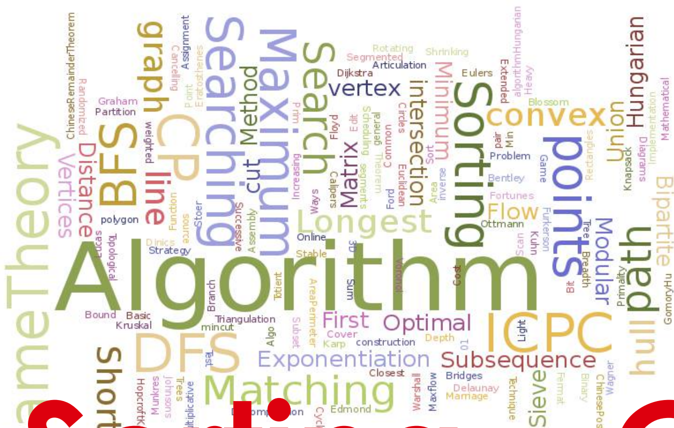
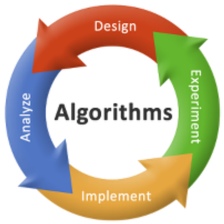


Sorting – Counting Sort, Radix Sort & Best ways



Course: Algorithms

Faculty: Dr. Rajendra Prasath

Autumn 2018

Sorting – Counting Sort, Radix Sort & Best Ways

This lecture covers two sorting algorithms for sorting a set of n elements. The first one is the Counting sort and the second one is the Radix Sort. We provide illustrations and the complexity analysis of these two algorithms. We also discuss the Criteria for choosing a Sorting Algorithm

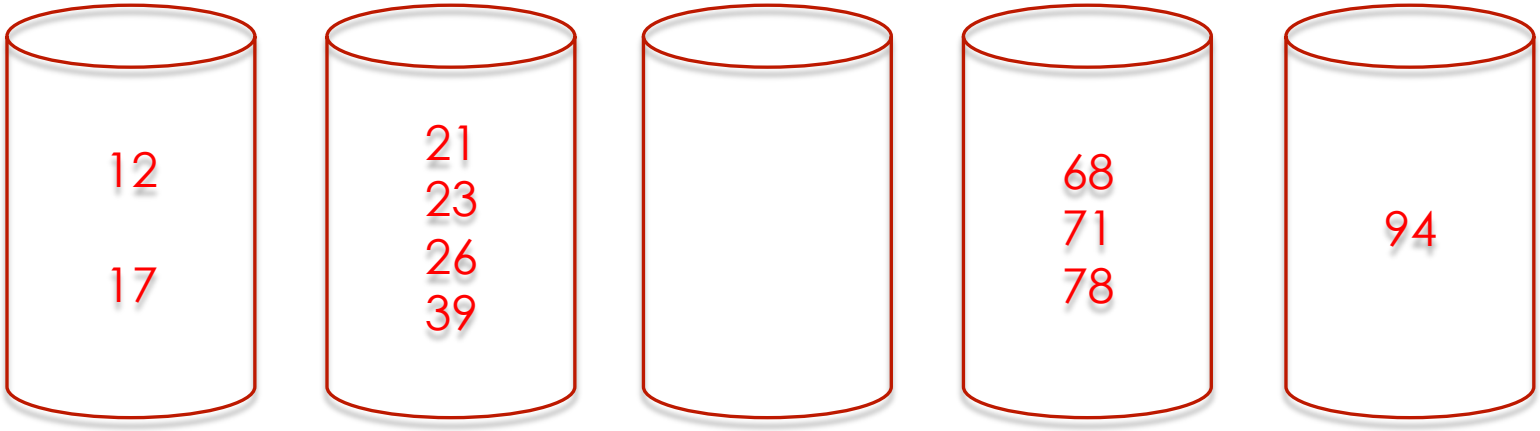
Recap: Sorting Algorithms

- Suggest a simple algorithm for Sorting n elements
  - **Correctness:** First Test whether will the algorithm work for a small set of the input? Then apply on a bigger set
  - **Choice of the Data Structures:**
    - Choose a suitable data structure
  - **Perform complexity analysis**
    - How much space and running time required in the worst case?
  - **Adaptability:**
    - Is the solution adaptable with the growing size of the input n?
- How to get the tight bound of the sorting algorithm in terms of the running time?

Recap: Bucket Sort – Example

- **Input sequence to be sorted:**

78	17	39	26	71	94	21	12	23	68



- **Sorted Sequence:**

12	17	21	23	26	39	68	71	78	94

Recap: Heap Property

- A HEAP is a complete binary tree
  - Has a smallest possible height – How?
  - A heap with n nodes ➔ O(log n) height

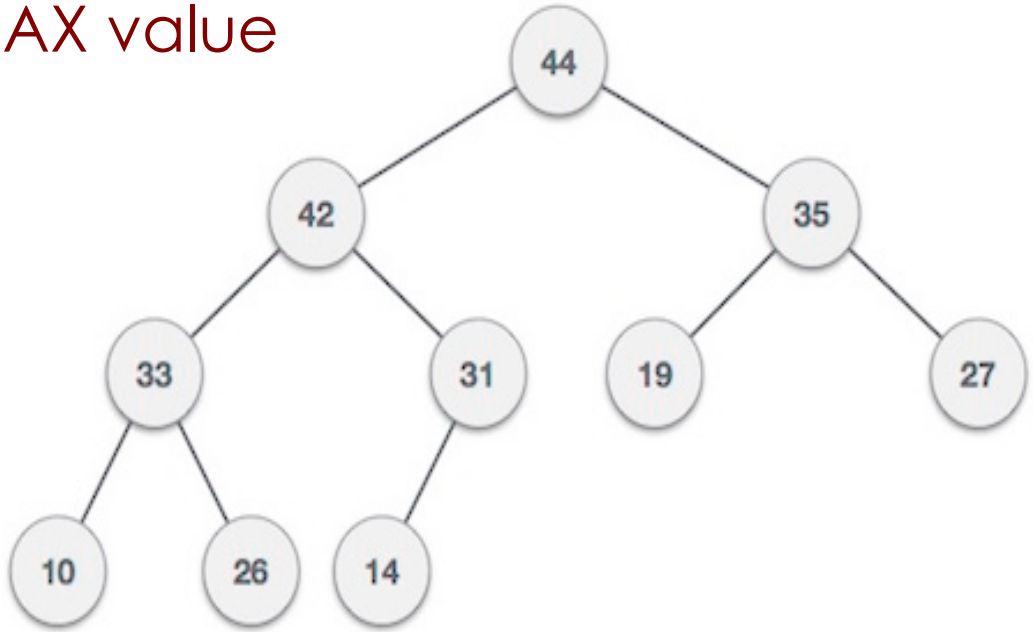
- Heap Property ?
  - To order elements in the heap
  - How to define Heap Property ?
  - Is ordering WELL – DEFINED??

- Two Types:
  - min-heap property
  - max-heap property

Max-Heap

- Max-Heap Property ?
- Value of each node

  ≤ value of its parent.
- Root ➔ MAX value



Counting Sort

- Recall:
  - Bucket Sort algorithm

- What are the issues in Bucket Sort?
  - Skewed Distributions of elements
  - Preserving the ordering of hash generated to handle buckets

- Is there any alternative approach using the concept of buckets?

Counting Sort - Algorithm

- Recall

Counting Sort – Basic Idea

- Counting sort assumes that each of the n input elements is an integer in the range 0 to k, for some integer k.
- When k = $O(n)$ the sort runs in $\Theta(n)$ time

- Two Arrays are required
  - One to hold the sorted output
  - Another array is used as a temporary working storage space
  - Several elements may have same value
    - Recall that the number of comparisons will not reduce in comparison based sorting !!
    - How to handle them efficiently?

Counting Sort – Algorithm

```
COUNTING-SORT(A, B, n, k)
1.  for i ← 0 to r                                  } O(r)
2.      do C[ i ] ← 0
3.  for j ← 1 to n                                  } O(n)
4.      do C[A[ j ]] ← C[A[ j ]] + 1
5.          C[i] contains the number of elements equal to i
6.  for i ← 1 to r                                  } O(r)
7.      do C[ i ] ← C[ i ] + C[i -1]
8.          C[i] contains the number of elements ≤ i
9.  for j ← n downto 1                              } O(n)
10.     do B[C[A[ j ]]] ← A[ j ]
11.         C[A[ j ]] ← C[A[ j ]] - 1
```

Overall time: O(n + r)

Counting Sort – An Example

- How to sort the following elements?

	1	2	3	4	5	6	7	8
A	2	5	3	0	2	3	0	3

- What is the range of elements (??)
- Count the number of occurrences of each element and put them in the auxiliary array

- The Sorted Sequence of elements

	1	2	3	4	5	6	7	8
B	0	0	2	2	3	3	3	5

Counting Sort – Analysis

- An integer Sorting algorithm
  - Why ??

- Complexity
  - Overall running time: O(n + r)
  - when r = O(n) ⇒ running time is O(n)

- Constraints?
  - The range of input data is not significantly greater than the number of objects to be sorted

Radix Sort

- Represents keys as d-digit numbers in some base-k

$$key = x_1 x_2 \ldots x_d \quad \text{where } 0 \le x_i \le k-1$$

- Example: key=15

$key_{10} = 15$, d=2, k=10 where $0 \le x_i \le 9$

$key_2 = 1111$, d=4, k=2 where $0 \le x_i \le 1$

Radix Sort – Basic Idea

Assumptions:

- d=O(1) and k =O(n)

- Sorting looks at one column at a time
- For a d digit number, sort the least significan digit first
- Continue sorting on the next least significan digit, until all digits have been sorted
- Requires only d passes through the list

326
453
608
835
751
435
704
690

Radix Sort – Algorithm

- Algorithm: RADIX-SORT(A, d)

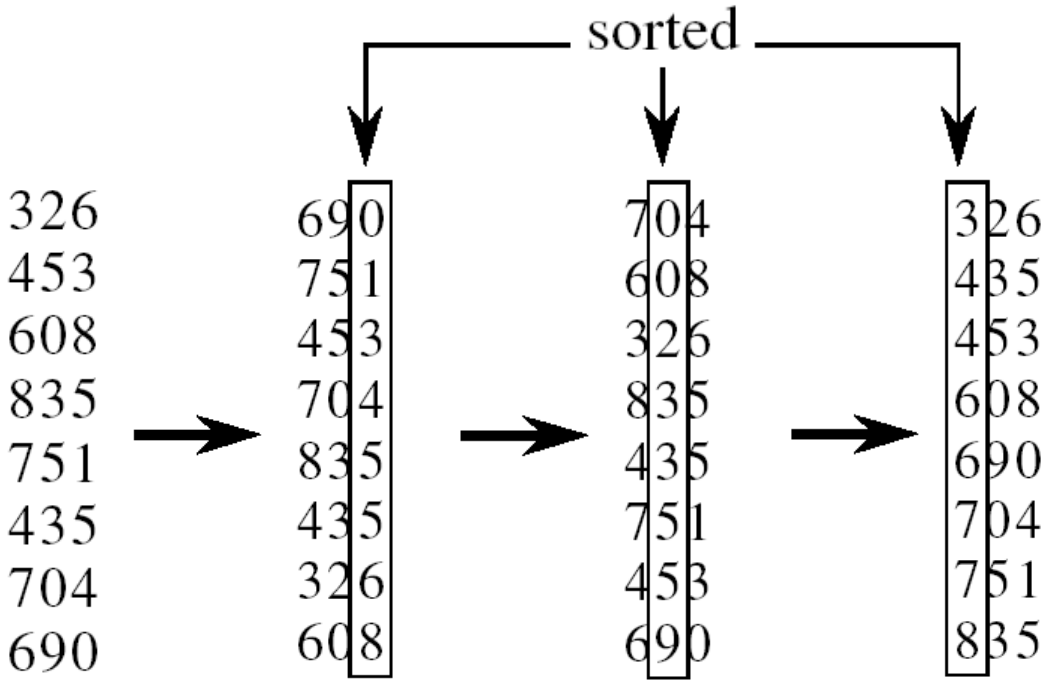
  for i ← 1 to d

  do use a stable sort to sort A on digit i

- Remember:
  - Stable sort preserves the order of identical elements

Radix Sort – An Example

- Let us consider the following example:



Radix Sort – Analysis

- Given n numbers of d digits each, where each digit may take up to k possible values
- RADIX-SORT correctly sorts the numbers in O(d(n+k)) time
- One pass of sorting per digit takes O(n+k) assuming that we use counting sort
- There are d passes (for each digit)
- Assuming d=O(1) and k=O(n), running time is O(n)

Help among Yourselves?

- **Perspective Students** (having CGPA above 8.5 and above)
- **Promising Students** (having CGPA above 6.5 and less than 8.5)

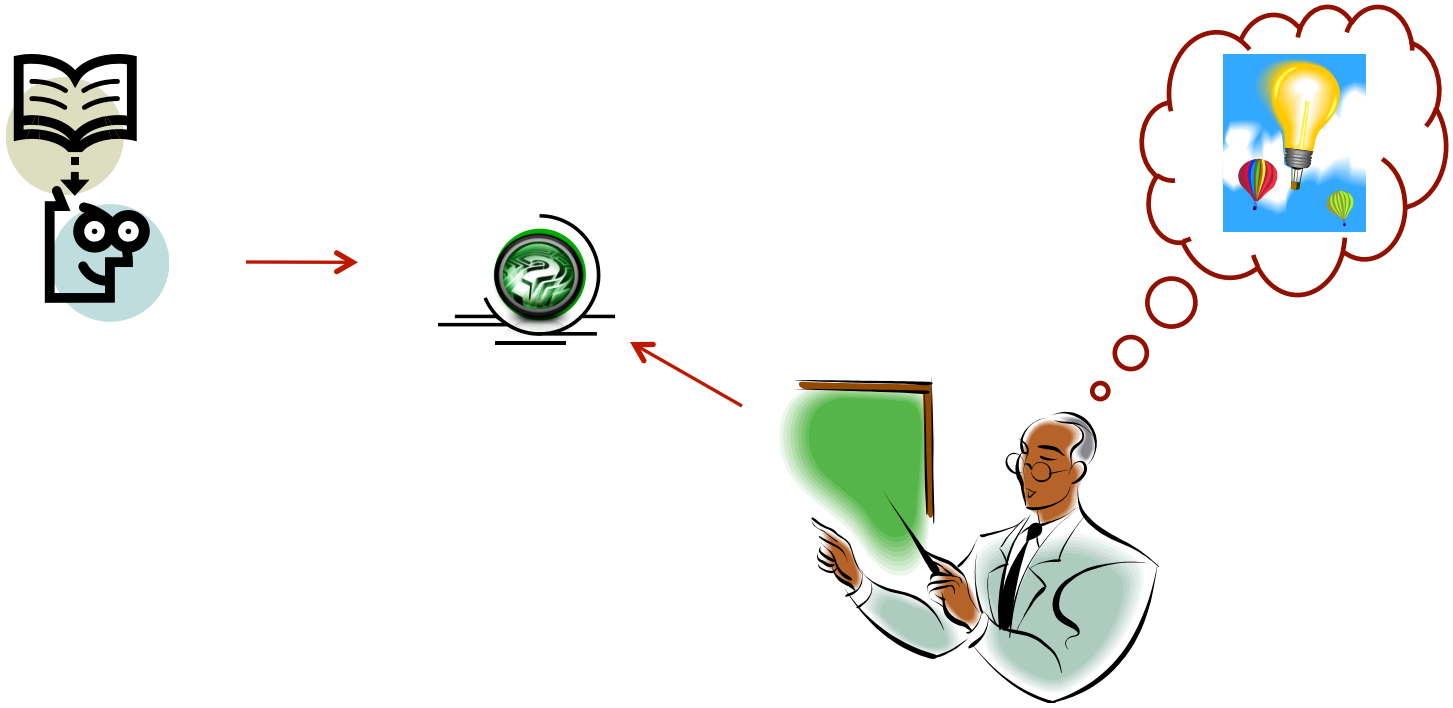
- **Needy Students** (having CGPA less than 6.5)
  - Can the above group help these students? (Your work will also be rewarded)

- You may grow a culture of **collaborative learning** by helping the needy students

Assistance

- You may post your questions to me at any time
- You may meet me in person on available time or with an appointment
- TA s would assist you to clear your doubts.
- You may leave me an email any time (email is the best way to reach me faster)

Thanks …



… Questions ???