Семинар 12. ПОИСК В ГЛУБИНУ И В ШИРИНУ

Важными задачами теории графов являются **задачи глобального анализа** как *неориентированных*, так и *ориентированных графов*. К этим задачам относятся, например, задачи поиска *циклов* или *контуров*, вычисление *длин путей* между парами *вершин*, перечисление *путей* с теми или иными свойствами и т.п.

Необходимо уметь обходить все вершины графа таким образом, чтобы каждая вершина была отмечена ровно один раз. Обычно такое „путешествие“ по графу сопровождается нумерацией вершин графа в том порядке, в котором они отмечаются, а также определенной „маркировкой“ *ребер* (или *дуг*) графа.

Существуют две основные стратегии таких обходов: **поиск в глубину** и **поиск в ширину**.

Поиск в глубину в неориентированном графе

Будем считать, что граф задан списками смежности, собранными в *массив лидеров*.

При поиске вершины графа нумеруются в порядке их посещения. Номер вершины v графа, присваиваемый ей при поиске в глубину, обозначим $D[v]$ и будем называть **D-номером**.

В процессе обхода будем находить **фундаментальные циклы** графа.

Пусть в неориентированном графе $G = (V, E)$ произвольно фиксирован *максимальный остовный лес*. Для связного графа это будет максимальное остовное дерево. Множество его ребер обозначим T. Все ребра из T назовем **древесными**, а ребра исходного графа G, не принадлежащие T, — **обратными**.

Любой цикл графа G, содержащий только одно обратное ребро, назовем фундаментальным.

Максимальный остовный лес, находимый с помощью алгоритма поиска в глубину, называют **остовным лесом поиска в глубину** или **глубинным остовным лесом**.

Заметим, что классификация ребер зависит от хода работы алгоритма, который определяется стартовой вершиной и расположением вершин в списках смежности.

Для организации работы алгоритма поиска в глубину используется способ хранения данных, называемый **стеком**. Элементы в стеке упорядочиваются в порядке поступления. В стек можно добавлять новые элементы и из него можно извлекать элементы. При этом доступен только последний добавленный элемент — **вершина стека**.

В алгоритме поиска в глубину используется стек вершин. Мы будем считать, что из стека можно считывать информацию без изменения его содержимого, но в том порядке, в каком ее удаляли бы из стека.

В ориентированном графе вершинам также присваиваются D-номера. Но классификация дуг при поиске в глубину в ориентированном графе сложнее по сравнению с аналогичной классификацией ребер при поиске в глубину в неориентированном графе. Различают четыре класса дуг:

1) **древесные дуги** — каждая такая *дуга ведет* от *отца* к *сыну* в глубинном остовном лесу;

2) **прямые дуги** — каждая такая дуга ведет от *подлинного предка* к *подлинному потомку* (но не от отца к сыну) в глубинном остовном лесу;

3) **обратные дуги** — от *потомков* к *предкам* (включая все петли);

4) **поперечные дуги** — все дуги, не являющиеся ни древесными, ни прямыми, ни обратными.

В результате работы алгоритма будут получены множества $Tree$ — древесных дуг, $Back$ — обратных дуг, $Forward$ — прямых дуг, C — поперечных дуг и массив D, содержащий D-номера вершин.

В процессе работы алгоритма по сравнению с алгоритмом поиска в глубину в неориентированном графе имеется ряд особенностей. Так, если очередная вершина w, извлеченная из списка смежности текущей вершины v, новая, то дуга $<v, w>$ является древесной.

Если вершина w не новая $(w \notin V_0)$, то дуга $<v, w>$ будет либо прямой, либо обратной, либо поперечной.

Если D-номер вершины v строго меньше D-номера вершины w $(D[v] < D[w])$, то дуга $<v, w>$ является прямой.

Если D-номер вершины v не меньше D-номера вершины w $(D[v] \geq D[w])$, необходимо проверить, есть ли в стеке $STACK$ вершина w. Если вершина w находится в стеке, то дуга $<v, w>$ является обратной. Если вершины w в стеке нет, то дуга является поперечной.

Если стек пуст, но не все вершины ориентированного графа обработаны, поиск продолжают из любой необработанной вершины.

В случае ориентированного графа поиск контуров на базе поиска в глубину существенно сложнее.

Ориентированный граф является бесконтурным тогда и только тогда, когда при поиске в глубину от некоторой начальной вершины множество обратных дуг оказывается пустым.

Заметим также, что для ориентированного графа нет такой простой связи между числом опустошений стека и числом компонент, как для неориентированного графа.

Алгоритм поиска в ширину в ориентированном графе

Вход. Граф $G = (V, E)$, заданный списками смежности; v_0 — начальная вершина (не обязательно первый элемент массива лидеров).

Выход. Массив M меток вершин, где каждая метка равна длине пути от v_0 до v.

0. Очередь Q положить пустой $(Q := \varnothing)$. Все вершины пометить как недостижимые из вершины v_0, присваивая элементам массива M значение $+\infty$ $(M[v_i] := +\infty,\ i = \overline{1,\ N})$.

Стартовую вершину v_0 пометить номером 0, т.е. длину пути от стартовой вершины v_0 до самой себя положить равной 0 ($M[v_0] := 0$). Поместить вершину v_0 в очередь Q. Перейти на шаг **1**.

1. Если очередь Q не пуста $(Q \neq \varnothing)$, то из „головы“ очереди извлечь (с удалением из очереди) вершину u и перейти на шаг **2**. Если очередь пуста, перейти на шаг **3**.

2. Если список смежности $L(u)$ вершины u пуст, вернуться на шаг **1**.

Если список смежности $L(u)$ вершины u не пуст, для каждой вершины w из списка смежности, где $M[w] = +\infty$, т.е. вершины, которую еще не посещали, положить длину пути из стартовой вершины v_0 до вершины w равной длине пути от v_0 до вершины u плюс одна дуга $(M[w] := M[u] + 1)$, т.е. отметить вершину w

и поместить ее в очередь Q. После просмотра всех вершин списка смежности $L(u)$ вернуться на шаг **1**.

3. Распечатать массив M. Закончить работу.

Алгоритм поиска в ширину может быть дополнен процедурой „обратного хода“, определяющей номера вершин, лежащих на кратчайшем пути из вершины v_0 в данную вершину u. Для этого необходимо завести массив PR размера $|V|$, каждый элемент $PR[w]$ которого содержит номер той вершины, из которой был осуществлен переход в вершину w при ее пометке.

Если вершина w находится в списке смежности $L(u)$ вершины u, заполнение элемента массива $PR[w]$ происходит при изменении метки вершины w $M[w]$ с $+\infty$ на единицу. При этом в элементе $PR[w]$ сохраняется номер вершины u $(PR[w] := u)$. Для начальной вершины $PR[v_0]$ можно положить равным 0, в предположении, что начальная вершина v_0 имеет номер 0 и остальные вершины пронумерованы от 1 до N.