



## MODULE6—SORT

### 6.0 Sort คือ การเรียงข้อมูล

- Sorting refers to arranging data in a particular format.
- Sorting algorithm specifies the way to arrange data in a particular order. <sup>ผลลัพธ์</sup>
- Most common orders are in numerical or lexicographical order.

0	21	19	11	15	13	9	6	2	4	8	
8	1	8	19	11	15	13	9	6	2	4	21
9	2	19	15	11	8	13	9	6	2	4	21
8+9	3	15	13	11	8	4	9	6	2	19	21



- An array containing the elements. Assume that  $N$ ,
- Data will start at position 0.

0	1	2	...	n-2	n-1
34	8	64	...	32	21



## Type

1. Selection Sort \* ๓๗ min
2. Bubble Sort
3. Insertion Sort
4. Shell Sort
5. Merge Sort
6. Quick Sort
7. Radix Sort
8. Heap Sort



## การสร้าง file

สุ่มตัวเลข 5 ตัว เก็บลง file ชื่อ outfile.txt

```
#include <iostream>
```

```
#include <fstream>
```

```
#include <cstdlib>
```

```
using namespace std;
```

```
int main ()
```

```
{  
    output file  
    ofstream myfile;
```

```
    myfile.open ("outfile.txt");
```

```
    for (int i = 0; i < 5; i++)
```

```
        myfile << rand()%1000 << endl;
```

```
    cout << "Create file";
```

```
    myfile.close();
```

```
}
```



### การอ่านข้อมูลจาก file

```
#include <iostream>
```

```
#include <fstream>
```

```
using namespace std;
```

```
int main()
```

```
{    int data[6];
```

```
    ifstream myfile; input file
```

```
    myfile.open("outfile.txt");
```

```
    for(int i = 0; i < 5 ; i++)
```

```
    {    myfile >> data[i];
```

```
        cout << data[i] << " ";
```

```
    }
```

```
    myfile.close();
```

```
}
```



# 1. Selection Sort \* ง่าย min

Big O (n)

input

0	1	2	3	4	5
34	8	64	51	32	21

p=1


34	8	64	51	32	21
8	34	64	51	32	21

p=2

8	34	64	51	32	21
8	21	64	51	32	34




**p=3**




8	21	64	51	32	34
8	21	32	51	64	34

**p=4**



8	21	32	51	64	34
8	21	32	34	64	51

**p=5**



8	21	32	34	64	51
8	21	32	34	51	64



### ขั้นตอน

ข้อมูลตำแหน่ง 0-(N-1)    หาค่า min และสลับ min มาตำแหน่ง 0    (N ตัว)  
ข้อมูลตำแหน่ง 1-(N-1)    หาค่า min และสลับ min มาตำแหน่ง 1    (N-1 ตัว)  
ข้อมูลตำแหน่ง 2-(N-1)    หาค่า min และสลับ min มาตำแหน่ง 2    (N-2 ตัว)

...

ข้อมูลตำแหน่ง (N-2)-(N-1)    หาค่า min และสลับ min มาตำแหน่ง N-2    (2 ตัว)

**รอบสุดท้ายที่เหลือตัวเดียว ไม่นิยมหา min**

### Analysis of Selection sort:

Time = (N) + (N-1) + .. +2

$$1 + 2 + \dots + n = \frac{n(n+1)}{2} = O(n^2)$$

$$= 1 \times (n^2 + n)$$

Input	34	8	64	51	32	21
p=1						
p=2						
p=3						
p=4						
p=5						
p=6						





## 2. Bubble Sort

	0	1	2	3	4	5
input	34	8	64	51	32	21

P=1

34	8	64	51	32	21
34	8	64	51	21	32
34	8	64	21	51	32
34		21	64	51	32



**P=2**

<b>8</b>	<b>34</b>	<b>21</b>	<b>64</b>	<b>51</b>	<b>32</b>



input

0	1	2	3	4	5
34	8	64	51	32	21

P=1

34	8	64	51	32	21
8	34	21	64	51	32

P=2

8	34	21	64	51	32
8	21	34	32	64	51



**P=3**

8	21	34	32	64	51
8	21	32	34	51	64

**P=4**

8	21	32	34	51	64
8	21	32	34	51	64

**P=5**

8	21	32	34	51	64
8	21	32	34	51	64



## ขั้นตอน

จะทำการเปรียบเทียบข้อมูลที่ละคู่ จากด้านหน้าไปด้านหลังหรือด้านหลังไปด้านหน้าก็ได้ ในที่นี้จะทำจากหลังไปหน้า

เฟส1 index N-1 ถึง 0

เฟส2 index N-1 ถึง 1

เฟส3 index N-1 ถึง 2

...

ทำเช่นนี้จนถึง N-1 ถึง N-2

$\text{Big } O(N^2)$

### Analysis of Selection sort:

$$\text{Time} = (N) + (N-1) + \dots + 2$$

$$\text{Time} = 5 + 4 + 3 + 2$$

Input	34	8	64	51	32	21
p=1						
p=2						
p=3						
p=4						
p=5						
p=6						



### 3. Insertion Sort

	0	1	2	3	4	5
input	34	8	64	51	32	21

P=1	34	8	64	51	32	21
	8	34	64	51	32	21

ไปเรียงตามหลักเลข 2 ตัว


P=2	8	34	64	51	32	21
ผล	8	34	64	51	32	21

ไปเรียงตามหลักเลข 3 ตัว



**P=3**

ผล




8	34	64	51	32	21
8	34	51	64	32	21

เปรียบเทียบแล้ว 4 ครั้ง

**P=4**

ผล




8	34	51	64	32	21
8	32	34	51	64	21

เปรียบเทียบแล้ว 5 ครั้ง

**P=5**

ผล



8	32	34	51	64	21
8	21	32	34	51	64

เปรียบเทียบแล้ว 6 ครั้ง



Input	34	8	64	51	32	21
p=1	34	8	64	51	32	21
p=2	8	34	64	51	32	21
p=3	8	34	64	51	32	21
p=4	8	34	51	64	32	21
p=5	8	32	34	51	64	21
ผล	8	21	32	34	51	64

Analysis of Insertion sort:

กรณีแย่สุด  $O(n^2)$  \* เปรียบเทียบทุกตัว ขึ้นอยู่กับข้อมูล

กรณีดีที่สุด ?  $O(n)$





**An inversion** in an array of numbers is any ordered pair  $(i,j)$  having the property that  $i < j$  but  $A[i] > A[j]$ .

สลับ ทั่ว ทั่ว ทั่ว ทั่ว ทั่ว คือ index น้อย แต่ค่ามากกว่า หรือ index ใหญ่ แต่ค่าน้อย

list 34,8,64,51,32,21 had nine inversions,

(34,8), (34,32),(34,21)

(64,51),(64,32),(64,21)

(51,32),(51,21),(32,21)

\* Avg จำนวน inversion + จำนวน ข้อมูล

running time of insertion sort  $O(I+N)$ , where  $I$  is the number of inversions in the original array.



void Insertionsort( int A[], int N) N คือจำนวนข้อมูล

{ เริ่มจากตัวท้าย ไปยัง 1

int j,p;

P=1

int Tmp;

เลขด้านหน้ามากกว่า

for ( p=1; p<N; p++ ) // phase

{ Tmp=A[p];

for( j=p; j>0 && A[j-1] >Tmp; j--)

A[j]=A[j-1]; สลับกับตัวหน้า

A[j] = Tmp;

}

}

34	8	64	51	32	21
8	34	64	51	32	21

p=1

Tmp=8

34 > Tmp

← j

เฟส 1 j = 1 → 0

เฟส 2 j = 2 → 0

เฟส 3 j = 3 → 0



## 4 ShellSort

- เป็นวิธีการที่ช่วยปรับปรุง Insertion Sort ให้มีจำนวนครั้งการสับเปลี่ยนข้อมูลลดลง
- Insertion Sort อาจจะทำให้เกิด Worst-Case ในกรณีที่ค่าตัวเลขน้อย ไปอยู่ที่ข้างท้ายๆ
- วิธีการ Shell Sort ปรับปรุงโดยทำการจัดเรียงข้อมูลชุดย่อยๆ
  - โดยแต่ละชุดย่อย  $k$  จะเป็นเลขลำดับที่เพิ่มมากขึ้นเรื่อยๆ เช่น  $k$  อาจจะเป็นเลข  $\{1, 3, 5\}$  หมายถึง ครั้งแรกที่ทำ 5-sorted จะจัดเรียงตัวเลขทุกตัวที่ห่างกันทีละ 5 ช่อง เมื่อเรียงเสร็จแล้วจะทำ 3-sorted โดยจัดเรียงตัวเลขทุกตัวที่ห่างกันทีละ 3 ช่อง และสุดท้ายจะทำ 1-sorted โดยจัดเรียงตัวเลขทุกตัวที่ติดกันอยู่ ซึ่งในรอบ สุดท้ายที่ 1-sorted จะเป็นวิธีการเดียวกับ Insertion Sort นั่นเอง
- สังเกตว่าวิธีการของ Shell Sort จะทำให้เลขเมื่อเรียงแต่ละรอบแล้ว ข้อมูลจะใกล้สู่การเรียงลำดับขึ้นสุดท้ายมากยิ่งขึ้นเรื่อยๆ ทุกครั้งที่ผ่าน  $k$ -sorted
- 1-sorted ข้อมูลเกือบจะเรียงกันหมดแล้ว ซึ่งจะมีผลทำให้ Insertion Sort ใช้เวลาน้อยมากกว่า Insertion Sort ปกติ



## **ShellSort**

One of the first algorithms to break the quadratic time. Work by comparing elements that are distant. The distance between comparisons decrease as the algorithm runs until the last phase, in which adjacent elements are compared. Shell sort uses a sequence  $h_1, h_2, \dots, h_t$  called the increment sequence.



# 03603212 : Module6 – Sort $h = \{1, 3, 5\}$ 21

	1	2	3	4	5	1	2	3	4	5			
	81	94	11	96	12	35	17	95	28	58	41	75	15
5 s	35	17	11	28	12	41	75	15	96	58	81	94	95
3 s	28	12	11	35	15	41	58	17	94	75	81	96	95
1 s	11	12	15	17	28	35	41	58	75	81	94	95	96
	12	28	11										
	11	12	28										
	11	12	28	35									
	11	12	15	28	35	41	58						
	11	12	15	17	28	35	41	58	94				
	11	12	15	17	28	35	41	58	75	94			
	11	12	15	17	28	35	41	58	75	81	94	96	
	11	12	15	17	28	35	41	58	75	81	94	95	96

6 ครั้ง P1

5 ครั้ง

11 ครั้ง



	81	94	11	96	12	35	17	95	28	58	41	75	15
5 s	35	17	11	28	12	41	75	15	96	58	81	94	95
3 s	28	12	11	35	15	41	58	17	74	75	81	96	95
1 s	11	12	15	17	28	35	41	58	75	81	74	95	96



- Hibbard's increment sequence  $1, 3, 7, \dots, 2^k - 1$

- $O(N^{5/4})$

$N = 20$

- $N = 14, \quad h = \{1, 3, 7\}, \quad k = 3$

$$h = \{1, 3, 7, 15\}$$

[illegible]



## Increment Sequence

- Sedgewick {1, 8, 23, 77, 281, ... }

$$4^k + 3 \cdot 2^{k+1} + 1$$

Worst-Case ที่  $O(N^{4/3})$  และคาดว่าโดยเฉลี่ยอาจจะได้ถึง  $O(N^{7/6})$

$$k = 1 \quad 4^1 + 3 \cdot 2^0 + 1 = 4 + 3 + 1 = 8$$

$$k = 2 \quad 4^2 + 3 \cdot 2^1 + 1 = 16 + 6 + 1 = 23$$

$$N^{5/4} > N^{4/3}$$

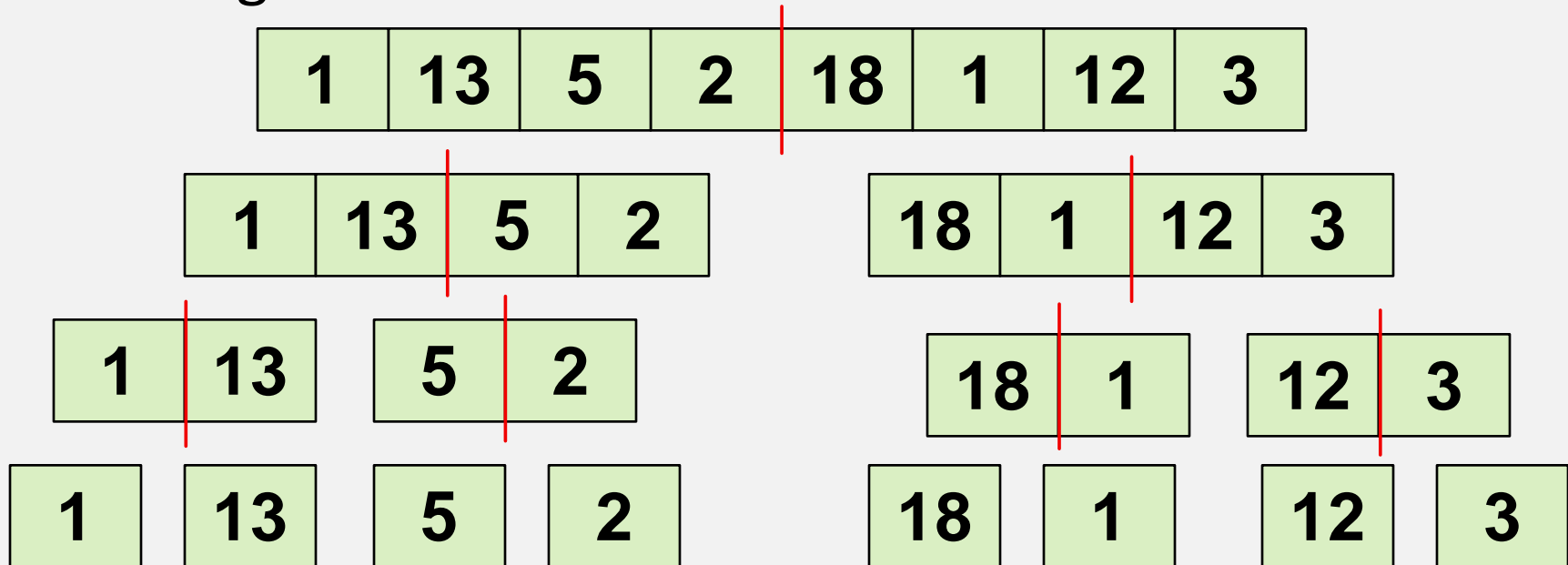
Hibb                      Sedgewick





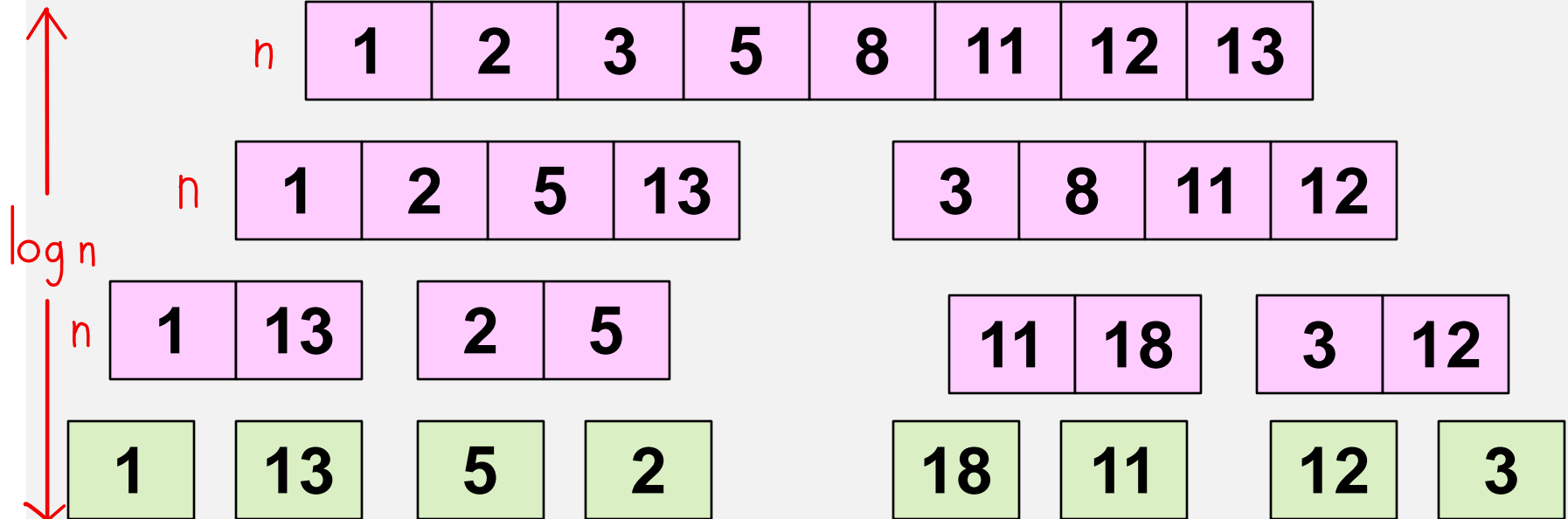
## 6.3 Merge Sort

- $O(N \log N)$  in 3 cases running time.
- Merge two sorted lists.





\* แบ่งแล้ว ตามด้วย Merge



**0****3**

```
1) void msort(int A[],int tmparray[],int left, int right)
2) { int center;
3)   if ( left < right)
4)   { center = (left+right)/2;
5)     msort(A,tmparray,left,center);
6)     msort(A,tmparray,center+1,right);
7)     merge(A,tmparray,left,center+1,right);
8)   }
9) }
```

ยังแบ่งได้

**1**

0	1	2	3
1	13	5	2

**0****1****A****2****3****B****0****2****3****C**



**A**

**0**

**1**

1) void msort(int A[],int tmparray[],int left, int right)

2) { int center;

3) if ( left < right)

4) { center = (left+right)/2;

**0**

5) msort(A,tmparray,left,center);

6) msort(A,tmparray,center+1,right);

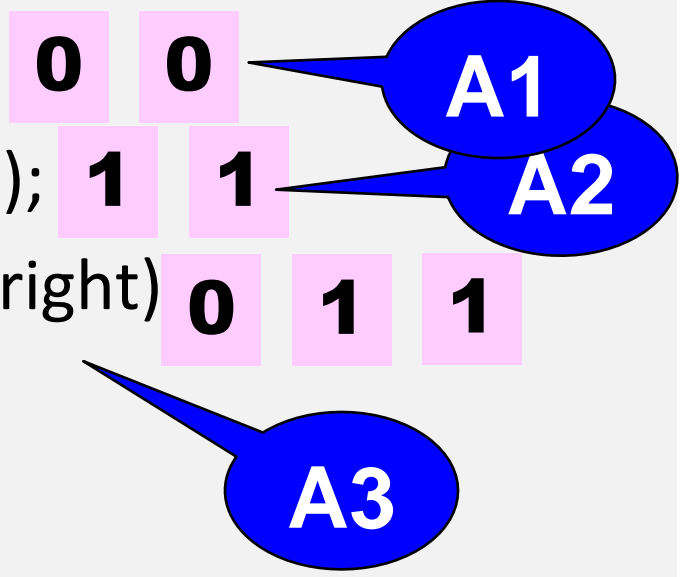
7) merge(A,tmparray,left,center+1,right)

8) }

9) }

0 1 2 3

1	13	5	2
---	----	---	---





**A1**

**0**

**0**

1) void msort(int A[],int tmparray[],int left, int right)

2) { int center;

3) if ( left < right) **F**

0	1	2	3
1	13	5	2

4) { center = (left+right)/2;

5) msort(A,tmparray,left,center);

6) msort(A,tmparray,center+1,right);

7) merge(A,tmparray,left,center+1,right);

8) }

9) }

10) }



**A2**

**1**

**1**

```
1) void msort(int A[],int tmparray[],int left, int right)
2) { int center;
3)   if ( left < right) F
4)   { center = (left+right)/2;
5)     msort(A,tmparray,left,center);
6)     msort(A,tmparray,center+1,right);
7)     merge(A,tmparray,left,center+1,right);
8)
9)   }
```

0	1	2	3
1	13	5	2

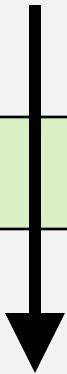


จำชื่อ

0	1	2	3	4	5	6	7
1	2	5	13	3	6	19	20



0	1	2	3	4	5	6	7
1	2	5	13	3	6	19	20



lpos = 0  
leftend = 3  
rpos = 4  
rightend = 7

lpos = 0  
leftend = 0  
rpos = 1  
rightend = 1



0360

A3

merge(A,tmparray,left,center+1,right);

0 1 1

0 1 1

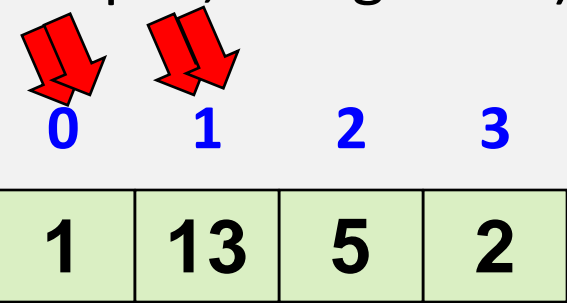
void merge(int A[], int tmparray[],int lpos,int rpos,int rightend)

{ int i, leftend, numelements,tmppos;

leftend = rpos -1;

tmppos = lpos;

numelements = rightend-lpos +1;



lpos = 0  
leftend = 0  
rpos = 1  
rightend = 1  
tmppos = 0  
numelement = 2

lpos	หน้าซ้าย
leftend	หลังซ้าย
rpos	หน้าขวา
rightend	หลังขวา

0

0

2





จัดไว้ไม่หมด

วางยังไม่หมด

while ( lpos <= leftend && rpos <= rightend )

if ( A[lpos] <= A[rpos] )

**1 <= 13**

tmparray[tmpppos++] = A[lpos++];

ตน. sort

ถ้าน้อย

else

tmparray[tmpppos++] = A[rpos++];

0 1 2 3

1	13	5	2
---	----	---	---

0 1 2 3

tmp

1			
---	--	--	--

lpos = 0

leftend = 0

rpos = 1

rightend = 1

tmppos = 0

numelement = 2



while(lpos <= leftend) <sup>ซ้าย ไปถึง เหลือ</sup>

tmparray[tmppos++] = A[lpos++];

while(rpos <= rightend) <sup>ขวา ไปถึง เหลือ T</sup>

tmparray[tmppos++] = A[rpos++];

for(i=0; i<numelement; i++, rightend--)

A[rightend] = tmparray[rightend];

<sup>\* ถ้า array tmp → array A</sup>

}  
0 1 2 3

A

1	13	5	2
---	----	---	---

tmp

1	13		
---	----	--	--

lpos = ~~0~~ 1

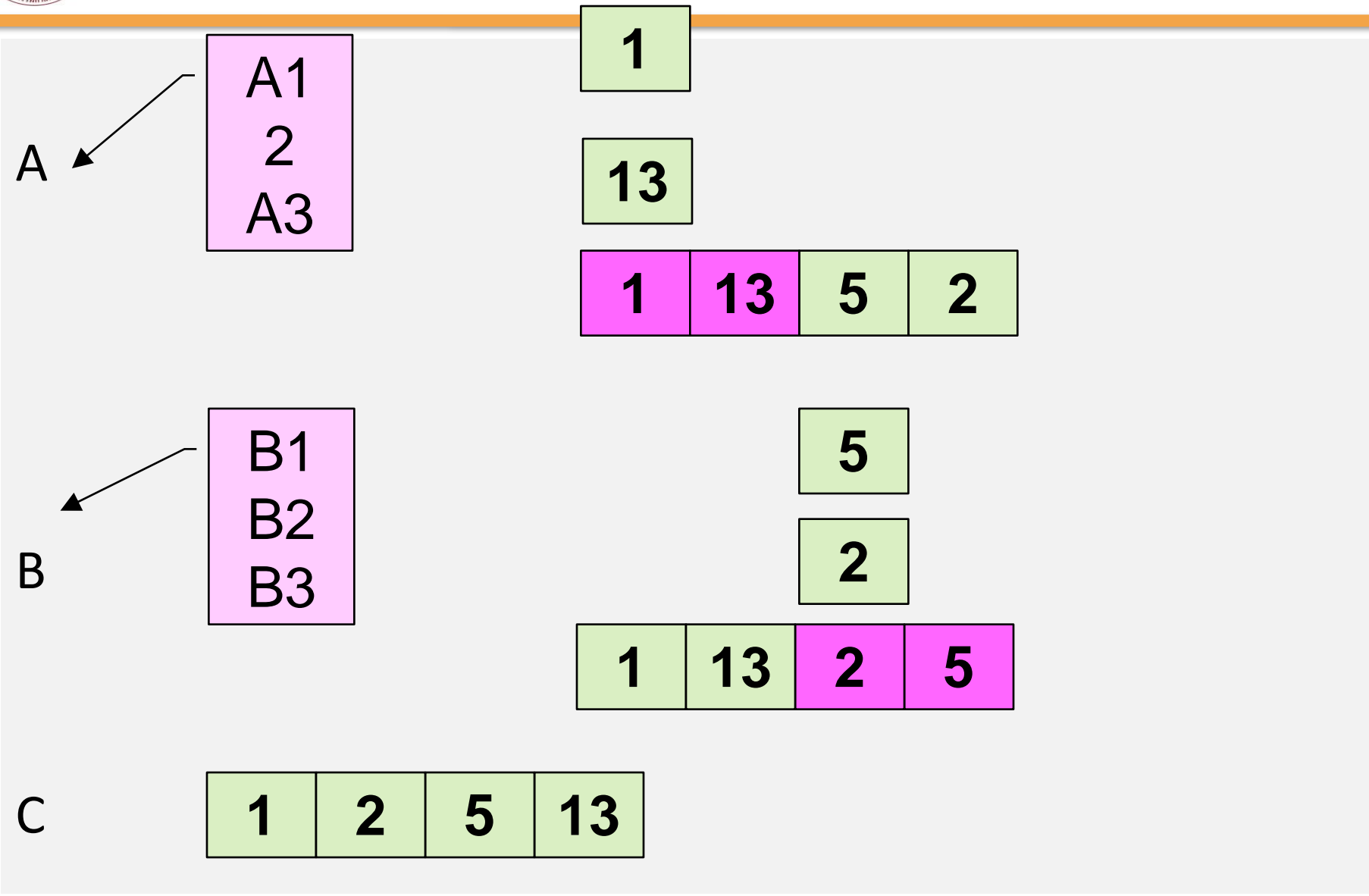
rpos = 1

rightend = 1

leftend = 0

tmppos = ~~0~~ 1

numelement = 2



**C****0****2****3**

```
void merge(int A[], int tmparray[], int lpos, int rpos, int rightend)
{
    int i, leftend, numelements, tmppos;
    leftend = rpos - 1;
    tmppos = lpos;
    numelements = rightend - lpos + 1;
```

**1****0***A*

1	13	2	5
---	----	---	---

*A* *B*

**4***lpos 0**lend 1**rpos 2**rend 3*



while ( lpos <= leftend && rpos <= rightend)  $\top$   $\text{f}$

if (A[lpos<sup>1</sup>] <= A[rpos<sup>2</sup>])  $\top$

tmparray[tmppos++] = A[lpos++];

else

tmparray[tmppos++] = A[rpos++];

lpos = ~~0~~ 1

leftend = 1

rpos = 2 ~~3~~ 4

rightend = 3

tmppos = ~~0~~ ~~1~~ ~~2~~ 3

numelement = 4

	0	1	2	3
A	1	13	2	5

tmp	1	2	5	
-----	---	---	---	--



```
while(lpos <= leftend)
```

```
    tmparray[tmpos++] = A[lpos++];
```

```
while(rpos <= rightend)
```

```
    tmparray[tmpos++] = A[rpos++];
```

```
for(i=0; i<numelement; i++, rightend--)
```

```
    A[rightend] = tmparray[rightend];
```

```
}
```

lpos = 0

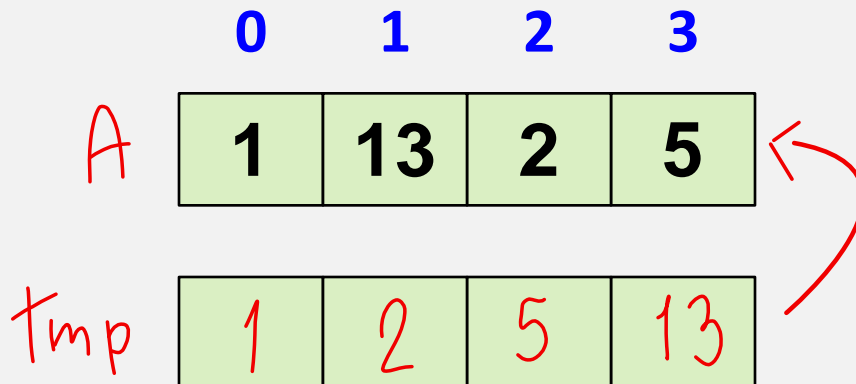
leftend = 1

rpos = 2

rightend = 3

tmppos = 0

numelement = 4



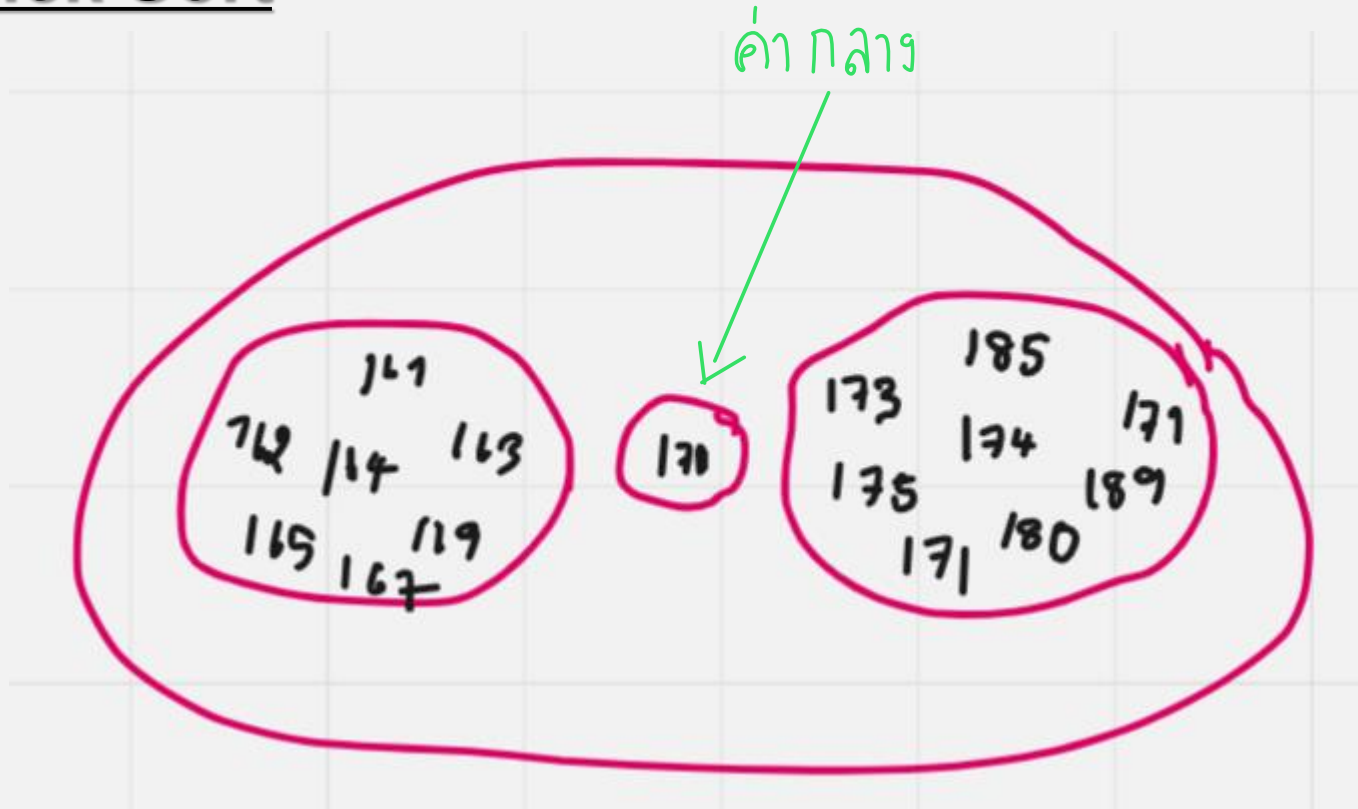


## 6.4 Quick Sort





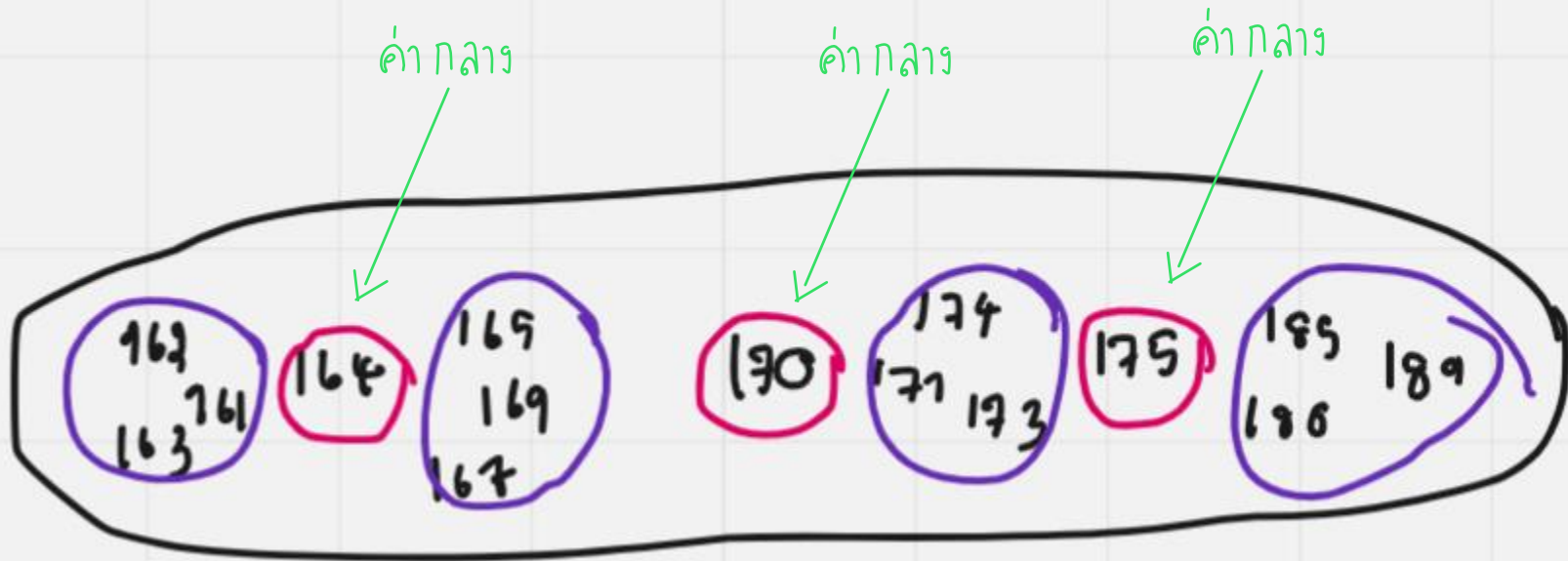
## 6.4 Quick Sort





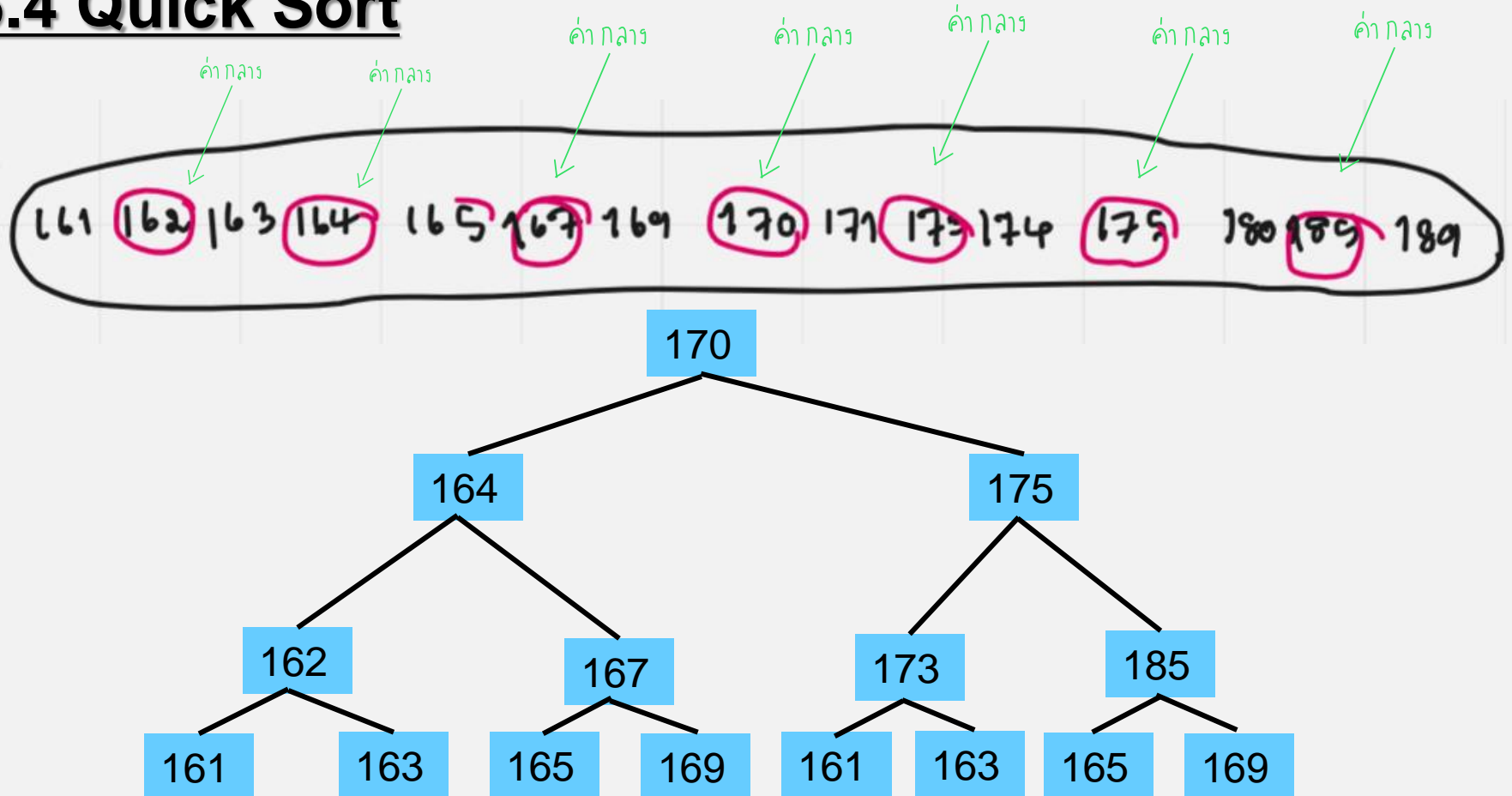


## 6.4 Quick Sort





## 6.4 Quick Sort





X,X,X,X,X,X,X,X,X,X,X,X,X,X,X

รอบ	เทียบ	จำนวนตำแหน่งที่ถูก	รวม
1	n	1	1

X,X,X,X,X,X,X,X,X,X,X,X,X,X,X

รอบ	เทียบ	จำนวนตำแหน่งที่ถูก	รวม
2	$n/2+n/2$	2	1+2



x, (x), x, x, (x), x, x, (x), x, x, (x), x

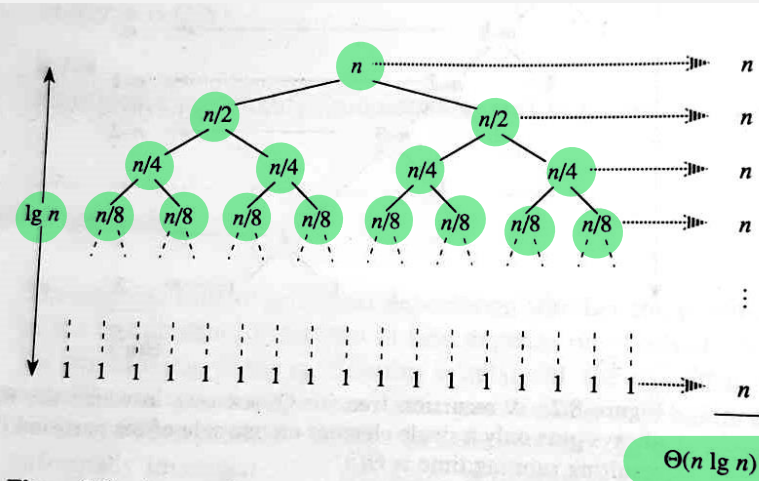
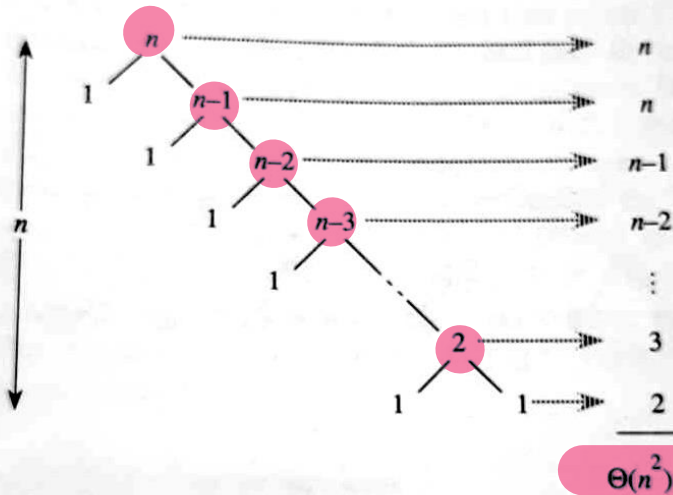
รอบ	เทียบ	จำนวนตำแหน่งที่ถูก	รวม
3	$n/4 + n/4 + n/4 + n/4$	4	$4 + 3$



Worst-case  $O(n^2)$

Avg  $O(n)$

Best-case  $O(n \log n)$





## Step

- 1) If the number of elements in  $S$  is 0 or 1, then return.
- 2) Pick any element  $v$  in  $S$ . This is called the pivot.
- 3) Partition  $S - \{v\}$  ( the remaining elements in  $S$ ) in to two disjoint groups:  
     $S_1$  and  $S_2$
- 4) Return { quicksort( $S_1$ ) followed by  $v$  followed by quicksort ( $S_2$ ) }.



### 6.4.1 Picking the pivot

- Wrong way : use the first element as the pivot.

9	8	7	6	5	4	3	2	1
---	---	---	---	---	---	---	---	---

Problem : Input is presorted or in reverse order.



- Median of three partitioning : the median of a group of N numbers is the  $N/2$  largest number.

Input 8,1,4,9,6,3,5,2,7,0

Input Center =  $(\text{Left} + \text{Right})/2 = 11/2 = 5$

$v = 6$  ( 1 + 10 ) / 2

1	2	3	4	5	6	7	8	9	10
8	1	4	9	6	3	5	2	7	0

1	13	5	2
---	----	---	---

Correct : sorted input.





### 6.4.2

1. Position = 5,  $v = 6$

8	1	4	9	6	3	5	2	7	0
---	---	---	---	---	---	---	---	---	---

2. Swap between position and Last array กำหนด  $i, j$

8	1	4	9	0	3	5	2	7	6
---	---	---	---	---	---	---	---	---	---

$i$  \* ถูก ที่ มากกว่า  $v$  , ดึงที่น้อยกว่า  $v$

$j$

3. จากตำแหน่ง  $i$  ( $i < j$ ) หาค่า  $a[i]$  ที่มีค่ามากกว่า  $v$  หยุด  
จากตำแหน่ง  $j$  หาค่า  $a[j]$  ที่มีค่าน้อยกว่า  $v$  หยุด

8	1	4	9	0	3	5	2	7	6
---	---	---	---	---	---	---	---	---	---

$i$

$j$



8	1	4	9	0	3	5	2	7	6
---	---	---	---	---	---	---	---	---	---

i

2	1	4	9	0	3	5	8	7	6
---	---	---	---	---	---	---	---	---	---

i



5. หาค่า  $i, j$  ต่อ

หาค่า  $a[i]$  ที่มีค่ามากกว่า  $v$  หาค่า  $a[j]$  ที่มีค่าน้อยกว่า  $v$

2	1	4	9	0	3	5	8	7	6
---	---	---	---	---	---	---	---	---	---

$i$

$j$

2	1	4	9	0	3	5	8	7	6
---	---	---	---	---	---	---	---	---	---

$i$

$j$

6. สลับ

2	1	4	5	0	3	9	8	7	6
---	---	---	---	---	---	---	---	---	---

$i$

$j$



7. ทำเหมือนเดิม หาค่า  $i, j$  loop สิ้นสุดเมื่อค่า  $i > j$

2	1	4	5	0	3	9	8	7	6
---	---	---	---	---	---	---	---	---	---

i

j

2	1	4	5	0	3	9	8	7	6
---	---	---	---	---	---	---	---	---	---

j

i



8. สลับระหว่างค่าpivot กับค่า l  
จะเห็นว่า 6 เป็นตำแหน่งที่ถูกต้อง เมื่อทำการเรียงข้อมูลแล้ว

2	1	4	5	0	3	9	8	7	6
---	---	---	---	---	---	---	---	---	---

j i

2	1	4	5	0	3	6	8	7	9
---	---	---	---	---	---	---	---	---	---

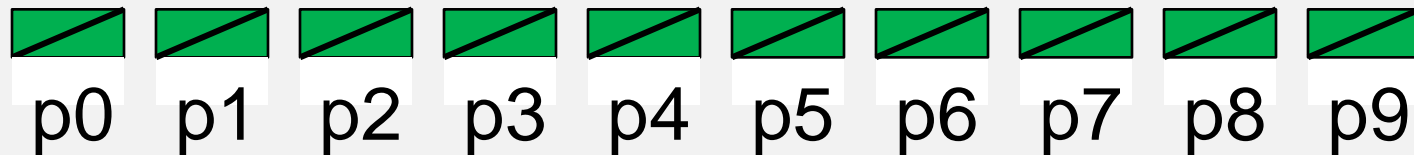
j i

9. จากนั้น recursive ทำซ้ำครึ่งหน้าและครึ่งหลัง



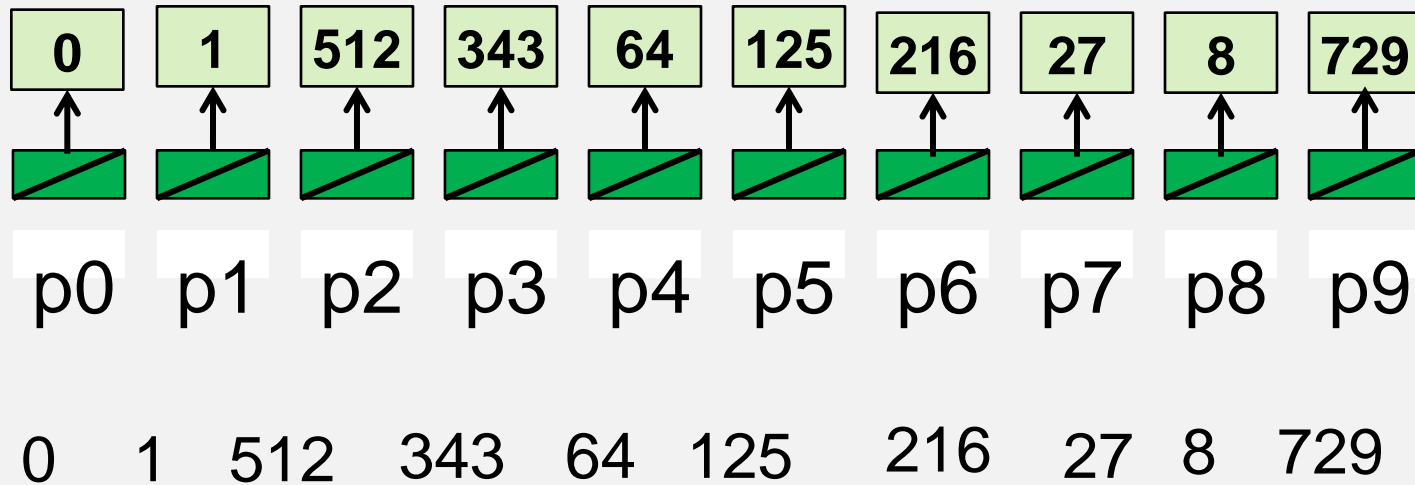
## 6.5 Radix Sort

Input 64, 8, 216, 512, 27, 729, 0, 1, 343, 125

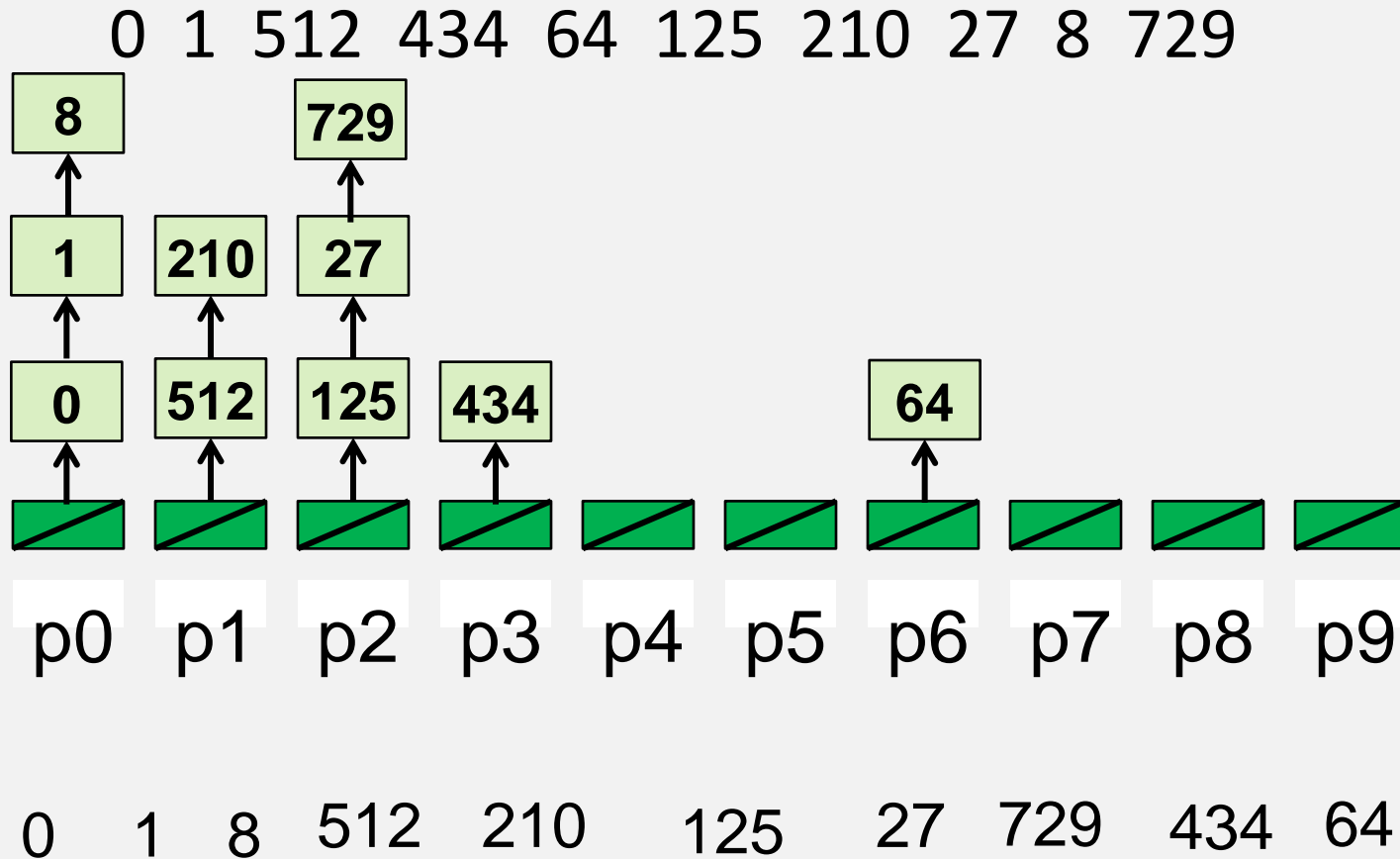




Input 64, 8, 216, 512, 27, 729, 0, 1, 343, 125



หลักหน่วย



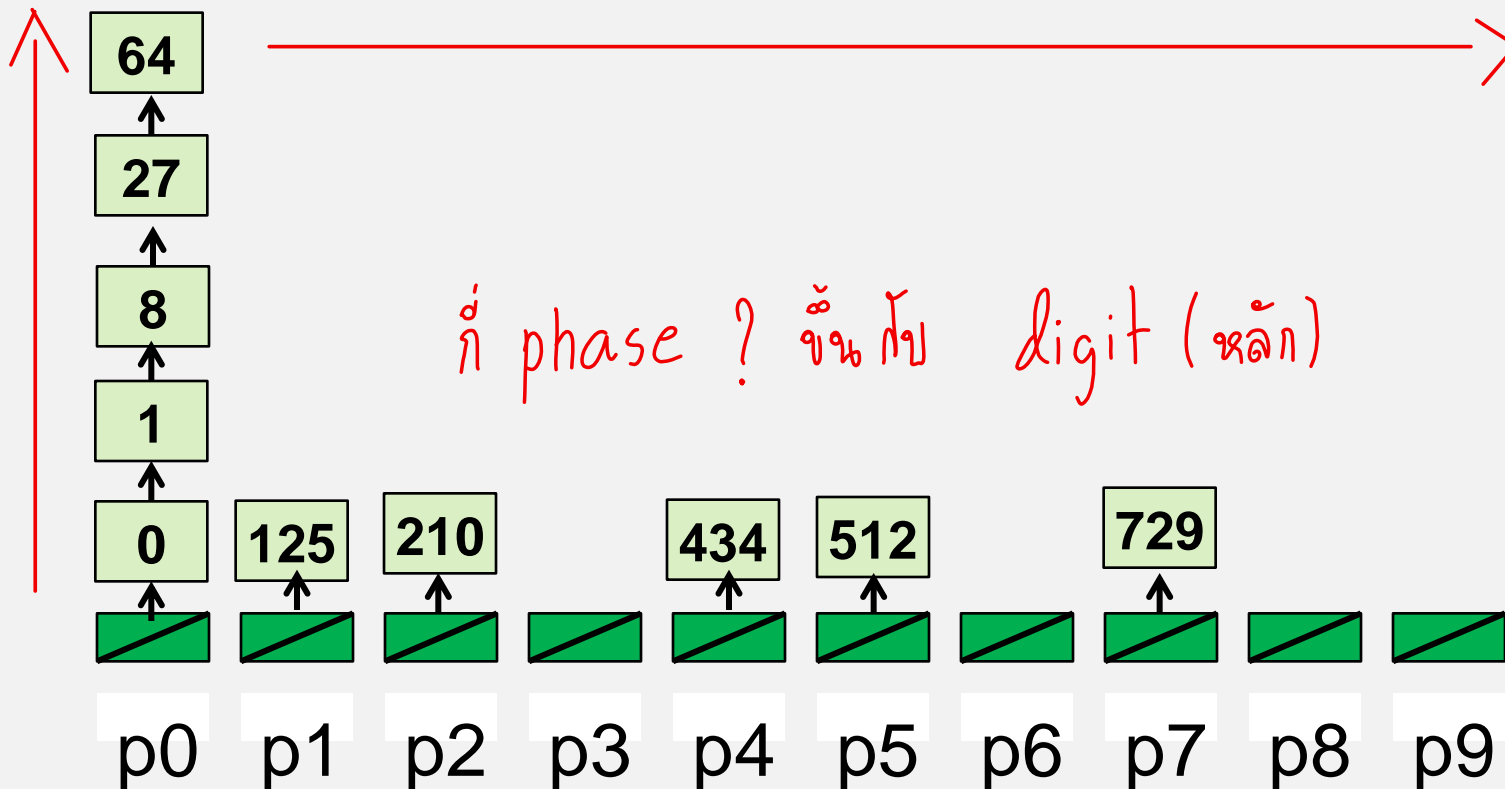
หลักสิบ





0 1 8 512 210 125 27 729 434 64

# หลักร้อย





การหาหลักหน่วย :  $n \% 10$     0-9

การหาหลักสิบ :  $(n \% 100) / 10$     0-99

การหาหลักร้อย = ?     $n / 100$

การหาหลักพัน = ?