МГТУ им. Баумана

Типы и структуры данных

Лабораторная работа №7. "Графы"

Студент Ивахненко Дмитрий - ИУ7-36Б

Описание условия задачи

Найти все вершины графа, к которым от заданной вершины можно добраться по пути не длиннее A.

Техническое задание

Входные данные:

Программа ожидает:

- название файла с кол-вом вершин и матрицей весов графа
- максимальную длину пути (неотрицательное целое число)
- вершина, из которой осуществляется поиск пути (вершина - неотрицательное целое число)

Выходные данные:

Результатом работы программы являются:

1. Граф в графическом виде
2. Массив расстояний во все вершины от заданной (с указанием, подходит ли расстояние под условие задачи)

Обращение к программе:

Запускается из терминала при помощи комманды `./bin/app "название_файла"`.

Аварийные ситуации:

1. ошибка в названии файла или его отсутствие (физическое и/или в аргументах командной строки);
2. ввод некорректной вершины начала поиска и/или максимальной длины пути

Во всех указанных случаях программа сообщит об ошибке

Структуры данных

Граф представлен в виде таблицы весов. Такое представление очень удобно при использовании алгоритма Дейкстры, который как раз используется при решении поставленной задачи.

```c
typedef struct {
    int **matrix;   // Матрица весов графа, считываемая из файла
    int size;       // Кол-во вершин графа
} Graph;
```

Описание основных функций

```c
/* Алгоритм Дейкстры */
int *Dijkstra(Graph *graph, int start) {
    int n = graph->size;

    int *dist = malloc(sizeof(int) * n);
    if (!dist)
        return NULL;

    bool visited[n];

    for (int i = 0; i < n; i++)
        dist[i] = INF, visited[i] = false;

    dist[start] = 0;

    for (int count = 0; count < n - 1; count++) {
        int u = minVertex(dist, visited, n);
        visited[u] = true;

        for (int v = 0; v < n; v++) {
            if (!visited[v] && graph->matrix[u][v] != INF && dist[u] != INF
                && dist[u] + graph->matrix[u][v] < dist[v])
                dist[v] = dist[u] + graph->matrix[u][v];
        }
    }

    return dist;
}
```

Алгоритм

- Каждой вершине сопоставим метку — минимальное известное расстояние от этой вершины до **start**.
- Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.
- Работа алгоритма завершается, когда все вершины посещены.

Инициализация.

- Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности. Это отражает то, что расстояния от a до других вершин пока неизвестны.

- Все вершины графа помечаются как непосещённые.

Шаг алгоритма.

- Если все вершины посещены, алгоритм завершается.
- В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку.
- Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом.
- Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим шаг алгоритма.

Завершение выполнения алгоритма.

- Алгоритм заканчивает работу, когда все вершины посещены.
- Если в какой-то момент все непосещённые вершины помечены бесконечностью, то это значит, что до этих вершин нельзя добраться (то есть граф несвязный). Тогда алгоритм может быть завершён досрочно.

Оценка по времени

Исходя из описания алгоритма, мы можем сделать вывод, что сложность алгоритма составляет: O(n^2).

Оценка по памяти

Память в данном алгоритме затрачивается на хранение графа, массива с ответом и булевого массива посещенных вершин

Вывод

Для решения поставленной задачи был реализован алгоритм Дейкстры для поиска кратчайших путей в графе. Выбор обоснован простотой реализации алгоритма, и соответствием поставленной задаче, поскольку в графе нет отрицательных ребер.

Этот алгоритм может быть использован, например, в системе навигации для поиска оптимального пути к, например, ресторану: вершина, из которой осуществляется поиск представляет собой местоположение пользователя, а остальные вершины - местоположения ближайших ресторанов и промежуточные пункты.

Программа подберет рестораны, находящиеся неподалеку.

Ответы на вопросы

1. Что такое граф?

Граф - это конечное множество вершин и ребер, соединяющих их.

2. Как представляются графы в памяти?

Графы могут быть представлены в виде таблицы смежности, списков достижимых вершин из каждой вершины, массива ребёр.

3. Какие операции возможны над графами?

- поиск вершин в графе
- поиск кратчайших путей от Vk до Vm
- поиск Эйлерова пути
- поиск Гамильтонова пути
- поиск кратчайших путей между всеми вершинами

4. Какие способы обхода графов существуют?

Поиск (обход) в глубину, в ширину.

5. Где используются графовые структуры?

Схемы авиалиний, схемы дорог.

6. Какие пути в графе Вы знаете?

Простой путь, гамильтонов, эйлеров, замкнутый.

7. Что такое каркасы графа?

Каркас графа - подграф данного графа, с тем же числом вершин, что и у исходного графа. Он получается из исходного графа удалением максимального числа рёбер, входящих в циклы, но без нарушения связности графа