LECTURE-12

Hashing and Hash Tables

Introduction, Motivation, Concerns

CS202: Data Structures (Fall 2025)

Dr Maryam Abdulghafur, Momina Khan

Department of Computer Science, SBASSE

Agenda

AVL Tree Deletion

Discuss storing unrelated/unordered data

- IP addresses and domain names

Consider conversions between these two forms

Introduce the idea of hashing:

- Reducing $\mathbf{O}(\log(n))$ operations to $\mathbf{O}(1)$

Consider some of the weaknesses

Helper Functions – Time Complexities

```
void updateHeight(Node* n){  O(1)

//sets the height of the node (n) to the 1+max(height of its children)
//code here

}

int getBalanceFactor(Node* n){  O(1)

// returns the balance factor of node, i.e., difference between the heights of
left-subtree and right-subtree
//code here

}
```

Insertion in AVL – Time Complexity

1. New node insertion O(log n)
2. Update height O(1)
3. Compute balance factor O(1)
4. Perform rotations O(1)

How many times are these functions called for one insertion?

Deletion in AVL

1. Delete node n from the tree using BST deletion algorithm
2. Update height of n
3. Compute balance factor of n and check for imbalance

If tree is unbalanced, perform rotations

There are Four possible cases

Left-left case → rightRotate(n)

Right-right case → leftRotate(n)

Right-left case → rightRotate(n→right) then leftRotate(n)

Left-Right case → leftRotate(n→left) then rightRotate(n)

Deletion in AVL

```
Node* BSTDelete(Node* n, int key){ // key = value to be removed from tree
                        //Find node to be deleted
                            if( n→ k > key )
                    n→left = BSTDelete(n→left , key); //recurse left
                            else if ( n→ k < key )
                    n→right = BSTDelete(n→right , key); // recurse right
                            else // key found, BSTDelete logic here
        // 1.Leaf node, 2. node with one child, 3. Node with two children

                                updateHeight(n);
                            int bf = getBalanceFactor(n);
        if(bf > 2 && FindBalanceFactor(n→left) > 0 )// 1. Left-left case
                                RotateRight(n);
                    //Similarly add checks for remaining three cases here
                // 2. Right-Right case, 3. Left-Right case, 4. Right-Left case

                                return n;
                                }
```

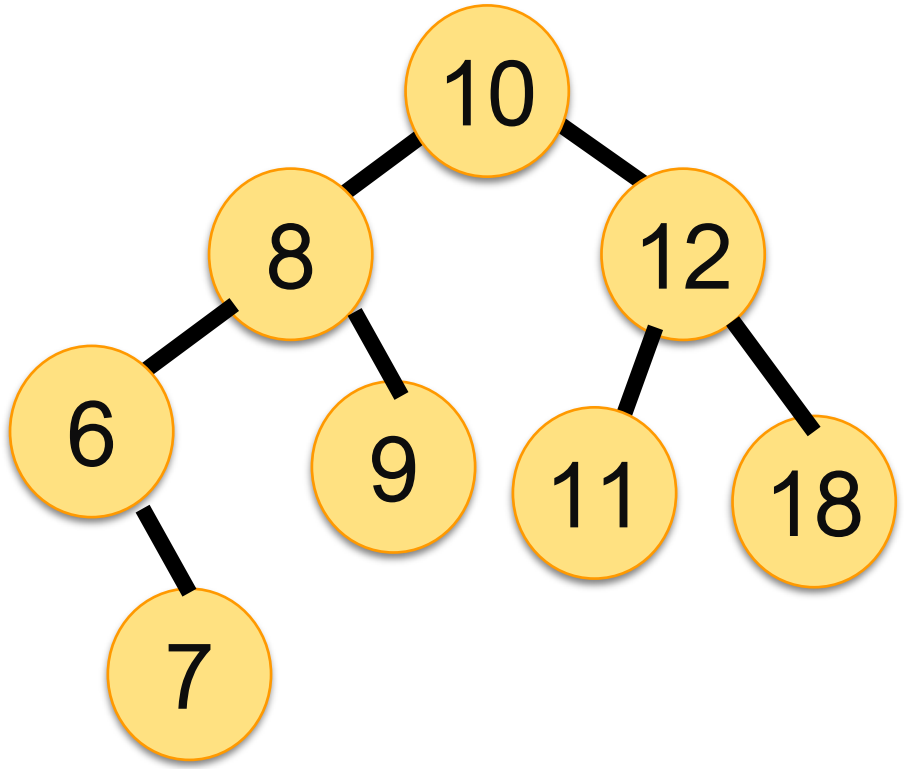
Deletion in AVL – Time Complexity

1. Delete Node O(log n)
2. Update height O(1)
3. Compute balance factor O(1)
4. Perform rotations O(1)

AVL Time Complexities

Search O(log n)
Insertion O(log n)
Deletion O(log n)

Delete



Delete ‘ 18’

Delete ‘10’

Quick recap

- You have covered many linked structures.
- Some with single links eg. Singly Linked Lists! Motivation ??
- Some with multiple links e.g.
  - Doubly Linked List
  - Trees … BSTrees, AVL trees

  Motivation ??

What is a **domain name** and what is an **IP address**?

www.daraz.pk **47.246.165.107**

How do they match?

Can you compare it to a university analogy?

Every Student name maps to a unique Student ID

Habib Ashraf **28100010**

Consider two problems:

- Type a domain name or URL in browser it must be resolved to the correct IP address?

  What is the challenge?

  Domain registrations **reached 368.4 million as of early 2025.**

  Most popular **.com** domain names! Nearly half!

- Type a student roll number and very quickly locate their data?

  What is the challenge?

  - What Data Structure would you use to hold the students' data?
  - An array, a linked structure (linked list, BSTs, AVL)?
  - What is the **efficiency** of search in each? Which is the **fastest**?

A Naïve Solution

Q: What if I do not have to search in an array?

A: What if I use the Student ID as the index into the student array!

Some natural questions that follow:

What is the number of digits in a roll number?

What is the size of array that I will have to declare to accommodate the students?

Any issues that you can identify?

A Naïve Solution

Q: What can you say about the utilization of this array?

A: First few digits are same for everyone! So lots of slots will remain empty!

Q: Can I use a smaller array for only for the last few digits of the students' roll number? Can I reduce 8 digit to 3 digit? How?

Q: Size of array is drastically reduced, can you identify a potential problem?

IMPORTANT TERMS:

Hashing

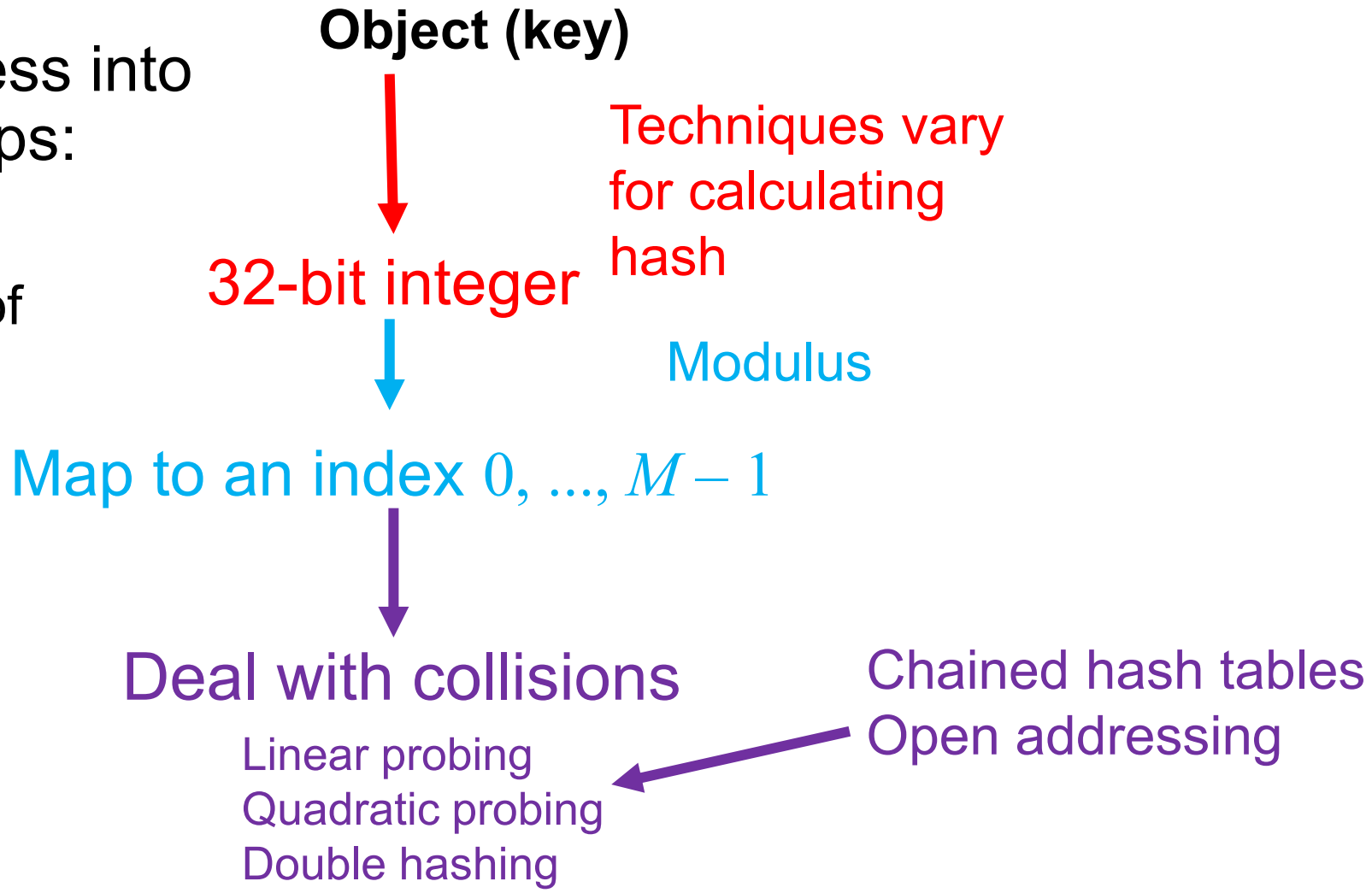
Collision

Hash tables

The hash process

We will break the process into three **independent** steps:

- We will try to get each of these down to $O(1)$



Hash Function examples

- What if the key is a string or alphanumeric (like car reg. numbers CAR 123)

- Naïve hash function:
  - Use Ascii value of 1st letter as an index.

- What is a good hash function?