ALGORITMA SORTING

Sorting memiliki kepentingan tambahan bagi perancang algoritma paralel; ia sering digunakan untuk melakukan permutasi data umum pada komputer dengan memori terdistribusi. Operasi pemidahan data ini dpt digunakan untuk menyelesaikan masalah pada:

- teori graf
- geometri komputasional
- image processing

dalam waktu optimal atau hampir optimal.

Algoritma yang akan dipelajari berikut merupakan internal sort - yaitu, tabel yang di-sort cukup kecil untuk masuk seluruhnya di memori primer. Semua algoritma berikut ini juga men-sort dengan membandingkan sepasang elemen.

ENUMERATION SORT

Asumsi:

- Tabel dengan n elemen: $a_0, a_1, ..., a_{n-1}$
- Untuk 2 elemen a_i dan a_j, salah satu kondisi ini pasti benar: $a_i < a_j$, $a_i = a_j$, atau $a_i > a_j$.
- Tujuan sorting: menemukan permutasi $(\pi_0, \pi_1, ..., \pi_{n-1})$ sedemikian sehingga $a_{\pi 0} \le a_{\pi 1} \le ... \le a_{\pi n-1}$.

Enumeration sort:

- menghitung posisi akhir setiap elemen pada list yang di-sort dan membandingkannya dengan elemen lain dan menghitung jumlah elemen yang memiliki nilai lebih kecil.
- Jika j elemen memiliki nilai lebih kecil dari a_i, maka $\pi_j = i$; yaitu, elemen ai adalah $(j + 1)$ elemen pada list yang di-sort setelah $a_{\pi 0}, ..., a_{\pi j-1}$.
- Jika diberikan n^2 prosesor, model CRCW PRAM dapat menghitung setiap pasang elemen dan menghitung posisi list setiap elemen dalam waktu konstan. Begitu mesin telah menghitung posisi setiap elemen, diperlukan satu langkah lagi untuk memindahkan elemen ybs ke lokasi yang telah urut.

ENUMERATION SORT (SPECIAL CRCW PRAM):

```
Parameter   n                   {number of elements}
Global      a[0...(n-1)]        {elements to be sorted}
            position[0...(n-1)] {sorted positions}
            sorted[0...(n-1)]   {Contains sorted elements}
begin
    spawn(P_{i,j}, for all 0 ≤ i, j < n)

    for all P_{i,j}, where 0 ≤ i, j < n do
        position[i] ← 0

        if a[i] < a[j] or (a[i] = a[j] and i < j) then
            position[i] ← 1
        endif
    endfor
```

```
        for all P_{i,0}, where 0 ≤ i < n do
                sorted[position[i]] ← a[i]
        endfor
end
```

Satu set n elemen dapat di-sort dalam waktu $\Theta(\log n)$ dengan n^2 prosesor, jika dipakai model PRAM CRCW dengan write simultan ke lokasi memori yang sama menyebabkan jumlah nilai di-assign. Jika waktu yang diperlukan untuk spawn prosesor tidak dihitung, algoritma berjalan dengan waktu konstan.

BATAS BAWAH UNTUK PARALLEL SORTING

Teorema 1:
Anggap ada n elemen yang akan di-sort pada array prosesor yang disusun menjadi mesh satu dimensi. Juga asumsikan bahwa sebelum dan sesudah sort, elemen akan didistribusikan merata, satu elemen per prosesor. Dengan demikian, batas bawah kompleksitas waktu pada algoritma sorting yang mana pun adalah $\Theta(n)$.

Bukti:
Lebar biseksi dari jaringan mesh satu dimensi adalah 1. Misalkan posisi yang ter-sort dari semua elemen yang pada awalnya ada pada satu sisi biseksi adalah pada sisi biseksi yang lain, dan sebaliknya. Maka seluruh n elemen harus melewati satu link untuk mencapai sisi yang lain. Karena link hanya dapat membawa satu elemen sekali, jumlah langkah yang diperlukan untuk menukar elemen melalui biseksi paling tidak adalah sebesar n. Dengan demikian, batas bawah kompleksitas jaringan mesh satu dimensi dengan syarat-syarat tersebut adalah $\Theta(n)$.

Teorema 2:
Anggap ada n elemen yang akan di-sort pada array prosesor yang tersusun sebagai mesh dua dimensi. Juga anggap bahwa sebelum dan sesudah sort, elemen-elemen tsb terdistribusi merata, satu elemen per prosesor. Maka, batas bawah kompleksitas waktu untuk algoritma sorting yang mana pun adalah $\Theta(\sqrt{n})$.

Bukti:
Lebar biseksi jaringan mesh dua dimensi dengan n node adalah lebih kecil atau sama dengan $\lceil \sqrt{n} \rceil$. Misalkan posisi ter-sort dari semua elemen yang pada awalnya ada pada satu sisi biseksi adalah di sisi lain biseksi tsb, dan sebaliknya. Maka seluruh n elemen harus melalui salah satu dari tidak lebih $\lceil \sqrt{n} \rceil$ link untuk mencapai sisi lainnya. Karena satu link hanya dapat membawa satu elemen sekali, jumlah langkah yang diperlukan untuk menukar elemen melintasi biseksi paling tidak adalah $n/\lceil \sqrt{n} \rceil$. Dengan demikian, batas bawah kompleksitas array prosesor bagaimanapun yang disusun sebagai mesh dua dimensi dan berjalan dengan ketentuan yang telah diberikan di atas adalah $\Theta(n/\lceil \sqrt{n} \rceil) = \Theta(\sqrt{n})$

Teorema 3:
Anggap ada $n = 2^k$ elemen yang akan di-sort pada array prosesor yang tersusun sebagai jaringan shuffle-exchange. Juga anggap bahwa sebelum dan sesudah sort, elemen terdistribusi merata, satu elemen per prosesor. Batas bawah untuk algoritma sort mana pun adalah $\Theta(\log n)$.

Bukti:
Misalkan posisi ter-sort elemen yang pada awalnya berada pada node 0 adalah node n-1. Pemindahan elemen tsb dari node 0 ke node n-1 menuntut paling tidak log n operasi pertukaran dan paling tidak log n-1 operasi shuffle. Dengan alasan ini, batas bawah pada algoritma sorting berbassis shuffle-exchange dengan batas-batas di atas adalah $\Theta(\log n)$.

ODD-EVEN TRANSPOSITION SORT

Odd-even transposition sort dirancang untuk model array prosesor dengan elemen-elemen processing yang disusun menjadi mesh satu dimensi.

Anggap $A = (a_0, a_1, ..., a_{n-1})$ adalah himpunan n elemen yang akan di-sort. Setiap n elemen processing berisi dua variabel lokal: a, elemen unik dari array A, dan t, variabel yang berisi nilai yang diambil dari elemen processing tetangganya.

Algoritma melakukan $n/2$ iterasi, dan setiap iterasi memiliki dua fase:
1. **odd-even exchange**: nilai a pada setiap prosesor bernomer ganjil (kecuali prosesor n-1) dibandingkan dengan nilai a yang tersimpan di prosesor berikutnya. Nlai-nilai ditukar, jika perlu, sehingga prosesor dengan nomer lebih kecil berisi nilai yang lebih kecil.
2. **even-odd exchange**: nilai a pada setiap prosesor bernomer genap dibandingkan dengan nilai a yang tersimpan di prosesor berikutnya. Nlai-nilai ditukar, jika perlu, sehingga prosesor dengan nomer lebih kecil berisi nilai yang lebih kecil.

Setelah $n/2$ iterasi, nilai-nilai harus ter-sort.

ODD-EVEN TRANSPOSITION SORT (ONE-DIMENSIONAL MESH PROCESSOR ARRAY):

```
Parameter   n
Global i
Local  a    {element to be sorted}
       t    {element taken from adjacent processor}
begin
       for i ← 1 to n/2 do
              for all Pj, where 0 ≤ j ≤ n-1 do
                     if j < n-1 and odd(j) then      {Odd-even exchange}
                        t ⇐ successor(a)             {Get value from successor}
                        successor(a)⇐max(a,t)        {Give away larger val}
                        a ← min(a,t)                 {Keep smaller value}
                     endif
                     if even(j) then                 {Even-odd exchange}
                        t ⇐ successor(a)             {Get value from successor}
                        successor(a)⇐max(a,t)        {Give away larger val}
                        a ← min(a,t)                 {Keep smaller value}
                     endif
              endfor
       endfor
end
```

Odd-even transposition sort dari delapan nilai pada model array prosesor mesh satu dimensi:

Indeks:	0	1	2	3	4	5	6	7
Nilai awal:	G	H	F	D	E	C	B	A
Setelah odd-even exchange:	G	F	<H	D	<E	B	<C	A
Setelah even-odd exchange:	F	<G	D	<H	B	<E	A	<C
Setelah odd-even exchange:	F	D	<G	B	<H	A	<E	C
Setelah even-odd exchange:	D	<F	B	<G	A	<H	C	<E
Setelah odd-even exchange:	D	B	<F	A	<G	C	<H	E
Setelah even-odd exchange:	B	<D	A	<F	C	<G	E	<H
Setelah odd-even exchange:	B	A	<D	C	<F	E	<G	H
Setelah even-odd exchange:	A	<B	C	<D	E	<F	G	<H

Teorema:
Kompleksitas sorting n elemen pada array prosesor mesh satu dimensi dengan n prosesor menggunakan odd-even transposition sort adalah $\Theta(n)$.

Bukti:
Bukti berdasar fakta bahwa setelah i iterasi loop for luar, tidak ada elemen yang bisa lebih jauh dari $n - 2i$ posisi dari posisi akhir dan ter-sortnya. Dengan demikian, $n/2$ iterasi cukup untuk men-sort elemen-elemen tsb, dan kompleksitas waktu algoritma paralel adalah $\Theta(n)$, jika diberikan n elemen processing.