



MODULE6–SORT

6.0 Sort

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



- 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



Algorithm

1. Basic/Simple Sorts $O(n^2)$
 - Bubble Sort
 - Selection Sort
 - Insertion Sort
2. Efficient Sorts $O(n \log_n)$ นำไปใช้จริงบ่อยที่สุด
 - Merge Sort
 - Quick Sort
 - Heap Sort
3. Non-Comparison Sorts $O(n+k)$ หรือ $O(n)$
 ใช้การกระจายตัวของข้อมูล
 - Counting Sort
 - Radix Sort
 - Bucket Sort



สิ่งที่จะต้องพิจารณา

1. ความซับซ้อนของเวลา (Time Complexity):

ประสิทธิภาพในการทำงานเมื่อขนาดข้อมูล (n) เพิ่มขึ้น

2. กรณีแย่มากที่สุด (Worst Case): $O(n^2)$ (Bubble, Selection, Insertion) หรือ $O(n \log_n)$ (Merge, Heap, Quick)

3. ความซับซ้อนของพื้นที่ (Space Complexity): พื้นที่หน่วยความจำที่ต้องใช้เพิ่มเติมนอกเหนือจากพื้นที่เก็บข้อมูลเริ่มต้น



1. Selection Sort

1. ค้นหาค่า **น้อยสุด** จากส่วนที่ยังไม่เรียงลำดับ (0 ถึง $n-1$)
แล้วนำไปวางที่ตำแหน่งแรก
2. ค้นหาค่า **สูงที่สุด** จากส่วนที่ยังไม่เรียงลำดับ (1 ถึง $n-1$)
แล้วนำไปวางที่ตำแหน่งถัดจากตำแหน่งแรก
3. ค้นหาค่า **สูงที่สุด** จากส่วนที่ยังไม่เรียงลำดับ (2 ถึง $n-1$)
แล้วนำไปวางที่ตำแหน่ง ถัด ถัดจากตำแหน่งแรก

.....

ทำเช่นนี้จนหมดข้อมูล



1. Selection Sort

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

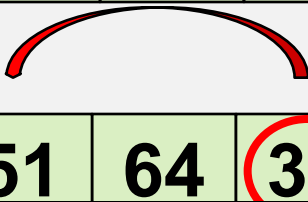


p=3



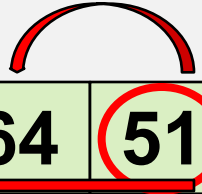
8	21	64	51	32	34
8	21	32	51	64	34

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



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



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)

รอบสุดท้ายที่เหลือตัวเดียวไม่นิยม หา min

Analysis of Selection sort:

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



2. Bubble Sort

ทำการเทียบค่าทีละคู่ จากด้านท้ายไปด้านหน้า ถ้าค่าใดน้อยกว่า จะถูกสลับไปด้านหน้า เช่นเดียวกับฟองอากาศในน้ำที่เบา จะลอยขึ้นไปด้านหน้า



2. Bubble Sort

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

P=1

34	8	64	51	32	21



P=2

8	34	21	64	51	32



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



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



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

Analysis of Selection sort:

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

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



3. Insert Sort

นำข้อมูลทีละค่ามาใส่ในตำแหน่งที่ถูกต้องของส่วนที่ เรียงลำดับ
แล้ว เหมือนกับการเรียงไฟในมือ



3. Insertion Sort

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

ผล

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




P=3



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

ผล


P=4



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

ผล

P=5



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

ผล



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 $O(n^2)$

กรณีดีที่สุด ?



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

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)

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



```
int a[6]={38,18,12,7};
int p,j,tmp,N=4;
for ( p=1; p<N; p++ ) // phase
{
    for(j=p;j>0;j--)
    {
        if(a[j]<a[j-1])
        {
            tmp=a[j];
            a[j]=a[j-1];
            a[j-1]=tmp;
        }
        else break;
    }
}
```



Efficient Sorts $O(n \log_n)$

4. Merge Sort

ใช้หลักการ **Divide and Conquer** (แบ่งและเอาชนะ): แบ่งข้อมูลออกเป็นส่วนเล็กๆ จนกว่าจะเหลือเพียงตัวเดียว จากนั้น **รวม (Merge)** ส่วนที่เรียงลำดับแล้วเข้าด้วยกัน



4. Merge Sort

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

1	13	5	2	18	1	12	3
---	----	---	---	----	---	----	---

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

18	1	12	3
----	---	----	---

1	13
---	----

5	2
---	---

18	1
----	---

12	3
----	---

1

13

5

2

18

1

12

3



1	2	3	5	8	11	12	13
---	---	---	---	---	----	----	----

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

3	8	11	12
---	---	----	----

1	13
---	----

2	5
---	---

11	18
----	----

3	12
---	----

1

13

5

2

18

11

12

3



สมมุติว่ามีข้อมูล 4 ตัว

Sort

0

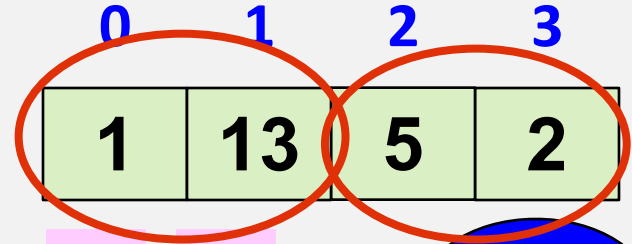
3

26

```
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

A

B

2

3

0

2

3

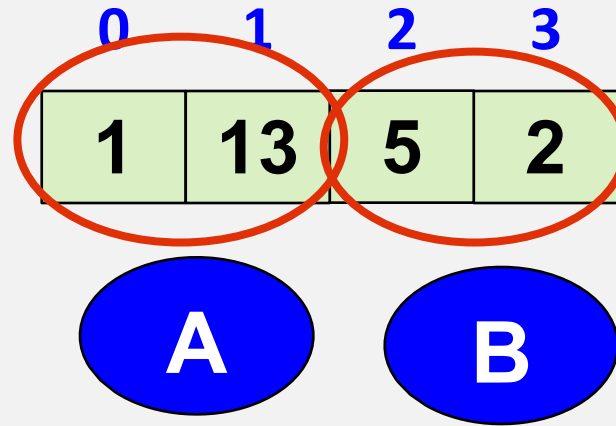
C



สมมติว่ามีข้อมูล 4 ตัว

Sort

27



พิจารณาที่ A



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;

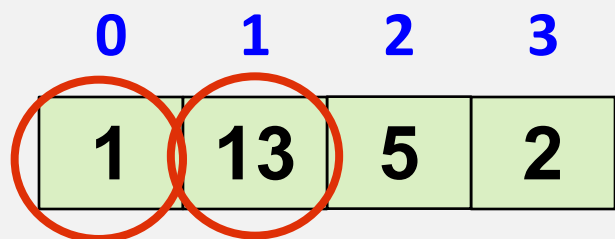
5) msort(A,tmparray,left,center);

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

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

8) }

9) }



0

0

0

A1

A2

1

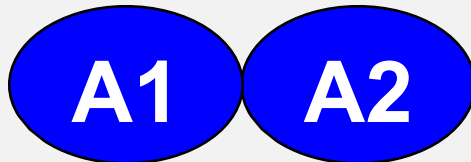
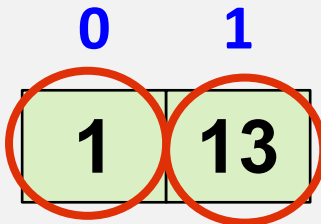
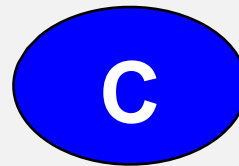
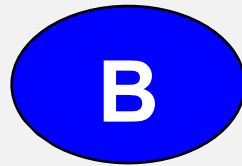
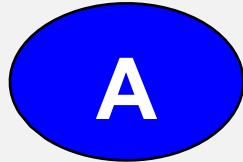
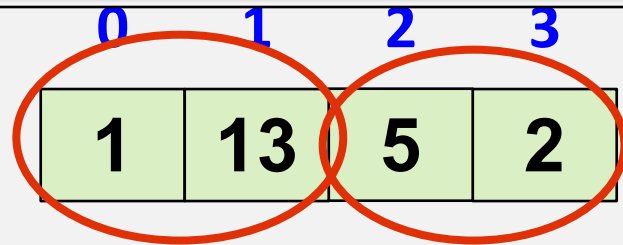
1

0

1

1

A3





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)}



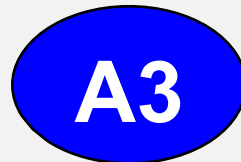
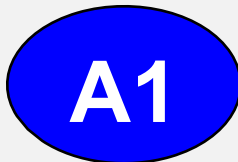
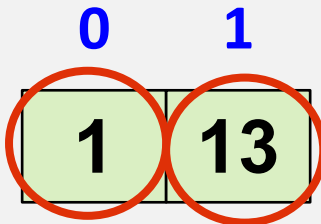
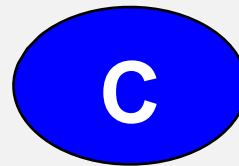
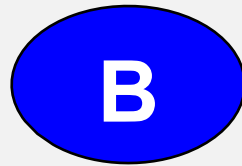
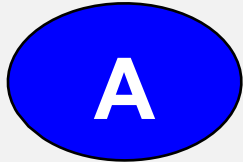
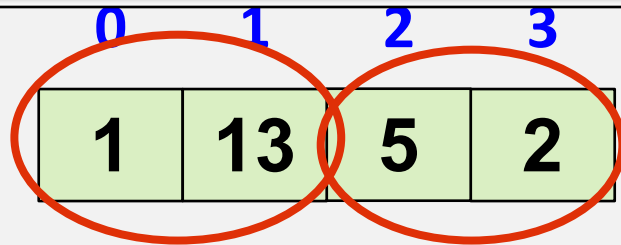
A2

1

1

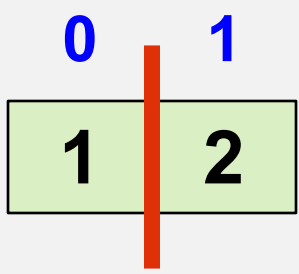
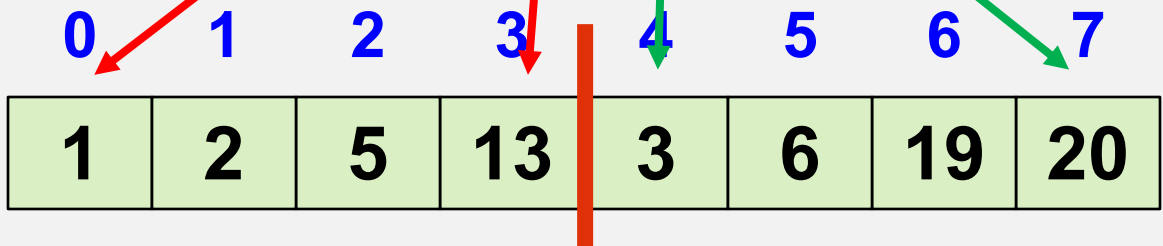
```
1) void msort(int A[],int tmparray[],int left, int right)
2) { int center;
3)   if ( left < right) F
4)   { center = (left+right)/2;
5)     msort(A,tmparray,left,center);
6)     msort(A,tmparray,center+1,right);
7)     merge(A,tmparray,left,center+1,right);
8)   }
9) }
```

0	1	2	3
1	13	5	2





จำชื่อ



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

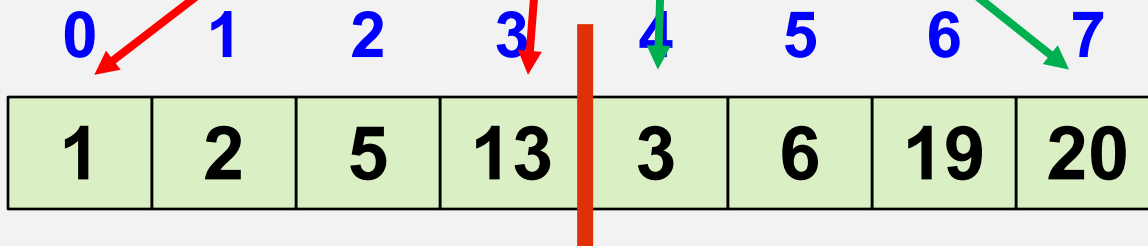
lpos =
leftend =
rpos =
rightend =



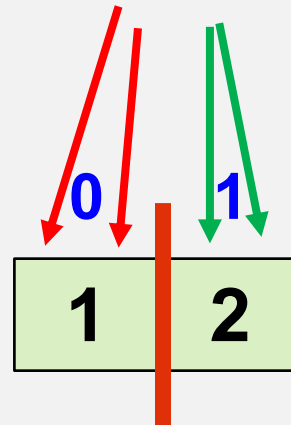
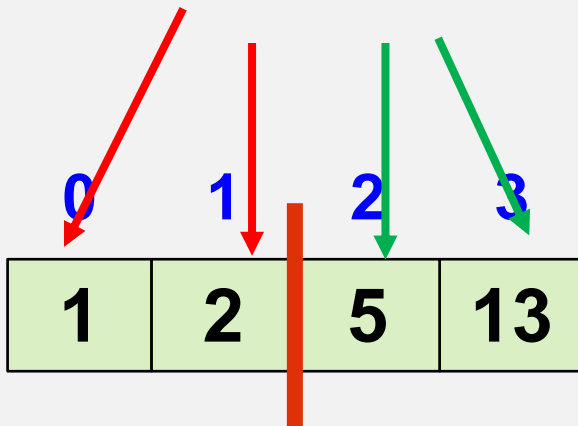
03603212 : Module6 – Sort

34

จำชื่อ



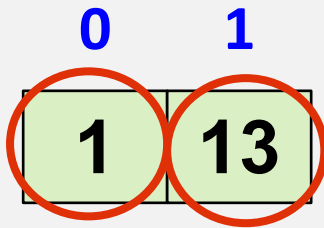
lpos = 0
leftend = 3
rpos = 4
rightend = 7



lpos =
leftend =
rpos =
rightend =



A



พิจารณาที่ A จะจัดเรียง A3



0	1
1	13

ขณะที่ ข้อมูลครึ่งซ้ายและขวายัง merge ไม่หมด
ถ้า value ด้านหน้าของครึ่งซ้าย ด้านหน้าของครึ่งขวา
เอาตัวมากไปใส่ Arrayผลลัพธ์
ตน. ด้านหน้า ของตัวมาก++
ตน. array ผลลัพธ์++

ถ้าข้อมูลหมดด้านเดียว
เอาข้อมูลที่เหลือวนใส่ array ผลลัพธ์



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;

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

0

0

0

A1

A2

1

1

0

1

1

A3



0 1 2 3

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

A

B

C

0 1

1	13
---	----

5	2
---	---

A1 A2

A3

B1 B2

B3

A1 A2

A3

A1 A2

A3



0360

A3

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

0 1 1

0 1 1

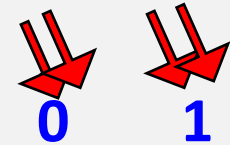
void merge(int A[], int tmparray[],int lpos,int rpos,int rightend)

{ int i, leftend, numelements,tmppos;

leftend = rpos -1;

tmppos = lpos;

numelements = rightend-lpos +1;



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

2

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

lpos	หน้าซ้าย
leftend	หลังซ้าย
rpos	หน้าขวา
rightend	หลังขวา



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

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

1 <= 13

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

```
    else
```

ต่น. sort

ค่าน้อย

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

0 1 2 3

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

1			
----------	--	--	--

lpos = 0

leftend = 0

rpos = 1

rightend = 1

tmpppos = 0

numelement = 2



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

0	1	2	3
1	13	5	2
1	13		



0 1 2 3

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

A

B

C

0 1

1	13
---	----

2 3

2	5
---	---

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

0

1

2

3

1

13

2

5

4



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

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

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

```
    else
```

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

0 1 2 3

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

--	--	--	--

lpos = 0

leftend = 1

rpos = 2

rightend = 3

tmpppos = 0

numelement = 4



```
while(lpos <= leftend)
    tmparray[tmppos++]= A[lpos++];
while(rpos <= rightend)
    tmparray[tmppos++]= 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

0	1	2	3
1	13	2	5



5. Quick Sort

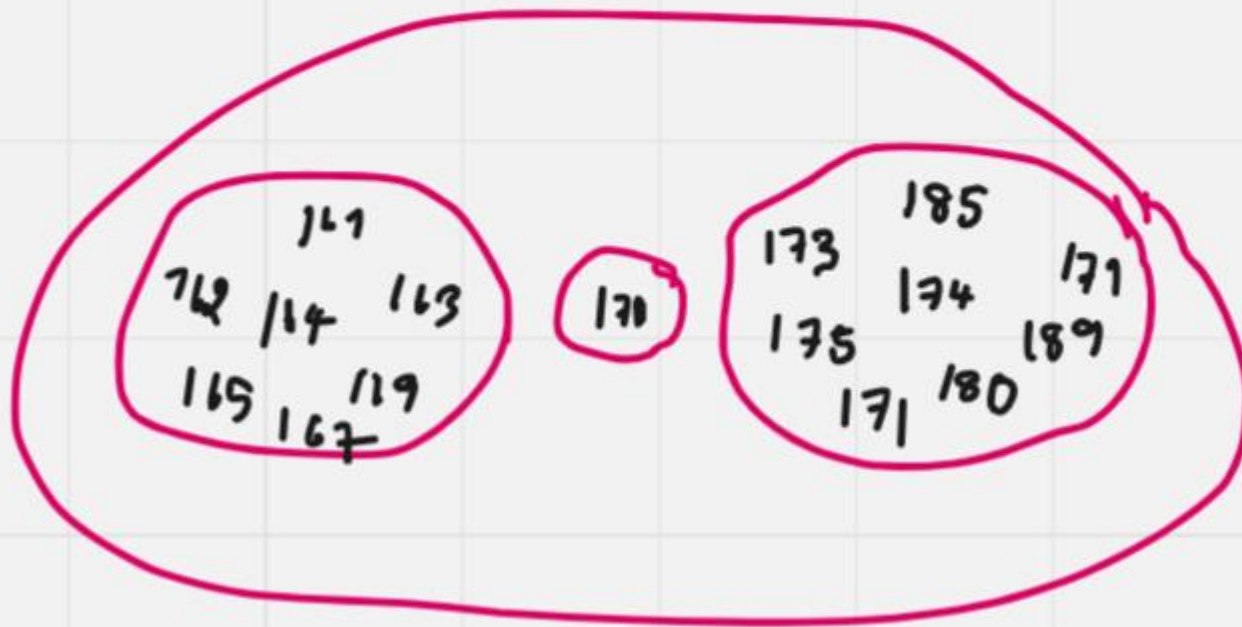
ใช้หลักการ Divide and Conquer: เลือก Pivot (ตัวหลัก) มาหนึ่งตัว แล้วแบ่ง (Partition) ข้อมูลที่เหลือออกเป็นสองกลุ่มคือกลุ่มที่น้อยกว่า Pivot และกลุ่มที่มากกว่า Pivot จากนั้นเรียกซ้ำ (Recursion) กับทั้งสองกลุ่ม 🚀

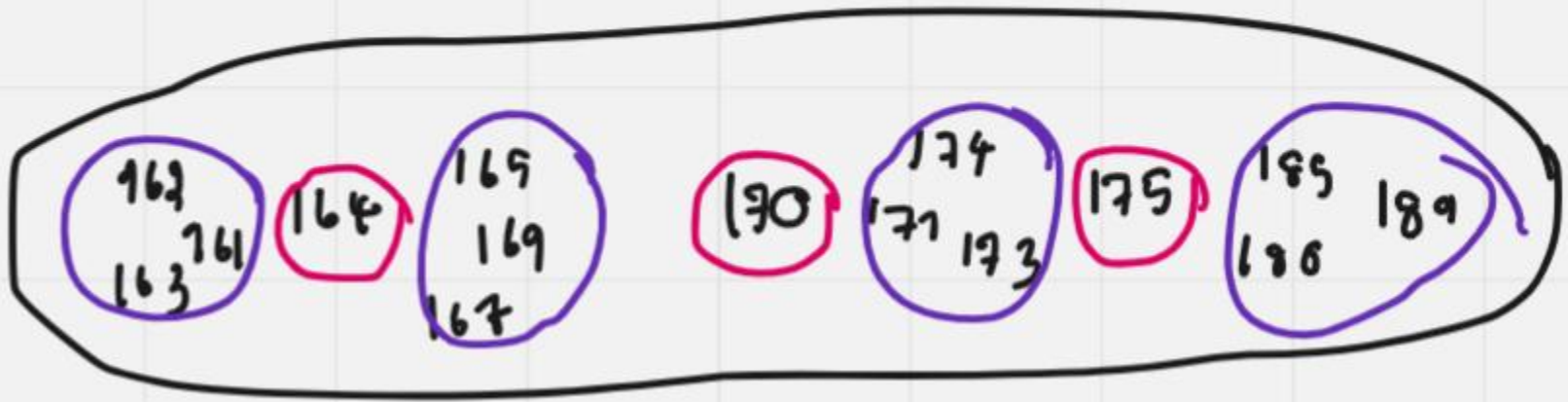
Heap

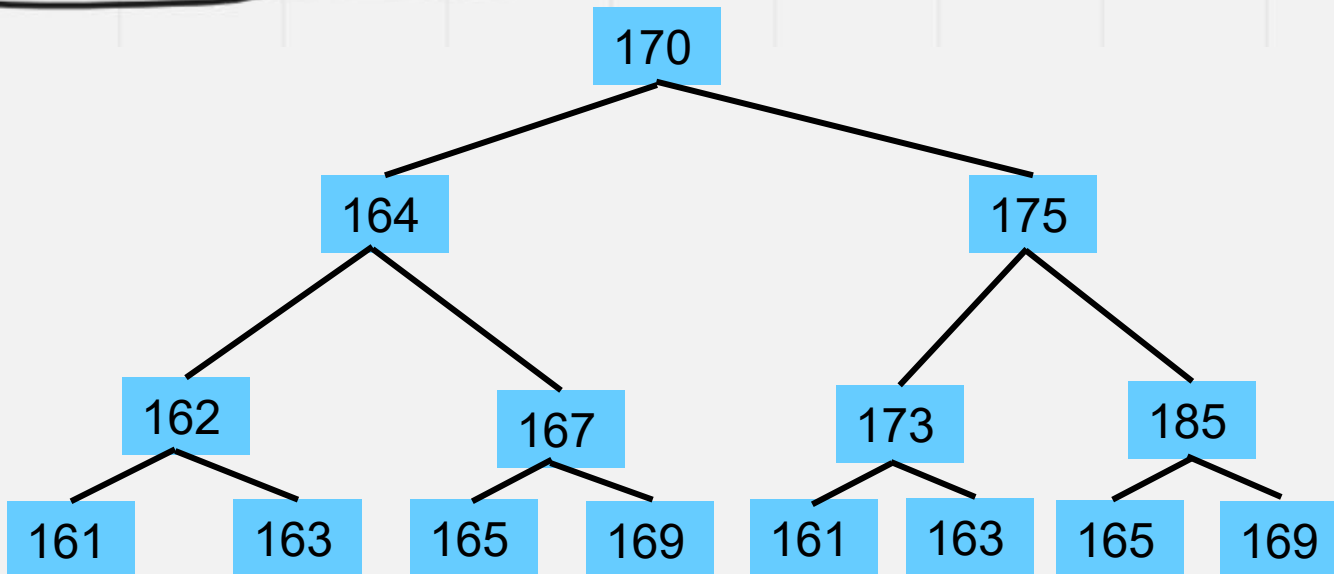
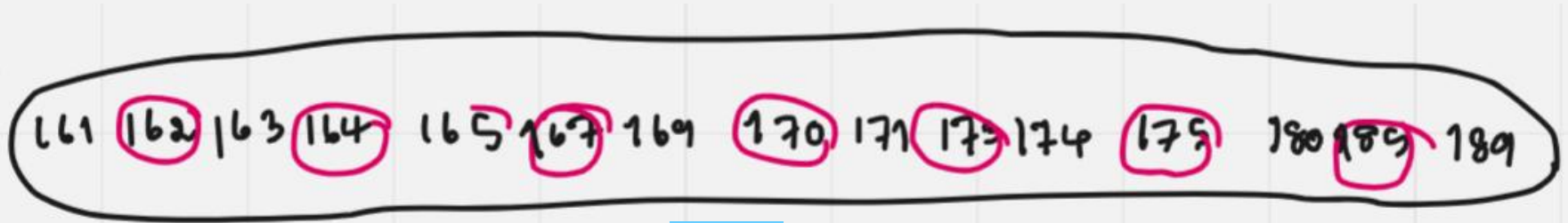


5. Quick Sort











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



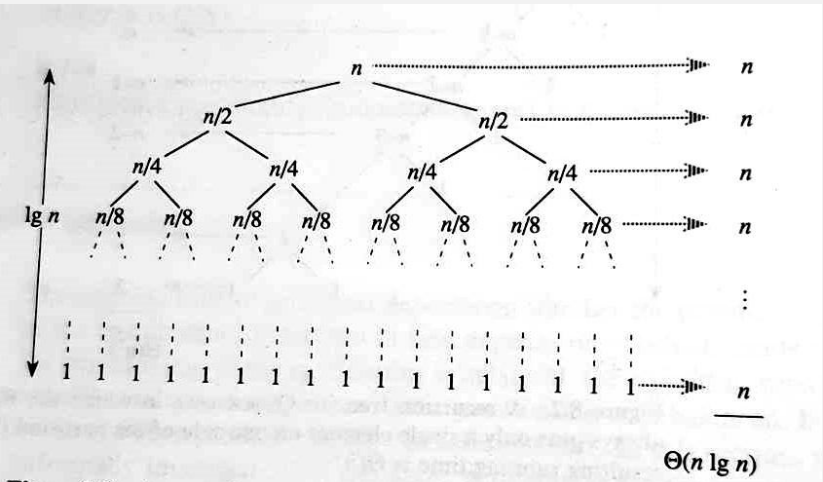
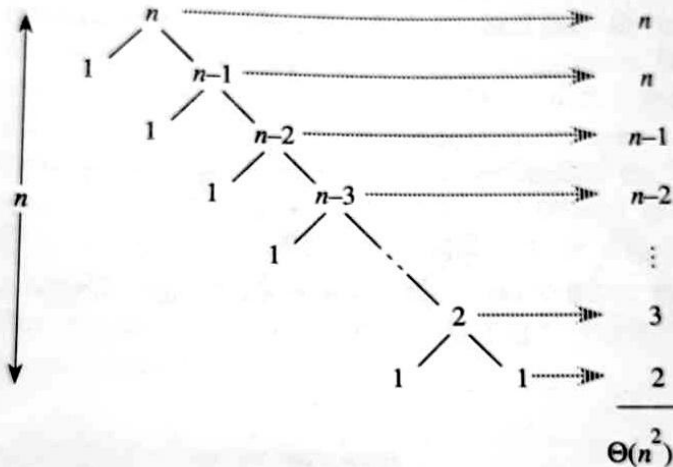
x, x, x, **x**, x, x, x, **x**, x, x, x, **x**, x, x, x

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



Worst-case $O(n^2)$

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 = $(\text{Left} + \text{Right})/2 = (1+10)/2 = 5$

$v = 6$

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

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

Correct : sorted input.



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