Median Selection, Partitioning Algorithm

(Class 10)

From Book's Page No 227 (Chapter 9)

- In previous class we studied selection problem to find the rank of a number from a set of numbers.
- At first, the brute force approach was to compare each number to every other number.
- So, we got $O(n^2)$ for that brute force algorithm of selection problem.

- Then, we improved the algorithm by using the sorting algorithm then search the specific number.
- The selection problem is useful in finding the median number of a given set of values.

The Sieve Technique

- This is a variation of divide and conquer strategy.
- This technique also divides the data to solve the problem.
- We will use a variation of the divide and conquer strategy called the sieve technique.
- In divide and conquer, we break the problem into a small number of smaller subproblems which we solve recursively.

- In the selection problem, we are looking for an item.
- We will divide the problem into subproblems.
- However, we will discard those small subproblems for which we determine that they do not contain the desired answer.

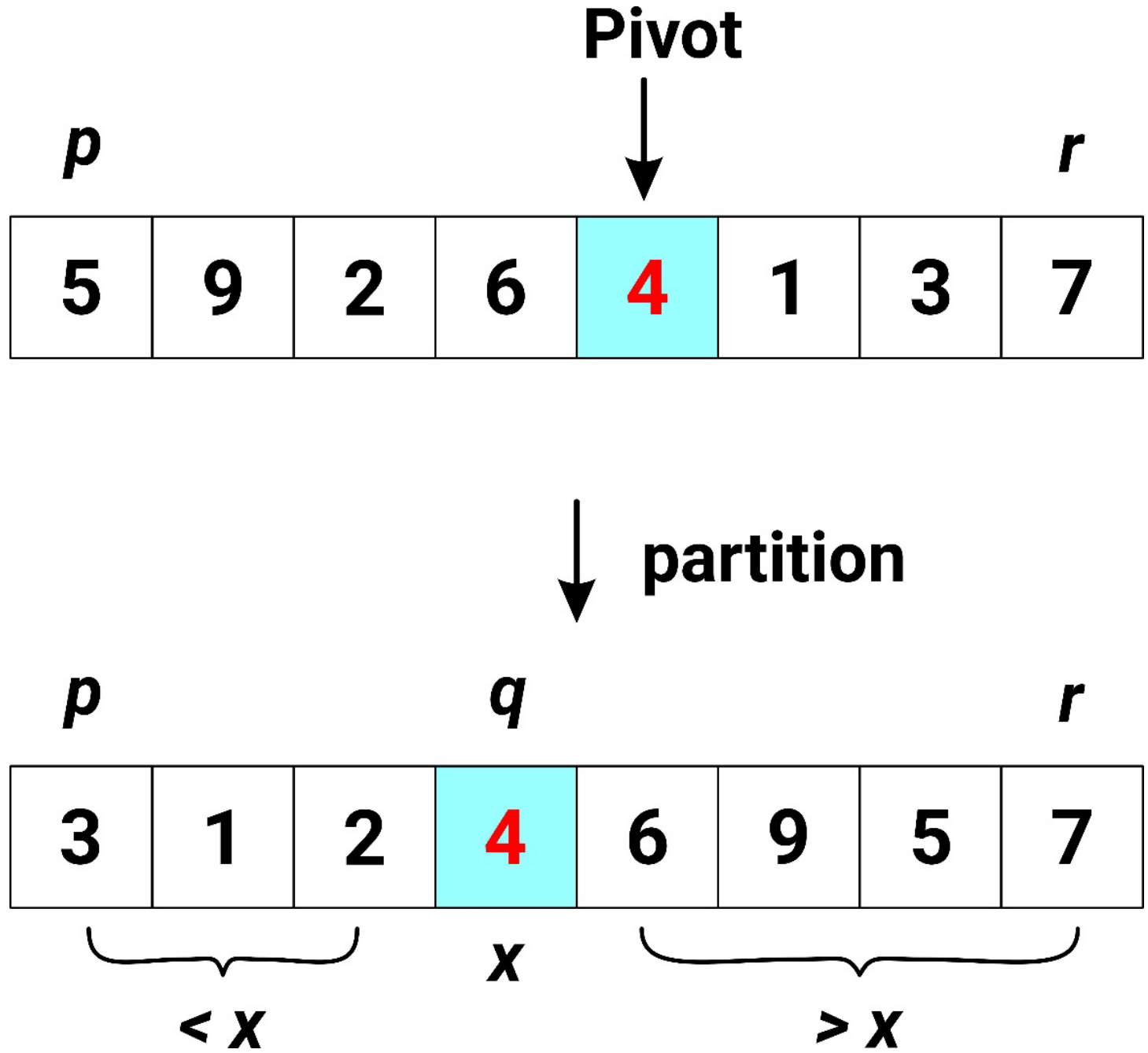
- We are given n numbers and want to find that specific number having rank k.
- We are searching for the number k, not sorting the number.
- We divide the given set into subsets and discard the subset which don’t have the value k.

- Here is how the sieve technique will be applied to the selection problem:
  - We will begin with the given array $A[1 \ldots n]$.
  - We will pick an item from the array, called the *pivot* element which we will denote by x.
  - We will talk about how an item is chosen as the pivot later; for now, just think of it as a random element of set A.

- We partition set A into three parts:
  - $A[q]$: contains the pivot element x.
  - $A[1 \ldots q-1]$: will contain all the elements that are less than x.
  - $A[q+1 \ldots n]$: will contains all elements that are greater than x.

- Within each sub array, the items may appear in any order (not sorted).

- There are different techniques for the selection of the pivot element.
- For example, Median, Mean as Pivot, Mode as Pivot, i^{th} Element as Pivot, Largest/Smallest Element, etc.
- But for now, we will pick the pivot element randomly.
- It is quite similar to the Quick Sort where we partition the data using pivot element.

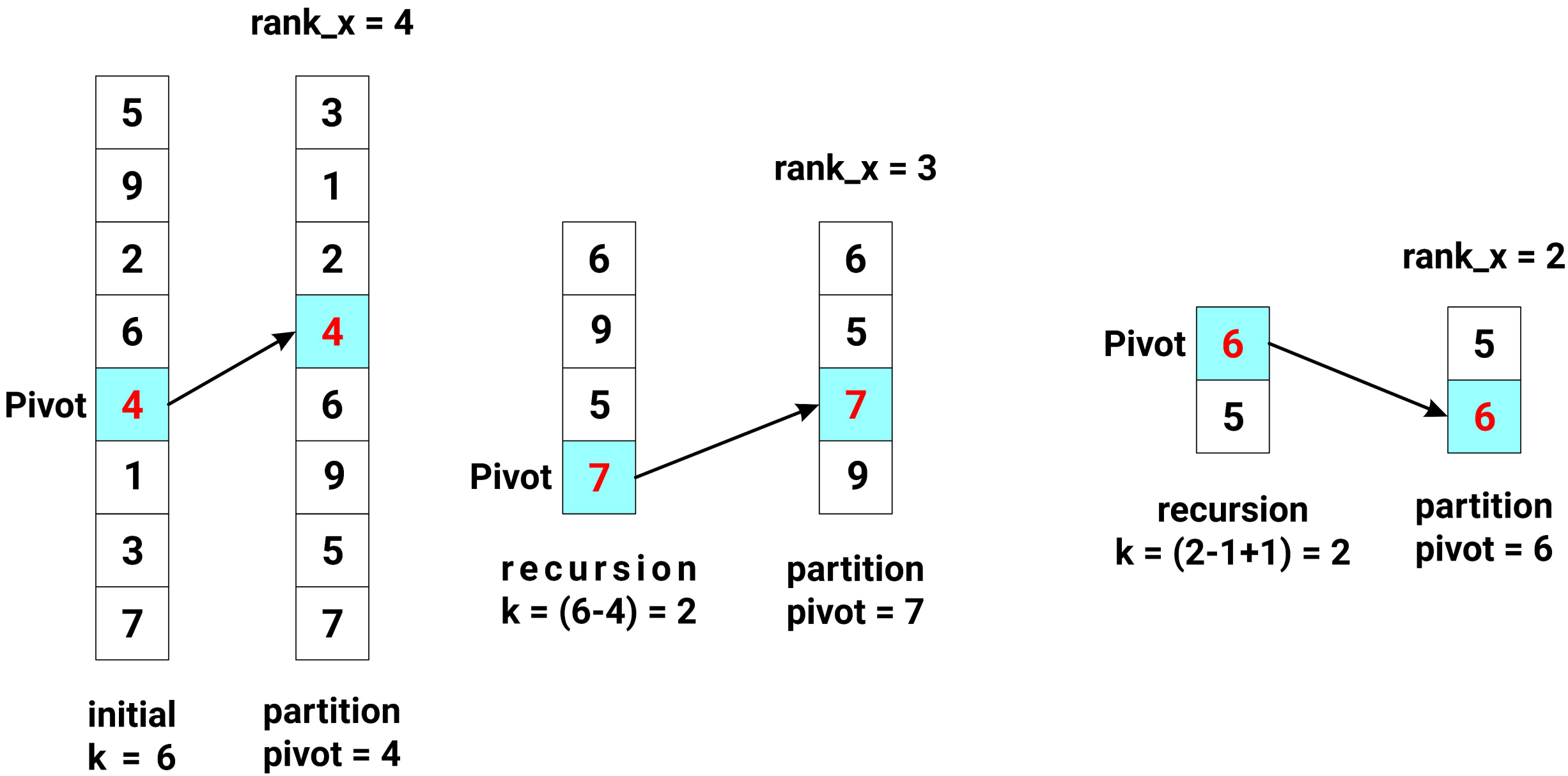

Pivot
p
r
5
9
2
6
4
1
3
7
partition
p
q
r
3
1
2
4
6
9
5
7
x
< x
> x

- We can see that the two subsets from the left and right of the q are not sorted.
- Do we need them sorted?
- As we only want the check that if the number k is in left or right side so there is no need to sort the values.
- If we sort the data, then the pivot (e.g., 4) will again go to this same position.

- The rank of the pivot x is $q - p + 1$ in $A[p \ldots r]$.
- Let $rank\_x = q - p + 1$.
- If $k = rank\_x$ then the pivot is kth smallest.
- If $k < rank\_x$ then search $A[p \ldots q - 1]$ recursively.
- If $k > rank\_x$ then search $A[q + 1 \ldots r]$ recursively. Find element of rank $(k - q)$ because we eliminated q smaller elements in A.

Select Algorithm

```
1   SELECT (array A, int p, int r, int k)
2   if (p = r)
3     then return A[p]
4   else x ← CHOOSE PIVOT(A, p, r)
5     q ← PARTITION(A, p, r, x)
6     rank_x ← (q-p+1)
7     if k = rank_x
8       then return x
9     if k < rank_x
10      then return SELECT(A, p, q-1, k)
11    else return SELECT(A, q+1, r, k-q)
```


rank_x = 4
5
9
2
6
Pivot
4
1
3
7
3
1
2
4
6
9
5
7
initial
k = 6
partition
pivot = 4
rank_x = 3
6
9
5
Pivot
7
6
5
7
9
recursion
k = (6-4) = 2
partition
pivot = 7
rank_x = 2
Pivot
6
5
5
6
recursion
k = (2-1+1) = 2
partition
pivot = 6

- In the future classes, we will also study this pivot selection in quick sort algorithm.
- Currently in this selection problem, we are not sorting but discarding the values.