**LAPORAN PRAKTIKUM**

**ANALISIS ALGORITMA**



**DISUSUN OLEH**

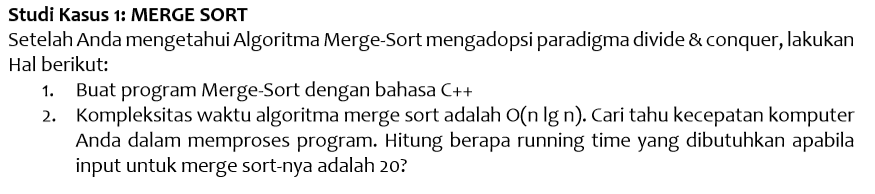
Gede Bagus Darmagita - 140810180068

**PROGRAM STUDI TEKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**2020**



Jawab:

/\*

    Nama    : Gede Bagus Darmagita

    NPM     : 140810180068

    Kelas   : B

\*/

#include <iostream>

#include <chrono>

using namespace std;

void Merge(int \*a, int low, int high, int mid)

{

    int i, j, k, temp[high - low + 1];

    i = low;

    k = 0;

    j = mid + 1;

    while (i <= mid && j <= high)

    {

        if (a[i] < a[j])

        {

            temp[k] = a[i];

            k++;

            i++;

        }

        else

        {

            temp[k] = a[j];

            k++;

            j++;

        }

    }

    while (i <= mid)

    {

        temp[k] = a[i];

        k++;

        i++;

    }

    while (j <= high)

    {

        temp[k] = a[j];

        k++;

        j++;

    }

    for (i = low; i <= high; i++)

    {

        a[i] = temp[i - low];

    }

}

void MergeSort(int \*a, int low, int high)

{

    int mid;

    if (low < high)

    {

        mid = (low + high) / 2;

        MergeSort(a, low, mid);

        MergeSort(a, mid + 1, high);

        Merge(a, low, high, mid);

    }

}

int main()

{

    int n, i;

    cout << "\nMasukkan Jumlah Data : ";

    cin >> n;

    int arr[n];

    for (i = 0; i < n; i++)

    {

        cout << "Masukkan elemen ke-" << i + 1 << ": ";

        cin >> arr[i];

    }

    auto start = chrono::steady\_clock::now();

    MergeSort(arr, 0, n - 1);

    auto end = chrono::steady\_clock::now();

    auto elapsed =

        chrono::duration\_cast<chrono::nanoseconds>(end - start);

    cout << "\nData Terurut: ";

    for (i = 0; i < n; i++)

        cout << " " << arr[i];

    cout << endl

         << "waktu yang dibutuhkan komputer : "

         << elapsed.count()

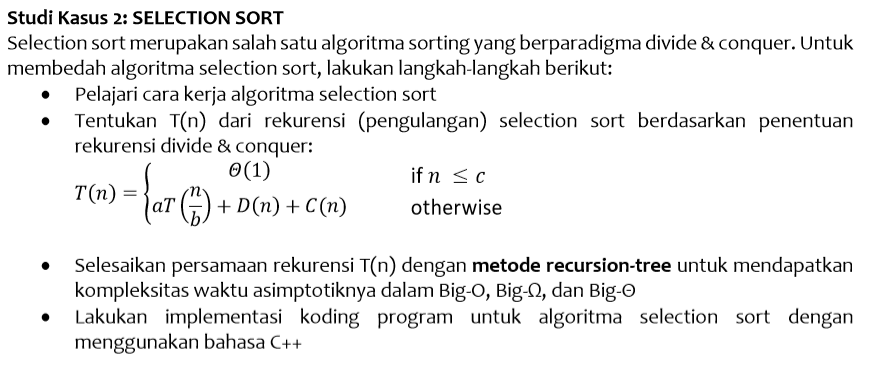
         << " nanosekon" << endl;

    return 0;

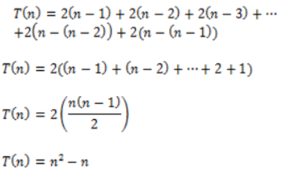
}

1. (0 nanosekon mungkin dikarnakan laptop saya yang terlalu cangih.)





Jawab:

1. Cara kerja  
   **Step 1** − Atur array indeks 0 (elemen pertama) sebagai lokasi/nilai minimum  
   **Step 2** − Cari elemen paling kecil yang ada di dalam list  
   **Step 3** − Tukar elemen terkecil tersebut ke lokasi/nilai minimum  
   **Step 4** − Atur elemen selanjutnya (sebelah kanannya) sebagai lokasi/nilai minimum  
   **Step 5** − Ulangi sampai list elemen-elemen kita berhasil terurut semua.
2. 
3. Big-O = Big-Ω = Big-θ = n2

/\*

    Nama    : Gede Bagus Darmagita

    NPM     : 140810180068

    Kelas   : B

\*/

#include <iostream>

using namespace std;

int main()

{

    int n, x[100], imaks, temp;

    cout << "Masukkan Jumlah Data : ";

    cin >> n;

    for (int i = 0; i < n; i++)

    {

        cout << "Bilangan ke - " << i + 1 << " : ";

        cin >> x[i];

    }

    cout << "\nSebelum di Sorting : ";

    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    cout << endl;

    for (int i = n - 1; i >= 1; i--)

    {

        imaks = 0;

        for (int j = 1; j <= i; j++)

        {

            if (x[j] > x[imaks])

                imaks = j;

        }

        temp = x[i];

        x[i] = x[imaks];

        x[imaks] = temp;

    }

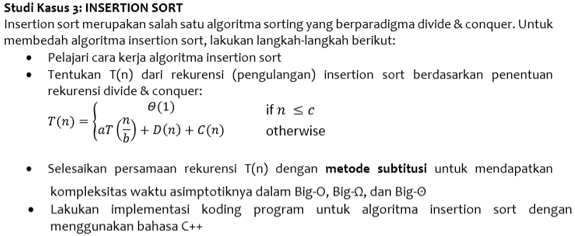
    cout << "Setelah di Sorting : ";

    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    return 0;

}



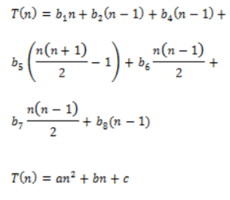
Jawab:

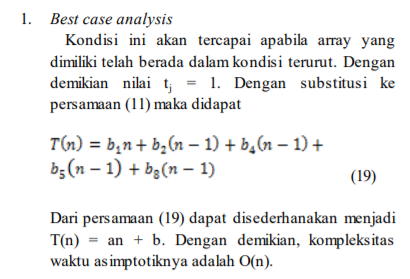
1. Cara Kerja  
   **Step 1** − Pengecekan mulai dari data ke-1 sampai data ke-n

**Step 2** − Bandingkan data ke-I ( I = data ke-2 s/d data ke-n )

**Step 3** − Bandingkan data ke-I tersebut dengan data sebelumnya(I-1), Jika lebih kecil maka data tersebut dapat disisipkanke data awal sesuai dgn posisisi yg seharusnya

**Step 4** − Lakukan langkah 2 dan 3 untuk bilangan berikutnya( I= I+1 ) sampai didapatkan urutan yg optimal.

1.   
   Worst case

 Best case

Nilai waktu asimtotiknya adalah O(n2)

1. Big-O = n

Big-Ω = Big-θ = n2

/\*

    Nama    : Gede Bagus Darmagita

    NPM     : 140810180068

    Kelas   : B

\*/

#include <iostream>

using namespace std;

int main()

{

    int n, x[100], j;

    int insert;

    cout << "Masukkan Jumlah Data : ";

    cin >> n;

    for (int i = 0; i < n; i++)

    {

        cout << "Bilangan ke - " << i + 1 << " : ";

        cin >> x[i];

    }

    cout << "\nSebelum di Sorting : ";

    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    cout << endl;

    for (int i = 1; i < n; i++)

    {

        insert = x[i];

        j = i - 1;

        while ((j >= 0) && (x[j] > insert))

        {

            x[j + 1] = x[j];

            j--;

        }

        x[j + 1] = insert;

    }

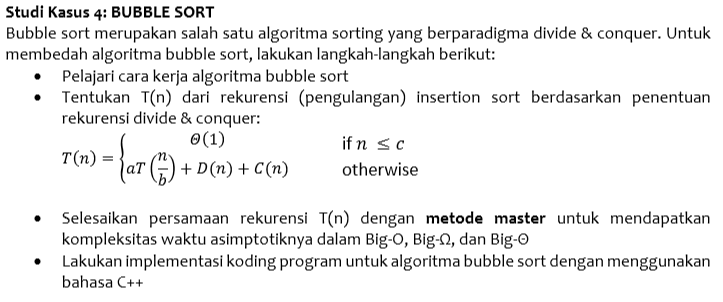
    cout << "Setelah di Sorting : ";

    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    return 0;

}



Jawab:

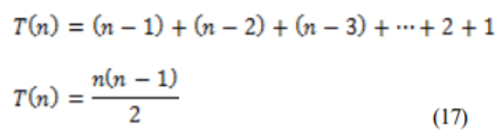
1. Cara kerja  
   **Step 1** − Pengecekan mulai dari data ke satu sampai data ke-n

**Step 2** − Bandingkan data ke-n dengan data sebelumnya (n-1)

**Step 3** − Jika lebih kecil maka pindahkan bilang tersebut dengan bilang yang ada didepannya (sebelumnya) satu persatu (n-1,n-2,n-3,….dts)

**Step 4** − Jika lebih besar maka tidak terjadi permindahan

**Step 5** − Ulang langkah 2 dan 3 s/d sort optimal.

1. 
2. Big-O = n

Big-Ω = Big-θ = n2

/\*

    Nama    : Gede Bagus Darmagita

    NPM     : 140810180068

    Kelas   : B

\*/

#include <iostream>

using namespace std;

int main()

{

    int arr[100], n, temp;

    cout << "\nMasukkan Jumlah Data : ";

    cin >> n;

    for (int i = 0; i < n; ++i)

    {

        cout << "Bilangan ke-" << i + 1 << " : ";

        cin >> arr[i];

    }

    for (int i = 1; i < n; i++)

    {

        for (int j = 0; j < (n - 1); j++)

        {

            if (arr[j] > arr[j + 1])

            {

                temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

            }

        }

    }

    cout << "\nOutput Bubble Sort : ";

    for (int i = 0; i < n; i++)

    {

        cout << " " << arr[i];

    }

    return 0;

}