Module #7 Sorting & Searching

Analysis of algorithm performance time ("complexity") which is important in ~~3 ways~~.

→ TreeSets return sorted, but is it the right way?

SORTING ALGOS:

- Easy algos are slow:-

  Selection sort: many visits, few moves.

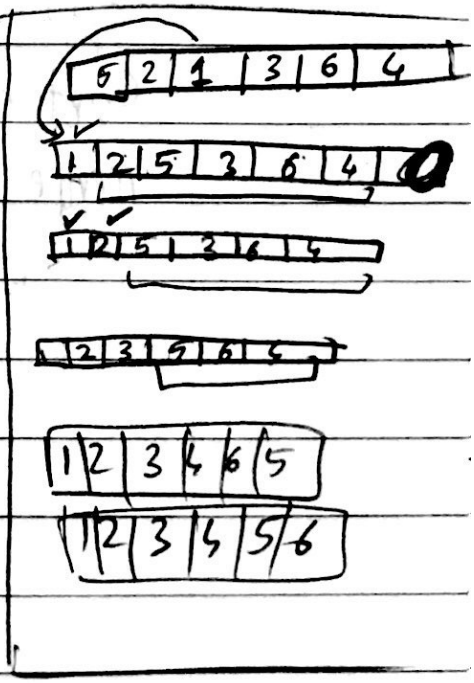
  Insertion sort: many moves, few visits.

SELECTION SORT:-

Swap array members until array is sorted.

find smallest member & put it in a[0].

↳ next smallest in a[1] ...




```
public class SelectionSorter {
    private int[] a;
    public SelectionSorter(int[] a) {
        this.a = a;
    }
    public void sortInPlace() { }
}
```

Complexity analysis:-

1. How will increasing input size affect execution time?

   ↳ time complexity — big-O

Redo complexity analysis

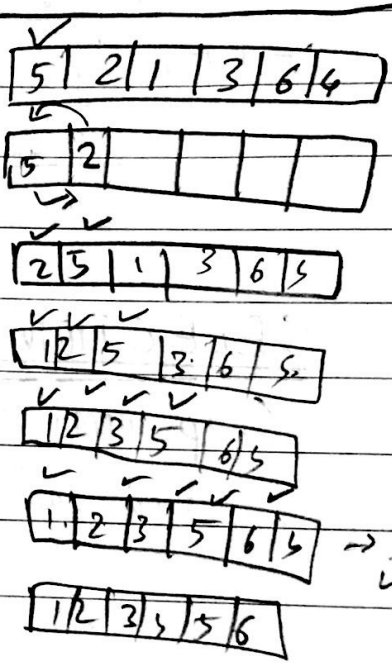
INSERTION SORT:—

→ Assume a[0] ... a[n] are in ascending order.

→

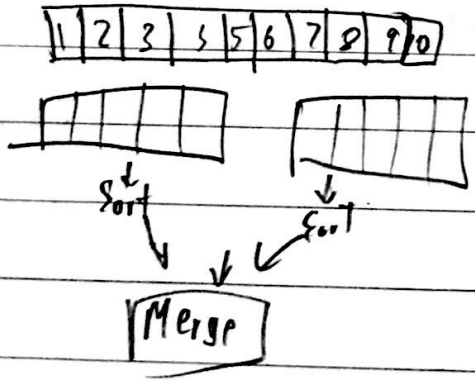
(MORE) Pg 69 and before

first thing we see is sorted:



MERGE SORT:—

Split data in half.
Sort each half using mergesort
merge the halves.



SEARCHING :-

Linear Search :-

```
int findK (int k) {
    for (i=0; i<a.length; i++)
        if (a[i] == k)
            return i
    return -1
}
```

Complexity of Linear search :-

(i) array doesn't contain k :-

$T(n) = n$ visits. $(n = arr.length)$.

$T(n) = n/2$ on avg if arr. contains k.

$T(n) = O(n)$.