

# **PENGANTAR PENCARIAN**

# **PENGANTAR PENCARIAN**

## **Masalah Pencarian**

Masalah pencarian adalah merupakan yang sering dijumpai oleh peneliti di bidang kecerdasan buatan. Permasalahan ini merupakan hal penting dalam memnentukan keberhasilan system kecerdasan buatan.

## **Algoritma Pencarian Beruntun**

Algoritma pencarian yang paling sederhana, yaitu metode Pencarian Beruntun *(Sequential Search)*. Nama lain algoritma Pencarian Beruntun adalah Pencarian Lurus *(Linear Search)*. Pada Dasarnya, Algoritma Pencarian Beruntun adalah proses membandingkan setiap elemen larik satu persatu secara beruntun, mulai dari elemen pertama, sampai elemen yang dicari ditemukan atau seluruh elemen sudah diperiksa.

Terdapat dua versi Algoritma Pencarian Beruntun. Pada Algoritma versi pertama, aksi pembandingan dilakukan sebagai kondisi pengulangan, sedangkan Algoritma versi kedua, aksi pembandingan dilakukan didalam badan pengulangan. Versi pertama tidak menggunakan peubah boolean dalam proses pencarian, sedangkan versi kedua menggunakan peubah boolean. Untuk masing-masing versi kita tuliskan dua macam algoritmanya berdasarkan hasil yang diinginkan : indeks larik atau nilai boolean. Selain itu, kita asumsikan jumlah elemen didalam larik adalah n buah. Berikut ini sintaknya :

## **Pencarian pada Larik Terstruktur dengan Pencarian Beruntun**

**Konsep :** membandingkan setiap elemen larik satu per satu secara urut (beruntun), mulai dari elemen pertama sampai dengan elemen yang terakhir Ada 2 macam pencarian beruntun,yaitu pencarian pada array yang sudah terurut, dan pencarian pada array yang belum terurut.

13 16 14 21 76 1 2 3 4 5 6

X = 21 Maka, elemen yang di cek adalah 13, 16, 14, 21 (ditemukan) Berarti data ditemukan pada indeks ke 3 elemen ke 4.

**Contoh :**

## **Pengantar Pengurutan**

Algoritma pengurutan *(sorting)* adalah :

* 1. Algoritma yang meletakkan elemen-elemen suatu kumpulan data dalam urutan tertentu atau,
	2. Prosees pengurutan data yang sebelumnya disusun secara acak sehingga menjadi tersusun secara teratur menurut suatu aturan tertentu.

Yang pada kenyataannya ‘urutan tertentu’ yang umum digunakan adala terurut secara numerikal ataupun secara leksikografi (urutan abjad sesuai kamus).

Ada 2 jenis *sorting* : ***Ascending & Descending***

## **Masalah Pengurutan**

Proses pengurutan seringkali menjadi bagian yang krusial dalam setiap pengolahan data sehingga mengapa ini merupakan topik yang penting. Contohnya dalam berbagai aplikasi dengan data ribuan bahkan jutaan, pengurutan bisa menghabiskan CPU-time paling banyak dari keseluruhan pengolahan data.

Masalah pengurutan itu sendiri selama ini memunculkan bervariasi algoritma. Yang akan dibahas disini adalah algoritma-algoritma yang memiliki keunikan tersendiri. Di luar algoritma-algoritma yang dibahas ada puluhan algoritma lain yang pada dasarnya variant-variant dari algoritma-algoritma yang akan dibahas berikut ini.

Mengikuti pembahasan buku teks yang digunakan, algoritma-algoritma tersebut dikategorikan ke dalam sejumlah kelompok :

Sebagai asumsi dasar dalam pembahasan selanjutnya tanpa menghilangkan sifat umumnya, pengurutan-pengurutan dilakukan menaik. Pengurutan menurun pada dasarnya sama kecuali pemeriksaan lojik adalah kebalikannya. Demikian halnya data yang diurutkan hanyalah array dari bilangan. Secara umum data dapat merupakan array dari suatu data record dari sejumlah field dan salah satunya digunakan kebagai key untuk pengurutan. Ragam dari field untuk pengurutan itu pun dapat pula berupa alfanumeris yang berimplikasi ekspresi-ekspresi lojik perlu disubstitusi oleh metoda-metoda lojik yang sesuai.

## **Algoritma Pengurutan Seleksi**

Ide algoritma ini amat sederhana. Mencari data terkecil dari data set yang belum terurut dan disusun dalam data set terurut. Tentu, data set terurut pada mulanya kosong. Setelah satu demi satu data disusun maka akan terbentuk keterurutan tersebut.

Dalam pengurutan ini kedua data set (terurut maupun yang belum) berada dalam array yang sama. Misalnya array tersebut adalah X, maka pada setiap saat terdapat i buah data terurut pada X[0], X[1], ..., X[i-1], dan data tak terurut pada X[i], X[i+1], ..., X[n-1]. Algoritma melakukan pencarian X[j] terkecil dari data set yang belum terurut tersebut, misalnya didapat X[m] lalu melakukan penukaran X[i] dengan X[m] sehingga kemudian sudah terurut i+1 buat data dalam X.

**ALGORITMA**

for (i=0; i < n-2; i++) {

 m = i;

 for (j = m+1; j < n-1; j++) {

 if (X[m] > X[j]) m = j;

 }

 if (m != i) {

 t = X[m]; X[m] = X[i]; X[i] = t;

 }

}

Dalam semua kondisi algoritma ini tepat akan melakukan iterasi luar sebanyak n-1 dan iterasi dalam sebanyak (n-i). Jumlah operasi "m = j;" bergantung pada kondisi data. Average case adalah setengah dari jumlah iterasi dalam yang dilakukan pada setiap iterasi luar, yang menghasilkan jumlah total 1/4 n2) operasi tersebut. Kompleksitas dari algoritma ini O(n2).

## **Pengurutan pada Larik Terstruktur dengan Pengurutan Seleksi**

Metode ini memilih elemen maksimum/minimum dari larik, lalu menempatkan elemen itu pada awal atau akhir larik (elemen terujung). Selanjutnya elemen terujung tersebut ‘diisolasi’ dan tidak disertakan pada proses selanjutnya. Proses yang sama diulang untuk elemen larik yang tersisa, yaitu memilih elemen maksimum/minimum berikutnya danmempertukarkannya dengan elemen terujung larik sisa. Dua varian algoritma pengurutan pilih ditinjau dari pemilihan elemen maksimum/minimum, yaitu :

* + - 1. **Pengurutan Pilih Maksimum**

Misalkan larik L dengan N=6 sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 29 | 27 | 10 |  | 8 | 76 | 21 |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 |  |  |  |
| maka langkah-langkah pengurutannya adalah: |  |
|  |
| Semula | 29 | 27 | 10 | 8 | 76 | 21 |  |
| Iterasi 1 | 29 | 27 | 10 | 8 | 21 | 76 |  |
| Iterasi 2 | 21 | 27 | 10 | 8 | 29 | 76 |  |
| Iterasi 3 | 21 | 8 | 10 | 27 | 29 | 76 |  |
| Iterasi 4 | 10 | 8 | 21 | 27 | 29 | 76 |  |
| Iterasi 5 | 8 | 10 | 21 | 27 | 29 | 76 | terurut |

**Contoh Algoritma :**

PROCEDURE

 SelectionSort(input/output L : Larikint, input N : integer)

{ *mengurutkan elemen larik L[1..N] sehingga tersusun menaik*

 *dengan* *metode pengurutan maksimum }*

*{ kondisi awal : elemen larik L sudah terdefinisi harganya }*

*{ kondisi akhir : elemen larik L terurut menaik sedemikian*

 *sehingga L[1]* ≤ *L[2]* ≤ *...* ≤ *L[N]* }

**Contoh Algoritma :**



* + - 1. **Pengurutan Pilih Minimum**

Pada algoritma pengurutan pilih minimum, basis pencarian adalah elemen minimum (terkecil). Elemen minimum ditempatkan di awal larik (agar larik terurut menaik) atau ditempatkan di akhir larik (agar larik terurut menurun).

Misalkan larik L dengan N=6 sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 29 | 27 | 10 |  | 8 | 76 | 21 |  |  |  |
| 1 | 2 | 3 | 4 | 5 | 6 |  |  |  |
| maka langkah-langkah pengurutannya adalah: |  |
|  |
| Semula | 29 | 27 | 10 | 8 | 76 | 21 |  |
| Iterasi 1 | 8 | 27 | 10 | 29 | 76 | 21 |  |
| Iterasi 2 | 8 | 10 | 27 | 29 | 76 | 21 |  |
| Iterasi 3 | 8 | 10 | 21 | 29 | 76 | 27 |  |
| Iterasi 4 | 8 | 10 | 21 | 27 | 76 | 29 |  |
| Iterasi 5 | 8 | 10 | 21 | 27 | 29 | 76 | terurut |



****

Dibandingkan dengan metode pengurutan gelembung, metode pengurutan pilih memiliki kinerja yang lebih baik. Alasannya, operasi pertukaran elemen hanya dilakukan sekali saja dengan demikian lama pengurutannya berkurang.

* + - 1. **Metode Pengurutan Sisip (*Insertion Sort*)**

Algoritma pengurutan sisip untuk memperoleh elemen larik yang terurut menaik adalah sebagai berikut :

Misalkan larik L dengan N=6 sebagai berikut:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 29 | 27 | 10 |  | 8 | 76 | 21 |  |  |  |
| 1 | 2 | 3 |  | 4 | 5 | 6 |  |  |  |
| maka langkah-langkah pengurutannya adalah: |  |
|  |
| Semula | 29 |  | 27 | 10 | 8 | 76 | 21 |  |
| Iterasi 1 | 29 |  | 27 | 10 | 8 | 76 | 21 |  |
| Iterasi 2 | 27 |  | 29 | 10 | 8 | 76 | 21 |  |
| Iterasi 3 | 10 |  | 27 | 29 | 8 | 76 | 21 |  |
| Iterasi 4 | 8 |  | 10 | 27 | 29 | 76 | 21 |  |
| Iterasi 5 | 8 |  | 10 | 27 | 29 | 76 | 21 |  |
| Iterasi 6 | 8 |  | 10 | 21 | 27 | 29 | 76 | terurut |





Kelemahan metode pengurutan sisip terletak pada banyaknya operasi pergeseran yang diperlukan dalam mencari posisi yang tepat untuk elemen larik. Untuk larik dengan N yang besar, jumlah operasi pergeseran meningkat secara kuadratik, sehingga pengurutan sisip kurang bagus untuk volume data yang besar.

# **DAFTAR PUSTAKA**

Rinaldi Munir. Algoritma dan Pemrograman dalam Bahasa Pascal dan C Edisi Revisi. Informatika, 2007.

Rosa A.S. Algoritma dan Pemrograman,, Modula, 2010