Гаар Владислав, 5130201/20102

Ниже представлены реализации алгоритмов кодирования графа кодом Прюфера и декодирования кода Прюфера.

vector<pair<int, double>> AbstractGraph::PruferEncode() {

vector<pair<int, double>> Encode;

vector<int> DegreeList(Nvertex, 0);

multimap<int, pair<int, double>> tempTree = TreeView;

vector<int> deleted(Nvertex, 1);

for (const auto& el : tempTree) {

DegreeList[el.first]++;

DegreeList[el.second.first]++;

}

cout << "--Current tree--" << endl;

for (int i = 0; i < Nvertex-1; i++) {

cout << "Iteration " << i << ":" << endl;

for (const auto& el : tempTree) {

cout << "(" << el.first << ", " << el.second.first << ") - " << el.second.second << endl;

}

// Ищем минимальную висячую вершину

int v = Nvertex;

cout << endl;

for (int j = 0; j < Nvertex; j++) {

//cout << j << ": " << DegreeList[j] << endl;

if (DegreeList[j] == 1 and j < v) {

v = j;

}

}

cout << endl;

// Кодируем вершину v

for (auto it = tempTree.begin(); it != tempTree.end(); ++it) {

if (it->second.first == v) {

Encode.push\_back(make\_pair(it->first, it->second.second));

DegreeList[v]--;

DegreeList[it->first]--;

tempTree.erase(it);

break;

}

else if (it->first == v) {

Encode.push\_back(make\_pair(it->second.first, it->second.second));

DegreeList[v]--;

DegreeList[it->second.first]--;

tempTree.erase(it);

break;

}

}

}

return Encode;

}

multimap<int, pair<int, double>> AbstractGraph::PruferDecode(const vector<pair<int, double>>& Encoded) {

// Исходный граф в виде дерева

multimap<int, pair<int, double>> Tree;

vector<pair<int, double>> tmpPrufer = Encoded;

// Список неиспользованных вершин

vector<bool> B(Nvertex, 1);

// Перебираем каждый из кодов

for (int i = 0; i < Encoded.size(); i++) {

// Ищем минимальную неиспользованную вершину, которая не встречается в коде Прюфера

for (int j = 0; j < Nvertex; j++) {

// Если вершина не использована

if (B[j]) {

bool flag = 1; // По умолчанию вершина подходит

// Ищем вершину в коде Прюфера

for (int k = i; k < tmpPrufer.size(); k++) {

// Если текущая вершина есть в коде Прюфера, переходим к следующей

if (Encoded[k].first == j) {

flag = 0;

break;

}

}

// Если вершина подходит, она и есть минимальная

if (flag) {

Tree.insert(make\_pair(Encoded[i].first, make\_pair(j, Encoded[i].second)));

B[j] = 0;

break;

}

}

}

}

return Tree;

}