrono::high_resolution_clock::now();
        auto dt = t1 - t0;
        long long dtms = chrono::duration_cast<chrono::microseconds>(dt).count();
        cout << "duration: " << dtms << " microseconds" << endl;
    }

    cout << "\nRandom case Algoritma-1\n";
    for(int n = 1; n <= 10000; n+=1000){
        int random[n];
        cout << "n = " << n << ", " ;
        for(int i = 0; i < n; i++){
            random[i] = rand() % 100;
        }
        auto t0 = chrono::high_resolution_clock::now();
        STAY(random, n);
        auto t1 = chrono::high_resolution_clock::now();
        auto dt = t1 - t0;
        long long dtms = chrono::duration_cast<chrono::microseconds>(dt).count();
        cout << "duration: " << dtms << " microseconds" << endl;
    }

    return 0;
}
```

#### a. Analisis Source Code

```
1  #include<iostream>
2  #include<chrono>
3  #include <array>
```

Baris 1-3 merupakan *preprocessor directive* yang berfungsi untuk menyisipkan berkas lain ke dalam berkas yang ingin dikompilasi

```
4  using namespace std;
```

Baris 4 digunakan untuk mengakses semua fungsi, class, atau file yang terdapat di dalam namespace 'std' tanpa harus menuliskan 'std::'

```
6  void swap(int * xp, int * yp){
7      int temp = *xp;
8      *xp = *yp;
9      *yp = temp;
10 }
```

Baris 6-10 merupakan fungsi swap yang berfungsi untuk menukarkan dua buah nilai (xp dan yp) dengan tipe data integer yang diakses dengan pointer.

```
11 void STAY(int arr[], int n){
12     int i, j;
13     for(i=0; i<n-1; i++){
14         for(j=0; j<n-i-1; j++){
15             if(arr[j]>arr[j+1]) swap(&arr[j], &arr[j+1]);
16         }
17     }
18 }
```

Baris 11-18 merupakan fungsi STAY yang mengurutkan nilai dengan menukar elemen yang berdekatan berulang kali (fungsi swap) jika urutannya salah ( $arr[j] > arr[j+1]$ ).

```
19 int main (){
```

Baris 19 adalah main function yang merupakan fungsi utama dari program C++ dengan nilai return berupa integer.

```
21     cout << "Best case Algoritma-1\n";
```

Baris 21 untuk mengeluarkan output berupa kalimat yang terletak di dalam tanda petik dua.

```
22     for(int n = 1; n <= 10000; n+=1000){
```

Baris 22 merupakan bentuk perulangan untuk n dimulai dari 1, lalu n terus ditambahkan dengan sebanyak 1000 hingga nilainya kurang dari atau sama dengan 10000.

```
23         int best[n];
```

Baris 23 mendeklarasikan array 'best' berisi n elemen.

```
24         cout << "n = " << n << ", " ;
```

Baris 24 mengeluarkan output berupa nilai yang terdapat pada variable n.

```

25     for(int i = 0; i < n; i++){
26         best[i] = i;
27     }

```

Baris 25-27 memuat perintah untuk mengassign array 'best' dengan nilai ascending

```

28     auto t0 = chrono::high_resolution_clock::now();

```

Baris 28 menyimpan waktu mulai dalam variable t0.

```

29     STAY(best, n);

```

Baris 29 memanggil fungsi STAY dengan best dan n sebagai argument.

```

30     auto t1 = chrono::high_resolution_clock::now();

```

Baris 30 menyimpan waktu selesai dalam variable t1.

```

31     auto dt = t1 - t0;

```

Baris 31 mendeklarasikan dt sebagai selisih dari t1 dan t0.

```

32     long long dtms = chrono::duration_cast<chrono::microseconds>(dt).count();

```

Baris 32 mendeklarasikan dtms sebagai durasi dari dt dengan satuan microsecond.

```

33     cout << "duration: " << dtms << " microseconds" << endl;

```

Baris 33 menampilkan nilai dtms.

```

38     int worst[n];

```

Baris 38 mendeklarasikan array 'worst' berisi n elemen.

```

40     for(int i = 0; i < n; i++){
41         worst[i] = n-i;
42     }

```

Baris 40-42 memuat perintah untuk mengassign array 'worst' dengan nilai descending.

```

44     STAY(worst, n);

```

Baris 44 memanggil fungsi STAY dengan worst dan n sebagai argument.

```

53     int random[n];

```

Baris 53 mendeklarasikan array 'random' berisi n elemen.

```

55     for(int i = 0; i < n; i++){
56         random[i] = rand() % 100;
57     }

```

Baris 55-57 memuat perintah untuk mengassign array 'random' dengan nilai random dalam rentang 100 nilai.

## **b. Analisis Program**

Pertama, akan dilakukan uji coba best case untuk algoritma-1, yakni dengan menggunakan 10 data integer ( $n = 1, 1001, 2001, \dots, 9001$ ) yang nilainya sudah urut dari yang terkecil hingga yang terbesar (ascending). Selanjutnya, dilakukan penyimpanan waktu mulai pada variable  $t_0$ , lalu fungsi STAY dijalankan dengan menggunakan 10 data ascending tadi. Pada fungsi STAY, terjadi pengurutan nilai dengan menukar elemen yang berdekatan berulang kali (fungsi swap) jika urutannya salah. Setelah fungsi STAY selesai dieksekusi, dilakukan penyimpanan waktu selesai. Selanjutnya, dilakukan penghitungan waktu eksekusi dengan mencari selisih antara waktu mulai dengan waktu selesai sehingga didapatkan durasi jalannya Algoritma-1 pada saat best case dengan satuan microsecond pada masing-masing  $n$ .

Selanjutnya, program akan melakukan uji coba worst case untuk algoritma-1, yakni dengan menggunakan 10 data integer ( $n = 1, 1001, 2001, \dots, 9001$ ) yang nilainya sudah urut dari yang terbesar ke yang terkecil (descending). Selanjutnya, dilakukan penyimpanan waktu mulai pada variable  $t_0$ , lalu fungsi STAY dijalankan dengan menggunakan 10 data descending tadi. Setelah fungsi STAY selesai dieksekusi, dilakukan penyimpanan waktu selesai. Selanjutnya, dilakukan penghitungan waktu eksekusi dengan mencari selisih antara waktu mulai dengan waktu selesai sehingga didapatkan durasi jalannya Algoritma-1 pada saat worst case dengan satuan microsecond pada masing-masing  $n$ .

Terakhir, program akan melakukan uji coba random case untuk algoritma-1, yakni dengan menggunakan 10 data integer ( $n = 1, 1001, 2001, \dots, 9001$ ) yang nilainya acak. Selanjutnya, dilakukan penyimpanan waktu mulai pada variable  $t_0$ , lalu fungsi STAY dijalankan dengan menggunakan 10 data acak tadi. Setelah fungsi STAY selesai dieksekusi, dilakukan penyimpanan waktu selesai. Selanjutnya, dilakukan penghitungan waktu eksekusi dengan mencari selisih antara waktu mulai dengan waktu selesai sehingga didapatkan durasi jalannya Algoritma-1 pada saat random case dengan satuan microsecond pada masing-masing  $n$ .

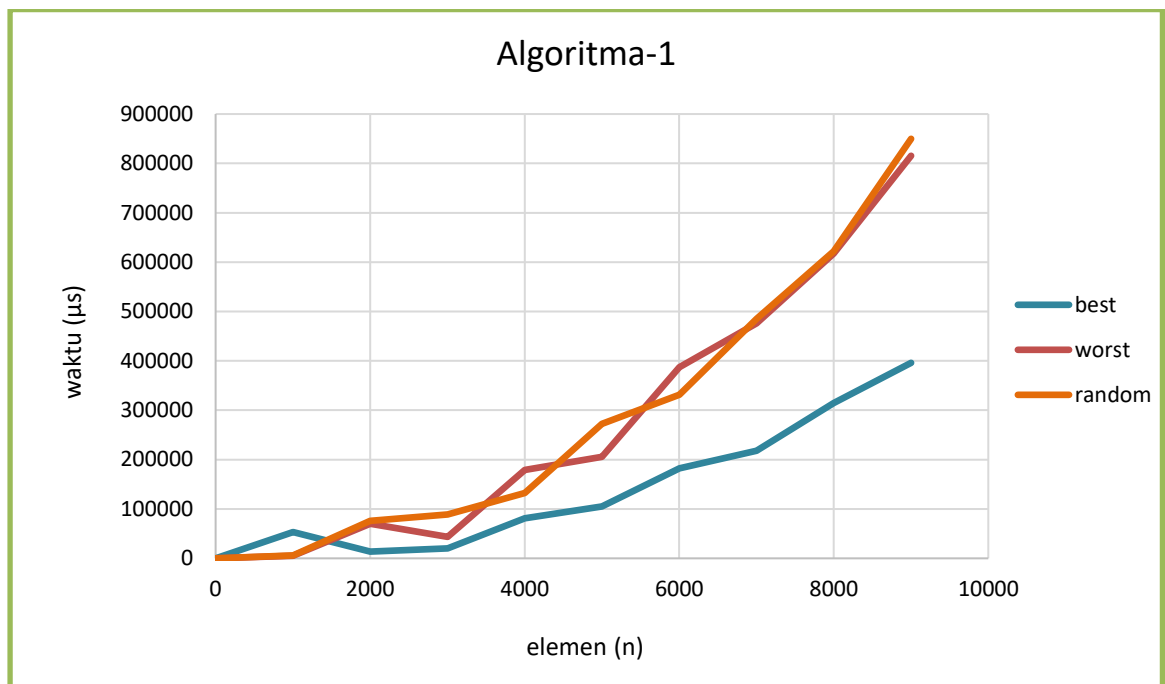
## Output Algoritma-1

```
❏ clang++-7 -pthread -std=c++17 -o main main.cpp
❏ ./main
Best case Algoritma-1
n = 1, duration: 0 microseconds
n = 1001, duration: 53536 microseconds
n = 2001, duration: 13393 microseconds
n = 3001, duration: 20132 microseconds
n = 4001, duration: 80870 microseconds
n = 5001, duration: 104942 microseconds
n = 6001, duration: 182499 microseconds
n = 7001, duration: 218025 microseconds
n = 8001, duration: 314491 microseconds
n = 9001, duration: 396064 microseconds

Worst case Algoritma-1
n = 1, duration: 0 microseconds
n = 1001, duration: 5439 microseconds
n = 2001, duration: 69801 microseconds
n = 3001, duration: 43714 microseconds
n = 4001, duration: 178818 microseconds
n = 5001, duration: 205442 microseconds
n = 6001, duration: 387420 microseconds
n = 7001, duration: 476154 microseconds
n = 8001, duration: 617078 microseconds
n = 9001, duration: 815419 microseconds

Random case Algoritma-1
n = 1, duration: 0 microseconds
n = 1001, duration: 5902 microseconds
n = 2001, duration: 76019 microseconds
n = 3001, duration: 88970 microseconds
n = 4001, duration: 132261 microseconds
n = 5001, duration: 272221 microseconds
n = 6001, duration: 331477 microseconds
n = 7001, duration: 485059 microseconds
n = 8001, duration: 621891 microseconds
n = 9001, duration: 849813 microseconds
```

### c. Grafik Algoritma-1



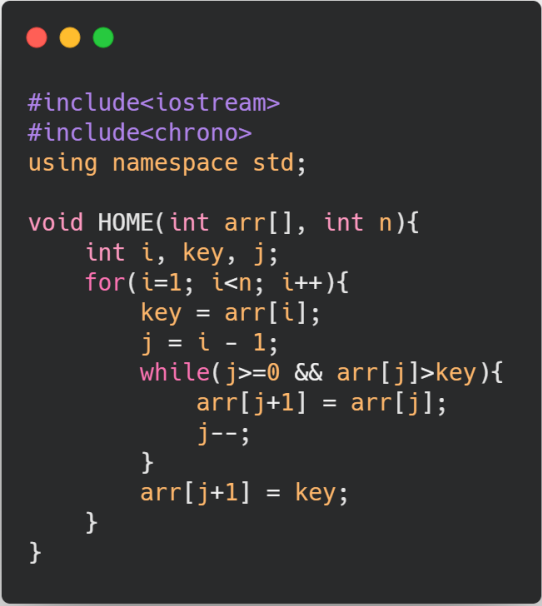
### d. Analisis Big-O

- Best Case =  $O(n)$ . Terjadi ketika array sudah diurutkan dari yang terkecil ke yang terbesar (ascending) sehingga tidak terjadi penukaran nilai (swap) dan hanya 1 iterasi dari  $n$  elemen.
- Worst Case =  $O(n^2)$ . Terjadi ketika array diurutkan terbalik (descending) sehingga terjadi penukaran nilai (swap) dan iterasi berlangsung dengan membandingkan  $n$  elemen, kemudian  $n-1$  elemen, dan seterusnya hingga hanya terjadi 1 perbandingan.  $(n + n - 1 + n - 2 + \dots + 1 = (n * (n + 1))/2 = O(n^2))$ .
- Random Case =  $O(n^2)$ . Terjadi ketika array disusun acak sehingga terjadi penukaran nilai (swap) dan iterasi berlangsung dengan membandingkan  $n$  elemen, kemudian  $n-1$  elemen, dan seterusnya.



## 2. Algoritma-2

Source code HOME function



```
#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;
    }
}
```

## Source code main function

```
int main (){

    cout << "Best case Algoritma-2\n";
    for(int n = 1; n <= 10000; n+=1000){
        int best[n];
        cout << "n = " << n << ", " ;
        for(int i = 0; i < n; i++){
            best[i] = i;
        }
        auto t0 = chrono::high_resolution_clock::now();
        HOME(best, n);
        auto t1 = chrono::high_resolution_clock::now();
        auto dt = t1 - t0;
        long long dtms = chrono::duration_cast<chrono::microseconds>(dt).count();
        cout << "duration: " << dtms << " microseconds" << endl;
    }

    cout << "\nWorst case Algoritma-2\n";
    for(int n = 1; n <= 10000; n+=1000){
        int worst[n];
        cout << "n = " << n << ", " ;
        for(int i = 0; i < n; i++){
            worst[i] = n-i;
        }
        auto t0 = chrono::high_resolution_clock::now();
        HOME(worst, n);
        auto t1 = chrono::high_resolution_clock::now();
        auto dt = t1 - t0;
        long long dtms = chrono::duration_cast<chrono::microseconds>(dt).count();
        cout << "duration: " << dtms << " microseconds" << endl;
    }

    cout << "\nRandom case Algoritma-2\n";
    for(int n = 1; n <= 10000; n+=1000){
        int random[n];
        cout << "n = " << n << ", " ;
        for(int i = 0; i < n; i++){
            random[i] = rand() % 100;
        }
        auto t0 = chrono::high_resolution_clock::now();
        HOME(random, n);
        auto t1 = chrono::high_resolution_clock::now();
        auto dt = t1 - t0;
        long long dtms = chrono::duration_cast<chrono::microseconds>(dt).count();
        cout << "duration: " << dtms << " microseconds" << endl;
    }

    return 0;
}
```

#### a. Analisis Source Code

```
1  #include<iostream>
2  #include<chrono>
```

Baris 1-2 merupakan *preprocessor directive* yang berfungsi untuk menyisipkan berkas lain ke dalam berkas yang ingin dikompilasi.

```
3  using namespace std;
```

Baris 3 digunakan untuk mengakses semua fungsi, class, atau file yang terdapat di dalam namespace 'std' tanpa harus menuliskan 'std::'

```
5  void HOME(int arr[], int n){
6      int i, key, j;
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     }
16 }
```

Baris 5-16 merupakan fungsi HOME yang mengurutkan nilai dengan membagi array menjadi 2 bagian, yakni bagian yang sudah diurutkan dan bagian yang belum diurutkan. Kemudian, nilai dari bagian yang belum diurutkan dibandingkan dengan posisi yang benar pada bagian yang sudah urut.

```
17 int main (){
```

Baris 17 adalah main function yang merupakan fungsi utama dari program C++ dengan nilai return berupa integer.

```
19     cout << "Best case Algoritma-2\n";
```

Baris 19 untuk mengeluarkan output berupa kalimat yang terletak di dalam tanda petik dua.

```
20     for(int n = 1; n <= 10000; n+=1000){
```

Baris 20 merupakan bentuk perulangan untuk n dimulai dari 1, lalu n terus ditambahkan dengan sebanyak 1000 hingga nilainya kurang dari atau sama dengan 10000.

```
21         int best[n];
```

Baris 21 mendeklarasikan array 'best' berisi n elemen.

```
22         cout << "n = " << n << ", " ;
```

Baris 22 mengeluarkan output berupa nilai yang terdapat pada variable n.

```

23     for(int i = 0; i < n; i++){
24         best[i] = i;
25     }

```

Baris 23-25 memuat perintah untuk mengassign array 'best' dengan nilai ascending

```

26     auto t0 = chrono::high_resolution_clock::now();

```

Baris 26 menyimpan waktu mulai dalam variable t0.

```

27     HOME(best, n);

```

Baris 27 memanggil fungsi HOME dengan best dan n sebagai argument.

```

28     auto t1 = chrono::high_resolution_clock::now();

```

Baris 28 menyimpan waktu selesai dalam variable t1.

```

29     auto dt = t1 - t0;

```

Baris 29 mendeklarasikan dt sebagai selisih dari t1 dan t0.

```

30 long long dtms = chrono::duration_cast<chrono::microseconds>(dt).count();

```

Baris 30 mendeklarasikan dtms sebagai durasi dari dt dengan satuan microsecond.

```

31     cout << "duration: " << dtms << " microseconds" << endl;

```

Baris 31 menampilkan nilai dtms.

```

36     int worst[n];

```

Baris 36 mendeklarasikan array 'worst' berisi n elemen.

```

38     for(int i = 0; i < n; i++){
39         worst[i] = n-i;
40     }

```

Baris 38-40 memuat perintah untuk mengassign array 'worst' dengan nilai descending.

```

42     HOME(worst, n);

```

Baris 42 memanggil fungsi HOME dengan worst dan n sebagai argument.

```

53     int random[n];

```

Baris 53 mendeklarasikan array 'random' berisi n elemen.

```

55     for(int i = 0; i < n; i++){
56         random[i] = rand() % 100;
57     }

```

Baris 55-57 memuat perintah untuk mengassign array 'random' dengan nilai random dalam rentang 100 nilai.

## **b. Analisis Program**

Pertama, akan dilakukan uji coba best case untuk algoritma-2, yakni dengan menggunakan 10 data integer ( $n = 1, 1001, 2001, \dots, 9001$ ) yang nilainya sudahurut dari yang terkecil hingga yang terbesar (ascending). Selanjutnya, dilakukan penyimpanan waktu mulai pada variable  $t_0$ , lalu fungsi HOME dijalankan dengan menggunakan 10 data ascending tadi. Pada fungsi HOME, terjadi pengurutan nilai dengan membagi array menjadi 2 bagian, yakni bagian yang sudah diurutkan dan bagian yang belum diurutkan. Kemudian, nilai dari bagian yang belum diurutkan dibandingkan dengan posisi yang benar pada bagian yang sudah urut. Setelah fungsi HOME selesai dieksekusi, dilakukan penyimpanan waktu selesai. Selanjutnya, dilakukan penghitungan waktu eksekusi dengan mencari selisih antara waktu mulai dengan waktu selesai sehingga didapatkan durasi jalannya Algoritma-2 pada saat best case dengan satuan microsecond pada masing-masing  $n$ .

Selanjutnya, program akan melakukan uji coba worst case untuk algoritma-2, yakni dengan menggunakan 10 data integer ( $n = 1, 1001, 2001, \dots, 9001$ ) yang nilainya sudahurut dari yang terbesar ke yang terkecil (descending). Selanjutnya, dilakukan penyimpanan waktu mulai pada variable  $t_0$ , lalu fungsi HOME dijalankan dengan menggunakan 10 data descending tadi. Setelah fungsi HOME selesai dieksekusi, dilakukan penyimpanan waktu selesai. Selanjutnya, dilakukan penghitungan waktu eksekusi dengan mencari selisih antara waktu mulai dengan waktu selesai sehingga didapatkan durasi jalannya Algoritma-2 pada saat worst case dengan satuan microsecond pada masing-masing  $n$ .

Terakhir, program akan melakukan uji coba random case untuk algoritma-2, yakni dengan menggunakan 10 data integer ( $n = 1, 1001, 2001, \dots, 9001$ ) yang nilainya acak. Selanjutnya, dilakukan penyimpanan waktu mulai pada variable  $t_0$ , lalu fungsi HOME dijalankan dengan menggunakan 10 data acak tadi. Setelah fungsi HOME selesai dieksekusi, dilakukan penyimpanan waktu selesai. Selanjutnya, dilakukan penghitungan waktu eksekusi dengan mencari selisih antara waktu mulai dengan waktu selesai sehingga didapatkan durasi jalannya Algoritma-2 pada saat random case dengan satuan microsecond pada masing-masing  $n$ .

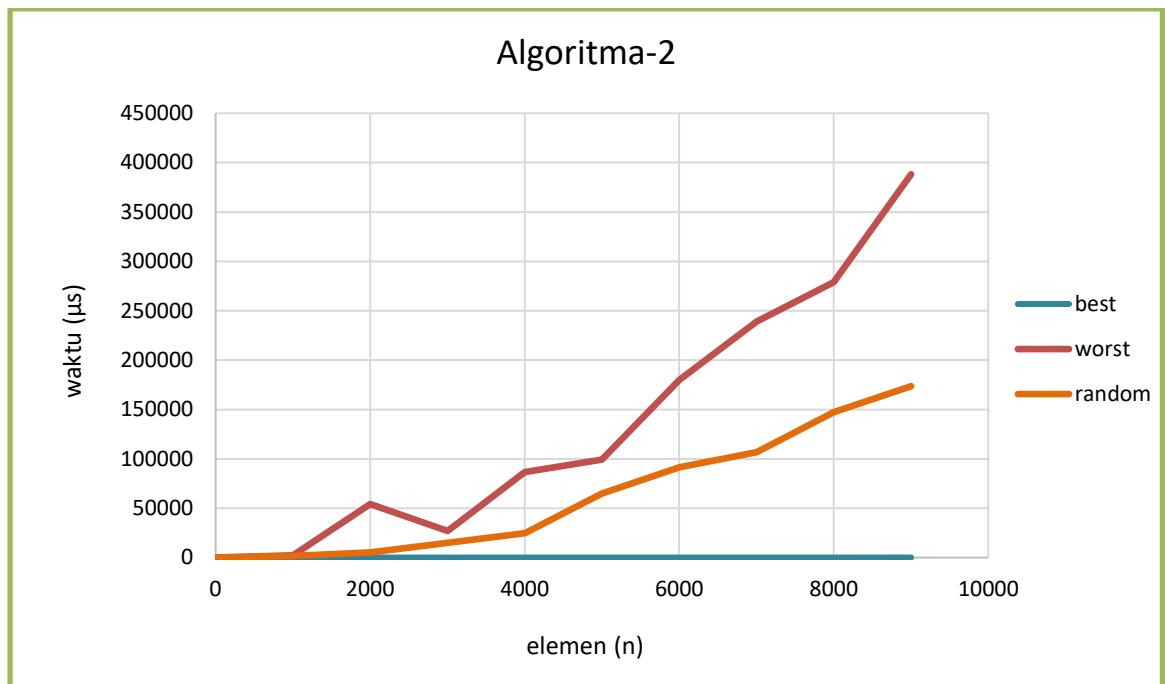
## Output

```
❏ clang++-7 -pthread -std=c++17 -o main main.cpp
❏ ./main
Best case Algoritma-2
n = 1, duration: 1 microseconds
n = 1001, duration: 7 microseconds
n = 2001, duration: 15 microseconds
n = 3001, duration: 41 microseconds
n = 4001, duration: 34 microseconds
n = 5001, duration: 41 microseconds
n = 6001, duration: 50 microseconds
n = 7001, duration: 64 microseconds
n = 8001, duration: 51 microseconds
n = 9001, duration: 69 microseconds

Worst case Algoritma-2
n = 1, duration: 0 microseconds
n = 1001, duration: 2444 microseconds
n = 2001, duration: 54117 microseconds
n = 3001, duration: 26816 microseconds
n = 4001, duration: 86761 microseconds
n = 5001, duration: 99222 microseconds
n = 6001, duration: 180031 microseconds
n = 7001, duration: 238991 microseconds
n = 8001, duration: 278982 microseconds
n = 9001, duration: 388231 microseconds

Random case Algoritma-2
n = 1, duration: 0 microseconds
n = 1001, duration: 1512 microseconds
n = 2001, duration: 5108 microseconds
n = 3001, duration: 15052 microseconds
n = 4001, duration: 24713 microseconds
n = 5001, duration: 64760 microseconds
n = 6001, duration: 91366 microseconds
n = 7001, duration: 106733 microseconds
n = 8001, duration: 147104 microseconds
n = 9001, duration: 173584 microseconds
```

c. **Grafik Algoritma-2**



d. **Analisis Big-O**

- Best Case =  $O(n)$ . Terjadi ketika array sudah diurutkan dari yang terkecil ke yang terbesar (ascending) sehingga tidak terjadi pergeseran nilai atau dapat kita katakan bahwa inner loop diabaikan sebab data sudah urut.
- Worst Case =  $O(n^2)$ . Terjadi ketika array diurutkan terbalik (descending) sehingga terjadi pergeseran nilai. Dalam hal ini inner loop dan outer loop digunakan untuk mengurutkan elemen tersebut.
- Random Case =  $O(n^2)$ . Terjadi ketika array disusun acak sehingga pergeseran nilai. Dalam hal ini inner loop dan outer loop digunakan untuk mengurutkan elemen tersebut.

### 3. Algoritma-3

Source code eraseAT function

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