

12 Selection Algorithms

For COMSC 132

Topics

- Selection by sorting
- Randomized selection
- Deterministic selection

Selection

- Given an unordered list
- You can find these things without sorting
 - **Mean** (average)
 - **Mode** (most common element)
- But you need to order the list to find
 - **Median** (the middle-sized element)
 - k^{th} smallest element

Selection by sorting

- Sorting the list makes it easy to find all those items
- But it is more work than necessary
- There are more efficient selection methods

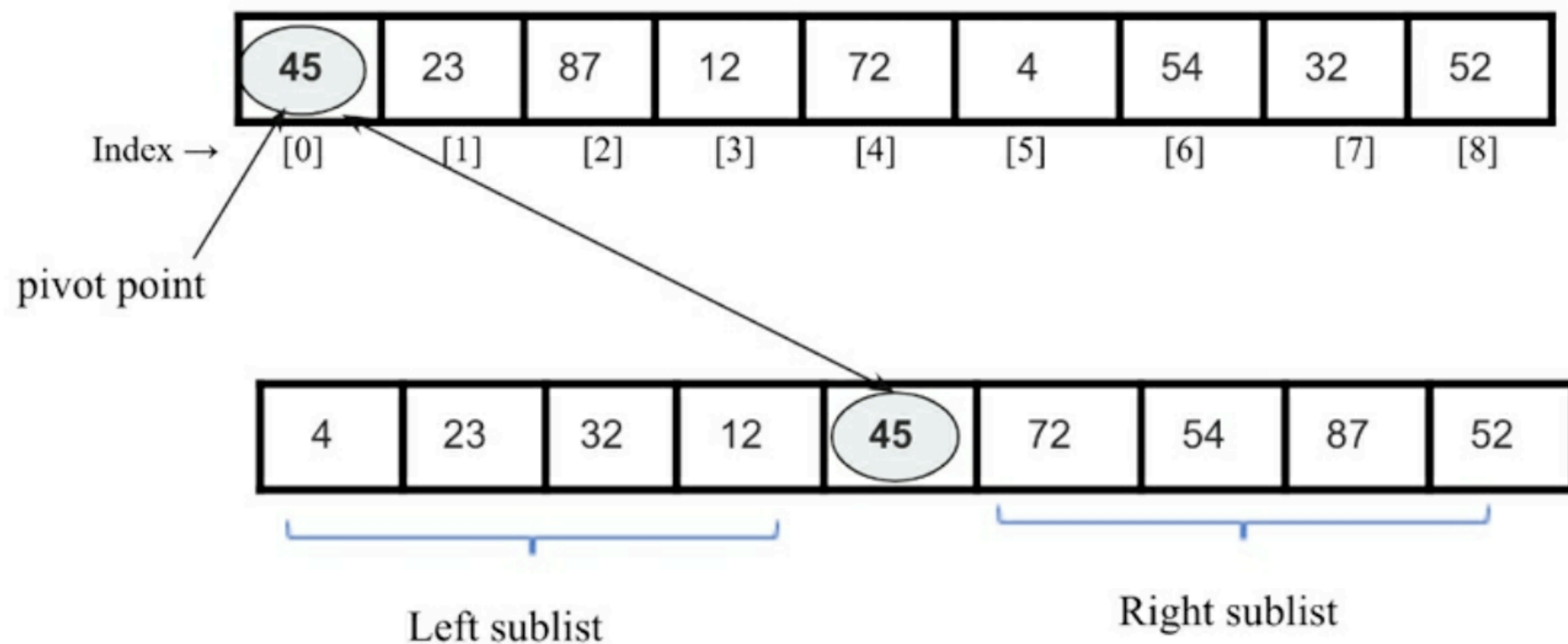
Randomized selection

- Based on *quicksort* algorithm
- Let's review that first

Quicksort

- Divide and conquer
 1. Choose a ***pivot element***
 - Such as the first element
 2. **Partition** the list around the pivot
 - Place smaller elements to its left
 - And larger elements to its right
 3. **Repeat** for the two sublists

Quicksort



- Notice that the **pivot** is already at its correct index
- It won't move any more as sorting proceeds

Randomized selection

- Finds the k^{th} smallest element
 - in an unordered list
- Also called ***quickselect***
 1. Choose a **pivot** and **partition** the elements into two sublists
 2. Compare index of pivot with desired value **k**
 3. Recursively look in appropriate sublist

Randomized selection

- Seeking 3rd smallest element ($k = 2$)

45	23	87	12	72	4	54	32	52
----	----	----	----	----	---	----	----	----

Assume, 45 is the pivot point

4	23	32	12	45	72	54	87	52
---	----	----	----	----	----	----	----	----

After first iteration, 45 is placed at its correct position.

Sub-list with values
>45



4	23	32	12
---	----	----	----

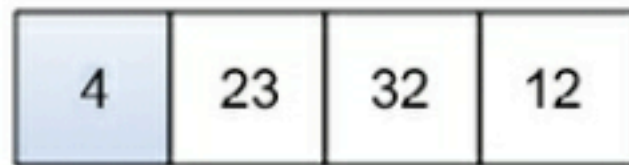
Sub-list with values
>45



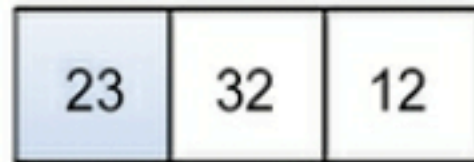
72	54	87	52
----	----	----	----

Now, consider only the left sublist
as the value of $k <$ index of the split point,
i.e. ($2 < 4$)

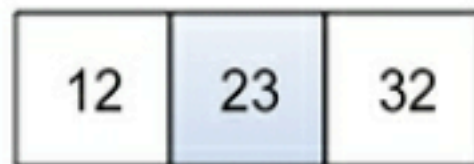
Randomized selection



Assuming 4 as the pivot point.



Now, 4 is placed as its correct position, in other words, at first place.
Now consider the right sublist.



Now considering 23 as the pivot point, after the partitioning, it is placed at its correct position, i.e. at 3rd position, which is required, so it will be returned.

Randomized selection

```
def quick_select(array_list, start, end, k):  
    split = partition(array_list, start, end)  
    if split == k:  
        return array_list[split]  
    elif split < k:  
        return quick_select(array_list, split + 1, end, k)  
    else:  
        return quick_select(array_list, start, split - 1, k)
```

- Worst-case complexity is $O(n^2)$
- When all pivots are at the wrong end, so the list only shrinks by 1 per iteration

Deterministic selection

- Finds the k^{th} smallest element
 - in an unordered list
- Works like randomized selection
- But chooses the pivot more efficiently
- To split the list into two equal halves (approximately)

Deterministic selection

- The **median** would split the list in half best
 - But it would require sorting the list
 - Which is too much work
- So we use **median of medians**
 1. Split list into groups of 5 elements (or 8)
 2. Sort the groups, find medians of each group
 3. Use median of the medians for pivot

Deterministic selection

6	45	23	87	12	72	4	54	32	52	1	34	38	13	57
---	----	----	----	----	----	---	----	----	----	---	----	----	----	----

break the whole list into sublists of 5 elements each.

6	45	23	87	12
---	----	----	----	----

Sort the list



6	12	23	45	87
---	----	----	----	----

Median of this sublist is 23.

72	4	54	32	52
----	---	----	----	----

Sort the list



4	32	52	54	72
---	----	----	----	----

Median of this sublist is 52.

1	34	38	13	57
---	----	----	----	----

Sort the list




1	13	34	38	57
---	----	----	----	----

Deterministic selection

Median of this sublist is 23.

Median of this sublist is 52.



23	52	34
----	----	----

List of median of each
sublist.

23	34	52
----	----	----

Sort the list.

Median of medians list is 34.

Use 34 as the pivot value, and apply the partition algorithm.

Now, we obtain the following list where 34 is placed at its correct position in the list.

6	13	23	1	12	32	4	34	72	52	87	54	38	45	57
---	----	----	---	----	----	---	----	----	----	----	----	----	----	----

Since we wish to obtain the 3rd smallest element, the index of the pivot value is 7 ($2 < 7$), so we recursively run the algorithm on the left sublist.

Deterministic selection

- Worst-case complexity is $O(n)$
- When all pivots are at the wrong end, so the list only shrinks by 1 per iteration

Kahoot!

Ch 12