СОДЕРЖАНИЕ ОТЧЕТА

1 Как определить, есть ли в графе цикл?

```
void HasCycle(Route &current, Set<int> &vertices, int n, int initial_v, bool
&found) const {
    Route t = GetVertexEdges(current.back());
    Set<int> route = ToSet<int>(t.begin(), t.end());
    Set<int> not_included;
    std::set_difference(route.begin(), route.end(),
                        vertices.begin(), vertices.end(),
                        std::inserter(not_included, not_included.begin()));

    for (const auto &vertex: not_included) {
        current.push_back(vertex);
        if (vertex == initial_v) {
            if (current.size() > 3) {
                found = true;
            }
        } else {
            vertices.insert(vertex);
            HasCycle(current, vertices, n, initial_v, found);
            vertices.erase(vertex);
        }

        current.pop_back();
        if (found) {
            return;
        }
    }
}

[[nodiscard]] bool HasCycle(const int origin) const {
    Route w{origin};
    Set<int> vertices{};
    bool found = false;
    HasCycle(w, vertices, this->size(), origin, found);

    return found;
}

[[nodiscard]] bool HasCycle() {
    for (auto &vertex: _vertices) {
        if (HasCycle(vertex.first)) {
            return true;
        }
    }

    return false;
}
```

```
int main() {
    List<Graph> graphs{Graph{{
                                Vertex{1, {2}},
                                Vertex{2, {3}},
                                Vertex{3, {4}},
                                Vertex{4, {5}},
                                Vertex{5, {4}},
                                Vertex{6, {7}},
                                Vertex{7, {1}}}},
                        Graph{{
                                {1, {2, 4, 6, 7}},
                                {2, {1, 3, 4}},
                                {3, {2, 4, 6, 5}},
                                {4, {1, 2, 3}},
                                {5, {3, 6}},
                                {6, {7, 1, 3, 5}},
                                {7, {6, 1}}}}

    };

 std::cout << std::boolalpha << graphs[0].HasCycle() << "\n" <<
graphs[1].HasCycle() << "\n";

    return 0;
}
```

```
false
true

Process finished with exit code 0
```

В качестве основы был взят алгоритм генерации всех циклов