Campus Navigation and Utility Planner – Report

Introduction

This project models campus buildings and routes using Trees and Graph structures. It includes BST, AVL, Graph traversals, Dijkstra shortest path, Kruskal MST, and Expression Tree evaluation.

Objectives

- Manage building data - Implement BST and AVL Trees - Build Graph (Adj List + Matrix) - BFS, DFS, Dijkstra, Kruskal - Expression Tree evaluation

Tree Structures

BST and AVL Trees store buildings by ID. Traversals help list campus locations.

Graph Implementation

Graph nodes represent buildings; edges represent weighted paths. BFS, DFS, Dijkstra, and Kruskal are implemented.

Expression Tree

Evaluates postfix expressions for sample energy billing.

Code Used

```
import java.util.*;

class Building{int id;String name,detail;Building(int i,String n,String d){id=i;name=n;detail=d;}public String toString(){re

class BSTNode{Building b;BSTNode l,r;BSTNode(Building x){b=x;}}
class BST{
 BSTNode root;
 void insert(Building x){
  if(root==null){root=new BSTNode(x);return;}
  BSTNode c=root;
  while(true){
   if(x.id<c.b.id){if(c.l==null){c.l=new BSTNode(x);return;}c=c.l;}
   else if(x.id>c.b.id){if(c.r==null){c.r=new BSTNode(x);return;}c=c.r;}
   else{c.b=x;return;}
  }
 }
 void in(BSTNode n,List<Building>a){if(n!=null){in(n.l,a);a.add(n.b);in(n.r,a);}}
 int h(BSTNode n){return n==null?0:1+Math.max(h(n.l),h(n.r));}
}

public class Main{
 public static void main(String[]a){
  Building[]b={
   new Building(0,"Admin","A"),new Building(1,"Lib","B"),
   new Building(2,"CSE","C"),new Building(3,"DS","D")
  };
  BST bst=new BST();
  for(Building x:b) bst.insert(x);
  List<Building> ino=new ArrayList<>();
  bst.in(bst.root,ino);
```

```
  System.out.println("BST Inorder: "+ino);
 }
}
```