CANTHO UNIVERSITY

Chapter 3: Basic Sort and Search
Part 3.1: Sorting

Lâm Hoài Bảo - FSE – CICT
Trương Minh Thái – FSE - CICT

www.ctu.edu.vn

1

CANTHO UNIVERSITY

Agenda

- Introduction
- Some O(n^2) algorithms
- Shell sort
- Summary

www.ctu.edu.vn 2

2

What is sorting?

- The process of placing elements in a collection in some kinds of order
  - Sort a list of word alphabetically
  - Sort a list of cities by population or area
- Problem
  - Input: A sequence of n elements: $<a_0, a_1, \ldots, a_{n-1}>$
  - Output: A permutation $<a'_0, a'_1, \ldots, a'_{n-1}>$ of the input sequence such that: $a'_0 <= a'_1 <= \ldots <= a'_{n-1}$

www.ctu.edu.vn 3

Sorting problem

- Example 1
  - Input: a list of integers: [5, 11, 7, 5, 11, 13, 2]
  - Output: a sorted list: [2, 5, 5, 7, 11, 11, 13]
- Example 2:
  - Input: a list of points: [(1, 2), (4, 5), (3, 4), (3, 3), (2, 3)]
  - Output: a list of points of which distances to the origin are ascending
  - [(1, 2), (2, 3), (3, 3), (3, 4), (4, 5)]

www.ctu.edu.vn 4

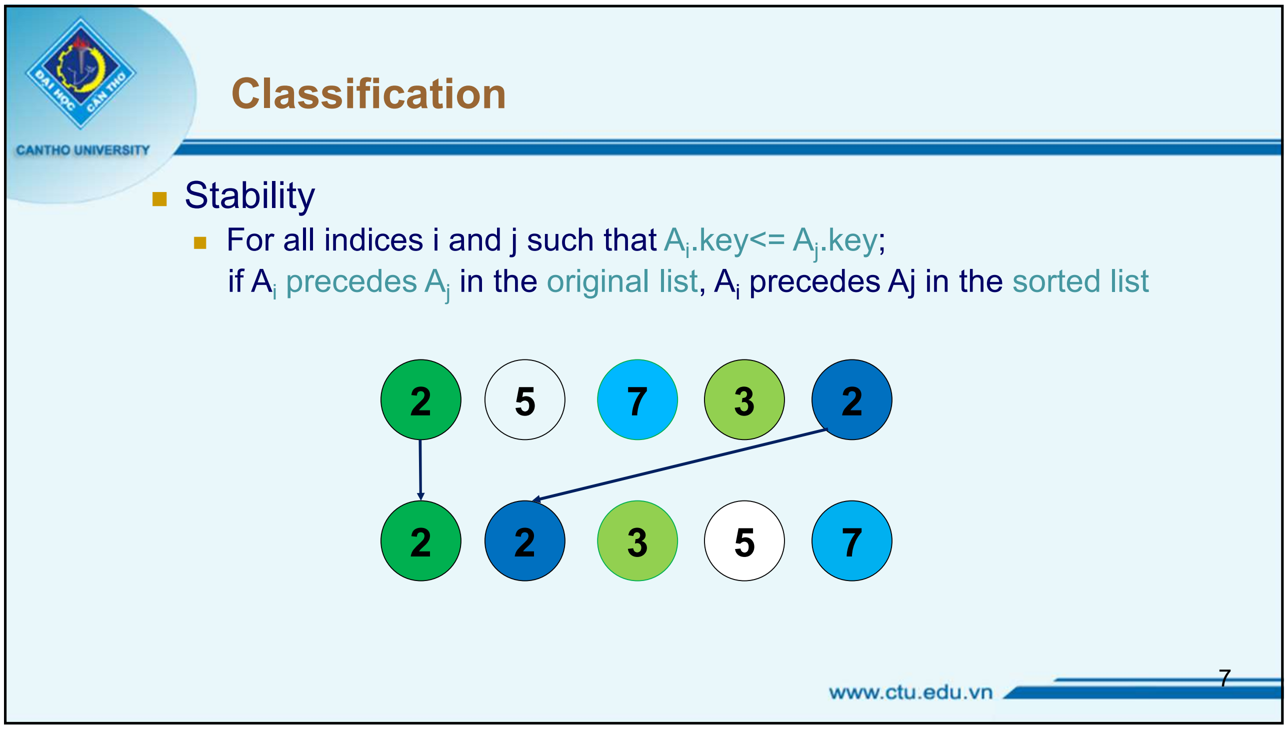
The structure of the data

- In practice, an element to be sorted ~ a record (struct)
  - key: value to be sorted
  - satellite data: remainder of the record
- The sorting algorithm permutes the key as well as the satellite data

```
typedef struct{
    KeyType key;
    SatelliteType other;
}ElementType;
```

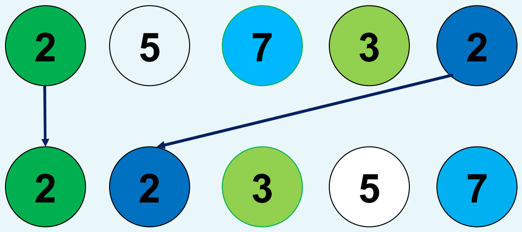
Classification

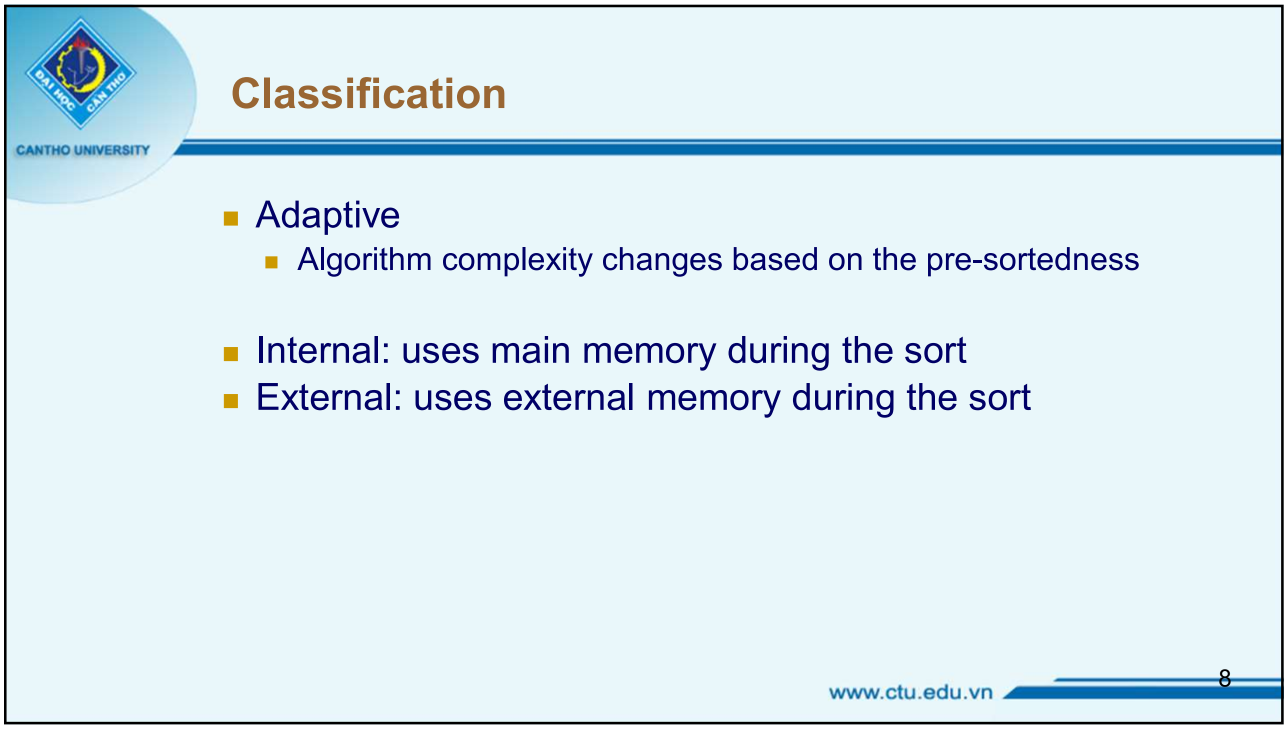
- Number of comparisons
  - Comparison sorting: bubble, selection, insertion, quick sort, …
  - Non comparison sorting: counting, radix, bucket sort, …
- Number of swaps (shifts) - inversions
- Memory usage:
  - Some algorithms are *in place*: need O(1) or O(logn) memory for temporary data



Classification

- Stability
  - For all indices i and j such that $A_i.key <= A_j.key$;
    if A_i precedes A_j in the original list, A_i precedes Aj in the sorted list





Classification

- Adaptive
  - Algorithm complexity changes based on the pre-sortedness
- Internal: uses main memory during the sort
- External: uses external memory during the sort

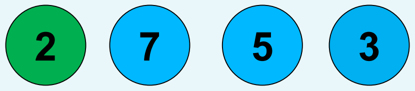
Agenda

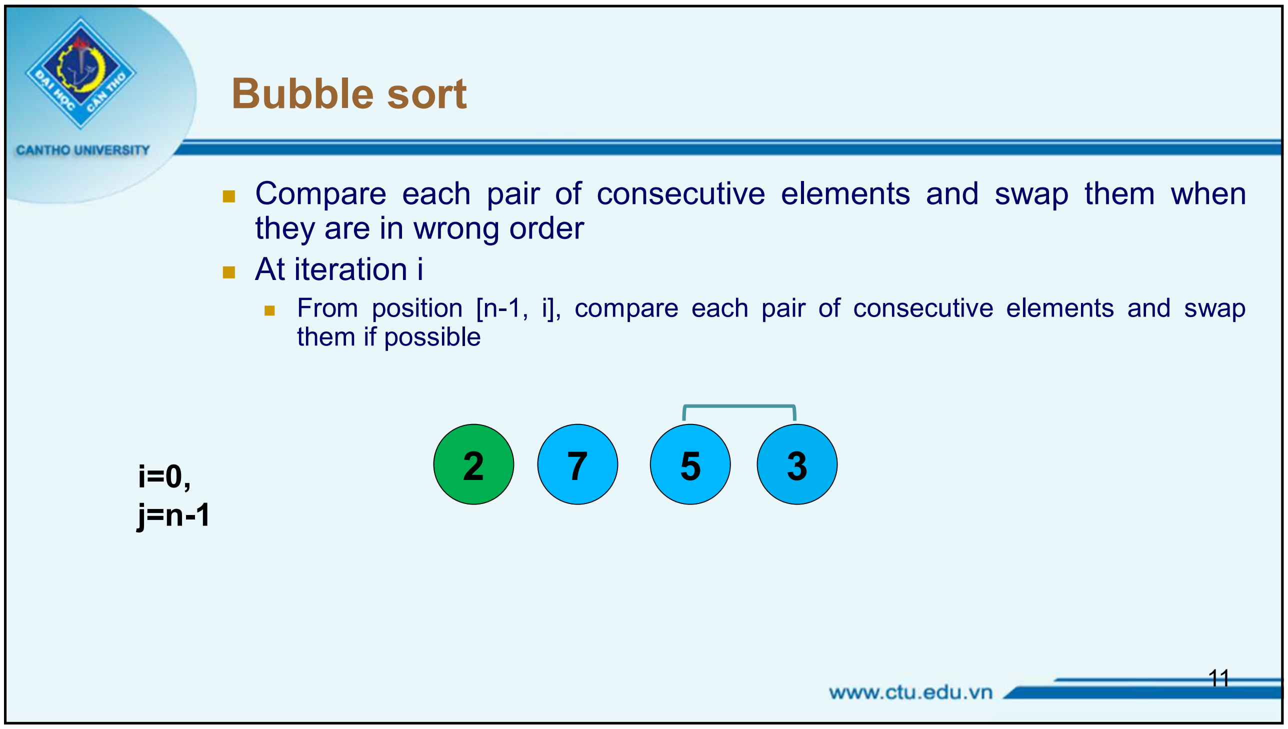
- Introduction
- Some O(n^2) algorithms
  - Bubble sort
  - Selection sort
  - Insertion sort
- Shell sort
- Summary

Bubble sort

- Compare each pair of consecutive elements and swap them when they are in wrong order
- At iteration i
  - From position [n-1, i], compare each pair of consecutive elements and swap them if possible

i=0,
j=n-1





Bubble sort

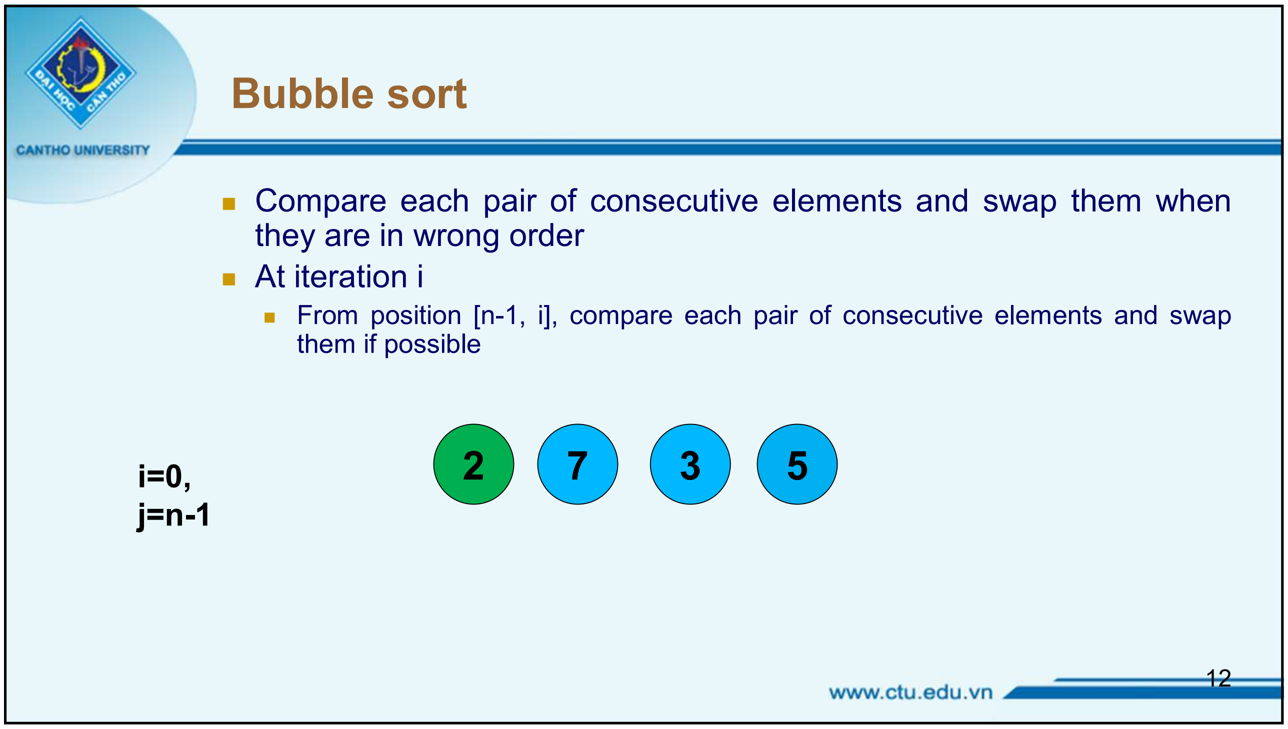
CANTHO UNIVERSITY

- Compare each pair of consecutive elements and swap them when they are in wrong order
- At iteration i
  - From position [n-1, i], compare each pair of consecutive elements and swap them if possible

i=0,
j=n-1

2 **7** **5** **3**

www.ctu.edu.vn



Bubble sort

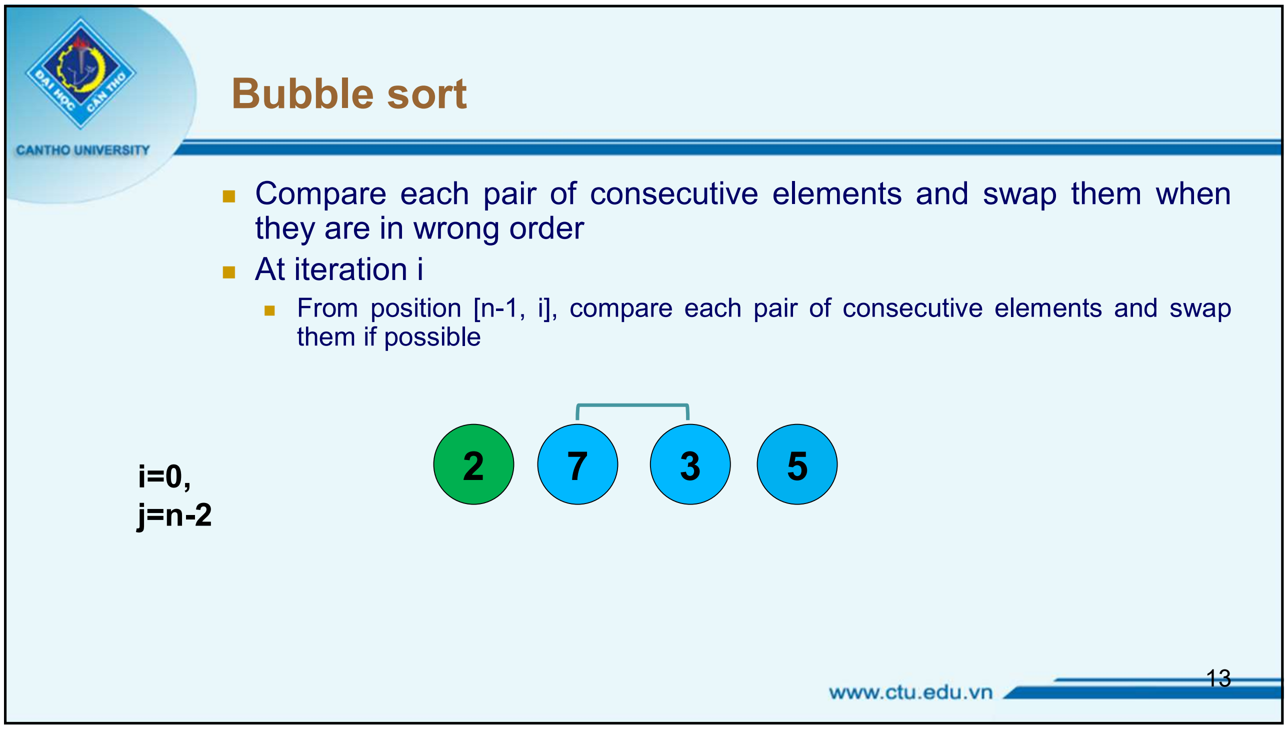
CANTHO UNIVERSITY

- Compare each pair of consecutive elements and swap them when they are in wrong order
- At iteration i
  - From position [n-1, i], compare each pair of consecutive elements and swap them if possible

i=0,
j=n-1

2 **7** **3** **5**

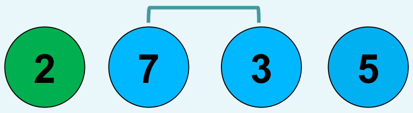
www.ctu.edu.vn



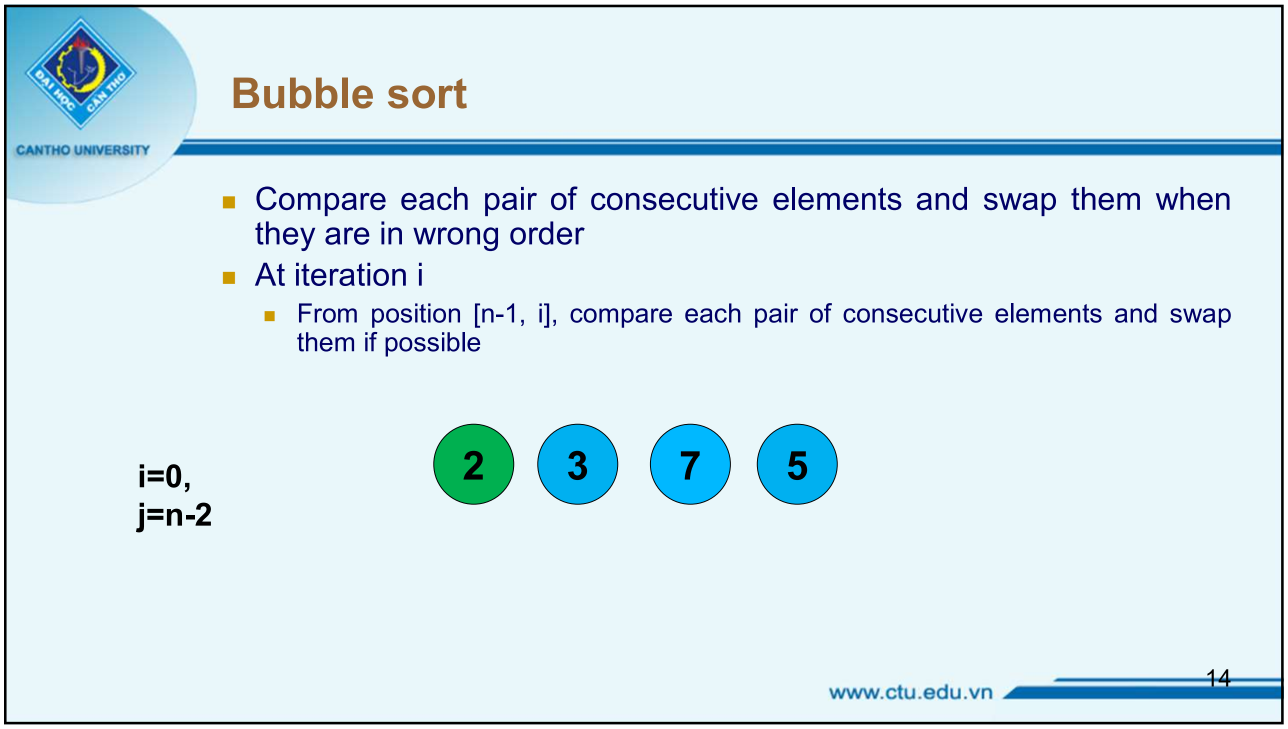
Bubble sort

- Compare each pair of consecutive elements and swap them when they are in wrong order
- At iteration i
  - From position [n-1, i], compare each pair of consecutive elements and swap them if possible

i=0,
j=n-2



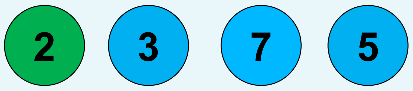
13



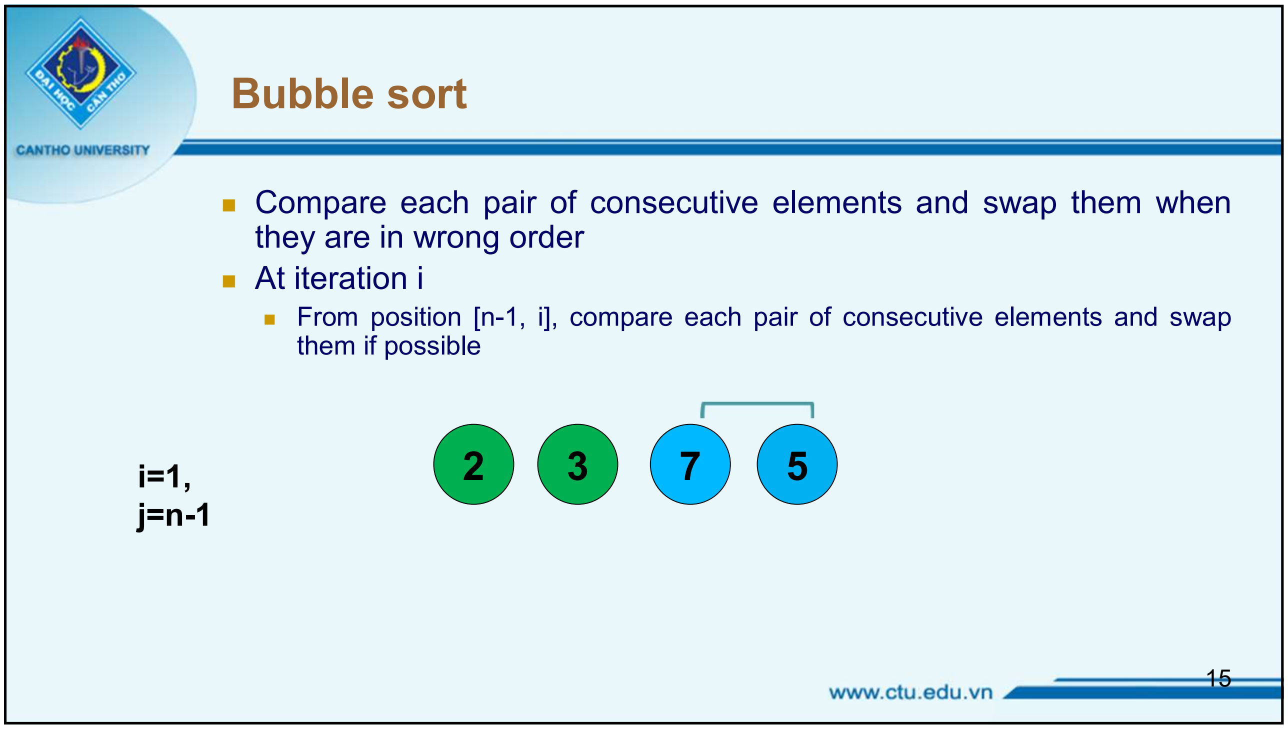
Bubble sort

- Compare each pair of consecutive elements and swap them when they are in wrong order
- At iteration i
  - From position [n-1, i], compare each pair of consecutive elements and swap them if possible

i=0,
j=n-2



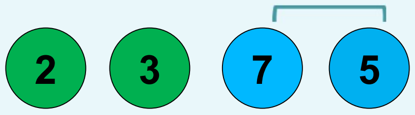
14



Bubble sort

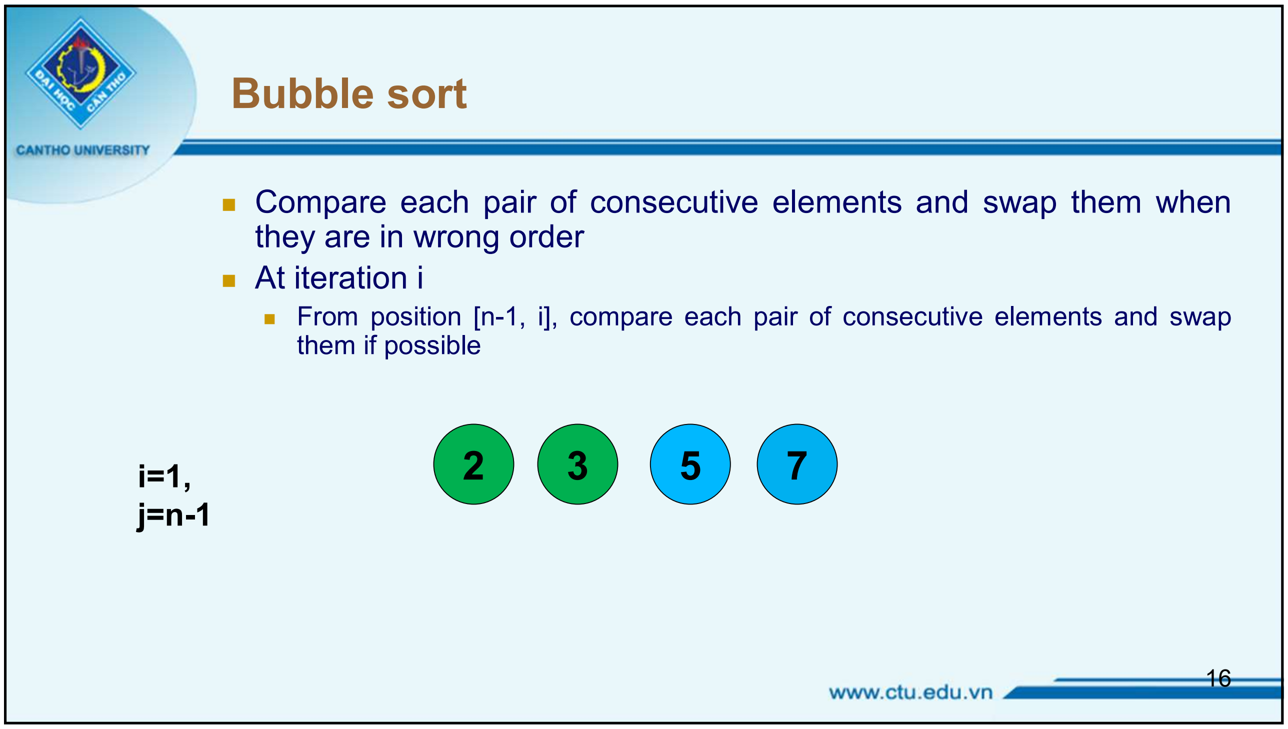
- Compare each pair of consecutive elements and swap them when they are in wrong order
- At iteration i
  - From position [n-1, i], compare each pair of consecutive elements and swap them if possible

i=1,
j=n-1



www.ctu.edu.vn 15

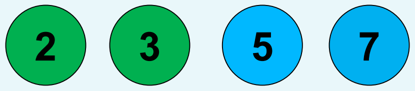
15



Bubble sort

- Compare each pair of consecutive elements and swap them when they are in wrong order
- At iteration i
  - From position [n-1, i], compare each pair of consecutive elements and swap them if possible

i=1,
j=n-1



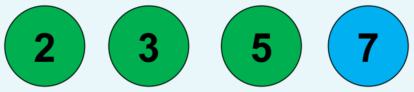
www.ctu.edu.vn 16

16

Bubble sort

- Compare each pair of consecutive elements and swap them when they are in wrong order
- At iteration i
  - From position [n-1, i], compare each pair of consecutive elements and swap them if possible

i=2,
j=n-1

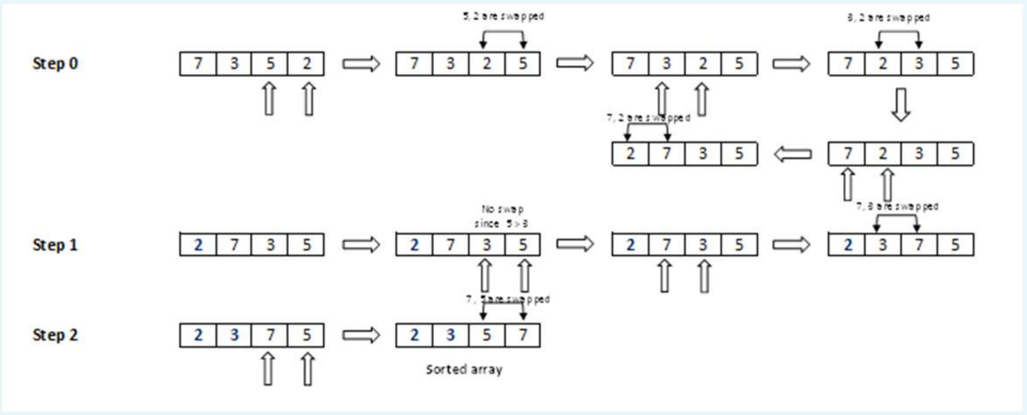


www.ctu.edu.vn 17

17

Bubble sort [2]

- Example: A = [7, 4, 5, 2]



www.ctu.edu.vn 18

18

Complexity

- Algorithm

```
void bubbleSort(ElementType A[], int n) {
    int i, j;
    for (i = 0; i < n-2; i++) {          //n-1
        for (j = n-1; j >= i+1; j--) {  //n-i-1
            if (A[j].key > A[j-1].key){ //1
                swap(&A[j], &A[j-1]);   //
            }
        }
    }
}
```

- Time complexity $T(n) = \sum_{i=0}^{n-2} n - i - 1 = \frac{n(n-1)}{2} = O(n^2)$

www.ctu.edu.vn 19

19

Agenda

- Introduction
- Some $O(n^2)$ algorithms
  - Bubble sort
  - Selection sort
  - Insertion sort
- Shell sort
- Summary

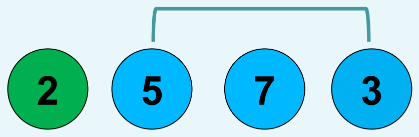
www.ctu.edu.vn 21

21

CANTHO UNIVERSITY

Selection sort

- Choose the element with smallest key (largest key) and put it to the right position
- At iteration i
  - From position [i .. n-1], choose the element with lowest key and swap it with the element i.



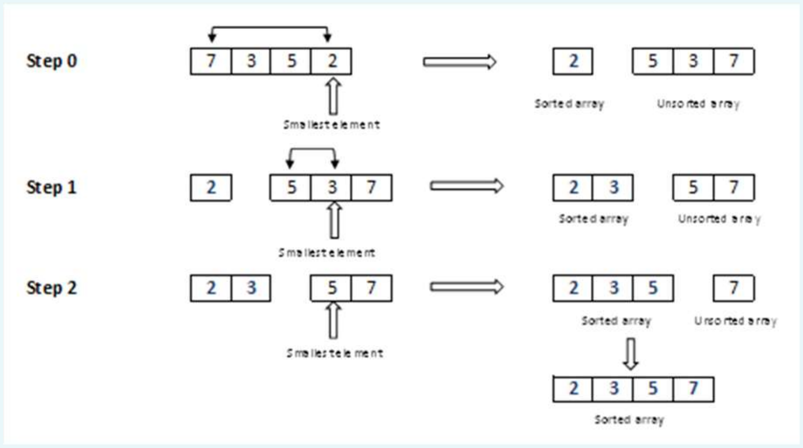
www.ctu.edu.vn 22

22

CANTHO UNIVERSITY

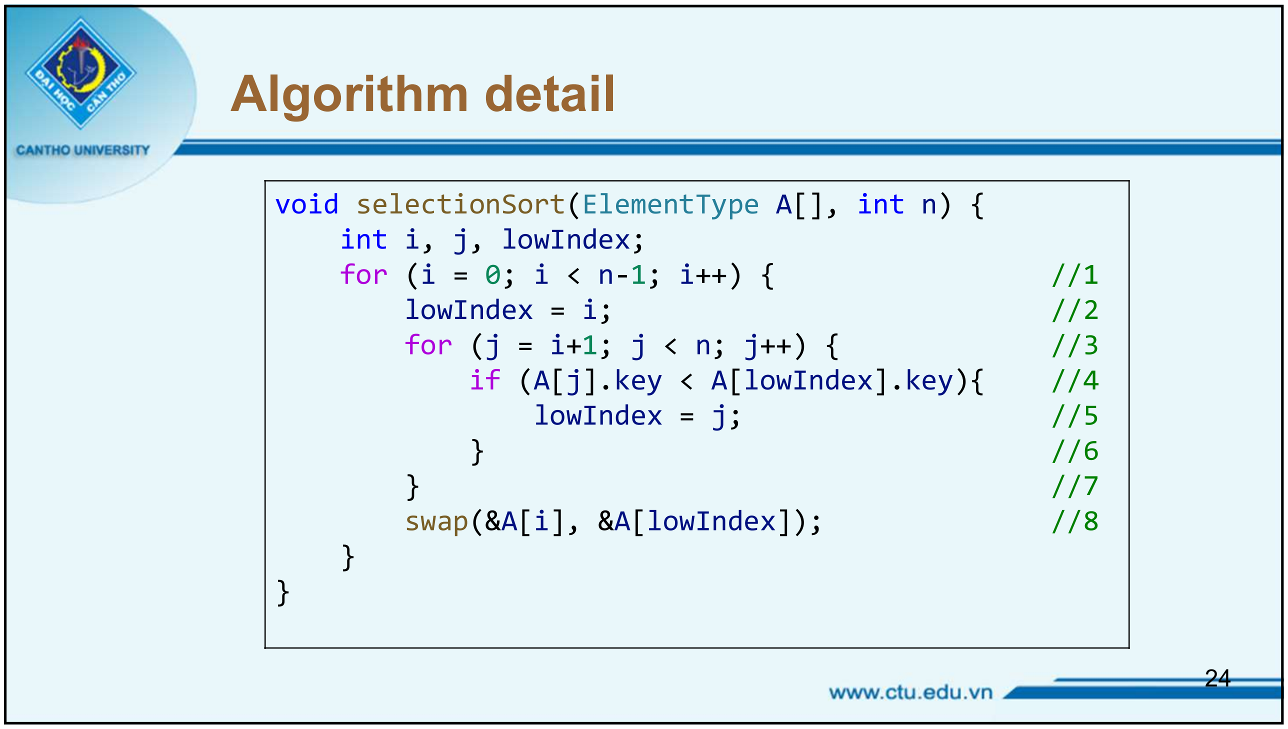
Example

- A = [7, 3, 5, 2]



www.ctu.edu.vn 23

23

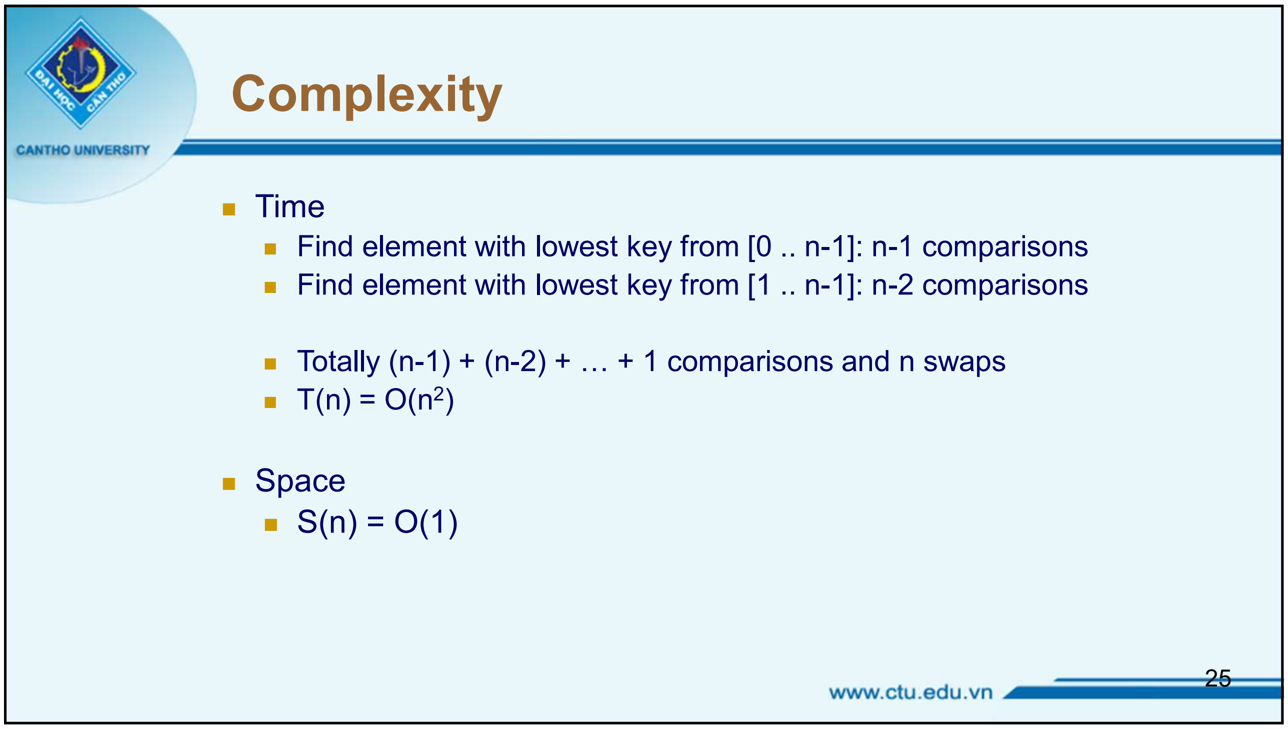


Algorithm detail

```
void selectionSort(ElementType A[], int n) {
    int i, j, lowIndex;
    for (i = 0; i < n-1; i++) {                     //1
        lowIndex = i;                               //2
        for (j = i+1; j < n; j++) {                 //3
            if (A[j].key < A[lowIndex].key){        //4
                lowIndex = j;                       //5
            }                                       //6
        }                                           //7
        swap(&A[i], &A[lowIndex]);                  //8
    }
}
```

www.ctu.edu.vn 24

24



Complexity

- Time
  - Find element with lowest key from [0 .. n-1]: n-1 comparisons
  - Find element with lowest key from [1 .. n-1]: n-2 comparisons

  - Totally (n-1) + (n-2) + … + 1 comparisons and n swaps
  - $T(n) = O(n^2)$
- Space
  - $S(n) = O(1)$

www.ctu.edu.vn 25

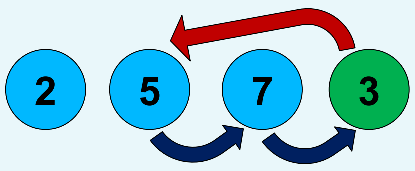
25

Agenda

- Introduction
- Some O(n^2) algorithms
  - Bubble sort
  - Selection sort
  - Insertion sort
- Shell sort
- Summary

Insertion sort

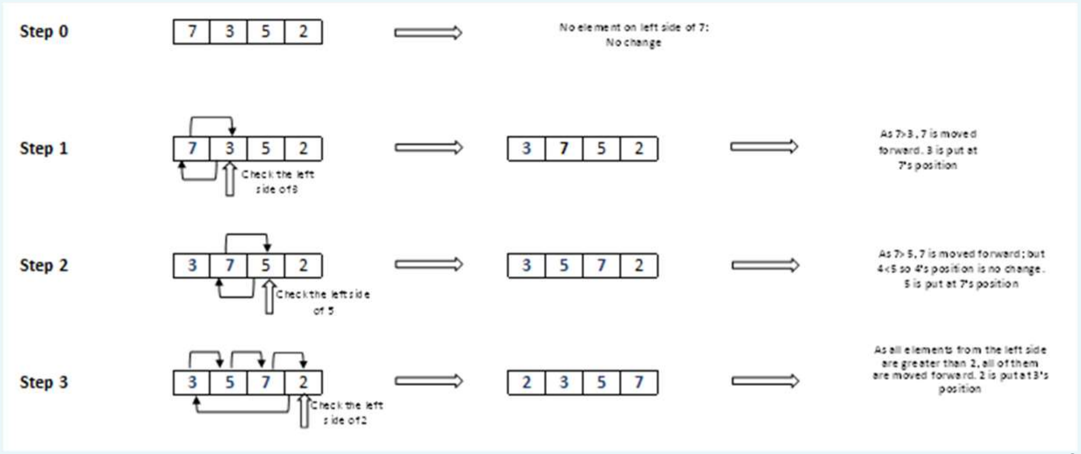
- The sorted array is grown in each iteration
- At iteration i, elements from 0 to i-1 are sorted
  - Each element in left side may be shifted forward
  - The element i is put to the correct position



CANTHO UNIVERSITY

Example

- A= [7, 3, 5, 2]


Step 0
7 3 5 2
No element on left side of 7:
No change
Step 1
7 3 5 2
Check the left side of 3
3 7 5 2
As 7>3, 7 is moved forward. 3 is put at 7's position
Step 2
3 7 5 2
Check the left side of 5
3 5 7 2
As 7> 5, 7 is moved forward; but 4<5 so 4's position is no change. 5 is put at 7's position
Step 3
3 5 7 2
Check the left side of 2
2 3 5 7
As all elements from the left side are greater than 2, all of them are moved forward. 2 is put at 3's position

www.ctu.edu.vn 28

CANTHO UNIVERSITY

Algorithm details

```
void insertionSort(ElementType A[], int n) {
    int i;
    for (i=0; i<n; i++){
        ElementType temp = A[i];
        int j = i;
        while (j>0 && temp.key < A[j-1].key){
            A[j] = A[j-1];
            j = j-1;
        }
        A[j] = temp;
    }
}
```

www.ctu.edu.vn 29

Complexity

- Time:
  - In the worst case, each element is compared with all other elements.
    - There are at most n^2 comparisons (at most n^2 shifts)
  - $T(n) = O(n^2)$
- Space
  - $S(n) = O(1)$

www.ctu.edu.vn 30

30

Agenda

- Introduction
- Some $O(n^2)$ algorithms
  - Bubble sort
  - Selection sort
  - Insertion sort
- Shell sort
- Summary

www.ctu.edu.vn 31

31

Shell sort

- Shell sort (Donald Shell) - Diminishing increment sort
  - Break the input into sublists of non contiguous items (k interval apart)
  - Sort those by insertion sort
  - Decrease k until ending up a normal insertion sort
- Recall: insertion sort costs linearly when the list is sorted or nearly sorted
- Intuition: tries to quickly make the list mostly sorted → insertion sort finishes the job

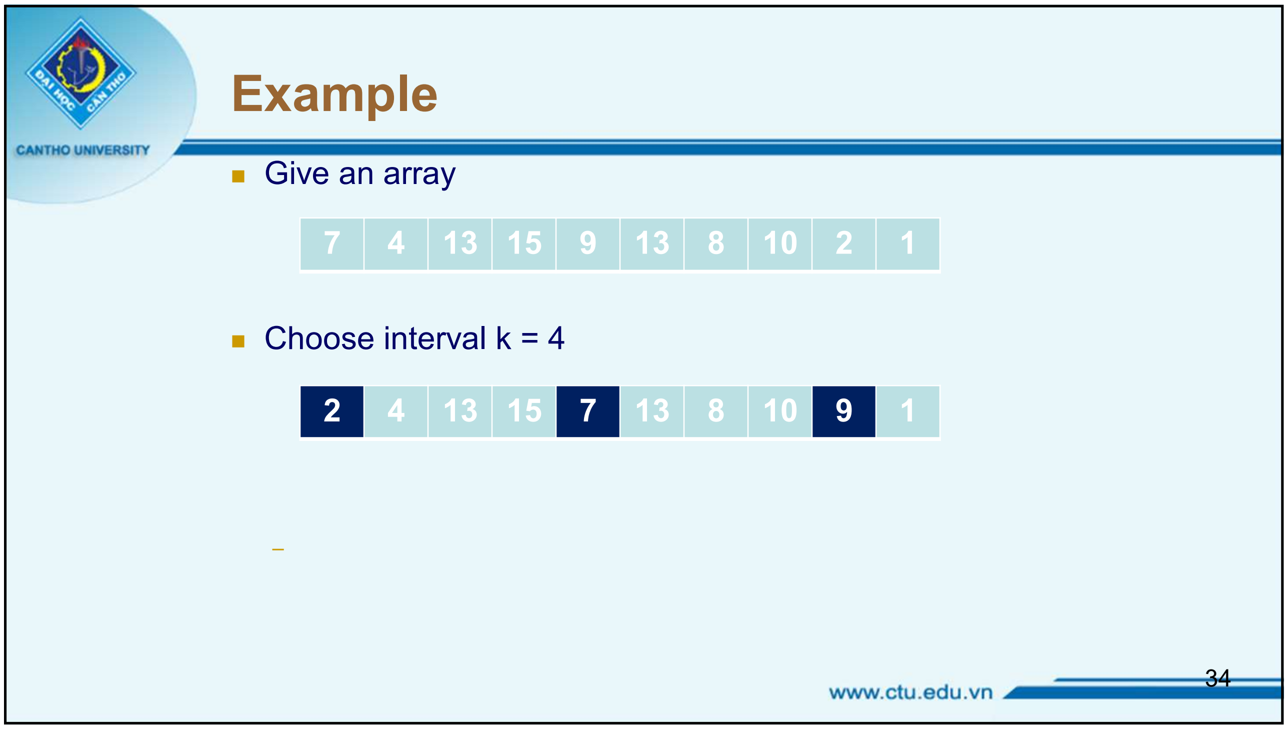
Example

- Give an array

7	4	13	15	9	13	8	10	2	1

- Choose interval k = 4

7	4	13	15	**9**	13	8	10	**2**	1



Example

- Give an array

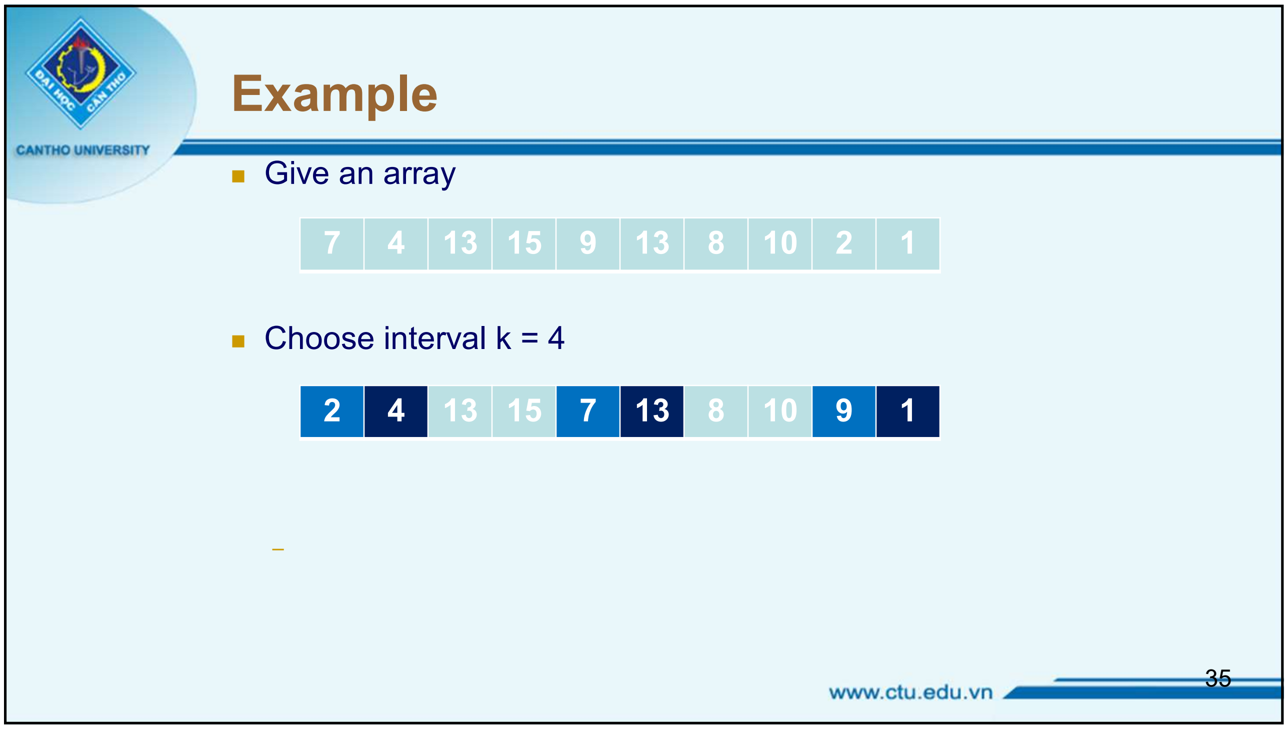
7	4	13	15	9	13	8	10	2	1

- Choose interval k = 4

2	4	13	15	**7**	13	8	10	**9**	1

www.ctu.edu.vn 34

34



Example

- Give an array

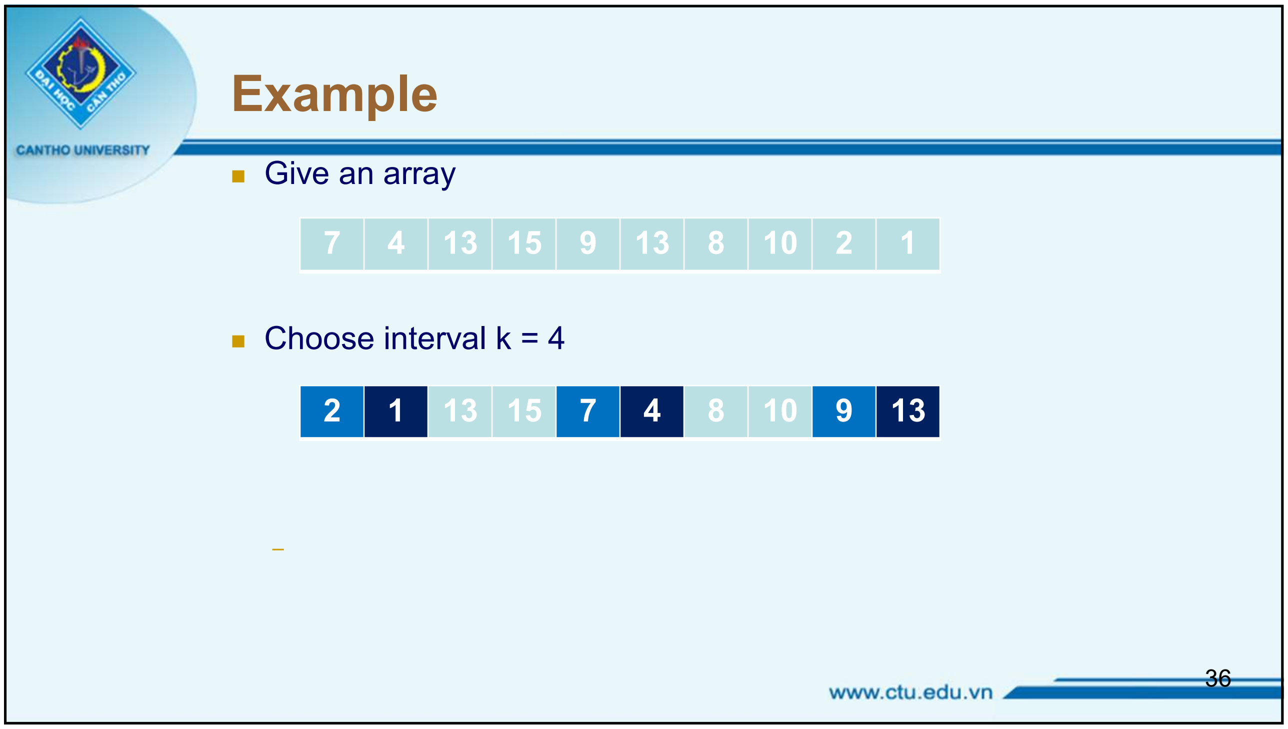
7	4	13	15	9	13	8	10	2	1

- Choose interval k = 4

2	**4**	13	15	**7**	**13**	8	10	**9**	**1**

www.ctu.edu.vn 35

35



Example

- Give an array

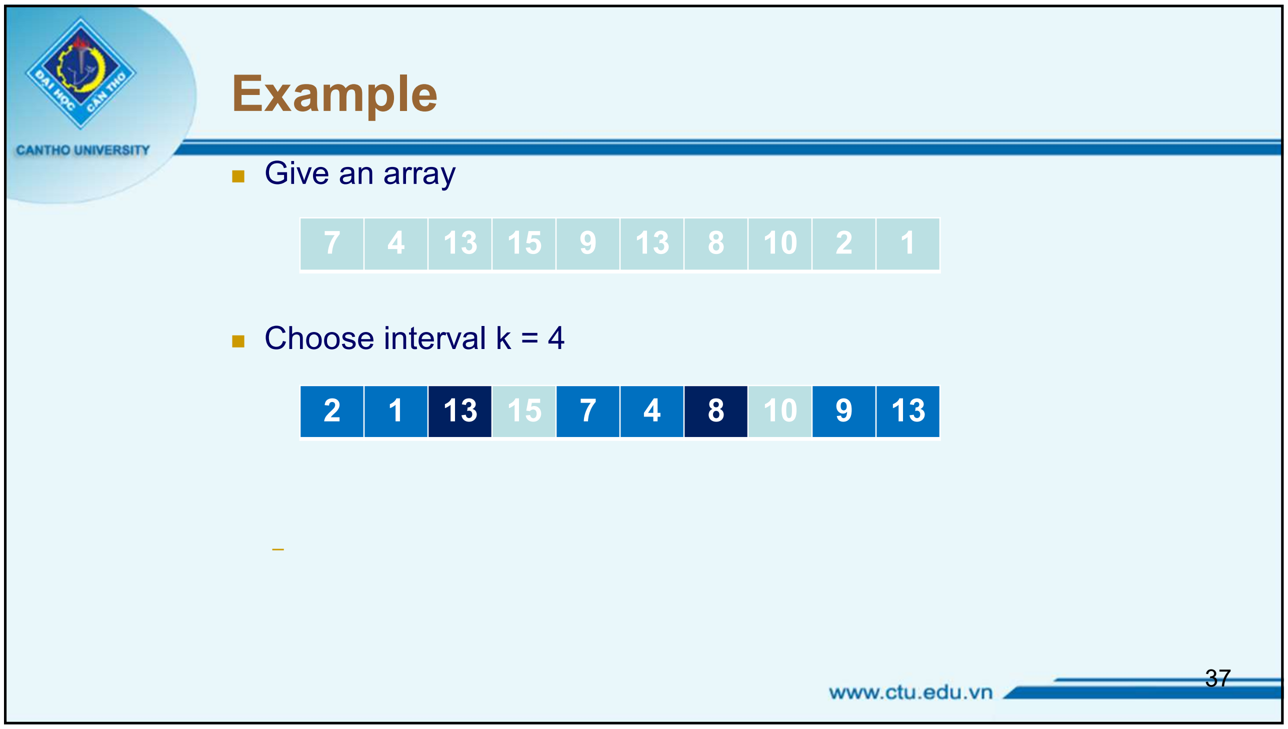
7	4	13	15	9	13	8	10	2	1

- Choose interval k = 4

2	1	13	15	7	4	8	10	9	13

www.ctu.edu.vn 36

36



Example

- Give an array

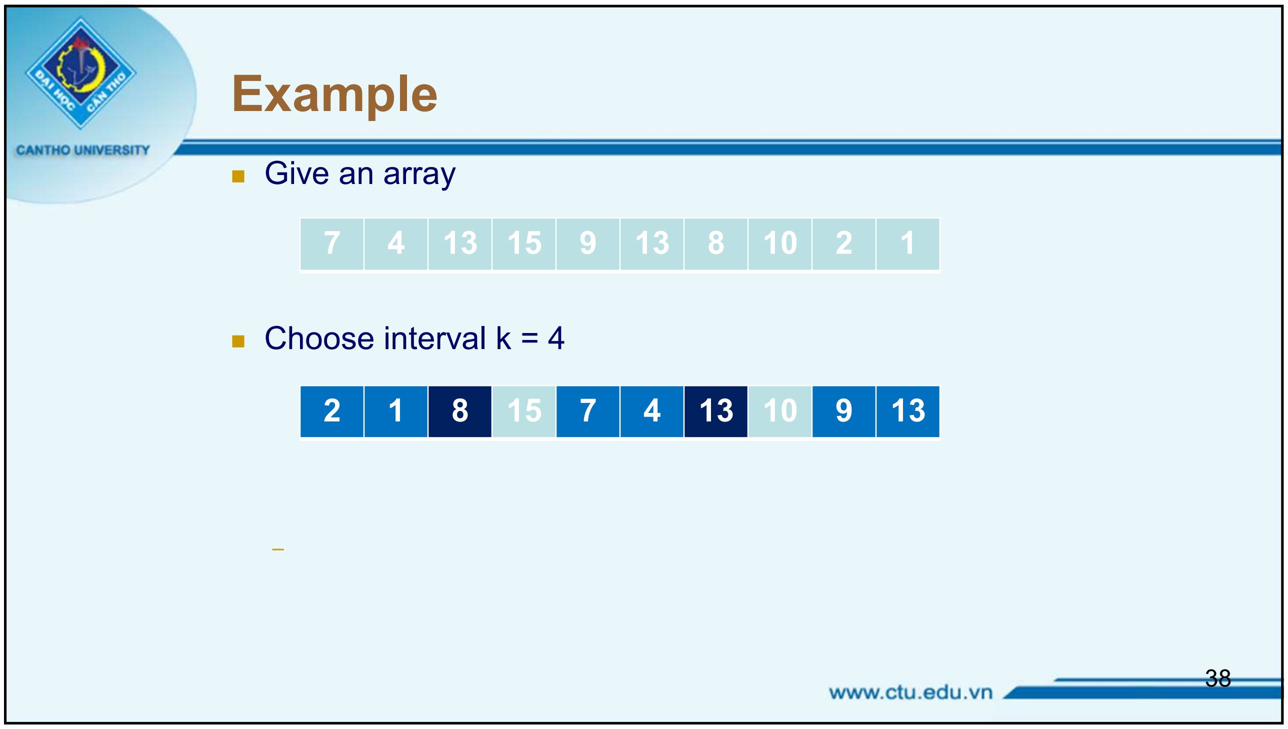
7	4	13	15	9	13	8	10	2	1

- Choose interval k = 4

2	1	13	15	7	4	8	10	9	13

www.ctu.edu.vn 37

37



Example

- Give an array

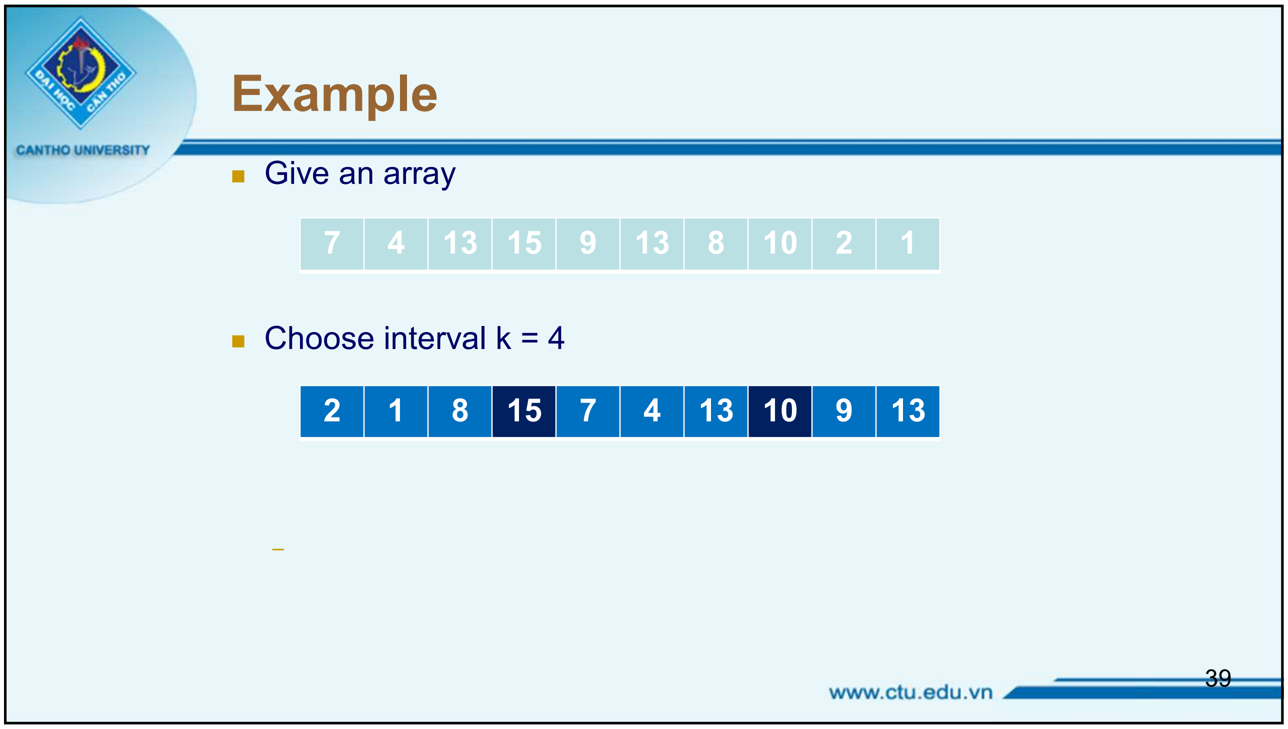
7	4	13	15	9	13	8	10	2	1

- Choose interval k = 4

2	1	8	15	7	4	13	10	9	13

www.ctu.edu.vn 38

38



Example

- Give an array

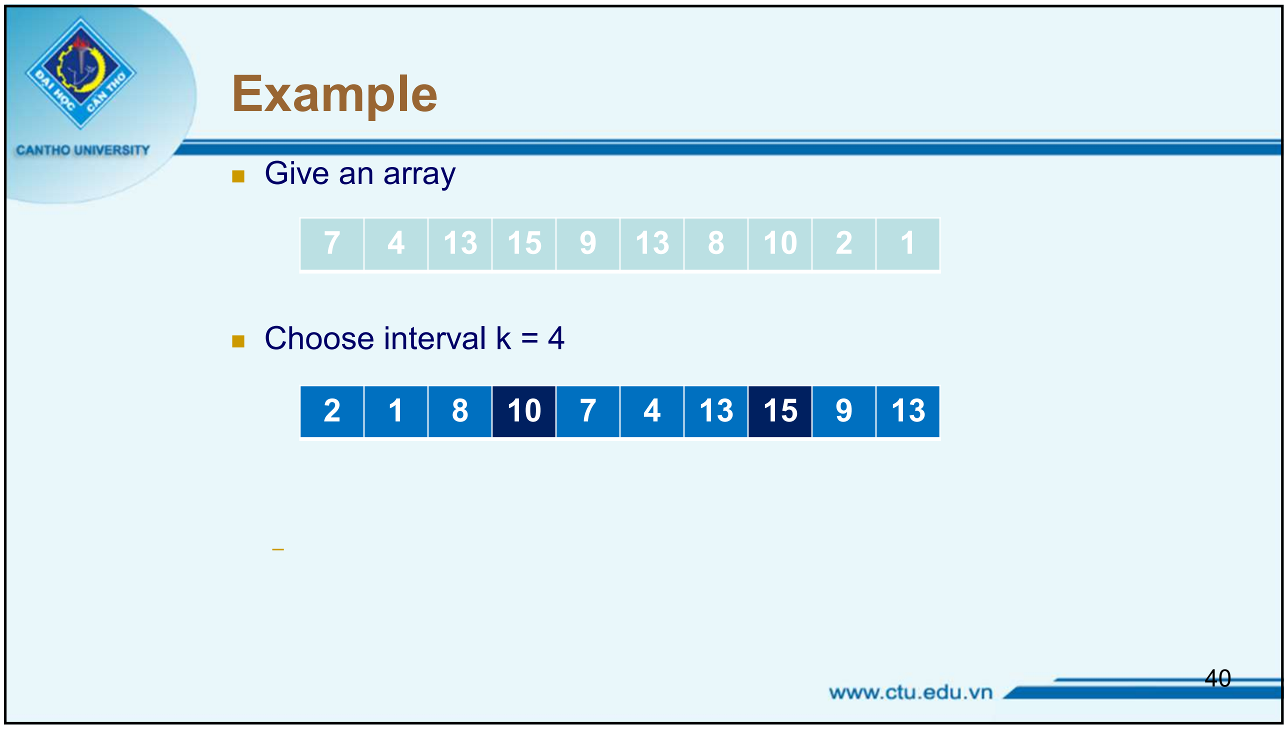
7	4	13	15	9	13	8	10	2	1

- Choose interval k = 4

2	1	8	15	7	4	13	10	9	13

www.ctu.edu.vn 39

39



Example

- Give an array

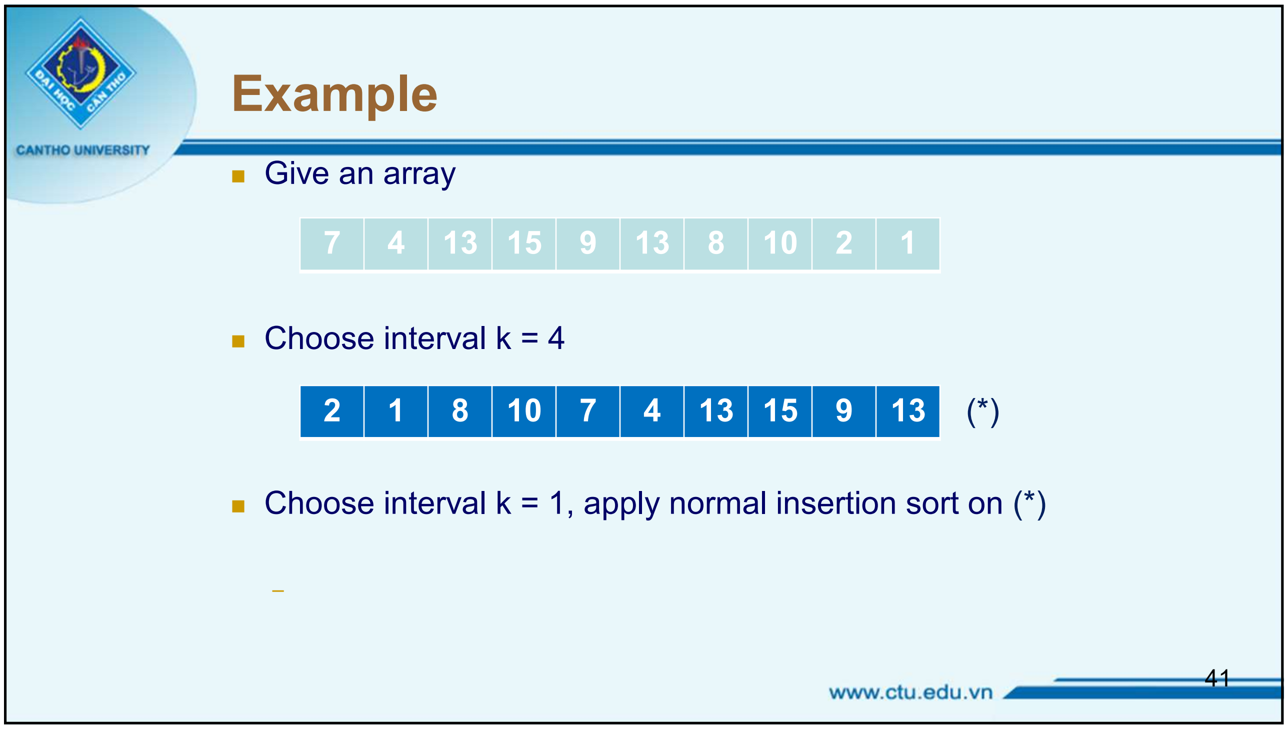
7	4	13	15	9	13	8	10	2	1

- Choose interval k = 4

2	1	8	10	7	4	13	15	9	13

www.ctu.edu.vn 40

40



Example

- Give an array

7	4	13	15	9	13	8	10	2	1

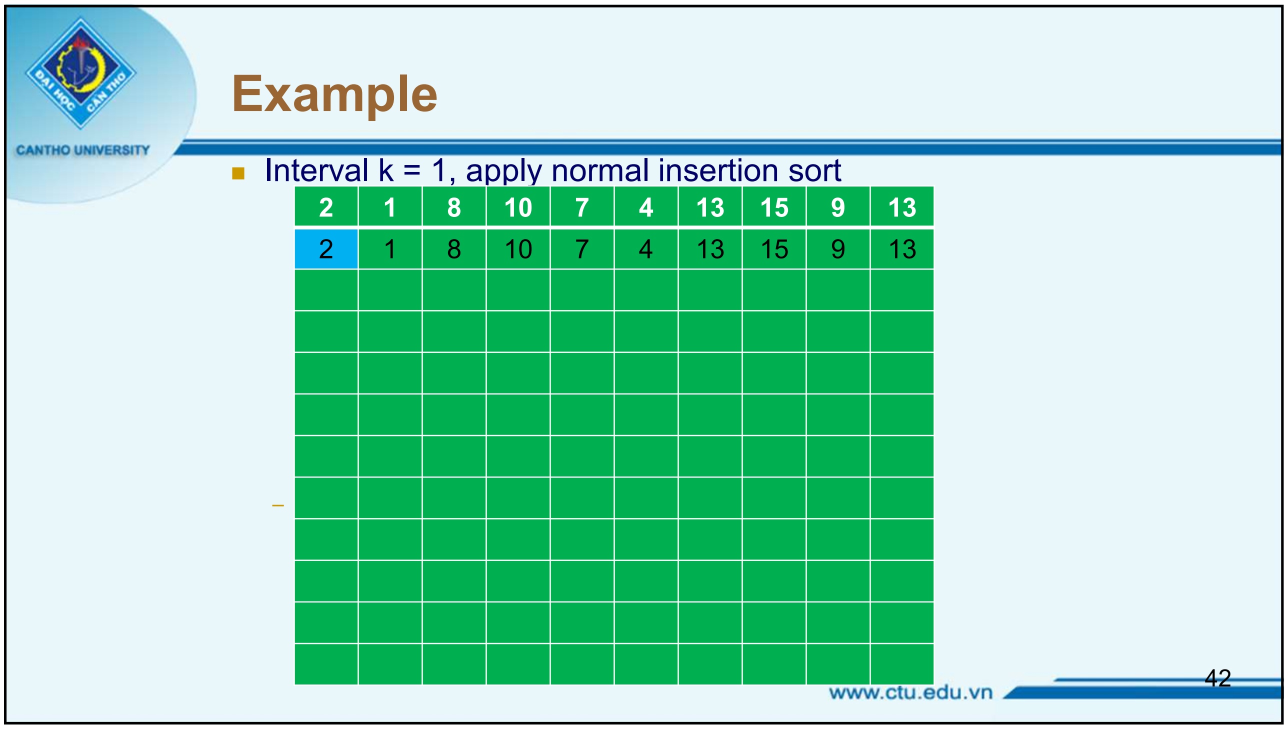
- Choose interval k = 4

2	1	8	10	7	4	13	15	9	13	(*)

- Choose interval k = 1, apply normal insertion sort on (*)

www.ctu.edu.vn 41

41



CANTHO UNIVERSITY

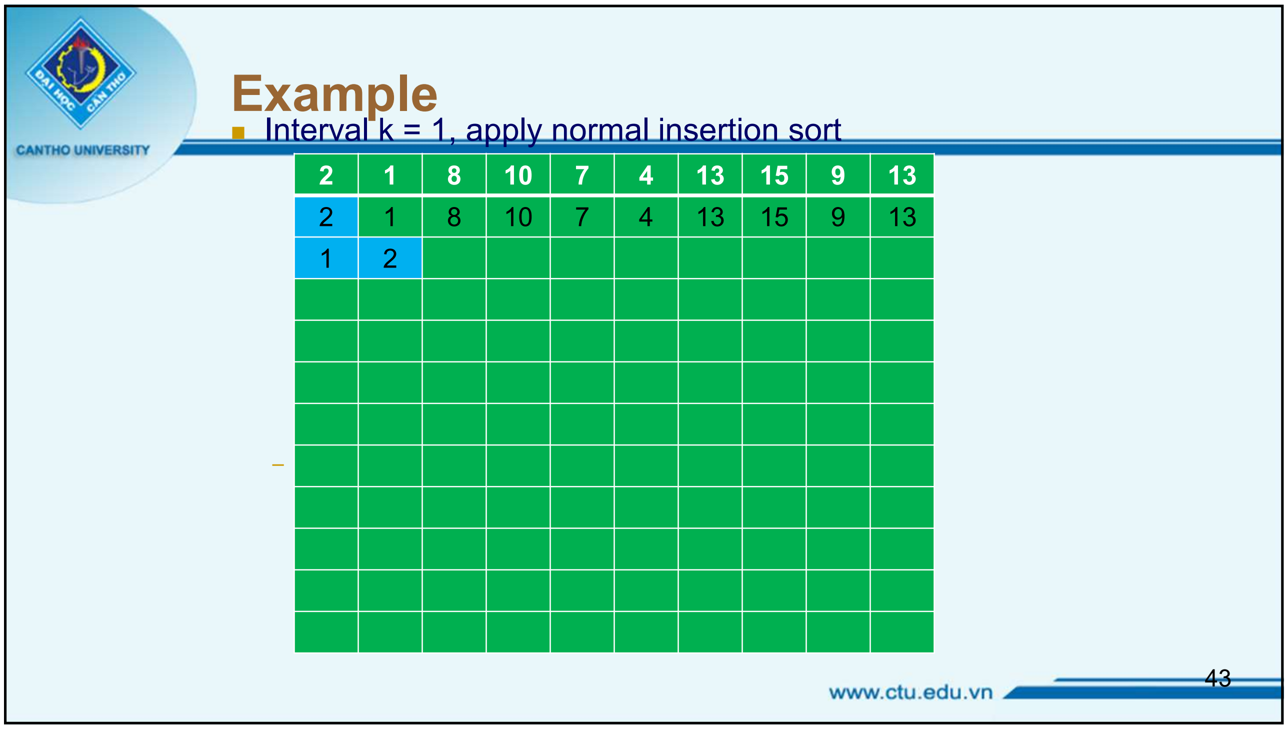
Example

- Interval k = 1, apply normal insertion sort

2	1	8	10	7	4	13	15	9	13
2	1	8	10	7	4	13	15	9	13

www.ctu.edu.vn 42

42



CANTHO UNIVERSITY

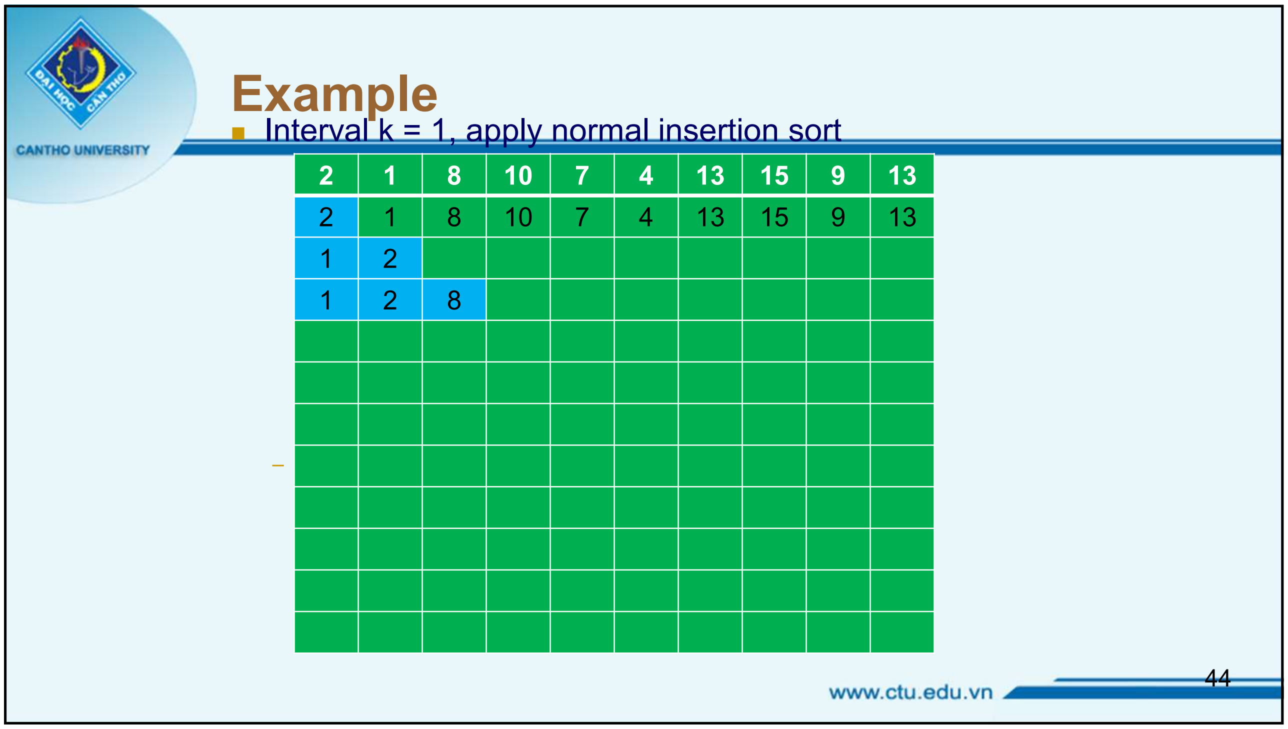
Example

- Interval k = 1, apply normal insertion sort

2	1	8	10	7	4	13	15	9	13
2	1	8	10	7	4	13	15	9	13
1	2								

www.ctu.edu.vn 43

43



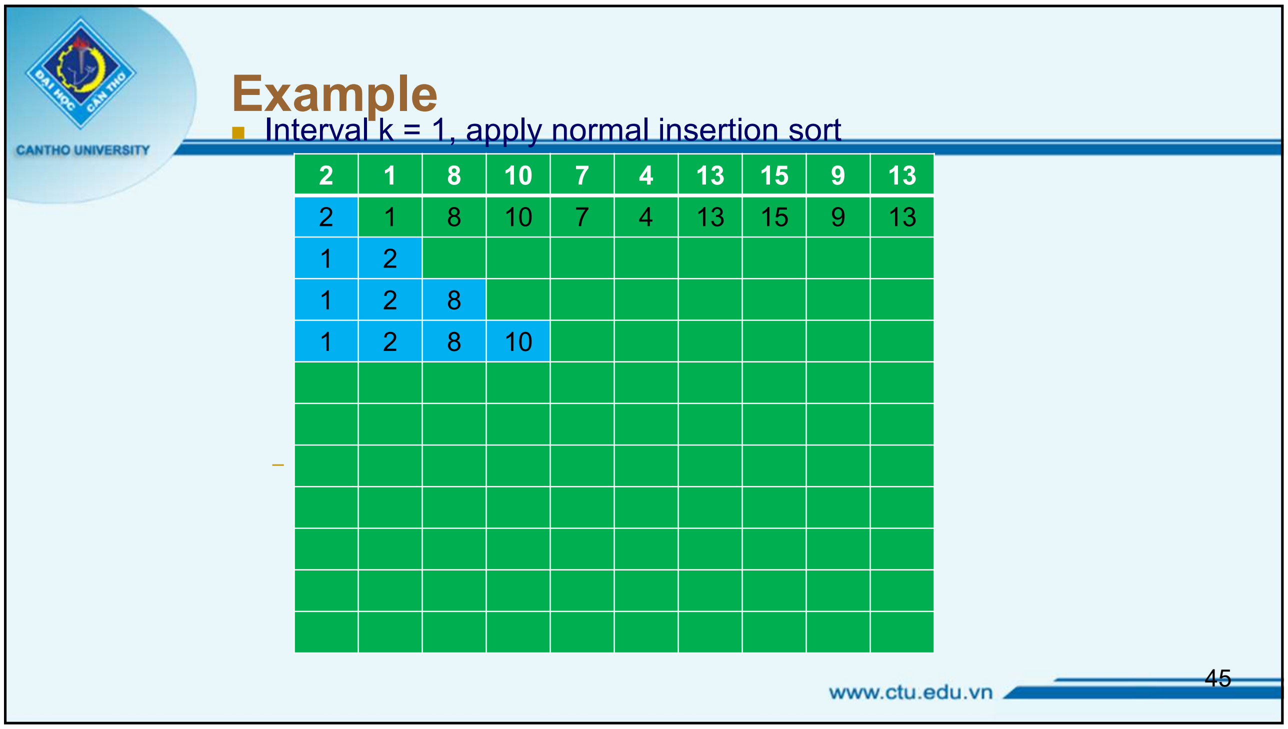
Example

- Interval k = 1, apply normal insertion sort

2	1	8	10	7	4	13	15	9	13
2	1	8	10	7	4	13	15	9	13
1	2								
1	2	8							

www.ctu.edu.vn 44

44



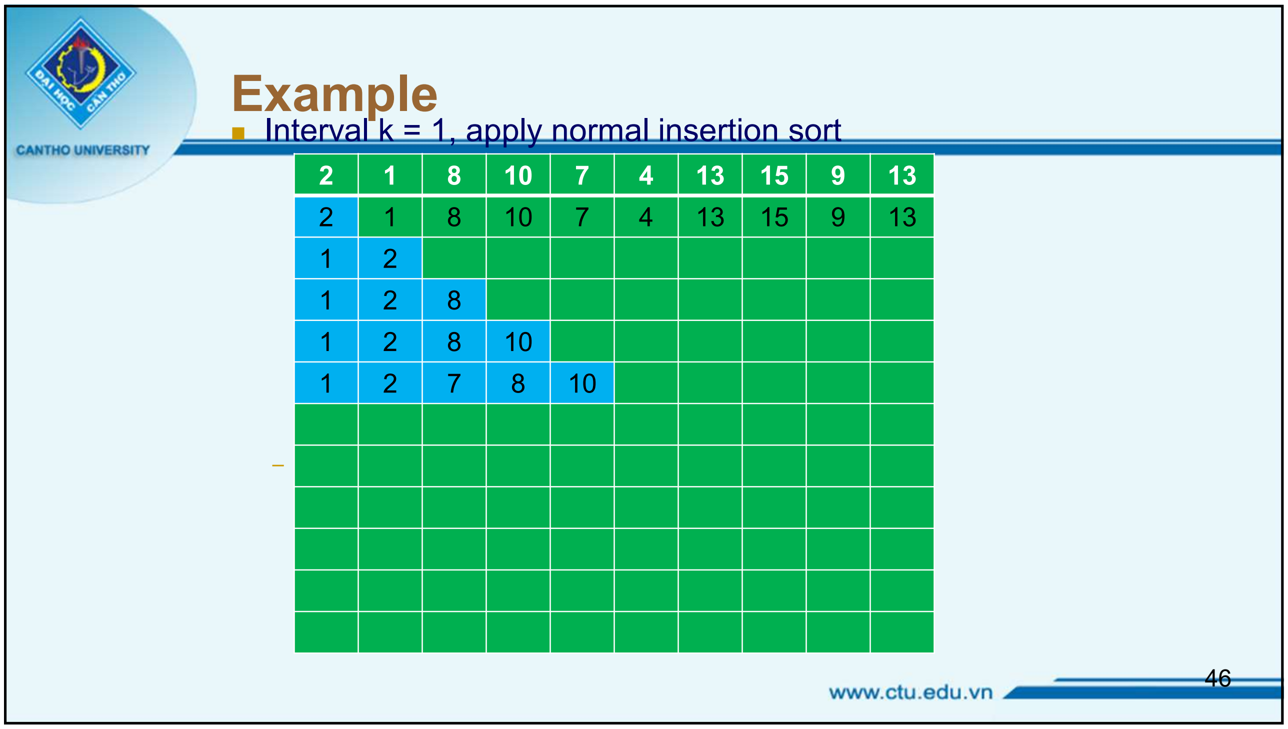
Example

- Interval k = 1, apply normal insertion sort

2	1	8	10	7	4	13	15	9	13
2	1	8	10	7	4	13	15	9	13
1	2								
1	2	8							
1	2	8	10						

www.ctu.edu.vn 45

45

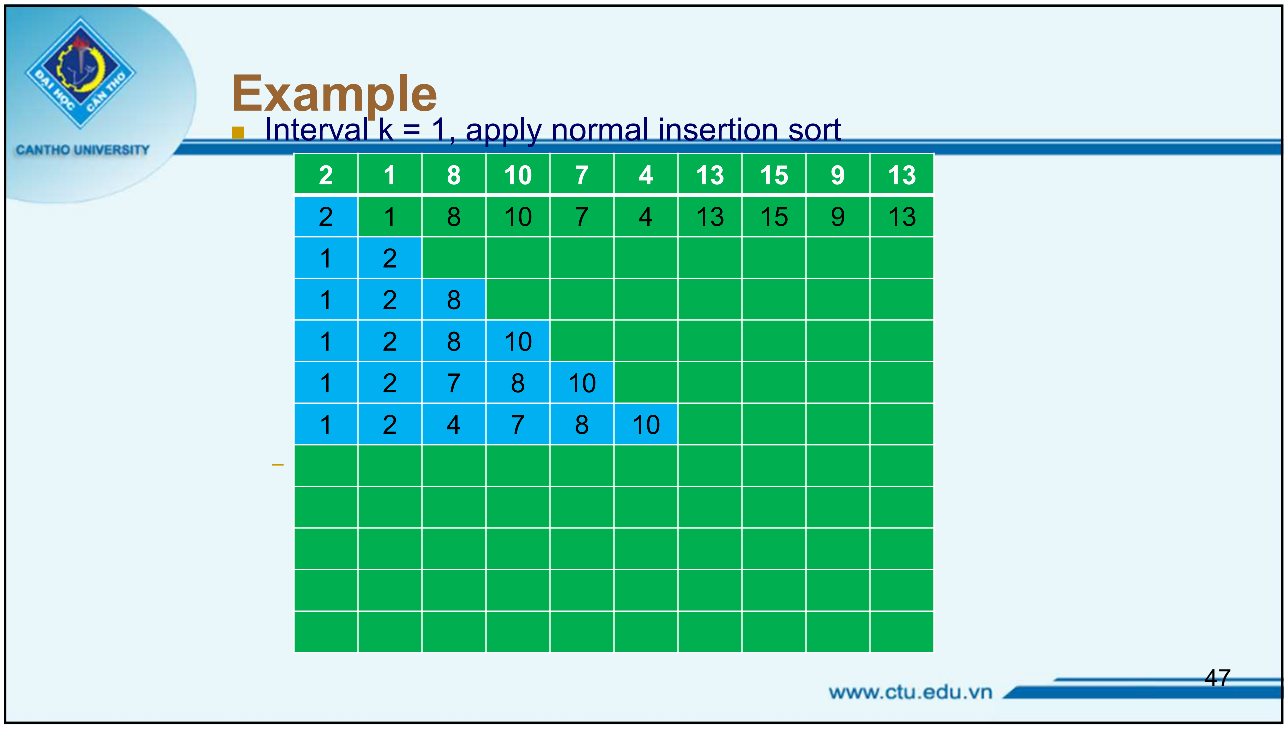


Example

- Interval k = 1, apply normal insertion sort

2	1	8	10	7	4	13	15	9	13
2	1	8	10	7	4	13	15	9	13
1	2								
1	2	8							
1	2	8	10						
1	2	7	8	10					

www.ctu.edu.vn 46

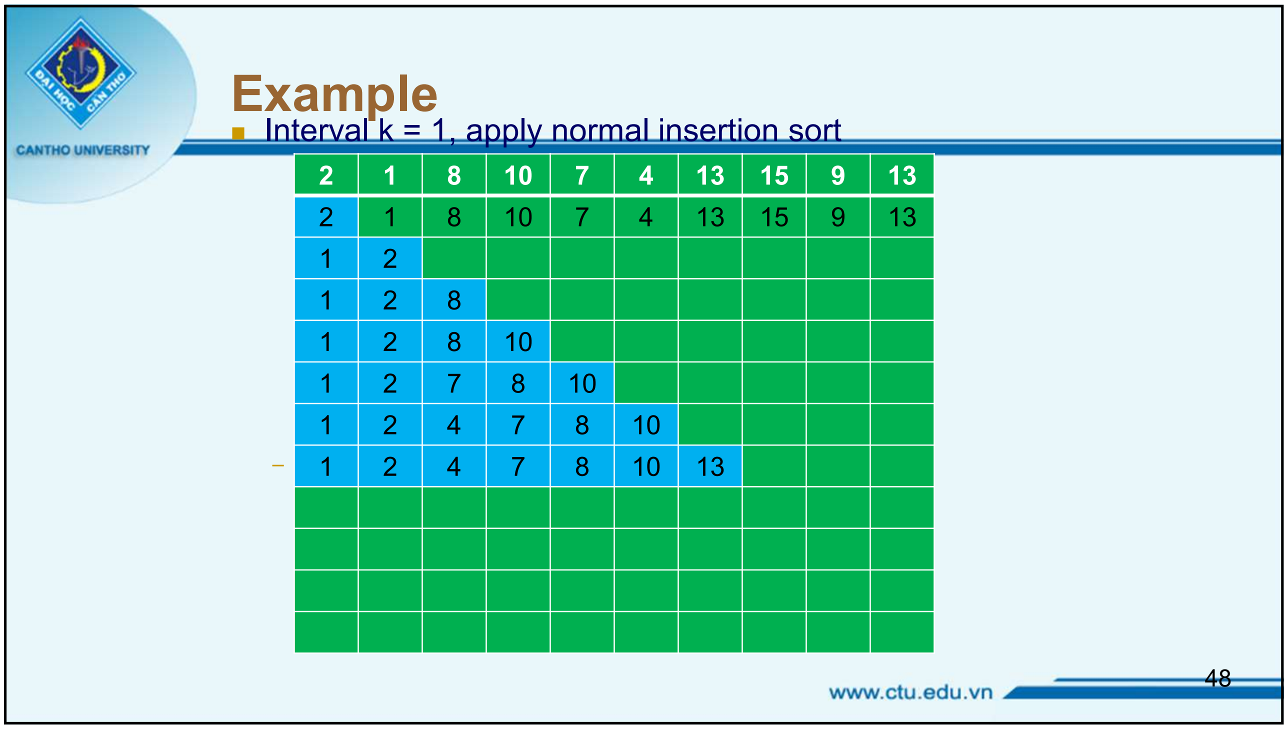


Example

- Interval k = 1, apply normal insertion sort

2	1	8	10	7	4	13	15	9	13
2	1	8	10	7	4	13	15	9	13
1	2								
1	2	8							
1	2	8	10						
1	2	7	8	10					
1	2	4	7	8	10				

www.ctu.edu.vn 47



Example

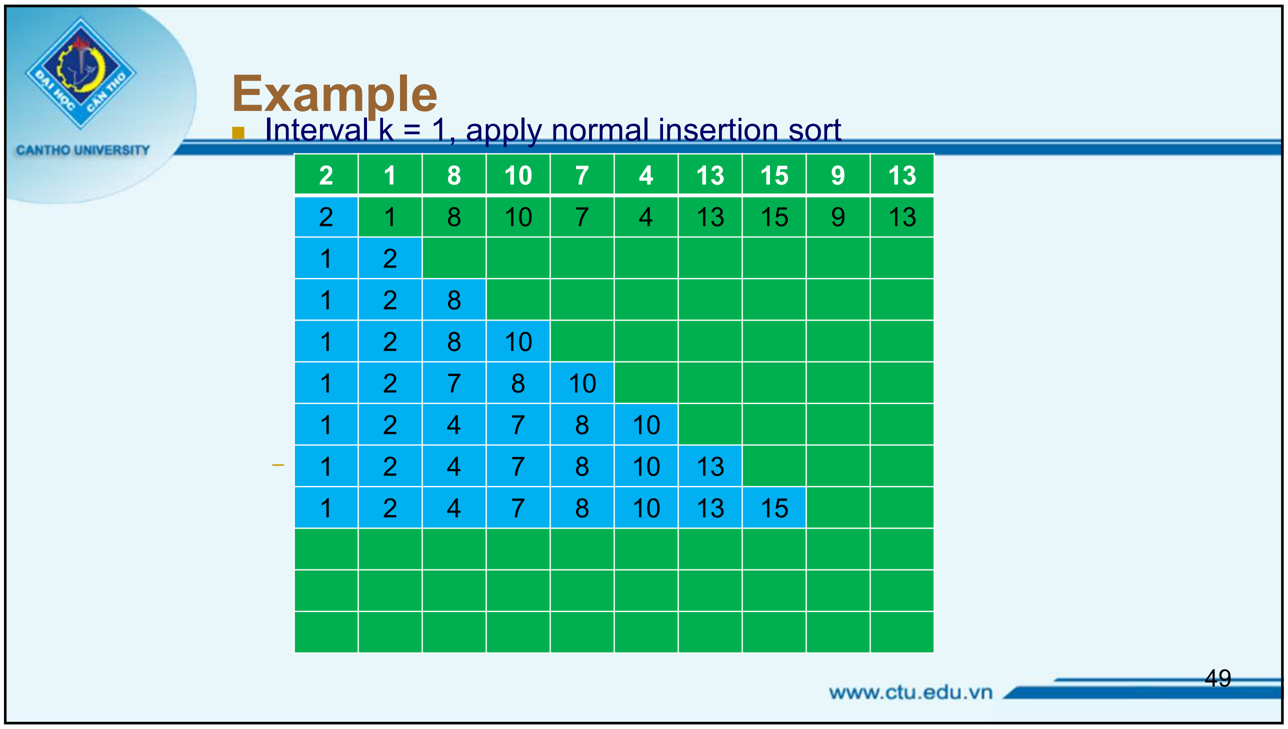
- Interval k = 1, apply normal insertion sort

2	1	8	10	7	4	13	15	9	13
2	1	8	10	7	4	13	15	9	13
1	2								
1	2	8							
1	2	8	10						
1	2	7	8	10					
1	2	4	7	8	10				
1	2	4	7	8	10	13			

www.ctu.edu.vn

48

48



Example

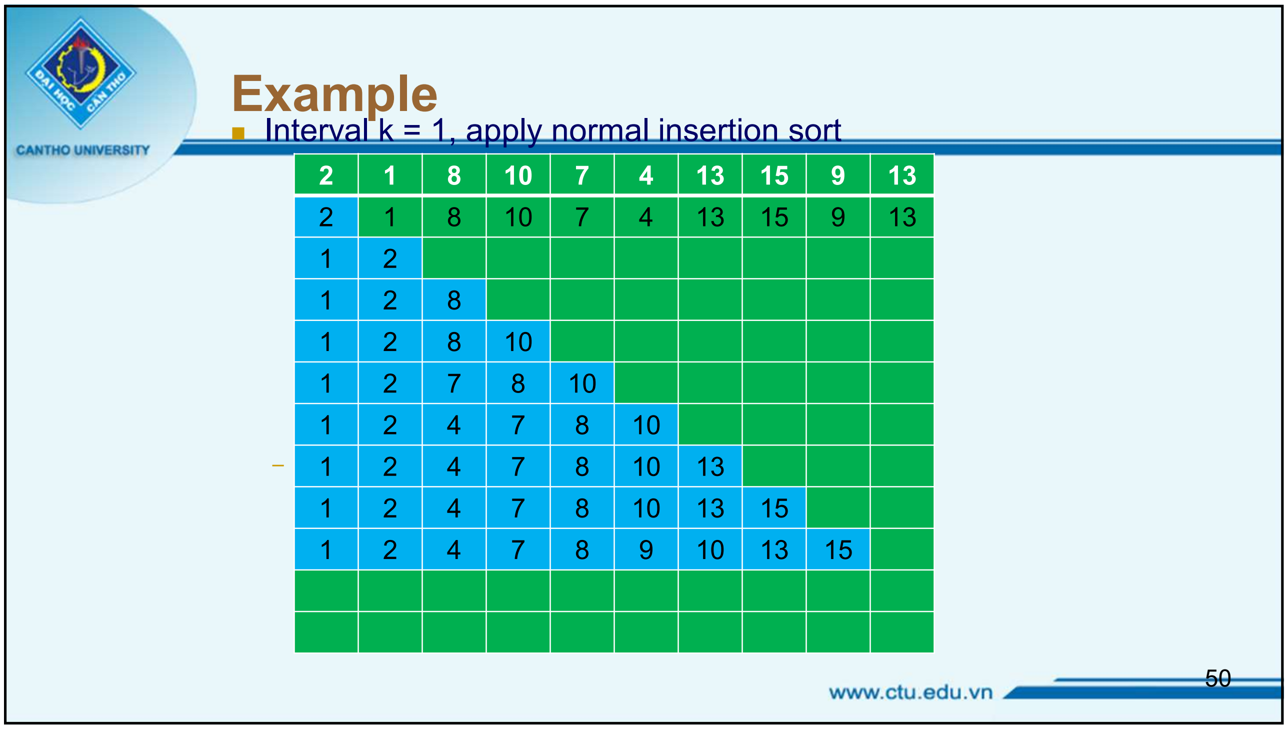
- Interval k = 1, apply normal insertion sort

2	1	8	10	7	4	13	15	9	13
2	1	8	10	7	4	13	15	9	13
1	2								
1	2	8							
1	2	8	10						
1	2	7	8	10					
1	2	4	7	8	10				
1	2	4	7	8	10	13			
1	2	4	7	8	10	13	15		

www.ctu.edu.vn

49

49



CANTHO UNIVERSITY

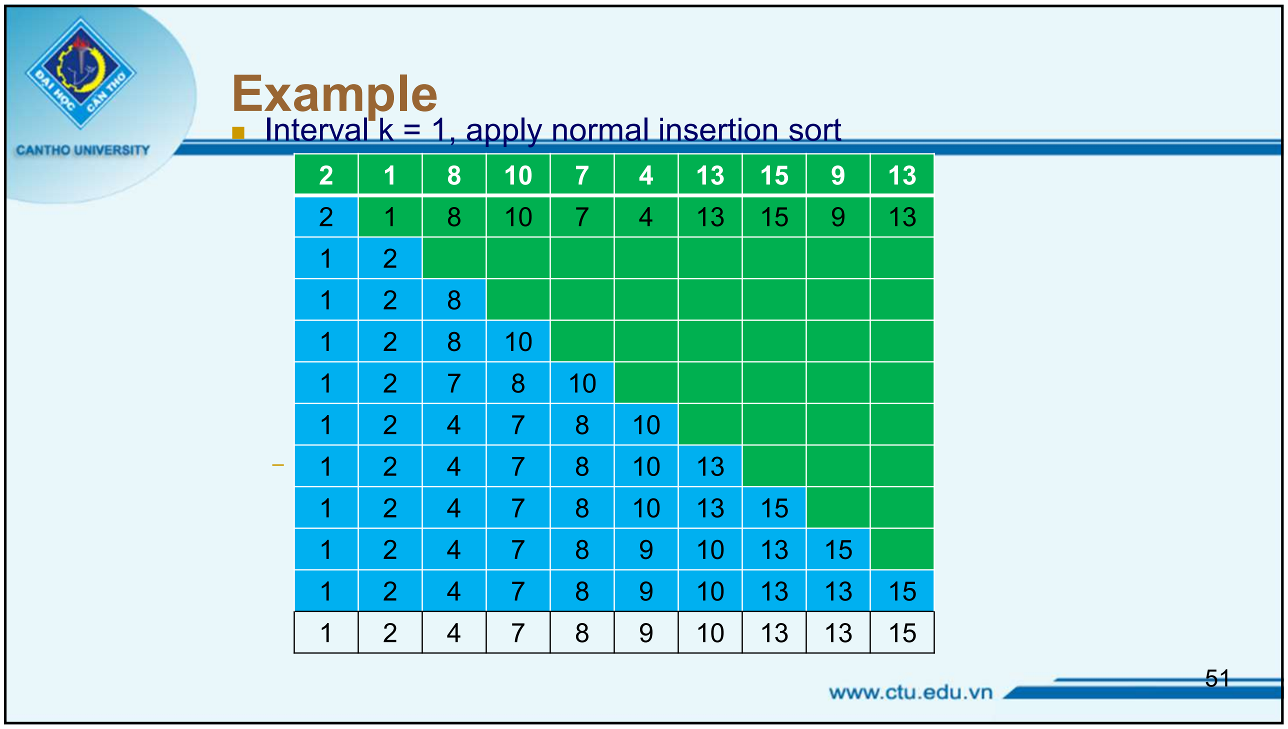
Example

- Interval k = 1, apply normal insertion sort

2	**1**	**8**	**10**	**7**	**4**	**13**	**15**	**9**	**13**
2	1	8	10	7	4	13	15	9	13
1	2								
1	2	8							
1	2	8	10						
1	2	7	8	10					
1	2	4	7	8	10				
1	2	4	7	8	10	13			
1	2	4	7	8	10	13	15		
1	2	4	7	8	9	10	13	15	

www.ctu.edu.vn 50

50



CANTHO UNIVERSITY

Example

- Interval k = 1, apply normal insertion sort

2	**1**	**8**	**10**	**7**	**4**	**13**	**15**	**9**	**13**
2	1	8	10	7	4	13	15	9	13
1	2								
1	2	8							
1	2	8	10						
1	2	7	8	10					
1	2	4	7	8	10				
1	2	4	7	8	10	13			
1	2	4	7	8	10	13	15		
1	2	4	7	8	9	10	13	15	
1	2	4	7	8	9	10	13	13	15
1	2	4	7	8	9	10	13	13	15

www.ctu.edu.vn 51

51

Time complexity

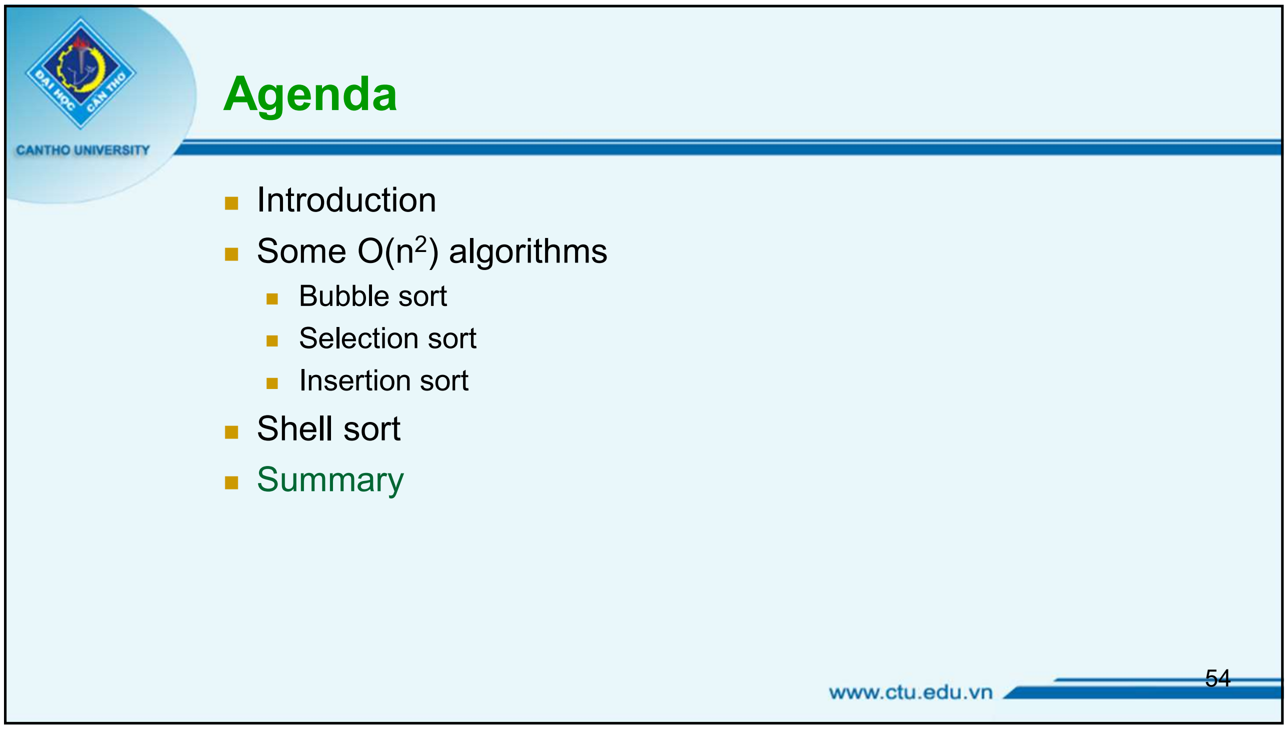
- The interval between elements is reduced based on the sequence used
- The time complexity depends on the type of sequence
- Some sequences

Sequence	Formula	Concrete interval	Worst case complexity
Shell	floor($\frac{n}{2^k}$)	1, 2, …, [n/4], [n/2]	O(n^2)
Knuth	$\frac{3^k-1}{2}$, not greater than [n/3]	1, 4, 13, 40, 121, …	O($n^{3/2}$)
Sedgewick	$4^k + 3.2^{k-1} + 1$, prefixed with 1	1, 8, 23, 77, …	O($n^{4/3}$)

Algorithm details

```
ALGORITHM getInterval(n): #Knuth sequence
    k ← 1
    while (k < n/3):
        k ← k*3 + 1
    return k

ALGORITHM shellSort(A, n):
    k ← getInterval(n)
    while (k >= 1):
        for i ← k to n - 1 step 1:
            temp ← A[i]
            j ← i
            while (j >= k and temp.key < A[j-k].key):
                A[j] ← A[j-k]
                j ← j - k
            A[j] ← temp
        k ← (k-1)/3
```

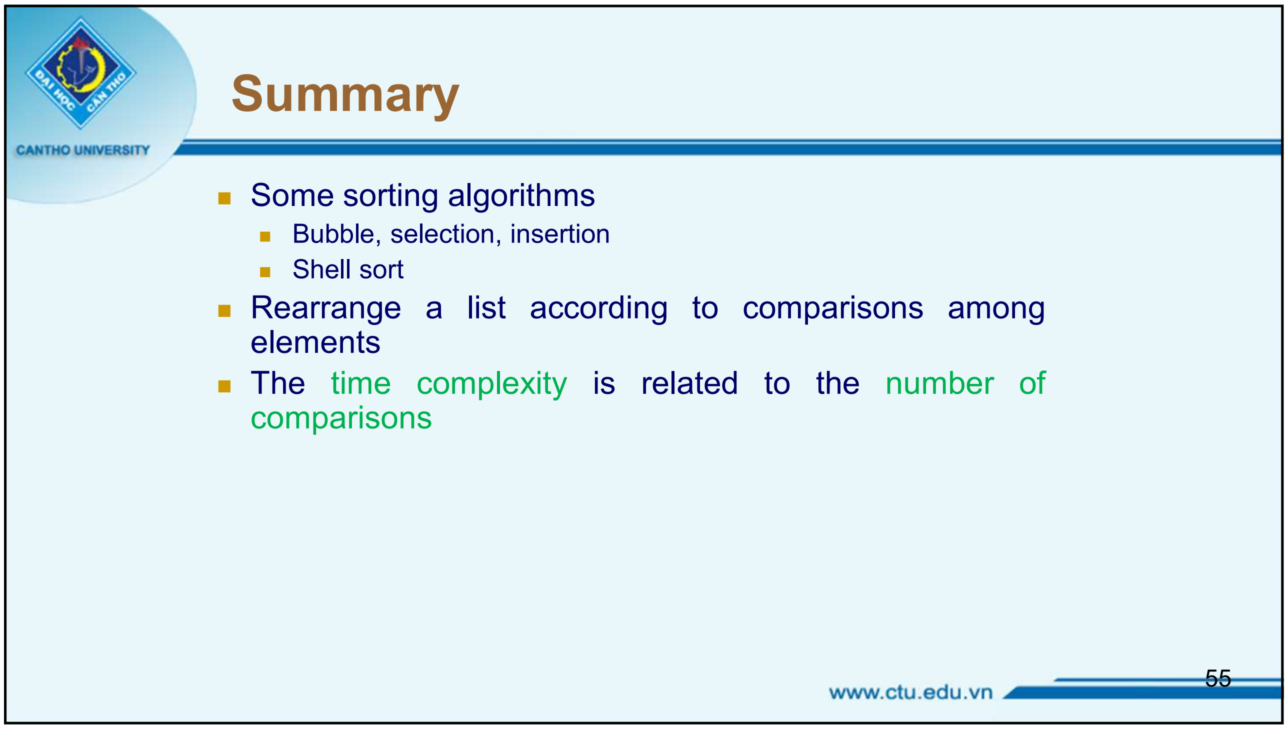


Agenda

- Introduction
- Some $O(n^2)$ algorithms
  - Bubble sort
  - Selection sort
  - Insertion sort
- Shell sort
- Summary

www.ctu.edu.vn 54

54



Summary

- Some sorting algorithms
  - Bubble, selection, insertion
  - Shell sort
- Rearrange a list according to comparisons among elements
- The time complexity is related to the number of comparisons

www.ctu.edu.vn 55

55

CANTHO UNIVERSITY

Q&A

www.ctu.edu.vn

56