Tugas 2 Praktikum Analisis Algoritma



Fikri Ikhsan

140810160016

Prodi S-1 Teknik Informatika

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Padjadjaran

2019

Studi Kasus 1 : Pencarian Nilai Maksimal

#include <iostream>

using namespace std;

int main()

{

    int n;

    int x[10];

    cout << "Masukkan Jumlah Data : ";

    cin >> n;

    for (int i = 0; i < n; i++)

    {

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

        cin >> x[i];

    }

    int maks = x[0];

    int i = 1;

    while (i <= n)

    {

        if (x[i] > maks)

            maks = x[i];

        i++;

    }

    cout << "Maksimum Number : " << maks << endl;

    return 0;

}

Studi Kasus 2 : Sequential Search

#include <iostream>

using namespace std;

int main()

{

    int n;

    int x[10];

    cout << "Masukkan Jumlah Data : ";

    cin >> n;

    for (int i = 0; i < n; i++)

    {

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

        cin >> x[i];

    }

    int y;

    cout << "Masukkan yang dicari : ";

    cin >> y;

    int i = 0;

    bool found = false;

    int idx;

    while ((i < n) && (!found))

    {

        if (x[i] == y)

            found = true;

        else

            i++;

    }

    if (found)

        idx = i+1;

    else

        idx = 0;

    cout << "Yang dicari berada di urutan : " << idx << endl;

    return 0;

}

Studi Kasus 3 : Binary Search

#include <iostream>

using namespace std;

int main()

{

    int n;

    int x[10];

    cout << "Masukkan Jumlah Data : ";

    cin >> n;

    for (int i = 0; i < n; i++)

    {

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

        cin >> x[i];

    }

    int y;

    cout << "Masukkan yang dicari : ";

    cin >> y;

    int i = 0;

    int j = n-1;

    bool found = false;

    int idx;

    int mid;

    while ((i <= j) && (!found))

    {

        mid = (i + j)/2;

        if (x[mid] == y)

            found = true;

        else

        {

            if (x[mid] < y)

                i = mid + 1;

            else

                j = mid - 1;

        }

    }

    if (found)

        idx = mid+1;

    else

        idx = 0;

    cout << "Yang dicari berada di urutan : " << idx << endl;

    return 0;

}

Studi kasus 4 : Insertion Sort

#include <iostream>

using namespace std;

int main()

{

    int n;

    int x[10];

    cout << "Masukkan Jumlah Data : ";

    cin >> n;

    for (int i = 0; i < n; i++)

    {

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

        cin >> x[i];

    }

    cout << "Data Sebelum di Sorting : ";

    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    cout << endl;

    int insert;

    int j;

    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 << "Data setelah di Sorting : ";

    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    return 0;

}

Kompleksitas waktu :

1. Best Case :

fori ß2 to n do 1 kali

insert ßxi n kali

j ßi n kali

x[j] = insert n kali

1. Average Case :

fori ß2 to n do 1 kali

insert ßxi n kali

j ßI n kali

x[j]ßx[j-1] n \* ½ n kali

jßj-1 n \* ½ n kali

x[j] = insert n kali

1. Worst Case :

fori ß2 to n do 1 kali

insert ßxi n kali

j ßi n kali

x[j]ßx[j-1] n \* n kali

jßj-1 n \* n kali

x[j] = insert n kali

Studi Kasus 5 : Selection sort

#include <iostream>

using namespace std;

int main()

{

    int n;

    int x[10];

    cout << "Masukkan Jumlah Data : ";

    cin >> n;

    for (int i = 0; i < n; i++)

    {

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

        cin >> x[i];

    }

    cout << "Data Sebelum di Sorting : ";

    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    cout << endl;

    int imaks;

    int temp;

    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 << "Data setelah di Sorting : ";

    for (int i = 0; i < n; i++)

        cout << x[i] << " ";

    return 0;

}

Operasi perbandingan dan operasi pertukaran

Jumlah operasi perbandingan element.

Untuk setiap pass ke-i,

         i = 1 –>  jumlah perbandingan  = n – 1

        i = 2 –>  jumlah perbandingan = n – 2

        i = 3  –> jumlah perbandingan = n – 3

:

        i = k –>  jumlah perbandingan = n – k

:

        i = n – 1  –> jumlah perbandingan = 1

Jumlah seluruh operasi perbandingan elemen-elemen larik adalah    T(n) = (n – 1) + (n –2) + … + 1

Ini adalah kompleksitas waktu untuk kasus terbaik dan terburuk, karena algoritma Urut tidak bergantung pada    batasan apakah data masukannya sudah terurut atau masih acak.

Jumlah operasi pertukaran

Untuk setiap i dari 1 sampai n – 1, ada satu kali pertukaran elemen, sehingga jumlah operasi pertukaran seluruhnya   adalah   T(n) = n – 1.

Jadi, algoritma pengurutan maksimum membutuhkan n(n – 1 )/2 buah operasi perbandingan elemen dan n – 1  buah operasi pertukaran.