Pengurutan (Sorting)

BAB 6

Pengurutan (Sorting)

6.1 Deklarasi Larik

6.2 Straight Insertion Sort

6.3 Binary Insertion Sort

6.4 Selection Sort

Introduction to Sorting

Pengurutan data (*sorting*) didefinisikan sebagai suatu proses untuk menyusun kembali humpunan obyek menggunakan aturan tertentu.

Menurut Microsoft Book-shelf, definisi algoritma pengurutan adalah algoritma untuk meletakkan kumpulan elemen data ke dalam urutan tertentu berdasarkan satu atau beberapa kunci dalam tiap-tiap elemen.

Ada dua macam urutan yang biasa digunakan dalam proses pengurutan yaitu
- urut naik (*ascending*) yaitu dari data yang mempunyai nilai paling kecil sampai paling besar
- urut turun (*descending*) yaitu data yang mempunyai nilai paling besar sampai paling kecil.

Contoh : data bilangan 5, 2, 6 dan 4 dapat diurutkan naik menjadi 2, 4, 5, 6 atau diurutkan turun menjadi 6, 5, 4, 2.

Pada data yang bertipe char, nilai data dikatakan lebih kecil atau lebih besar dari yang lain didasarkan pada urutan relatif (*collating sequence*) seperti dinyatakan dalam tabel ASCII

Keuntungan Sorting Data

data mudah dicari (misalnya dalam buku telepon atau kamus bahasa), mudah untuk dibetulkan, dihapus, disisipi atau digabungkan.

Dalam keadaan terurutkan, kita mudah melakukan pengecekan apakah ada data yang hilang melakukan kompilasi program komputer jika tabel-tabel simbol harus dibentuk

mempercepat proses pencarian data yang harus dilakukan berulang kali.

Faktor yang berpengaruh pada efektifitas algoritma sorting

banyaknya data yang diurutkan

kapasitas memori, apakah mampu menyimpan semua data yang kita miliki

tempat penyimpanan data, misalnya piringan, pita atau kartu, atau media penyimpan yang lain.

Pemilihan Algoritma

Pemilihan algoritma sangat ditentukan oleh struktur data yang digunakan. Metode pengurutan yang digunakan dapat diklasifikasikan menjadi dua katagori yaitu :

1. pengurutan internal, yaitu pengurutan dengan menggunakan larik (*array*). Larik tersimpan dalam memori utama komputer
2. pengurutan eksternal, yaitu pengurutan dengan menggunakan berkas (*sequential access file*). Berkas tersimpan dalam pengingat luar, misalnya cakram atau pita magnetis.

Untuk menggambarkan pengurutan dengan larik, bisa kita bayangkan semua kartu terletak di hadapan kita sehingga semua kartu terlihat dengan jelas nomornya.

Pada penyusunan kartu sebagai sebuah berkas, kita bayangkan semua kartu kita tumpuk sehingga hanya kartu bagian atas saja yang bisa kita lihat nomornya.

6.1 Deklarasi Larik

Deklarasi larik yang digunakan adalah larik dimensi satu (*vektor*) dengan elemennya bertipe integer.

Pada deklarasi di samping, Max adalah banyaknya elemen vektor. Kita bisa mengubah nilai konstantaa Max sesuai kebutuhan. Indeks larik dimulai dari 0. Data yang sebenarnya disimpan mulai dari indeks 0.

Prosedur tersebut akan digunakan pada beberapa prosedur pengurutan yang akan dijelaskan selanjutnya

```
#define Max 100;
int Data[Max];
void Tukar (int *a, int *b)
{
        int temp;
        temp = *a;
        *a = *b;
        *b = temp;
}
```

6.2 Straight Insertion Sort (Penyisipan Langsung)

Data dicek satu per satu mulai dari yang kedua sampai dengan yang terakhir.

Apabila ditemukan data yang lebih kecil daripada data sebelumnya, maka data tersebut disisipkan pada posisi yang sesuai.

Akan lebih mudah apabila membayangkan pengurutan kartu.

Pertama-tama anda meletakkan kartu-kartu tersebut di atas meja, kemudian melihatnya dari kiri ke kanan.

Apabila kartu di sebelah kanan lebih kecil daripada kartu di sebelah kiri, maka ambil kartu tersebut dan sisipkan di tempat yang sesuai.

Algoritma Straight Insertion Sort

1 -- i ← 1
2 -- selama (i < N) *kerjakan baris 3 sampai dengan 9*
3 -- x ← Data[i]
4 -- j ← i – 1
5 -- selama (x < Data[j]) *kerjakan baris 6 dan 7*
6 -- Data[j + 1] ← Data[j]
7 -- j ← j – 1
8 -- Data[j+1] ← x
9 -- i ← i + 1

```
void StraighInsertSort()
{
int i, j, x;
for(i=1; i<Max; i++){
        x = Data[i];
        j = i - 1;
        while (x < Data[j]){
        Data[j+1] = Data[j];
        j--;
        }
Data[j+1] = x; }
}
```

Tabel 6.1 Proses Pengurutan dengan Metode Penyisipan Langsung

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=1	12	**35**	9	11	3	17	23	15	31	20
i=2	12	35	**9**	11	3	17	23	15	31	20
i=3	9	12	35	**11**	3	17	23	15	31	20
i=4	9	11	12	35	**3**	17	23	15	31	20
i=5	3	9	11	12	35	**17**	23	15	31	20
i=6	3	9	11	12	17	35	**23**	15	31	20
i=7	3	9	11	12	17	23	35	**15**	31	20
i=8	3	9	11	12	15	17	23	35	**31**	20
i=9	3	9	11	12	15	17	23	31	35	**20**
Akhir	3	9	11	12	15	17	20	23	31	35

Penjelasan Tabel

- Pada saat **i=1**, **x** sama dengan **Data[1] = 35** dan **j=0**. Karena **Data[0] = 12** dan **35 > 12** maka proses dilanjutkan untuk **i=2**.
- Pada saat **i=2**, **x = Data[2] = 9** dan **j=1.** Karena **Data[1] = 35** dan **9 < 35**, maka dilakukan pergeseran sampai ditemukan data yang lebih kecil dari **9**. Hasil pergeseran ini, **Data[1] = 12** dan **Data[2] = 35** sedangkan **Data[0] = x = 9**.
- Pada saat **i=3**, **x = Data[3] = 11** dan **j=3**. Karena **Data[2] = 35** dan **11 < 35**, maka dilakukan pergeseran sampai ditemukan data yang lebih kecil dari **11**. Hasil pergeseran ini, **Data[2] = 12** dan **Data[3] = 35** sedangkan **Data[1] = x = 11**.
- Dan seterusnya

Menghitung jumlah perbandingan dan pergeseran pada Straight Insertion Sort

Jumlah Perbandingan (=C)

$Cmin = N - 1$
$Crata\text{-}rata = (N2 + N + 2) / 4$
$Cmax = (N2 + N - 2) / 2$

Jumlah Pergeseran (=M)

$Mmin = 2(N - 1)$
$Mrata\text{-}rata = (N2 + 7N - 8) / 4$
$Mmax = (N2 + 3N - 4) / 2$

6.3 Binary Insertion Sort

Metode ini merupakan pengembangan dari metode penyisipan langsung.

Dengan cara penyisipan langsung, perbandingan selalu dimulai dari elemen pertama (data ke-0), sehingga untuk menyisipkan elemen ke i kita harus melakukan perbandingan sebanyak i-1 kali.

Ide dari metode ini didasarkan pada kenyataan bahwa pada saat menggeser data ke-i, data ke 0 s/d i-1 sebenarnya sudah dalam keadaan terurut.

Sebagai contoh pada tabel sebelumnya, pada saat i=4, data ke 0 s/d 3 sudah dalam keadaan urut : 3, 9, 12, 35. Dengan demikian posisi dari data ke-i sebenarnya dapat ditentukan dengan pencarian biner.

Salah Satu Contoh Binary Sort

Misalnya pada saat i = 7, data yang akan dipindah adalah 15 sedangkan data di sebelah kiri 15 adalah sebagai berikut :

3	9	11	12	17	23	35	15

Pertama-tama dicari data pada posisi paling tengah diantara data diatas. Data yang terletak di tengah adalah data ke-3, yaitu 12. Karena 12 < 15, berarti 15 harus disisipkan di sebelah kanan 12. Oleh karena itu, proses pencarian dlanjutkan lagi untuk data berikut

17	23	35

Dari hasil ini, didapat data tengahnya adalah data 23. Karena 15 < 23, berarti 15 harus disisipkan di sebelah kiri 23. Proses dilaniutkan kembali untuk data

17

Karena 17 > 15, berarti 15 harus disisipkan di sebelah kiri 17

Algoritma Binary Sort

1 -- i ← 1
2 -- selama (i < N) kerjakan baris 3 sampai dengan 14
3 -- x ← Data[i]
4 -- l ← 0
5 -- r ← i – 1
6 -- selama (l<=r) kerjakan baris 7 dan 8
7 -- m ← (l + r) / 2
8 -- jika (x < Data[m]) maka r ← m – 1, jika tidak l ← m + 1
9 -- j ← i – 1
10 -- selama (j >=l) kerjakan baris 11 dan 12
11 -- Data[j+1] ← Data[j]
12 -- j ← j – 1
13 -- Data[l] ← x
14 -- l ← i + 1

```
void BinaryInsertSort()
{ int i, j, l, r, m, x;
for (i=1; i<Max; i++){
        x = Data[i];
        l = 0;
        r = i - 1;
        while(l <= r){
        m = (l + r) / 2;
        if(x < Data[m])
        r = m - 1;
        else
        l = m + 1; }
for(j=i-1; j>=l; j--)
Data[j+1] = Data[j];
Data[l]=x; } }
```

Menghitung jumlah perbandingan dan pergeseran pada Binary Insertion Sort

Dari algoritma dan prosedur diatas, jumlah perbandingan (=*C*) untuk metode penyisipan biner adalah :

$$C = \sum \lceil 2\log(i) \rceil$$

Sedangkan jumlah penggeseran (=*M*) untuk metode penyisipan biner sama dengan metode penyisipan langsung (Insertion Sort)

6.4 Selection Sort

Metode seleksi melakukan pengurutan dengan cara mencari data yang terkecil kemudian menukarkannya dengan data yang digunakan sebagai acuan atau sering dinamakan pivot.

Langkah Selection Sort

Langkah pertama dicari data terkecil dari data pertama sampai data terakhir.

Kemudian data terkecil ditukar dengan data pertama.

Dengan demikian, data pertama sekarang mempunyai nilai paling kecil dibanding data yang lain.

Langkah kedua, data terkecil kita cari mulai dari data kedua sampai terakhir.

Data terkecil yang kita peroleh ditukar dengan data kedua dan demikian seterusnya sampai semua elemen dalam keadaan terurutkan.

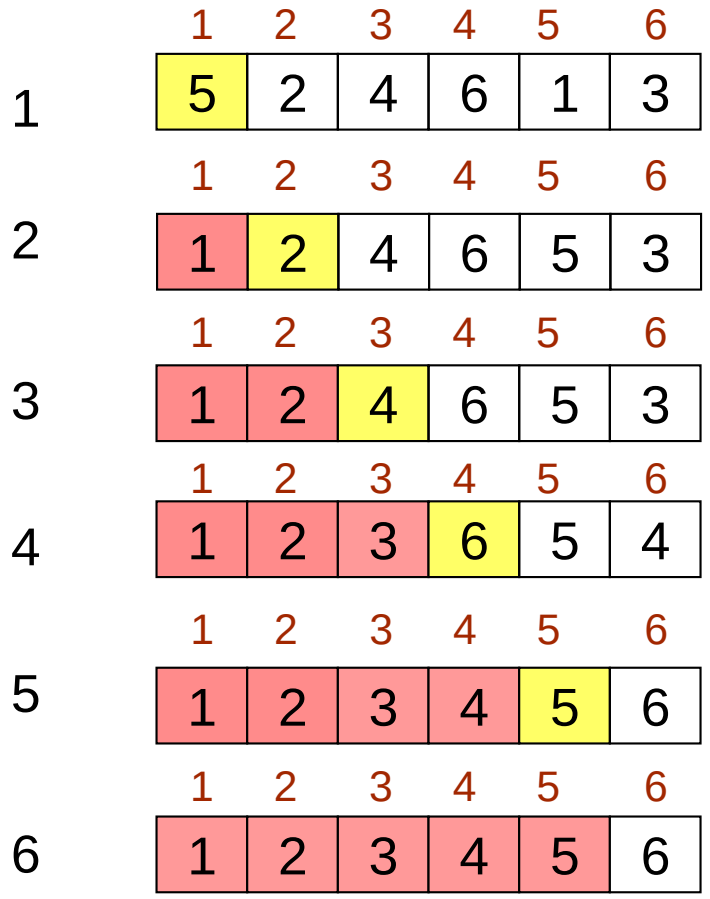
Algoritma Selection Sort

1 -- i ← 0
2 -- selama (i < N-1) kerjakan baris 3 sampai dengan 9
3 -- k ← i
4 -- j ← i + 1
5 -- Selama (j < N) kerjakan baris 6 dan 7
6 -- Jika (Data[k] > Data[j]) maka k ← j
7 -- j ← j + 1
8 -- Tukar Data[i] dengan Data[k]
9 -- i ← i + 1

```
void SelectionSort()
{
int i, j, k;
for(i=0; i<Max-1;i++){
        k = i;
        for (j=i+1; j<Max; j++)
        if(Data[k] > Data[j])
        k = j;
        Tukar(&Data[i], &Data[k]);
        }
}
```

Iterasi	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]	Data[8]	Data[9]
Awal	12	35	9	11	3	17	23	15	31	20
i=0	**12**	35	9	11	**3**	17	23	15	31	20
i=1	3	**35**	**9**	11	12	17	23	15	31	20
i=2	3	9	**35**	**11**	12	17	23	15	31	20
i=3	3	9	11	**35**	**12**	17	23	15	31	20
i=4	3	9	11	12	**35**	17	23	**15**	31	20
i=5	3	9	11	12	15	**17**	23	35	31	20
i=6	3	9	11	12	15	17	**23**	35	31	**20**
i=7	3	9	11	12	15	17	20	**35**	31	**23**
i=8	3	9	11	12	15	17	20	23	**31**	35
Akhir	3	9	11	12	15	17	20	23	31	35

Selection Sort: contoh



Carilah elemen terkecil & tukar dengan “5”

1 fixed. Carilah elemen terkecil & tukar dengan “2”

1,2 fixed. Carilah elemen terkecil & tukar dengan “4”

1,2,3 fixed. Carilah elemen terkecil & tukar dengan “6”

1,2,3,4 fixed. Carilah elemen terkecil & tukar dengan “5”

1,2,3,4,5 fixed, otomatis elemen terakhir sudah pada posisi yang benar

Langkah pada Insertion Sort

Untuk lebih memperjelas langkah-langkah algoritma seleksi dapat dilihat pada tabel 6.2. Proses pengurutan pada tabel 6.2 dapat dijelaskan sebagai berikut:

- Pada saat i=0, data terkecil antara data ke-1 s/d ke-9 adalah data ke-4, yaitu 3, maka data ke-0 yaitu 12 ditukar dengan data ke-4 yaitu 3.
- Pada saat i=1, data terkecil antara data ke-2 s/d ke-9 adalah data ke-2, yaitu 9, maka data ke-1 yaitu 35 ditukar dengan data ke-2 yaitu 9.
- Pada saat i=2, data terkecil antara data ke-3 s/d ke-9 adalah data ke-3, yaitu 11, maka data ke-2 yaitu 35 ditukar dengan data ke-3 yaitu 11.
- Pada saat i=3, data terkecil antara data ke-4 s/d ke-9 adalah data ke-4, yaitu 12, maka data ke-3 yaitu 35 ditukar dengan data ke-4 yaitu 12.
- Dan seterusnya.

Menghitung jumlah perbandingan dan pergeseran pada Binary Insertion Sort

Dari algoritma dan prosedur diatas, jumlah perbandingan (=*C*) metode seleksi adalah

$C = N(N - 1) / 2$

Jumlah penukaran (=*M*) pada metode seleksi tergantung keadaan datanya.

Penukaran minimum terjadi bila data sudah dalam keadaan urut, sebaliknya jumlah penukaran maksimum terjadi bila data dalam keadaan urut terbalik.

Jumlah penukaran minimum dan maksimum dapat dirumuskan sebagai berikut :

$Mmin = 3(N - 1)$

$Mmax = \lfloor N2 / 4 \rfloor + 3(N - 1)$

Terima Kasih

Pertanyaan?