# ENG 346
# Data Structures and Algorithms for Artificial Intelligence
## Sorting

Dr. Kürşat İnce
kince@gtu.edu.tr

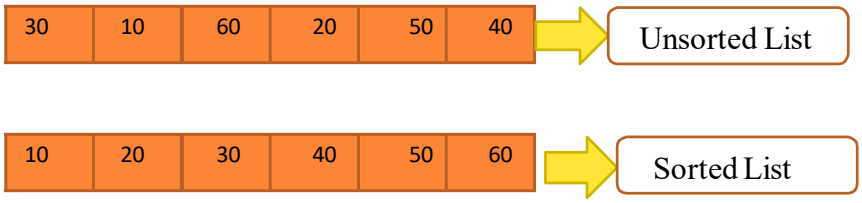


## Agenda

- Bubble Sort
- Insertion Sort
- Selection Sort
- Merge Sort
- Quick Sort

# SORTING

- Sorting refers to operations of arranging a set of data in a given order.






# Sorting

- Search Operations:
  - Sorted data allows for efficient search operations. Binary search, for example, can be performed on a sorted array or list in O(log n) time, which is significantly faster than linear search on unsorted data (O(n)).
- Insertion and Deletion:
  - In certain data structures like binary search trees, maintaining sorted order simplifies and accelerates the process of insertion and deletion. It ensures that the tree remains balanced and allows for faster search operations.
- Efficient Merging:
  - When working with data structures like heaps or priority queues, merging two sorted structures becomes a more straightforward task. This is particularly important in applications such as external sorting.
- Range Queries:
  - Sorting facilitates efficient range queries. For example, finding all elements within a given range in a sorted array or list can be done much more efficiently than in an unsorted structure.

# Sorting

- Data Retrieval:
  - In scenarios where data needs to be retrieved in a specific order, having the data pre-sorted simplifies and speeds up the retrieval process. This is particularly important in databases and information retrieval systems.
- Optimizing Algorithms:
  - Certain algorithms and data structures perform better or are easier to implement when working with sorted data. For example, dynamic programming algorithms may take advantage of sorted input to optimize their runtime.
- Duplicate Removal:
  - Sorting facilitates the removal of duplicates in a dataset. Adjacent duplicate elements can be easily identified and removed in a single pass through sorted data.
- Intersection of Sets:
  - When working with sets, determining the intersection of two sets becomes more efficient if the sets are sorted. This is especially relevant in applications involving database queries.



# Applications of Sorting

- **Search Algorithms:**
  - Efficient searching often relies on pre-sorted data, improving search times.
- **Database Indexing:**
  - Crucial for creating indexes in databases, allowing for faster query operations.
- **E-commerce and Online Marketplaces:**
  - Used to display products in a particular order, such as by price or popularity.
- **Log Analysis:**
  - Sorting log entries by timestamp aids in identifying patterns and analyzing system behavior.
- **Task Scheduling:**
  - Sorting is applied in scheduling tasks to optimize the order in which they are executed.
- …

# Types of Sorting

- Internal Sorting:
  - If all the data to be sorted can be adjusted in main memory then it is called as Internal Sorting.
- External Sorting:
  - If data to be stored is large and requires external memory then the type is called as External Sorting.

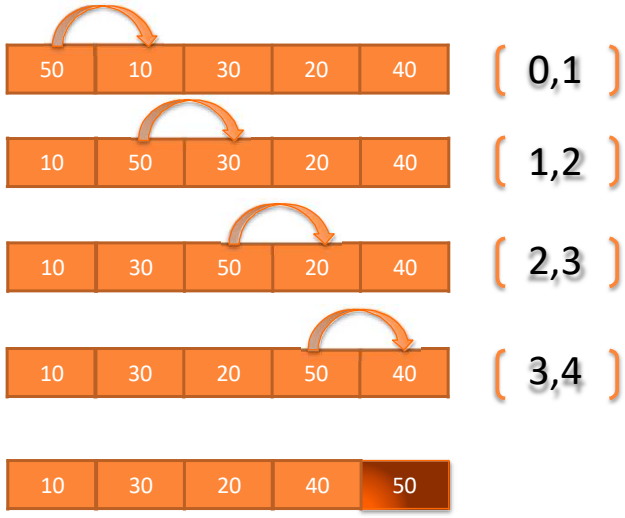


# Bubble Sort

- Algorithm of bubble sort includes two steps repeated until the list is sorted.
- Compare adjacent elements, if the element on right side is smaller then swap their positions.
- Compare first element, second element and so on completion of Pass 1 the largest element is at last position.


Pass 1:

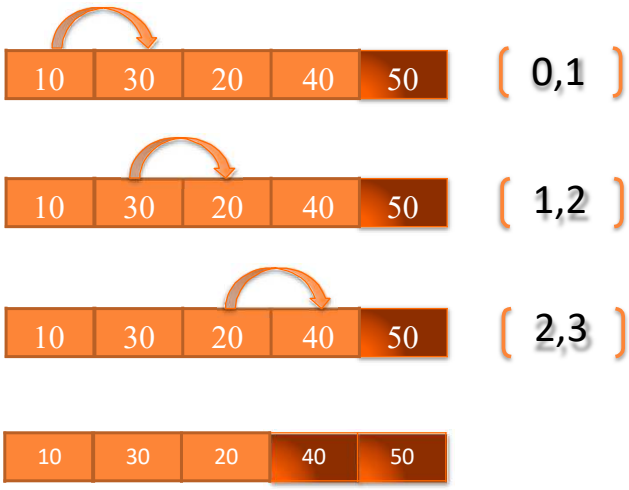
Number Of Passes = Max – 1 = 5 – 1 = 4



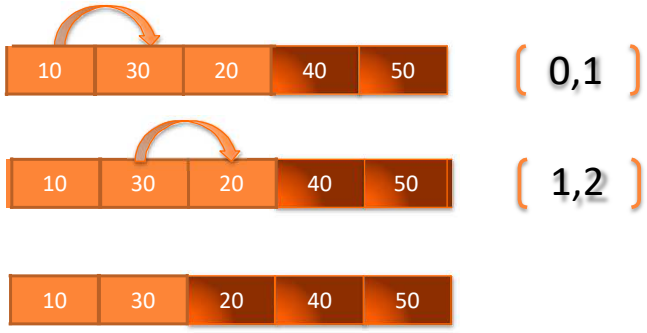





Pass 2:



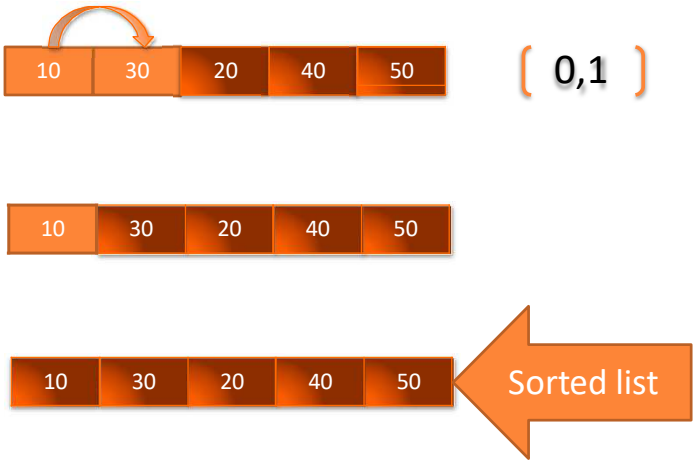
## Pass 3:


10 30 20 40 50
0,1
10 30 20 40 50
1,2
10 30 20 40 50





## Pass 4:

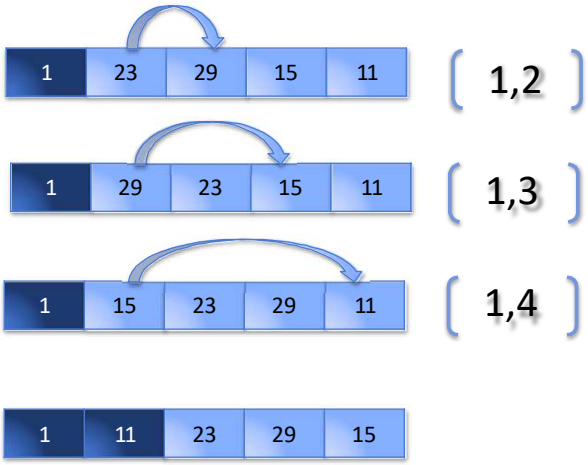

10 30 20 40 50
0,1
10 30 20 40 50
10 30 20 40 50
Sorted list

# Selection Sort

- Here in selection sort the algorithm depends on the zeroth element majorly.
- The Zeroth element is compared with the first element and if the element at right is found smaller then their positions are swapped or exchanged.
- The same procedure is carried with all elements of the list resulting into a fully sorted list.

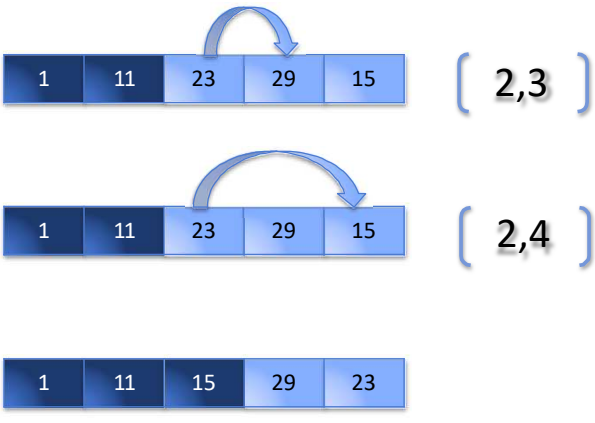


## Pass 2:

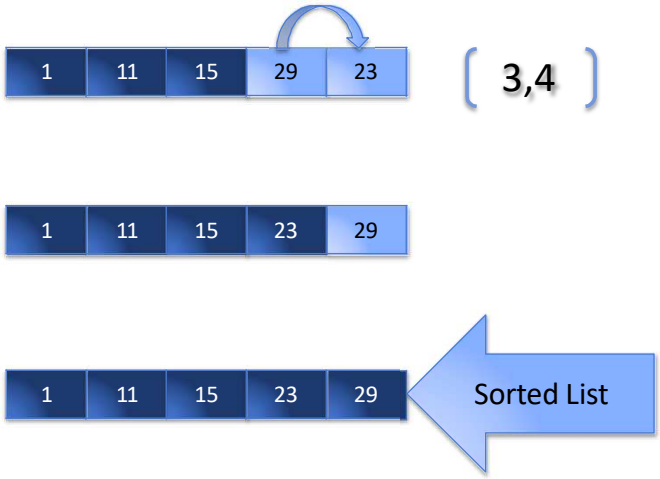

1 23 29 15 11
1,2
1 29 23 15 11
1,3
1 15 23 29 11
1,4
1 11 23 29 15



## Pass 3:


1 11 23 29 15
2,3
1 11 23 29 15
2,4
1 11 15 29 23

Pass 4:






# Insertion Sort

- In insertion sort the elements are compared and inserted to respective index place.
- It starts with comparision of 1st and 0th element in pass 1.
- In pass 2 the second element is compared with the 1st and 0th element.
- Doing so with all the elements in the list appropriate element is inserted by shifting elements on right.

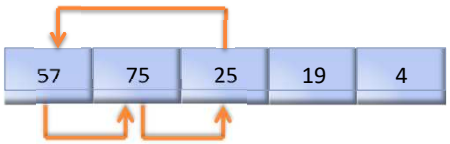

Pass 1:

Number Of Passes = Max – 1 = 5 – 1 = 4


0,1

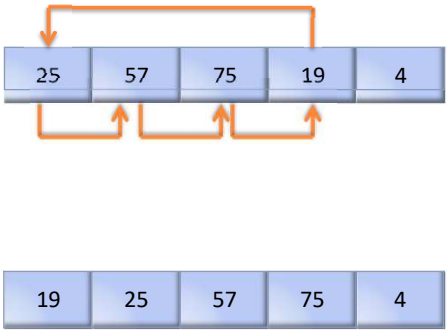



Pass 2:




0,2

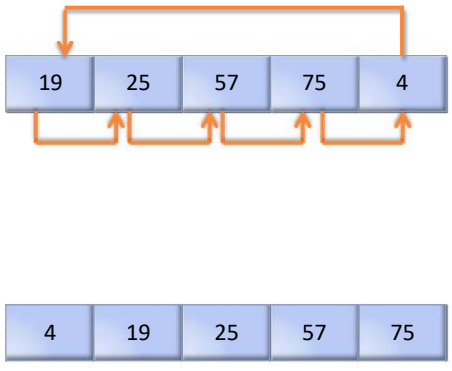
## Pass 3:




0,3



## Pass 4:




0,4