Прізвище: Мойсак

Ім'я: Юрій

Група: КН-410

Інститут: ІКНІ

Кафедра: САПР

Дисципліна: Дискретні моделі в САПР

Перевірив: Кривий Ростислав Зіновійович

# Звіт з Лабораторної роботи № 1

# АЛГОРИТМ ПОБУДОВИ ДЕРЕВ

# Мета роботи

Метою даної лабораторної роботи є вивчення алгоритмів рішення задач побудови остових дерев.

## Короткі теоретичні відомості

**Алгоритм пошуку мінімального остового дерева**

1. Вибираємо початкову вершину і ребро з мінімальною вагою, що інцидентне вершині. Будуємо це ребро з інцидентними вершинами.
2. Вибираємо найменше по вазі ребро, з тих що залишилися в графі. Будуємо його з інцидентними вершинами.
3. Вибираємо наступне по величині ребро з найменшою вагою і перевіряємо умову: якщо ребро не створює замкнутого циклу, то будуємо його , в іншому випадку виключаємо його з розгляду.
4. Повторюємо крок 3 поки всі вершини не будуть включені в дерево або в них не залишиться сусідніх вершин

## Хід роботи

Перед початком розробки програми було побудовано графічне представлення графу (рис. 1) відповідно до такої матриці суміжності з Таблиці 1

Таблиця 1 матриця суміжності

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 0 | 0 | 3 | 0 | 0 | 0 | 34 | 0 | 80 |
| 1 | 3 | 0 | 0 | 1 | 0 | 0 | 0 | 68 |
| 2 | 0 | 0 | 0 | 0 | 23 | 0 | 12 | 0 |
| 3 | 0 | 1 | 0 | 0 | 53 | 0 | 0 | 39 |
| 4 | 0 | 0 | 23 | 53 | 0 | 0 | 68 | 14 |
| 5 | 34 | 0 | 0 | 0 | 0 | 0 | 0 | 25 |
| 6 | 0 | 0 | 12 | 0 | 68 | 0 | 0 | 99 |
| 7 | 80 | 68 | 0 | 39 | 14 | 25 | 99 | 0 |

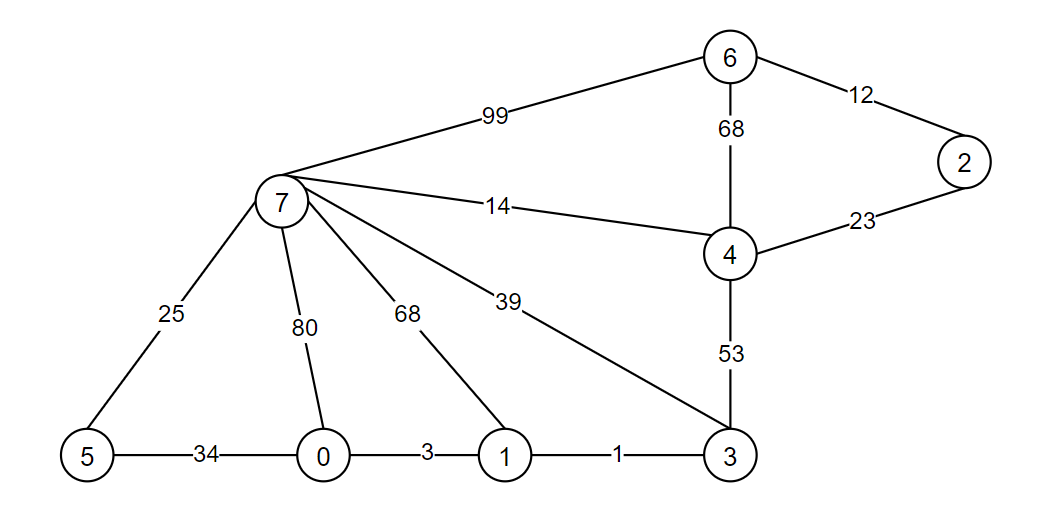
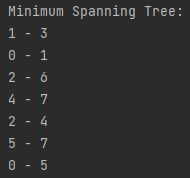


Рис. 1 Графічне представлення графу відповідно до тестових даних

**Мінімальне кістякове дерево у вигляді списку пар вершин:**

(1;3), (0;1), (2;6), (4;7), (2;4), (5;7), (0;5)



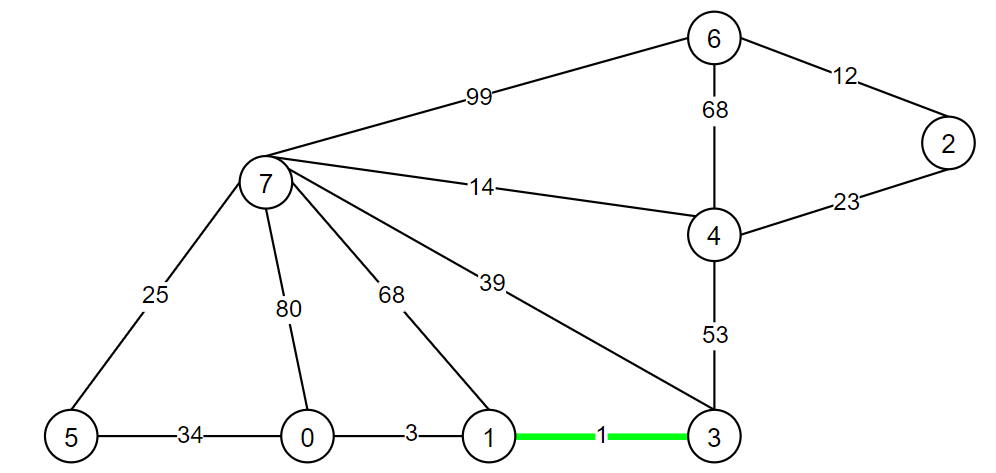


Рис. 2 Побудова кістякового графу, крок 1

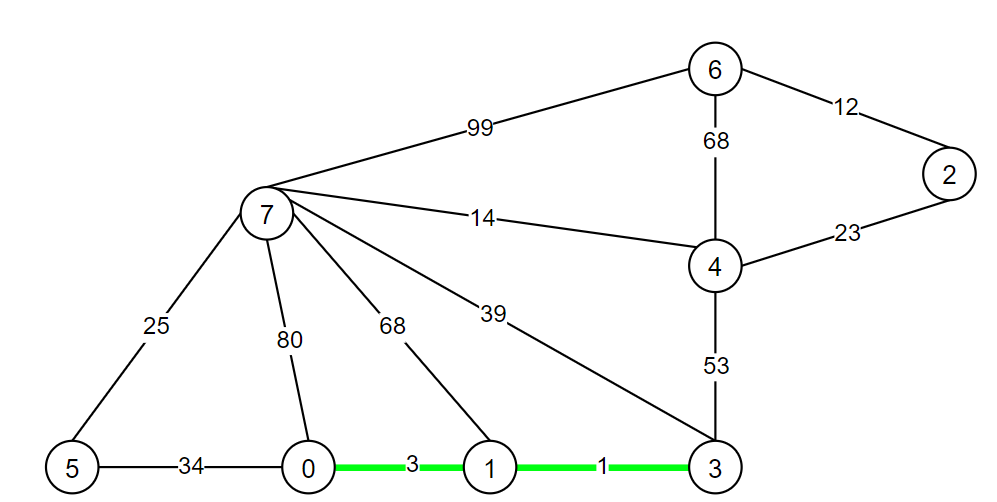


Рис. 3 Побудова кістякового графу, крок 2

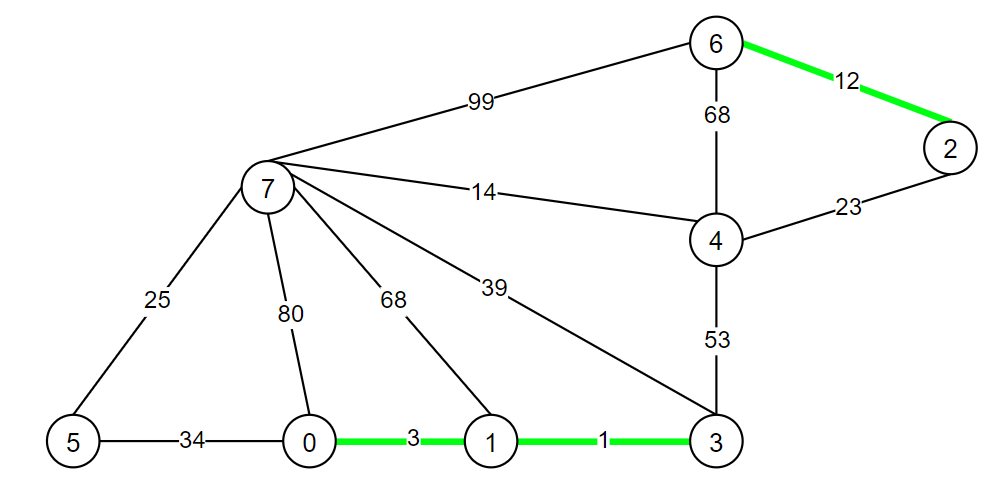


Рис. 4 Побудова кістякового графу, крок 3

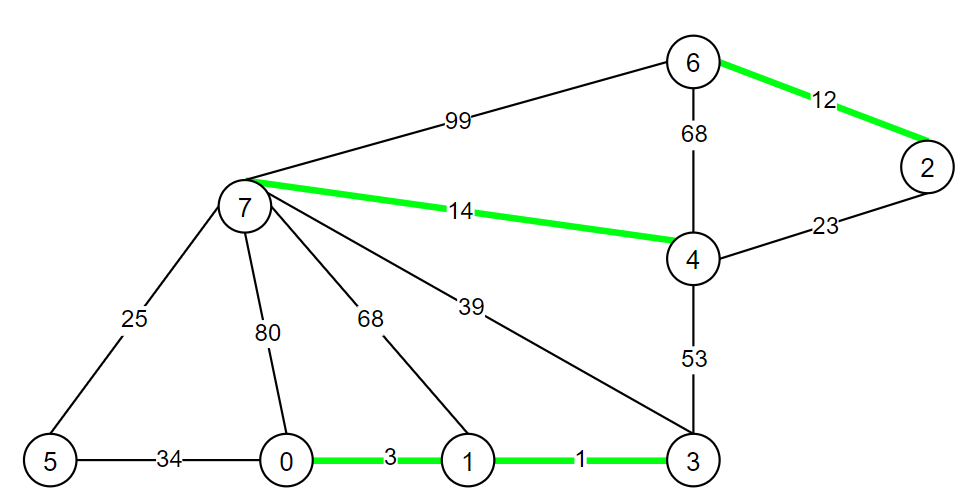


Рис. 5 Побудова кістякового графу, крок 4

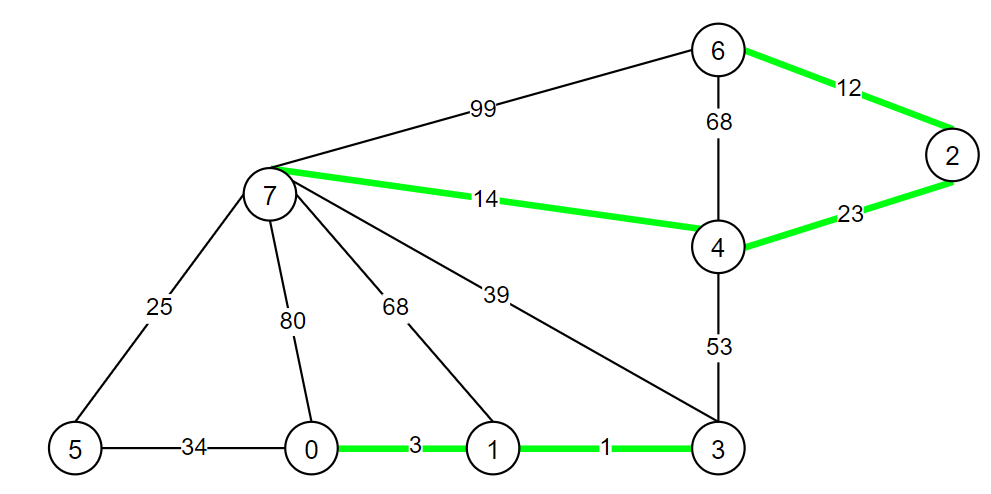


Рис. 6 Побудова кістякового графу, крок 5

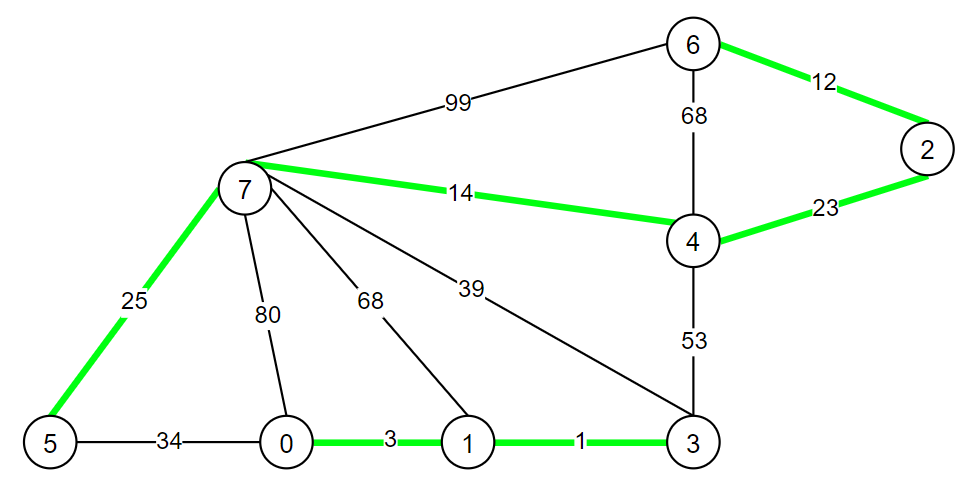


Рис. 7 Побудова кістякового графу, крок 6

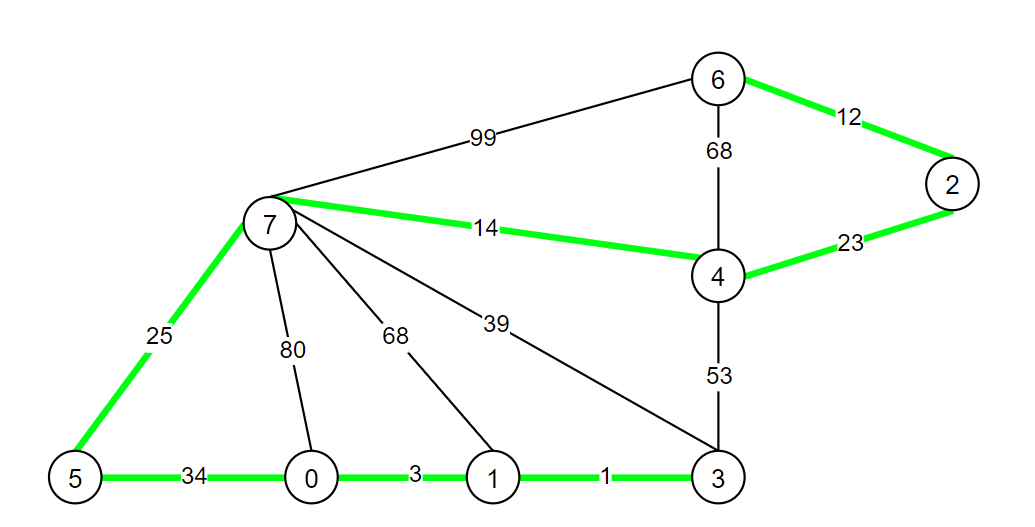


Рис. 8 Побудова кістякового графу, крок 7

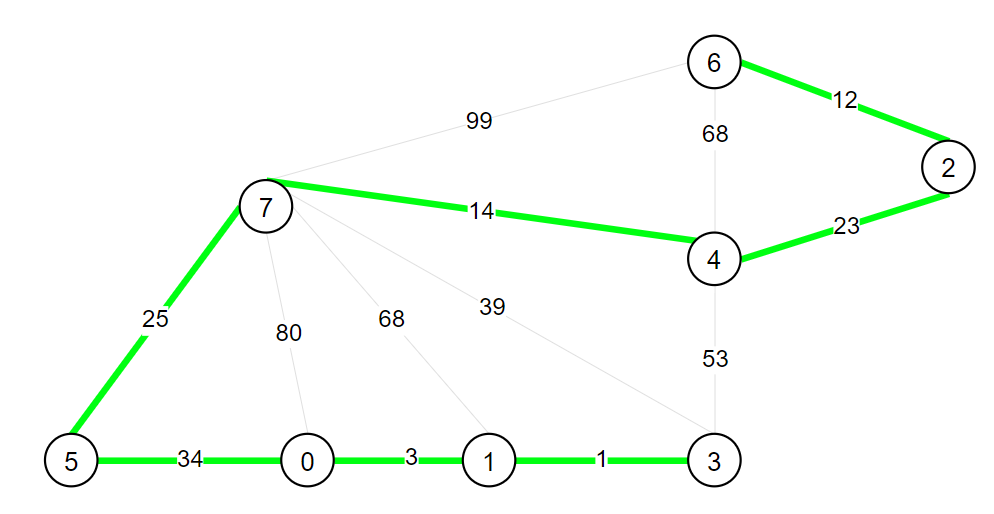


Рис. 9 Побудований кістяковий граф

## Програмна реалізація

Посилання на репозиторій з кодом до алгоритму Крускала:

///!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

public class KruskalAlgoritm {  
  
 private int[][] graph;  
 private int numVertices;  
 private int[] parent;  
  
 public KruskalAlgoritm(int[][] graph) {  
 this.graph = graph;  
 this.numVertices = graph.length;  
 this.parent = new int[numVertices];  
 Arrays.*fill*(parent, -1);  
 }  
 public int[][] findMST() {  
 int[][] mst = new int[numVertices-1][2];  
 List<Edge> edges = new ArrayList<Edge>();  
  
 for (int i = 0; i < numVertices; i++) {  
 for (int j = i+1; j < numVertices; j++) {  
 if (graph[i][j] > 0) {  
 edges.add(new Edge(i, j, graph[i][j]));  
 }  
 }  
 }  
  
   
 Collections.*sort*(edges);  
  
 int count = 0;  
 int index = 0;  
 while (count < numVertices-1 && index < edges.size()) {  
 Edge edge = edges.get(index);  
 int u = edge.u;  
 int v = edge.v;  
 int weight = edge.weight;  
 int pu = find(u);  
 int pv = find(v);  
 if (pu != pv) {  
 mst[count][0] = u;  
 mst[count][1] = v;  
 count++;  
 union(pu, pv);  
 }  
 index++;  
 }  
  
 return mst;  
 }  
  
 private int find(int u) {  
 if (parent[u] == -1) {  
 return u;  
 }  
 parent[u] = find(parent[u]);  
 return parent[u];  
 }  
  
 private void union(int u, int v) {  
 int pu = find(u);  
 int pv = find(v);  
 parent[pu] = pv;  
 }  
   
 public static void main(String[] args) {  
  
 File file = new File("src/l1\_2.txt");  
  
 GraphLoader graphLoaderObj;  
 try {  
 graphLoaderObj = new GraphLoader(file);  
 int[][] graph = graphLoaderObj.getAdjacencyMatrix();  
 KruskalAlgoritm ka = new KruskalAlgoritm(graph);  
 int[][] mst = ka.findMST();  
 System.*out*.println("Minimum Spanning Tree:");  
 for (int i = 0; i < mst.length; i++) {  
 System.*out*.println(mst[i][0] + " - " + mst[i][1]);  
 }  
 } catch (FileNotFoundException e) {  
 throw new RuntimeException(e);  
 }  
  
 }  
}

## Висновок

В результаті виконання лабораторної роботи було реалізовано алгоритм Крускала для побудови кістякового мінімального дерева та перевірено правильність його роботи, виконавши покрокову побудову кістякового дерева.