Pengurutan 2 (Sorting 2)

BAB 6

Pengurutan 2 (Sorting 2)

6.5 Bubble Sort

6.6 Quick Sort

6.7 Merge Sort

6.5 Bubble Sort

Metode gelembung (*bubble sort*) sering juga disebut dengan metode penukaran (*exchange sort*) adalah metode yang mengurutkan data dengan cara membandingkan masing-masing elemen, kemudian melakukan penukaran bila perlu.

Metode ini mudah dipahami dan diprogram, tetapi bila dibandingkan dengan metode lain yang kita pelajari, metode ini merupakan metode yang paling tidak efisien.

Proses pengurutan metode gelembung ini menggunakan dua kalang.

Kalang pertama melakukan pengulangan dari elemen ke 2 sampai dengan elemen ke N-1 (misalnya variable i), sedangkan kalang kedua melakukan pengulangan menurun dari elemen ke N sampai elemen ke i (misalnya variable j).

Pada setiap pengulangan, elemen ke j-1 dibandingkan dengan elemen ke j. Apabila data ke j-1 lebih besar daripada data ke j, dilakukan penukaran.

Algoritma Bubble Sort

1 -- i ← 0
2 -- selama (i < N-1) kerjakan baris 3 sampai dengan 7
3 -- j ← N - 1
4 -- Selama (j >= i) kerjakan baris 5 sampai dengan 7
5 -- Jika (Data[j-1] > Data[j]) maka tukar Data[j-1] dengan Data[j]
6 -- j ← j – 1
7 -- i ← i + 1

Proses Pengurutan Bubble Sort (1)

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; j=9	12	35	9	11	3	17	23	15	**31**	**20**
j=8	12	35	9	11	3	17	23	**15**	**20**	31
j=7	12	35	9	11	3	17	**23**	**15**	20	31
j=6	12	35	9	11	3	**17**	**15**	23	20	31
j=5	12	35	9	11	**3**	**15**	17	23	20	31
j=4	12	35	9	**11**	**3**	15	17	23	20	31
j=3	12	35	**9**	**3**	11	15	17	23	20	31
j=2	12	**35**	**3**	9	11	15	17	23	20	31
j=1	**12**	**3**	35	9	11	15	17	23	20	31

Proses Pengurutan Bubble Sort (2)

i=2; j=9	3	12	35	9	11	15	17	23	**20**	**31**
j=8	3	12	35	9	11	15	17	**23**	**20**	31
j=7	3	12	35	9	11	15	**17**	**20**	23	31
j=6	3	12	35	9	11	**15**	**17**	20	23	31
j=5	3	12	35	9	**11**	**15**	17	20	23	31
j=4	3	12	35	**9**	**11**	15	17	20	23	31
j=3	3	12	**35**	**9**	11	15	17	20	23	31
j=2	3	**12**	**9**	35	11	15	17	20	23	31
i=3; j=9	3	9	12	35	11	15	17	20	**23**	**31**
j=8	3	9	12	35	11	15	17	**20**	**23**	31
j=7	3	9	12	35	11	15	**17**	**20**	23	31
j=6	3	9	12	35	11	**15**	**17**	20	23	31
j=5	3	9	12	35	**11**	**15**	17	20	23	31
j=4	3	9	12	**35**	**11**	15	17	20	23	31
j=3	3	9	**12**	**11**	35	15	17	20	23	31

Proses Pengurutan Bubble Sort (3)

i=5; j=9	3	9	11	12	15	35	17	20	**23**	**31**
j=8	3	9	11	12	15	35	17	**20**	**23**	31
j=7	3	9	11	12	15	35	**17**	**20**	23	31
j=6	3	9	11	12	15	**35**	**17**	20	23	31
j=5	3	9	11	12	**15**	**17**	35	20	23	31
i=6; j=9	3	9	11	12	15	17	35	20	**23**	**31**
j=8	3	9	11	12	15	17	35	**20**	**23**	31
j=7	3	9	11	12	15	17	**35**	**20**	23	31
j=6	3	9	11	12	15	**17**	**20**	35	23	31
i=7; j=9	3	9	11	12	15	17	20	35	**23**	**31**
j=8	3	9	11	12	15	17	20	**35**	**23**	31
j=7	3	9	11	12	15	17	**20**	**23**	35	31
i=8; j=9	3	9	11	12	15	17	20	23	**35**	**31**
j=8	3	9	11	12	15	17	20	23	31	35
Akhir	3	9	11	12	15	17	20	23	31	35

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	7	4	5	8	10

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	7	4	5	10	8

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	7	4	10	5	8

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	7	10	4	5	8

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8
Step-2	10	7	4	5	8

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8
Step-2	10	7	4	8	5

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8
Step-2	10	7	8	4	5

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8
Step-2	10	8	7	4	5

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8
Step-2	10	8	7	4	5
Step-3	10	8	7	4	5

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8
Step-2	10	8	7	4	5
Step-3	10	8	7	5	4

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8
Step-2	10	8	7	4	5
Step-3	10	8	7	5	4

Bubble Sort: tahap demi tahap

Awal	7	4	5	8	10
Step-1	10	7	4	5	8
Step-2	10	8	7	4	5
Step-3	10	8	7	5	4
Step-4	10	8	7	5	4

Prosedur Bubble Sort

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

Jumlah Pembandingan dan Pertukaran

Jumlah Pembandingan

$C = (N2 – N) / 2$

Jumlah Pertukaran Data

$Mmin = 0$
$Mrata\text{-}rata = 3(N2 - N) / 4$
$Mmax = 3(N2 - N) / 2$

6.6 Quick Sort

Metode Quick sering disebut juga metode partisi (partition exchange sort).

Untuk mempertinggi efektifitas dari metode ini, digunakan teknik menukarkan dua elemen dengan jarak yang cukup besar.

Proses Quick Sort

Proses penukaran dengan metode quick dapat dijelaskan sebagai berikut.:

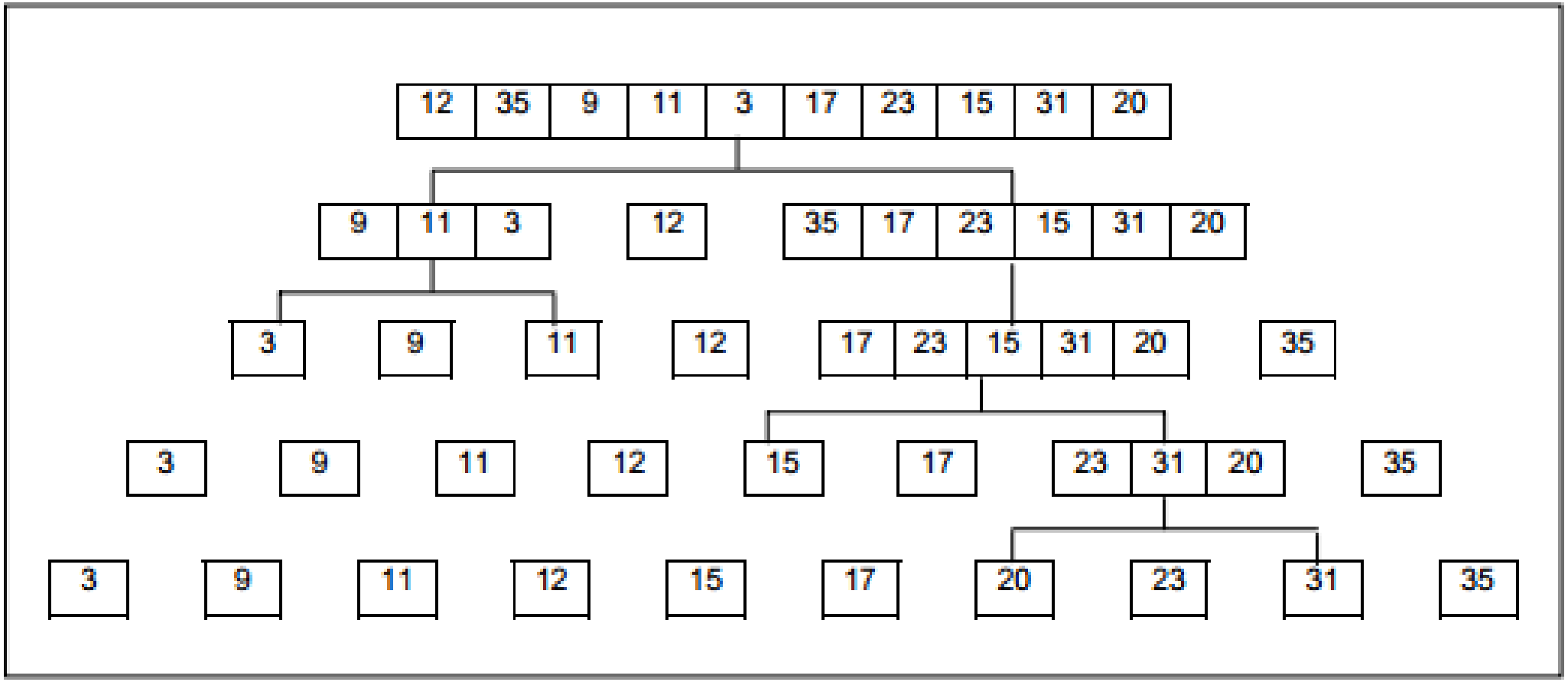
mulamula dipilih data tertentu yang disebut pivot, misalnya x. Pivot dipilih untuk mengatur data di sebelah kiri agar lebih kecil daripada pivot dan data di sebelah kanan agar lebih besar daripada pivot.

Pivot ini diletakkan pada posisi ke j sedemikian sehingga data antara 1 sampai dengan j-1 lebih kecil daripada x.

Sedangkan data pada posisi ke j+1 sampai N lebih besar daripada x. Caranya dengan menukarkan data diantara posisi 1 sampai dengan j-1 yang lebih besar daripada x dengan data diantara posisi j+1 sampai dengan N yang lebih kecil daripada x.

Secara default Quick Sort dilakukan secara rekursif, namun dapat juga tidak menggunakan prosedur rekursif

Ilustrasi Quick Sort



Quick Sort Rekursif

1 x ← Data[(L + R) / 2]

2 i ← L

3 j ← R

4 Selama (i <= j) kerjakan baris 5 sampai dengan 12

5 Selama (Data[i] < x) kerjakan i ← i + 1

6 Selama (Data[j] > x) kerjakan j ← j – 1

7 Jika (i <= j) maka kerjakan baris 8 sampai dengan 10; jika tidak kerjakan baris 11

8 Tukar Data[i] dengan Data[j]

9 i ← i + 1

10 j ← j –1

11 Jika (L < j) kerjakan lagi baris 1 dengan R = j

12 Jika (i < R) kerjakan lagi baris 1 dengan L = I

Quick Sort Non Rekursif

1 Tumpukan[1].Kiri ← 0

2 Tumpukan[1].Kanan ← N-1

3 Selama ujung ≠ 0 kerjakan baris 4 sampai dengan 22

4 L ← Tumpukan[ujung].Kiri

5 R ← Tumpukan[ujung].Kanan

6 ujung ← ujung – 1

7 Selama (R > L) kerjakan baris sampai 8 dengan 22

8 i ← L

9 j ← R

10 x ← Data[(L + R) / 2]

11 Selama i <= j kerjakan baris 12 sampai dengan 14

12 Selama (Data[i] < x), i ← i + 1

13 Selama (x < Data[j]), j ← j – 1

14 Jika (i <= j) maka kerjakan baris 15 sampai dengan 17, jika tidak ke baris 11

15 Tukar Data[i] dengan Data[j]

16 i ← i + 1

17 j ← j –1

18 Jika (L < i) maka kerjakan baris 19 sampai dengan 21

19 ujung ← ujung + 1

20 Tumpukan[ujung].Kiri = i

21 Tumpukan[ujung].Kanan = R

22 R ← j

Prosedur Quick Sort Rekursif

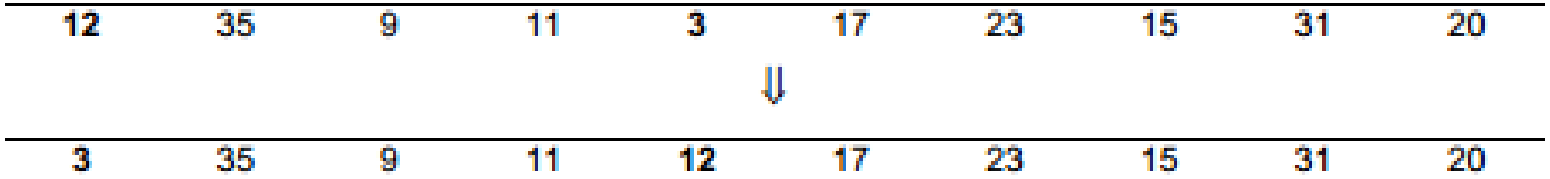
```
void QuickSortRekursif(int L, int R)
{
     int i, j, x;
     x = data[(L+R)/2];
     i = L;
     j = R;
     while (i <= j){
          while(Data[i] < x)
               i++;
          while(Data[j] > x)
               j--;
          if(i <= j){
               Tukar(&Data[i], &Data[j]);
               i++;
               j--;
          }
     }
          if(L < j)
               QuickSortRekursif(L, j);
          if(i < R)
               QuickSortRekursif(i, R);
}
```

Proses Quick Sort Non Rekursif

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
L=0;R=9	**12**	35	9	11	**3**	17	23	15	31	20
L=0;R=4	3	**35**	**9**	11	12	17	23	15	31	20
L=1;R=3	3	9	**35**	**11**	12	17	23	15	31	20
L=1;R=9	3	9	11	**35**	12	17	23	**15**	31	20
L=2;R=4	3	9	11	**15**	**12**	17	23	35	31	20
L=5;R=9	3	9	11	12	15	17	23	**35**	31	**20**
L=6;R=7	3	9	11	12	15	17	**23**	**20**	31	35
L=8;R=9	3	9	11	12	15	17	20	23	31	35
Akhir	3	9	11	12	15	17	20	23	31	35

Langkah2 (1)

Mula-mula L=0 dan R=9 dan pivot adalah pada data ke-4 yaitu 3. Kita mencari data di sebelah kiri pivot yang lebih besar daripada 3, ternyata data ke-0 yaitu 12 lebih besar daripada 3. Untuk data di sebelah kanan pivot, ternyata tidak ada data yang lebih kecil daripada 3, yang berarti 3 adalah data terkecil. Sekarang 3 harus ditukar dengan 12



Langkah berikutnya membuat dua kumpulan data baru berdasarkan hasil ini. Kumpulan data pertama adalah data yang memiliki indeks 0 s/d 4-0=4 yaitu

Kumpulan data kedua adalah data yang memiliki indeks 0 + 1 = 1 s/d 9. Kumpulan data kedua ini belum bisa ditentukan sekarang karena masih tergantung dari hasil pengurutan kumpulan data pertama.

Langkah2 (2)

Kembali ke kumpulan data pertama, dicari pivot kemudian menggunakan aturan yang serupa. Pivot pada kumpulan data ini adalah data ke-2 yaitu 9. Perhatikan terdapat data yang lebih besar dari 9 di sebelah kiri yaitu 35 sehingga 35 ditukar dengan 9.

3	9	35	11

Langkah selanjutnya membuat dua kumpulan data lagi. Kumpulan pertama mempunyai indeks 0 sampai dengan 4-1=3. Jadi kumpulan data pertama menjadi

3	9	35

Pivot dari kumpulan data ini adalah 9. Perhatikan tidak ada data yang lebih besar dari 9 di sebelah kiri dan lebih kecil dari 9 di sebelah kanan.

Kumpulan kedua dari indeks 0 s/d 4 adalah data ke 2 dan ke 4-1=3 yaitu 35 dan 11. Ternyata 35 lebih besar dari 11 sehingga kedua data ini ditukar.

Langkah2 (3)

Kembali ke kumpulan data kedua yang memiliki indeks 1 s/d 9. Kumpulan ini dibagi menjadi dua yaitu kumpulan data berindeks 1 s/d 4 dan kumpulan data berindeks 5 s/d 9. Pivot dari data ini adalah data ke 5 yaitu 17. Kumpulan data ini dapat dituliskan sebagai berikut :

9	11	35	12	17	23	15	31	20

Data yang lebih besar dari 17 di sebelah kiri adalah 35 dan data yang lebih kecil dari 17 di sebelah kanan adalah 15, jadi kedua data ini ditukar menjadi

9	11	15	12	17	23	35	31	20

Dari hasil penukaran ini dilakukan pembagian menjadi 2 kumpulan data. Kumpulan pertama yaitu dari indeks 2 s/d 4, pivot pada data ke 3 dan terjadi penukaran data 15 dan 12 sehingga menjadi

11	12	15

Langkah2 (4)

Kumpulan kedua yaitu dari indeks 4 s/d 5 tidak terjadi pertukaran data

Kembali ke kumpulan kedua dari indeks 5 s/d 9. Pivot dari kumpulan ini adalah 35. Dicari data yang lebih besar dari 35 di sebelah kiri dan data yang lebih kecil dari 35 di sebelah kanan. Ternyata terjadi pertukaran data 35 dan 20.

Dari hasil pertukaran ini dilakukan pembagian kumpulan data yaitu data yang mempunyai indeks 6 s/d 7 dan 8 s/9. Kumpulan data pertama terjadi pertukaran data 23 dan 20 sedangkan kumpulan data kedua tidak terjadi pertukaran data

Prosedur Quick Sort Non Rekursif

```
void QuickSortNonRekursif(int N)
{
   const M = MaxStack;
   struct tump {
      int Kiri;
      int Kanan;
   }Tumpukan[M];

   int i, j, L, R, x, ujung = 1;
   Tumpukan[1].Kiri = 0;
   Tumpukan[1].Kanan = N-1;

   while (ujung!=0){
      L = Tumpukan[ujung].Kiri;
      R = Tumpukan[ujung].Kanan;
      ujung--;
      while(R > L){
         i = L;
         j = R;
         x = Data[(L+R)/2];
         while(i <= j){
            while(Data[i] < x)
                  i++;
            while(x < Data[j])
                  j--;
            if(i <= j){
                  Tukar(&Data[i], &Data[j]);
                  i++;
                  j--;
            }
         }
         if(L < i){
            ujung++;
            Tumpukan[ujung].Kiri = i;
            Tumpukan[ujung].Kanan = R;
         }
         R = j;
      }
   }
}
```

6.7 Merge Sort

Metode penggabungan biasanya digunakan pada pengurutan berkas. Prinsip dari metode penggabungan sebagai berikut : mula-mula diberikan dua kumpulan data yang sudah dalam keadaan urut. Kedua kumpulan data tersebut harus dijadikan satu table sehingga dalam keadaan urut.

Misalnya kumpulan data pertama (T1) adalah sebagai berikut :

3	11	12	23	31

Sedangkan kumpulan data kedua (T2) adalah sebagai berikut :

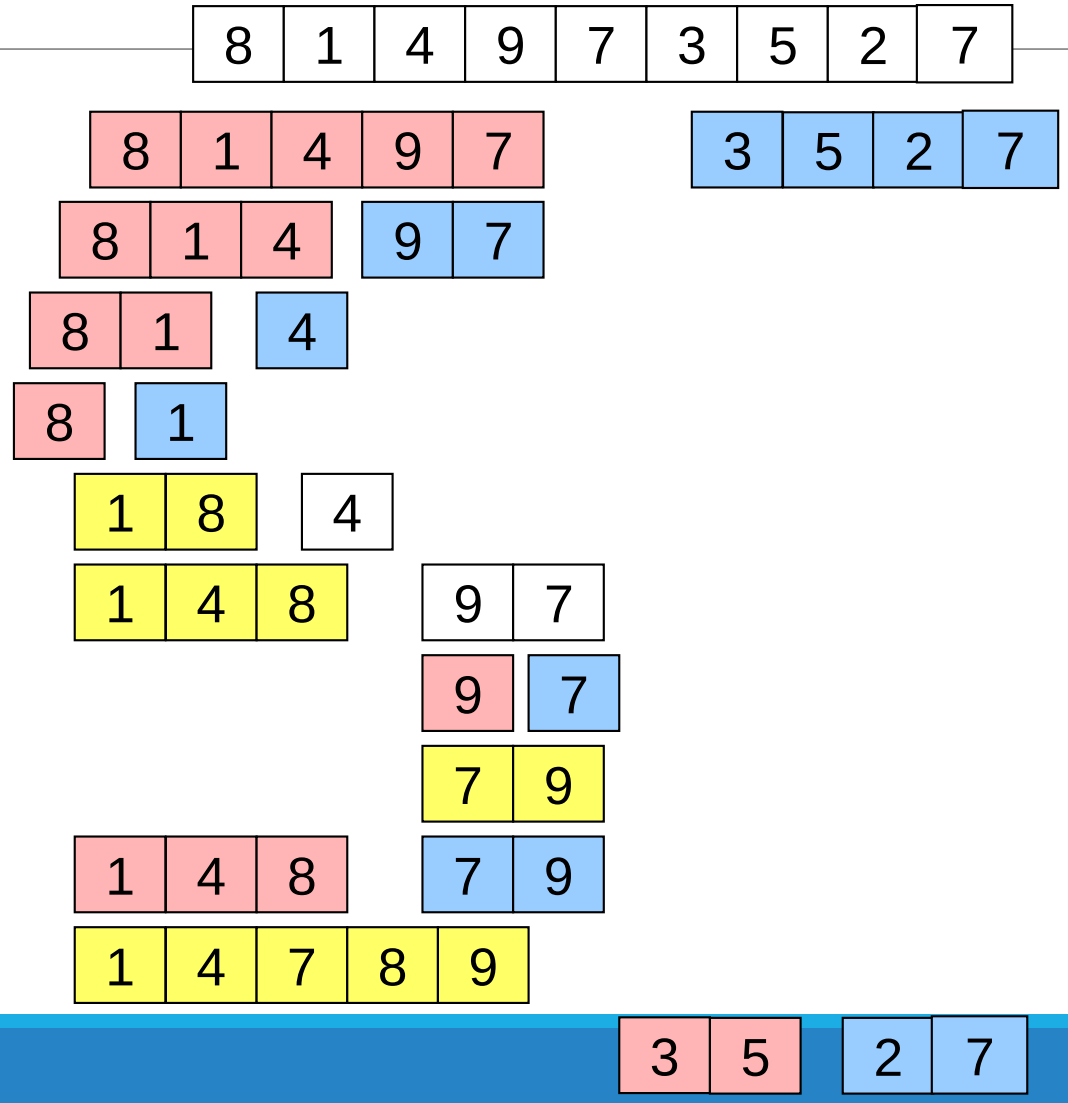
9	15	17	20	35

Proses penggabungan ini dapat dijelaskan sebagai berikut : mula-mula diambil data pertama dari T1 yaitu 3 dan data pertama dari T2 yaitu 9. Data ini dibandingkan, kemudian yang lebih kecil diletakkan sebagai data pertama dari hasil pengurutan, misalnya T3.

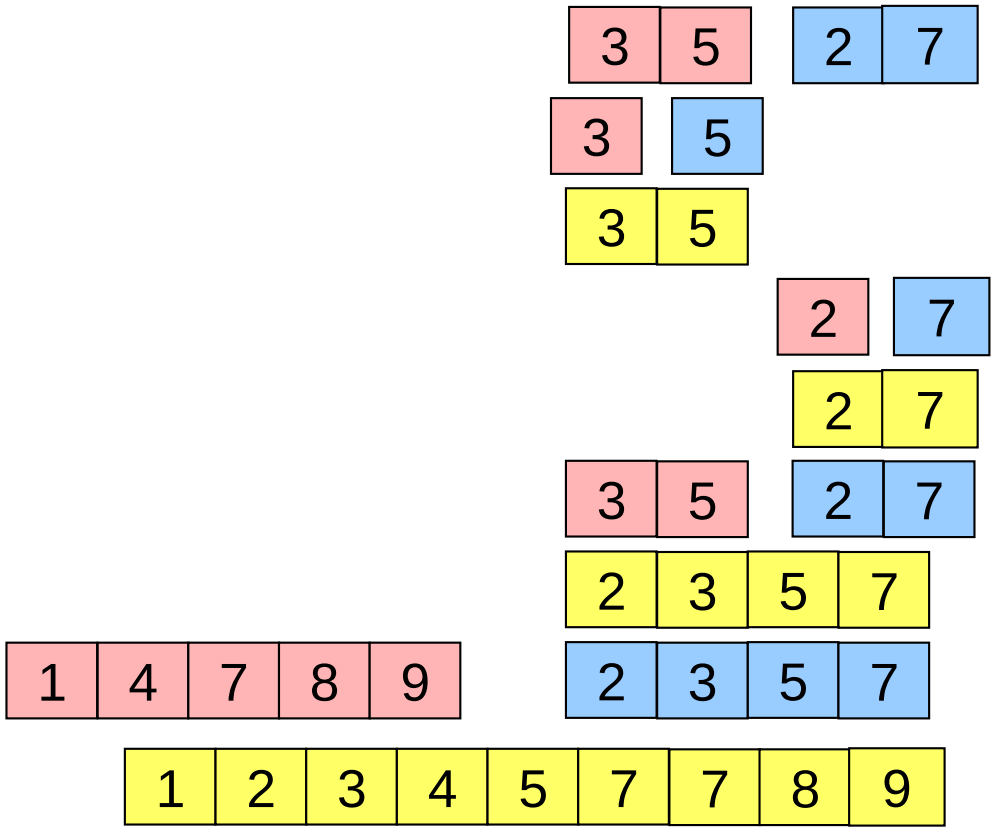
Jadi T3 akan memiliki satu data yaitu 3. Data yang lebih besar yaitu 9 kemudian dibandingkan dengan data kedua dari T1, yaitu 11. Ternyata 9 lebih kecil dari 11, sehingga 9 diletakkan sebagai data kedua dari T3. Demikian seterusnya sehingga didapat hasil sebagai berikut :

3	9	11	12	15	17	20	23	31	35

Merge Sort: contoh



Merge Sort: contoh



Algoritma Merge Sort

1 i ← 0

2 j ← 0

3 J3 ← 0

4 Kerjakan baris 5 sampai dengan 7 selama (i < J1) atau (j < J2)

5 J3 ← J3 + 1

6 Jika (T1[i] < T2[j]) maka T3[J3] ← T1[i], i ← i + 1

7 Jika (T1[i] >= T2[j]) maka T3[J3] ← T2[j], j ← j + 1

8 Jika (i > J1) maka kerjakan baris 9, jika tidak kerjakan baris 15

9 i ← j

10 Selama (i < J2) kerjakan baris 11 sampai dengan 13

11 J3 ← J3 + 1

12 T3[J3] ← T2[i]

13 i ← i + 1

14 Selesai

15 j ← i

16 Selama (j < J1) kerjakan baris 17 sampai dengan 19

17 J3 ← J3 + 1

18 T3[J3] ← T1[j]

19 j ← j + 1

Prosedur Merge Sort

```
void MergeSort(int T1[],int T2[],int J1,int J2, int T3[],int *J3)
{
	int i=0, j=0;

	 int t=0;
	 while ((i<J1)||(j<J2)){
		if(T1[i]<T2[j]){
			T3[t] = T1[i];
			i++;
		}
		else{
			T3[t] = T2[j];
			j++;
		}
		t++;
	 }

		if(i>J1)
			for(i=j; i<J2; i++){
				T3[t] = T2[i];
				t++;
			}

		if(j>J2)
			for(j=i; j<J1; j++){
				T3[t] = T1[j];
				t++;
			}
		*J3 = t;
}
```

Terima Kasih

Pertanyaan?