ГУО "Белорусский государственный университет
информатики и радиотехники"
Факультет информационных технологий и управления
Кафедра интеллектуальных информационных технологий

Расчетная работа

«Графы. Решение теоретико-графовой задачи»

Подготовил:
Криващёкая А.
Гр.421702
Проверил:
Малиновская Н. В.

Минск 2024

Цель работы:

- Ознакомиться с понятием графов.
- Выяснить, какие виды графов бывают.
- Ознакомиться со способами представления графов в памяти компьютера.
- Научиться решать теоретико-графовые задачи.

Задачи:

- Реализовать алгоритм на языке программирования C++, решающий теоретико-графовую задачу в соответствии с выданным вариантом. Программа должна соответствовать следующим требованиям:
  - Для представления графа должна использоваться структура данных, соответствующая выданному варианту.
  - Использование глобальных переменных в программе недопустимо.
- Проверить данный алгоритм на корректность, написав для этого минимум 5 тестов.

Вариант:

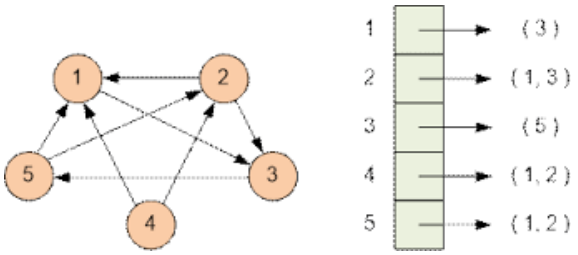
Мой вариант - 2.1 руководства. Структура данных — список смежности. Необходимо найти радиус графа.

Список ключевых понятий:

- Граф – математическая абстракция реальной системы любой природы, объекты которой обладают парными связями.
- Граф бывает ориентированным и неориентированным:
  - Ориентированный граф – граф, рёбрам которого присвоено направление.
  - Неориентированный граф – граф, ни одному ребру которого не присвоено направление.
- Граф бывает взвешенным и невзвешенным:
  - Взвешенный граф – граф, каждому ребру которого поставлено в соответствие некое значение (вес ребра).
  - Невзвешенный граф – вес рёбер не имеет значения и обычно в математических задачах берётся равным за единицу.
- Эксцентриситет вершины графа – расстояние до самой дальней вершины графа.
- Радиус графа – минимальный из эксцентриситетов вершин связного графа.

Алгоритмы на графах:

- Список смежности – один из способов представления графа в виде коллекции списков вершин.



- Поиск в ширину (англ. breadth-first search, BFS) — один из методов обхода графа.
  Пусть задан граф G=(V,E) и выделена исходная вершина s. Алгоритм поиска в ширину систематически обходит все ребра G ля «открытия» всех вершин, достижимых из s, вычисляя при этом расстояние (минимальное количество рёбер) от s до каждой достижимой из s вершины. Алгоритм работает как для ориентированных, так и для неориентированных графов.
  В коде (C++) поиск в ширину можно реализовать так:

```cpp
vector<int> BFS(int start) {
    vector<int> distance(V, INT_MAX);
    queue<int> q;
    q.push(start);
    distance[start] = 0;

    while (!q.empty()) {
        int v = q.front();
        q.pop();
        for (int neighbor : adj[v]) {
            if (distance[neighbor] == INT_MAX) {
                distance[neighbor] = distance[v] + 1;
                q.push(neighbor);
            }
        }
    }
    return distance;
}
```

Алгоритм решения задачи в C++:

1. Задаём количество вершин и рёбер графа. Вводим рёбра смежности.

```
cout << "Количество вершин:\n";
cin >> n;
Graph g(n);

cout << "Количество рёбер:\n";
cin >> m;

cout << "Введите ребра смежности: \n";
for (int i = 0; i < m; i++) {
    cin >> x;
    cin >> y;
    g.addEdge(x, y);
}
```

2. Выводим граф в виде списка смежности.

```
void printGraph() {
    for (int i = 0; i < V; i++) {
        cout << "Вершина " << i << ": ";
        for (int j : adj[i]) {
            cout << j << " ";
        }
        cout << endl;
    }
}
```

3. Ищем радиус графа.

```
int findRadius() {
    int radius = INT_MAX;

    for (int i = 0; i < V; i++) {
        vector<int> distances = BFS(i);
        int maxDistance = 0;

        for (int d : distances) {
            if (d != INT_MAX) {
                maxDistance = max(maxDistance, d);
            }
        }

        if (maxDistance < radius) {
            radius = maxDistance;
        }
    }
    return radius;
}
```

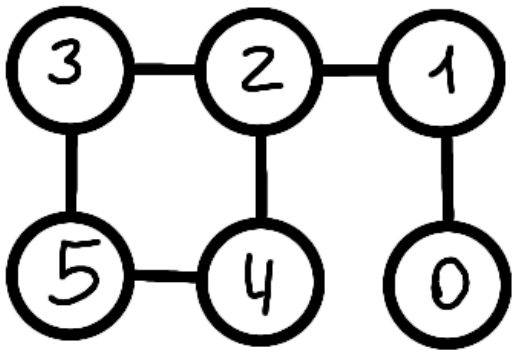
4. Выводим радиус графа.

```
int radius = g.findRadius();
cout << "Радиус графа: " << radius << endl;
```

5. Завершение программы.

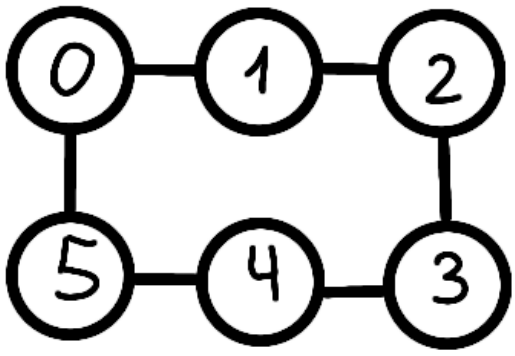
Тестовые примеры:

1. Пример:



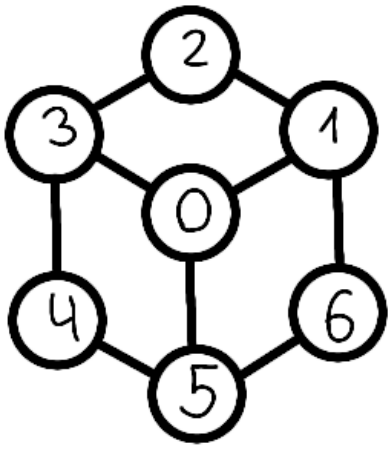

```
Количество вершин:
6
Количество рёбер:
6
Введите ребра смежности:
0 1
1 2
2 3
2 4
3 5
4 5
Вершина 0: 1
Вершина 1: 0 2
Вершина 2: 1 3 4
Вершина 3: 2 5
Вершина 4: 2 5
Вершина 5: 3 4
Радиус графа: 2
```

2. Пример:



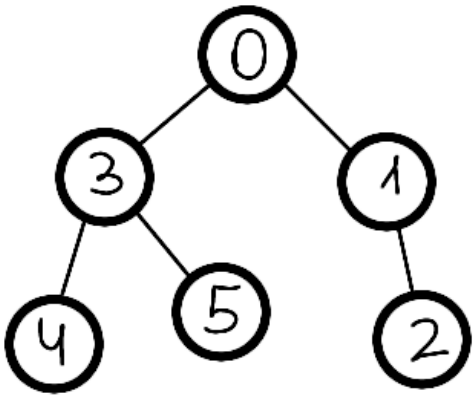

```
Количество вершин:
6
Количество рёбер:
6
Введите ребра смежности:
0 1
1 2
2 3
3 4
4 5
5 0
Вершина 0: 1 5
Вершина 1: 0 2
Вершина 2: 1 3
Вершина 3: 2 4
Вершина 4: 3 5
Вершина 5: 4 0
Радиус графа: 3
```

3. Пример:



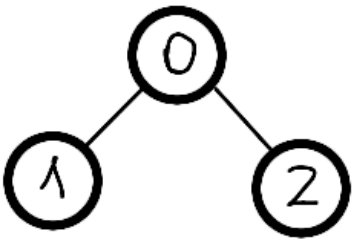

```
Количество вершин:
7
Количество рёбер:
9
Введите ребра смежности:
0 1
0 3
0 5
1 2
3 2
3 4
4 5
5 6
1 6
Вершина 0: 1 3 5
Вершина 1: 0 2 6
Вершина 2: 1 3
Вершина 3: 0 2 4
Вершина 4: 3 5
Вершина 5: 0 4 6
Вершина 6: 5 1
Радиус графа: 2
```

4. Пример:




```
Количество вершин:
6
Количество рёбер:
5
Введите ребра смежности:
2 1
1 0
0 3
3 4
3 5
Вершина 0: 1 3
Вершина 1: 2 0
Вершина 2: 1
Вершина 3: 0 4 5
Вершина 4: 3
Вершина 5: 3
Радиус графа: 2
```

5. Пример:




```
Количество вершин:
3
Количество рёбер:
2
Введите ребра смежности:
1 0
0 2
Вершина 0: 1 2
Вершина 1: 0
Вершина 2: 0
Радиус графа: 1
```

Преобразование входной конструкции графа в выходные данные (радиус графа) на примере 1 :

Для нахождения радиуса графа, требуется выполнить обход в ширину (BFS) от каждой вершины, чтобы определить максимальное расстояние от данной вершины до всех других вершин. Радиус графа – это минимальное из всех этих максимальных расстояний.

1. Инициализация данных:
   - Список смежности графа.
   - Значение радиуса графа.
2. Обход в ширину (BFS):

   Для каждой вершины запускаем BFS для подсчета максимального расстояния до всех остальных.
3. Рассмотрим каждый шаг:
   - Обход от вершины 0:
     - Начинаем с вершины 0, посещаем 1.
     - Далее от 1 посещаем 2.
     - Достигнув 2, можем посетить 3 и 4.
     - Затем от 3 до 5. Максимальное расстояние: 3.
     - Значению радиуса графа присваиваем: 3.
   - Обход от вершины 1:
     - Начинаем с 1, достигаем 0, затем 2, затем 3 и 4.
     - Максимальное расстояние: 3.
     - Сравниваем это максимальное расстояние с прошлым.
     - Значению радиуса графа присваиваем: 3.
   - Обход от вершины 2:
     - С 2 достигаем 1, 3 и 4.
     - Далее от 3 до 5. Максимальное расстояние: 2.
     - Сравниваем это максимальное расстояние с прошлым.
     - Значению радиуса графа присваиваем: 2.
   - Обход от вершины 3:
     - С 3 посещаем 2 и 5. Максимальное расстояние: 2.
     - Сравниваем это максимальное расстояние с прошлым.
     - Значению радиуса графа присваиваем: 2.
   - Обход от вершины 4:
     - С 4 посещаем 2 и 5. Максимальное расстояние: 2.
     - Сравниваем это максимальное расстояние с прошлым.
     - Значению радиуса графа присваиваем: 2.
   - Обход от вершины 5:
     - С 5 можем достичь 3 и 4. Максимальное расстояние: 3.
     - Сравниваем это максимальное расстояние с прошлым.
     - Значению радиуса графа присваиваем: 2.
4. Выводим значение радиуса графа

Вывод:

В ходе выполнения данной расчётной работы я:

- Ознакомился с понятием графов.
- Выяснил, какие виды графов бывают.
- Ознакомился с таким способом представления графов, как список смежности.
- Реализовал алгоритм решения задачи 2.1 руководства на языке программирования C++ с использованием списка смежности.
- Проверил данный алгоритм на корректность с помощью пяти придуманных мною тестов.

Использованные источники:

1. Сайт "Habr"[Электронный ресурс] — Режим доступа: https://habr.com/ru/articles/661577/
2. Сайт "Олимпиадное программирование в Бресте и Беларуси"[Электронный ресурс]- режим доступа: https://brestprog.by/topics/bfs
3. Сайт "Олимпиадное программирование в Бресте и Беларуси"[Электронный ресурс]- режим доступа: https://brestprog.by/topics/graphs/