Тема 6. Некоторые алгоритмы

Лекция 18. 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 алгортимами 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 продвидается 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. пометить все вершины меткой -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 имееют метку k+1
```

шаг 3. Если еще есть не посещенные вершины, то перейти в **шаг 2.**

отдельно можно рассмотреть случай несвязных компонент.

Метка вершины 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
             регистрируем переход  x <-> y
          } иначе {
             есть цикл неориентированного графа
          }
     }
     удалить x из M
 }
```

аналогично можно проследить и все пути из A в B
Сложность: количество вершин + количество ребер
Поиск в глубину. Знакомый нам рекурсивный обход внизу-вверх.
Раскраска 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(A) {
    если A --- это то, куда надо попасть,
      то обратная цепочка по рекурсии есть путь
    если A не белый return // если серый, то это цикл ориентированного графа
    A красим серым
    для каждого соседа x вершины A {
        DepthSearch(x)
    }
    A красим черным
    Обрабатываем A, если надо.
}
```

Очень естественно обнаруживаются все пути.

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 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 ‘‘внутри’’  известны длины
// кратчайших путей от A через просмотренные точки
Выбираем из M вершину x с наименьшим весом w.
для каждого соседа y вершины x {
```

```
        если вес(y) > вес(x) + w(x,y) то {
            вес(y) = 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.
```