**Tugas Kelas I**

**Analisis Algoritma**



dibuat oleh :

Angga Kresnabayu 140810160001  
Fauzi Faruq Nabbani : 140810160007

**UNIVERSITAS PADJADJARAN**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**2019**

1. Tulis algoritma dan program memangkatkan bilangan 2 pangkat n(integer) menggunakan algoritma iteratif dan rekursif

**Algoritma Rekursif Pangkat**

Teori dasarnya bahwa pangkat terdiri dari bilangan pangkat dan bilangan basis

Misal : 2^4

2 sebagai bilangan basis dan 4 sebagai bilangan pangkat

2^4 = 2 x 2 x 2 x 2

Bisa ditulis

2^4 = 2 \* 2 \* 2 \* 2 => 2^3\* 2

2^3 = 2 \* 2 \*2 => 2^2 \* 2

2^2 = 2\*2 => 2^1 \* 2

2^1 = 2 => 2^0 \* 2

2^0 = 1

Function

pangkat(x,y) // x sebagai bilangan basis dan y sebagai bilangan pangkat

if( y = 0) then // jika bilangan pangkat nya no maka pangkat disini menjadi 1

pangkat = 1

else

pangkat(x,y-1)\*x

// jika bukan y = 0 , maka y - 1. Akan terus melakukan dengan memanggil fungsi itu sendiri , hingga kondisinya berhenti pada y = 0.

Program (Javascript)

/\*

Nama : Fauzi Faruq Nabbani

Program : Rekursif Pangkat

Tanggal : 6 Maret 2019

\*/

var x,y;

var hasil = 1;

function pangkat(x,y){

  if (y === 0){

      return 1;

  } else {

          return pangkat(x,y-1)\*x;

      }

}

console.log(pangkat(2,4));

**Algoritma Iteratif Pangkat**

pangkat(x,y)

y sebagai bilangan pangkat di jadikan kondisi pada perulangan

Deklarasi hasil sebagai nilai untuk menyimpan nilai

Deklarasi  : x , y // x sebagai bilangan basis dan y sebagai bilangan pangkat

    : hasil // variabel untuk menyimpan hasil perpangkat

hasil = 1

for i = 1 to y do  // lakukan perulangan sampai i <= y

hasil = hasil\*x

endfor

write(hasil)

pangkat(2,4)

Hasil = 1

y = 1

Hasil = 1 \* 2

Hasil = 2

Y = 2

Hasil = 2 \*2

Hasil = 4

Y = 3

Hasil = 4 \* 2

Hasil = 8

Y = 4

Hasil = 8\*2

Hasil = 16

Program menggunakan Javascript

/\*

Nama : Fauzi Faruq Nabbani

Program : Iteratif Pangkat

Tanggal : 6 Maret 2019

\*/

var z = 2;

var c = 4;

var hasil = 1;

for(var i = 1 ; i<=c ; i++){

  hasil = hasil\*z;

}

console.log(hasil)

1. Buat algoritma dan program pencarian linear dan pencarian biner

**Pencarian Linear**

* Merupakan algoritma pencarian paling sederhana dan lebih mudah dipahami
* pencarian sekuensial dibuat berdasarkan semua item satu per satu. Setiap item diperiksa dan jika ada kecocokan maka item tertentu dikembalikan, jika tidak pencarian berlanjut sampai akhir pengumpulan data.

Algoritma

* ( Array A, Value x)
* Langkah 1 = deklarasi misal i = 1
* Langkah 2 = jika i > n ,di lanjutkan ke langkah 7
* Langkah 3 = jika array[i] = x dilanjutkan ke langkah 6
* Langkah 4 = set i = i + 1 (mengalami increment)
* Langkah 5 = kembali cek menggunakan langkah 2
* Langkah 6 = cetak element x ditemukan di index ke i
* Langkah 7 : cetak element tidak ditemukan
* Langkah 8 : selesai

Program menggunakan Javascript

/\*

Nama : Fauzi Faruq Nabbani

NPM : 140810160007

Program : Pencarian Linear

Tanggal : 6 Maret 2019

\*/

function cari(array,target){

  for (var i = 1 ; i < array.length ; i++){

      if ( array[i] == target ){

          return "Element  ditemukan :" + " " + array[i] + " di index ke " + i;

      }

  }

  return "Element tida ada dalam array"

}

const array = [2,3,4,5,9,10];

console.log(cari(array,10));

**Pencarian Biner**

Pencarian biner dapat dilakukan jika **data sudah dalam keadaan urut.** Dengan kata lain, apabila data belum dalam keadaan urut, pencarian biner tidak dapat dilakukan.

Kompleksitasnya O (log n)

Algoritma

Deklarasi :

Cari , Batas Bawah , Batas Atas , Tengah : sebagai integer

Array [1 …. N]

Ketemu : boolean

* Langkah 1 : input value yang akan di cari
* Langkah 2 : Menentukan batas bawah = 1
* Langkah 3 : dan batas atas adalah = N
* Langkah 4 : Cek ketemu = false
* Langkah 5 : cek dengan perulangan

while(Batas Atas < Batas Bawah) and (not ketemu) do

Tengah = (Batas Bawah + Batas Atas) dibagi 2

If array[tengah] = cari then

Ketemu = true

Else

if(array[tengah] < cari) then // mencari bagian kanan

Batas Atas = Tengah + 1

Else

Batas Bawah = Tengah - 1// Mencari di bagian kiri

Endif

Endif

Endwhile

If (ketemu) then

Output (“Data berada di index ke “, Tengah)

Else

Output (“Data tidak ditemukan’)

Endif

Program Menggunakan Javascript

/\*

Nama : Fauzi Faruq Nabbani

NPM : 140810160007

Program : Pencarian Biner

Tanggal : 6 Maret 2019

\*/

function pencarianBiner(array, target) {

  let kiri = 0;

  let kanan = array.length - 1;

  while (kiri <= kanan) {

      const tengah = kiri + Math.floor((kanan - kiri) / 2);

      if (array[tengah] === target) {

          return tengah;

      }

      if (array[tengah] < target) {

          kiri = tengah + 1;

      } else {

          kanan = tengah - 1;

      }

  }

  return -1;

}

const array= [1, 2, 3, 35, 67, 76, 88, 89]

console.log(pencarianBiner(array, 88))

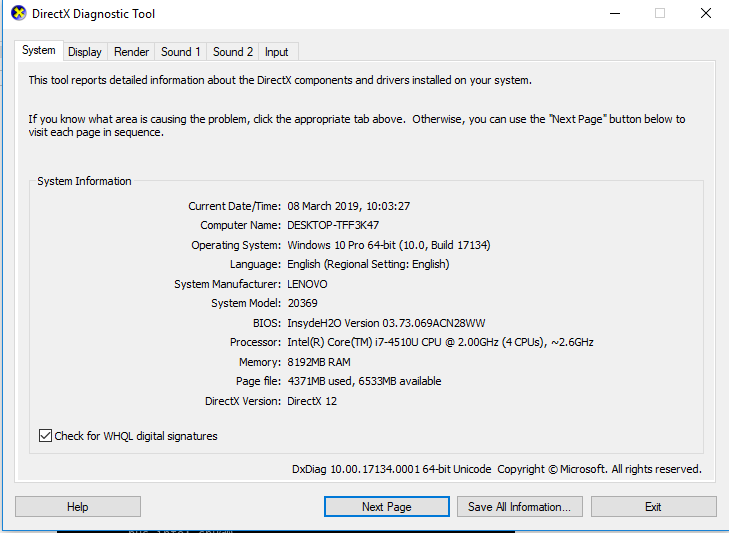
1. Analisis “running time” nya program-program tersebut dengan mencoba data yang berbeda dan bandingkan satu sama lain, tuliskan spesifikasi komputer yang digunakan, tuliskan kesimpulannya.

Untuk melakukan running time pada bahasa pemrograman javascript data menggunakan console.time() pada awal baris program dan menambhakan console.timeEnd() pada baris akhir program

Program ini dibuat pada spesifikasi Komputer sebagai berikut

Dual OS : Windows dan Linux Debian 9



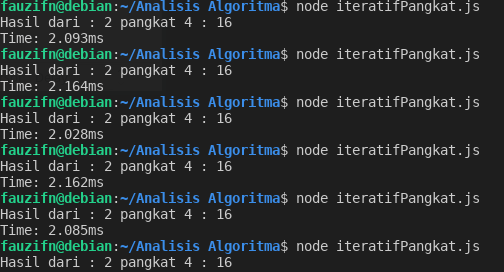


Running time selalu berubah-ubah bahkan untuk program dengan code dan data yang sama sekalipun. Running time dipengaruhi oleh kerja CPU dan kita tidak pernah tahu apa saja proses yang sedang dikerjakan oleh CPU kita.

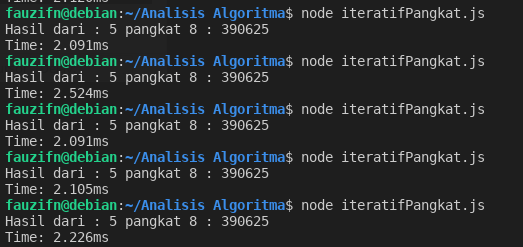
Beikut hasil running time pada program program yang telah dibuat

* 1. Pangkat Iteratif

Beberapa kali mencoba mamasukan nilai yang sama, hasil running nya seperti ini , memiliki nilai running yang tidak terlalu besar selisisnyanya

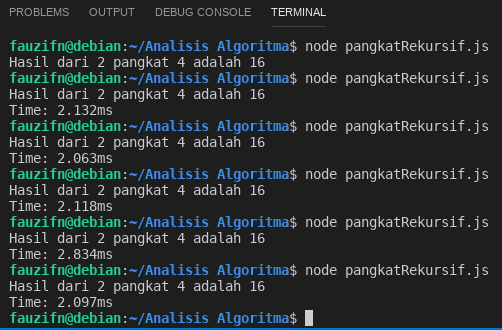


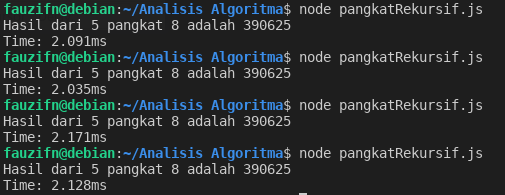
Dan ketika diberikan nilai berbeda :



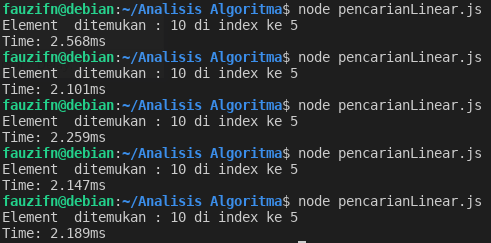
* 1. Pangkat Rekursif

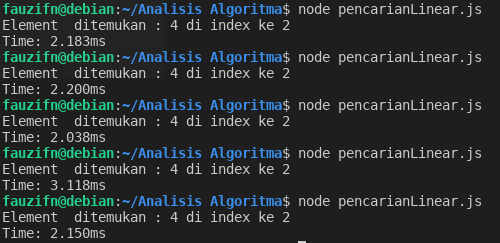
Sama seperti pada pangkata iteratif pun hasilnya seperti ini





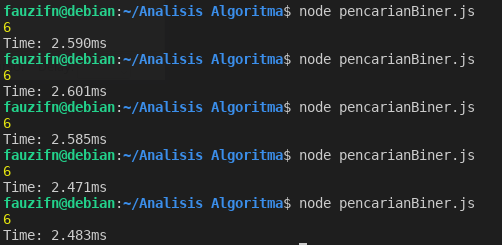
* 1. Pencarian Linear

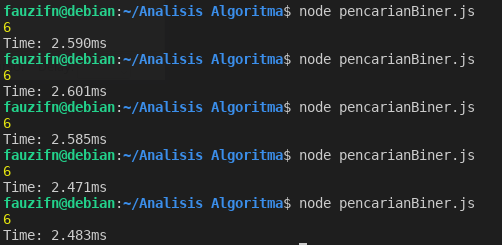




metode Rekursif lebih lambat dibandingkan dengan metode Looping. Karena jika dilihat dari segi kompleksitas algoritmanya, kedua metode ini memiliki kompleksitas yang berbeda.

* 1. Pencarian Biner





Binary Search membutuhkan waktu yang lebih lama dibandingkan Linear Search, dapat dikatakan bahwa kompleksitas waktu Linear Search lebih besar dibandingkan Binary Search.