CSA201 – Applied Data Structures and Algorithms

Unit 6 Advanced Data Structures

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

What is a Binary Heap?

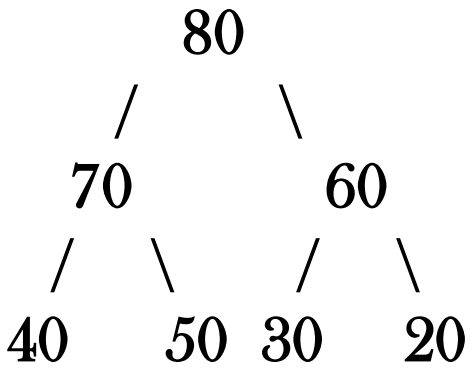
- A Binary Heap *is a Binary Tree with the following properties:*

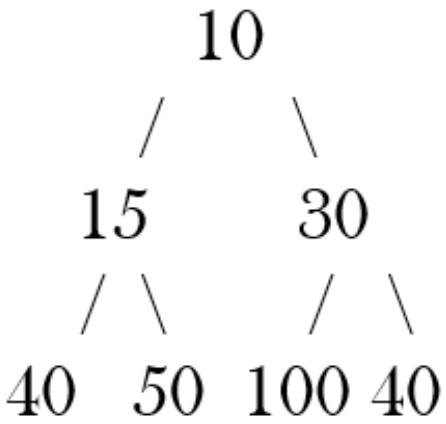
  - *It's a complete tree (All levels are completely filled except possibly the last level and the last level has all keys as left as possible).* This property of Binary Heap makes them suitable to be stored in an array

  - *A Binary Heap is either Min Heap or Max Heap.* In a Min Binary Heap, the key at root must be minimum among all keys present in Binary Heap. The same property must be recursively true for all nodes in a Binary Tree. Max Binary Heap is similar to MinHeap

Types of Binary Heap

- *Min heap* - the value of each node is less than or equal to the value of both its children.
- *Max heap* - it is exactly the opposite of min heap that is the value of each node is more than or equal to the value of both its children.

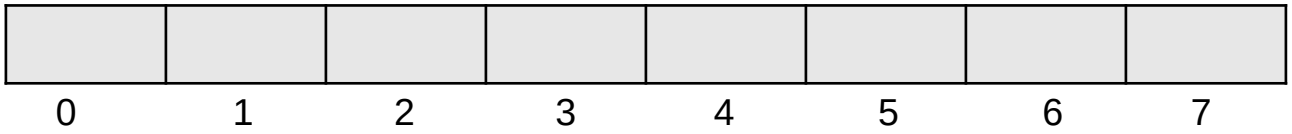




GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY

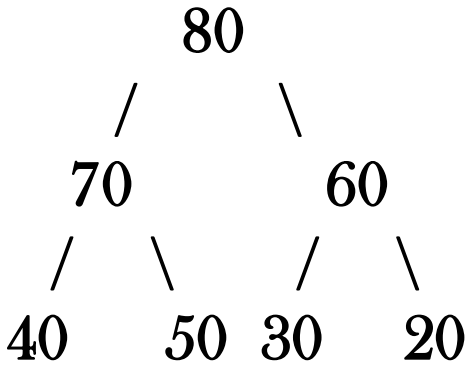
Common Operations on Binary Heap

- *Creation of Binary Heap,*
  - *Initialize Array*
  - *Set size of Binary Heap to 0*



Common Operations on Binary Heap

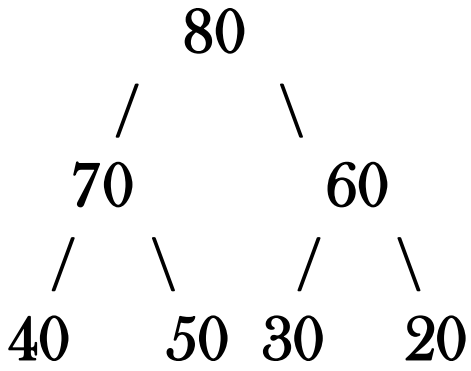
- *Peek of Binary Heap,*
  - *Return array[1]*



	80	70	60	40	50	30	20
0	1	2	3	4	5	6	7

Common Operations on Binary Heap

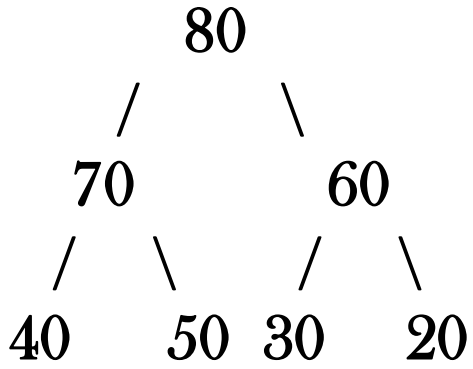
- *Size of Binary Heap,*
  - *Return number of filled cells*



	80	70	60	40	50	30	20
0	1	2	3	4	5	6	7

Common Operations on Binary Heap

- *Level Order Traversal*

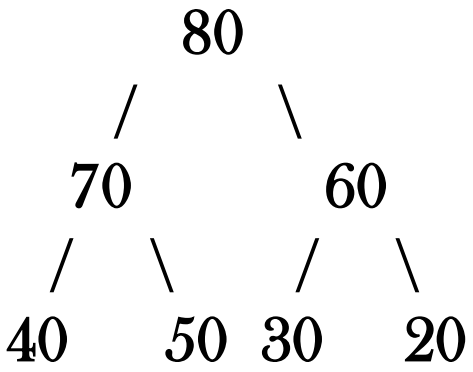


	80	70	60	40	50	30	20
0	1	2	3	4	5	6	7

Binary Heap – Insert a Node

Algorithm for insertion in Max Heap

```
If there is no node,
  create a newNode.
else (a node is already present)
  insert the newNode at the end (last node from left to right.)

heapify the array
```

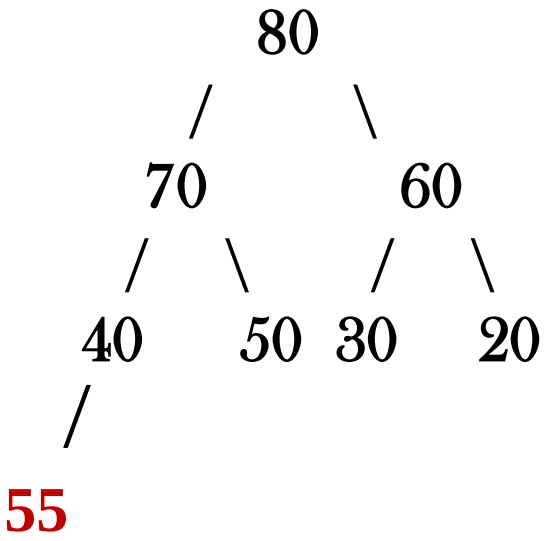


- *Heapify is the process of creating a heap data structure from a binary tree. It is used to create a Min-Heap or a Max-Heap.*

Note: For Min Heap, the above algorithm is modified so that parentNode is always smaller than newNode

Binary Heap – Insert a Node

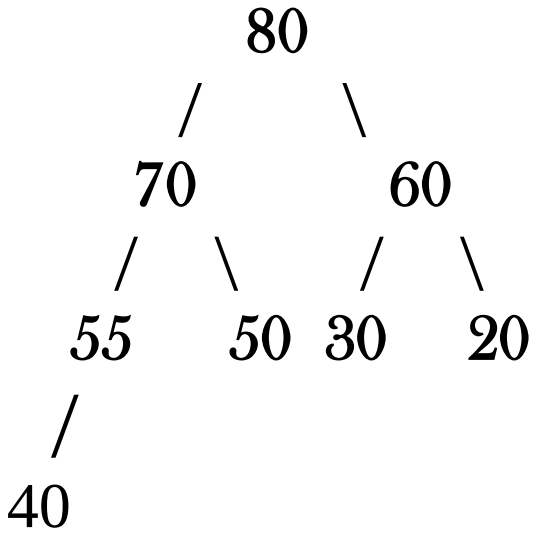
- *Insert the new element at the end of the tree.*



Binary Heap – Insert a Node

Heapify the tree.

```
Heapify(array, size, i)
  set i as largest
  leftChild = 2i + 1
  rightChild = 2i + 2

  if leftChild > array[largest]
    set leftChildIndex as largest
  if rightChild > array[largest]
    set rightChildIndex as largest

  swap array[i] and array[largest]
```

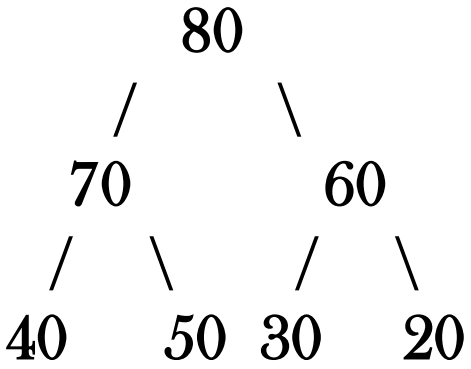


Note: Repeat the heapify until the subtrees are also heapified.

Binary Heap – Delete a Node

Algorithm for deletion in Max Heap

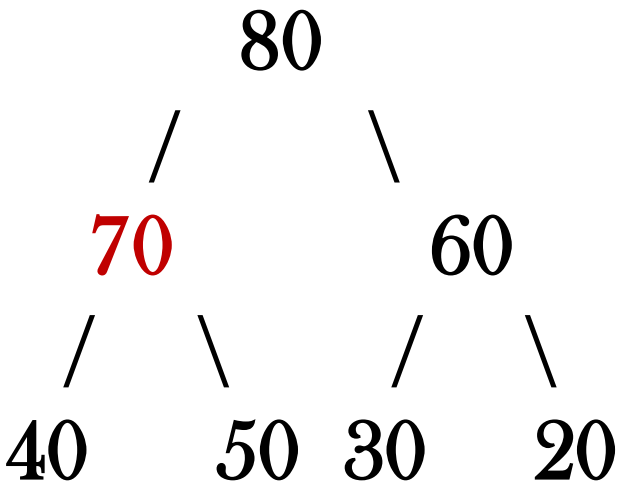
```
If nodeToBeDeleted is the leafNode
  remove the node
Else swap nodeToBeDeleted with the lastLeafNode
  remove noteToBeDeleted

heapify the array
```



For Min Heap, above algorithm is modified so that both childNodes are greater than currentNode.

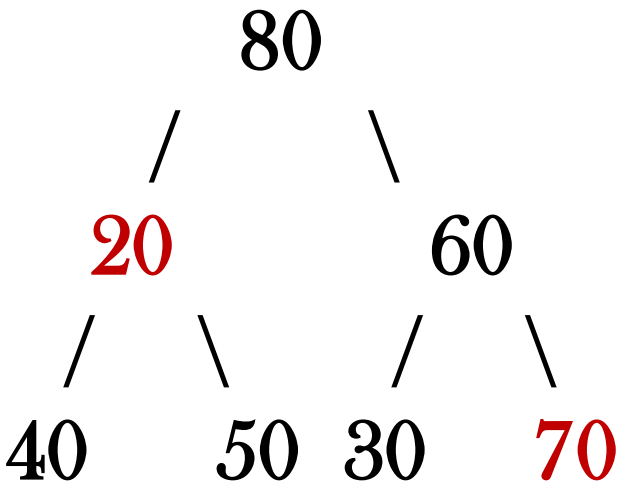
Binary Heap – Delete a Node

1. *Select the element to be deleted.*



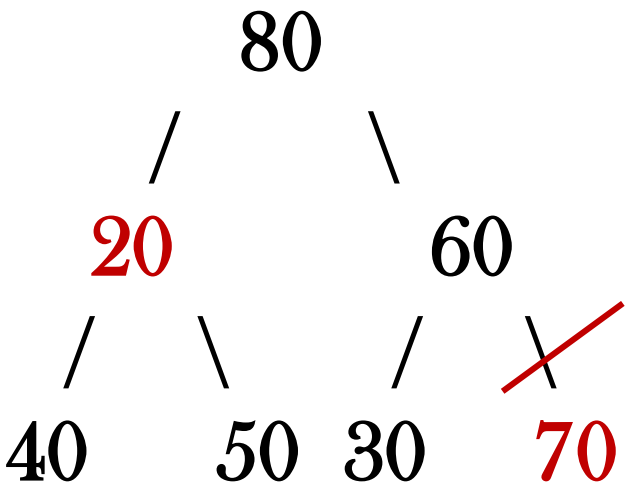
Binary Heap – Delete a Node

2. *Swap it with the last element.*



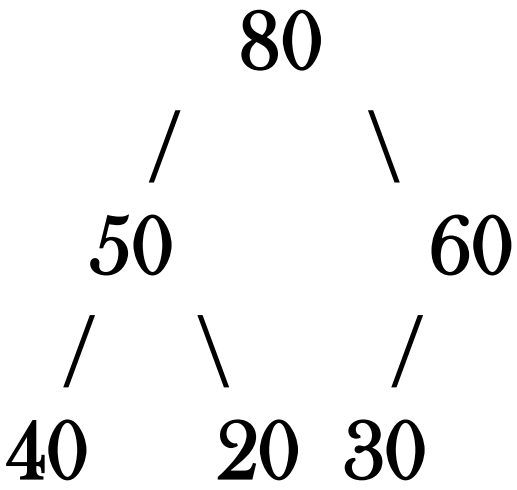
Binary Heap – Delete a Node

3. *Remove the last element.*



Binary Heap – Delete a Node

4. *Heapify the tree.*



Applications of Binary Heap

1) Heap Sort: Heap Sort uses Binary Heap to sort an array in O(nLogn) time.

2) Priority Queue: Priority queues can be efficiently implemented using Binary Heap because it supports insert(), delete() and extractmax(), decreaseKey() operations in O(logn) time. Binomial Heap and Fibonacci Heap are variations of Binary Heap. These variations perform union also efficiently.

3) Graph Algorithms: The priority queues are especially used in Graph Algorithms like Dijkstra's Shortest Path and Prim's Minimum Spanning Tree.

4) Many problems can be efficiently solved using Heaps. See following for example.

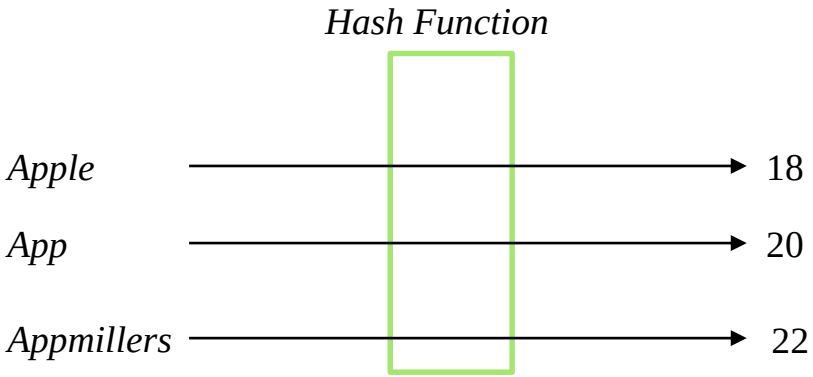
a) K'th Largest Element in an array.

b) Sort an almost sorted array

c) Merge K Sorted Arrays.

What is Hashing?

- Hashing is a method of *sorting* and *indexing* data. The idea behind hashing is to allow large amounts of data to be indexed using keys commonly created by formulas



		...	Apple		App		Appmillers
0	1	...	18	19	20	21	22

Why Hashing?

- It is time efficient in case of SEARCH Operation

Data Structure	Time complexity for SEARCH
Array	O(logN)
Linked List	O(N)
Tree	O(logN)
Hashing	O(1) / O(N)

Hashing Terminology

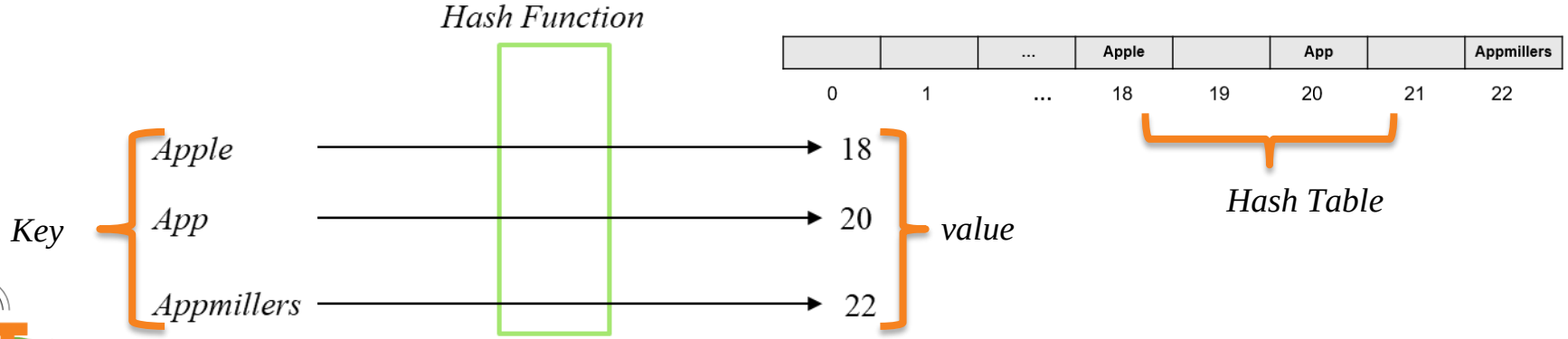
Hash function : It is a function that can be used to map of arbitrary size to data of fixed size.

Key : Input data by a user

Hash value : A value that is returned by Hash Function

Hash Table : It is a data structure which implements an associative array abstract data type, a structure that can map keys to values

Collision : A collision occurs when two different keys to a hash function produce the same output.



Hashing Terminology

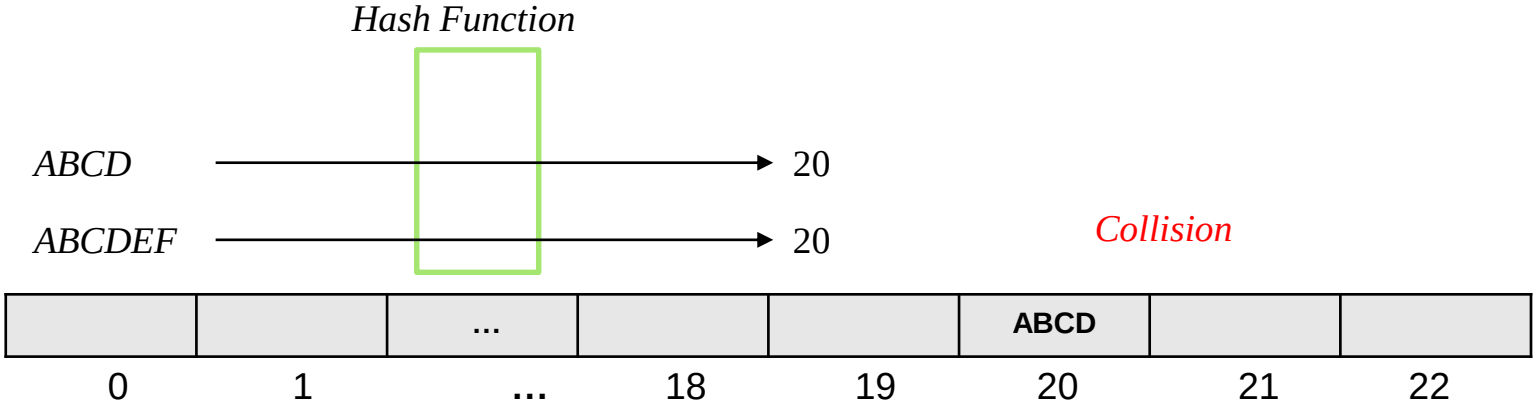
Hash function : It is a function that can be used to map of arbitrary size to data of fixed size.

Key : Input data by a user

Hash value : A value that is returned by Hash Function

Hash Table : It is a data structure which implements an associative array abstract data type, a structure that can map keys to values

Collision : A collision occurs when two different keys to a hash function produce the same output.



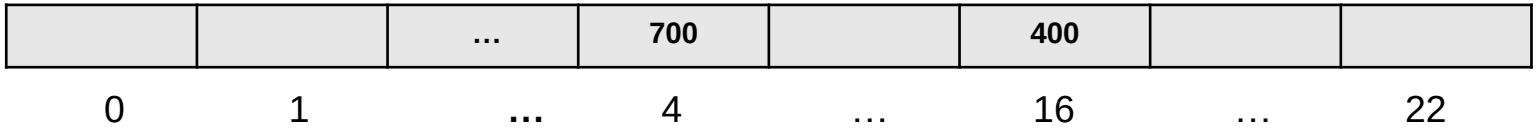
Hash Functions

Mod function

```
int mod(int number, int cellNumber) {
return number % cellNumber;
}
```

mod(400, 24) ⟶ 16

mod(700, 24) ⟶ 4

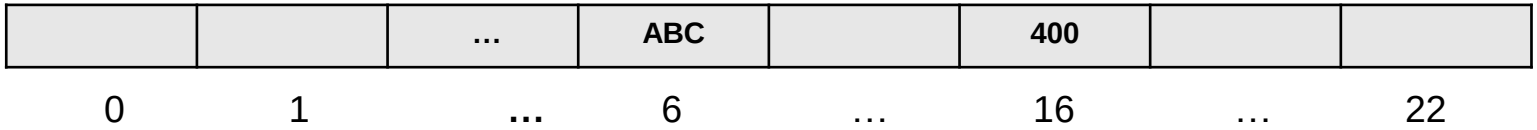


Hash Functions

ASCII function

```
public int modASCII(String word, int cellNumber) {
    int total = 0;
    for (int i=0; i<word.length(); i++) {
        total += word.charAt(i);
        System.out.println(total);
    }
    return total % cellNumber;
}
```

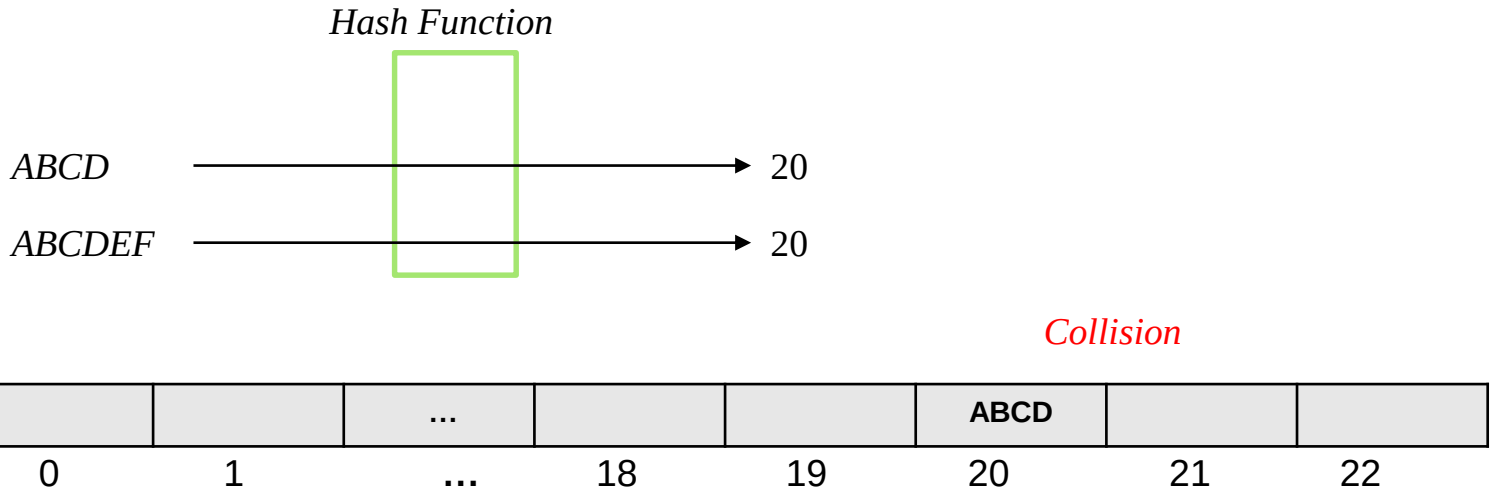
modASCII("ABC", 24) ⟶ 6



Hash Functions

Properties of good Hash function

- It distributes hash values uniformly across hash tables

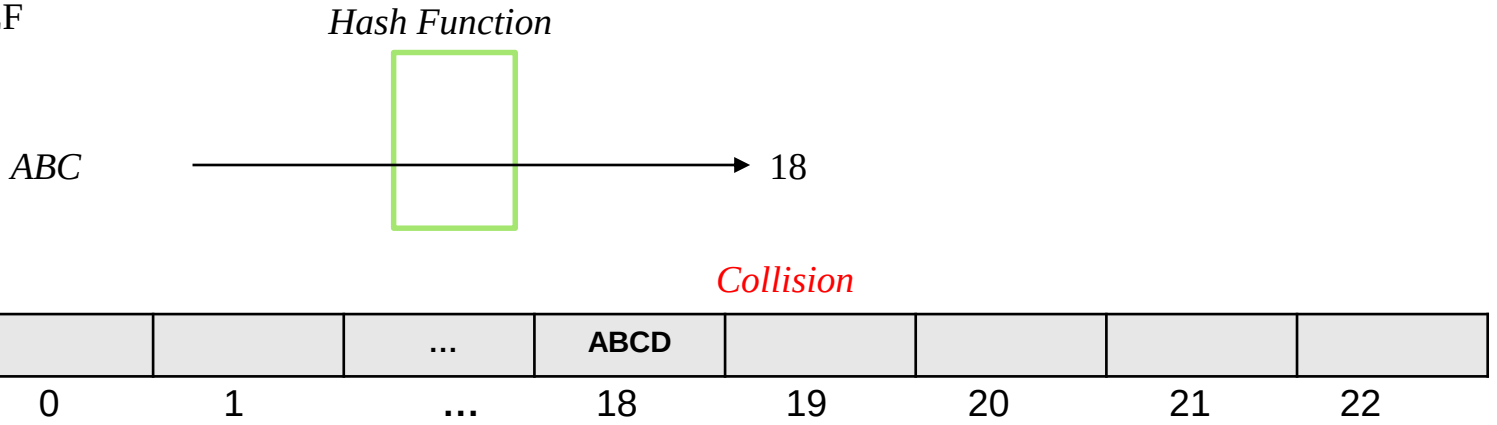


Hash Functions

Properties of good Hash function

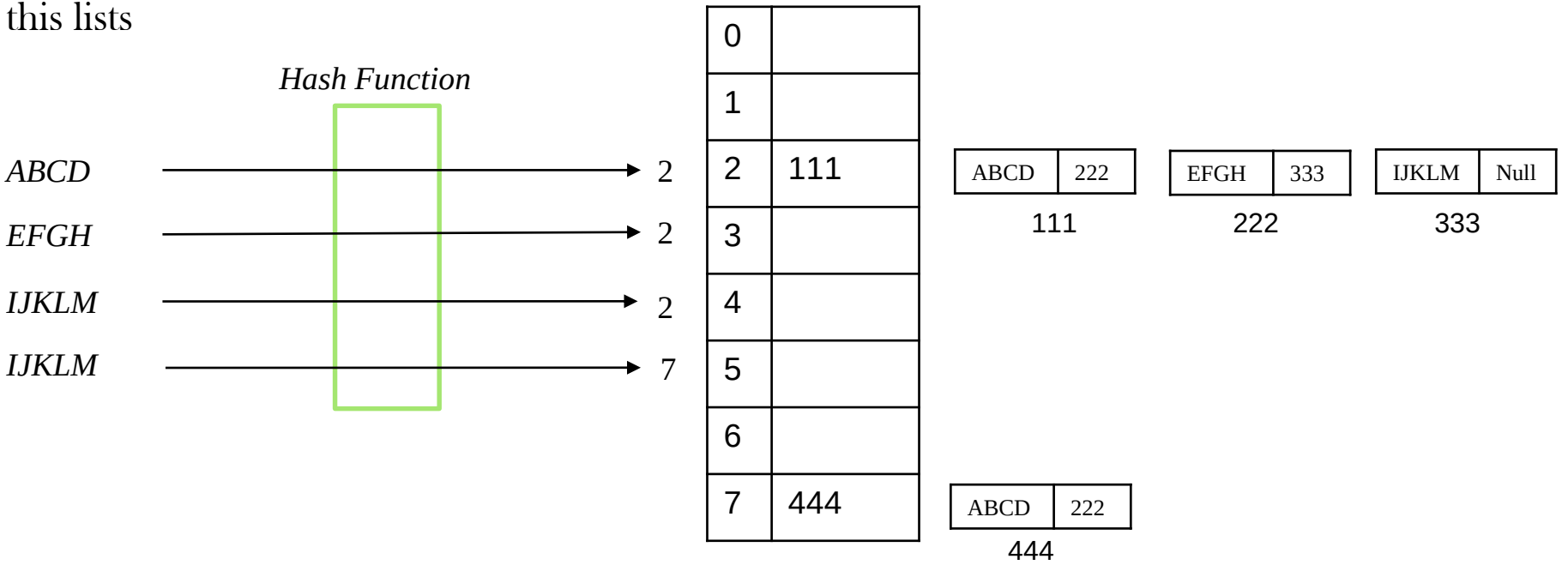
- It distributes hash values uniformly across hash tables
- It has to use all the input data

ABCD
ABCDEF



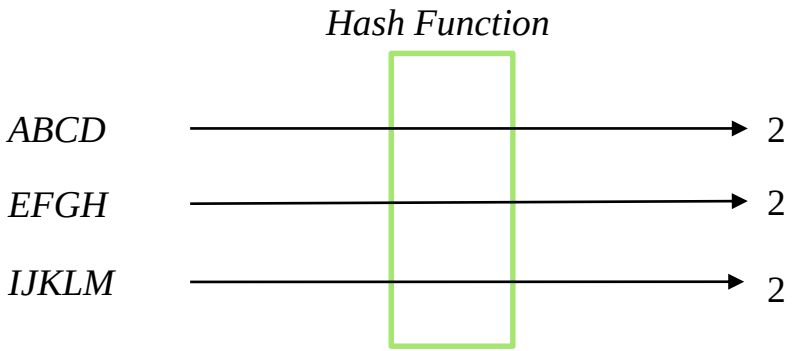
Collision Resolution Techniques

1. Direct Chaining : Implements the buckets as linked list. Colliding elements are stored in this lists



Collision Resolution Techniques

2. Linear probing: It places new key into closest following empty cell



0	
1	
2	ABCD
3	EFGH
4	IJKLM
5	
6	
7	