Sorting in Arrays

Course Code: CSC 2106 Course Title: Data Structure (Theory)

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**5.1**	**Week No:**	**5**	**Semester:**	**Fall 24**
Lecturer:	*Mashiour Rahman (mashiour@aiub.edu)*				

Lecture Outline

1. Sorting: Definition and example
2. Bubble Sort
3. Selection Sort
4. Insertion Sort [next class]

Sorting

Definition and example

Definition: Sorting arranges values in an array in an specific order. It may be alphabetical, increasing numerical and decreasing numerical.

Example:

An unsorted array of numbers

12	45	9	55	10	7

Sorted List (increasing or ascending order)

7	9	10	12	45	55

A sorted array of names

adam	david	deb	jack	neil

Sorting

Applications

Commercial computing: Organizations organize their data by sorting it. Accounts to be sorted by name or number, transactions to be sorted by time or place, mail to be sorted by postal code or address, files to be sorted by name or date, or whatever, processing such data is sure to involve a sorting algorithm somewhere along the way.

Search for information: Keeping data in sorted order makes it possible to efficiently search through it using the classic binary search algorithm.

Operations research: Suppose that we have N jobs to complete. We want to maximize customer satisfaction by minimizing the average completion time of the jobs. The shortest processing time first rule, where we schedule jobs in increasing order of processing time, is known to accomplish this goal.

Sorting

Applications

Combinatorial search: A classic paradigm in artificial intelligence is to define a set of configurations with well-defined moves from one configuration to the next and a priority associated with each move.

Prim's algorithm, Kruskal's algorithm and Dijkstra's algorithm are classical algorithms that process graphs. These uses sorting.

Huffman compression is a classic data compression algorithm that depends upon processing a set of items with integer weights by combining the two smallest to produce a new one whose weight is the sum of its two constituents.

String processing algorithms are often based on sorting.

Sorting

Bubble Sort

Definition: Bubble sort compares adjacent numbers in pairs from one end (beginning) exchanging them if the are in wrong order. When it reaches the end of data, it starts over until all the data is in right order.

Algorithm:

Input: A (array), N (#elements)

$pass = 1$

Step 1: $u = N - pass$

Compare the pairs $(A[0], A[1]), (A[1], A[2]), (A[2], A[3]), \dots, (A[u-1], A[u])$ and Exchange pair of elements if they are not in order.

Step 2: $pass = pass + 1$. If the array is unsorted and $pass < N - 1$ then go to step 1

Bubble Sort

When not to use

When dealing with a large set of data.

When you are looking for a quick algorithm. Compared to other sorting algorithm, bubble sort is really slow.

Bubble Sort

When to use

Not much use in the real world, but it's easy to understand and fast to implement.

It is used when a fast algorithm is needed to sort:
1) an extremely small set of data (Ex. Trying to get the books on a library shelf back in order.) or
2) a nearly sorted set of data. (Ex. Trying to decide which laptop to buy, because it is easier to compare pairs of laptops one at a time and decide which you prefer, than to look at them all at once and decide which was best.)

Bubble Sort

Simulation

Increasing order

Correct order!

wrong order!

PASS 1

Bubble Sort

Simulation

Increasing order

Correct order!

wrong order!

PASS 2

Bubble Sort

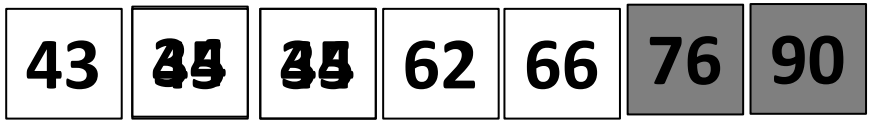
Simulation

Increasing order

Correct order!

wrong order!

PASS 3



Bubble Sort

Simulation

Increasing order

Correct order!

wrong order!

PASS 4

~~77~~ ~~54~~	~~77~~ ~~54~~	45	62	66	76	90

Bubble Sort

Simulation

Increasing order

PASS 5

Correct order!

34	43	45	62	66	76	90

There was **no exchange or swap** in pass 5. So we can stop here otherwise it could go to pass 6

Sorting

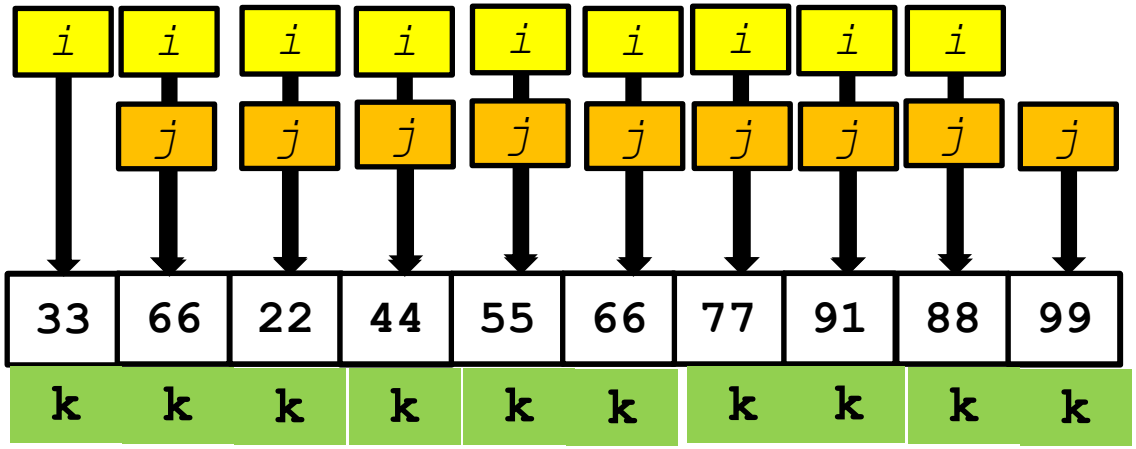
Selection Sort

Selection sort: Locate smallest element in array. Interchange it with element in position 0. Locate next smallest element in array. Interchange it with element in position 1. Continue until all elements are arranged in order.

Algorithm:
Input: A (array), N (#elements)
$i = 0$
Step 1: Find the smallest element from the positions starting at index i to $N - 1$. Swap the smallest element with the element at index i.
Step 2: $i = i + 1$. If $i < N - 1$ go to step 1

Selection Sort

Simulation



Selection Sort

Conclusion

More efficient than Bubble Sort, since fewer exchanges.

An array with N elements needs exactly N-1 selections for fixing index 0 to N-2

Check: What will happen if a sorted array is given as input to Bubble Sort and Selection Sort? You can think in terms of number of comparisons required for the both algorithms.

References

1. https://algs4.cs.princeton.edu/25applications/
2. https://en.wikipedia.org/wiki/Bubble_sort
3. https://en.wikipedia.org/wiki/Selection_sort
4. https://en.wikipedia.org/wiki/Sorting_algorithm
5. Nice animations are available here https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

Books

- **“Schaum's Outline of Data Structures with C++”**. By John R. Hubbard
- **“Data Structures and Program Design”,** Robert L. Kruse, 3rd Edition, 1996.
- **“Data structures, algorithms and performance”,** D. Wood, Addison-Wesley, 1993
- **“Advanced Data Structures”,** Peter Brass, Cambridge University Press, 2008
- **“Data Structures and Algorithm Analysis”,** Edition 3.2 (C++ Version), Clifford A. Shaffer, Virginia Tech, Blacksburg, VA 24061 January 2, 2012
- **“C++ Data Structures”,** Nell Dale and David Teague, Jones and Bartlett Publishers, 2001.
- **“Data Structures and Algorithms with Object-Oriented Design Patterns in C++”,** Bruno R. Preiss,