Last update: June 8, 2022 Translated From: e-maxx.ru

Minimum spanning tree - Kruskal with Disjoint Set Union

For an explanation of the MST problem and the Kruskal algorithm, first see the main article on Kruskal's algorithm.

In this article we will consider the data structure "Disjoint Set Union" for implementing Kruskal's algorithm, which will allow the algorithm to achieve the time complexity of $O(M \log N)$.

Description

Just as in the simple version of the Kruskal algorithm, we sort all the edges of the graph in non-decreasing order of weights. Then put each vertex in its own tree (i.e. its set) via calls to the `make_set` function - it will take a total of $O(N)$. We iterate through all the edges (in sorted order) and for each edge determine whether the ends belong to different trees (with two `find_set` calls in $O(1)$ each). Finally, we need to perform the union of the two trees (sets), for which the DSU `union_sets` function will be called - also in $O(1)$. So we get the total time complexity of $O(M \log N + N + M)$ = $O(M \log N)$.

Implementation

Here is an implementation of Kruskal's algorithm with Union by Rank.

```
vector<int> parent, rank;

void make_set(int v) {
    parent[v] = v;
    rank[v] = 0;
}

int find_set(int v) {
    if (v == parent[v])
        return v;
    return parent[v] = find_set(parent[v]);
}

void union_sets(int a, int b) {
    a = find_set(a);
    b = find_set(b);
    if (a != b) {
```

```
        if (rank[a] < rank[b])
            swap(a, b);
        parent[b] = a;
        if (rank[a] == rank[b])
            rank[a]++;
    }
}

struct Edge {
    int u, v, weight;
    bool operator<(Edge const& other) {
        return weight < other.weight;
    }
};

int n;
vector<Edge> edges;

int cost = 0;
vector<Edge> result;
parent.resize(n);
rank.resize(n);
for (int i = 0; i < n; i++)
    make_set(i);

sort(edges.begin(), edges.end());

for (Edge e : edges) {
    if (find_set(e.u) != find_set(e.v)) {
        cost += e.weight;
        result.push_back(e);
        union_sets(e.u, e.v);
    }
}
```

Notice: since the MST will contain exactly $N - 1$ edges, we can stop the for loop once we found that many.

Practice Problems

See main article on Kruskal's algorithm for the list of practice problems on this topic.

Contributors:

jakobkogler (57.83%) bimalkant-lauhny (22.89%) adamant-pwn (7.23%) tcNickolas (3.61%) singamandeep (2.41%) RodionGork (2.41%) likecs (2.41%) ChamanAgrawal (1.2%)