Введение в графы

Лекция 9

Что такое граф?

Граф – совокупность двух множеств E и V, где E – множество ребер, а V – множество вершин.

- Путь
- Степень вершины
- Цикл

Виды графов

- (Не)ориентированные
- (Не)связные
- (Не)ациклические
- (Не)взвешенные
- (Не)полные
- Двудольные
- Планарные
- Деревья

Способы хранения графов

- Матрица смежности
- Список ребер
- Списки смежности

Способ	Память	Проход по соседям	Проверка наличия ребра
MC	O(N^2)	O(N)	O(1)
CP	O(M)	O(M)	O(M)
CC	O(M)	O(deg V)	O(deg V)

Обход в глубину (DFS)

Обход в глубину

- Рекурсивно запускаем обход из всех непосещенных соседей вершины
- O(|V| + |E|)

```
dfs(v, used)
    used[v] = true
    for (v, u) in E
        if not used[u]
            dfs(u, used)

for v in V
    if not used[v]
        dfs(v, used)
```

Другие применения DFS

- Поиск компонент связности
- Поиск цикла (проверка ацикличности)
- Топологическая сортировка
- И другие

Поиск компонент связности

- Пройдемся по всем вершинам
- Непосещенные вершины будем раскрашивать в отдельный цвет
- Количество уникальных цветов будет равно количеству компонент связности

```
dfs(v, color, cur)
    color[v] = cur
    for (v, u) in E:
        if color[u] == -1:
            dfs(u, color, cur)

cnt = 0
for v in V
    if color[v] == -1
        cnt++
        dfs(v, color, cnt)
```

Поиск цикла

- 0 – вершина посещена
- 1 – из вершины запущен dfs
- 2 - для вершины завершился dfs
- Если пришли из 1 в 1, то есть цикл

```
dfs(v, color)
      color[v] = 1
      for (v, u) in E
            if color[u] == 0
                  dfs(u, color)
            if color[u] == 1
                  … # обрабатываем цикл
      color[v] = 2

for v in V
      if color[v] == 0
            dfs(v, color)
```

Топологическая сортировка

- Отсортируем вершины так, чтобы не было ребер, по которым можно попасть из правой вершины в левую
- Допустим, что граф ациклический
- Будем фиксировать порядок, в котором вершины «покидают» наш DFS

```
dfs(v, used, tout)
      used[v] = true
      for (v, u) in E
            if not used[v]
                  dfs(u, used, tout)
      tout[v] = T++
T = 0
for v in V
      if not used[v]
            dfs(v, used, tout)
# сортируем вершины по убыванию tout
```

Обход в ширину (BFS)

Обход в ширину

- O(|E|)

```
bfs(V, E, s)
      used[s] = true
      queue.push(s)
      while not queue.isEmpty()
            v = queue.pop()
            for (v, u) in E
                  if not used[u]
                        used[u] = true
                        queue.push(u)
```