Posted: 5 Jul 2008 22:26
Edited: 6 Dec 2012 1:41

Find points of articulation

Contents [hide]

- Find points of articulation
  - Algorithm
  - Implementation
  - Problem in online judges

Let a connected undirected graph. **articulation point** (or point of articulation, English. "cut vertex" or "articulation point") is called a vertex, which makes the removal of the graph disconnected.

We describe an algorithm based on the search for in-depth, working at $O(n + m)$, where n- the number of vertices m- edges.

Algorithm

Start the tour in depth from an arbitrary vertex of the graph; denote it through root. We note the following **fact** (which is not difficult to prove):

- Suppose we are in a circuit in depth, looking now all edges from the top $v \neq \text{root}$. Then, if the current edge $(v, to)$is such that from the top $to$and from any of its descendant in the tree traversal in depth do not have an edge in the ancestor of a vertex v, then the vertex $v$is an articulation point. Otherwise, i.e. if the bypass in depth through all the edges of the vertices v, and have not found satisfying the above conditions edges, the vertex $v$is not an articulation point. (In fact, we have this condition, check if there are other ways of va to)
- We now consider the remaining case: $v = \text{root}$. Then this vertex is an articulation point if and only if this vertex has more than one son in the tree traversal in depth. (In fact, this means that, going out rooton an arbitrary edge, we were not able to get around the whole graph, which immediately implies that root- the point of articulation).

(Wed wording of this criterion with the wording of the criteria for the search algorithm bridges .)

Now it is necessary to learn how to verify this fact for each vertex effectively. To do this, use the "time of entry into the top," is calculated by the search algorithm in depth .

So, let $tin[v]$- this time call DFS at the top v. Now we introduce an array $fup[v]$, which will allow us to answer the above questions. Time $fup[v]$is the minimum time of entry into the very top $tin[v]$, the time of entering into each vertex p, which is the end of a reverse edges (v, p), as well as of all the values $fup[to]$for each vertex to, which is a direct son $v$of the search tree:

$$fup[v] = \min \begin{cases} tin[v], \\ tin[p], & \text{for all (v,p) — back edge} \\ fup[to], & \text{for all (v,to) — tree edge} \end{cases}$$

(Here "back edge" - the opposite edge, "tree edge" - edge of the tree)

Then, from the top $v$or her offspring have the opposite edge in its parent if and only if there is such a son tothat $fup[to] < tin[v]$.

Thus, if the current edge (v, to)(belonging to the tree search) is performed $fup[to] \geq tin[v]$, then the vertex $v$is a point of articulation. For the initial vertex of $v = \text{root}$the other criteria: for this it is necessary to count the number of vertices of immediate sons in the tree traversal in depth.

Implementation

If we talk about the actual implementation, here we need to be able to distinguish three cases: when we are on the edge of the tree depth-first search, when we go to the opposite edge, and when we try to go along the edge of the tree in the opposite direction. It is, accordingly, cases $used[to] = false$, $used[to] = true \ \&\& \ to \neq parent$and the $to = parent$. Thus, we need to pass in the search function in the top of the depth of the parent of the current node.

```
vector<int> g[MAXN];
bool used[MAXN];
int timer, tin[MAXN], fup[MAXN];

void dfs (int v, int p = -1) {
        used[v] = true;
        tin[v] = fup[v] = timer++;
        int children = 0;
        for (size_t i=0; i<g[v].size(); ++i) {
                int to = g[v][i];
                if (to == p)  continue;
                if (used[to])
                        fup[v] = min (fup[v], tin[to]);
                else {
                        dfs (to, v);
                        fup[v] = min (fup[v], fup[to]);
                        if (fup[to] >= tin[v] && p != -1)
                                IS_CUTPOINT(v);
                        ++children;
                }
        }
        if (p == -1 && children > 1)
                IS_CUTPOINT(v);
}

int main() {
        int n;
        ... чтение n и g ...
```

```
        timer = 0;
        for (int i=0; i<n; ++i)
                used[i] = false;
        dfs (0);
}
```

Here, the constant $MAXN$ must be set to a value equal to the maximum possible number of vertices in the input graph.

Function $IS\_CUTPOINT(v)$ in the code - it's a function that will respond to the fact that the vertex v is an articulation point, for example, to deduce that the top of the screen (it should be borne in mind that for the same vertex, this function can be called several times).

Problem in online judges

A list of tasks that require the search for points of articulation:

- UVA # 10199 **"Tourist Guide"** [Difficulty: Easy]
- UVA # 315 **"Network"** [Difficulty: Easy]

---

e_maxx Модератор → filo • 2 года назад

Очевидно, не обязательно - т.к. если граф ацикличен и связен, то это дерево, т.е. абсолютно неинтересный случай (любой не-лист дерева - это его точка сочленения).

3 • Ответить • Поделиться ›

Бекжан • 2 года назад

А что если все вершины образуют замкнутую цепочку, т.е 0 -> 1, 1 -> 2, 2 -> 3, 3 -> 0? Как я понял, этот алгоритм пометит вершину 0 как точку сочленения, тогда как точек сочленения в таком графе нет.

• Ответить • Поделиться ›

e_maxx Модератор → Бекжан • 2 года назад

Почему же, не пометит. Для вершины, из которой запускается обход в глубину, критерий отдельный: она точка сочленения, если количество рекурсивных вызовов из неё больше единицы.

2 • Ответить • Поделиться ›

Роман • 2 года назад

vector < int > g[MAXN];

По идее, нужно vector < vector < int > > G[MAXN]?

• Ответить • Поделиться ›

e_maxx Модератор → Роман • 2 года назад

Почему же, в вашем коде фактически получается трёхмерный массив (т.е. MAXN векторов, состоящих из векторов). В статье используется массив из MAXN векторов: т.е. граф задан в виде списков смежности (т.е. i-ый вектор содержит номера вершин, в которые ведут рёбра из i-ой вершины).

2 • Ответить • Поделиться ›

Роман → e_maxx • 2 года назад

А, понятно, спасибо. Не обратил внимание на комбинацию вектора и массива.

• Ответить • Поделиться ›

Axiom • 2 года назад

В описании алгоритма:
>> Если ... нет обратного ребра в v или какого-либо предка вершины v, то вершина v является точкой сочленения
Но обратное ребро в v вполне может быть. Да и в алгоритме ведь fup(u)>=tin(v)

• Ответить • Поделиться ›

e_maxx Модератор → Axiom • 2 года назад

Вы правы, эта ошибка в описании, видимо, закралась из соседней статьи про поиск мостов.

• Ответить • Поделиться ›

Guest • 2 года назад

Почему p нигде не меняется? Какой его физ. смысл?

• Ответить • Поделиться ›

KondorB ➚ Guest • 2 года назад

p передается в DFS как предок обрабатываемой вершины в дереве обхода, т. е. та вершина из которой мы попадаем в v

1 • Ответить • Поделиться ›

KondorB • 2 года назад

Здравствуйте, дайте пожалуйста тесты на задачу поиска точек сочленения (в любом удобном вам формате). Решаю такую задачу на codeforces, мой код(реализовал этот алгоитм на delphi) валиться на третьем тесте. Бьюсь уже неделю, не могу понять где косяк.

• Ответить • Поделиться ›

aleksey • 2 года назад

Зачем выполнять проверку if (to == p)?

• Ответить • Поделиться ›

e_maxx Модератор ➚ aleksey • 2 года назад

Потому что мы не хотим рассматривать ребро, по которому мы прошли в текущую вершину: это ребро прямое, и оно уже было обработано, когда мы стояли на другом конце этого ребра (в вершине to).

• Ответить • Поделиться ›

ПРЕСТИЖ • 2 года назад

опечатка :
и из любого её потомка в дереве обхода в глубину нет обратного ребра в или

• Ответить • Поделиться ›

e_maxx Модератор ➚ ПРЕСТИЖ • 2 года назад

Спасибо, исправлено.

• Ответить • Поделиться ›

Ivan • 2 года назад

А почему 'if(fup[to] >= tin[v] && p!=-1)' а не 'if(fup[to] > tin[v] && p!=-1)'

• Ответить • Поделиться ›