**Лекция 4. Каркасы, фундаментальное множество циклов**

**Каркас (стягивающее дерево, остов)**

Каждый связный неориентированный граф имеет каркас. Более того, у одного графа может быть несколько разных каркасов. Дадим теперь определение каркаса.

Для произвольного связного графа *G=<V, E>* любое дерево *D=<V, T>*, где *T* *Е* будет его **каркасом** (рис. 4.1).

(Напомним, что дерево – произвольный неориентированный связный граф без циклов). Другими словами, каркас *D* соединяет все вершины графа *G* так, чтобы не было циклов.



Рис. 4.1. Каркас графа

**Ребра** **каркаса** называются **ветвями**; ребра, не вошедшие в каркас, **хордами**.

***Вопрос.*** Перечислите хорды графа *G* относительно каркаса *D* на рис. 4.1.

Каркасы можно строить как поиском в глубину, так и поиском в ширину (рис. 4.2). В обоих случаях достижение новой вершины *u* из старой *v* означает включение в каркас ветви (*v-u*).

**АЛГОРИТМ 4.1** *{Построение каркаса методом поиска в глубину}*

*Данные:* Неориентированный связный граф *G=<V, E>,* заданный списками инцидентности ЗАПИСЬ[*v*], *v*  *V*.

*Результаты:* Каркас *D*=*<V, T*> графа *G*.

*Глобальные переменные:* НОВЫЙ, Т.

1 procedure DFS\_tree(v);

2 begin

3 НОВЫЙ[v]:= false;

4 for u  ЗАПИСЬ[v] do

5 if НОВЫЙ[u] then {нашли новую ветвь (v–u)}

6 begin

7 T:= T(v-u); {и присоединили ее к каркасу}

9 DFS\_tree(u)

10 end

11 end;

1 begin {основная программа}

2 for u  V do НОВЫЙ[u]:= true; {инициализация}

3 T:= 0; {T-множество найденных ветвей}

4 DFS\_tree(k) {корень k – произвольная вершина}

5 end.

**Вычислительная сложность и корректность алгоритма**

Алгоритм строит связный граф, так как каждое новое ребро (*v—u*) продолжает уже существующий путь от *k* к *v*.

Построенный граф не содержит циклов. Каждая новая ветвь (*v—u*) соединяет уже рассмотренную *v* с новой *u*. Чтобы замкнуть цикл, требуется ребро, соединяющее две уже рассмотренные вершины.

Построенный граф содержит все вершины графа *G* – это свойство поиска в глубину. По этой же причине вычислительная сложность алгоритма О(*n + m*).

**Свойства каркаса**

Любой каркас обладает важным свойством – от корня *k* до произвольной вершины *v* существует единственный путь, состоящий из ветвей каркаса. Если бы их было два, получился бы цикл, если бы ни одного, каркас не был бы связным.

Кроме того, **если каркас *D* построен поиском в глубину**, то для двух вершин *u* и *v*, соединенных ребром*e*  *E*, всегда можно сказать: или *v* – потомок *u*, или *u* – потомок *v* (относительно каркаса *D*).

Первое означает, что *u* лежит на пути из корня *k* в вершину *v*, второе – *v* на пути из *k* в *u*. Это легко доказать.

Пусть одна из вершин, например, *v*, просмотрена раньше *u*. Построен путь от корня *k* до *v*. Процесс поиска в глубину начинается с вершины *v*. Так как *u* и *v* соединены ребром, то рано или поздно будет рассмотрена вершина *u* и построен путь от *v* до *u*. Получился путь *k–v–u*.

Если *v* и *u* соединены **ветвью** каркаса, то одна из них – **сын**, другая – **отец**.

**АЛГОРИТМ 4.2** *{Построение каркаса методом поиска в ширину}*

*Данные*: Неориентированный связный граф *G=<V, E>,* представленный списками ЗАПИСЬ[*v*], *v*  *V*.

*Результаты*: Каркас *D*=<*V, T*> графа *G*.

1 begin

2 for u  V do НОВЫЙ[u]:= true;{инициализация}

3 T:= 0 {T – множество найденных ветвей}

4 ОЧЕРЕДЬ:= NIL;

5 ОЧЕРЕДЬ <= k; {поместили в очередь корень k}

6 НОВЫЙ[k]:= false; {пометили k как просмотренный}

7 while ОЧЕРЕДЬ <> NIL do

8 begin tail <= ОЧЕРЕДЬ;

9 for u  ЗАПИСЬ[tail] do

10 if НОВЫЙ[u] then {нашли новую ветвь v–u}

11 begin

12 ОЧЕРЕДЬ <= u; НОВЫЙ[u]:= false;

13 T:=T(v-u) {и присоединили ее к каркасу}

14 end

15 end

16 end.

Легко доказать, что этот алгоритм строит каркас произвольного графа за О(*n + m*) шагов.



Рис. 4.2. Два каркаса графа G

Если каркас *D* графа *G* построен поиском в ширину, то путь от корня *k* до произвольной вершины *v* будет не только единственным, но и кратчайшим. Это следует из свойств поиска в ширину.

***Ответ***. {(2–3), (3–4), (5–6), (6–1), (2–5), (1–4)}.

**5. ФУНДАМЕНТАЛЬНОЕ МНОЖЕСТВО ЦИКЛОВ ГРАФА**

Предположим, нам нужно с помощью закона Кирхгофа составить систему линейных уравнений для электрической цепи. Цепь имеет сложную структуру и содержит большое количество циклов. Запишем закон Кирхгофа для каждого цикла и начнем решать полученную систему уравнений. Сразу обнаружится, что часть уравнений являются линейно зависимыми, т.е. получаются из других. Значит, какие-то циклы были не нужны?!

В этой главе будет доказано, что любой цикл графа можно однозначно разложить по фундаментальным циклам этого графа так же, как любой вектор трехмерного пространства однозначно представляется линейной комбинацией базисных векторов *i*, *j*, *k*. Так же будет дан метод построения этого множества.

В дальнейшем под графом будем понимать связный неориентированный граф, под циклом – элементарный цикл.

Вначале для произвольных множеств *А* и *В* введем операцию взятия симметрической разности  (рис. 5.1).

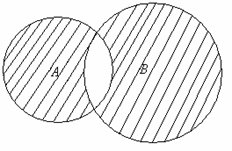


Рис. 5.1. Симметрическая разность двух множеств

**Утверждение 1**

*Симметрическая разность множеств А1, А2, …, Аk содержит в точности те элементы, которые принадлежат нечетному числу множеств.*

Доказать это можно индукцией по числу множеств.

Также легко доказать, что **симметрическая разность любого числа циклов графа *G* является циклом графа *G*.**

Вернемся к фундаментальным циклам. Это множество определяется через каркас графа.

Пусть уже построен каркас *D=<V, T>* графа *G=<V, E*>. Если к каркасу добавить произвольную хорду *e*  (*E—T*), то получившийся граф (*D**e*) будет содержать ровно один **цикл**. Назовем этот цикл *Се* (рис. 5.2). Добавляя к каркасу поочередно все хорды, получим **фундаментальное множество циклов Ф относительно каркаса *D:*** Ф = {*Cе* : *e* (*E*-*T*)}.

***Вопрос* 1.** Укажите фундаментальное множество циклов графа *G* относительно каркаса *D* (рис. 5.2, хорды обозначены пунктиром).

***Вопрос* 2.** В графе *G* (рис. 5.2) найти *C*1 *C*2, если *C*1=(1–2–3), *C*2=(2–3–4).

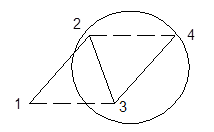


Рис. 5.2. Ф-цикл

Сформулируем без доказательства следующее утверждение.

**Утверждение 2**

*Произвольный цикл C графа G можно однозначно представить как симметрическую разность некоторого числа фундаментальных циклов.*

Другими словами, любой цикл раскладывается по базису фундаментальных циклов.

Как уже говорилось, знание фундаментальных циклов имеет существенное значение при анализе электрических цепей. Для каждого фундаментального цикла данной цепи можно записать закон Кирхгофа: сумма падений напряжений вдоль цикла равна 0. Все эти уравнения независимы, уравнения для остальных циклов будут следовать из них.

Опишем простой алгоритм построения фундаментальных циклов. Он основан на поиске в глубину из произвольного корня *k* и является рекурсивным. Каждая встреченная новая вершина *v* помещается в стек и получает номер DN[*v*] (порядок просмотра) и удаляется из него после использования. Поэтому стек всегда содержит последовательность вершин от рассматриваемой в данный момент вершины *v* до корня *k*.

Анализируемое алгоритмом ребро (*v—u*) будет замыкать цикл, если (*v—u*) – хорда. Ребро (*v—u*) будет хордой, если *u*– найденная соседка *v* – уже встречалась раньше при поиске в глубину. В этом случае она находится в стеке и ее номер DN[*u*] меньше соответствующего номера DN[*v*]. Если из *u* мы попали в вершину *v*, т. е. *u* является отцом *v*, то DN[*u*] < DN[*v*], но (*v—u*) не хорда, а ветвь каркаса. Эту ситуацию легко распознать, так как в этом случае вершины *u* и *v* стоят в стеке рядом.

**Критерий хорды (*v*—*u*)**:

* обе вершины *u* и *v* просмотрены;
* DN[*u*] < DN[*v*];
* в стеке находятся не рядом.

***Вопрос* 3.** Где гарантия того, что найденная вершина *u* всё ещё находится в стеке, а не вытолкнута при возврате?

Если (*v*—*u*) – хорда, то цикл состоит из верхней группы элементов стека, начиная с *v* и кончая *u* (рис. 5.3).

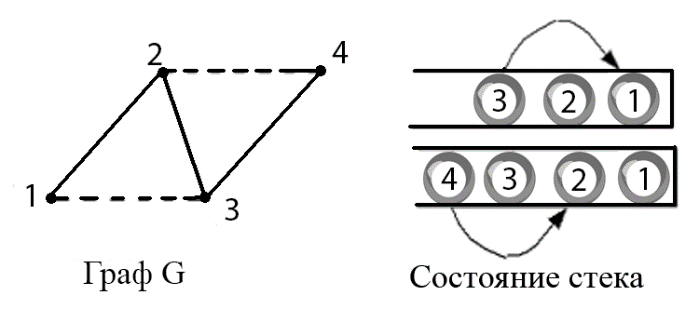


Рис. 5.3. Стек, содержащий фундаментальный цикл

Мы заранее знаем, что наибольшее число вершин, одновременно находящихся в стеке, не превышает *n* – длины пути от корня до самой удаленной вершины, поэтому стек можно имитировать массивом переменной длины, не превышающей *n*.

Текущее значение длины будет храниться в переменной *d*, вталкивание в стек вершины *v* будет изображаться: STACK[d] := v, выталкивание верхушки стека: d := d – 1.

**АЛГОРИТМ 5.1** *{Нахождение множества* Ф-*циклов графа}*

Данные: Неориентированный связный граф *G=<V, E>,* представленный списками ЗАПИСЬ[*v*], *v*  *V*.

Результаты: Множество фундаментальных циклов Ф графа *G*.

Глобальные переменные: d, num, STACK, DN*.*

1 procedure CYCLE(v);

{находит фундаментальное множество циклов для компоненты связности

графа, содержащей вершину v}

2 begin

3 d:= d+1; STACK[d]:= v; {помещаем v в стек}

4 num:= num+1; DN[v]:= num; {присваиваем v номер}

5 for u  ЗАПИСЬ[v] do{перебор соседок v}

6 if DN[u]=0 then

7 CYCLE(u) {для новой вершины u}

8 else if (u <> STACK[d-1]) AND (DN[u] < DN[v]) then

9 печать STACK[d], STACK[d-1],.., STACK[c]=u;

{ребро (v—u) – хорда}

10 d:= d-1; {использованная v удаляется из стека}

11 end;

1 begin {основная программа}

2 for v Є V do DN[v] := 0;

3 num:= 0; {инициализация перед началом нумерации}

4 d:= 0; {d – количество элементов в стеке}

5 for v  V do

6 if DN[v]=0 then CYCLE(v)

7 end.

**Вычислительная сложность алгоритма**

Если отбросить число шагов, требующих выписывания всех фундаментальных циклов (строки 8–9), сложность имеет порядок О(*n+m*), как у всех алгоритмов, основанных на поиске в глубину.

Число шагов, потраченное на печать всех фундаментальных циклов, пропорционально суммарной длине всех циклов. Подсчитаем количество фундаментальных циклов.

**Количество Ф-циклов** = число всех хорд = число всех рёбер – число всех ветвей каркаса = 

Длина любого цикла не превосходит *n*. Таким образом, суммарная длина всех циклов не превосходит  Если отбросить случай, когда число ребер *m* равна 0, вычислительная сложность алгоритма равна О(*mn*).

**Утверждение 3**

*Циклы неориентированного связного графа образуют линейное пространство, количество базисных циклов равно*

***Ответ* 1**. Ф = {*C*1 = (1–2–3), *C*2 = (2–3–4)}.

***Ответ* 2**. *С*1 *С*2 = (1–2–3–4).

***Ответ* 3.** Стек при поиске в глубину на неориентированном графе содержит последовательность вершин от рассматриваемой в данный момент вершины *v* до корня *k*. DN[*u*] < DN[*v*], поэтому *u* была просмотрена раньше *v* и находится в стеке между *v* и *k*.