Maximum flow - Push-relabel method improved

Table of Contents

We will modify the push-relabel method to achieve a better runtime.

Description

The modification is extremely simple: In the previous article we chosen a vertex with excess without any particular rule. But it turns out, that if we always choose the vertices with the **greatest height**, and apply push and relabel operations on them, then the complexity will become better. Moreover, to select the vertices with the greatest height we actually don't need any data structures, we simply store the vertices with the greatest height in a list, and recalculate the list once all of them are processed (then vertices with already lower height will be added to the list), or whenever a new vertex with excess and a greater height appears (after relabeling a vertex).

Despite the simplicity, this modification reduces the complexity by a lot. To be precise, the complexity of the resulting algorithm is $O(VE + V^2\sqrt{E})$, which in the worst case is $O(V^3)$.

This modification was proposed by Cheriyan and Maheshwari in 1989.

Implementation

```
const int inf = 1000000000;

int n;
vector<vector<int>> capacity, flow;
vector<int> height, excess;

void push(int u, int v)
{
    int d = min(excess[u], capacity[u][v] - fl
    flow[u][v] += d;
    flow[v][u] -= d;
    excess[u] -= d;
    excess[v] += d;
}

void relabel(int u)
{
    int d = inf;
    for (int i = 0; i < n; i++) {
        if (capacity[u][i] - flow[u][i] > 0)
            d = min(d, height[i]);
    }
```

```
    if (d < inf)
        height[u] = d + 1;
}

vector<int> find_max_height_vertices(int s, in
    vector<int> max_height;
    for (int i = 0; i < n; i++) {
        if (i != s && i != t && excess[i] > 0)
            if (!max_height.empty() && height[
                max_height.clear();
            if (max_height.empty() || height[i
                max_height.push_back(i);
        }
    }
    return max_height;
}

int max_flow(int s, int t)
{
    height.assign(n, 0);
    height[s] = n;
    flow.assign(n, vector<int>(n, 0));
    excess.assign(n, 0);
    excess[s] = inf;
    for (int i = 0; i < n; i++) {
        if (i != s)
            push(s, i);
    }

    vector<int> current;
    while (!(current = find_max_height_vertice
        for (int i : current) {
            bool pushed = false;
```

```
                for (int j = 0; j < n && excess[i]
                    if (capacity[i][j] - flow[i][j
                        push(i, j);
                        pushed = true;
                    }
                }
                if (!pushed) {
                    relabel(i);
                    break;
                }
            }
        }

        int max_flow = 0;
        for (int i = 0; i < n; i++)
            max_flow += flow[0][i];
        return max_flow;
    }
```