Algoritma

Akhmad Sofwan
sofwan@sofwan.net
http://www.sofwan.net

Pendahuluan

Algoritma adalah urutan aksi-aksi yang di nyatakan dengan jelas dan tidak rancu untuk memecahkan suatu masalah dalam rentang waktu tertentu. Setiap aksi harus dapat di kerjakan dan mempunyai efek tertentu. Algoritma di temukan oleh Al-Khowarizmi, seorang ahli Matematika Persia yang hidup pada abad 9.

Tidak setiap masalah atau tugas pemrograman memiliki algoritma untuk memecahkan masalah tersebut. Masalah-masalah tersebut adalah:

1. Masalah yang tidak dapat di pecahkan.
2. Masalah yang sulit. Masalah ini memiliki algoritma yang panjang
3. Masalah yang tidak di ketahui algoritma nya.

BAB I
Penulisan Algoritma

I.1 Penulisan Algoritma

1. Menggunakan bahasa sehari-hari
   - Terlalu bertele-tele
   - Terlalu subyektif dipahami untuk setiap orang.

2. Menggunakan bahasa pemrograman formal.
   - Terlalu low level. Agak jauh dengan bahasa manusia.
   - Berhubungan dengan syntax pemrograman yang rumit.

3. Pseudo-code : Bahasa alami yang di design seperti pernyataan yang tersedia di beberapa bahasa pemrograman.

4. Flow Chart : Suatu bagan dengan notasi tertentu.

   Pada tutorial ini, penlisan algoritma tidak berdasarkan bahasa pemrograman tertentu, kecuali pada beberapa contoh script yang mempergunakan Java. Anda dapat mengimplementasikan algoritma dalam tutorial ini ke dalam bahasa pemrograman tertentu, seperti : PHP, JavaScript dan Python

I.2.Pseudo code

Terdapat 3 standard dalam penulisan Pseudo code.

1. Penomoran di dalam setiap instruksi. Untuk instruksi operasi sub ordinate atau operasi iterative, kita menggunakan dot natation, contoh : 3.1,3.2,3.3
   Setiap instruksi harus jelas dan tidak rancu.
   Lengkap. Tidak ada yang tertinggal.

Beberapa keyword yang di gunakan untuk menggambarkan input, output dan operasi processing.

Input : Read, Obtain, Get
Output : Print, Display, Show
Menghitung : compute, calculate, determine
Initialisasi : set, init
Tambah satu : increment, bump

I.3 Tiga kategori operasi Algoritma

1. Operasi Sequential → Instruksi yang di eksekusi secara berurut.
   Operasi Conditional → Sebuah struktur kontrol yang menguji kondisi tertentu dan mengeksekusi perintah tergantung keadaan kondisi nya.
   Contoh : For … Next.

2. Operasi Iterative (Looping) → Sebuah struktur kontrol yang mengulang sebuah blok eksekusi tertentu. Contoh : for .. next.

Contoh :

1. Operasi Sequential.

Contoh :

Di dalam operasi penjumlahan (Bahasa sehari-hari) :

1. Inisialisasi a,b dan c
2. Masukkan nilai a dan b

3. Jumlah kan a dan b ke dalam c
4. Tampilkan nilai c

(Bahasa Pseudo-code)
1. init a,b,c
2. read a,b
3. c = a + b
4. print c

BAB II
Struktur Kontrol dan Array

Struktur Kontrol adalah instrumen pengendali alur pemrograman, agar pemrograman dapat menghasilkan output yang diinginkan dari inputan yang tersedia, dapat berupa nilai sebuah variabel yang ditulis langsung di dalam sebuah bahasa pemrograman atau inputan dari keyboard. Array adalah variabel majemuk yang memiliki index dan berguna untuk membantu penyelesaian beberapa masalah pemrograman. Kita akan membahas array di bagian terakhir dalam bab ini

II.1. If-then-else

Merupakan salah satu bentuk conditional. Jika kondisi true, maka statement 1 di jalankan, jika false, maka statement 2 yang di jalankan.

Bentuk Umum :

```
IF kondisi THEN
    statement 1
ELSE
    statement 2
ENDIF
```

Jika kondisi bernilai true, maka statement 1 yang akan di eksekusi, selain itu, statement 2 yang akan di eksekusi.

Contoh :

```
1.  if a=2 then
1.1     print “A bernilai 2”
1.2 else
1.3     print “A Bukan bernilai 2”
```

II.2.While

Merupakan salah satu control looping / perulangan, yang selama kondisi terpenuhi, maka statement akan terpenuhi.

Bentuk Umum :

```
while   kondisi
    statement
endwhile
```

Contoh :

```
1. init   a
2. while   a<=5
3.      print “Looping ke “+a
4.      a++.
5.   endwhile
```

II.3. Do While

Merupakan salah satu control looping, yang selama kondisi terpenuhi, maka statement akan terpenuhi, minimal sekali :

Bentuk Umum :

```
while   kondisi
      statement
endwhile
```

Contoh :

```
1. init   a
2.do
3.      print “Looping ke “+a
4.      a++.
6.   while a<=5
```

II.4. Case

Adalah bentuk kondisi seperti If … End if, yang menjalankan statement tertentu
Bentuk Umum :

```
case (variabel)
      {
        statement 1
      }
 case (kondisi 2)
      {
        statement 2
      }
```

Misal :

```
init a=3
case (a)
      1 : print “Angka 1”
      break;
case (b)
      2 : print “Angka 2”
      break;
case ( c )
      3: print “Angka 3”
      break;
case (d)
       d: print “Angka 4”
default :
       print “Tidak ada nilai”
end case
```

II.5. For …. Next

Adalah struktur untuk melakukan perulangan statement tertentu.

Bentuk Umum :

```
for (var a;a < =angka;increment a)
      {
         statement 1
      }
```

Contoh :

```
init b;
for (b=1;b<=10;b++)
    {
        print “Nilai b :”+b
     }
```

II.6. Array

Adalah variabel yang memiliki indeks dan nilai tertentu pada setiap elemen nya.

Bentuk Umum :

```
array_a[4] = {“elemen_1”,”elemen_2”,”elemen_3”,”elemen_4”}
```

Contoh :

```
init a[5]={“senin”,”selasa”,”rabu”,”kamis”,”jum'at”}
```

Contoh penerapan dalam Java :

```
class array_1
    {
        public static void main (String args[])
          {
              int a[5]= {“Senin”,”Selasa”,”Rabu”,”Kamis”,”Jum'at”};
              int b=0;
              for (b=0;b<=4;b++)
               {
                   System.out.println(“Array :”+b+”=”+a[0]);
               }
          }
   }
```

BAB III
Sorting

Algoritma Sorting atau pengurutan adalah sebuah algoritma yang meletakkan element-element sebuah daftar di dalam urutan tertentu. Urutan yang paling banyak di gunakan adalah urutan numerik dan urutan lexicograpical. Sorting yang efisien adalah suatu hal yang penting untuk pengoptimalan penggunaan algoritma yang lain (seperti algoritma search dan merge) yang membutuhkan pengurutan daftar untuk dapat berfungsi dengan baik. Sorting juga berfungsi untuk menghasilkan report yang dapat di lihat dengan lebih baik.

Terdapat beberapa algoritma sorting, di antara nya adalah :
Bubble Sort
Selection Sort
Insertion Sort
Shell Sort
Comb Sort
Merge Sort
Heap Sort
Quick Sort
Counting Sort
Bucket Sort
Radix Sort
Distribution Sort

III.1 Bubble Sort

Bubble Sort adalah sebuah metode langsung dan simple dari pengurutan data. Ide dari Algoritma adalah mengulang proses pembandingan antara tiap-tiap elemen array dan menukar nya apabila urutannya salah. Perbandingan terjadi antara elemen yang terletak paling kiri dari list, di banding kan dengan elemen sebelah kanan nya / elemen kedua, jika elemen sebelah kanan nya lebih kecil, maka akan di tukar, jika tidak, tidak di tukar. Selanjut nya elemen yang kedua tadi, di bandingkan dengan elemen yang ketiga, dst. Proses tsb akan terus berlangsung, hingga tidak di perlukan lagi penukaran.

Algoritma ini termasuk dalam algoritma comparison sort, karena menggunakan perbandingan dalam operasi antar elemen nya.

Contoh :
Terdapat elemen array : 9 5 4 6 8 3

Pass Pertama :

9 5 4 6 8 3 → **5 9** 4 6 8 3
5 **9 4** 6 8 3 → 5 **4 9** 6 8 3
5 4 **9 6** 8 3 → 5 4 **6 9** 8 3
5 4 6 **9 8** 3 → 5 4 6 **8 9** 3
5 4 6 8 **9 3** → 5 4 6 8 **3 9**

Pass Kedua :
5 4 6 8 3 9 → **4 5** 6 8 3 9
4 **5 6** 8 3 9 → 4 **5 6** 8 3 9
4 5 **6 8** 3 9 → 4 5 **6 8** 3 9
4 5 6 **8 3** 9 → 4 5 6 **3 8** 9
4 5 6 3 **8 9** → 4 5 6 3 **8 9**

Pass Ketiga :
4 5 6 3 8 9 → **4 5** 6 3 8 9
4 **5 6** 3 8 9 → 4 **5 6** 3 8 9
4 5 **6 3** 8 9 → 4 5 **3 6** 8 9
4 5 3 **6 8** 9 → 4 5 3 **6 8** 9
4 5 3 6 **8 9** → 4 5 3 6 **8 9**

Pass Keempat :
4 5 3 6 8 9 → **4 5** 3 6 8 9
4 **5 3** 6 8 9 → 4 **3 5** 6 8 9
4 3 **5 6** 8 9 → 4 3 **5 6** 8 9
4 3 5 **6 8** 9 → 4 3 5 **6 8** 9
4 3 5 6 **8 9** → 4 3 5 6 **8 9**

Pass kelima :
4 3 5 6 8 9 → **3 4** 5 6 8 9
3 **4 5** 6 8 9 → 3 **4 5** 6 8 9
3 4 **5 6** 8 9 → 3 4 **5 6** 8 9
3 4 5 **6 8** 9 → 3 4 5 **6 8** 9
3 4 5 6 **8 9** → 3 4 5 6 **8 9**

Pass Keenam :
3 4 5 6 8 9 → **3 4** 5 6 8 9
3 **4 5** 6 8 9 → 3 **4 5** 6 8 9
3 4 **5 6** 8 9 → 3 4 **5 6** 8 9
3 4 5 **6 8** 9 → 3 4 5 **6 8** 9
3 4 5 6 **8 9** → 3 4 5 6 **8 9**

Pada pass kelima, langkah ke 2, urutan sudah sempurna, namun pass kelima tetap di teruskan hingga selesai dan pass keenam di jalankan karena definisi terurut dalam algoritma Bubble sort adalah, jika sebuah pass tidak ada satupun penukaran.

Kura-kura dan kelinci pada Bubble Sort
Pada list di atas, kita lihat, bahwa angka terbesar, 9 dengan cepat menempati urutan sebenar nya, yaitu pada pass pertama langkah kelima, sedangkan angka terkecil , 3 menempati urutan nya pada sesi akhir, yaitu pada pass kelima pada langkah pertama. Hal ini di istilahkan dengan Kura-kura dan Kelinci. Kura-kura adalah angka 3, sedangkan kelinci adalah angka 9. Fenomena ini mengakibatkan proses Bubble sort menjadi lama, terutama elemen kura-kura, yang membutuhkan 1 kali pass hanya untuk berpindah 1 kali.

Implementasi Dalam Bubble sort :

```
procedure BubbleSort(A : list of sortable items) define as :
 do
     swapped =false
          for each I in 0 to length (A) – 2
            inclusive do :
              if A[i] > A[i+1] then
                   swap (A[i], A[i+1])
                   swapped:=true
             end if
          end for
      while swapped
  end procedure
```

Kompleksitas Algoritma Bubble sort

Kompleksitas Algoritma ini dapat di lihat dari beberapa jenis kasus, yaitu :
Best-case, Worst-case dan Average-case

1. Kondisi Best-case
   Dalam kasus ini, data yang di sorting telah terurut, sehingga proses sorting hanya berlangsung selama 1 pass.
   Contoh : List : 1 2 3 4
   Pass Pertama :
   1 2 3 4 → **1 2** 3 4
   1 **2 3** 4 → 1 **2 3** 4
   1 2 **3 4** → 1 2 **3 4**

   Proses perbandingan hanya di lakukan sebanyak n-1 kali dengan nilai n adalah jumlah elemen list.

2. Kondisi Worst-case
   Contoh : 4 3 2 1
   Dalam contoh tsb, data terkecil berada pada ujung array. Setiap kali melakukan 1 pass, data terkecil akan bergeser ke arah awal sebanyak 1 step. Dengan kata lain, untuk menggeser data terkecil dari urutan keempat menuju urutan pertama, di butuhkan pass sebanyak 3 kali.

3. Kondisi Average Case
   Pada kondisi average case, jumlah pass di tentukan dari elemen mana yang mengalami pergeseran ke kiri paling banyak. Hal ini dapat di tunjukkan oleh proses pengurutan suatu array.

BAB IV
Algoritma Search

Algoritma Searching atau Algoritma pencarian adalah algoritma untuk mencari sebuah data atau angka di dalam sebuah kumpulan data, seperti di dalam array atau dalam sebuah database. Pada kesempatan kali ini, kita akan membahas Sequential Search dan Binary Search.

IV.1. Sequential Search

Sequential Search adalah teknik pencarian data di mana data di cari secara urut dari depan ke belakang atau dari awal sampai akhir. Kelebihan dari proses pencarian secara sequential ini, jika data yang di cari terletak di depan, maka akan dapat di temukan dengan cepat, namun jika data yang di cari ada di belakang, maka data akan lama di dapatkan, terlebih jika array data cukup panjang. Selain memakan waktu lebih lama, juga memakan sumber daya komputer yang cukup banyak, di banding jika data di awal list.

Sequential Search ini adalah algoritma search yang paling simpel.

Pseudocode :

```
1.  deklarasi array 1 dimensi
deklarasi cacah as int, ketemu=false
input nilai a
for (cacah=1 to cacah=a, cacah++)
4.1  if ( elemen_array==a)
4.2       ketemu=true
4.3       nilai_temu=elemen_array
4.3 end if
2.  next
```

Contoh Source sequential di Java (Bukan dari algoritma di atas) :

```
/**
 * Requires j2se 5.0 (version 1.5).   This uses class Scanner, static import,
 * and some methods of class Arrays.
 */

import java.util.Arrays;
import java.util.Scanner;
import static java.lang.Math.*;

public class Prog3c_1
{
     public static void main(String... args)
     {
          Scanner sc = new Scanner(System.in);

          int j, k, number, size;
          boolean found = false;

          System.out.print("Enter the array size: ");
          size = sc.nextInt();

          int[] intArray = new int[size];

          for (j=0; j<size; j++)
```

```
            intArray[j] = (int) (random()*100);

        System.out.println("A " + size + " element int array of random "
                           + "ints between 0 and 99 has been loaded.");
        System.out.print("Guess one of the numbers: ");
        number = sc.nextInt();

        k=0;
        while (!found && k<size)
        if (intArray[k]==number)
            found=true;
        else
            k +=1;

        System.out.println();
        if (found)
            System.out.println("Found in position " + k);
        else
            System.out.println("The number is not found");

        System.out.println("\nThe random array for verification is:\n");
        System.out.println(Arrays.toString(intArray));
    }
}
```

IV.2. Binary Search

Adalah pencarian secara biner, di gunakan ketika sebuah komputer harus mencari posisi sebuah simbol dalam daftar urut. Komputer akan mencari simbol dari tengah daftar sampai data terakhir dan membandingkannya dengan simbol yang sedang di cari. Apabila simbol tersebut sudah di temukan, pencarian pada setengah daftar sisa nya akan di hentikan.

Menemukan nilai tengah sering di kodekan sebagai :

mid = (high + low) / 2.

Algoritma :

1. Dapatkan nilai tengah

Jika nilai tengah sama dengan nilai yang di cari, algoritma berhenti.
Selain itu, 2 kasus yang mungkin :
- Nilai yang di cari lebih kecil dari element tengah. Dalam kasus ini, jalankan langkah 1 untuk bagian dari array, sebelum nilai tengah.
- Nilai yang di cari lebih besar dari nilai tengah. Dalam kasus ini, jalankan langkah 1 untuk bagian array setelah nilai tengah.

Iterasi akan berhenti jika :

1. Elemen yang di cari ketemu.

Ketika sub array tidak memiliki elemen lagi atau kosong. Kita simpulkan, bahwa elemen yang di cari tidak ada.

Contoh :

Temukan 4 di dalam {1,4,6,8,9,10,11,15}

Element tengah : 9

9 > 4

Maka : Sub array : { 1,4,6,8}

Element Tengah : 4

4 = 4.

Ketemu.

Algoritma Biner ini, membutuhkan array yang sudah terurut agar dapat bekerja dengan baik.

Contoh di dalam Java :

```
/**
 * searches for a value in sorted array
 *
 * @param array
 *          array to search in
 * @param value
 *          searched value
 * @param left
 *          index of left boundary
 * @param right
 *          index of right boundary
 * @return position of searched value, if it presents in the array or -1, if
 *         it is absent
 */
int binarySearch(int[] array, int value, int left, int right) {
    if (left > right)
        return -1;
    int middle = (left + right) / 2;
    if (array[middle] == value)
        return middle;
    else if (array[middle] > value)
        return binarySearch(array, value, left, middle - 1);
    else
        return binarySearch(array, value, middle + 1, right);
}
```

Referensi :

1. http://en.wikipedia.org/wiki/Search_algorithm
2. http://world-of-programmer.blogspot.com/2010/08/sequential-search-program-in-c.html
3. http://www.algolist.net/Algorithms/Binary_search
4. http://en.wikipedia.org/wiki/Sorting_algorithm . Tanggal akses 13 Desember 2010
5. Analisis Algoritma Bubble Sort, Ryan Rheinadi (Nim : 13508005), Teknik Informatika ITB

Biografi Penulis

Akhmad Sofwan *- Menyelesaikan pendidikan Sarjana di Teknik Informatika Universitas Budi Luhur ,tahun 2001 dan Magister Ilmu Komputer, Fakultas Ilmu Komputer ,Universitas Indonesia, tahun 2017. Penulis saat ini adalah seorang Freelance Software Developer. Selain itu, penulis juga mengajar mata kuliah Teknologi Informasi dan Kearsipan di Departemen Manajemen Rekod dan Arsip, Program Vokasi, Universitas Indonesia. Penulis juga aktif di Pusat Kajian Biostatistik dan Informatika Kesehatan (PKBIK) Fakultas Kesehatan Masyarakat, Universitas Indonesia sebagai Software Developer. Bidang penelitian beliau adalah Machine* Learning dan Data Mining.