Bubble, Insertion & Tree Sorts

Kuan-Yu Chen (陳冠宇)

Sorting

- Sorting means arranging the elements of an array so that they are placed in some relevant order which may be either **ascending** or **descending**

- A sorting algorithm is defined as an algorithm that puts the elements of a list in a certain order, which can be either **numerical** order, **lexicographical** order, or **any user-defined** order
  - Bubble, Insertion, Selection, Tree
  - Merge, Quick, Radix, Heap, Shell

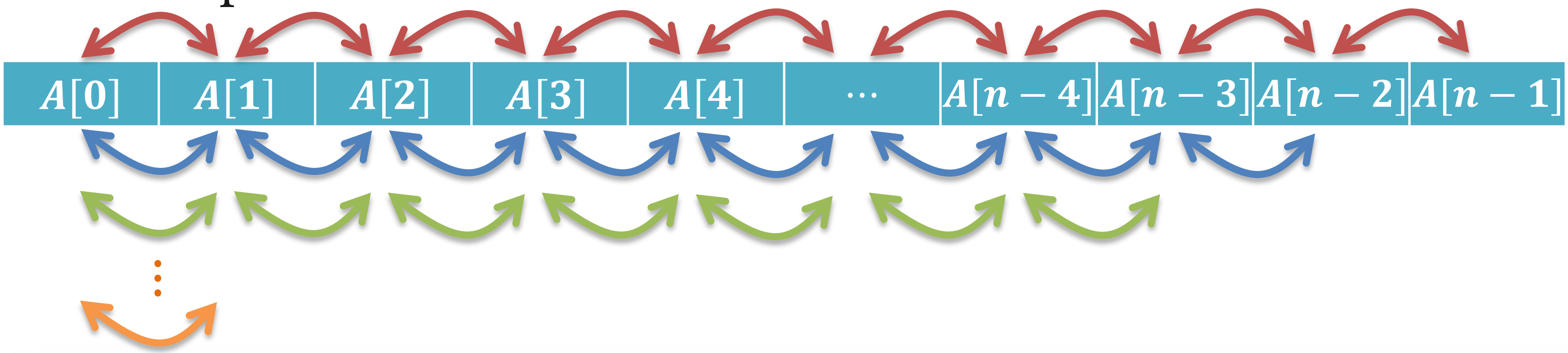
Bubble Sort.

- Bubble sort is a very simple method that sorts the array elements by repeatedly moving the largest element to the highest index position of the array segment
  - Consecutive adjacent pairs of elements in the array are compared with each other
  - If the element at the lower index is greater than the element at the higher index, the two elements are interchanged

- This procedure of sorting is called bubble sorting because elements “bubble” to the top of the list
  - Bubble sort is referred to as “Sorting by exchange” in 1956
  - It is referred to as “Exchange Sorting” in 1959
  - The term “Bubble Sort ” was first used by Iverson in 1962

Bubble Sort..

- The procedure!



The basic methodology of the working of bubble sort is given as follows:

(a) In Pass 1, `A[0]` and `A[1]` are compared, then `A[1]` is compared with `A[2]`, `A[2]` is compared with `A[3]`, and so on. Finally, `A[N-2]` is compared with `A[N-1]`. Pass 1 involves `n-1` comparisons and places the biggest element at the highest index of the array.

(b) In Pass 2, `A[0]` and `A[1]` are compared, then `A[1]` is compared with `A[2]`, `A[2]` is compared with `A[3]`, and so on. Finally, `A[N-3]` is compared with `A[N-2]`. Pass 2 involves `n-2` comparisons and places the second biggest element at the second highest index of the array.

(c) In Pass 3, `A[0]` and `A[1]` are compared, then `A[1]` is compared with `A[2]`, `A[2]` is compared with `A[3]`, and so on. Finally, `A[N-4]` is compared with `A[N-3]`. Pass 3 involves `n-3` comparisons and places the third biggest element at the third highest index of the array.

(d) In Pass `n-1`, `A[0]` and `A[1]` are compared so that `A[0]<A[1]`. After this step, all the elements of the array are arranged in ascending order.

Example.

- Please sort a given data array by using bubble sort

```
A[] = {30, 52, 29, 87, 63, 27, 19, 54}
```

  - Pass 1:

(a) Compare 30 and 52. Since 30 < 52, no swapping is done.
(b) Compare 52 and 29. Since 52 > 29, swapping is done.
30, **29, 52**, 87, 63, 27, 19, 54
(c) Compare 52 and 87. Since 52 < 87, no swapping is done.
(d) Compare 87 and 63. Since 87 > 63, swapping is done.
30, 29, 52, **63, 87**, 27, 19, 54
(e) Compare 87 and 27. Since 87 > 27, swapping is done.
30, 29, 52, 63, **27, 87**, 19, 54
(f) Compare 87 and 19. Since 87 > 19, swapping is done.
30, 29, 52, 63, 27, **19, 87**, 54
(g) Compare 87 and 54. Since 87 > 54, swapping is done.
30, 29, 52, 63, 27, 19, **54, 87**

Example..

- Please sort a given data array by using bubble sort

```
A[] = {30, 52, 29, 87, 63, 27, 19, 54}
```

  - Pass 1:

    30, 29, 52, 63, 27, 19, **54, 87**

  - Pass 2:

(a) Compare 30 and 29. Since 30 > 29, swapping is done.
    29, 30, 52, 63, 27, 19, 54, 87
(b) Compare 30 and 52. Since 30 < 52, no swapping is done.
(c) Compare 52 and 63. Since 52 < 63, no swapping is done.
(d) Compare 63 and 27. Since 63 > 27, swapping is done.
    29, 30, 52, **27, 63**, 19, 54, 87
(e) Compare 63 and 19. Since 63 > 19, swapping is done.
    29, 30, 52, 27, **19, 63**, 54, 87
(f) Compare 63 and 54. Since 63 > 54, swapping is done.
    29, 30, 52, 27, 19, **54, 63**, 87

Example...

- Please sort a given data array by using bubble sort

```
A[] = {30, 52, 29, 87, 63, 27, 19, 54}
```

  - Pass 2:

29, 30, 52, 27, 19, **54**, **63**, 87

  - Pass 3:

```
(a) Compare 29 and 30. Since 29 < 30, no swapping is done.
(b) Compare 30 and 52. Since 30 < 52, no swapping is done.
(c) Compare 52 and 27. Since 52 > 27, swapping is done.
    29, 30, 27, 52, 19, 54, 63, 87
(d) Compare 52 and 19. Since 52 > 19, swapping is done.
    29, 30, 27, 19, 52, 54, 63, 87
(e) Compare 52 and 54. Since 52 < 54, no swapping is done.
```

Example….

- Please sort a given data array by using bubble sort

```
A[] = {30, 52, 29, 87, 63, 27, 19, 54}
```

  - Pass 3:

```
29, 30, 27, 19, 52, 54, 63, 87
```

  - Pass 4:

```
(a) Compare 29 and 30. Since 29 < 30, no swapping is done.
(b) Compare 30 and 27. Since 30 > 27, swapping is done.
    29, 27, 30, 19, 52, 54, 63, 87
(c) Compare 30 and 19. Since 30 > 19, swapping is done.
    29, 27, 19, 30, 52, 54, 63, 87
(d) Compare 30 and 52. Since 30 < 52, no swapping is done.
```

Example.....

- Please sort a given data array by using bubble sort

```
A[] = {30, 52, 29, 87, 63, 27, 19, 54}
```

  - Pass 4:

```
29, 27, 19, 30, 52, 54, 63, 87
```

  - Pass 5:

```
(a) Compare 29 and 27. Since 29 > 27, swapping is done.
    27, 29, 19, 30, 52, 54, 63, 87
(b) Compare 29 and 19. Since 29 > 19, swapping is done.
    27, 19, 29, 30, 52, 54, 63, 87
(c) Compare 29 and 30. Since 29 < 30, no swapping is done.
```

Example……

- Please sort a given data array by using bubble sort

```
A[] = {30, 52, 29, 87, 63, 27, 19, 54}
```

  - Pass 5:

```
27, 19, 29, 30, 52, 54, 63, 87
```

  - Pass 6:

```
(a) Compare 27 and 19. Since 27 > 19, swapping is done.
    19, 27, 29, 30, 52, 54, 63, 87
(b) Compare 27 and 29. Since 27 < 29, no swapping is done.
```

Example…….

- Please sort a given data array by using bubble sort

```
A[] = {30, 52, 29, 87, 63, 27, 19, 54}
```

  - Pass 6:

19, 27, 29, 30, 52, 54, 63, 87

  - Pass 7:

```
(a) Compare 19 and 27. Since 19 < 27, no swapping is done.
```

Bubble Sort…

```
BUBBLE_SORT(A, N)

Step 1: Repeat Step 2 For I = 0 to N
Step 2:     Repeat For J = 0 to N - I -1
Step 3:                 IF A[J] > A[J + 1]
                        SWAP A[J] and A[J+1]
            [END OF INNER LOOP]
        [END OF OUTER LOOP]
Step 4: EXIT
```

Insertion Sort.

- Insertion sort is a very simple sorting algorithm in which the sorted array (or list) is built one element at a time
  - It was mentioned by John Mauchly as early as 1946

- The procedure of the insertion sort
  - The array of values to be sorted is divided into two sets
    - One stores sorted values
    - Another contains unsorted values
  - The sorting algorithm will proceed until there are no elements in the unsorted set

Example

- Please sort a given data array by using insertion sort

39	9	45	63	18	81	108	54	72	36

39	9	45	63	18	81	108	54	72	36

A[0] is the only element in sorted list

39	9	45	63	18	81	108	54	72	36

(Pass 1)

9	39	45	63	18	81	108	54	72	36

(Pass 2)

9	39	45	63	18	81	108	54	72	36

(Pass 3)

9	39	45	63	18	81	108	54	72	36

(Pass 4)

9	18	39	45	63	81	108	54	72	36

(Pass 5)

9	18	39	45	63	81	108	54	72	36

(Pass 6)

9	18	39	45	63	81	108	54	72	36

(Pass 7)

9	18	39	45	54	63	81	108	72	36

(Pass 8)

9	18	39	45	54	63	72	81	108	36

(Pass 9)

Insertion Sort..

```
INSERTION-SORT (ARR, N)

Step 1: Repeat Steps 2 to 5 for K = 1 to N – 1
Step 2:      SET TEMP = ARR[K]
Step 3:      SET J = K - 1
Step 4:      Repeat while TEMP <= ARR[J]
                  SET ARR[J + 1] = ARR[J]
                  SET J = J - 1
             [END OF INNER LOOP]
Step 5:      SET ARR[J + 1] = TEMP
        [END OF LOOP]
Step 6: EXIT
```

				J	K					
	9	39	45	63	18	81	108	54	72	36
index	0	1	2	3	4					

Insertion Sort…

			J	K					
9	39	45	63	18	81	108	54	72	36
0	1	2	3	4					

```
INSERTION-SORT (ARR, N)

Step 1: Repeat Steps 2 to 5 for K = 1 to N - 1
Step 2:         SET TEMP = ARR[K]
Step 3:         SET J = K - 1
Step 4:         Repeat while TEMP <= ARR[J]
                      SET ARR[J + 1] = ARR[J]
                      SET J = J - 1
                [END OF INNER LOOP]
Step 5:         SET ARR[J + 1] = TEMP
        [END OF LOOP]
Step 6: EXIT
```

- K=4
  - TEMP=18
  - J=3
    - 18<=63

			J						
9	39	45	63	63	81	108	54	72	36

  - J=2
    - 18<=45

		J							
9	39	45	45	63	81	108	54	72	36

  - J=1
    - 18<=39

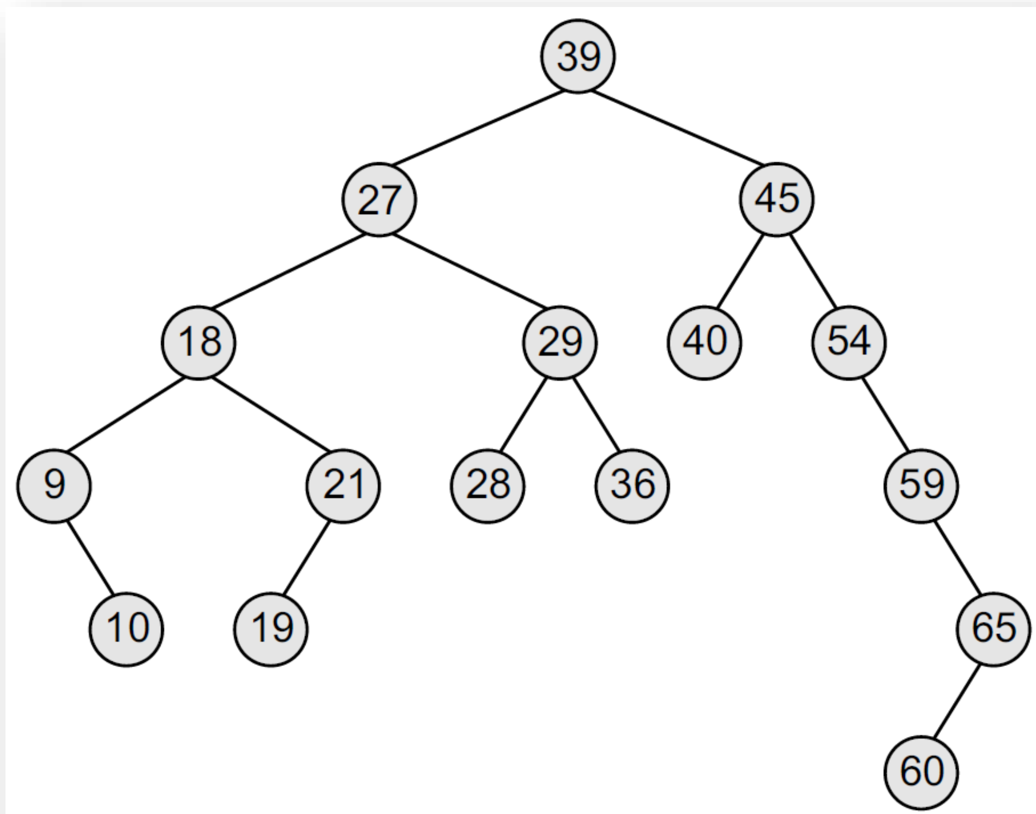
	J								
9	39	39	45	63	81	108	54	72	36

  - J=0
    - 18>9

J									
9	18	39	45	63	81	108	54	72	36

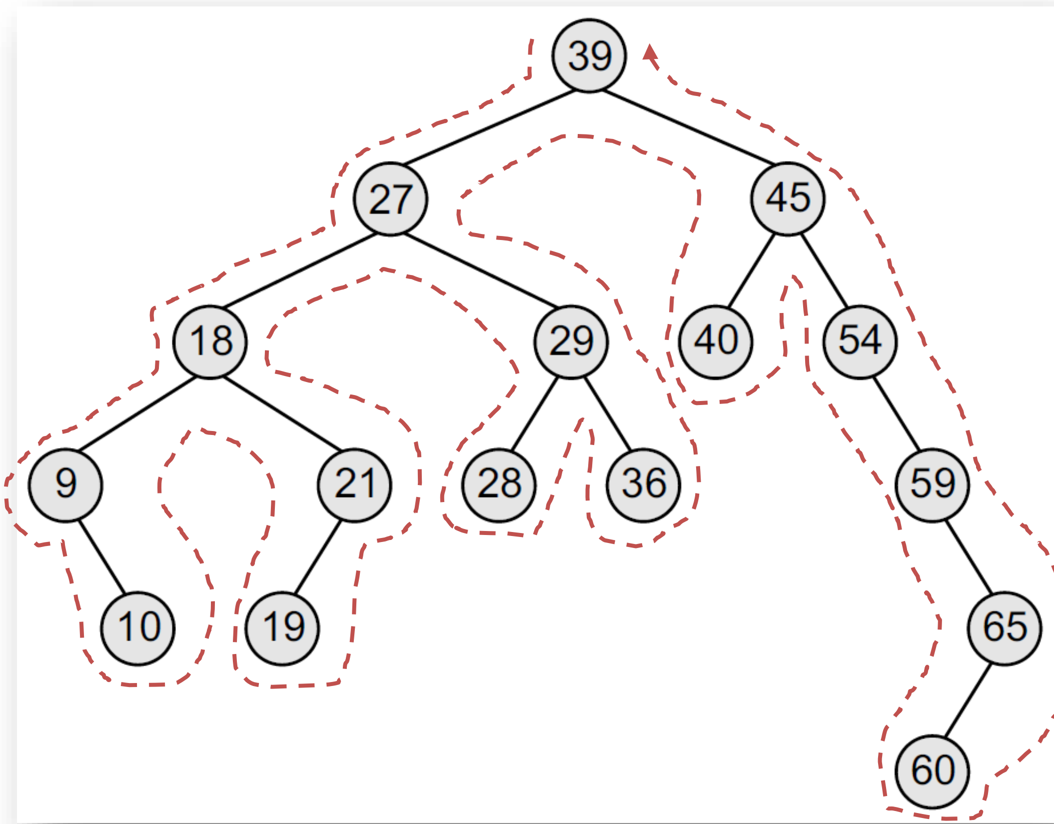
Tree Sort.

- A tree sort is a sorting algorithm that sorts numbers by making use of the properties of binary search tree
  - Build a binary search tree
  - Do an in-order traversal



Tree Sort..

- A tree sort is a sorting algorithm that sorts numbers by making use of the properties of binary search tree
  - Build a binary search tree
  - Do an in-order traversal



About the Midterm

- If you have bad behavior please come to me by Wednesday!

Questions?

kychen@mail.ntust.edu.tw