added: 10 Jun 2008 19:24
Edited: 23 Aug 2011 12:42

Topological Sort

Contents [hide]

Given a directed graph with n vertices and $m$edges. Required **to renumber** the vertices so that each rëbro led from the top with a smaller number in the top with a lot.

In other words, find a permutation of the vertices (**topological order**) corresponding to the order given by all edges of the graph.

Topological sort can be **not only** (for example, if the graph - empty, or if there are three such vertices a, b, $c$that of $a$there way to $b$and in c, but none of the $b$in c or out $c$of $b$reach impossible).

Topological sort may **not exist** at all - if the graph contains cycles (as there is a contradiction: there is a way and from one vertex to another, and vice versa).

A common problem on a topological sort - next. There are $n$variables that are unknown to us. We only know about some of the pairs of variables that one variable is less than another. Need to check whether these are not contradictory inequality, and if not, to give the variables in ascending order (if there are several solutions - to give any). Easy to see that this is exactly is the problem of finding a topological sort of the graph $n$vertices.

Algorithm

Solutions for use in bypass depth .

Suppose that the graph is acyclic, ie, solution exists. What makes a detour into the depths? When you run out of some vertex $v$he tries to run along all the edges emanating from v. Along the edges, the ends of which have already been visited before, it does not pass, and along all the others - goes and calls himself from them all.

Thus, by the time of the call, $\mathrm{dfs}(v)$all the vertices reachable from $v$both directly (one edge) and indirectly (by the way) - all such vertices already visited bypass. Therefore, if we are at the moment out of the $\mathrm{dfs}(v)$top of our to add to the top of a list, then in the end of this list will **be a topological sorting** .

These explanations can be presented in a slightly different way, using the concept of **"time release"** crawl into the depths. Retention time for each vertex v - a point in time at which the call ended up working $\mathrm{dfs}(v)$in the crawl depth of it (retention times can be numbered from 1before n). It is easy to understand that in

going in depth while leaving a vertex v is always greater than the output of all the vertices reachable from it (as they have been visited or to call $\mathrm{dfs}(v)$ or during it). Thus, the desired topological sorting - sorting in descending order since the release.

Implementation

We present the implementation, it is assumed that the graph is acyclic, ie, desired topological sort exists. If necessary, check the graph into acyclic easily inserted into the bypass in depth, as described in the article of circumvention in depth .

```
int n; // число вершин
vector<int> g[MAXN]; // граф
bool used[MAXN];
vector<int> ans;

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

void topological_sort() {
	for (int i=0; i<n; ++i)
		used[i] = false;
	ans.clear();
	for (int i=0; i<n; ++i)
		if (!used[i])
			dfs (i);
	reverse (ans.begin(), ans.end());
}
```

Here, the constant $MAXN$ should be set equal to the maximum possible number of vertices in the graph.

The main function of the solution - it topological_sort, it initializes tagging bypass in depth, starts it, and end up with a response vector ans.

Problem in online judges

A list of tasks that need to search for topological sort:

- UVA # 10305 **"Ordering Tasks"** [Difficulty: Easy]

- UVA # 124 **"Following Orders"** [Difficulty: Easy]
- UVA # 200 **"Rare Order"** [Difficulty: Easy]

---

int to = g[v][i];

Ошибка!

• Ответить • Поделиться ›

e_maxx Модератор ➚ Volodymyr Kulyk • 2 года назад

А в чём тут ошибка?
g[v] - это список смежности вершины v, т.е. список вершин, связанных с v ребром. Мы идём по всему этому списку, и каждая очередная вершина g[v][i] - это то, куда ведёт очередное ребро.

• Ответить • Поделиться ›

Volodymyr Kulyk ➚ e_maxx • 2 года назад

g - вектор типа int.
Думаю, правильно будет vector< vector<int> >g;

• Ответить • Поделиться ›

e-maxx ➚ Volodymyr Kulyk • 2 года назад

e-maxx

• Ответить • Поделиться ›

e_maxx Модератор ➚ Volodymyr Kulyk • 2 года назад

Ну так у нас же массив векторов - как раз для этого. Вместо массива векторов можно было, да, сделать вектор векторов, но это уже на вкус.

• Ответить • Поделиться ›

Volodymyr Kulyk ➚ e_maxx • 2 года назад