Типы и структуры данных — Сиденко Анастасия
Лабораторная работа 7
Вариант 11 — ИУ7-33Б
Графы

Техническое задание:

Задача:

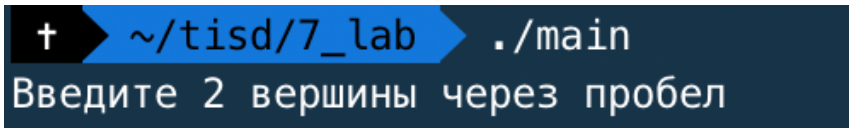
Задана система двусторонних дорог. Определить, можно ли, закрыв какие-нибудь три дороги, добиться того, чтобы из города A нельзя было попасть в город B.

Входные данные: файл с числами.

Выходные данные: граф и ответ на вопрос.

Взаимодействие с программой

Взаимодействие через консольное меню.



Аварийные ситуации:

1. Некорректные данные. Вывод сообщения

Используемые структуры:

1. Матрица смежности

int **matrix

2. Список смежности

```
struct spisok {
    int top;
    int data;
    struct spisok *next;
};
```

Алгоритмы:

1. Для матрицы

- Инициализация
  int **init_matrix(int n)
- Освобождение памяти
  void free_matrix(int **ukazat)
- Считывание кол-ва из файла
  int read_kol(FILE *f, int *n)
- Считывание матрицы из файла
  int **allocate_matrix(const char *argv, int *n)
- Вывод в консоль
  int write_matrix(const int n, int **matrix)
- Поиск пути
  int find_way(int kol, int **matrix, int start, int end)
- Возможность удаления
  int delete_ways(int kol, int **matrix, int start, int end)

2. Для списка

- Печать
  void list_print(struct spisok *head)
  void spisok_print(struct spisok *table[], int n)
- Добавление элемента
  struct spisok *list_push(struct spisok *st, int data, int top)
- Инициализация и считывание
  void list(const char *argv, struct spisok *st[])
- Поиск пути
  int find_way_sp(int kol, struct spisok *table[], int start, int end)
- Удаление
  struct spisok* delete_list(struct spisok *head, struct spisok *top)
- Доступ по номеру
  struct spisok *number_list(struct spisok *list, int n)
- Проверка
  int check(int kol, int start, int end, struct spisok *table[], int i, struct spisok *bufi, int k, struct spisok *bufk, int n, struct spisok *bufn)
- Количество элементов в списке
  int kol_list(struct spisok *list)
- Возможность удаления
  int delete_ways_sp(int kol, struct spisok *table[], int start, int end)

Тесты:

1. Некорректные данные

Ввод некорректных данных

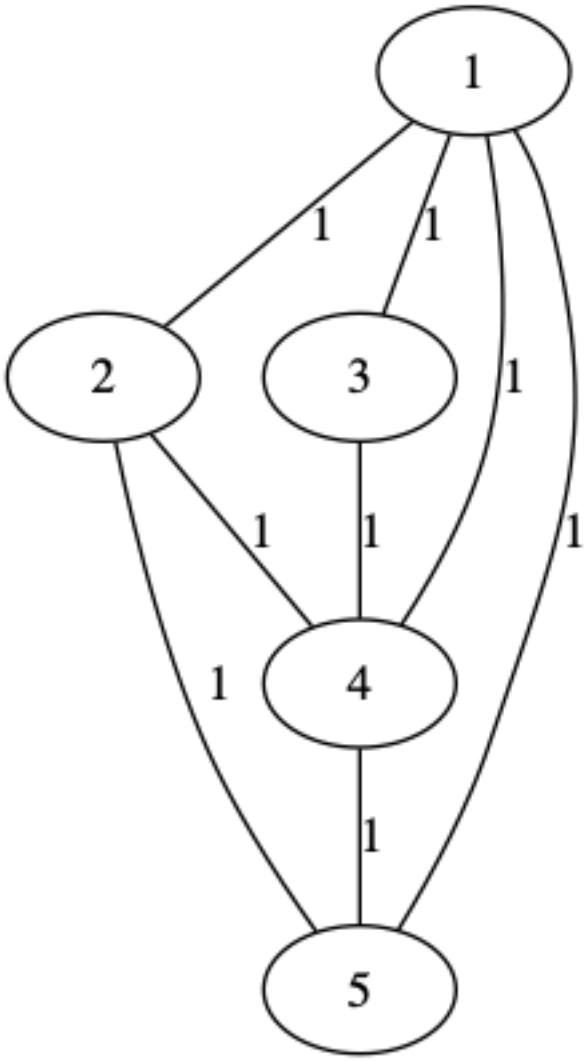
```
Введите 2 вершины через пробел
q
Некорректный ввод попробуйте еще раз
```

```
Введите 2 вершины через пробел
7
1
Некорректный ввод попробуйте еще раз
```

2. Правильные тесты

```
Введите 2 вершины через пробел
4 1
   0    1    1    1    1
   1    0    0    1    1
   1    0    0    1    0
   1    1    1    0    1
   1    1    0    1    0
[1] = 5 4 3 2
[2] = 5 4 1
[3] = 4 1
[4] = 5 3 2 1
[5] = 4 2 1
Нельзя удалить
```

```
Введите 2 вершины через пробел
2 3
   0    1    1    1    1
   1    0    0    1    1
   1    0    0    1    0
   1    1    1    0    1
   1    1    0    1    0
[1] = 5 4 3 2
[2] = 5 4 1
[3] = 4 1
[4] = 5 3 2 1
[5] = 4 2 1
Можно удалить
```



Сравнение времени работы.

```
Введите 2 вершины через пробел
1 3
   0    1    1    1    1
   1    0    0    1    1
   1    0    0    1    0
   1    1    1    0    1
   1    1    0    1    0
32
[1] = 5 4 3 2
[2] = 5 4 1
[3] = 4 1
[4] = 5 3 2 1
[5] = 4 2 1
yes
7
Можно удалить
```

Вопросы:

1. Что такое граф?
   Граф – это конечное множество вершин и ребер, соединяющих их, т. е.: G = < V,E >, где V – конечное непустое множество вершин; E – множество ребер (пар вершин).

2. Как представляются графы в памяти?
   Графы в памяти могут представляться различным способом. Один из видов представления графов – это матрица смежности B(n*n); В этой матрице элемент b[i,j] != 0, если ребро, связывающее вершины Vi и Vj существует и b[i,j]=0, если ребра нет.
   Граф можно хранить в виде так называемого списка смежностей. Список смежностей содержит для каждой вершины из множества вершин V список тех вершин, которые непосредственно связаны с этой вершиной. Каждый элемент (ZAP[u]) списка смежностей является записью, содержащей данную вершину и указатель на следующую запись в списке (для последней записи в списке этот указатель – пустой).

3. Какие операции возможны над графами?
   - поиск кратчайшего пути от одной вершины к другой (если он есть);
   - поиск кратчайшего пути от одной вершины ко всем другим;
   - поиск кратчайших путей между всеми вершинами;
   - поиск эйлерова пути (если он есть);
   - поиск гамильтонова пути (если он есть).

4. Какие способы обхода графов существуют?
   Один из основных методов проектирования графовых алгоритмов – это поиск (или обход графа) в глубину (depth first search, DFS), при котором, начиная с произвольной вершины v0, ищется ближайшая смежная вершина v, для которой, в свою очередь, осуществляется поиск в глубину (т.е. снова ищется ближайшая, смежная с ней вершина) до тех пор, пока не встретится ранее просмотренная вершина, или не закончится список смежности вершины v (то есть вершина полностью обработана).
   Другой метод обхода графа – поиск в ширину (breadth first search, BFS). Обработка вершины v осуществляется путем просмотра сразу всех новых соседей этой вершины. При этом полученный путь является кратчайшим путем из одной вершины в другую.

5. Где используются графовые структуры?
   Коммуникационные линии между городами (карты города, метро).

6. Какие пути в графе Вы знаете?
   Произвольный путь в графе, проходящий через каждое ребро графа точно один раз, называется эйлеровым путем.
   Путь в графе, проходящий в точности один раз через каждую вершину графа (а не каждое ребро) и соответствующий цикл называются гамильтоновым путем.

7. Что такое каркасы графа?
   Каркас - некоторый поджар графа.

Вывод:

Списки использовать эффективнее по времени. Так как в списках мы храним не все пути, а только существующие засует этого количество просмотров уменьшается.

Нам заранее известно число вершин и путей, поэтому матрицу смежности мы создаем на уже ограниченное количество элементов, список же мы динамически расширяем. В зависимости от количества путей графа эффективность будет различной. Чем больше путей тем менее эффективно использование списков по памяти.

Используемый мной алгоритм Дейкстры, его сложность n^2.