*ALTERNATIF*

*METODE PENGURUTAN DATA*

*Edson Yahuda Putra*

**Abstrak**

Seorang programmer yang sudah berpengalaman tentunya akan memikirkan juga masalah efisiensi penggunaan suatu algoritma. Suatu program bisa dibikin dari beberapa macam algoritma yang berbeda. Hal itu tidak menjadi masalah dalam sebuah pemrograman komputer, karena suatu sistem, bagi pemakai (user) yang penting adalah memasukkan data benar dan hasil keluarannya juga benar. Masalah prosesnya urusan komputer yang mengaturnya.

Pemilihan algoritma yang tepat sangat mendukung kecepatan akses suatu program aplikasi yang dibuat. Dengan semakin cepat program diakses tentunya akan lebih efisien.

Artikel ini menunjukkan bagaimana pemilihan struktur data terhadap sejumlah data yang diberikan mempengaruhi algoritma-algoritma yang digunakan untuk menyelesaikan suatu persoalan. Pengaturan data juga merupakan contoh yang baik untuk menunjukkan bahwa suatu persoalan bisa diselesaikan dengan sejumlah algoritma yang berbeda sati sama lain. Berikut ini akan dibahas beberapa alternatif metode pengurutan data.

**Keywords:** Insertion Sort, Selection Sort, Bubble sort, Algoritma, Prosedur.

Pengaturan data (***sorting***) secara umum bisa didefinisikan sebagai suatu proses untuk menyusun kembali himpunan objek menggunakan aturan tertentu. Secara umum pengurutan data dibedakan atas 2 junis. Yang pertama adalah pengurutan data secara naik (***ascending***), yaitu mengurutkan data dari kecil ke besar. Sebaliknya yang kedua adalah mengurutkan data secara urut turun (***descending***), yaitu mengurutkan data dari besar ke kecil (*Santoso,* 1993). Pengurutan data ini menggunakan data yang bertipe ***char*** (karakter) atau string, dimana nilai data dikatakan oleh sebuah standar internasional yang disebut dengan ASCII.

**Tujuan Pengurutan Data**

Tujuan pengurutan data adalah untuk mempermudah dalam pencairan kembali (*Santoso,* 1993). Bisa dibayangkan seandainya sebuah tumpukan kartu resmi yang tidak urut dan kita akan mencari satu kartu diantaranya. Atau, jika 1000 nama mahasiswa, tentu akan lama dan membosankan. Sebaliknya jika sudah diurutkan maka akan lebih mudah untuk mencarinya. Contoh data yang sudah urut penataannya adalah kamus, dan buku telepon.

Demikian juga sebuah Komputer yang mempunyai data sekian banyak, dan komputer disuruh untuk mencari satu kata kunci, komputer akan membutuhkan waktu yang lama untuk mencari jika data-data tersebut belum diurutkan.

Selain untuk memudahkan dalam pencairan, tujuan pengaturan data adalah untuk memudahkan dalam editing, yaitu penghapusan (***delete***), penyisipan (***insert***), juga penggabungan (***joint***).

**Pengurutan Larik (Array Sorting)**

Oleh karena itu pengurutan larik memanfaatkan memory utama (RAM) dalam proses pengurutannya, maka perlu adanya pertimbangan aspek ekonomis. Artinya, berapa persen dari kapasitas RAM yang dipakai untuk peroses pengurutan data, dan berapa lama waktu yang dibutuhkan oleh RAM untuk memproses pengurutan tersebut. Karena ada beberapa tugas yang lain lagi yang akan RAM laksanakan setelah tugas pencarian selesai.

Ukuran efisiensi yan baik bisa diperoleh dari banyaknya perbandingan (C) dan perpindahan (M) yang harus dilakukan. Deretan angka yang akan diurutkan disebut dengan vektor. Angka-angka ini merupakan fungso dari N yaitu banyaknya elemen yang akan diurutkan. Algoritma yang baik memerlukan pembandingan sebanyak N log N kali.

Ada beberapa metode pengurutan data yang akan dibahas di artikel ini. Tentu saja banyak metode yang ada, dan kita hanya mengambil beberapa contoh saja, yaitu penyisipan langsung (***insertion sort***), pengurutan seleksi (***selection sort***) dan metode gelembung (***Bubble sort***).

1. **Metode Penyisipan Langsung (*insertion sort*)**

Metode penyisipan langsung banyak digunakan oleh pemain kartu. Dalam metide ini elemen-elemen (kartu) terbagi menjadi dua kelompok. Kelompok pertama, yaitu kelompok tujuan (kartu yang ada di tangan pemain yang sudah dalam keadaan urut) dengan urutan A1 ... AI-1. Kelompok kedua disebut dengan sumber (yaitu kartu yang masih di meja, dan masih belum dalam kondisi belum urut). Dengan urutan AI ... AN. Dalam setiap langkah, dimulai dari I = 2, dengan pertambahan (step) = 1, elemen ke I diambil dari kelompok sumber akan dipindahkan ke kelompok tujuan dengan cara menyisipkannya pada tempatnya yang sesuai.

Dari ilustrasi di atas bisa dilihat bahwa jika suatu elemen ke I akan disisipkan ke suatu tempat, misalnya pada posisi ke J, maka perlu dilakukan penggeseran ke kanan (ke subkrib yang lebih besar). Berikut adalah contoh algoritma pengurutan dengan penyisipan langsung.

Algoritma penyisipan langsung

*Langkah 0 = Baca vektor yang akan diurutkan (dalam program utama)*

*Langkah 1 = kerjakan langkah 2 sampai langkah 5 untuk 1 = 2 sampai N.*

*Langkah 2 = tentukan T = A [i] (elemen yang akan disisipkan),*

*A[0] = T (data sentinel), dan*

*J = I – 1.*

*Langkah 3 = (lakukan penggeseran)*

*Lakukan langkah 4 selama T > A[J]*

*Langkah 4 = Tentukan : A[J+1] = A[J], dan*

*J = J – 1*

*Langkah 5 = Tentukan : A[J+1] = T.*

*Langkah 6 = Selesai.*

Prosedur untuk mengurutkan elemen vektor dengan metode penyisipan langsung

*Procedure SISIP\_LANGSUNG(Var A : Larik; N; Integer);*

*Var I, J : Interger ; T : real;*

*Begin*

*For I : = 2 to N do*

*Begin*

*T : = A[I] ; J : = I – 1;*

*A [0] : = T; {\*data sentinel\*}*

*While T < A[J] do*

*{\*Selama elemen sebelah kiri data yang ditinjau\**

*\* lebih kecil dari data yang akan disisipkan, geser ke kiri\*}*

*Begin*

*A[J+1] ;= A [J];*

*Dec(J)*

*End;*

*A[J+1] ; =*

*End;*

*End;*

Masukan dinyatakan dengan vektor A dalam kondisi belum turut, dan N adalah banyaknya elemen. Keluaran adalah vektor A yang sudah dalam kedaan terurut.

Contoh, jika kita mempunyai data sebanyak 9 buah data dalam keadaan acak dan kita akan mengurutkan berdasarkan metode ini, kita bisa melihat di setiap iterasinya seperti berikut.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Itera si | A[ 0] | A[ 1] | A[ 2] | A[ 3] | A[ 4] | A[ 5] | A[ 6] | A[ 7] | A[ 8] | A[ 9] |
| I = 1 | 0 | 7 | 10 | 3 | 5 | 1 | 2 | 6 | 14 | 8 |
| I = 2 | **10** | 7 | 10 | 3 | 5 | 1 | 2 | 6 | 14 | 8 |
| I = 3 | **3** | 3 | 7 | 10 | 5 | 1 | 2 | 6 | 14 | 8 |
| I = 4 | **5** | 3 | 5 | 7 | 10 | 1 | 2 | 6 | 14 | 8 |
| I = 5 | **1** | 1 | 3 | 5 | 7 | 10 | 2 | 6 | 14 | 8 |
| I = 6 | **2** | 1 | 2 | 3 | 5 | 7 | 10 | 6 | 14 | 8 |
| I = 7 | **6** | 1 | 2 | 3 | 5 | 6 | 7 | 10 | 14 | 8 |
| I = 8 | **14** | 1 | 2 | 3 | 5 | 6 | 7 | 10 | 14 | 8 |
| I = 9 | **8** | 1 | 2 | 3 | 5 | 6 | 7 | 8 | 10 | 14 |

Algoritma diatas mempunyai jumlah perbandingan (C) untuk nilai I tertentu sebanyak I – 1 kali, dengan paling sedikit satu kali, dan rata-rata sebesar I/2 kali. Sedangkan banyaknya pemindahan atau penggeseran (M) untuk nilai I tertenru adalah C1 + 2 (termasuk sentinel). Banyaknya perbandingan dan pemindahan akan minimum jika elemen-elemen semua sudah dalam keadaan urut tetapi secara urut turun (sedang tujuan kita adalah urut naik).

1. **Pengukuran Seleksi (*Selection Sort*)**

Metode ini menggunakan cara kerja pada pencarian elemen dalam vektor dengan nilai terkecil, setelah ketemu dilakukan penukaran dengan elemen ke I. Secara singkat dijelaskan sebagai berikut. Pada langkah pertama, dicari data yang terkecil tersebut kita tukar dengan data pertama. Dengan demikian, data pertama sekarang mempunyai nilai paling kecil dibandingkan data yang lain. Pada langkah kedua, data terkecil kita cari mulai data kedua sampai data terakhir. Data terkecil yang kita peroleh ditukar dengan data kedua. Demikian seterusnya sampau seluruh vektor dalam keadaan urut. Untuk lebih memperjelas proses pengurutan dengan metode seleksi, berikut disajikan contoh metode ini.

Algoritma seleksi

*Langkah 0 = Baca vektor yang diurutkan (dalam program utama)*

*Langkah 1 = Kerjakan langkah a2 sampai 4 unutk I = 1 samapai N-1*

*Langkah 2 = Tentukan Lok = I*

*Kerjakan langkah 3 untuk J = I + 1 sampai N*

*Langkah 3 = Mencari data terkecil*

*Test : Apakah A [Lok] > A [J] ?*

*Jika Ya, tentukan : Lok = J*

*Langkah 4 = Tukarkan nilai A [Lok] dengan A [J]*

*Langkah 5 = Selesai*

Prosedur untuk pengurutan menggunakan metode seleksi ini bisa dilihat seperti berikut:

*Procedure Seleksi (var A : Larik; N : Integer);*

*Var I, J, Lok : Integer;*

*Begin*

*For I : = 1 to N – 1 do*

*Begin*

*{\*Lokasi elemen terkecil\*}*

*Lok : = I;*

*{\*Mencari elemen terkecil dan mencari posisinya\*}*

*For J : = I + 1 to N do*

*If A[Lok] > A[J] then*

*{\*Lokasi elemen pad lokasi ke I\**

*\*dengan elemen pada lokasi ke Lok\*}*

*TUKARAN (A[I], A[Lok])*

*End;*

*End;*

*Procedure TUKARKAN ( var A, B : real);*

*Var T : real;*

*Begin*

*T : = A;*

*A : = B;*

*B : = T*

*End;*

Misalkan kita mempunyai 9 buah angka yang akan diurutkan menggunakan metode seleksi. Kita bisa melihat bagaimana tiap iterasi program dieksekusi, seperti berikut ini.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Iterasi Ke | A[1] | A[2] | A[3] | A[4] | A[5] | A[6] | A[7] | A[8] | A[9] |
| I = 1, Lok = 4 | 3 | 4 | 7 | **1** | 6 | 2 | 11 | 20 | 5 |
| I = 2, Lok = 6 | 1 | 4 | 7 | 3 | 6 | **2** | 11 | 20 | 5 |
| I = 3, Lok = 4 | 1 | 2 | 7 | 3 | 6 | 4 | 11 | 20 | 5 |
| I = 4, Lok = 6 | 1 | 2 | 3 | 7 | 6 | **4** | 11 | 20 | 5 |
| I = 5, Lok = 9 | 1 | 2 | 3 | 4 | 6 | 7 | 11 | 20 | **5** |
| I = 6, Lok = 9 | 1 | 2 | 3 | 4 | 5 | 7 | 11 | 20 | **6** |
| I = 7, Lok = 7 | 1 | 2 | 3 | 4 | 5 | 6 | 11 | 20 | **7** |
| I = 8, Lok = 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 20 | **11** |
| Akhir | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 11 | 20 |

Metode ini secara garis besar merupakan kebalikkan dari metode penyisipan langsung. Dalam setiapp langkah pada metode penyisipan langsung kira hanya memperhatikan satu elemen dari sumber dan semua elemen dari larik tujuan untuk menentukan posisinya yang tepat; sehingga sering disebut dengan ***one source-multiple destinations.*** Dalam metode seleksi terjadi sebaliknya, yanki kita memperhatikan semua elemen dalam larik sumber untuk menentukan elemen terkecil yang akan kita tempatkan pada tujuan; sehingga sering disebut dengan ***multi source one destination.*** Dalam metode ini banyaknya pembandingan (untuk mencari elemen dengan nilai terkecil) besarnya tak berpengaruh terhadap urutan semula. Banyaknya pembandingan adalah sebesar

C= (N2 – N) / 2

Metode ini tidak dilakukan penggeseran (M), hanya terjadi penukaran tempat dengan nilainya di iterasi yang ke I dengan lokasi yang ditemukan (Lok).

1. **Metode Gelembung (*bubble sort*)**

Metode gelembung (bubble sort) ini biasa disebut dengan metode penukaran (***Exchange Sort***). Metode gelembung ini mendasarkan penukaran dua buah elemen untu mencapau keadaan urut yang diinginkan. Metode ini cukup mudah untuk dipahami dan program, tetapi dari beberapa metode yang ada, metode ini yang paling tidak efisien.

*Langkah 0 = Bca vektor yang akan diurutkan (dalam program utama)*

*Langkah 1 = Kerjakan langkah 2 untuk I = 1 sampau N – 1*

*Langkah 2 = Kerjakan langkah 3 untuk J = 1 sampau N – 1*

*Langkah 3 = Tes apakah A[J] > A[J+1]?*

*Jika Ya, tukarkan nilai kedua elemen itu.*

*Langkah 4 = Selesai*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Iterasi ke | A[1] | A[2] | A[3] | A[4] | A[5] |
| Awal | 40 | 50 | 20 | 15 | 10 |
| I = 1 | 40 | 50 | 20 | 15 | 10 |
|  | 40 | 20 | 50 | 15 | 10 |
|  | 40 | 20 | 15 | 50 | 10 |
|  | 40 | 20 | 15 | 10 | 50 |
| I = 2 | 20 | 40 | 15 | 10 | 50 |
|  | 20 | 15 | 40 | 10 | 50 |
|  | 20 | 15 | 10 | 40 | 50 |
| I = 3 | 15 | 20 | 10 | 40 | 50 |
|  | 15 | 10 | 20 | 40 | 50 |
| I = 4 | 10 | 15 | 20 | 40 | 50 |

Untuk membaca vektor menjadi dalam keadaan urut bisa dilaksanakan dengan dua cara. Cara pertama adalah selalu meletakkan elemen dengn nilai paling besar pasa posisi terakhir (posisi ke N). Kemudian elemen dengan nilai paling besar kedua diletakkan pada posisi ke N-1, begitu seterusnya, cara kedua adalah kebalikan cara pertama. Dalam hal ini yang digunakan sebagau patokan adalah nilai terkecil. Dengan kata lain, pada iterasi pertama kita akaan meletakkan elemen dengan nilai terkecil pada posisi ke-1, kemudian nilai terkecil kedua diletakkan pada posisi ke-2, dan seterusnya. Gambar diatas adalah menggunakan cara pertama.

Banyaknya iterasi dengan vektor sejumlah N pada metode ini memerlukan iterasi sebanyak N-1 kali. Di samping itu bisa kita perhatikan pula bahwa apabila nomor iterasi kita tambah dengan banyaknya langkah pada iterasi tersebut, besarnya selalu tetap, yaitu sama dengan N. Sebagai contoh, untuk iterasi ke 1 terdapat 4 langkah; iterasi ke-2 terdapat 3 langkah, iterasi ke-3 terdapat 2 langkah, dan ke-4 ada 1 langkah.

Metode ini paling tidak efisien, karena apabila kita mengurutkan vektor sebanyak N elemen dan pada iterasi yang kurang dari N-1, maka iterasi tersebut harus tetap dilaksanakan samapau N-1. Dengan demikian, dalam metode Gelembung akan terjadi pembandingan dan pemindahan atau penukaran dua elemen dapat dihitung seperti berikut.

C = (N2 – N) / 2

Mmin = 0

Mrata-rata = 3 (N2 – n) / 4

Mmax = 3 (N2 – 4) / 2

Prosedur untuk pengurutan vector menggunakan metode gelembung

*Procedure Bubble\_Sort (var A : Larik; N : Integer);*

*Var I, J, Lok : Integer;*

*Begin*

*For I : = 1 to N – 1 do*

*For J : = 1 to N – 1 do*

*If A[J] >[J+1] then*

*TUKARKAN (A[J], A[J+1]);*

*End;*

*Procedure TUKARKAN (Var A, B : real);*

*Var T : real;*

*Begin*

*T : = A;*

*A : = B;*

*B : = T*

*End.*

**Kesimpulan**

1. Apabila data yang kita urutkan sedikit jumlahnya, maka penggunaan algoritma Bubble sort tidak ada pengaruh. Karena RAM masih bisa menampung semua data.
2. Apabila data yang akan kita urutkan sudah lebih banyak sedangkan kita menggunakan RAM hanya sebagian saja, karena masih ada proses yang lain yang sedang ditangani oleh RAM maka menggunakan 2 pilihan yaitu Insertion Sort atau Selection Sort.
3. Apabila datanya sudah ada dan perbah dibuat, tetapi belum kondisi terurut sama sekali, seperti sebuah data base yang besar tetapi belum ada fungsi untuk mengurutkan, maka alternatif yang terbaik fungsi atau procedure untuk mengurutkan adalah menggunakan Selection Sort. Metode ini lebih cepat karena pergerakan dari pemindahan (M) hanya dilakukan 1 kali tiap iterasi. Yaitu pemindahan antara bilangan yang dicari di Lokasi Array dengan iterasi yang I. Sehingga kalau datanya banyak waktu untuk akses pengurutan bisa dihemat di pemindahan (M).
4. Tetapi apabila datanya sudah ada dan dalam kondisi yang hampir terurut, contohnya database dari sebuah perusahaan / institusi, maka alternatif yang terbaik adalah menggunakan prosedur Insertion Sort. Metode ini menjadi alternatif karena banyaknya pembandingan dan pemindahan akan minimum jika elemen-elemen semua sudah dalam keadaan urut; akan maksimum jika elemen-elemen semua sudah dalam keadaan urut tetapi secara urut turun (sedang tujuan kita adalah unrut naik).

***.Edson Yahuda Putra*** adalah dosen di Fakultas Ilmu Komputer Universitas Klabat.

**DAFTAR PUSTAKA**

Forouzan Gilberg. (1998). *Data Structure a Pseudocode Approach with C.* Boston. PWS Publishing Comp.

Roman A. Mata-Toledo. (2000). *Introduction to Computer Scince.* Singapore. MC-Graw Hill, Comp.

Santoso, P. I. (1993). *Struktur Data menggunakan Turbo Pascal 6.0.* Jogyakarta. Andi offset.