RAMA UNIVERSITY

www.ramauniversity.ac.in

FACULTY OF ENGINEERING & TECHNOLOGY

Lecture : 06

Mr. Nilesh
Assistant Professor
Computer Science & Engineering

Outline

- Introduction Sorting and Order Statistics
- Types of Sorting algorithms
- Order statistics
- Selection Sort

Introduction Sorting and Order Statistics

Introduction

Algorithms that solve the following sorting problem:

Input: A sequence of n numbers <a1, a2, . . . , an>.

Output:

A permutation (reordering) <a'1, a'2, . . . , a'n> of the input sequence such that a'1 a'2 a'n.

The input sequence is usually an n-element array,

although it may be represented in some other fashion, such as a linked list.

➢ The structure of the data

- The numbers to be sorted are rarely isolated values.
- That may be a record (Each record contains a key)
- which is the value to be sorted,
- When a sorting algorithm permutes the keys, it must permute the satellite data as well.
- If each record is large in size - we often permute an array of pointers to the records rather than the records themselves in order to minimize data movement.

❑ Types of Sorting algorithms

- **How many Sorting also. We have?**

- Selection Sort
- Bubble Sort
- Insertion Sort
- Merge Sort
- Quick Sort
- Heap Sort
- Radix Sort
- ShellSort
- Pigeonhole Sort
- Cycle Sort
- Cocktail Sort
- Strand Sort
- Recursive Bubble Sort
- Recursive Insertion Sort
- Iterative Merge Sort
- Iterative Quick Sort
- Counting Sort
- Bucket Sort
- TimSort

Order statistics

- Order statistics are sample values placed in ascending order. The study of order statistics deals with the applications of these ordered values and their functions.

Let's say you had three weights:

X_1 = 22 kg, X_2 = 44 kg, and X_3 = 12 kg.

To get the **order statistics (Y_n)**, put the items in **numerical increasing order**:

Y_1 = 12 kg

Y_2 = 22 kg

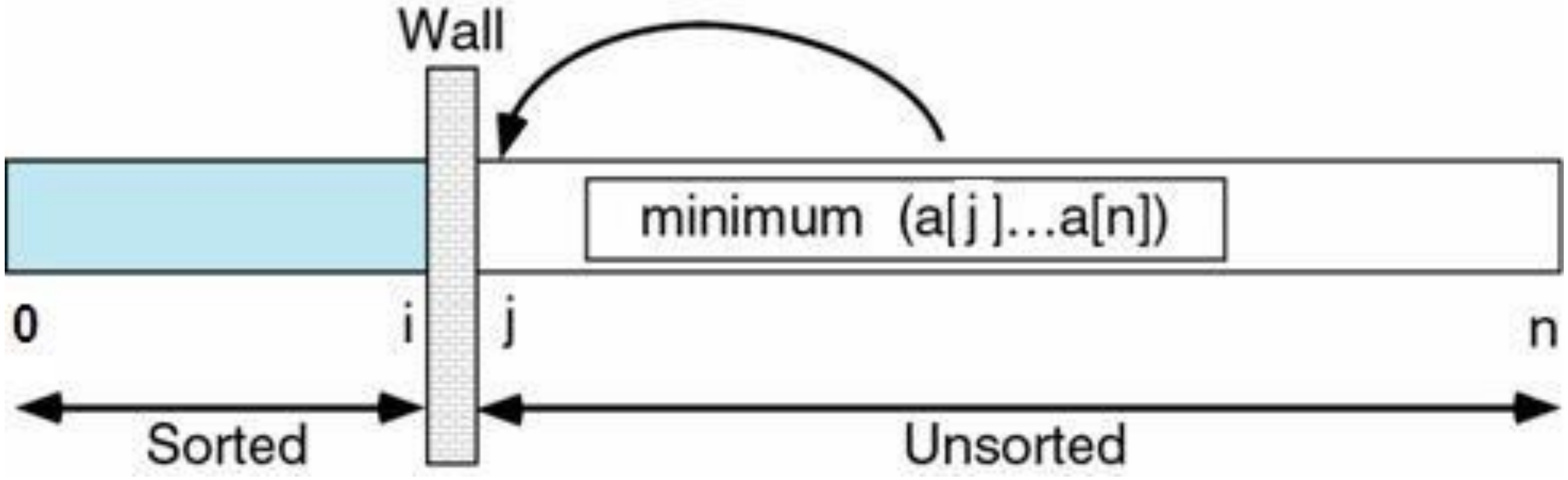
Y_3 = 44 kg

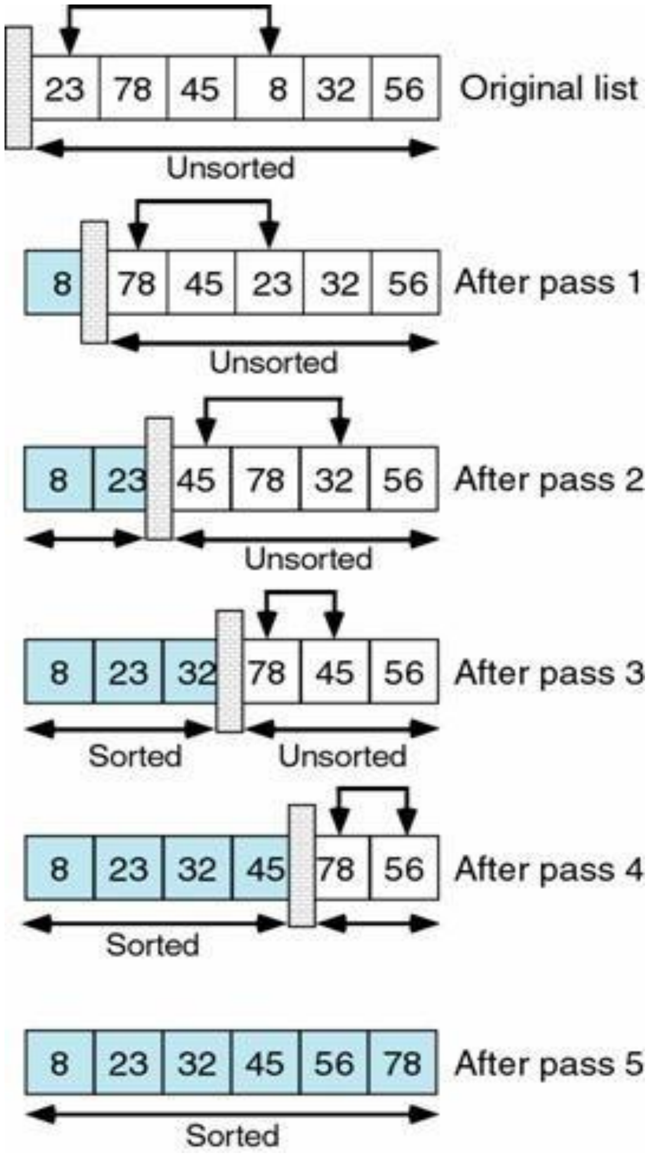
The kth smallest X value is normally called the *k*th order statistic.

Selection Sort

Suppose an array A with N

1. Find the smallest element in the list & put it in the first position.
2. Find 2nd smallest element in the list and put in the second position
3. So on




23 78 45 8 32 56 Original list
Unsorted
8 78 45 23 32 56 After pass 1
Unsorted
8 23 45 78 32 56 After pass 2
Unsorted
8 23 32 78 45 56 After pass 3
Sorted
Unsorted
8 23 32 45 78 56 After pass 4
Sorted
8 23 32 45 56 78 After pass 5
Sorted

- Time Complexity

  Notes:

  Outer loop executes N-1 times(No each rd.. For final elements)

  Inner loop executes N-i-1 comparisons.

  (N-1)* (N-2)+….+2+1 = N*(N-1)/2

  Best case: Square(n)

  Average Case: Square(n)

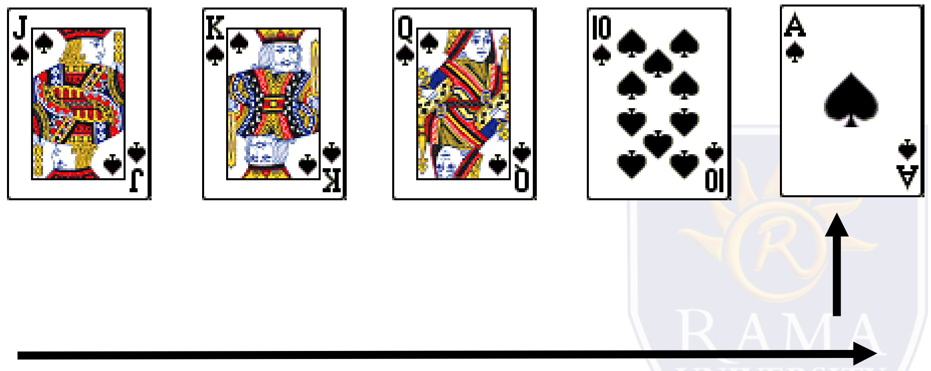
Advantages

- Easy to write
- Can be done “In place”
- Can be done on linked lists too(Keep a tail Pointer)

Disadvantages

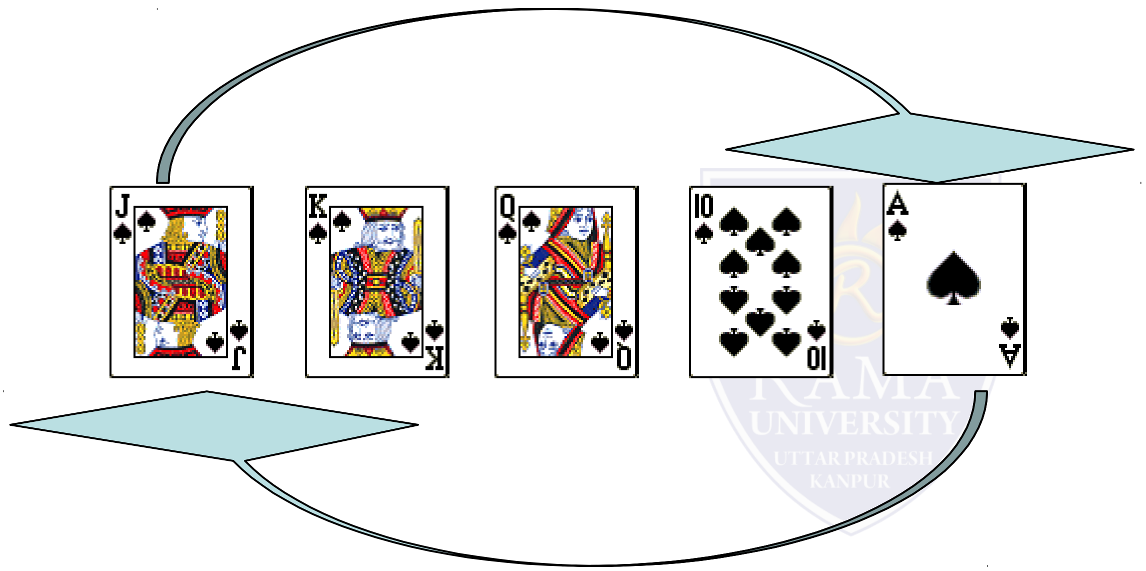
- It is about square(n) even in the best case for comparisons.
- So the running time is Approx. Square(n)

Example

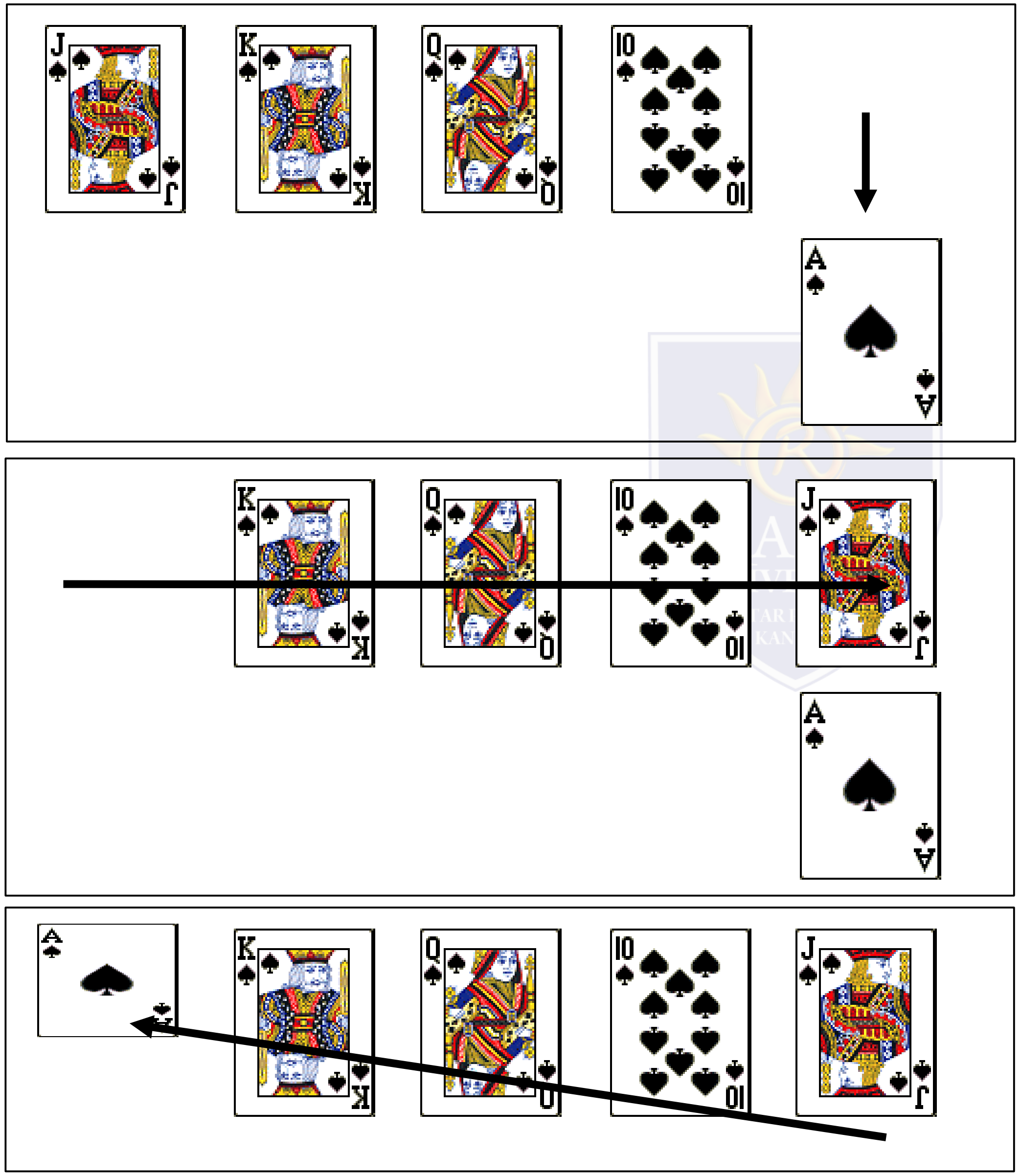


Initial Configuration

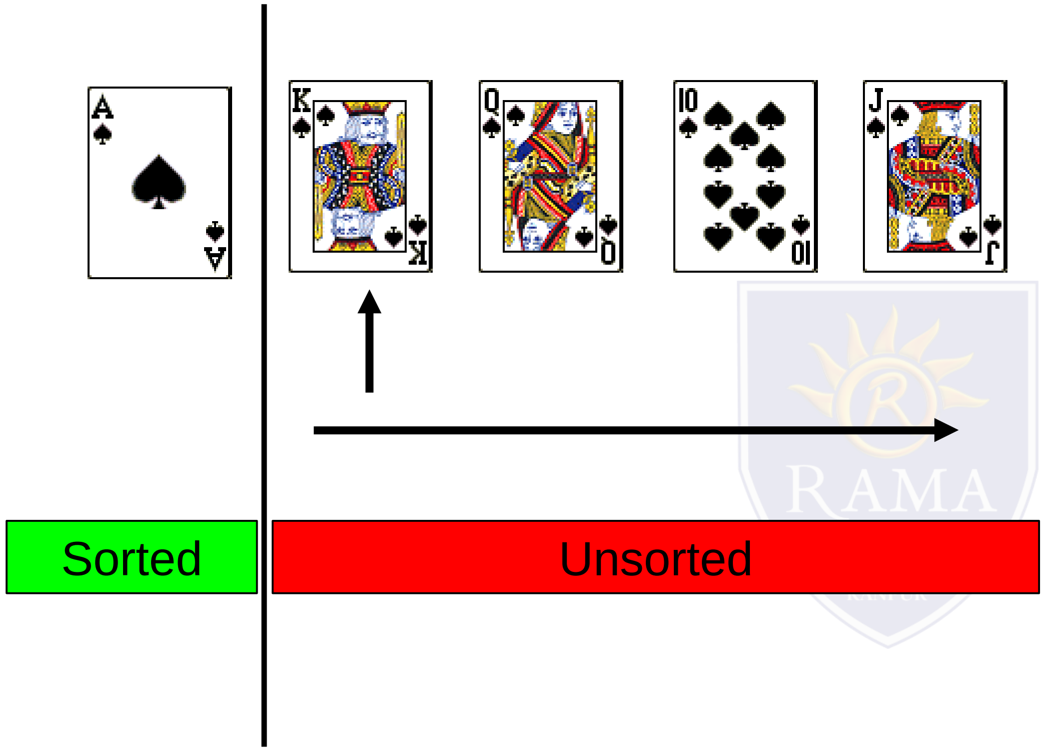
(search all cards and find the largest)



Swap the two cards



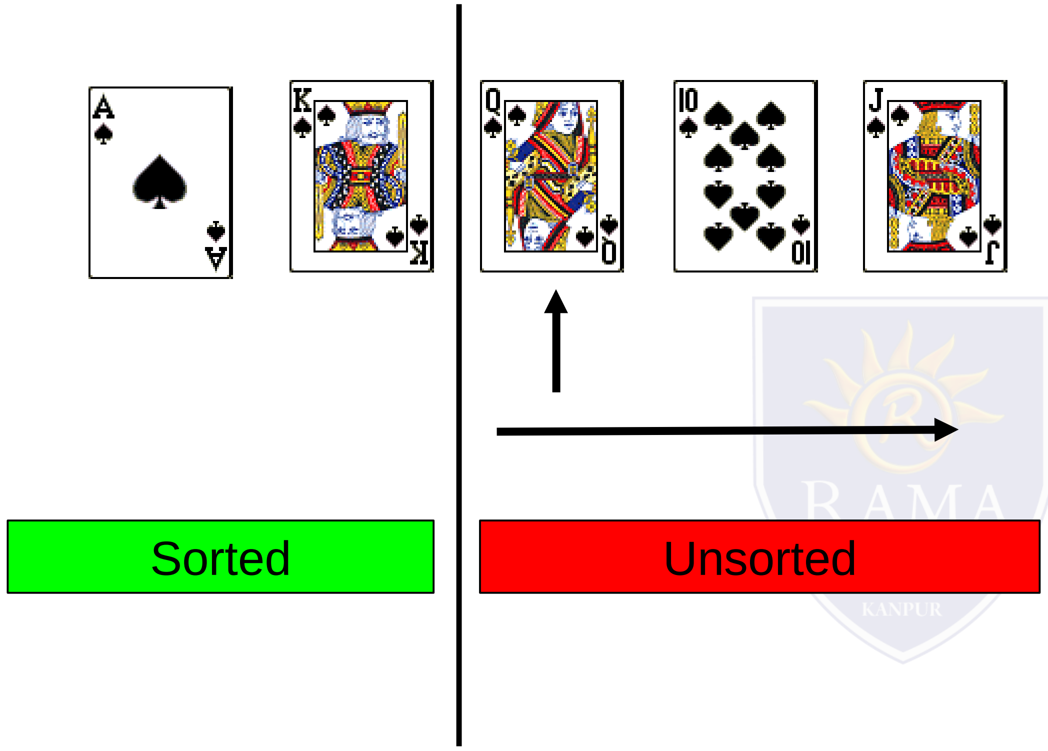
As before, the swap is performed in three steps.



Among the remaining cards
the king is the largest.

It will remain in place.

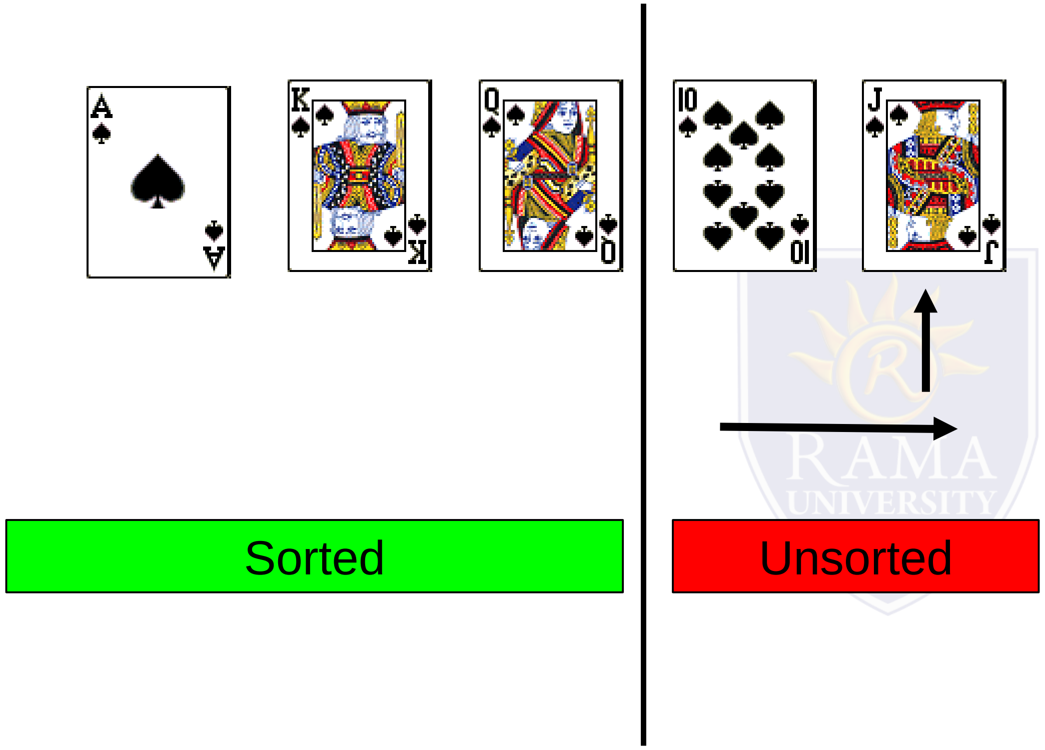
But the algorithm may perform
Some empty operations
(ie., swap it with itself in
place)



Among the remaining cards
the queen is the largest.

It will remain in place.

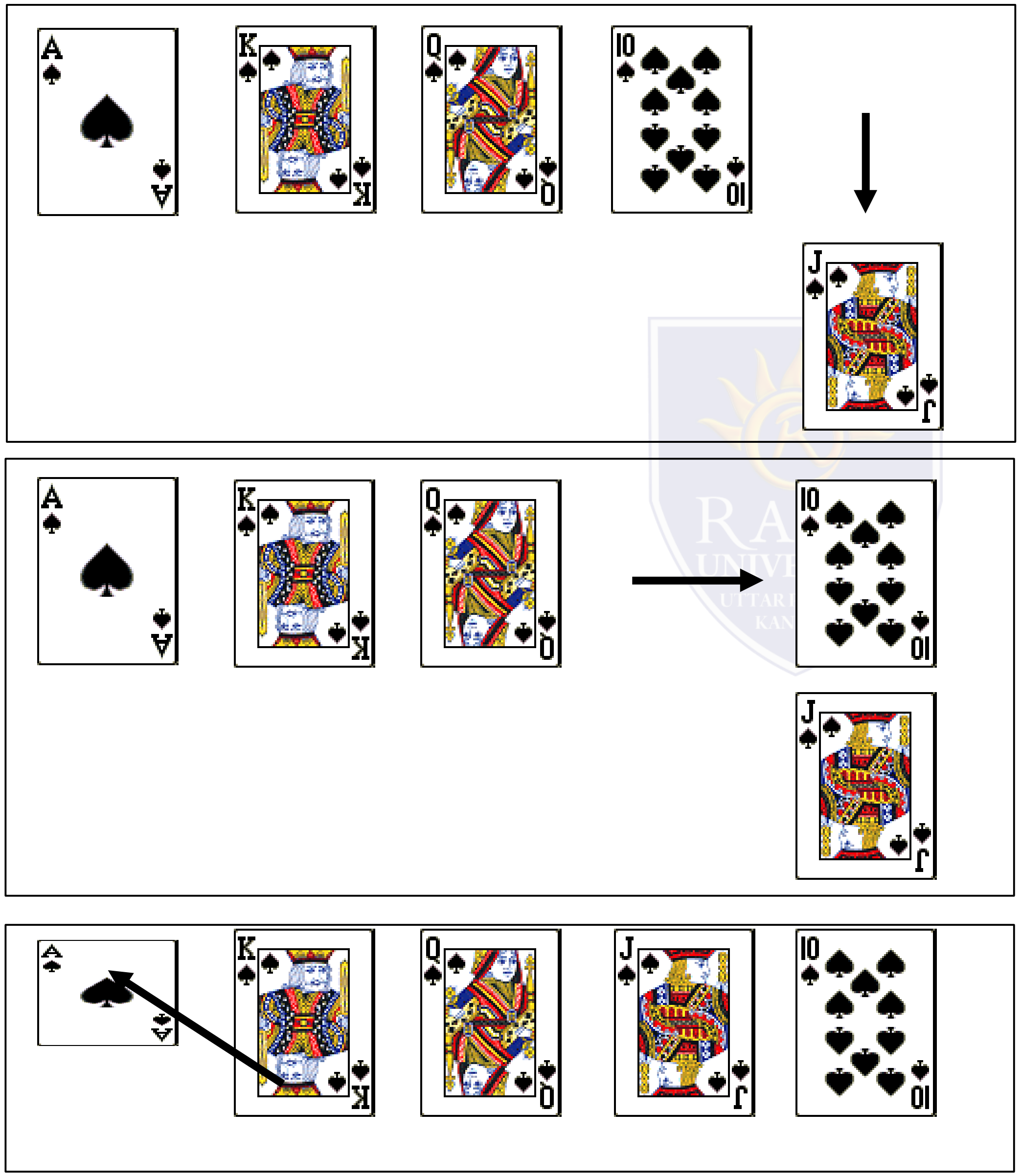
But the algorithm may perform
Some empty operations
(i.e., swap it with itself in place)



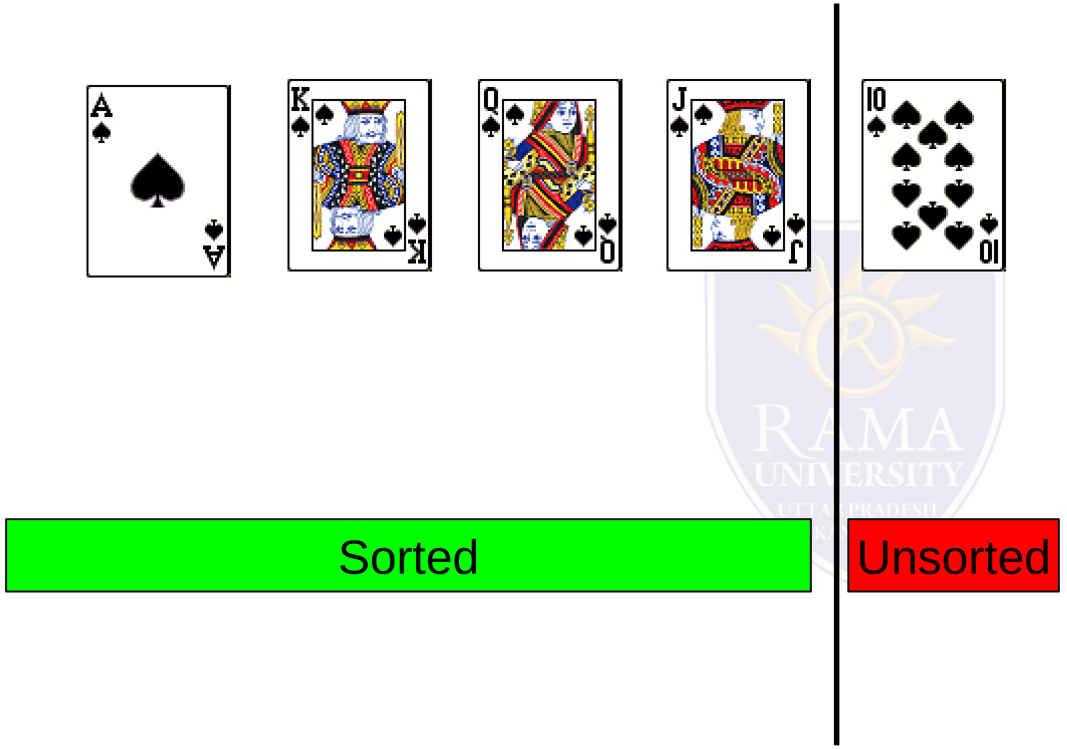
Among the remaining cards
the Jack is the largest.

It will remain in place.

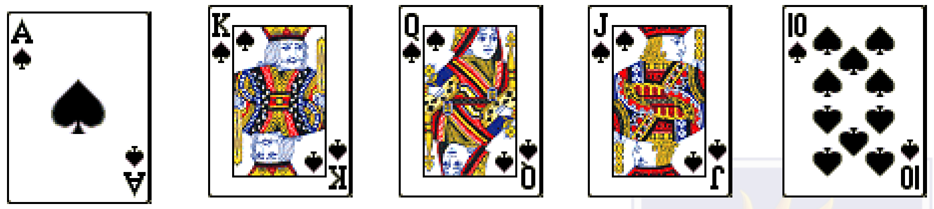
But the algorithm may perform
Some empty operations
(i.e., swap it with itself in place)



As before, the swap is performed in three steps.



We are down to the last card.
Because there is only one and
Because we know that it is
Smaller than all the rest
We don't need to do anything
Else with it. This is why the
Algorithm goes up to < N-1



Sorted

All cards are now sorted.