

SORTING - Merging

Definisi:

$$A = \{a_1, a_2, \dots, a_r\}$$

$$B = \{b_1, b_2, \dots, b_s\}$$

merupakan dua deret angka yang terurut naik; merge A dan B merupakan deret

$$C = \{c_1, c_2, \dots, c_{r+s}\}$$

yang juga terurut naik, sedemikian sehingga setiap c_i pd C berasal dari A atau B dan setiap a_i dan b_i muncul tepat satu kali pada C .

Aplikasi:

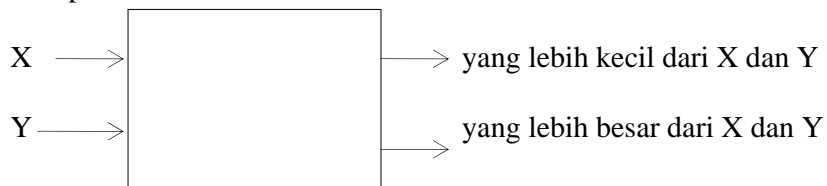
- basis data
- manajemen file

Jika $r = s = n$, algoritma sekuensial memerlukan waktu $O(n)$, sedangkan pada algoritma paralel yang memakai N prosesor diperlukan waktu $\Omega(n/N)$.

JARINGAN UNTUK MERGING

- Merging akan dilakukan oleh sekumpulan prosesor sederhana yang berkomunikasi melalui jaringan *special-purpose*.
- Arsitektur paralel special purpose ini disebut (r, s) -merging network.
- Semua prosesor identik dan disebut *comparator*.

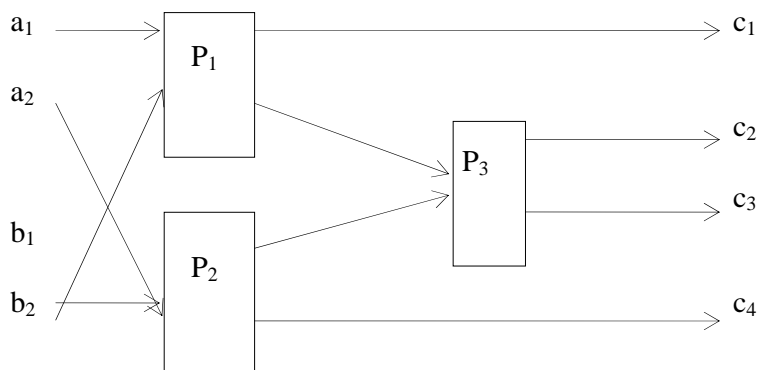
Comparator:



Asumsi:

- kedua deret input berukuran sama, $r = s = n \geq 1$
- n merupakan suatu nilai pangkat 2.

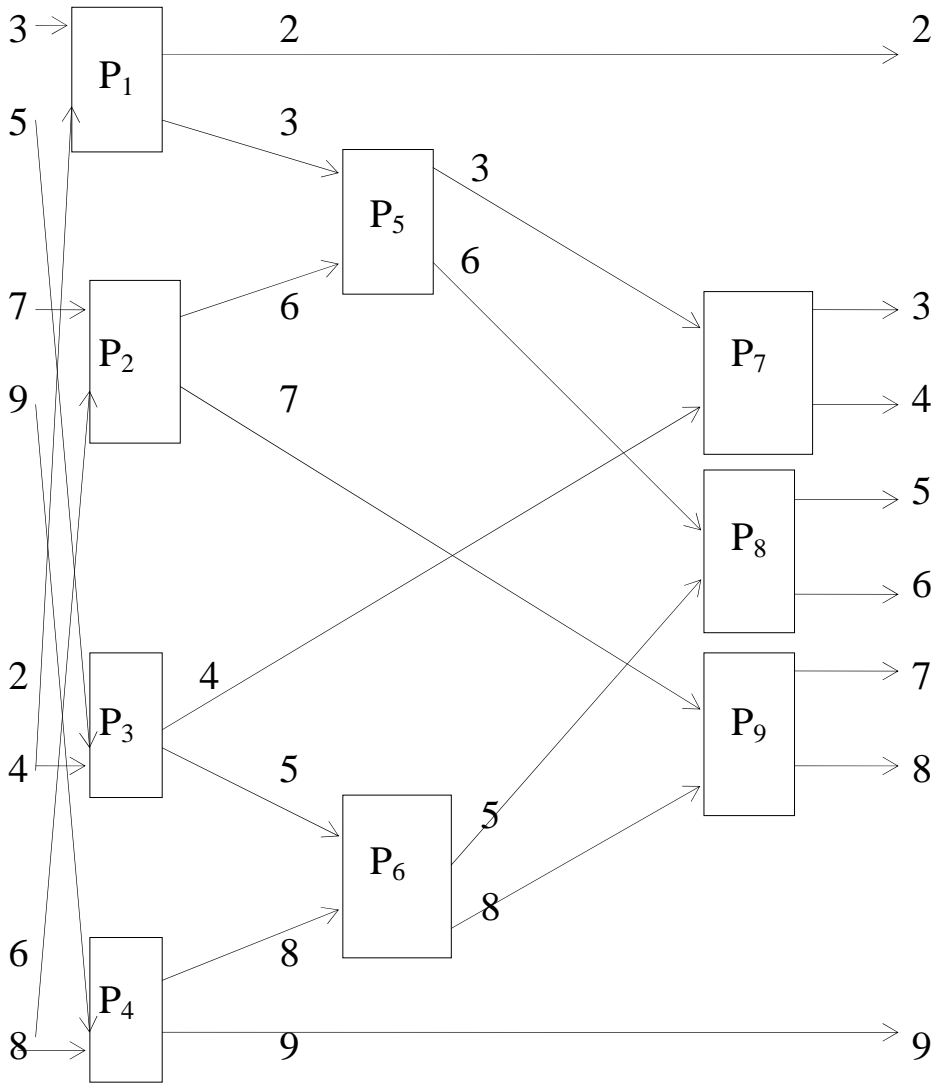
Merging 2 deret yang masing-masing terdiri dari 2 elemen:



- P_1 membandingkan elemen terkecil A dengan elemen terkecil B ; outputnya adalah elemen terkecil C , yaitu c_1 .

- P_2 membandingkan elemen terbesar A dengan elemen terbesar B dan; outputnya adalah elemen terbesar C , yaitu c_4 .
- P_3 berfungsi menghasilkan dua elemen tengah.

Merging 2 deret yang masing-masing memiliki 4 elemen:



Secara umum, (n, n) -merging network didapat dengan konstruksi rekursif berikut:

1. Elemen A dan B bernomor ganjil, yaitu, $\{a_1, a_3, a_5, \dots, a_{n-1}\}$ dan $\{b_1, b_3, b_5, \dots, b_{n-1}\}$, di-merge dengan menggunakan $(n/2, n/2)$ -merging network untuk menghasilkan $\{d_1, d_2, d_3, \dots, d_n\}$.
2. Secara simultan, elemen bernomor genap dari kedua deret tsb, $\{a_2, a_4, a_6, \dots, a_n\}$ dan $\{b_2, b_4, b_6, \dots, b_n\}$, juga di-merge dengan menggunakan $(n/2, n/2)$ -merging network untuk menghasilkan deret $\{e_1, e_2, e_3, \dots, e_n\}$.
3. Deret terakhir $\{c_1, c_2, c_3, \dots, c_{2n}\}$ didapat dari:

$$c_1 = d_1,$$

$$c_{2n} = e_n,$$

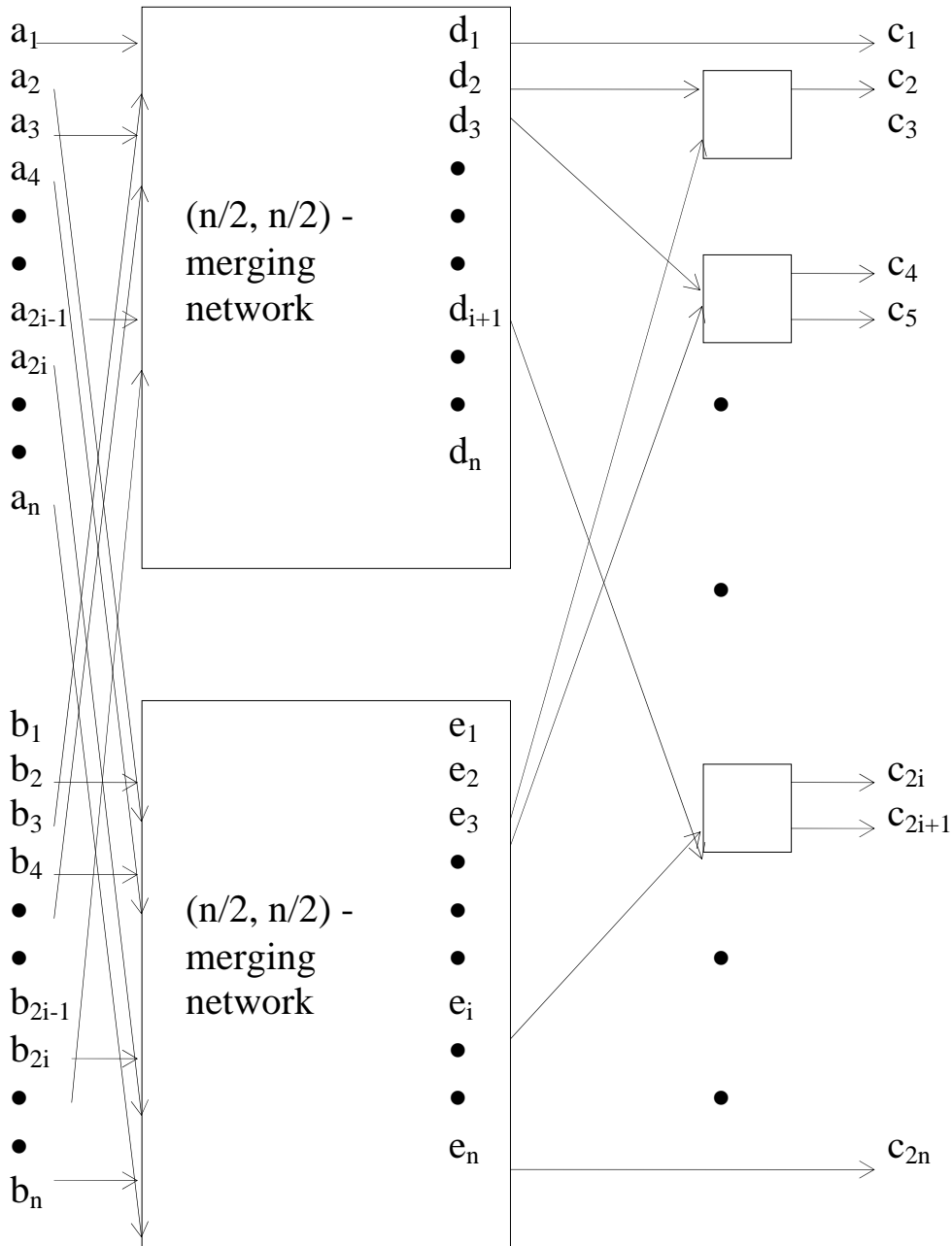
$$c_{2i} = \min(d_{i+1}, e_i), \text{ dan}$$

$$c_{2i+1} = \max(d_{i+1}, e_i), \text{ untuk } i = 1, 2, \dots, n-1.$$

$(n/2, n/2)$ -merging network dibangun dengan menerapkan aturan yang sama secara rekursif, yaitu dengan menggunakan dua $(n/4, n/4)$ -merging network diikuti oleh comparator rank $(n/2)-1$.

Cara ini disebut odd-even merging.

Odd-even merging:



Analisis:

1. Running time

Asumsi:

- sebuah comparator dapat membaca input, melakukan perbandingan, dan menghasilkan output dalam satu satuan waktu.

- $t(2n)$ menunjukkan waktu yang diperlukan oleh (n, n) -merging network untuk melakukan merge dua deret yang panjang masing-masingnya n .
Sifat rekursif jaringan seperti ini menghasilkan:
 $t(2)=1$ untuk $n=1$ (lihat gambar pertama)
 $t(2n) = t(n)+1$ untuk $n>1$ (lihat gambar keempat)
Dengan demikian $t(2n) = 1 + \log n$, jauh lebih lebih cepat dari $O(n)$, yang merupakan waktu terbaik pada komputer sekuensial.
2. Jumlah prosesor
 $p(2n)$ menyatakan jumlah comparator pada (n, n) -merging network.
 $p(2) = 1$ untuk $n=1$ (lihat gambar pertama)
 $p(2n) = 2p(n) + (n-1)$, untuk $n>1$ (lihat gambar keempat)
Dengan demikian $p(2n) = 1 + n \log n$.
 3. Biaya
 $t(2n) = 1 + \log n$ dan $p(2n) = 1 + n \log n$.
Total perbandingan yang dilakukan oleh (n, n) -merging network, yaitu, yang merupakan biaya jaringan, adalah:
$$c(2n) = p(2n) \times t(2n) = O(n \log^2 n)$$

 \therefore Jaringan ini tidak optimal dalam hal biaya karena melakukan lebih banyak operasi dari $O(n)$ yang dibutuhkan untuk merge sekuensial.

Pembahasan:

Properti *merging network* ini: Deret perbandingan telah ditentukan sebelumnya.

Analisis di atas menunjukkan bahwa (n, n) -merging network sangat cepat. Dua deret dengan masing-masing 2^{20} elemen dapat di-merge dalam 21 langkah. Hal yang sama memerlukan lebih dari dua juta langkah pada komputer sekuensial.

Namun, kecepatan tsb didapat dengan prosesor yang sangat banyak. Untuk $n = 2^{20}$, (n, n) -merging network terdiri dari lebih dari dua puluh juta comparator.

Arsitektur jaringan sangat ireguler, penghubung comparator memiliki panjang yang bervariasi dengan n .

\therefore Walaupun secara teoritis bagus, *merging network* tidak praktis jika n bernilai besar.

MERGING PADA MODEL CREW (Concurrent Read, Exclusive Write)

Merging Sekuensial

Dua deret bilangan

$$A = \{a_1, a_2, \dots, a_r\}$$

$$B = \{b_1, b_2, \dots, b_s\}$$

dengan urutan naik di-merge membentuk deret C yang juga urut naik.

- Merging dilakukan oleh satu prosesor.
- Digunakan dua pointer, 1 untuk setiap deret.
- Dibuat dua elemen fiktif a_{r+1} dan b_{s+1} yang bernilai tak hingga.

procedure SEQUENTIAL MERGE (A, B, C)

Step 1: (1.1) $i \leftarrow 1$

(1.2) $j \leftarrow 1$

```

Step 2: for  $k = 1$  to  $r + s$  do
    if  $a_i < b_j$  then (i)  $c_k \leftarrow a_i$ 
                    (ii)  $i \leftarrow i+1$ 
    else (i)  $c_k \leftarrow b_j$ 
        (ii)  $j \leftarrow j+1$ 
    end if
end for

```

- Jumlah perbandingan: $r + s$
- Worst case: $r = s = n$, algoritma berjalan dalam waktu $O(n)$.
- Procedure ini optimal.

Merging Paralel

- Dikerjakan oleh komputer CREW SM SIMD (Concurrent Read, Exclusive Write; Shared Memory; Single Instruction stream, Multiple Data stream).
- $r \leq s$
- Digunakan N prosesor: P_1, P_2, \dots, P_N .
- $N \leq r$
- Worst case: $r = s = n$; waktu $O((n/N) + \log n)$
- Optimal biaya untuk $N \leq n / \log n$.

Setiap prosesor dianggap mampu melakukan dua prosedur sekuensial berikut:

1. Prosedur SEQUENTIAL MERGE
2. Prosedur BINARY SEARCH:
 - Mengambil deret input $S = \{s_1, s_2, \dots, s_n\}$ bilangan yg terurut naik dan suatu bilangan x .
 - Jika x ada di S , prosedur mengembalikan indeks k dari elemen s_k pd S sedemikian shg $x = s_k$.
 - Jika tidak, prosedur memberi nilai nol.
 - Pd setiap tahap, dilakukan perbandingan antara x dan elemen S .
 - Jika keduanya sama, prosedur berhenti atau setengah elemen pada deret diabaikan.
 - proses diteruskan sampai jumlah elemen yg tertinggal adalah 0 atau 1.

procedure BINARY SEARCH (S, x, k)

```

Step 1:      (1.1)       $i \leftarrow 1$ 
             (1.2)       $h \leftarrow n$ 
             (1.3)       $k \leftarrow 0$ 
Step 2: while  $i \leq h$  do
             (2.1)       $m \leftarrow \lfloor (i + h)/2 \rfloor$ 
             (2.2)      if  $x = s_m$       then (i)  $k \leftarrow m$ 
                               (ii)  $i \leftarrow h + 1$ 
             else if  $x < s_m$  then  $h \leftarrow m-1$ 
                               else  $i \leftarrow m + 1$ 
             end if
             end if
end while

```

procedure CREW MERGE (A, B, C)

Step 1: {Pilih $N - 1$ elemen A yg membagi deret menjadi N subsequence dgn ukuran yg kurang lebih sama. Namai subsequence yg dibentuk oleh $N-1$ elemen ini A' . Subsequence B' dari $N-1$ elemen B juga dipilih dgn cara yg sama. Step ini dilakukan sbb:}

for $i = 1$ to $N-1$ **do in parallel**

Processor P_i menentukan a_i' dan b_i' dari

$$(1.1) a_i' \leftarrow a_{\lceil i r/N \rceil}$$

$$(1.2) b_i' \leftarrow a_{\lceil i s/N \rceil}$$

end for

Step 2: {Merge A' dan B' menjadi deret triple $V = \{v_1, v_2, \dots, v_{2N-2}\}$, di mana setiap triple terdiri dari elemen A' atau B' diikuti posisinya pada A' atau B' diikuti oleh nama deret asalnya, yaitu, A atau B . Hal ini dilakukan sbb:}

(2.1) for $i = 1$ to $N-1$ **do in parallel**

(i) Prosesor P_i memakai BINARY SEARCH pd B' untuk menemukan j terkecil sedemikian sehingga $a_i' < b_j'$.

(ii) if j exists then $v_{i+j-1} \leftarrow (a_i', i, A)$
else $v_{i+N-1} \leftarrow (a_i', i, A)$

end if

end for

(2.2) for $i = 1$ to $N-1$ **do in parallel**

(i) Prosesor P_i memakai BINARY SEARCH pada A' untuk menemukan j terkecil sehingga $b_i' < a_j'$

(ii) if j exist then $v_{i+j-1} \leftarrow (b_i', i, B)$
else $v_{i+N-1} \leftarrow (b_i', i, B)$

end if

end for

Step 3: {Setiap prosesor merge dan menyisipkan ke dalam C , elemen-elemen dari kedua subsequence, satu dari A dan satu dari B . Indeks kedua elemen (satu di A dan satu di B) di mana setiap prosesor mulai merging, terlebih dulu dihitung dan disimpan pada array Q yg menyimpan pasangan terurut. Langkah ini dilakukan sbb:}

$$(3.1) Q(1) \leftarrow (1, 1)$$

(3.2) for $i = 2$ to N **do in parallel**

if $v_{2i-2} = (a_k', k, A)$ then prosesor P_i

(i) memakai BINARY SEARCH atas B untuk menemukan j terkecil sedemikian shg $b_j < a_k'$.

$$(ii) Q(i) \leftarrow (k \lceil r/N \rceil, j)$$

else prosesor P_i

(i) memakai BINARY SEARCH atas A untuk menemukan j terkecil sedemikian shg $a_j < b_k'$.

$$(ii) Q(i) \leftarrow (j, k \lceil r/N \rceil)$$

end if

end for

- (3.3) for $i = 1$ to N **do in parallel**
 Prosesor P_i memakai SEQUENTIAL MERGE dan $Q(i) = (x, y)$ untuk merge dua subsequence, satu mulai pd a_x dan yg lain pd b_y dan menempatkan hasil merge pd array C mulai pd posisi $x + y - 1$. Merge diteruskan sampai
- (i) elemen yg lebih besar atau sama dgn komponen pertama dari v_{2i} ditemukan masing-masing pd A dan B (ketika $i \leq N-1$)
 - (ii) tidak ada elemen yg tersisa pd A atau B (ketika $i = N$)
- end for.

Ada dua hasil observasi dari algoritma di atas:

- Jika a_i dibandingkan dgn b_j dan $a_i = b_j$, maka algoritma memutuskan bhw a_i lebih kecil.
- Operasi concurrent read dilakukan ketika prosedur BINARY SEARCH dipanggil, yaitu pd step 2.1, 2.2, dan 3.2. Di sini, beberapa prosesor mengeksekusi binary search pd deret yg sama.

Analisis

Step 1: Dgn semua prosesor bekerja paralel, setiap prosesor menghitung dua subscript. Dgn demikian langkah ini memerlukan waktu konstan.

Step 2: Langkah ini terdiri dari dua aplikasi prosedur BINARY SEARCH pada deret dgn panjang $N-1$, masing-masing diikuti oleh statement assignment. Ini memerlukan waktu $O(\log N)$.

Step 3: Step 3.1 terdiri dari assignment waktu konstan, dan step 3.2 membutuhkan paling banyak waktu $O(\log s)$. Untuk menganalisa step 3.3, pertama kita meneliti bahwa V terdiri dari $2N-2$ elemen yg membagi C menjadi $2N-1$ subsequence dgn ukuran maksimum sama dgn $(\lceil r/N \rceil + \lceil s/N \rceil)$. Ukuran maksimum ini terjadi jika, misalnya, satu elemen a_i dari A sama dgn elemen b_j dari B ; maka $\lceil r/N \rceil$ elemen yg lebih kecil atau sama dgn a_i (dan lebih besar atau sama dgn a_{i-1}) juga lebih kecil atau sama dgn b_j , dan dgn cara yg sama, $\lceil s/N \rceil$ elemen yg lebih kecil atau sama dgn b_j (dan lebih besar atau sama dgn b_{j-1}) juga lebih kecil atau sama dgn a_i . Pd step 3 setiap prosesor membentuk dua subsequence spt ini dari C yg ukuran totalnya tidak lebih besar dari $2(\lceil r/N \rceil + \lceil s/N \rceil)$, kecuali P_N , yg hanya membuat satu subsequence C . berarti prosedur SEQUENTIAL MERGE paling banyak memerlukan waktu $O((r+s)/N)$.

Worst case, $r = s = n$, dan karena $n \geq N$, waktu jalan algoritma didominasi oleh waktu yg dibutuhkan oleh step 3. Dgn demikian $t(2n) = O((n/N) + \log n)$

Karena $p(2n) = N$, $c(2n) = p(2n) \times t(2n) = O(n + N \log n)$, dan algoritma optimal biaya ketika $N \leq n/\log n$.