

## Тема 8. Графы

### Обходы, поиск

Граф — множества вершин и ребер.

Ориентированный и неориентированный графы.

Есть разные другие свойства (планарный, двудольный ...) — мы в теорию графов не лезем. Просто знакомство с некоторыми алгоритмами и намеки по реализации.

Представления графов.

1. Матрица инцидентности.  $a_{i,j}$  — соответствует ребру из вершины  $i$  в вершину  $j$ .

Для неориентированного графа можно рассматривать треугольную часть.

2. По принципу ссылочной реализации. Вершина — узел со списком (массивом) соседей.

Достоинства и недостатки каждого способа очевидны.

#### Базовые алгоритмы.

Поиск конкретной вершины.

Обнаружение циклов.

Поиск одного или всех путей из вершины в вершину.

Поиск пути с минимальным (максимальным) весом.

Рассматриваем два фундаментальных алгоритма. Поиск в ширину и поиск в глубину.

**Поиск в ширину.** Дана вершина  $A$ . Определить на каком расстоянии (по числу ребер) находится от нее вершина  $B$ .

Пускаем из вершины  $A$  волну, которая за один шаг продвигается на расстояние одного ребра. Каждый шаг распространения волны отмечается номерами 0, 1, 2, на вершинах, находящихся на фронте волны.

$M$  — граничное множество.  $x(k)$  — вершина  $x$  имеет метку  $k$

**шаг 0.** Поставить во всех вершинах метку  $-1$  (не посещалась).  $M = \emptyset$ .

**шаг 1.** Пометить  $A$  меткой 0 (т.е.  $A(0)$ ). Добавить  $A$  в  $M$ .

**шаг 2.** Пусть  $k$  есть метка всех вершин из  $M$ .

```

    Для каждой вершины  $x(k)$  из  $M$  {
        для каждого соседа  $y$  вершины  $x$  {
            если  $y(-1)$  то {
                 $y(k+1)$ 
                добавить  $y$  в  $M$ 
            }
        }
        удалить  $x$  из  $M$ 
    }
    утв: Все вершины в  $M$  имеют метку  $k+1$ 
    
```

**шаг 3.** Если на шаге 2 вершина  $B$  добавлена в  $M$ , то конец, иначе перейти к шагу 2.

Отдельно можно рассмотреть случай несвязных компонент.

Метка вершины  $B$  и есть ее расстояние от  $A$ .

#### Обнаружение циклов и поиск путей.

**шаг 2.** Пусть  $k$  есть метка всех вершин из  $M$ .

```

Для каждой вершины  $x(k)$  из  $M$  {
  для каждого соседа  $y$  вершины  $x$  {
    если  $y(-1)$  то {
       $y(k+1)$ 
      добавить  $y$  в  $M$ 
    } иначе {
      есть цикл неориентированного графа
      (столкнулись два фронта волны)
    }
  }
  удалить  $x$  из  $M$ 
}

```

Путь из  $A$  в  $B$  определяется из следующих соображений. У вершины с меткой  $k$  должен быть хотя бы один сосед с меткой  $k - 1$ , у этого соседа — сосед с меткой  $k - 1$  и так до  $A(0)$ .

Сложность: количество вершин + количество ребер

**Поиск в глубину.** Знакомый нам рекурсивный обход, стартуя из данной вершины.

Чтобы не попасть на цикл, красим вершины по ходу дела в три цвета — белый, серый, черный:

белый — не была посещена;

серый — посещена на проходе вниз;

черный — окончательно обработана.

Предварительно все вершины покрашены в белый.

```

DepthSearch(C, B) {
  если  $C=B$  (то, куда надо попасть),
    то обратная цепочка по рекурсии есть путь
  если  $C$  не белый return // если серый, то это цикл ориентированного графа
   $C$  красим серым
  для каждого соседа  $x$  вершины  $C$  {
    DepthSearch( $x$ , B)
  }
   $C$  красим черным
  обрабатываем  $C$ , если надо.
}

```

Очень естественно обнаруживаются все пути.

Сложность: количество вершин + количество ребер.

**Алгоритм Дейкстры** — поиск пути из  $A$  в  $B$  с наименьшим весом.

Модифицированный поиск в ширину.

Метка вершины — ее расстояние от  $A$  в смысле суммы весов ребер вдоль найденного пути.

$w(x, y)$  — вес ребра из  $x$  в  $y$ .

**шаг 0.** пометить все вершины меткой “бесконечность”.  $M = \emptyset$ .

**шаг 1.** Пометить  $A$  меткой 0 (т.е.  $A(0)$ ). Добавить  $A$  в  $M$ .

**шаг 2.** Есть непустое граничное множество  $M$ .

```
// для всех точек M и точек ‘внутри’ M известны длины
// кратчайших путей от A через просмотренные точки
Выбираем из M вершину x с наименьшим весом.
для каждого соседа y вершины x {
    если  $\text{вес}(y) > \text{вес}(x) + w(x,y)$  то {
         $\text{вес}(y) = \text{вес}(x) + w(x,y)$ 
        добавить y в M
    }
}
удалить x из M
повторять шаг 2, пока не будут рассмотрены все ребра, входящие в B.
```