Ash Sorting: Easy & Less Time Consuming Sorting Algorithm

Dr. Leena Bhatia

Concept....

- Ash Sorting algorithm is comparison based less time consuming simple algorithm.
- Ash sorting is based on the very simple real life smoke concept that is ***when we burn the coal the smoke which is lighter, fly in the air and the heaviest ash remains at the ground.***

Concept…

- In ash sorting we start from first element and compare it with next element (i.e., 2nd element) as well as with last element.
- Then we put least value at first position, mid value at 2nd position and highest value at the last position.

Lets consider an example:

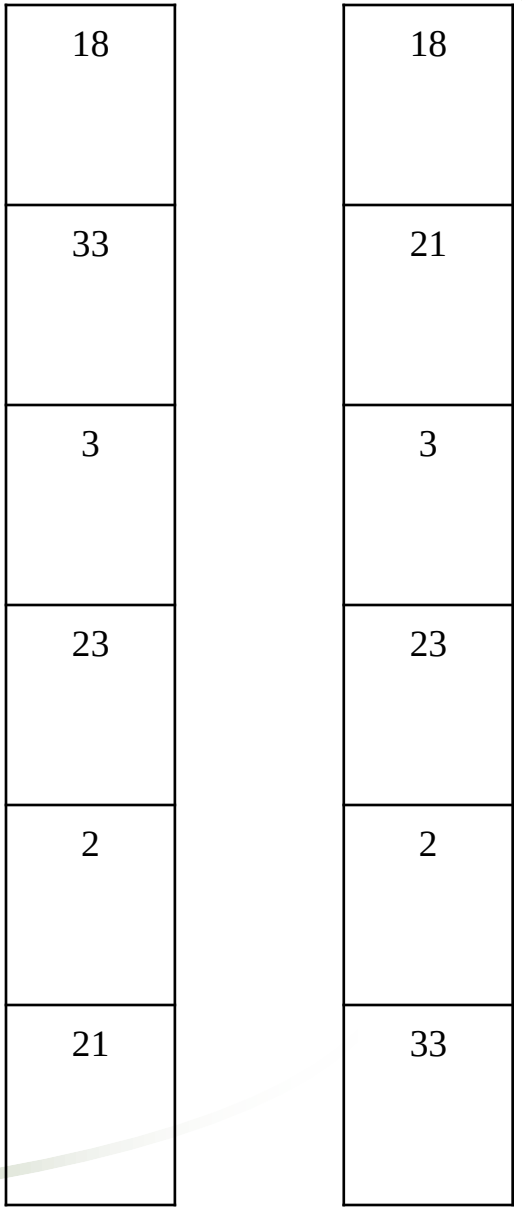
- Take an array of 6 elements
- Compare 1st, 2nd and 6th elements and put 18 at 1st position, 21 at 2nd and 33 at last position

Now, the 1st element will be compared with next element (i.e., 3rd). After comparison there might be three basic options:

Case I: 1st element >3rd element

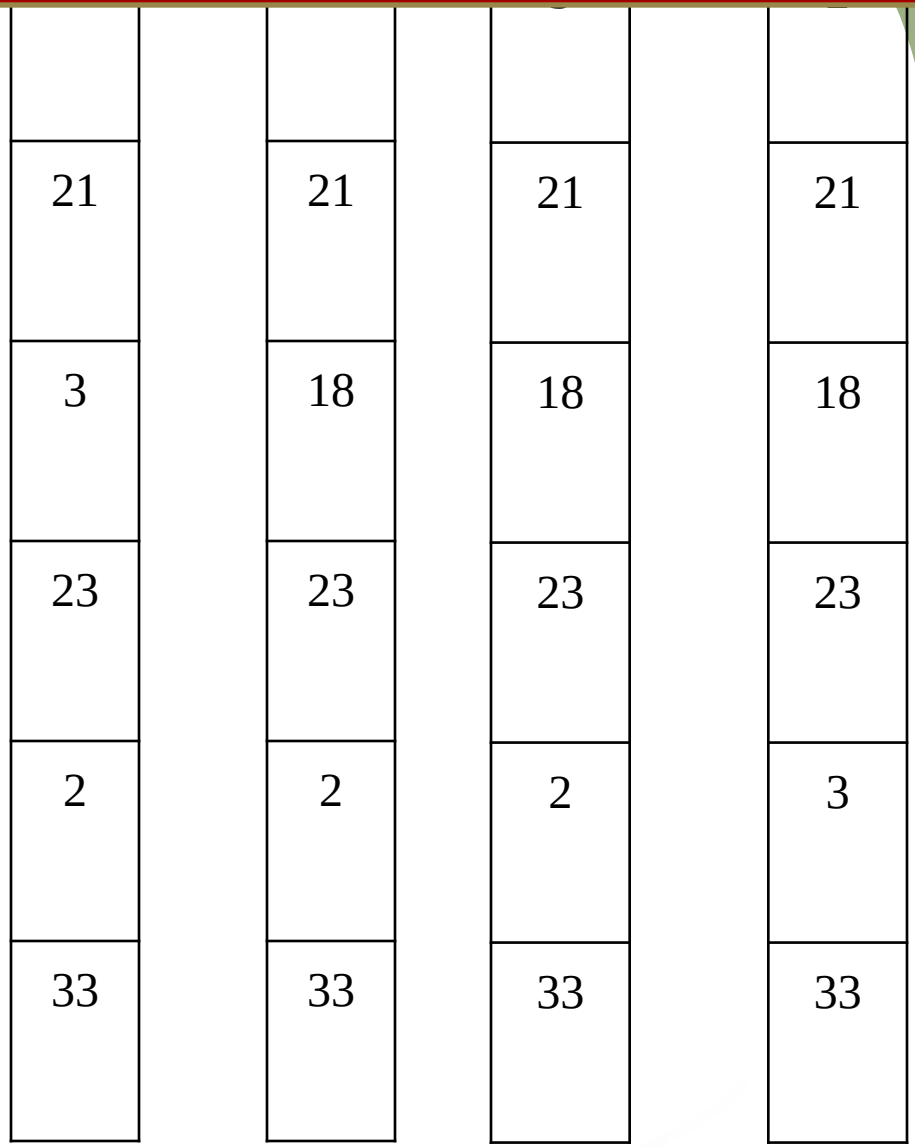
Case II: 3rd element >1st element

Case III: both are equal



- In **Case I**, if 1st element >3rd element, there is no need to compare the 3rd element with last element (as last element is already greater to 1st element) and just swapping of 1st element with 3rd element is required.
- But in **Case II**, 3rd element must also be compare with last element) as it could be greater than last element). If the 3rd element is also greater than the last element then we have to swap the values of 3rd element and last element.
- In the last case i.e., **Case III** no swapping or further comparison is required.

- **N**ow the 1st element will be compared with 3rd, 4th and 5th one by one in the manner explained earlier.
- **A**fter I pass, least and highest elements will be placed at correct positions



- **II pass**: At the start of II pass, the same procedure will be followed with 2nd, 3rd and second last element i.e., 5th in the present example (Fig: f) and the values are 21, 18 and 3.
- After the first comparison, 2nd position will be occupied by 3, 3rd position will be occupied by 18 and 21 will be stored at 5th position.

2
21
18
23
3
33

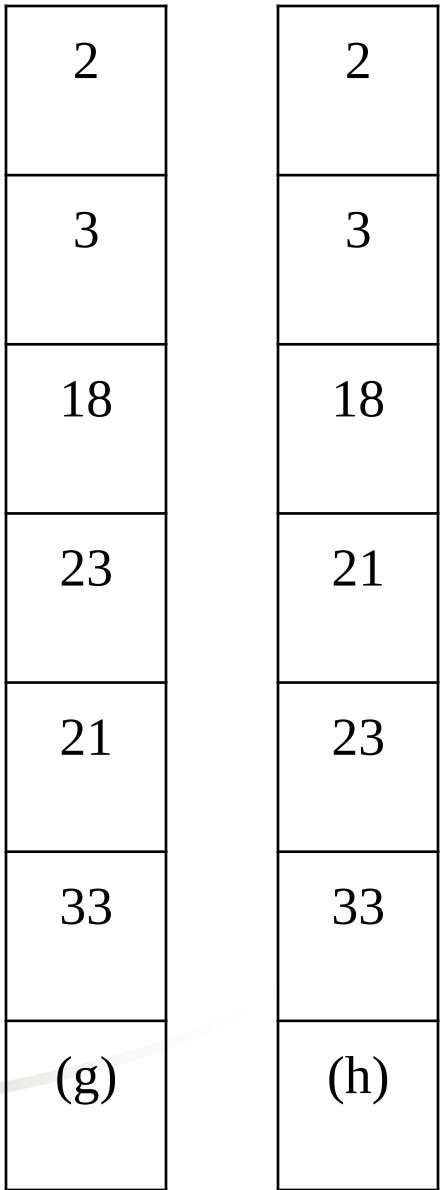
(f)

2
3
18
23
21
33

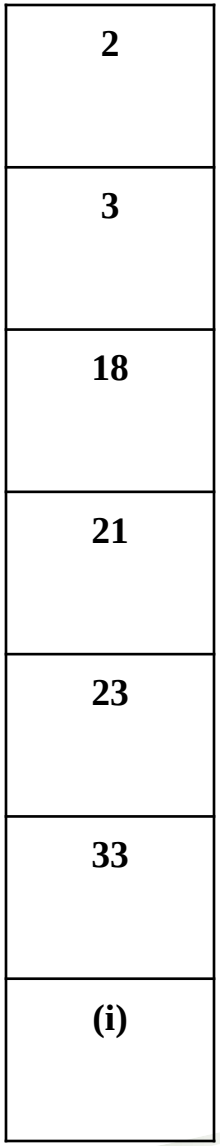
(g)

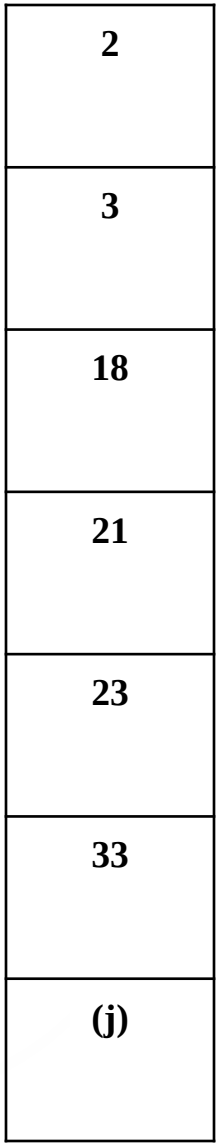
- **N**ext comparison would be among 2nd, 4th and 5th elements (Fig: g). As 3 (2nd element) is less than 23 (4th element) that's why 23 will also be compared with 5th element i.e, 21 (Case II). And swapping would be performed between 4th and 5th elements.

II pass is now completed and after this 2nd least and 2nd highest elements will be placed at correct positions (Fig: h).



- **III pass**: in the third pass only 3rd and 4th elements would be compared (Fig: i) and positioned at correct places.
- ***After III pass all the elements get sorted and placed at right positions (Fig j).***



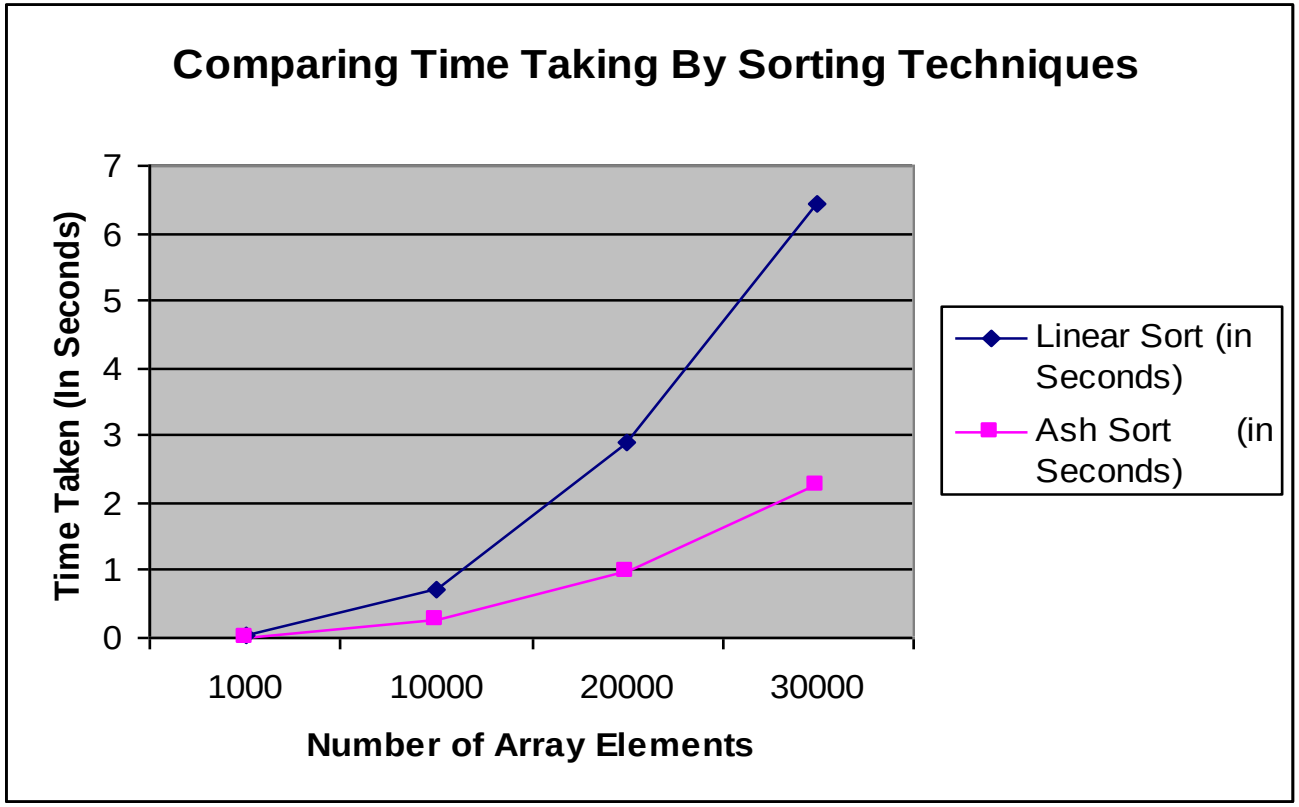


Procedure Ash (Array arr, Number initial_index, Number lst)

```
Begin
  For i= initial_index to lst/2
  Begin
     Flag=0
     For j=initial_index+1 to lst-1
     Begin
        If flag=0 then
         Sort (arr[i],arr[j], arr[lst])
         flag=1
       else
           if arr[i]>arr[j] then      Rem  Case I
              tmp=arr[i]
              arr[i]=arr[j]
              arr[j]=tmp
         else
           if arr[j]>arr[lst] then         Rem: Case II
              tmp=arr[j]
              arr[j]=arr[lst]
              arr[lst]=tmp
           end if
        end if
    end if
end loop
        lst=lst-1
    end loop
end procedure
```

Real Time Comparison between Linear Sort and Ash Sort

No of elements in array → Mean Time Taken in Sorting ↓	1000	10000	20000	30000
Linear Sort (in Seconds)	0.025	0.72	2.9	6.43
Ash Sort (in Seconds)	0	0.28	0.99	2.27



Graph 1: Comparing Time Taken By Sorting Techniques

Conclusion…

- Though Ash sorting is based on comparison but it needs less number of comparisons as compare to linear sort.
- Ash sorting is found 2 to 3 times faster as compared to linear sort.

Thank You All