Design and Analysis of Algorithms

Lecture-14

Dharmendra Kumar (Associate Professor)
Department of Computer Science and Engineering
United College of Engineering and Research,
Prayagraj

Stable sorting

A sorting algorithm is said to be stable if two objects with equal keys appear in the same order in sorted output as they appear in the input array to be sorted.

Some stable sorting algorithms

1. Counting sort
2. Insertion sort
3. Merge Sort
4. Bubble Sort

Some unstable sorting algorithms

1. Selection sort
2. Quicksort
3. Heap sort

Radix sorting

- This sorting algorithm assumes all the numbers must be of equal number of digits.

- This algorithm sort all the elements on the basis of digits used in the number.

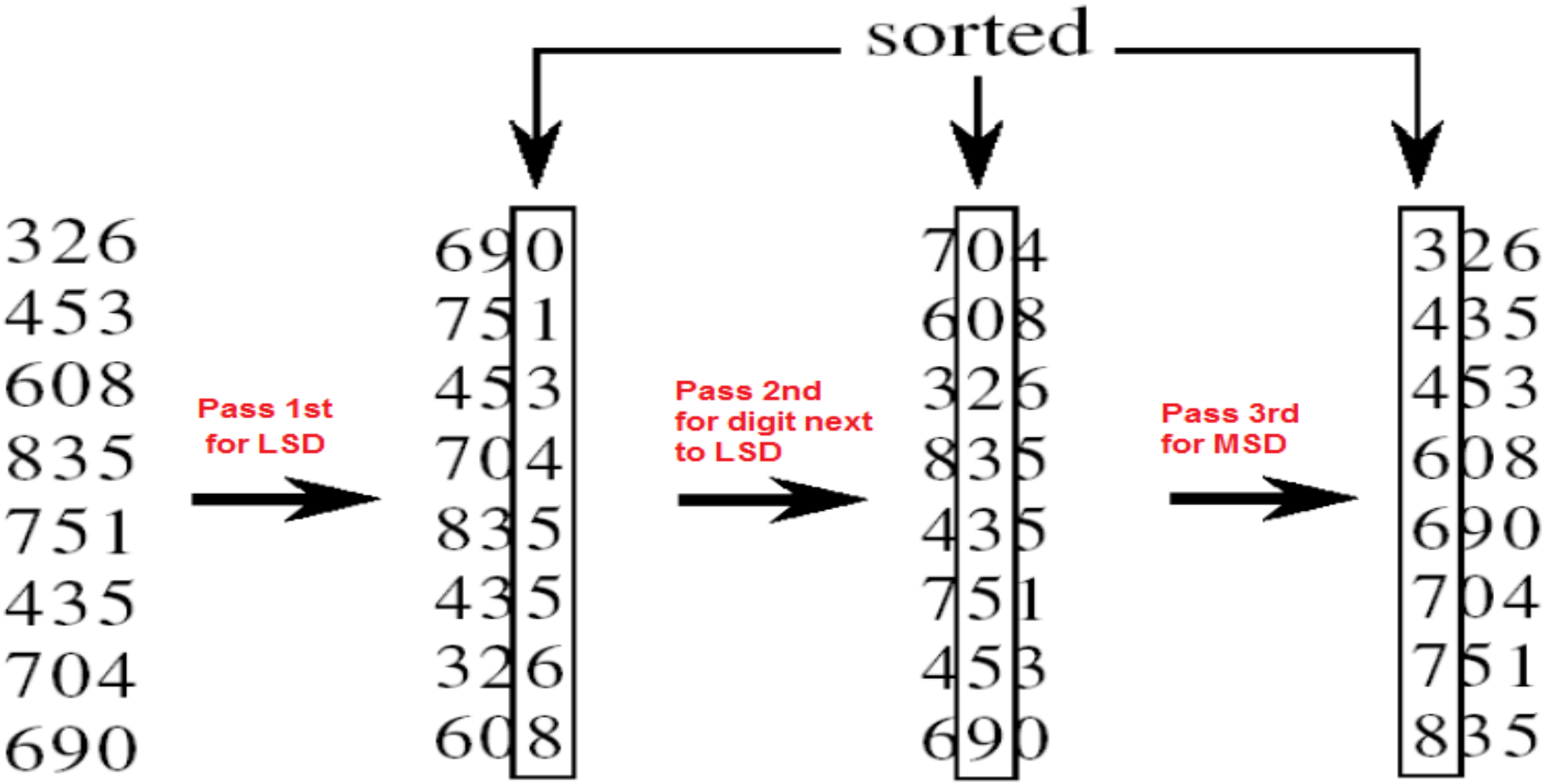
- First sort the numbers on the basis of least significant digit. Next time on the basis of 2^{nd} least significant digit. After it on the basis of 3^{rd} least significant digit. And so on. At the last, it sort on the basis of most significant digit.

Radix sorting

Example: Sort the following elements using radix sort

326, 453, 608, 835, 751, 435, 704, 690.

Solution: In these elements, number of digits is 3. Therefore, we have to used 3 iterations.



Radix sorting

```
1  radix_sort(A,d,k)
2  {
3      for i=1 to d
4          counting_sort(A,i,k)
5  }
```

COUNTING-SORT(A, B, k)

```
1   let C[0..k] be a new array
2   for i = 0 to k
3       C[i] = 0
4   for j = 1 to A.length
5       C[A[j]] = C[A[j]] + 1
6   // C[i] now contains the number of elements equal to i.
7   for i = 1 to k
8       C[i] = C[i] + C[i − 1]
9   // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] − 1
```

Radix sorting

Time complexity of radix sort is

$$T(n) = \theta(d(n+k))$$

Note: When d is constant and k=O(n), we can make radix sort run in linear time i.e $T(n) = \theta(n)$.

Bucket sorting

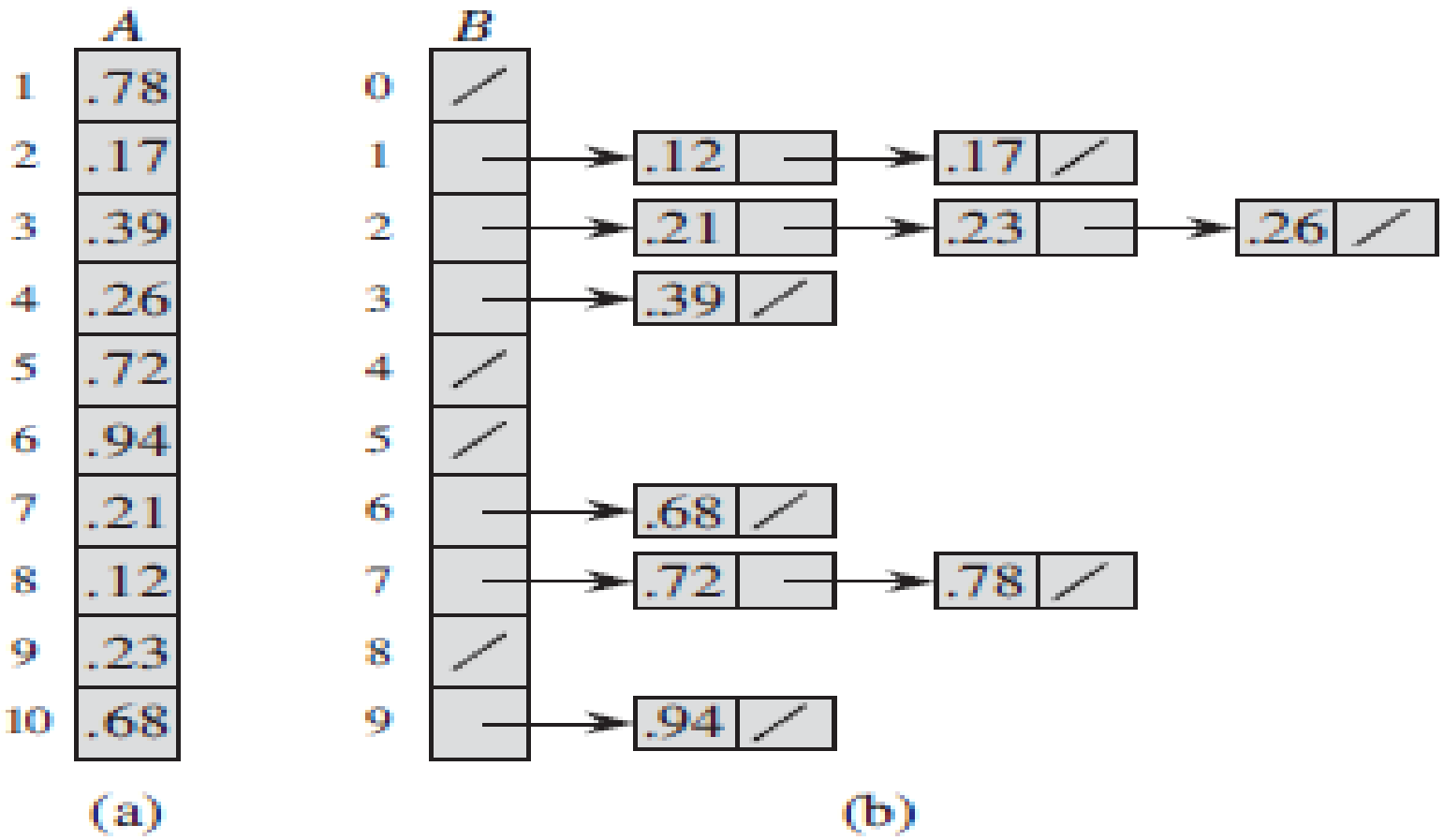
- Bucket sort assumes that the input is generated by a random process that distributes elements uniformly and independently over the interval [0,1).

- Bucket sort divides the interval [0,1) into n equal-sized subintervals, or ***buckets***, and then distributes the n input numbers into the buckets. Since the inputs are uniformly and independently distributed over [0,1), we do not expect many numbers to fall into each bucket. To produce the output, we simply sort the numbers in each bucket and then go through the buckets in order, listing the elements in each.

Bucket sorting

Example: Sort the following elements using bucket sort

0.78, 0.17, 0.39, 0.26, 0.72, 0.94, 0.21, 0.12, 0.23, 0.68



Bucket sorting

```
BUCKET-SORT(A)
1  let B[0..n − 1] be a new array
2  n = A.length
3  for i = 0 to n − 1
4       make B[i] an empty list
5  for i = 1 to n
6       insert A[i] into list B[⌊nA[i]⌋]
7  for i = 0 to n − 1
8       sort list B[i] with insertion sort
9  concatenate the lists B[0], B[1], . . . , B[n − 1] together in order
```

Time complexity $T(n) = \theta(n)$

Linear search

- It is a simple search algorithm that search a target value in an array or list.
- It sequentially scan the array, comparing each array item with the search element.
- If search element is found in an array, then the index of an element is returned.
- The time complexity of linear search is O(n).
- It can be applied to both sorted and unsorted array.

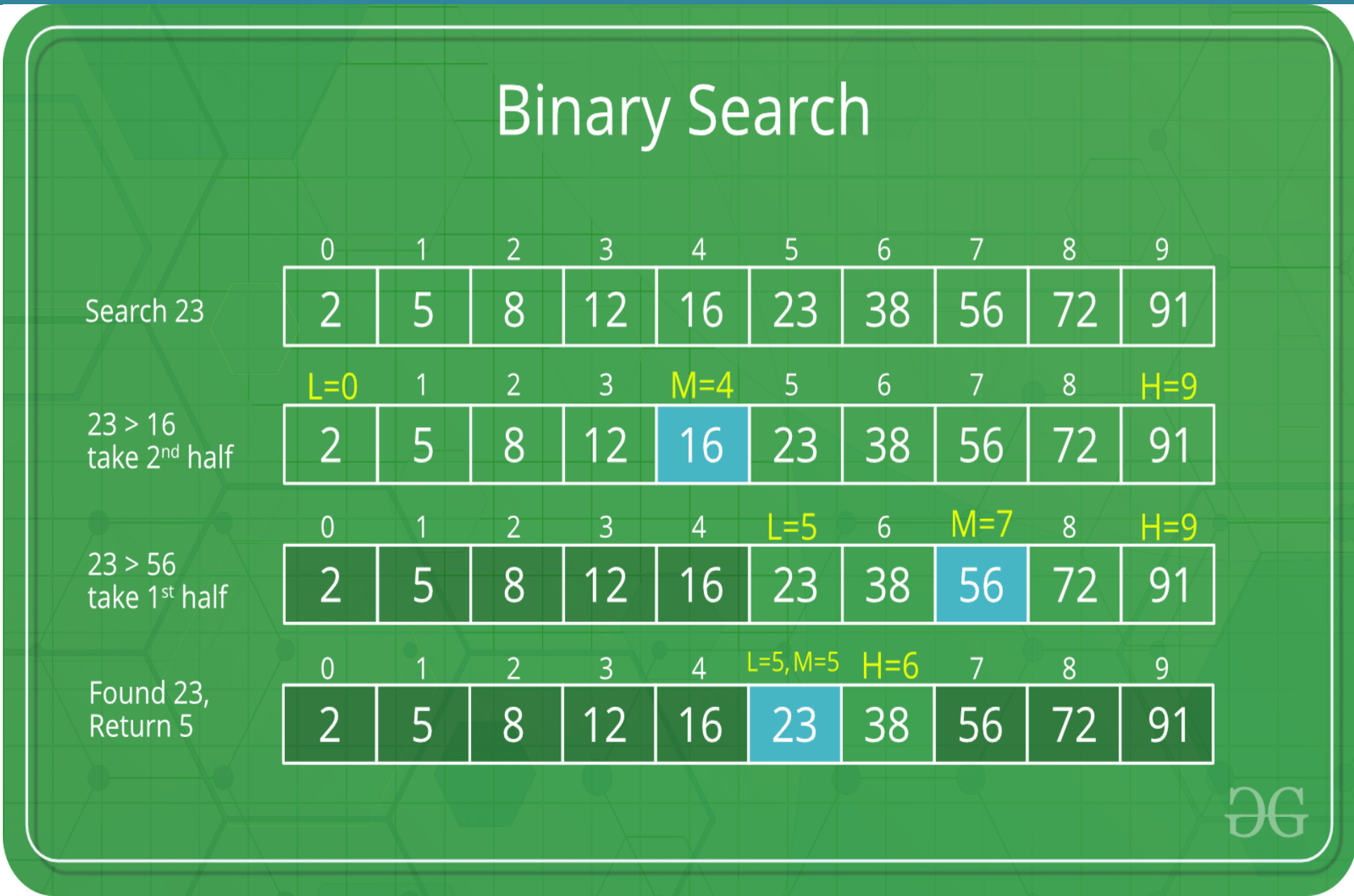
Binary search

- **Binary search** is the most popular Search algorithm. It is efficient and also one of the most commonly used techniques that is used to solve problems.

- Binary search works only on a sorted set of elements. To use binary search on a collection, the collection must first be sorted.

- When binary search is used to perform operations on a sorted set, the number of iterations can always be reduced on the basis of the value that is being searched.

Binary search process



Binary search

```
Binary-search(A, n, x)
l = 1
r = n
while l ≤ r
do
        m = ⌊(l + r) / 2⌋
        if A[m] < x then
                l = m + 1
        else if A[m] > x then
                r = m − 1
        else
                return m
return unsuccessful
```

Time complexity **T(n) = O(lgn)**

AKTU Previous Year Questions

1. How analyze the performance of an algorithm in different cases?
2. Derive the time complexity of Merge sort.
3. Solve the recurrence i) T (n) =3T (n/4) + cn2 using recursion tree method. ii) T (n) = n + 2T (n/2) using Iteration method. (Given T(1)=1)
4. Write Merge sort algorithm and sort the following sequence {23, 11, 5, 15, 68, 31, 4, 17} using merge sort.
5. What do you understand by stable and unstable sorting? Sort the following sequence {25, 57, 48, 36, 12, 91, 86, 32} using heap sort.
6. What is recurrence relation? How is a recurrence relation solved using Master's theorem?
7. What is asymptotic notation? Explain Omega(Ω) notation.
8. Write an algorithm for counting sort. Illustrate the operation of counting sort on the following array: A = {4,0,2,0,1,3, 5,4,1,3,2,3}.
9. Solve the following recurrence relations:-
   (a) $T(n) = T(n-1)+n^4$
   (b) $T(n) = T(n/4)+T(n/2) + n^2$
10. Write an algorithm for insertion sort? Find the time complexity in all the case.

Thank you.