SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Sorting

Somaiya TRUST

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Outline

- Sorting- concept
- Sorting Terms
- Bubble sort
- Insertion sort
- Counting sort
- Sorting applications

Somaiya TRUST

Sorting

- **Sorting** is any process of arranging items systematically in a particular order
  - Sorting in ascending order :arrange n keys in such a way that $key_i < key_j$ for any i & j such that i<j
  - Sorting in descending order: arrange n keys in such a way that $key_i > key_j$ for any i & j any i & j such that i<j

Sorting Terms

- Stable sort
- Inplace sort
- Number of Passes

Bubble Sort

- *Compares adjacent* array elements
  - *Exchanges* their values if they are *out of order*
- Smaller values *bubble up* to the top of the array
  - Larger values sink to the bottom

FIGURE 10.1
One Pass of Bubble Sort

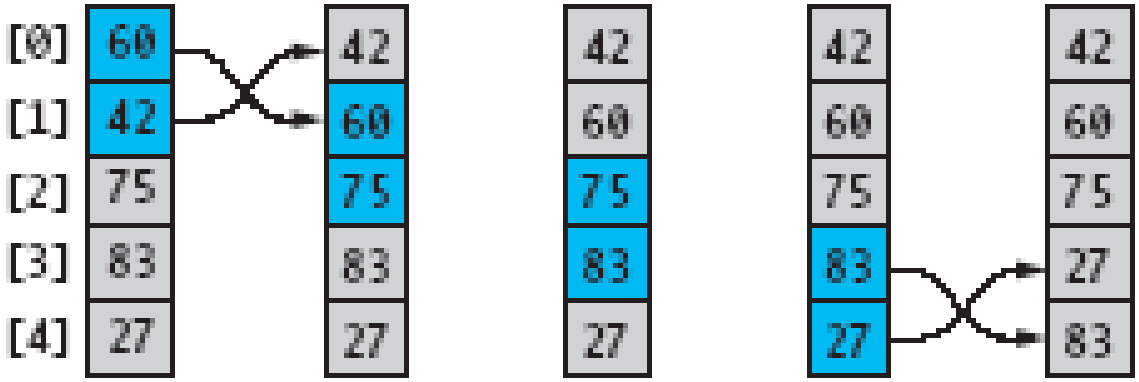
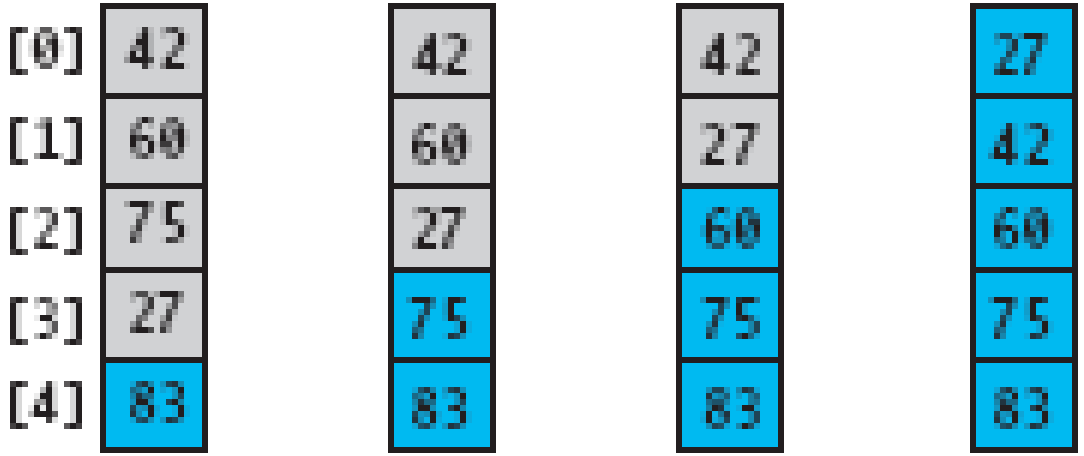


FIGURE 10.2
Array After Completion of Each Pass



Bubble Sort Algorithm

1. do
2. for each pair of adjacent array elements
3. if values are out of order
4. Exchange the values
5. while the array is not sorted

Bubble Sort Algorithm, Refined

1. do
2. Initialize **exchanges** to **false**
3. for each pair of adjacent array elements
4. if values are out of order
5. Exchange the values
6. Set **exchanges** to **true**
7. while **exchanges**

Analysis of Bubble Sort

- Excellent performance *in some cases*
  - But *very poor* performance in others!
- Works ***best*** when array is nearly sorted to begin with
- Worst case number of comparisons: $O(n^2)$
- Worst case number of exchanges: $O(n^2)$
- *Best case* occurs when the array is already sorted:
  - O(n) comparisons
  - O(1) exchanges (none actually)

```
bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

   // Last i elements are already in place
   for (j = 0; j < n-i-1; j++)
      if (arr[j] > arr[j+1])
         swap(&arr[j], &arr[j+1]);
}
```

Insertion Sort

- Based on technique of card players to arrange a hand
  - Player keeps cards picked up so far in *sorted order*
  - When the player picks up a new card

FIGURE 10.3
Picking Up a Hand of Cards

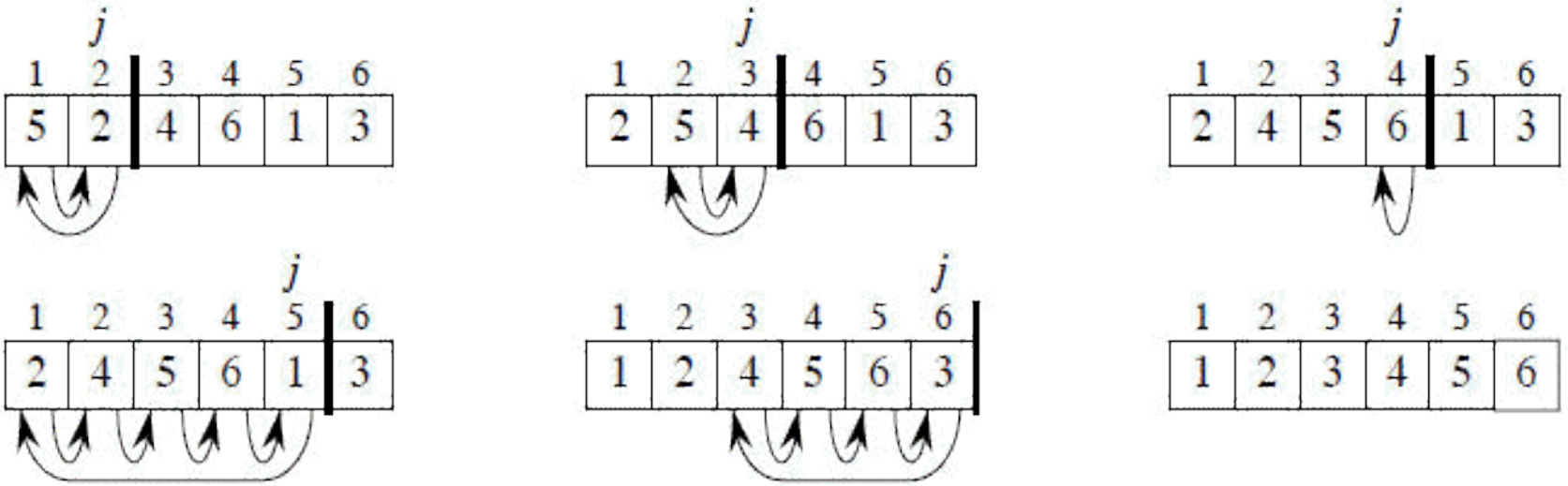
Insertion sort algorithm

INSERTION_SORT takes as parameters an array $A[1..n]$ and the length n of the array. The array A is sorted in place: the numbers are rearranged within the array, with at most a constant number outside the array at any time.

INSERTION_SORT (A)

```
1.    FOR j ← 2 TO length[A]
2.          DO  key ← A[j]
3.              {Put A[j] into the sorted sequence A[1 . . j − 1]}
4.              i ← j − 1
5.              WHILE i > 0 and A[i] > key
6.                        DO A[i +1] ← A[i]
7.                                 i ← i − 1
8.              A[i + 1] ← key
```

Insertion sort algorithm



Stability :

Since multiple keys with the same value are placed in the sorted array in the same order that they appear in the input array, Insertion sort is stable.

Courtesy: Analysis of Algorithm, Coreman

Counting sort

- sorting is based on keys between a specific range.
- It works by counting the number of objects having distinct key values
- Followed by computation of position of each object in the output sequence.

Counting sort

- Initialize count array of the size of input range
- Update the count array to store the count of each unique key.
- Further update the count array with cumulative additions of previous counts
- Shift the count array to right by one position; no circular shift
- Initialize sort array of the size of input sequence
- Update sort array by entering keys from input array at location from count array and increment the count by 1

Counting sort example

- i/p : 2 3 1 2 4 5 2 1 5 4
- N= 10, range: 1:5

Initialize count array of the size of input range						
count array	0	1	2	3	4	5
	0	0	0	0	0	0
Update the count array to store the count of each unique key.						
count array	0	1	2	3	4	5
	0	2	3	1	2	2
Further update the count array with cumulative additions of previous counts						
count array	0	1	2	3	4	5
	0	2	5	6	8	10
Shift the count array to right by one position; no circular shift						
count array	0	1	2	3	4	5
	0	0	2	5	6	8

Initialize sort array of the size of input sequence

Sort Array	0	1	2	3	4	5	6	7	8	9

Update sort array by entering keys from input array at location from count array and increment the count by 1

i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0	2	5	6	8				
Output Sorted	0	1	2	3	4	5	6	7	8	9
y	1	1	2	2	2	3	4	4	5	5

Counting sort step by step

i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0	2 3		5	6	8			
Sort Array	0	1	2	3	4	5	6	7	8	9
			2							
i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0	2 3	5 6	6	8				
Sort Array	0	1	2	3	4	5	6	7	8	9
			2			3				

Somaiya TRUST

Counting sort step by step

i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0 1	2 3	5 6	6	8				
Sort Array	0	1	2	3	4	5	6	7	8	9
	1		2			3				
i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0 1	2 3 4	5 6	6	8				
Sort Array	0	1	2	3	4	5	6	7	8	9
	1		2	2		3				

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Counting sort step by step

i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0 1	2 3 4	5 6	6 7	8				
Sort Array	0	1	2	3	4	5	6	7	8	9
	1		2	2		3	4			
i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0 1	2 3 4	5 6	6 7	8 9				
Sort Array	0	1	2	3	4	5	6	7	8	9
	1		2	2		3	4		5	

Somaiya TRUST

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Counting sort step by step

i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0 1	2 3 4 5	5 6	6 7	8 9				
Sort Array	0	1	2	3	4	5	6	7	8	9
	1		2	2	2	3	4		5	

i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0 1 2	2 3 4 5	5 6	6 7	8 9				
Sort Array	0	1	2	3	4	5	6	7	8	9
	1	1	2	2	2	3	4		5	

Somaiya TRUST

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Counting sort step by step

i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0 1 2	2 3 4 5	5 6	6 7	8 9 10				
Sort Array	0	1	2	3	4	5	6	7	8	9
	1	1	2	2	2	3	4		5	5

i/p	2	3	1	2	4	5	2	1	5	4
count array	0	1	2	3	4	5				
	0	0 1 2	2 3 4 5	5 6	6 7 8	8 9 10				
Sort Array	0	1	2	3	4	5	6	7	8	9
	1	1	2	2	2	3	4	4	5	5

Somaiya TRUST

Analysis of sorting algorithms

Sr. no.	Algorithm	Stable?	Inplace?	#passes?
1	Bubble			
2	Insertion			
3	Counting			

Thank you