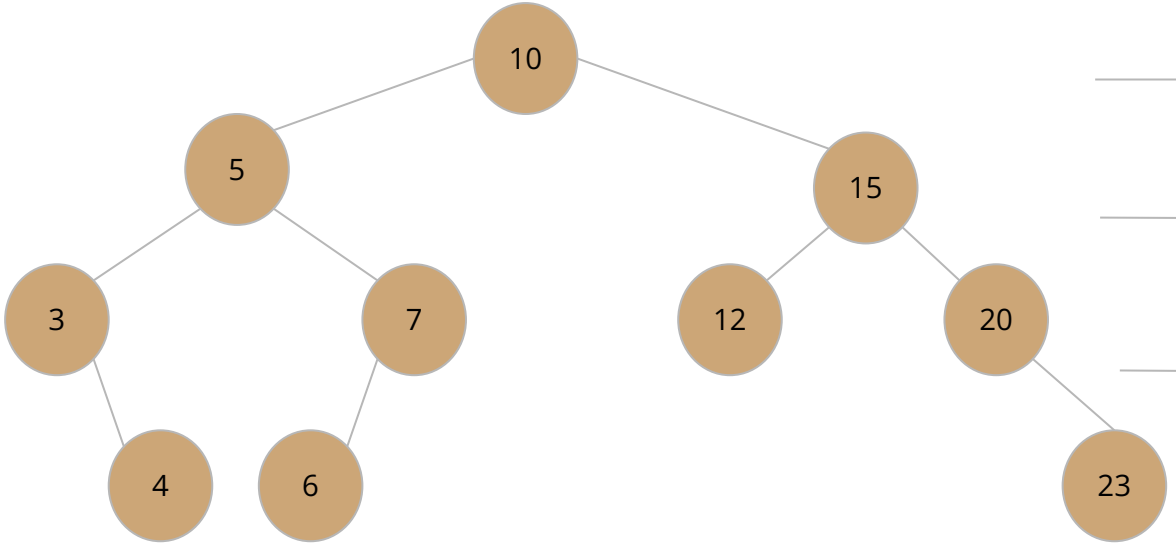
BLG 223E
Data Structures

RECITATION
8.05.2024

TREES

- Binary Search Tree(BST)
  - Values in the left subtree are smaller than the root
  - Values in the right subtree are greater than the root
- **Question**:
  - Take an array as input. (First element is root)
  - Construct a BST by inserting each element of the array.
  - Implement BFS to traverse the BST and calculate the average at each level.
  - Print the average depth of nodes at each level.

int arr[] = {10, 5, 15, 3, 7, 12, 20, 4, 6, 23};



10

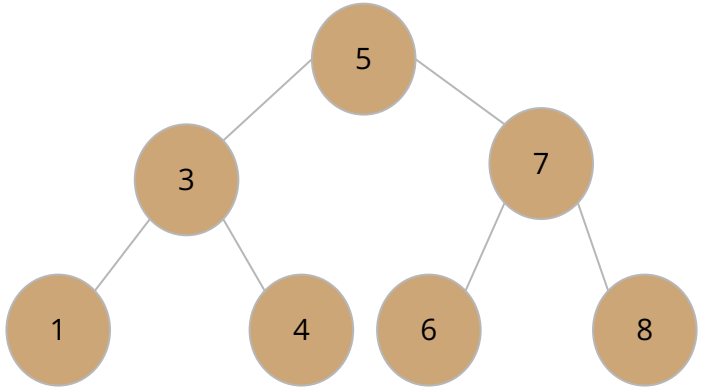
10

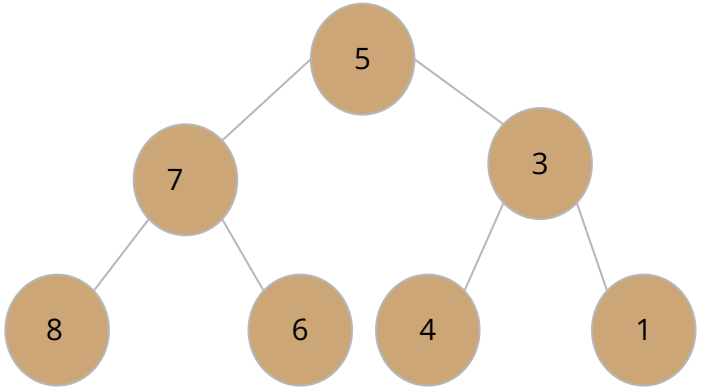
10,5

11

- Question:
  - Take an array as input.
  - Construct a BST by inserting each element of the array.
  - Create the mirror image of the constructed BST.
  - Print all levels with BFS.

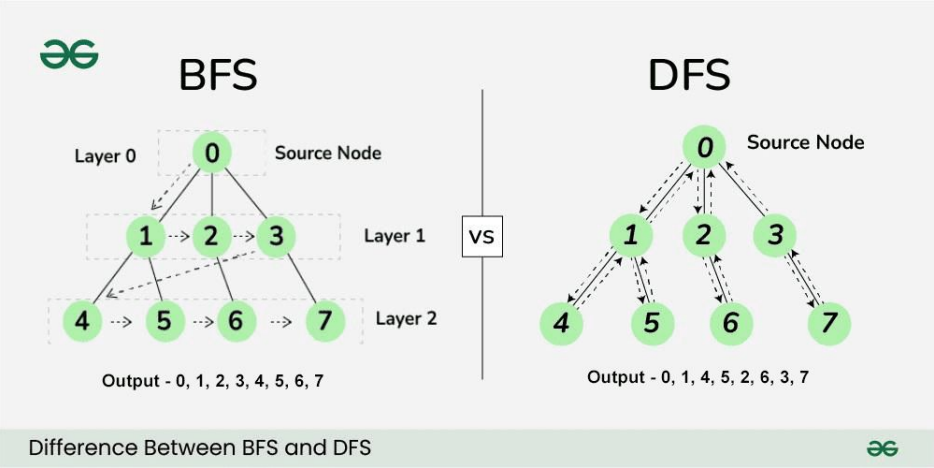
int arr[] = {5, 3, 7, 1, 4, 6, 8};

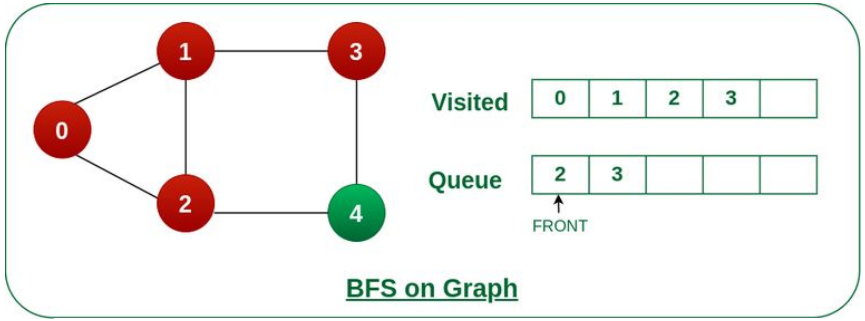


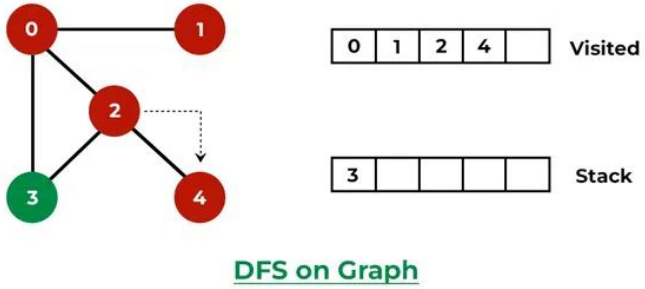


GRAPH

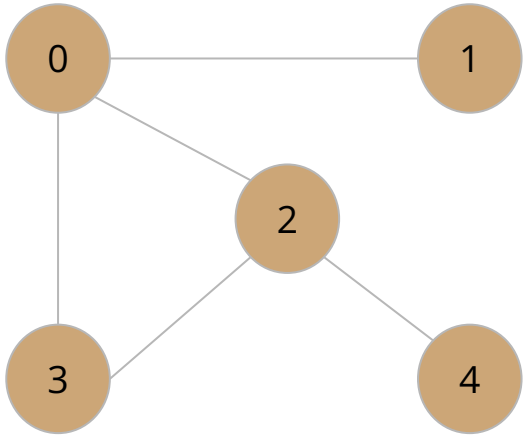
- BFS/DFS tree vs. graph





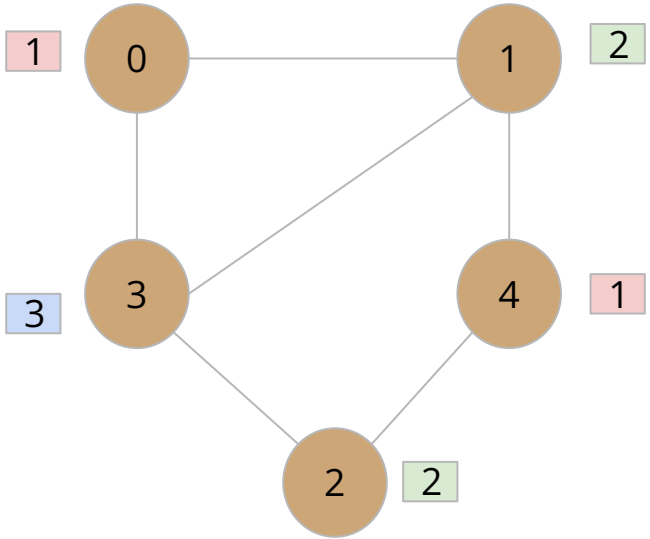


- **Question:**
  - Implement a graph using adjacency list
  - Traverse using BFS and DFS start from node 2



- **Question:**
  - Implement a graph with adjacency matrix
  - Color vertices of the graph in such a way that no two adjacent vertices have the same color. (3 colors)

```
{0, 1, 1, 0, 0},
{1, 0, 1, 1, 0},
{1, 1, 0, 0, 1},
{0, 1, 0, 0, 1},
{0, 0, 1, 1, 0}
```



MAP

- **Question:**
  - Create a map to store student IDs and scores.
  - Implement a function to display the score of a specific student by searching the map using their ID.
  - Implement a function to find and display the student with the lowest score
  - You can use STL

Polynomial hash code

- **Question:**
- (From the slides) To use position information of each letter, we may calculate $xn + a(xn-1 + a(xn-2 + \ldots + a\,(x1 + ax0)))$.
- Use a = 33
- Find the hash of “abc”
  - Result: ‘c’ + 33(‘b’ + 33 * ‘a’))
  - ASCII values: ‘a’ = 97, ‘b’ = 98, ‘c’ = 99
  - Result = 108966

Collision Handling

- **Question:**
  - To avoid collision use separate chaining
  - Define a hash function(modulus) that takes a key and the size of the hash table with a fixed size of 10 buckets.
  - Insert the following keys into the hash table: 15, 22, 33, 41, 57, 42, 45. Use the hash function to determine the bucket for each key and store them accordingly.
  - Implement an erase function that removes a key from a given bucket in the hash table.
  - Print the hash table.
  - Erase the keys 15 and 41 from the hash table.
  - Print the hash table again to verify that the keys were removed.