LECTURER: TAI LE QUY

ALGORITHMS, DATA STRUCTURES, AND PROGRAMMING LANGUAGES

TOPIC OUTLINE

TOPIC OUTLINE

UNIT 4

BASIC ALGORITHMS

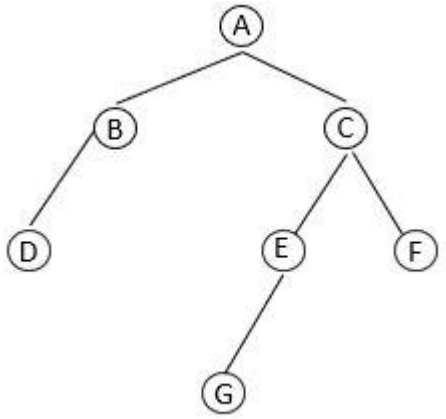
STUDY GOALS

- Understand algorithms for traversal of trees.
- Understand basic algorithms for searching.
- Differentiate between various algorithms for sorting.
- Understand the trie data structure.
- Know about hashing techniques in solving search problems.
- Understand fundamental algorithms for pattern recognition.

EXPLAIN SIMPLY

1. Explain what you understand by linear search and binary search.
2. Explain two different ways that data may be stored in a trie.
3. Describe how you may find out which words from a list of words you have written down occur in a text book using Hash Tables.

LINEARIZATION OF TREES



PREORDER TRAVERSAL

Visit root, recursively visit left subtree, recursively visit right subtree.
Order of nodes: A, B, D, C, E, G, F.

INORDER TRAVERSAL

Recursively visit left subtree, visit root, recursively visit right subtree.
Order of nodes: D, B, A, G, E, C, F

TREE TRAVERSAL ALGORITHMS

Visit nodes of a rooted tree in specific order. Order of nodes visited by different traversal algorithms for the tree in the figure are shown.

POSTORDER TRAVERSAL

Recursively visit left subtree, recursively visit right subtree, visit root.
Order of nodes: D, B, G, E, F, C, A.

BREADTH FIRST TRAVERSAL

Visit tree level by level.
Order of nodes: A, B, C, D, E, F, G.

Sequential or Linear Search

Unordered sequence: Walk through the sequence, comparing each element in turn to the user given key value x, until we find an element equal to x, or we reach the end.
Sequence ordered in non-decreasing order: Terminate unsuccessful search early if element in sequence is greater than x. Takes O(n) time.

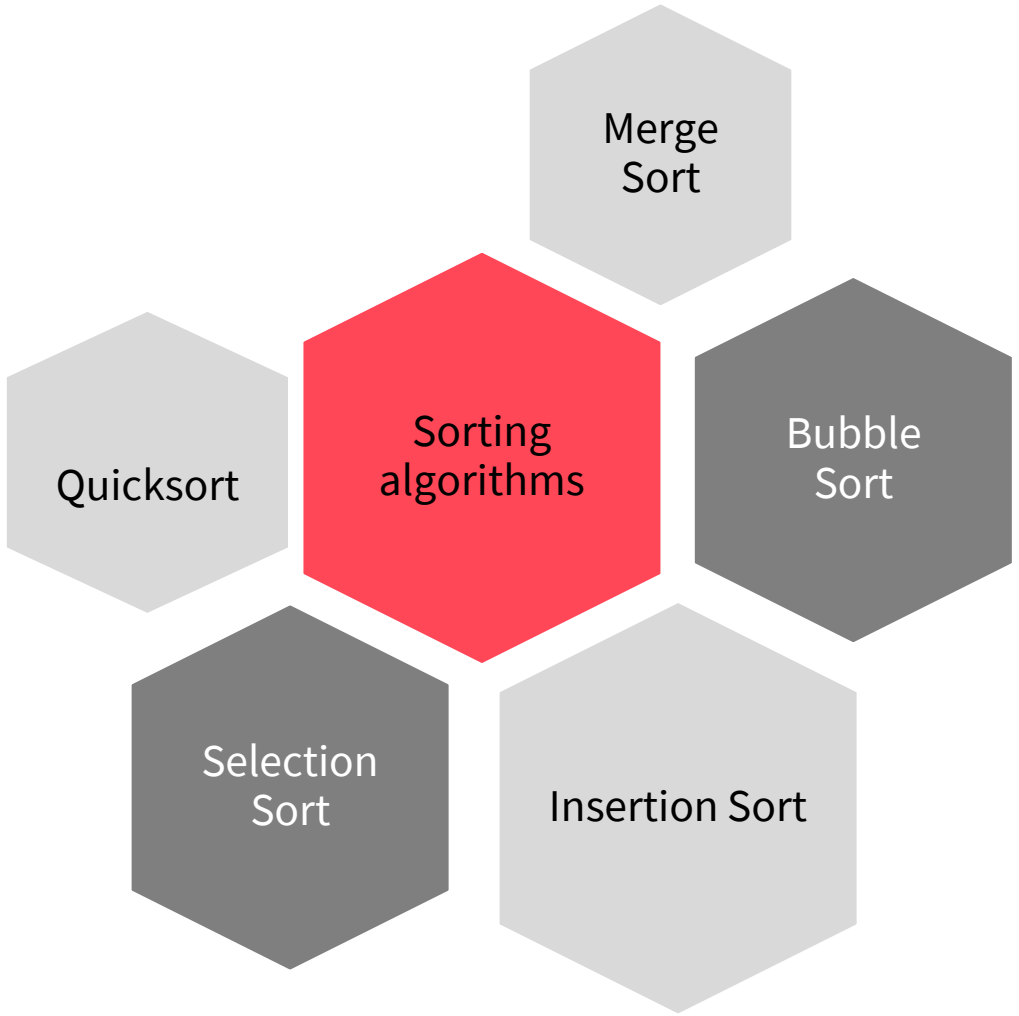
Binary Search

Sequence must be **ordered**.
Compares the user-defined search key x with the middle element. If they are equal, terminates successfully. If x is larger (smaller) than the middle element, the lower (upper) half of the sequence is removed from consideration. Takes O(log n) time.

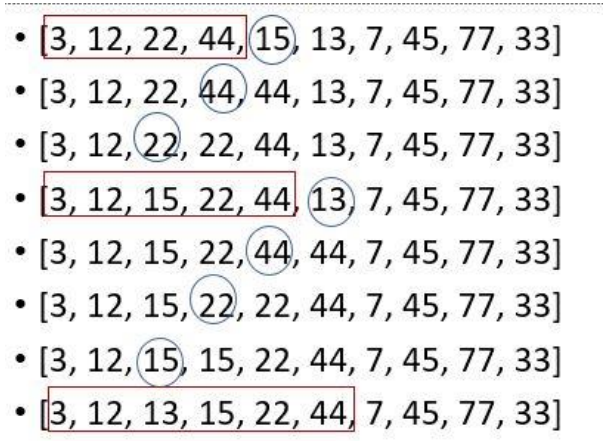
Sorting applications

- Grouping same key values together
- Matching two sets of items
- Searching for a specific item
- Removing duplicates
- Locating elements with unique keys
- Finding median and mode
- Finding the closest pair of key values
- Selecting the k-th largest element

Graphix X: Sorting Algorithms.

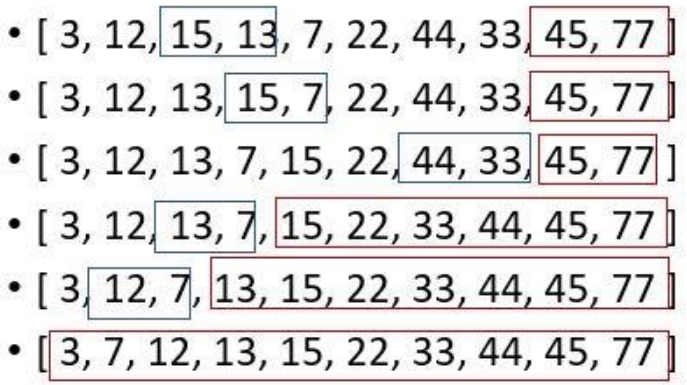


SORTING ALGORITHM EXAMPLES



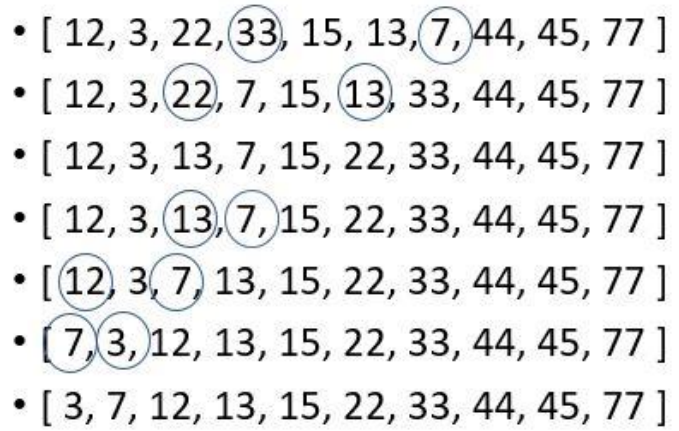
- [3, 12, 22, 44, 15, 13, 7, 45, 77, 33]
- [3, 12, 22, 44, 44, 13, 7, 45, 77, 33]
- [3, 12, 22, 22, 44, 13, 7, 45, 77, 33]
- [3, 12, 15, 22, 44, 13, 7, 45, 77, 33]
- [3, 12, 15, 22, 44, 44, 7, 45, 77, 33]
- [3, 12, 15, 22, 22, 44, 7, 45, 77, 33]
- [3, 12, 15, 15, 22, 44, 7, 45, 77, 33]
- [3, 12, 13, 15, 22, 44, 7, 45, 77, 33]

In **Insertion Sort**, a new element is inserted into a subsequence which is already sorted.



- [3, 12, 15, 13, 7, 22, 44, 33, 45, 77]
- [3, 12, 13, 15, 7, 22, 44, 33, 45, 77]
- [3, 12, 13, 7, 15, 22, 44, 33, 45, 77]
- [3, 12, 13, 7, 15, 22, 33, 44, 45, 77]
- [3, 12, 7, 13, 15, 22, 33, 44, 45, 77]
- [3, 7, 12, 13, 15, 22, 33, 44, 45, 77]

In **Bubble Sort**, we iterate through the sequence, swapping adjacent elements if they are not sorted.



- [12, 3, 22, 33, 15, 13, 7, 44, 45, 77]
- [12, 3, 22, 7, 15, 13, 33, 44, 45, 77]
- [12, 3, 13, 7, 15, 22, 33, 44, 45, 77]
- [12, 3, 13, 7, 15, 22, 33, 44, 45, 77]
- [12, 3, 7, 13, 15, 22, 33, 44, 45, 77]
- [7, 3, 12, 13, 15, 22, 33, 44, 45, 77]
- [3, 7, 12, 13, 15, 22, 33, 44, 45, 77]

Selection Sort locates the i-th largest element in the i-th iteration and moves it to its correct position.

SORTING ALGORITHM EXAMPLES

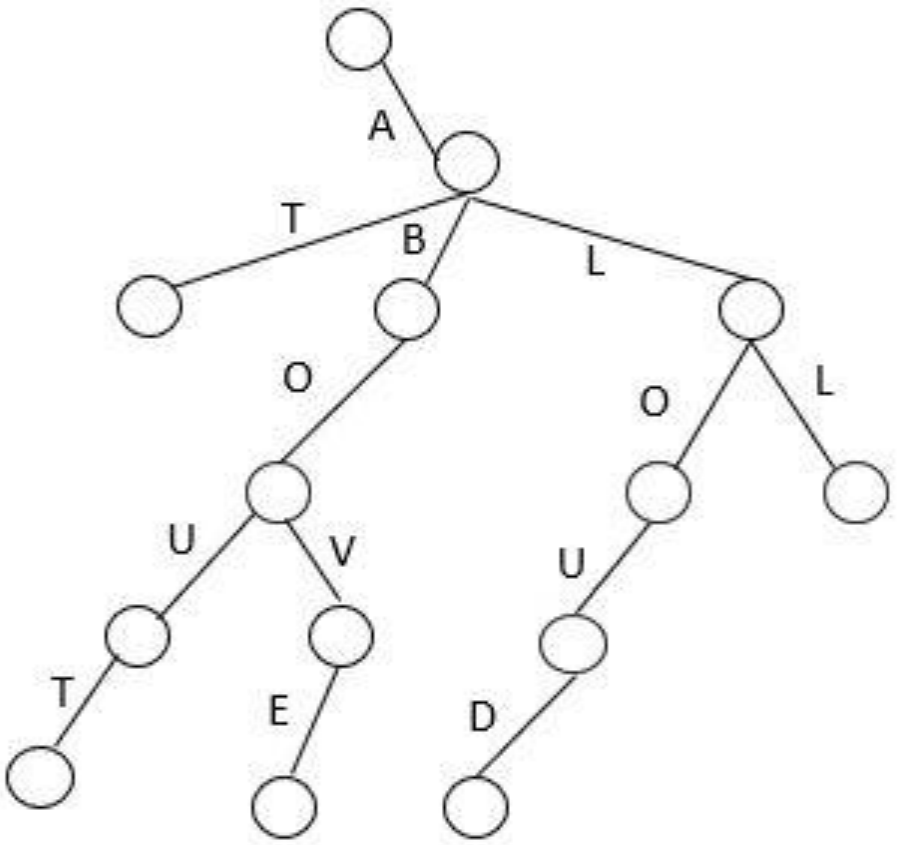
- **Quicksort** chooses a pivot element p and partitions the sequence into two groups of elements: the elements x ≤ p and elements x ≥p. The algorithm then recurses into the two partitions.

- [12, 3, 22, 44, 15, 13, 7, 45, 77, 33] Choose Pivot
- [7, 3, 12, 44, 15, 13, 22, 45, 77, 33] Partition
- [3, 7, 12, 44, 15, 13, 22, 45, 77, 33] Sort Left
- [3, 7, 12, 13, 15, 22, 33, 44, 45, 77] Sort Right

- **Mergesort** divides the sequence into two equal parts and sorts them. Then it merges the wo sorted subsequences.

- [12,3,22,44,15,13,7,45,77,33]
- [12,3,22,44,15,13,7,45,77,33]
- [3,12,22,44,15,13,7,45,77,33]
- [3,12,22,44,15,13,7,45,77,33]
- [3,12,22,15,44,13,7,45,77,33]
- [3,12,15,22,44,13,7,45,77,33]
- [3,12,15,22,44,7,13,45,77,33]
- [3,12,15,22,44,7,13,45,77,33]
- [3,12,15,22,44,7,13,45,33,77]
- [3,12,15,22,44,7,13,33,45,77]
- [3,12,15,22,44,7,13,33,45,77]
- [3,7,12,13,15,22,33,44,45,77]

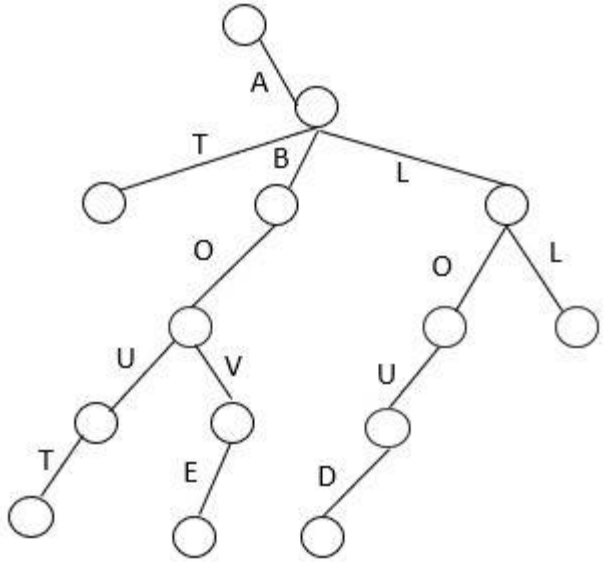
- Trie or digital search trees
- Standard tries
  - Let Σ be an alphabet. Let S be a set of strings from Σ with total length n satisfying the **prefix property**. We define a trie over S to be a tree satisfying the following properties:
    - Each edge is labeled with a character from Σ.
    - Each node has, at most, |Σ| children.
    - Edges connecting a node to its child nodes are all labeled differently.
    - The number of leaf nodes is exactly |S| .
    - Each leaf node v is associated with a string that is the concatenation of the characters on the path from the root to v.
    - The total number of nodes in the trie is n + 1.
    - The height of the trie is the same as the size of the longest string in S.



A trie of words
“all","aloud","above","at","about"

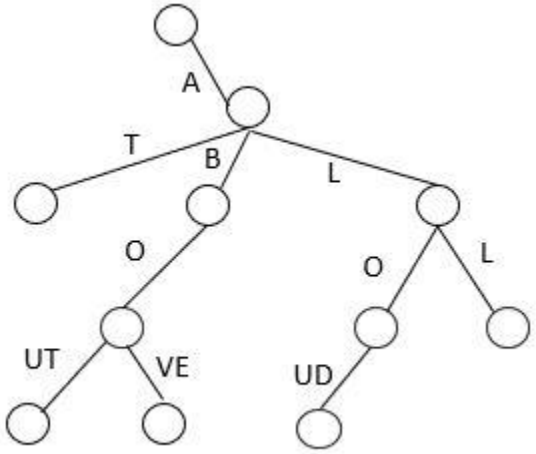
Tries

- Problem solved: User searching for a specific words in a set of words
- Important data structure in infomation retrieval
- Represents elements as sequence of characters and digits



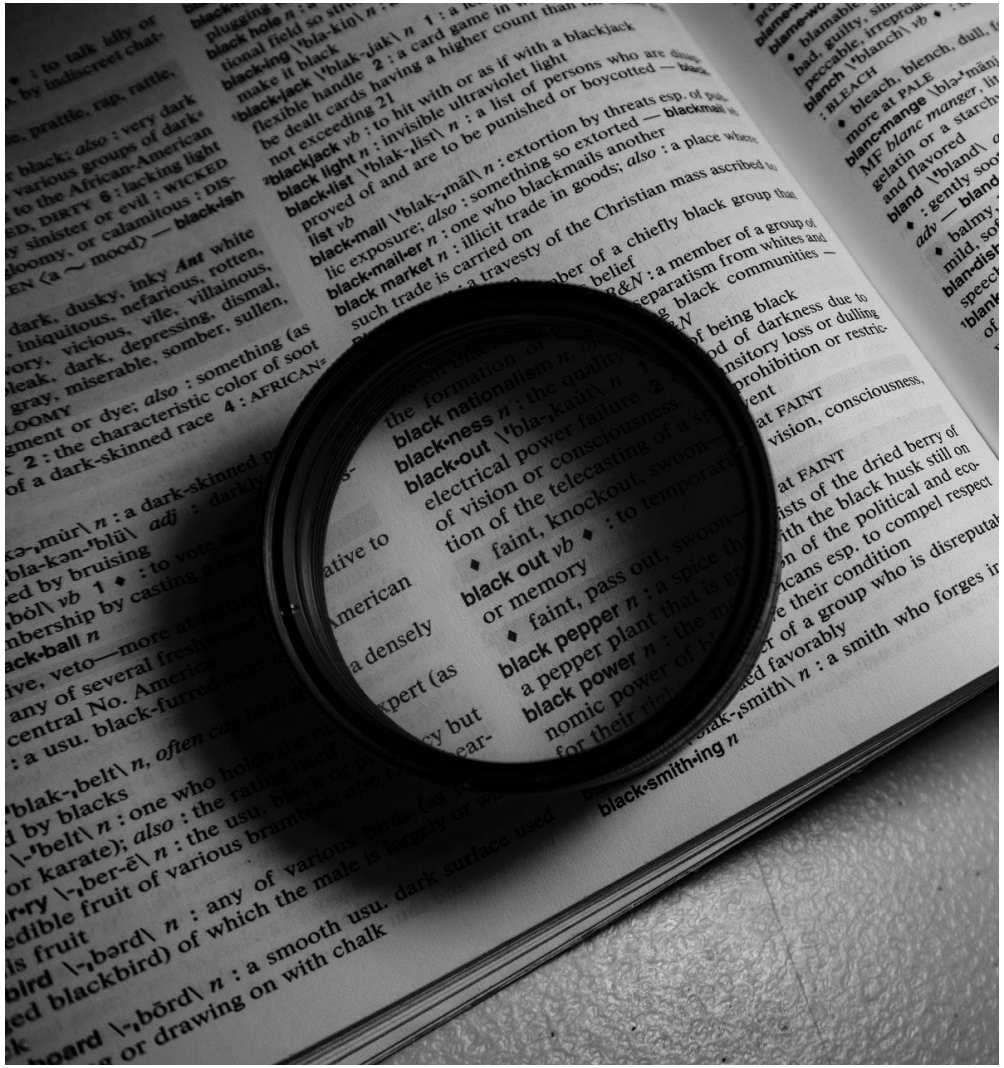
Patricia tries

- Simple compression idea to reduce a redundant chain of edges into a single edge



SEARCHING DICTIONARIES WITH HASH TABLES

- Hash tables are a very popular structure for dictionaries.
- The algorithms for search queries are content-based as opposed to being comparison-based.
- The data is stored in locations which are computed by simple functions from the data itself.
- Uses keys which are composed of attribute(s) of the values.



Source of the image: Pexels.

- Hashing is the mapping of keys to locations of a one-dimensional array (hash table T of size m).
- The mapping is computed by a hash function.
  - If K is the set of keys, the hash function h maps k ∈ K to h k ∈ 0,1, …,m − 1
- A “collision” is said to occur if two keys map to the same location:

  $k_1 \in K$, $k_2 \in K$, $h(k1) = h(k2)$, $k_1 \neq k_2$.
- Hashing scheme
  - What should be the hash function?
  - What should be the collision resolution algorithm

GRAPHIC X HASHING SCHEMES

HASHING SCHEME
is composed of the following components

HASHING SCHEMES
require mapping keys to locations in the table

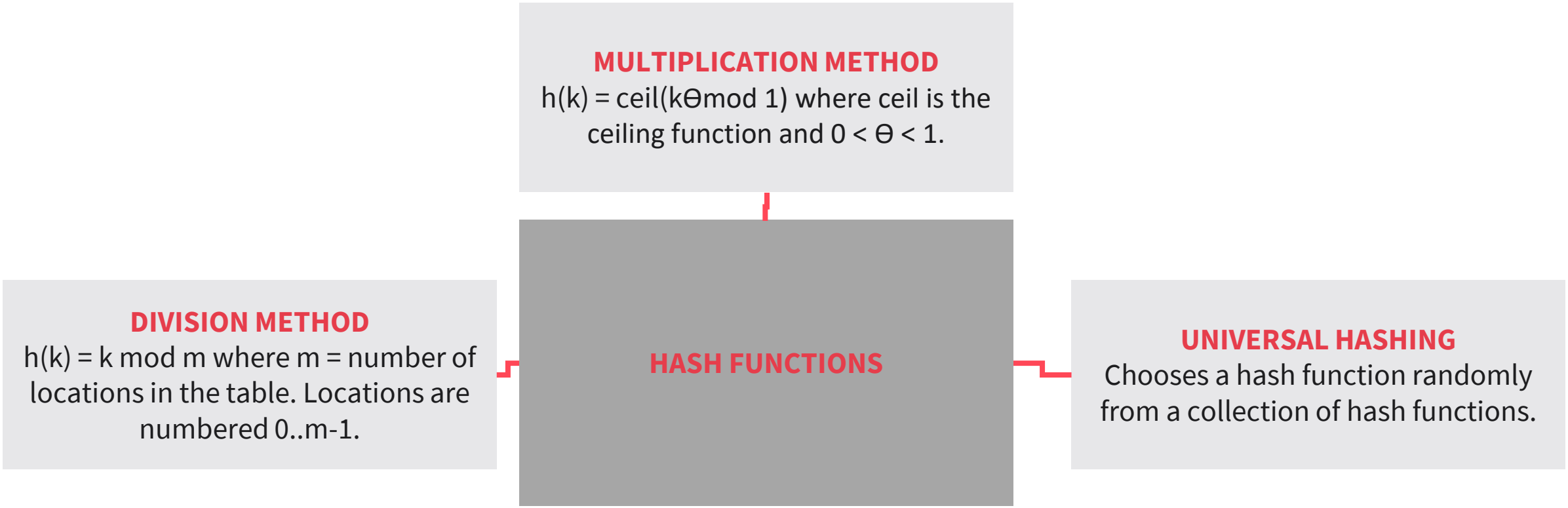
Hash Function

- Generates key values which are indices of the hash table. The search algorithm compares the search key with key values stored at those indices.
- Should be easy to compute.
- Should distribute keys well across the table.

Collision Resolution Scheme

When keys map to the same location, the collision must be resolved.

HASH FUNCTION EXAMPLES



COLLISION RESOLUTION SCHEMES

OPEN ADDRESSING WITH QUADRATIC PROBING

We probe at locations $g(k,i) = (h(k) + c_1.i + c_2.i^2) \bmod m$, $0 \le i < m$.

OPEN ADDRESSING WITH LINEAR PROBING

We probe at locations $g(k,i) = (h(k) + i) \bmod m$ for $0 \le i < m$.

COLLISION RESOLUTION SCHEMES

Some commonly used schemes. Open addressing schemes look for an empty slot.

OPEN ADDRESSING WITH DOUBLE HASHING

We generate the probe sequence using two hash functions $h(k,i) = (h_1(k) + i.h_2(k) \bmod m$.

CHAINING

Each index i of the hash table is associated with a linked list which stores all keys that map to i.

Pattern Matching: We have to find all occurrences of a pattern P[0..m-1] in the text T[0..n-1]. Shift s is a **valid shift** if P occurs in T with a shift s, starting at position s. Otherwise, the shift is invalid.

0	1	2	3	4	5	6	7	8	9
c	b	a	b	a	b	a	b	a	a

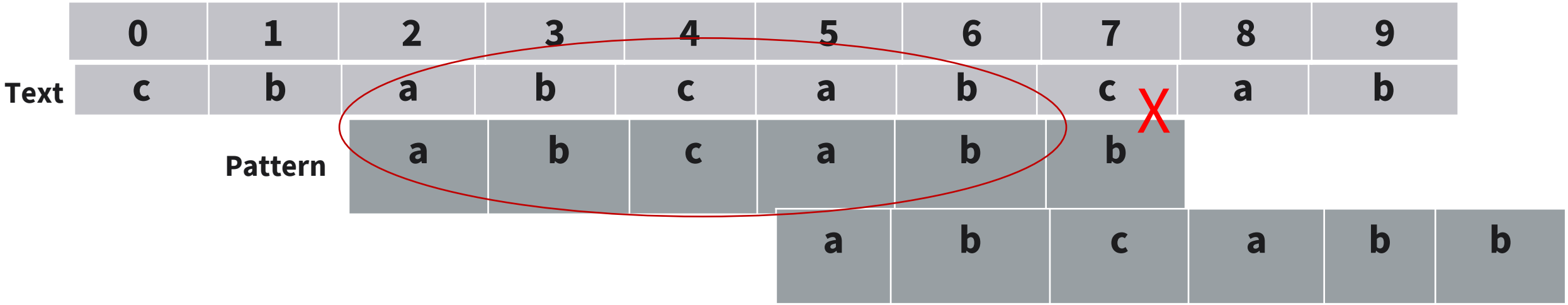
a	b	a							

		a	b	a					

				a	b	a			

Here valid shifts s=2 ,4, 6. Naïve Pattern Matching attempts all n-m+1 shifts.

Knuth-Morris-Pratt or KMP algorithm



If a match fails and a suffix of the matched part is a prefix of the pattern, we need not match that prefix when we shift and attempt a match again. The KMP algorithm runs in O(m+n) time.

REVIEW STUDY GOALS

- Understand algorithms for traversal of trees.
- Understand basic algorithms for searching.
- Differentiate between various algorithms for sorting.
- Understand the trie data structure.
- Know about hashing techniques in solving search problems.
- Understand fundamental algorithms for pattern recognition.

SESSION 3

TRANSFER TASK

Explain how you can use sorting to solve the following problems for an array of integers:

a) Find all duplicate integers.
b) Find a specific integer.
c) Find the k-th smallest element.
d) Find the mean, median and mode.

TRANSFER TASK
PRESENTATION OF THE RESULTS

Please present your results.

The results will be discussed in plenary.

1. The sequence of nodes visited during inorder traversal on a tree with three nodes is B, A, and C. Which of the following is a possible valid sequence of nodes visited?
   a) C, A, B is a preorder sequence.
   b) A, B, C is a preorder sequence.
   c) A, B, C is a postorder sequence.
   d) A, C, B is a postorder sequence.

2. Which of the following is true for linear searches on ordered and unordered lists?
   a) An unsuccessful search on an ordered list is faster.
   b) A successful search on an ordered list is faster.
   c) An unsuccessful search on an unordered list is faster.
   d) A successful search on an unordered list is faster

3. How many leaves does a trie storing the words “bear, bent, ball, and bat” have?
   a) 8
   b) 5
   c) 4
   d) 15

LIST OF SOURCES

Cormen, T.H., Leiserson, C.E., Rivest, R.L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
Goodrich, M.T., Tamassia, R., & Goldwasser, M.H. (2013). *Data Structures and Algorithms in Python.* Wiley.