MAXimal

added: 10 Jun 2008 19:25
Edited: 24 Aug 2011 12:31

Search algorithm for connected components in a graph

Contents [hide]

Given an undirected graph $G$with $n$vertices and $m$edges. You want to find in it all the connected components, ie, beat the vertices of the graph into several groups so that within the same group can be reached from one node to any other, and between different groups - the path does not exist.

An algorithm for solving

Alternatively you can use as a bypass in depth , and a preorder traversal .

In fact, we will be producing **a series of rounds** , first run the bypass from the first vertex and all the vertices that in doing so he walked - form the first connected component. Then we find the first of the remaining vertices that have not yet been visited, and run circumvention of it, thus finding a second connected component. And so on, until all the vertices will not be marked.

The final **asymptotic behavior** will be $O(n+m)$: in fact, such an algorithm will not start from the same vertex twice, which means that each edge will be seen exactly twice (at one end and the other end).

Implementation

To implement a little more convenient to bypass the deep :

```
int n;
vector<int> g[MAXN];
bool used[MAXN];
vector<int> comp;

void dfs (int v) {
        used[v] = true;
        comp.push_back (v);
        for (size_t i=0; i<g[v].size(); ++i) {
                int to = g[v][i];
                if (! used[to])
                        dfs (to);
        }
}

void find_comps() {
```

```
        for (int i=0; i<n; ++i)
                used[i] = false;
        for (int i=0; i<n; ++i)
                if (! used[i]) {
                        comp.clear();
                        dfs (i);

                        cout << "Component:";
                        for (size_t j=0; j<comp.size(); ++j)
                                cout << ' ' << comp[j];
                        cout << endl;
                }
}
```

The main function to call - $\text{find_comps}()$she finds and displays the components of the graph.

We believe that the graph is given adjacency lists, ie $g[i]$a list of vertices, in which there are edges from the top i. Constant MAXNshould be set equal to the maximum possible number of vertices in the graph.

Vector compcontains a list of vertices in the current connected component.