

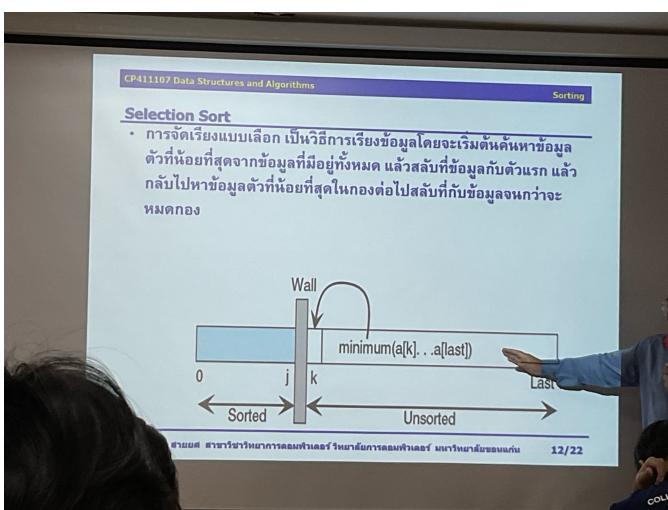
template <class Item>  
void insertion\_sort(Item data[], size\_t n)  
{  
 size\_t i, j;  
 Item temp;  
  
 if(n < 2) return; // nothing to sort!!  
  
 for(i = 1; i < n; ++i)  
 {  
 // take next item at front of unsorted part of array  
 // and insert it in appropriate location in sorted part of array  
 temp = data[i];  
 for(j = i; data[j-1] > temp and j > 0; --j)  
 data[j] = data[j-1]; // shift element forward  
  
 data[j] = temp;  
 }  
}

## Insertion Sort Time Analysis

- In O-notation, what is:
  - Worst case running time for  $n$  items?
  - Average case running time for  $n$  items?
- Steps of algorithm:
 

```
for i = 1 to n-1
        take next key from unsorted part of array
        insert in appropriate location in sorted part of array:
          for j = i down to 0,
            shift sorted elements to the right if key > key[i]
        increase size of sorted array by 1
```

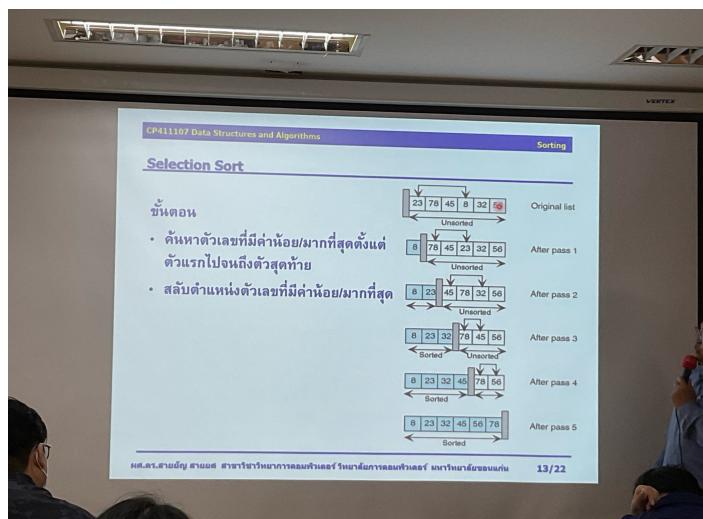
Outer loop:  $O(n)$   
Inner loop:  $O(n)$



$$A = [8, 23, 32, 45, 56, 78]$$

$$A[0], A[1] = A[1], A[0]$$

$$A = ?$$



```

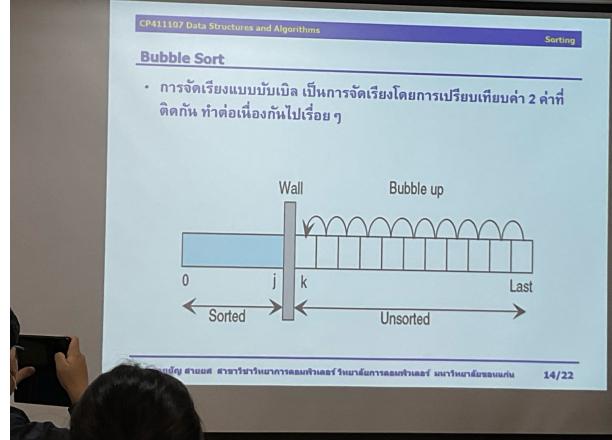
template<class Item>
void selection_sort(Item data[ ], size_t n)
{
    size_t i, j, smallest;
    Item temp;

    if(n < 2) return; // nothing to sort!!

    for(i = 0; i < n-1; ++i)
    {
        // find smallest in unsorted part of array
        smallest = i;
        for(j = i+1; j < n; ++j)
            if(data[smallest] > data[j]) smallest = j;

        // put it at front of unsorted part of array (swap)
        temp = data[i];
        data[i] = data[smallest];
        data[smallest] = temp;
    }
}

```



CP411107 Data Structures and Algorithms      Sorting

### Bubble Sort

i=0	1	2	3	4	5	6
42	13	13	13	13	13	13
20	42	14	14	14	14	14
17	20	42	15	15	15	15
13	17	20	42	17	17	17
28	14	17	20	42	20	20
14	28	15	17	20	42	23
23	15	28	23	23	23	28
15	23	23	28	28	28	42

ผลลัพธ์ สามารถดู รายการเรียนการสอนพื้นที่และห้องเรียนการสอนพื้นที่ ห้องเรียนการสอนพื้นที่ มหาวิทยาลัยขอนแก่น  
15/22

CP411107 Data Structures and Algorithms      Sorting

```

template <class Item>
void bubble_sort(Item data[ ], size_t n)
{
    size_t i, j;
    Item temp;

    if(n < 2) return; // nothing to sort!!

    for(i = 0; i < n-1; ++i)
    {
        for(j = 0; j < n-1-i; ++j)
            if(data[j] > data[j+1]) // if out of order, swap!
            {
                temp = data[j];
                data[j] = data[j+1];
                data[j+1] = temp;
            }
    }
}

```

CP411107 Data Structures and Algorithms      Sorting

### การวัดประสิทธิภาพ

**Big-O**

เป็นตัวบ่งชี้ความซับซ้อนของอัลกอริズึม ซึ่งจะแสดงให้เห็นประสิทธิภาพ ของอัลกอริズึมหนึ่ง

n	Number of Loops			
	Straight Insertion	Shell	Heap	Quick
25	625	55	116	Bubble
100	10,000	316	664	
500	250,000	2,364	4,482	
1,000	1,000,000	5,623	9,965	
2,000	4,000,000	13,374	10,985	

ผลลัพธ์ สามารถดู รายการเรียนการสอนพื้นที่ และห้องเรียนการสอนพื้นที่ ห้องเรียนการสอนพื้นที่ มหาวิทยาลัยขอนแก่น  
16/22

CP411107 Data Structures and Algorithms      Sorting

### การวัดประสิทธิภาพ

```

void swap(int data[], int a, int b)
{
    int temp;
    temp = data[a];
    data[a] = data[b];
    data[b] = temp;
} /* swap */

void insertion_sort(int data[], int num_elts)
{
    int i, j;
    for (i=0; i<num_elts; i++)
        for (j=i; (j>0) && (data[j] < data[j-1]); j--)
            swap(data, j, j-1);
    } /* insertion_sort */

```

Big-O =  $n^2$

ผลลัพธ์ สามารถดู รายการเรียนการสอนพื้นที่ และห้องเรียนการสอนพื้นที่ ห้องเรียนการสอนพื้นที่ มหาวิทยาลัยขอนแก่น  
17/22

CP411107 Data Structures and Algorithms

### การวัดประสิทธิภาพ

```

void selection_sort(int data[], int num_elts)
{
    int i, j, lowindex;
    for (i=0; i<num_elts; i++) /* Select i'th record */
    {
        lowindex = i; /* Remember its index */
        for (j=num_elts-1; j>i; j--) /* Find the least value */
            if (data[j] < data[lowIndex])
                lowIndex = j; /* Put it in place */
        if (i != lowIndex) /* To reduce swap() operation */
            swap(data, i, lowIndex);
    }
} /* selection_sort */

```

**Big-O =  $n^2$**

หมายเหตุ ค่ารากวิภาคการออมที่น้อยที่สุดในการออมที่น้อยที่สุด มากกว่าหนึ่งเดือนก่อน

18/22

CP411107 Data Structures and Algorithms

### การวัดประสิทธิภาพ

```

void bubble_sort(int data[], int num_elts)
{
    int i, j;
    for (i=0; i<num_elts; i++)
        for (j=num_elts-1; j>i; j--)
            if (data[j] < data[j-1])
                swap(data, j, j-1);
} /* bubble_sort */

```

**Big-O =  $n^2$**

หมายเหตุ ค่ารากวิภาคการออมที่น้อยที่สุดในการออมที่น้อยที่สุด มากกว่าหนึ่งเดือนก่อน

19/22

CP411107 Data Structures and Algorithms

### การวัดประสิทธิภาพ

Name	Average	Worst
Bubble sort	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Quicksort	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$
Selection sort	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Insertion sort	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Shell sort	—	$\mathcal{O}(n \log^2 n)$
Binary tree sort	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$
Merge sort	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$
Heapsort	$\mathcal{O}(n \log n)$	$\mathcal{O}(n \log n)$

*k*, the size of each key  
*s*, the chunk size used by the implementation.

least significant digit (LSD)  
most significant digit (MSD)

หมายเหตุ ค่ารากวิภาคการออมที่น้อยที่สุดในการออมที่น้อยที่สุด มากกว่าหนึ่งเดือนก่อน

20/22

CP411107 Data Structures and Algorithms

### งาน

การบันทึกข้อมูลด้วยโปรแกรมเรียงซ่อมูลตัวเลข โดยเก็บชื่อในลิส ขนาด 5 จำนวน

โดยใช้วิธีการ

- = 0 Insertion Sort
- = 1 Selection Sort
- = 2 Bubble Sort

หมายเหตุ ค่ารากวิภาคการออมที่น้อยที่สุดในการออมที่น้อยที่สุด มากกว่าหนึ่งเดือนก่อน

23/22