Design and Analysis of Algorithms

Lecture – 16
Non Comparison based Sorting algorithms

Success is always inevitable with Hard Work and Perseverance

N. Ravitha Rajalakshmi

Learning Objective

- Comparison based Sorting algorithm and their complexity
- Linear time Sorting algorithms

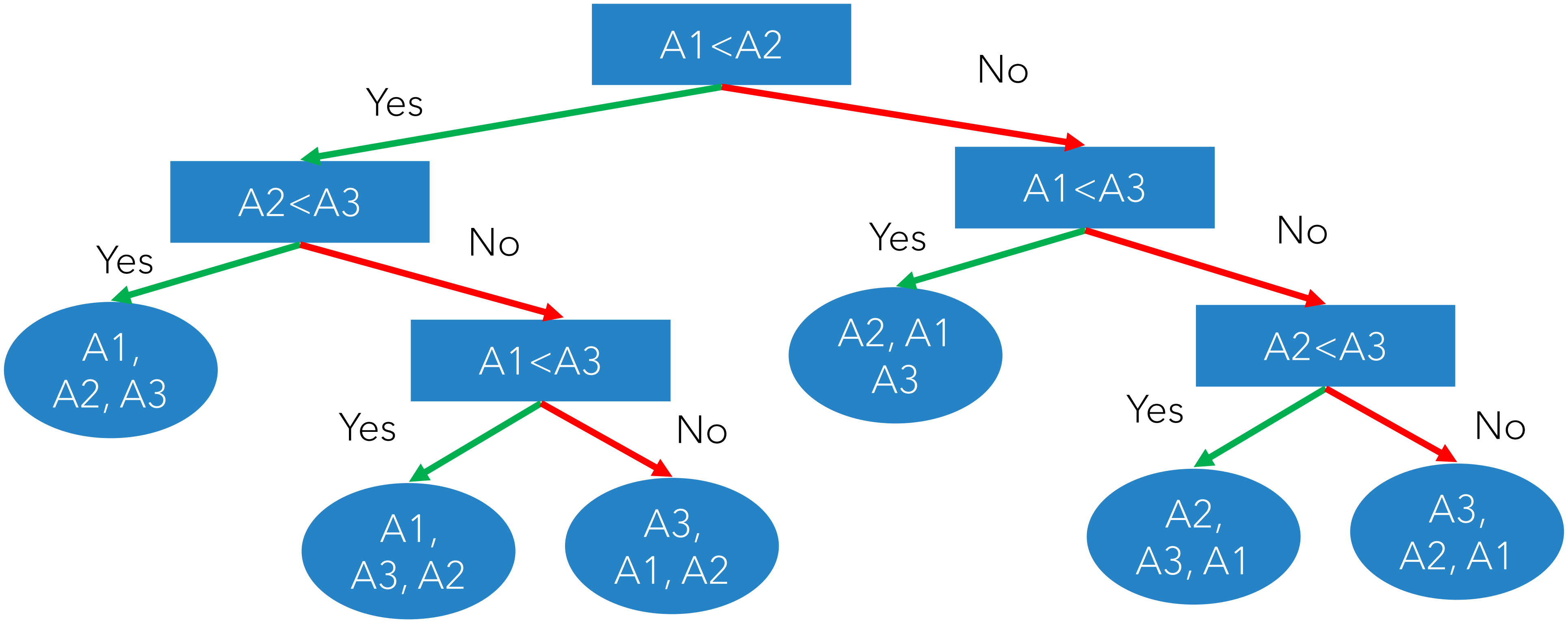
Comparison based sorting algorithm

- Sorts objects by comparing pairs of them
- Ex: Selection sort , Merge Sort

Any comparison based sorting algorithm at least takes $\Omega(n \log n)$ comparisons to sort n objects

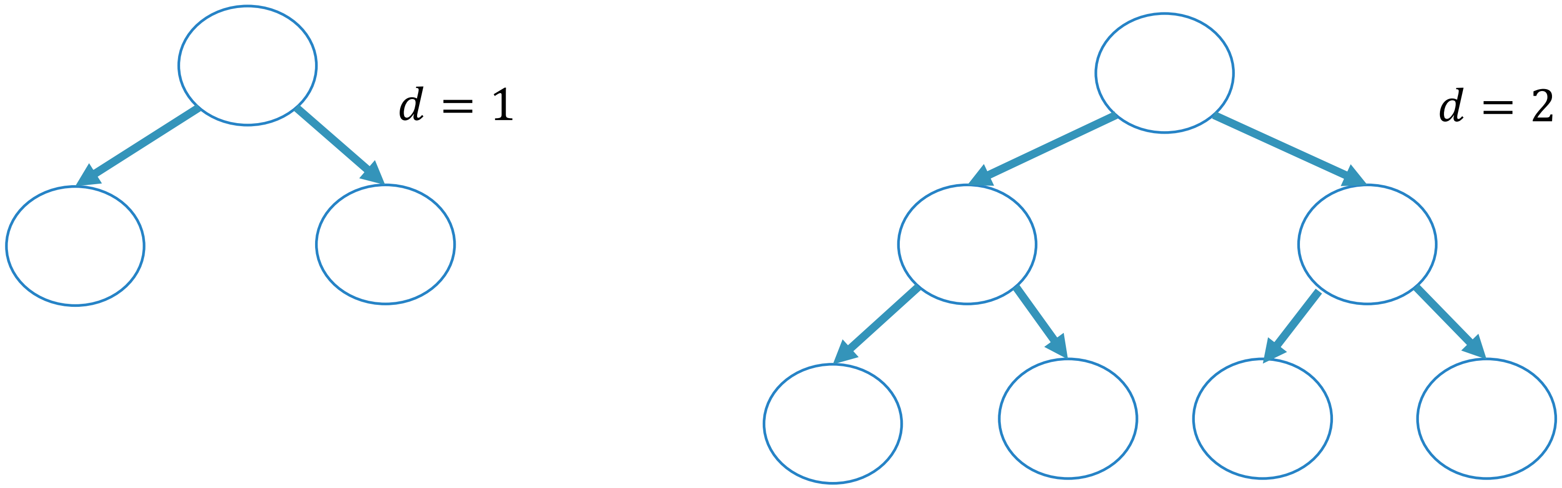
Why ??

Visualization – Decision Tree



- Maximum number of comparisons is based on depth of the tree
- All we know about the tree is the number of leaves
- Number of leaves = Number of permutations

- For n elements , the number of permutations = n!
- Is it possible to relate depth and number of leaves in a binary tree?



- $d = 2^l$, l denote the number of leaves
- $l = \log(d)$
- Number of leaves = Number of permutations

$$l = \log(n!)$$

$= \log(1 * 2 * 3 \cdot \cdots n)$

$= \log(1) + \log(2) + \log(3) + \cdots \cdot \log(n)$

$\geq \log({}^{n}/_{2}) + \cdots \log(n)$

$\geq {}^{n}/_{2} \log({}^{n}/_{2})$

$\Omega(n \log n)$

Non-Comparison based sorting

- Imposes constraint over the input instance
- Counting sort
  - Frequency of occurrence of elements (Key)
- It is expected to know the range of array elements beforehand
- Smallest number of integers

4	2	3	1	5	3	4

Range of elements is known
1 - 5

1	2	3	4	5	Elements
1	1	2	2	1	Count
1	2	4	6	7	Cumulative Count

Does the Count tells you anything about its position ?

Pause & Think

- Index of elements

Index	0	1	2	3	4	5	6
Original Array	4	2	3	1	5	3	4
Sorted Array	1	2	3	3	4	4	5

1 – 0

2 – 1

3 – 2, 3

4 – 4, 5

5 - 6

Is there any relation between the cumulative count and index of element in sorted array?

4	2	3	1	5	3	4

For every element, use cumulative count and place it in correct index position

1	2	3	4	5	Elements
1	1	2	2	1	Count
0	1	2	4	6	Cumulative Count

0	1	2	3	4	5	6
1	2	3	3	4	4	5

Function Counting_Sort(A, n)

```
Count[k] = {0} B[0….n]= {}
# Populate frequency
for i  in range(0,n)
        Count[A[i]]+=1
#Compute Cumulative Frequency
for i in range(1,k)
        Count[i] = Count[i] + Count[i-1]
for i in range(0,n)
        m = A[i]  # Element
        B[Count[m]-1] = m  # Place element in its correct position
        Count[m]-=1 #Decrement Count
```

Time Complexity

- Assumption : Let the array contains elements from 1 to k
- Basic Operation : Addition and Assignment
- Input Size: n (number of elements in the array)
- Time Complexity = n + k + n = $0(n + k)$

If the value of k $\leq \mathrm{n}$, then the time complexity = $O(n)$

Sorting Algorithm	Best	Average	Worst	Memory	Stable
Merge	$n \log n$	$n \log n$	$n \log n$	n	Yes
Quick	$n \log n$	$n \log n$	n^2	$\log n$	No
Heap	$n \log n$	$n \log n$	$n \log n$	1	No
Selection	n^2	n^2	n^2	1	Yes
Counting	$n + k$	$n + k$	$n + k$	$n + k$	No

Summary

- Discussed Linear time sorting algorithm

Thank You
Happy Learning

Success is always inevitable with Hard Work and Perseverance