Search cutpoints

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

We describe an algorithm based on DFS working behind $O(n+m)$where n- number of vertices m- edges.

Algorithm

Start the tour in depth from an arbitrary vertex of the graph, denoted by root. We note the following **fact** (which is not hard to prove):

- Suppose we are bypassed in depth, looking now all edges from the top $v \neq \text{root}$. Then, if the current edge $(v, to)$is that from the top $to$and from any of its descendant tree traversal depth of no return edges in any ancestor vertex v, then the vertex $v$is an articulation point. Otherwise, i.e. if bypass in depth through all the tops of the ribs v, and found satisfying the above conditions ribs, the top v is not an articulation point. (In fact, this condition we 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 node has more than one son in the tree traversal in depth. (In fact, this means that passing out rooton an arbitrary edge, we could not get around the whole graph, which implies that root- the point of articulation).

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

It now remains to learn how to check this fact for each vertex effectively. For this we use the "time of entry into the summit," 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 entering into the very top $tin[v]$, sunset times in each vertex p, which is the end of a reverse edge (v, p), as well as of all the values $fup[to]$for each vertex to, which is a direct son $v$in 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 a son tothat $fup[to] < tin[v]$.

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

Implementation

If we talk about the actual implementation, here we must be able to distinguish three cases: when we go on the edge of the tree depth-first search, when we go to the opposite edge, and when trying to go along the edge of the tree in the opposite direction. It is, accordingly, the 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 current node ancestor.

```
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 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 react to the fact that the vertex $v$is an articulation point, for example, to display this on the top of the screen (it should be borne in mind that for the same vertex, this function can be called multiple times.)

Problem in online judges

Task List, which requires seek points of articulation:

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