ACA

THE HUFFMAN HIGHWAY

BY: Anshika Agrawal (230160)

What all we covered in the project

Firstly we covered basic C++ (user input output ,data types,conditions and loops, functions), STL(vectors,pair,array) and Git.Understanding version control with Git, including basic commands for cloning repositories, committing changes, branching, and merging. After that we covered OOPS(Object oriented programming) in C++. We learnt about classes and objects, Encapsulation, Inheritance and Polymorphism,constructors and deconstructors and also about overloading. Then we covered strings in detail and linked lists. Moving on further we learned about stack and queue along with Heap Data Structure. After that we had mid term evaluation which was based on topics covered till now.

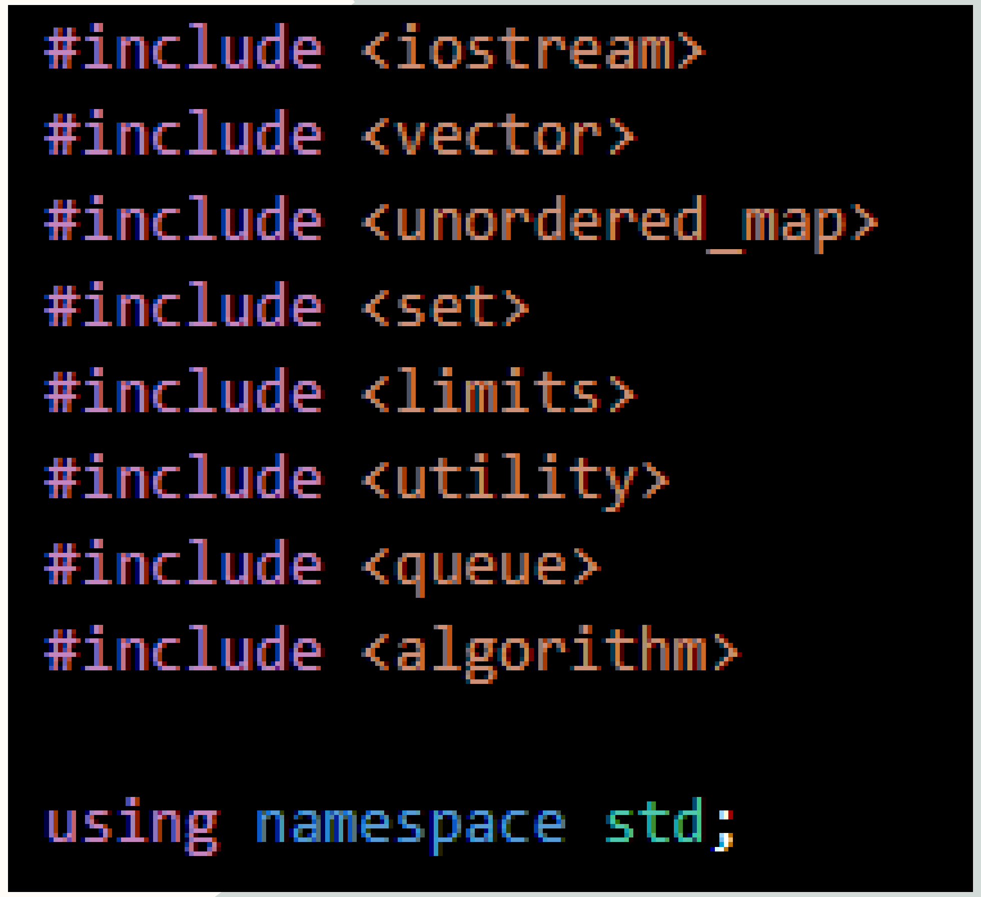
We did text decompression using Huffman Decoding and image compression using Huffman coding. Next we covered bit manipulation and graph. Then we were taught Dynamic Programming in detail with many examples.Finally for the end term evaluation we were given task to convert map of IITK to a graph with node and edges using Dijkstra Algorithm to find the shortest route between any two nodes.

Why we used Dijkstra’s Algorithm

Dijkstra's algorithm is used in this code for finding the shortest path between two nodes in a weighted graph. This is useful when you want to find the shortest path from a specified start location to any other location. It works well with graphs that have non-negative edge weights.It is efficient for graphs with a reasonable number of nodes and edges. This makes it suitable for the given map with a moderate number of locations and connections.The algorithm not only computes the shortest distances but also allows the reconstruction of the actual shortest path from the source to the destination. This is done using a previous map, which keeps track of the preceding node for each node in the shortest path.

Explanation of code

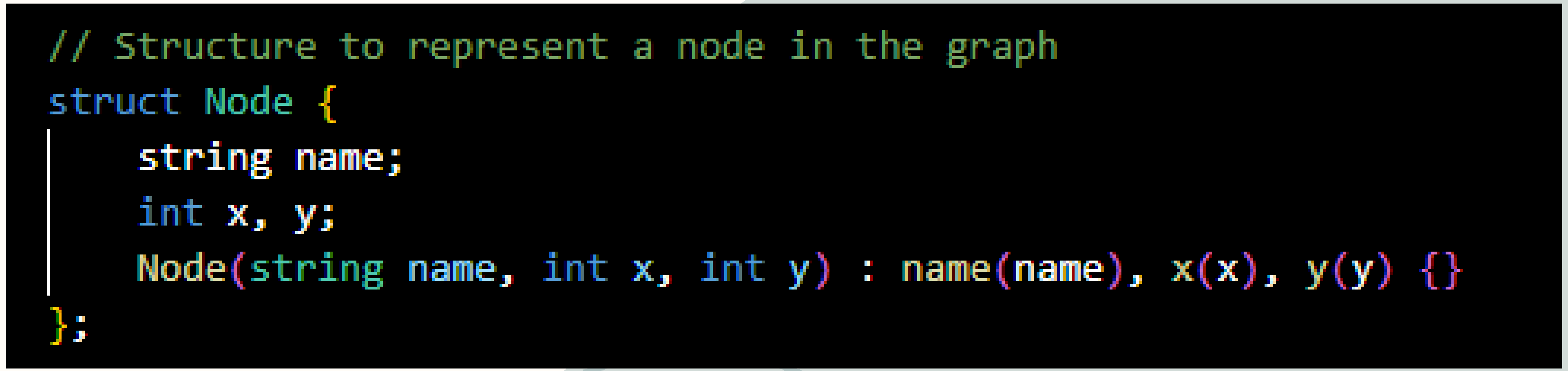
These headers include the necessary libraries for input/output operations (iostream), using vectors (vector), hash maps (unordered_map), sets (set), managing large integer values (limits), handling pairs (utility), using priority queues (queue), and using standard algorithms (algorithm).



```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <set>
#include <limits>
#include <utility>
#include <queue>
#include <algorithm>

using namespace std;
```

This structure defines a Node with a name and coordinates (x, y). However, in this particular implementation, the coordinates are not used



```cpp
// Structure to represent a node in the graph
struct Node {
    string name;
    int x, y;
    Node(string name, int x, int y) : name(name), x(x), y(y) {}
};
```

GRAPH CLASS

- “adjList” is an adjacency list that represents the graph where each key is a node, and the value is a list of pairs (neighboring node and distance).
- addEdge Function:

This function adds an edge between two nodes with a specified distance. Since the graph is undirected, edges are added in both directions.

- dijkstra Function:

Implements Dijkstra's algorithm to find the shortest path from a start node to an end node.

- Initialization:
  - distances keeps track of the minimum distance to each node from the start node.
  - previous stores the previous node in the optimal path for each node.
  - nodes is a set of pairs (distance, node) used to select the node with the smallest tentative distance.
- Algorithm Execution:
  - The algorithm iteratively selects the node with the smallest distance, updates the distances to its neighbors, and tracks the optimal path.
  - If the end node is reached, it reconstructs the path by following the previous map.
  - If the smallest distance is numeric_limits<int>::max(), the node is unreachable.

MAIN FUNCTION

- Graph Construction:

  The graph is built by adding edges between nodes with specified distances.
- User Input:

  The user is prompted to enter the start and end locations using getline.
- Path Finding:

  The dijkstra function is called with the user-provided start and end locations to find the shortest path.
- The path is printed if found, otherwise an appropriate message is displayed.

THANK YOU