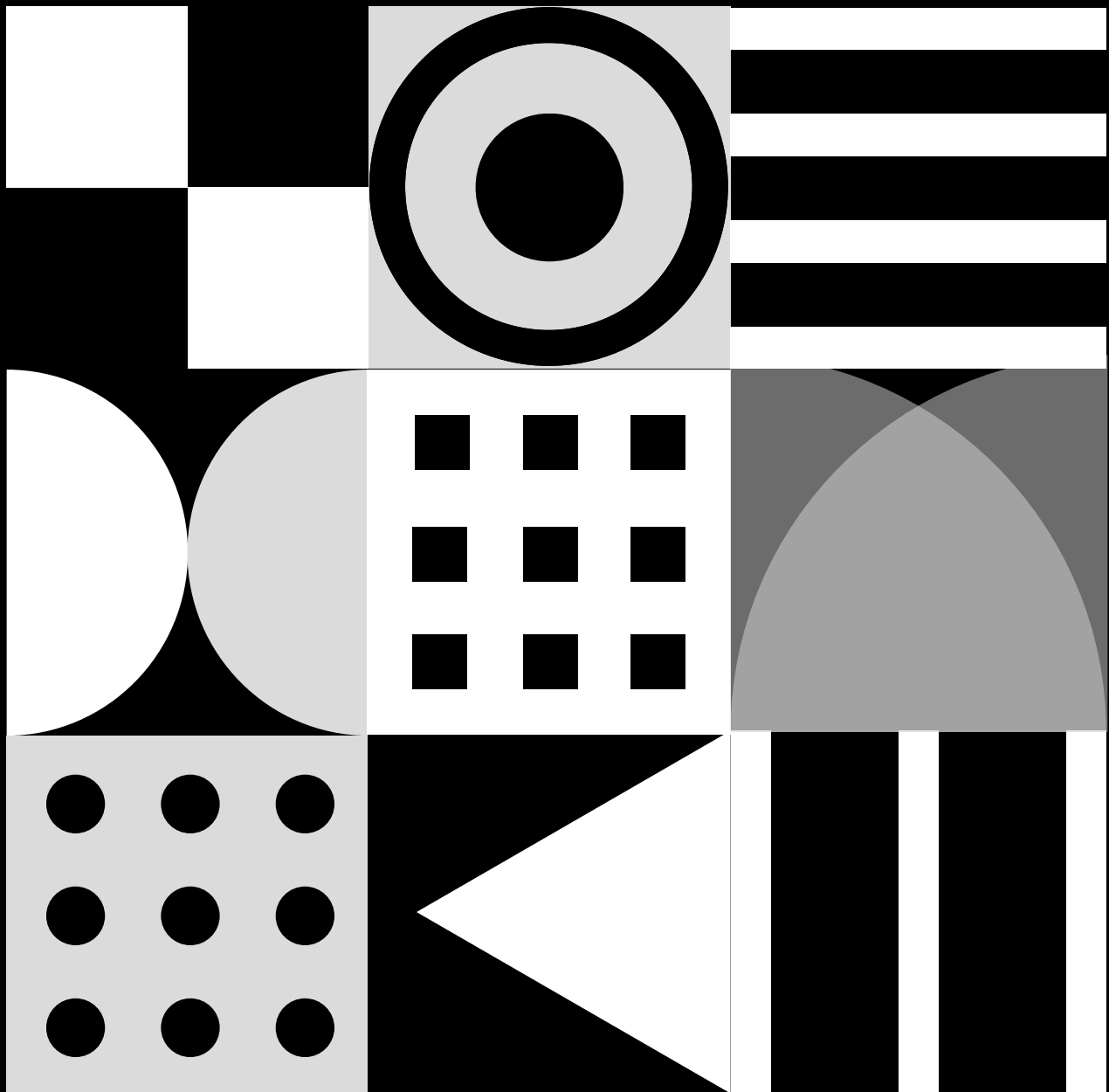


# POST- PRESENTATION REPORT

## GROUP - 7


# 2023



# Table of Content



Team and contributions	01
Introduction to File Organisation	02
Hashing Algorithms	02
Techniques of Hashing	04
Collision and Synonyms	05
Key Findings	07
Conclusion	08
Acknowledgements	09



# Team and contributions

---

Contributor	Contribution
Adarsh Chauhan-011	Creating visual elements and graphics for the PPT, designing PPT layout
Vansh Chawla-015	Creating visual elements and graphics for the PPT, designing PPT layout
Pulkit-037	Finding content for the PPT individually
Jatin-076	Finding content for the PPT individually
Chirag-040	Finding content for the PPT individually
Himanshu-063	Finding content for the PPT individually
Team effort	Contributed to an old PPT

**Submitted To**  
*Mrs. Prageesha Chawla*  
*Astt. Prof. Dyal Singh*  
*College, Karnal.*

It's important to note that this table only reflects the individual contributions of each team member and does not capture any collaborative or joint efforts that may have taken place during the presentation preparation process. Additionally, it's possible that some team members may have contributed in ways beyond their assigned tasks or responsibilities.

---

# Introduction

---

File organization is a fundamental concept in computer science, which refers to the way in which data is stored and retrieved from secondary storage devices such as hard drives, flash drives, and magnetic tapes. There are several methods of file organization, each with its advantages and disadvantages, depending on the type of data being stored and the requirements of the application.

One of the most common methods of file organization is sequential access, in which data is stored in a linear fashion, and access is possible only by reading from the beginning of the file. This method is useful for storing data that needs to be processed in a sequential manner, such as log files or transaction records. However, it can be slow for random access and can lead to data fragmentation if files are frequently updated or deleted.

Another method of file organization is direct access, in which data is stored in fixed-length blocks, and access is possible by specifying the block number or offset. This method is faster for random access than sequential access and is commonly used for storing large files, such as images or videos. However, it can be less efficient for storing small files and can waste space if the block size is too large.

Indexed file organization is a hybrid method that combines the benefits of sequential and direct access. In this method, data is stored in a sequential manner, but an index is created that provides a mapping between the data and its location on disk. This method is useful for storing data that needs to be accessed randomly but does not require direct access to specific blocks.

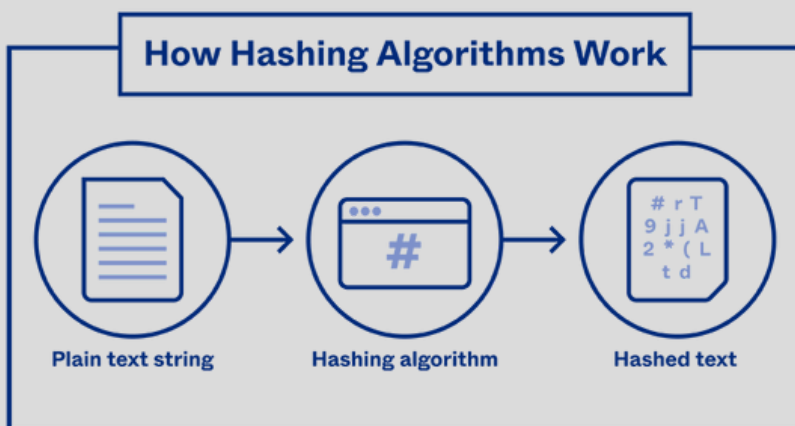
Hashing is another method of file organization that uses a hash function to map data to its storage location. This method is useful for storing large amounts of data that need to be accessed quickly, such as a database of customer records. However, it can lead to collisions if two or more pieces of data map to the same storage location, which can slow down access times.

In conclusion, the choice of file organization method depends on the type of data being stored, the requirements of the application, and the performance considerations. Understanding the different methods of file organization is important for computer science students and professionals, as it is essential for designing efficient and scalable systems


---

# Hashing Algorithm

---



***Hashing algorithm maps data of arbitrary size to fixed-size values.***



In data structures, hashing algorithms are commonly used to implement hash tables, which are used to store and retrieve data in constant time. Hash tables are particularly useful for applications that require efficient searching and insertion of large amounts of data.

Hashing algorithms work by taking a key, which is a unique identifier for a piece of data, and applying a mathematical function to generate an index in an array. The key-value pair is then stored at the corresponding index. When retrieving the data, the key is used again to generate the index and the value is returned.

One of the challenges of using hashing algorithms is handling collisions, which occur when two keys generate the same index. This can be addressed through techniques such as chaining or open addressing, where collisions are resolved by storing the values in linked lists or by finding an alternate index to store the data. Overall, hashing algorithms are an important tool in data structures and are widely used in applications such as databases, compilers, and network protocols.

# Techniques of Hashing

---

Techniques of hashing in data structures refer to methods used to address the issue of collisions, which can occur when two keys map to the same index. Techniques such as chaining, open addressing, and rehashing are commonly used to resolve collisions and ensure efficient storage and retrieval of data in hash tables.

FEW OF THEM ARE :-

## 01 | Division method

Hash value is obtained by dividing the key by a prime number and taking the remainder as the index. This method is simple but can lead to clustering.

## 02 | Truncation method

Hash value is obtained by selecting a part of the key and using it as the index. This method is easy to implement but can lead to collisions.

## 03 | Shifting method

Hash value is obtained by shifting the bits of the key and using a part of the result as the index. This method is efficient and reduces the likelihood of collisions.

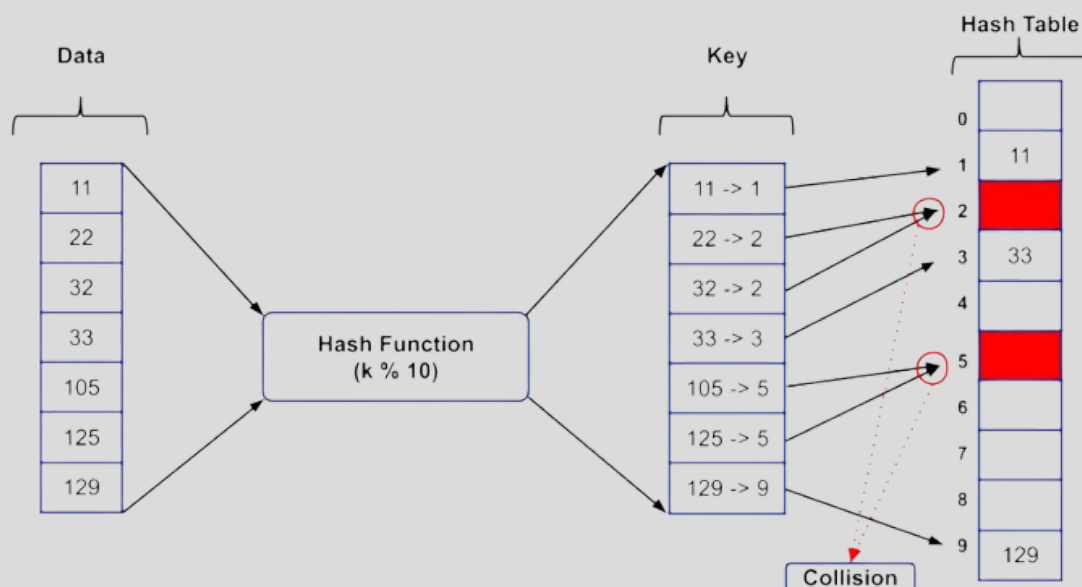
## 04 | Folding method

Hash value is obtained by dividing the key into equal-sized parts and adding them together. The sum is then reduced to the desired size and used as the index.

# Collision and Synonyms

---

In data structures, collision occurs when two or more keys map to the same index in a hash table. This can happen due to the finite size of the table and the potentially infinite number of possible keys. When a collision occurs, it is necessary to resolve it in order to store and retrieve the data accurately.



In the given example, a collision occurs because the hash function generates the same hash value (key) of 2 for two different data items (22 and 32) and the same hash value of 5 for two different data items (105 and 125). This collision is due to the finite size of the hash table and the potential for multiple keys to generate the same hash value. Techniques such as chaining or open addressing can be used to resolve collisions and ensure accurate storage and retrieval of data in the hash table.

**Open addressing and chaining are two common techniques used to resolve collisions in hash tables.**

**Open addressing** involves finding an alternate location within the hash table to store the value when a collision occurs. This can be done in a variety of ways, such as linear probing, quadratic probing, or double hashing. In linear probing, the next available slot in the hash table is used, while in quadratic probing, a series of slots are checked in a quadratic sequence. Double hashing uses a secondary hash function to find an alternate location for the value.

Here is an example of open addressing using linear probing:

Key	Value
10	A
22	B
31	C
4	D
15	E
28	F

In this example, a hash function maps the keys to indices in the hash table. When a collision occurs, linear probing is used to find the next available slot. If the slot is already occupied, the next slot is checked until an empty slot is found.

**Chaining** involves storing multiple values at the same index in the hash table. This is typically done using linked lists or other data structures. When a collision occurs, the new value is added to the linked list at the index. When retrieving a value, the linked list is traversed until the correct value is found.

Here is an example of chaining using linked lists:

Key	Value
10	A -> D
22	B
31	C
4	E
15	F
28	G



# Key Findings

---

Where do you go from here? Sustainability reports are not just about looking back, but also looking forward. This SDG Progress Report is a continuous work in progress - a way for your organization to track its impact and improvements over time. This section outlines your strategy for continuing the good work done so far.

## 01 | Advantages of Hashing Algorithm

1. Fast access and retrieval of data.
2. Efficient use of memory.
3. Hashing algorithms can be adapted to different types of data.

## 02 | Challenges in Hashing Algorithm

1. Collisions can occur, resulting in decreased efficiency.
2. Selecting an appropriate hash function can be challenging.
3. Hash tables can become full, requiring resizing or rehashing.

## 03 | Collision Resolution Techniques

1. Chaining: Storing multiple values at the same index.
  2. Open addressing: Finding an alternate location within the hash table to store the value.
  3. Hybrid approaches: Combining chaining and open addressing techniques.
-

# Conclusion

---

In conclusion, hashing algorithms and collision resolution techniques are critical components of data structures. Hashing algorithms are used to map keys to indices in a hash table, allowing for efficient storage and retrieval of data. The efficiency of the hash table depends on the quality of the hash function and the size of the hash table.

However, collisions can occur when two different keys map to the same index in the hash table. This can result in the loss of data or the retrieval of incorrect data. Collision resolution techniques such as open addressing and chaining are used to resolve these collisions and ensure accurate storage and retrieval of data in the hash table.

Open addressing involves finding an alternate location within the hash table to store the value when a collision occurs, while chaining involves storing multiple values at the same index in the hash table. Each technique has its advantages and disadvantages, and the choice of technique depends on the application and the specific requirements of the system.

It is important to note that a good hash function can reduce the likelihood of collisions, but it cannot eliminate them completely. Therefore, collision resolution techniques are crucial in ensuring the accuracy of data stored in hash tables.

Overall, hashing algorithms and collision resolution techniques are fundamental to the performance and efficiency of data structures. By using these techniques, it is possible to store and retrieve data quickly and accurately, making them essential tools for a wide range of applications, from databases to search engines and beyond.

---

# Acknowledgements

---

We would like to acknowledge that the information presented in this report and the accompanying presentation may be subject to copyright protection and may have multiple owners or contributors. We have made every effort to ensure that all content used in this project is either original or properly licensed. We believe it is important to respect the intellectual property rights of the owners and contributors of the information presented in this report and presentation.

- GeeksforGeeks - <https://www.geeksforgeeks.org/hashing-data-structure/>
  - 
  - Stack Overflow - <https://stackoverflow.com/questions/tagged/hashing>
  - 
  - Codecademy - <https://www.codecademy.com/learn/hash-tables>
  - 
  - HackerRank - <https://www.hackerrank.com/domains/data-structures/hash-tables>
  - 
  - edX - <https://www.edx.org/learn/data-structures-algorithms>
  - 
  - Brilliant - <https://brilliant.org/wiki/hashing/>
- 

***We express our gratitude to all the respected owners and creators whose work we may have used in our files.***