ANALISIS ALGORITME
Pengantar
Teknik Divide and Conquer

Kuliah ke-8

Terdapat 2 aspek dalam algoritma :

- Analisis:
  - Kriteria kebaikan algoritma
  - Kompleksitas algoritma → laju pertumbuhan fungsi
- Design:
  - teknik-teknik perancangan algoritma
    - *Divide and conquer*
    - *Dynamic programming*
    - *Greedy*
    - *Backtracking*
    - *Graph*

Teknik *Divide and Conquer*

- *Divide :*
  - Membagi masalah menjadi sub-sub masalah yang lebih kecil sehingga lebih mudah diselesaikan
- *Conquer :*
  - Menyelesaikan sub-sub masalah tersebut secara rekursif
- *Combine:*
  - Solusi masalah awal dibentuk dari solusi-solusi sub masalah (original problem)

3

```
function Solution(I);
begin
    if size(I) ≤ small_size then
        Solution := DirectSolution(I);
    else
        Decompose(I, I_1, I_2,…,I_k);
        for i := 1 to k do
            Si := Solution(I_i)
        end {for}
        Solution := Combine(S_1,S_2,…,S_k)
    end {if}
end {Solution}
```

4

Andaikan:

- **S(n)** : banyaknya langkah yang dikerjakan **Direct Solution**
- **D(n)** : banyaknya langkah yang dikerjakan **Decompose**
- **C(n)** : banyaknya langkah yang dikerjakan **Combine**

untuk ukuran input = n

maka bentuk umum relasi berulang yang menggambarkan banyaknya langkah yang dilakukan algoritma di atas adalah :

$$T(n) = \begin{cases} S(n) \quad ; \quad untuk \quad n \le smallsize \\ D(n) + \sum_{i=1}^{k} T(size(I_i)) + C(n) \quad ; untuk \ n \ selainnya \end{cases}$$

Contoh beberapa masalah yang dapat diselesaikan menggunakan teknik D&C

- Masalah *Binary Search*:

  Mencari sebuah elemen dalam *list*. Pertama, dicek elemen di tengah *list*, dibandingkan dengan elemen yang dicari. Jika sama, selesai. Jika tidak sama, pada bagian yang bersesuaian dikerjakan *binary search.* Pada setiap tahap dari 2 sub masalah, hanya dikerjakan satu saja.

- Masalah *Merge Sort*:

  Mengurutkan elemen dalam *list* dengan menggunakan teknik *merge sort*. Mula-mula *list* dipecah menjadi dua bagian. Secara rekursif, pada setiap bagian diurutkan dengan *merge sort*. Setelah diurutkan, dua bagian tersebut digabungkan kembali.

Binary search: target x = 70

	1	2	3	4	5	6	7	8	9	10	11	12
v	12	15	33	35	42	45	51	62	73	75	86	98

L: 1

Mid: 6

R: 12

v(Mid) <= x

So throw away the left half…

Binary search: target x = 70

	1	2	3	4	5	6	7	8	9	10	11	12
v	12	15	33	35	42	45	51	62	73	75	86	98

L: 6

Mid: 9

R: 12

x < v(Mid)

So throw away the right half…

Binary search: target x = 70

	1	2	3	4	5	6	7	8	9	10	11	12
v	12	15	33	35	42	45	51	62	73	75	86	98

L: 6

Mid: 7

R: 9

v(Mid) <= x

So throw away the left half…

Binary search: target x = 70

v	12	15	33	35	42	45	51	62	73	75	86	98
	1	2	3	4	5	6	7	8	9	10	11	12

L: 7

Mid: 8

R: 9

v(Mid) <= x

So throw away the left half…

Binary search: target x = 70

v	12	15	33	35	42	45	51	62	73	75	86	98
	1	2	3	4	5	6	7	8	9	10	11	12

L: 8

Mid: 8

R: 9

Done because

R-L = 1

```
function L = BinarySearch(a,x)
% x is a row n-vector with x(1) < ... < x(n)
% where x(1) <= a <= x(n)
% L is the index such that x(L) <= a <= x(L+1)

L = 1;  R = length(x);
% x(L) <= a <= x(R)
while R-L > 1
    mid = floor((L+R)/2);
    % Note that mid does not equal L or R.
    if a < x(mid)
        % x(L) <= a <= x(mid)
        R = mid;
    else
        % x(mid) <= a <== x(R)
        L = mid;
    end
end
```


Binary search is efficient, but how do we sort a vector in the first place so that we can use binary search?

- Many different algorithms out there...
- Let's look at **merge sort**
- An example of the "divide and conquer" approach

Cara kerja *Merge sort :*

Divide:

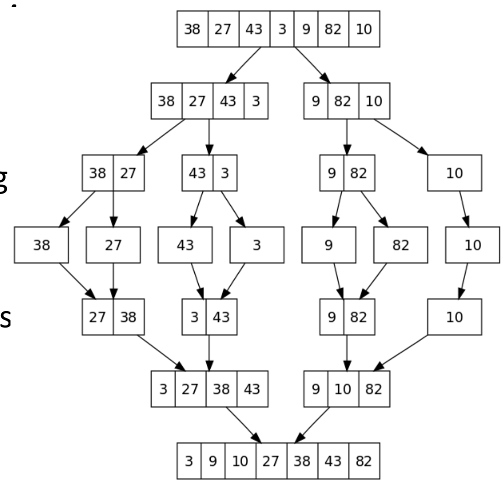
membagi sekuens dg n elemen menjadi 2 subsekuens masing-masing dgn n/2 elemen.

Conquer:

mengurutkan 2 subsekuens tersebut secara rekursif menggunakan *merge sort*.

Combine:

menggabungkan 2 subsekuens yang telah diurutkan untuk menghasilkan output yang dikehendaki.



15

Subdivide the sorting task

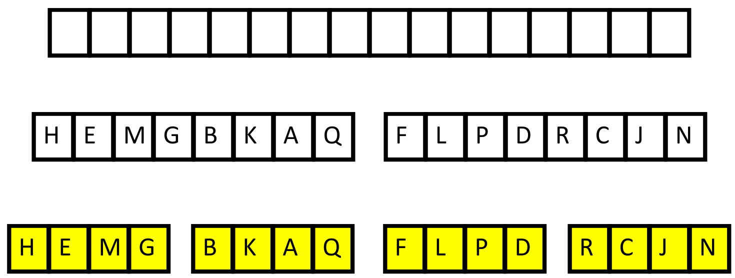
H	E	M	G	B	K	A	Q	F	L	P	D	R	C	J	N

H	E	M	G	B	K	A	Q

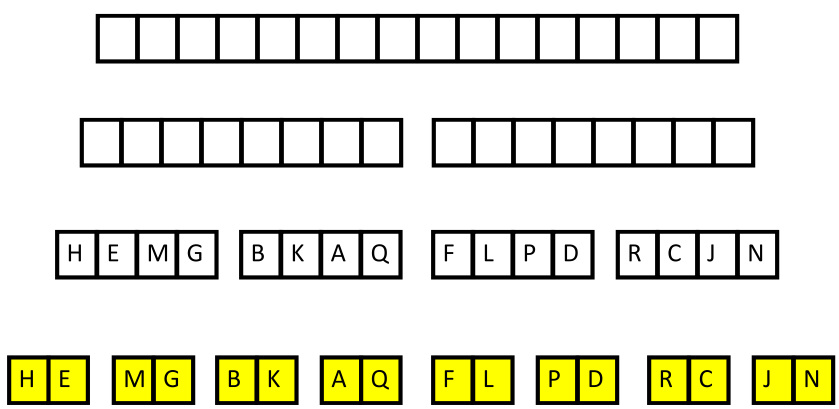
F	L	P	D	R	C	J	N

Insight Through Computing

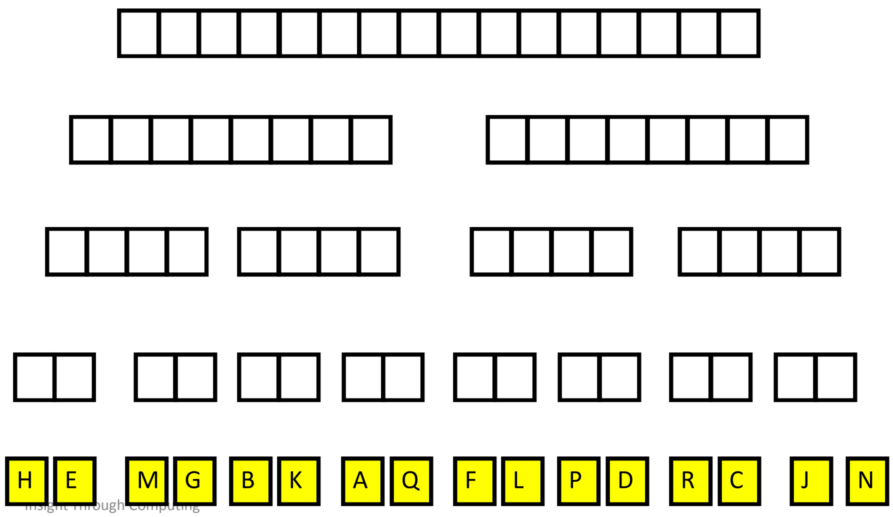
Subdivide again



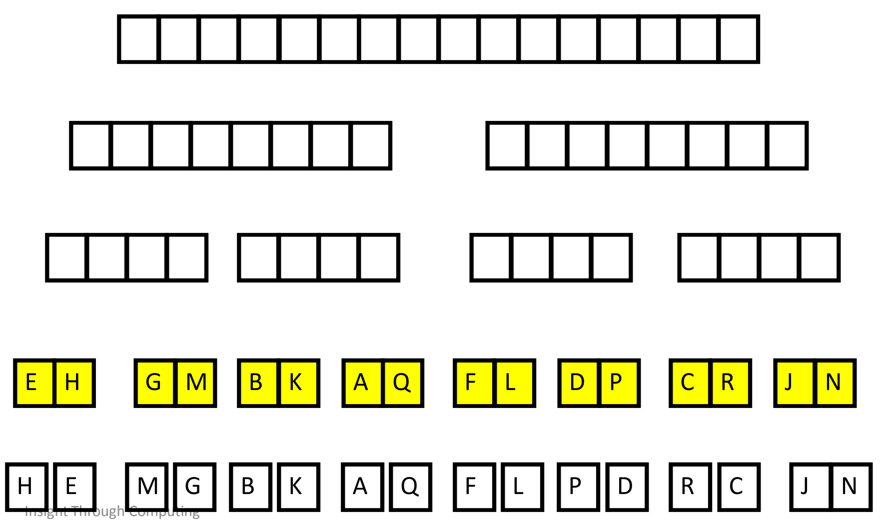
And again



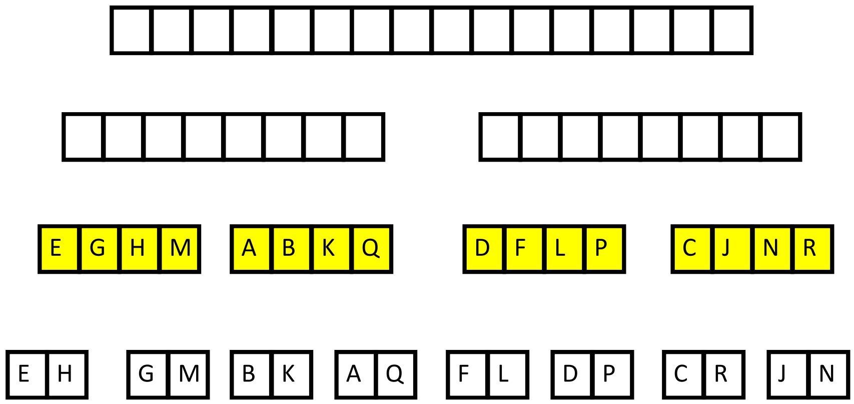
And one last time



Now merge

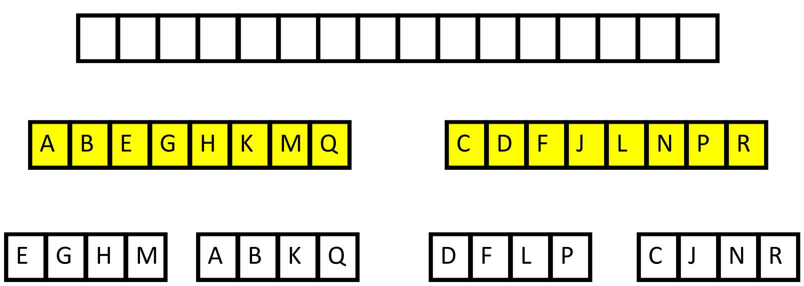


And merge again



Insight Through Computing

And again



Insight Through Computing

And one last time

A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R

A	B	E	G	H	K	M	Q

C	D	F	J	L	N	P	R

Done!

A	B	C	D	E	F	G	H	J	K	L	M	N	P	Q	R

```
function y = mergeSort(x)
% x is a vector.  y is a vector
% consisting of the values in x
% sorted from smallest to largest.

n = length(x);
if n==1
      y = x;
else
      m  = floor(n/2);
      yL = mergeSortL(x(1:m));
      yR = mergeSortR(x(m+1:n));
      y  = merge(yL,yR);
end
```

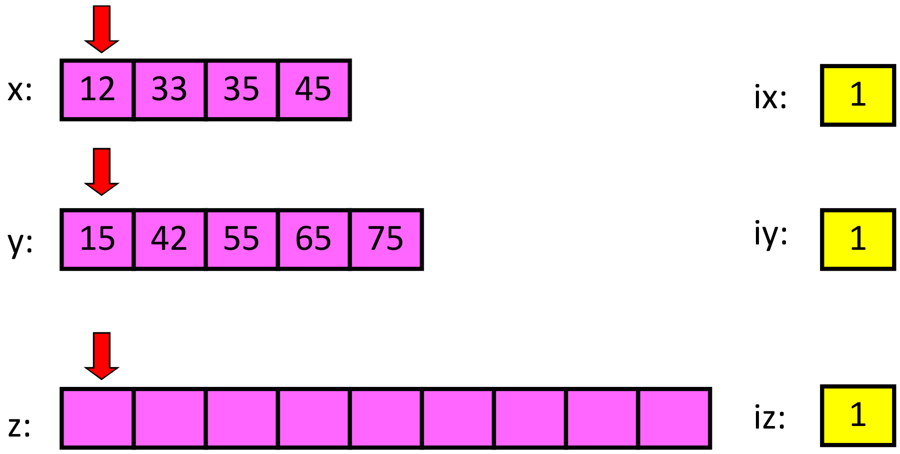

The central sub-problem is the merging of two sorted arrays into one single sorted array

12	33	35	45

15	42	55	65	75

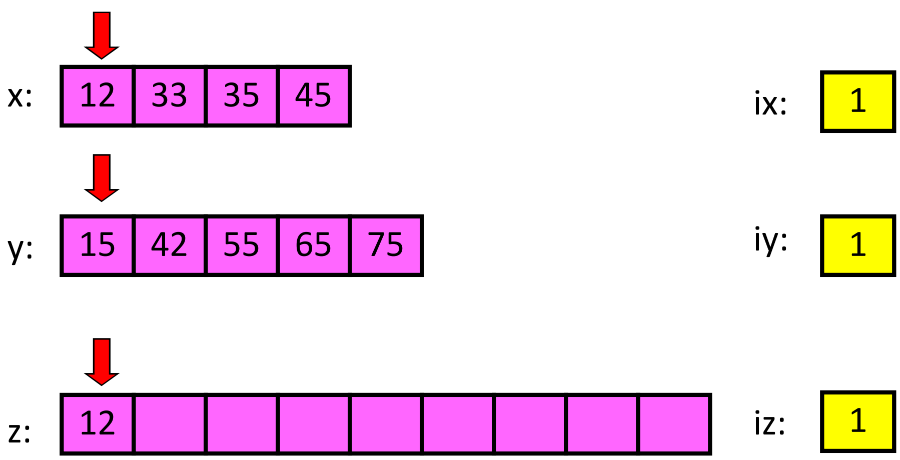
12	15	33	35	42	45	55	65	75

Merge



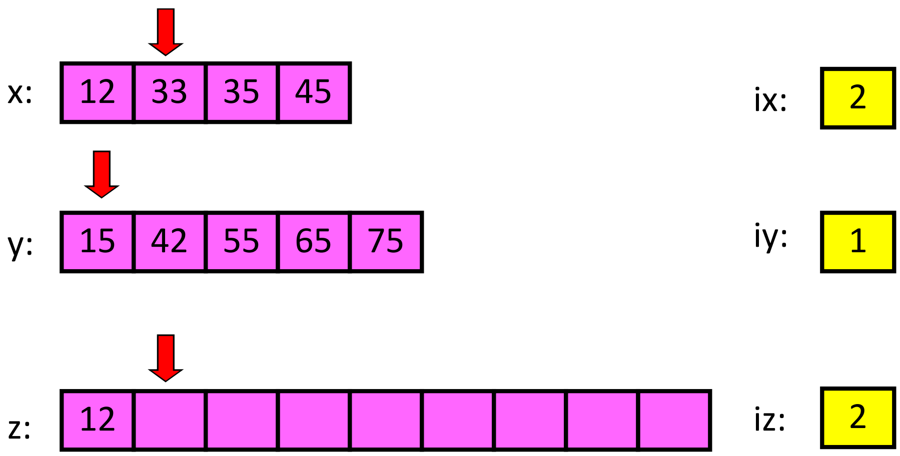
ix<=4 and iy<=5: x(ix) <= y(iy) ???

Merge



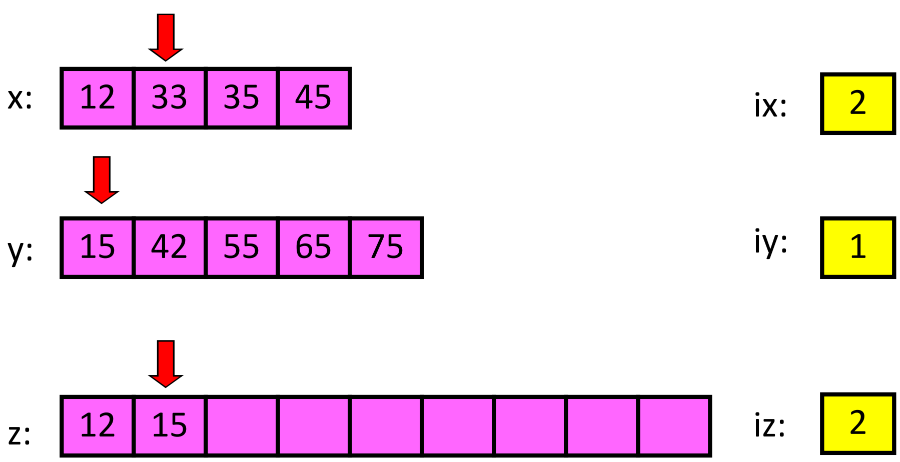
ix<=4 and iy<=5: x(ix) <= y(iy) YES

Merge



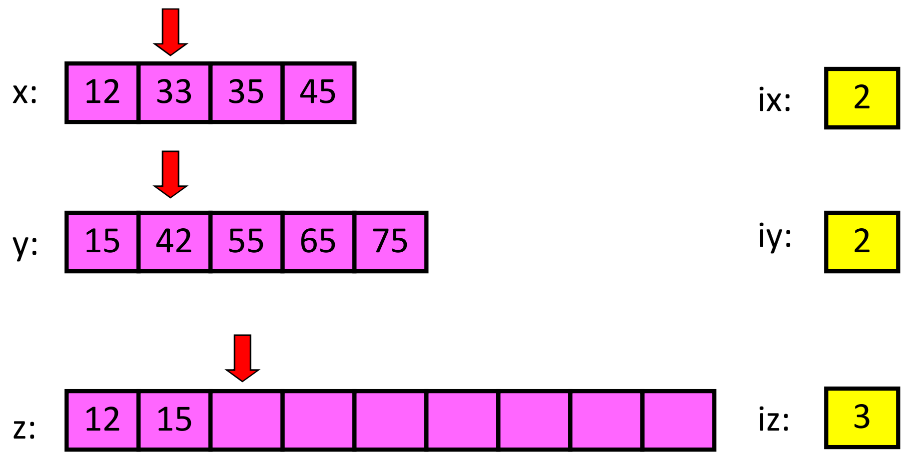
ix<=4 and iy<=5: x(ix) <= y(iy) ???

Merge



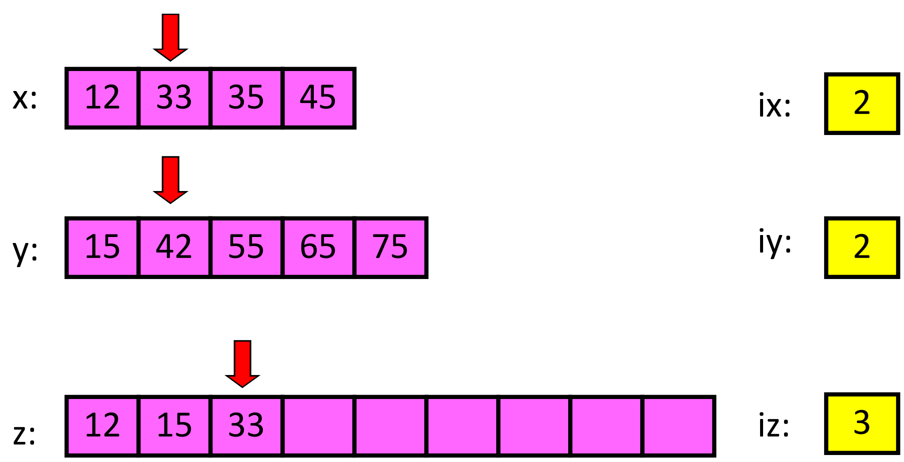
ix<=4 and iy<=5: x(ix) <= y(iy) NO

Merge



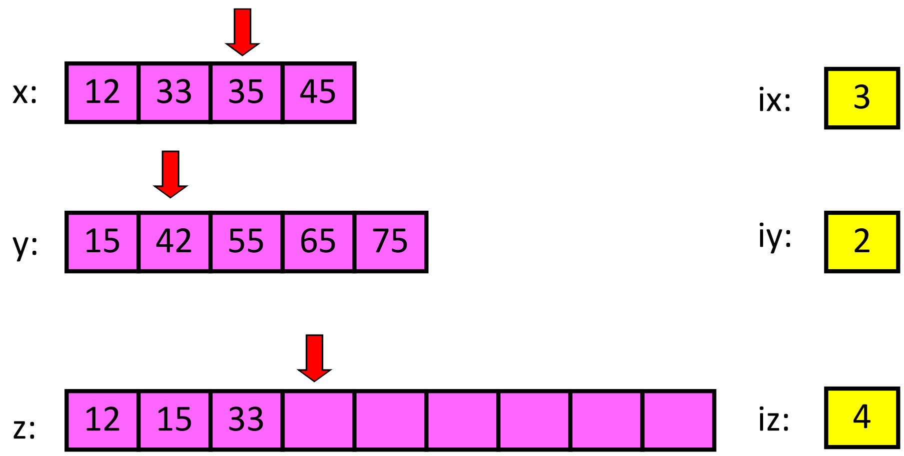
ix<=4 and iy<=5: x(ix) <= y(iy) ???

Merge



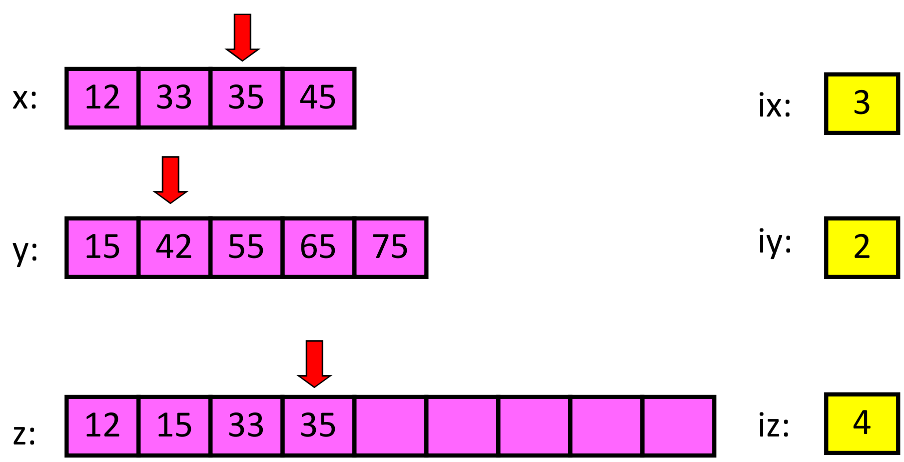
ix<=4 and iy<=5: x(ix) <= y(iy) YES

Merge



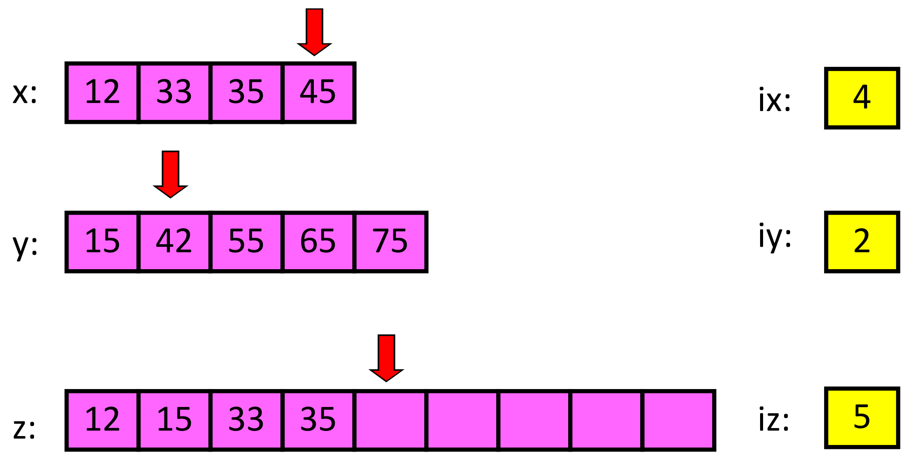
ix<=4 and iy<=5: x(ix) <= y(iy) ???

Merge



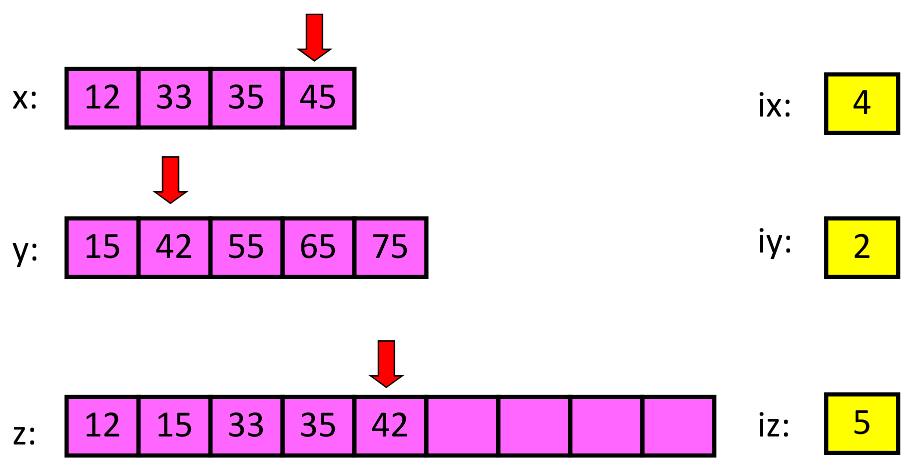
ix<=4 and iy<=5: x(ix) <= y(iy) YES

Merge



ix<=4 and iy<=5: x(ix) <= y(iy) ???

Merge



ix<=4 and iy<=5: x(ix) <= y(iy) NO

Merge

x: | 12 | 33 | 35 | 45 |

y: | 15 | 42 | 55 | 65 | 75 |

z: | 12 | 15 | 33 | 35 | 42 | | | | |

ix: 4

iy: 3

iz: 5

ix<=4 and iy<=5: x(ix) <= y(iy) ???

Merge

x: | 12 | 33 | 35 | 45 |

y: | 15 | 42 | 55 | 65 | 75 |

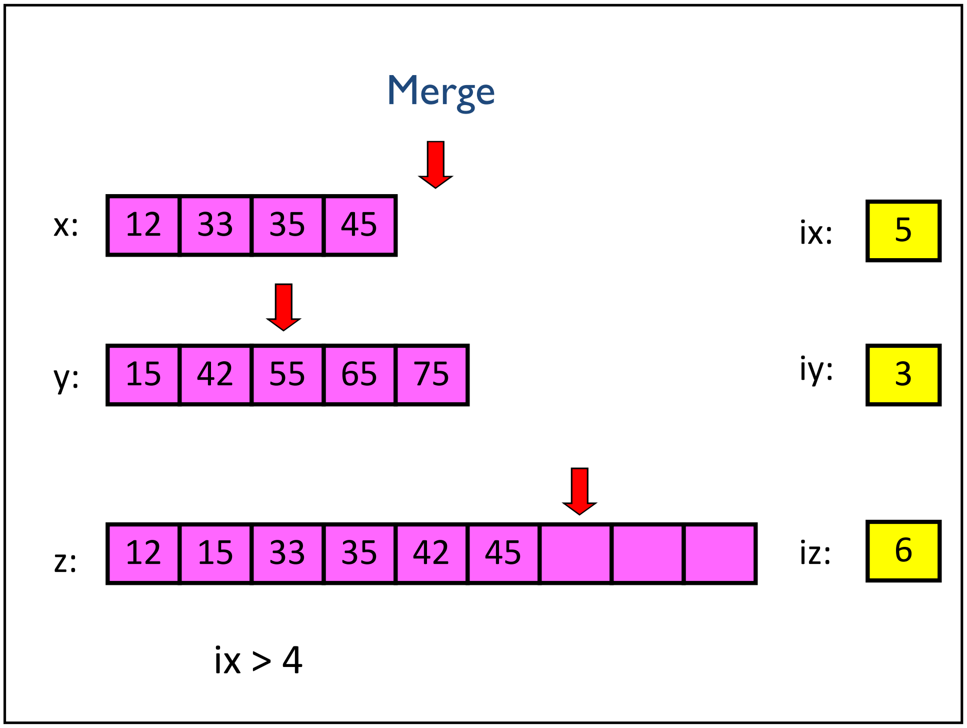
z: | 12 | 15 | 33 | 35 | 42 | 45 | | | |

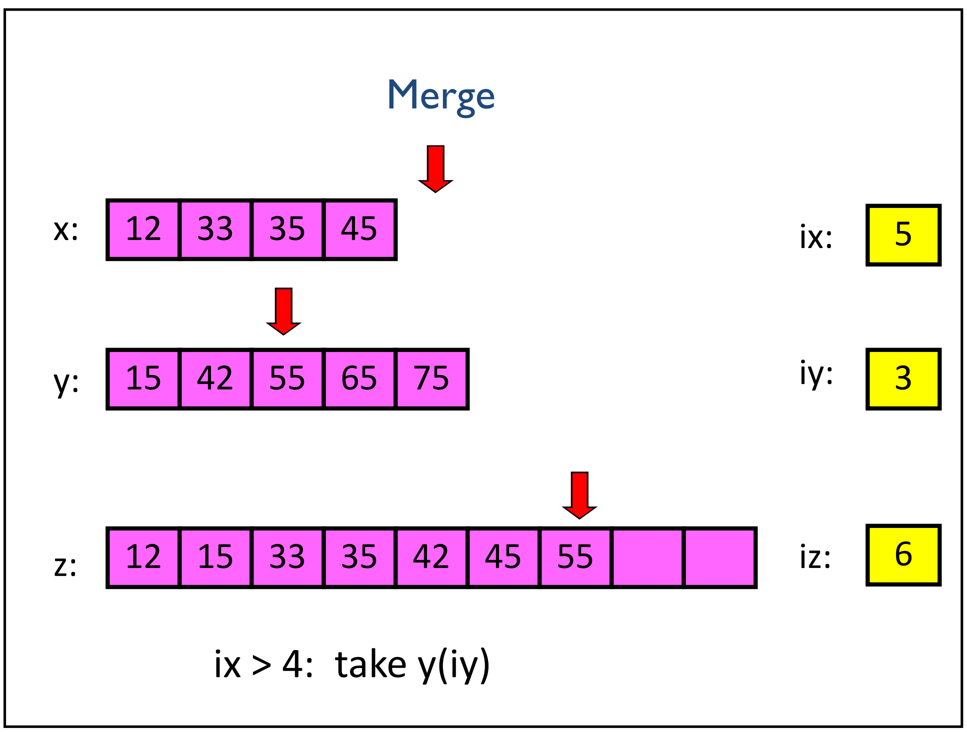
ix: 4

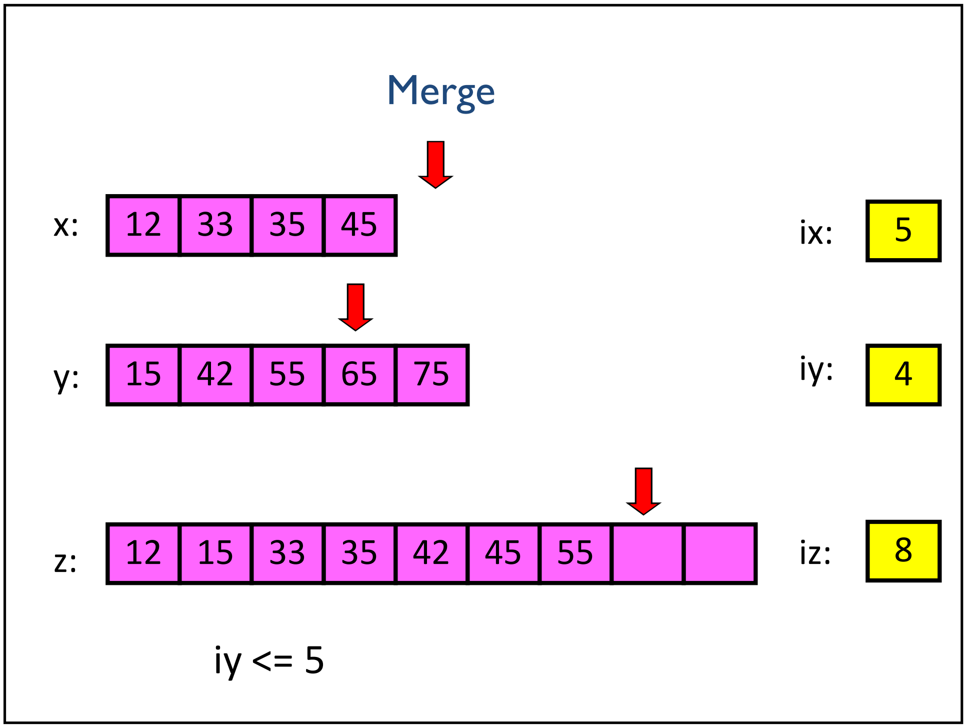
iy: 3

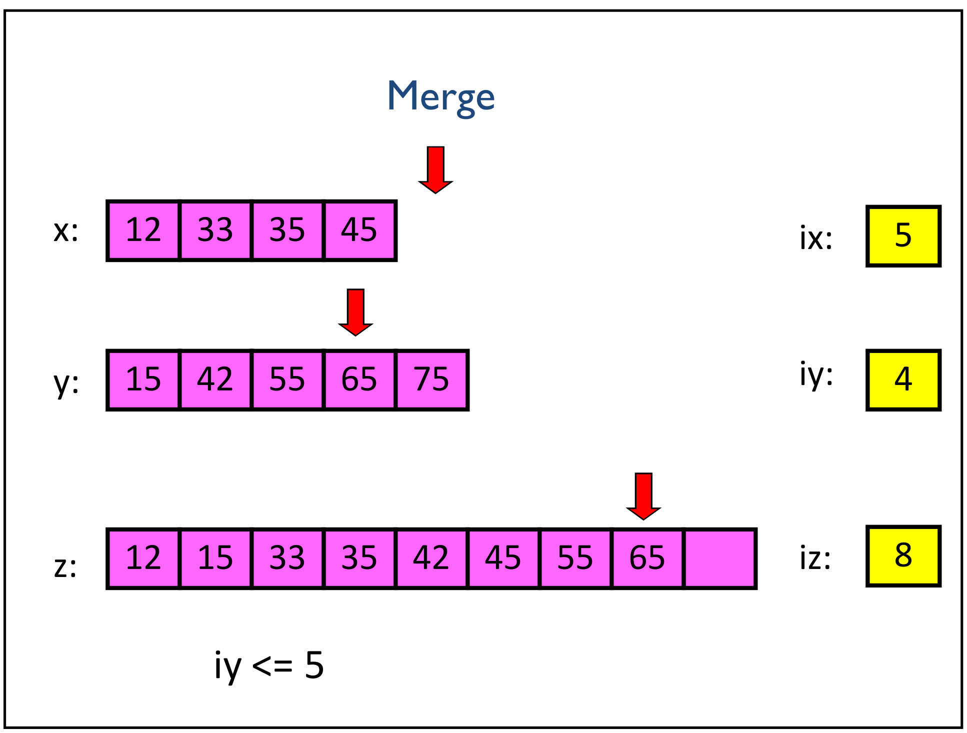
iz: 5

ix<=4 and iy<=5: x(ix) <= y(iy) YES

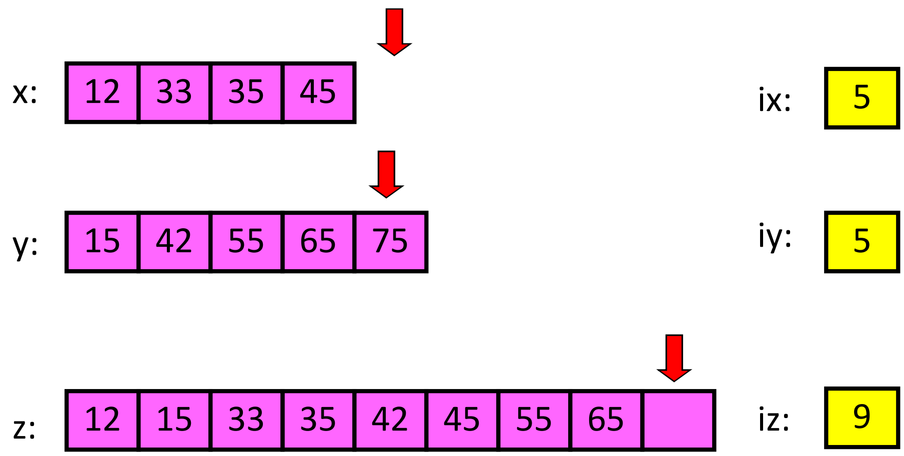

Merge
x: 12 33 35 45
ix: 5
y: 15 42 55 65 75
iy: 3
z: 12 15 33 35 42 45
iz: 6
ix > 4


Merge
x: 12 33 35 45
ix: 5
y: 15 42 55 65 75
iy: 3
z: 12 15 33 35 42 45 55
iz: 6
ix > 4: take y(iy)


Merge
x: 12 33 35 45
ix: 5
y: 15 42 55 65 75
iy: 4
z: 12 15 33 35 42 45 55
iz: 8
iy <= 5

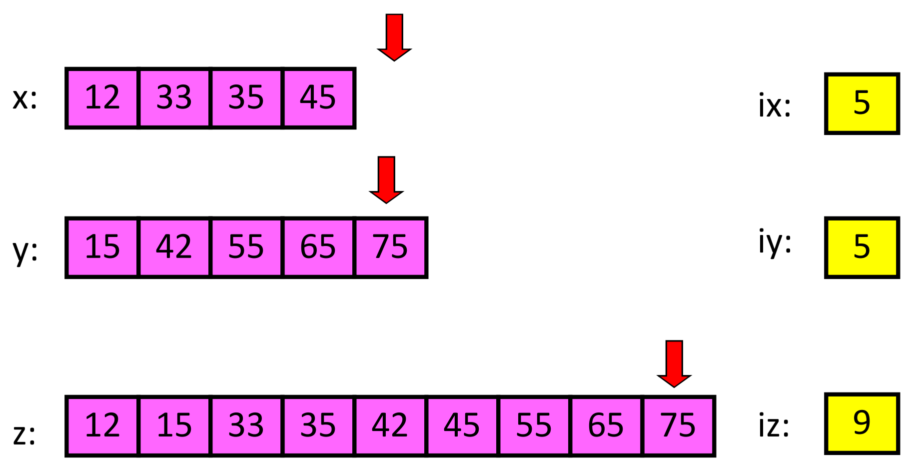

Merge
x: 12 33 35 45
ix: 5
y: 15 42 55 65 75
iy: 4
z: 12 15 33 35 42 45 55 65
iz: 8
iy <= 5

Merge



iy <= 5

Merge



iy <= 5

```
function z = merge(x,y)
nx = length(x); ny = length(y);
z = zeros(1,nx+ny);
ix = 1; iy = 1; iz = 1;
```

```
function z = merge(x,y)
nx = length(x); ny = length(y);
z = zeros(1, nx+ny);
ix = 1; iy = 1; iz = 1;
while ix<=nx && iy<=ny




end
% Deal with remaining values in x or y
```

```
function z = merge(x,y)
nx = length(x); ny = length(y);
z = zeros(1, nx+ny);
ix = 1; iy = 1; iz = 1;
while ix<=nx && iy<=ny
    if  x(ix) <= y(iy)
        z(iz)= x(ix);  ix=ix+1;  iz=iz+1;
    else
        z(iz)= y(iy);  iy=iy+1;  iz=iz+1;
    end
end
% Deal with remaining values in x or y
```

```
function z = merge(x,y)
nx = length(x); ny = length(y);
z = zeros(1, nx+ny);
ix = 1; iy = 1; iz = 1;
while ix<=nx && iy<=ny
    if  x(ix) <= y(iy)
        z(iz)= x(ix);  ix=ix+1;  iz=iz+1;
    else
        z(iz)= y(iy);  iy=iy+1;  iz=iz+1;
    end
end
while ix<=nx  % copy remaining x-values
  z(iz)= x(ix);  ix=ix+1;  iz=iz+1;
end
while iy<=ny  % copy remaining y-values
  z(iz)= y(iy);  iy=iy+1;  iz=iz+1;
end
```

```
function y = mergeSort(x)
% x is a vector.  y is a vector
% consisting of the values in x
% sorted from smallest to largest.

n = length(x);
if n==1
      y = x;
else
      m  = floor(n/2);
      yL = mergeSortL(x(1:m));
      yR = mergeSortR(x(m+1:n));
      y  = merge(yL,yR);
end
```

Insight Through Computing

```
function y = mergeSortL(x)
% x is a vector.  y is a vector
% consisting of the values in x
% sorted from smallest to largest.

n = length(x);
if n==1
      y = x;
else
      m  = floor(n/2);
      yL = mergeSortL_L(x(1:m));
      yR = mergeSortL_R(x(m+1:n));
      y  = merge(yL,yR);
end
```

Insight Through Computing

```
function y = mergeSortL_L(x)
% x is a vector.  y is a vector
% consisting of the values in x
% sorted from smallest to largest.

n = length(x);
if n==1
      y = x;
else
      m  = floor(n/2);
      yL = mergeSortL_L_L(x(1:m));
      yR = mergeSortL_L_R(x(m+1:n));
      y  = merge(yL,yR);
end
```

There should be just one mergeSort function!

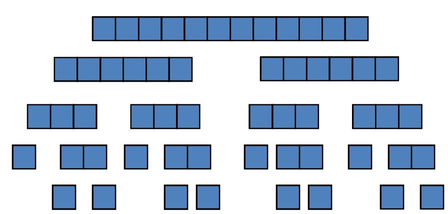

```
function y = mergeSort(x)
% x is a vector.  y is a vector
% consisting of the values in x
% sorted from smallest to largest.

n = length(x);
if n==1
      y = x;
else
      m  = floor(n/2);
      yL = mergeSort(x(1:m));
      yR = mergeSort(x(m+1:n));
      y  = merge(yL,yR);
end
```

```
function y=mergeSort(x)
n=length(x);
if n==1
      y=x;
else
      m=floor(n/2);
      yL=mergeSort(x(1:m));
      yR=mergeSort(x(m+1:n));
      y=merge(yL,yR);
end
```



```
function y=mergeSort(x)
n=length(x);
if n==1
      y=x;
else
      m=floor(n/2);
      yL=mergeSort(x(1:m));
      yR=mergeSort(x(m+1:n));
      y=merge(yL,yR);
end
```



Running Time

- Masalah berukuran n diselesaikan secara rekursif dengan cara menyelesaikan submasalah berukuran n/b.
- Untuk menggabungkan jawaban dari submasalah2 tersebut perlu waktu sebesar d(n)
- Untuk menyelesaikan pekerjaan sebesar 1 satuan perlu waktu sebesar 1 satuan
- $T(n) = a\,T(n/b) + d(n)$ untuk $n > 1$
  $= 1$ untuk $n = 1$

55

Teorema :

Solusi dari persamaan :

$T(n) = a\,T(n/b) + \theta(n^k)$ untuk $n > 1$,
dengan $a \geq 1$ dan $b > 1$

adalah :

$T(n) = O(n^{\log_b a})$, jika $a > b^k$
$O(n^k \log n)$, jika $a = b^k$
$O(n^k)$, jika $a < b^k$

56

- *Binary Search* : a = 1, b = 2, k = 0,
  $a = b^k \to O(n^k \log n)$
- **$T(n) = T(n/2) + c \rightarrow T(n) = O(\lg n)$**

- *Merge Sort* : a = 2, b = 2, k = 1,
  $a = b^k \to O(n^k \log n)$
- **$T(n) = 2T(n/2) + cn \rightarrow T(n) = O(n \lg n)$**

- *Max-min* : a = 2, b = 2, k = 0,
  $a > b^k \to O(n^{\log_b a})$
- **$T(n) = 2T(n/2) + c \rightarrow T(n) = O(n^{\log_2 2})$**
  $\rightarrow T(n) = O(n)$

57

#1 Problem: Counting inversions

Input: Array A Containing the numbers 1, 2, 3,...., n in some arbitrary order

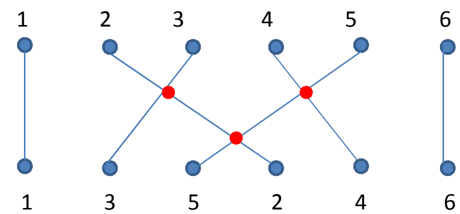
Output: Number of inversions = number of pairs (i,j) of array indices with i < j and A[i]> A[j]

58

Example and Motivation

<u>Example :</u> Array A =[1 3 5 2 4 6]

[1 3 5 2 4 6] → Inversions: (3,2), (5,2), (5,4)



<u>Motivation</u>

Numerical similiarity Measure between Two rank list (quantify how close 2 different rank list with each other)

What is the largest- possible number of inversions that a–6 element array can have? $O(n)=n^2$

59

D&C in counting inversions

- Possible adapt merge sort in counting inversions
- Sort and count
- Example: Misalkan L

1 3 5 2 4 6

→Membagi menjadi 2 sublist/subarray L menjadi n/2, misalkan A adalah sublist kiri dengan n/2 integer, B adalah sublist kanan dengan n/2 integer

→ Bagaimana merge sort? Pada saat merge pada operasi banding sisipkan count untuk inversions

Masalah Perkalian Bilangan Bulat

Andaikan a dan b adalah 2 buah bilangan bulat masing-masing N-digit:

Operasi penjumlahan: a+b => O(N)

1	1	1	1	1	1	0	1	
	1	1	0	1	0	1	0	1
+	0	1	1	1	1	1	0	1
1	0	1	0	1	0	0	1	0

Operasi perkalian : $O(N^2)$

```
                   1 1 0 1 0 1 0 1
                 * 0 1 1 1 1 1 0 1
                 -----------------
                 1 1 0 1 0 1 0 1 0
               0 0 0 0 0 0 0 0 0
             1 1 0 1 0 1 0 1 0
           1 1 0 1 0 1 0 1 0
         1 1 0 1 0 1 0 1 0
       1 1 0 1 0 1 0 1 0
     1 1 0 1 0 1 0 1 0
   0 0 0 0 0 0 0 0 0
 ---------------------------------
 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 1 0
```


Menggunakan teknik D&C

- Partisi setiap bilangan menjadi 2 @ N/2 digit
- Kalikan 4 bilangan berukuran N/2 digit tersebut
- Tambahkan 2 bilangan berukuran N/2 digit, kemudian di*shift* utk mendapat hasil akhir

$$
\begin{aligned}
123{,}456 \times 987{,}654 &= (10^3 w + x) \times (10^3 y + z) \\
&= 10^6 (wy) + 10^3 (wz + xy) + 10^0 (xz) \\
&= 10^6 (121{,}401) + 10^3 (80{,}442 + 450{,}072) + 10^0 (298{,}224) \\
&= 121{,}401{,}299{,}224
\end{aligned}
$$

$$
\begin{aligned}
w &= 123 \\
x &= 456 \\
y &= 987 \\
z &= 654
\end{aligned}
$$

Berapa kompleksitasnya?

$$ab = (10^{N/2} w + x)(10^{N/2} y + z)$$

$$T(N) = \underbrace{4T(N/2)}_{\text{recursive calls}} + \underbrace{\Theta(N)}_{\text{add, shift}} \Rightarrow T(N) = \Theta(N^2)$$

Pendekatan Karatsuba

$$\begin{aligned} 123{,}456 \times 987{,}654 &= (10^3 w + x) \times (10^3 y + z) \\ &= 10^6 (wy) + 10^3 (wz + xy) + 10^0 (xz) \\ &= 10^6 (p) + 10^3 (r - p - q) + 10^0 (q) \\ &= 10^6 (121{,}401) + 10^3 (950{,}139 - 121{,}401 - 298{,}224) + 10^0 (298{,}224) \\ & \quad 121{,}401{,}299{,}224 \end{aligned}$$

$$\begin{aligned} w &= 123 \\ x &= 456 \\ y &= 987 \\ z &= 654 \end{aligned}$$

$$\begin{aligned} p &= wy \\ q &= xz \\ r &= (w + x)(y + z) \end{aligned}$$

$$(wz + xy) = r - p - q$$

Pendekatan Karatsuba

Untuk mengalikan 2 buah bilangan berukuran N digit:

- Tambahkan 2 bilangan N/2 digit
- Kalikan 3 bilangan N/2 digit
- Kurangkan 2 bilangan N/2 digit, kemudian lakukan *shift* untuk mendapatkan hasil akhir

Berapa kompleksitasnya?

$$ab = (10^{N/2}w + x)(10^{N/2}y + z)$$

$$\mathrm{T}(N) \le \underbrace{T(\lfloor N/2 \rfloor) + T(\lceil N/2 \rceil) + T(1+\lceil N/2 \rceil)}_{\text{recursive calls}} + \underbrace{\Theta(N)}_{\text{add, subtract, shift}}$$

$$\Rightarrow \mathrm{T}(N) = O(N^{\log_2 3})$$

Karatsuba-Ofman (1096) : *T(n) = O($N^{1.585}$)*

Masalah Perkalian matriks

Hitung C = A.B, dengan A dan B masing-masing berukuran NxN, sebagai contoh:

$$\begin{pmatrix} 26 & 62 & 98 \\ \boxed{80} & 224 & 368 \\ 134 & 386 & 638 \end{pmatrix} = \begin{pmatrix} 0 & 2 & 4 \\ \boxed{6 \quad 8 \quad 10} \\ 12 & 14 & 16 \end{pmatrix} \times \begin{pmatrix} \boxed{1} & 7 & 13 \\ \boxed{3} & 9 & 15 \\ \boxed{5} & 11 & 17 \end{pmatrix}$$

$$\begin{pmatrix} c_{11} & c_{12} & c_{13} & \cdots & c_{1N} \\ c_{21} & c_{22} & c_{23} & \cdots & c_{2N} \\ c_{31} & c_{32} & c_{33} & \cdots & c_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{N1} & c_{N2} & c_{N3} & \cdots & c_{NN} \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1N} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2N} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & a_{N3} & \cdots & a_{NN} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & b_{13} & \cdots & b_{1N} \\ b_{21} & b_{22} & b_{23} & \cdots & b_{2N} \\ b_{31} & b_{32} & b_{33} & \cdots & b_{3N} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{N1} & b_{N2} & b_{N3} & \cdots & b_{NN} \end{pmatrix}$$

Kompleksitas algoritma adalah : $\theta(N^3)$

67

Teknik D&C

Divide: Partisi A dan B menjadi 4 buah N/2 x N/2

Conquer: Kalikan 8 matriks tersebut secara rekursif

Combine: Gabungkan hasilnya dengan 4 operasi penjumlahan

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\begin{aligned} C_{11} &= (A_{11} \times B_{11}) + (A_{12} \times B_{21}) \\ C_{12} &= (A_{11} \times B_{12}) + (A_{12} \times B_{22}) \\ C_{21} &= (A_{21} \times B_{11}) + (A_{22} \times B_{21}) \\ C_{22} &= (A_{21} \times B_{12}) + (A_{22} \times B_{22}) \end{aligned}$$

$$\mathrm{T}(N) = \underbrace{8T(N/2)}_{\text{recursive calls}} + \underbrace{\Theta(N^2)}_{\text{add, form submatrices}} \Rightarrow \mathrm{T}(N) = \Theta(N^3)$$

68

Strassens's Matrix Multiplication

$$\begin{vmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{vmatrix} = \begin{vmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{vmatrix} \begin{vmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{vmatrix}$$

$P_1 = (A_{11} + A_{22}) * (B_{11} + B_{22})$
$P_2 = (A_{21} + A_{22}) * B_{11}$
$P_3 = A_{11} * (B_{12} - B_{22})$
$P_4 = A_{22} * (B_{21} - B_{11})$
$P_5 = (A_{11} + A_{12}) * B_{22}$
$P_6 = (A_{21} - A_{11}) * (B_{11} + B_{12})$
$P_7 = (A_{12} - A_{22}) * (B_{21} + B_{22})$

$C_{11} = P_1 + P_4 - P_5 + P_7$
$C_{12} = P_3 + P_5$
$C_{21} = P_2 + P_4$
$C_{22} = P_1 + P_3 - P_2 + P_6$

Total : 7 perkalian, 18 penjumlahan/ pengurangan

Teknik D&C

Divide: Partisi A dan B menjadi 4 buah N/2 x N/2

Conquer: Kalikan 8 matriks tersebut secara rekursif

Combine: Gabungkan hasilnya dengan 4 operasi penjumlahan

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\begin{aligned} C_{11} &= (A_{11} \times B_{11}) + (A_{12} \times B_{21}) \\ C_{12} &= (A_{11} \times B_{12}) + (A_{12} \times B_{22}) \\ C_{21} &= (A_{21} \times B_{11}) + (A_{22} \times B_{21}) \\ C_{22} &= (A_{21} \times B_{12}) + (A_{22} \times B_{22}) \end{aligned}$$

$$T(N) = \underbrace{8T(N/2)}_{\text{recursive calls}} + \underbrace{\Theta(N^2)}_{\text{add, form submatrices}} \Rightarrow T(N) = \Theta(N^3)$$

$$\begin{aligned} C_{11} &= P_1 + P_4 - P_5 + P_7 \\ &= (A_{11} + A_{22})(B_{11} + B_{22}) + A_{22} * (B_{21} - B_{11}) - (A_{11} + A_{12}) * B_{22} + \\ &\quad (A_{12} - A_{22}) * (B_{21} + B_{22}) \\ &= A_{11} B_{11} + A_{11} B_{22} + A_{22} B_{11} + A_{22} B_{22} + A_{22} B_{21} - A_{22} B_{11} - \\ &\quad A_{11} B_{22} - A_{12} B_{22} + A_{12} B_{21} + A_{12} B_{22} - A_{22} B_{21} - A_{22} B_{22} \\ &= A_{11} B_{11} + A_{12} B_{21} \end{aligned}$$

70

Metode Strassen

Divide: Partisi A dan B menjadi N/2 x N/2
Hitung 14 matriks berukuran N/2 x N/2 dengan 10 operasi penjumlahan/ pengurangan matriks

Conquer: Secara rekursif, hitung Pi = Ai.Bi, untuk i = 1,..7

Combine: Gabungkan 7 perkalian menjadi elemen C menggunakan 8 operasi penjumlahan/ pengurangan

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$$\mathrm{T}(N) = \underbrace{7T(N/2)}_{\text{recursive calls}} + \underbrace{\Theta(N^2)}_{\text{add, subtract}} \Rightarrow \mathrm{T}(N) = \Theta(N^{\log_2 7}) = O(N^{2.81})$$

71

Latihan

- Gunakan algoritme Strassen untuk menghitung perkalian matriks (cormen *et.al* 2009 – 4.2-1):

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}$$

Tugas:

1. Tuliskan algoritme strassen dalam bentuk pseudocode (Cormen *et.al* 2009 – 4.2-2)
2. Modifikasi algoritme pada merge sort, menjadi merge-inversions counting
3. Tugas Baca: algoritme karatsuba (tahapan, algoritme dalam konsep divide & conquer)