1 Maximally-matchable edges

- *Algorithm:* Maximally-matchable edges (algo. 1)
- *Input:* A bipartite graph with a set of Edges E and a set of Vertices V, and $V = V_1 \cup V_2$, and a given maximum matching in G.
- *Complexity:* $\mathcal{O}(|V| + |E|)$
- *Data structure compatibility:* Bipartite Graph
- *Common applications:* matching skeleton, chemistry analysis, marriage problem

Maximally-matchable edges

Given a bipartite graph G with a set of vertices V and a set of edges E, find the edges in the matching that has greatest cardinality.

Description

1. Problem Clarification: Suppose we have a bipartite graph, our goal is to find the set of edges, that the set of edges covers greatest cardinality of vertices.
2. Scope Clarification: We assume we are given the maximum matching, since the matching problem is out of the scope. (Another problem (30))
3. Algorithm clarification: The algorithm uses an idea called from specific to general occasions.
   - It first considers a case that the bipartite graph is balanced (Same cardinality for two bipartite) . There are several definitions worth of discussing.
     (1) Perfect Matching: Suppose the graph is $G = (V, E)$, if the cardinality of the maximum matching is t, due to symmetry, we can define the perfect matching as:

     $$M = \{(v_1, v_1'), (v_2, v_2')..., (v_{n/2}, v_{n/2}')\}$$

     where $v_i \in V_1$ and $v_i' \in V_2$, $|V_1| = |V_2| = \frac{n}{2}$. The matching is defined as :
   - Second we need to consider the general bipartite graph with following definitions:
     (1) Upper and Lower nodes: The upper node is the node with index smaller the cardinality of the match. $i \leq t$. Otherwise it is called lower node.
     (2) Upper and Lower edges: Upper edge connects two upper nodes, and all other situations are lower edges.
4. The idea to deal with the normal bipartite graph is to first calculate the perfect matching part in a graph G', then use a directed graph H, which is defined as drawing an edge in H if exists an edge in G s.t. $\{v_i, v_j'\} \in E$. By concatenating the new graph H with the original derived G', we can get the maximum

Algorithm 1: find max matching edge from perfect graph

Input : A perfect bipartite graph G, that can be perfectly matched with V_1 and V_2
Output: All maximally-matchable edges

```
Function find_max_matchingEdge(G):
    Set all the edges from G.E as not maximally-matchable;
    H = G.direct_graph();
    for all edges e in H.edges do
        if e.start and e.end belong to the same connected component of H then
            Append e' corresponds to e in G to the list of maximally-matchable edges;
        end if
    end for
    Append all (v_i, v_i') ∈ E to the list of maximally-matchable;
end
```

Now it comes to the common case.

Algorithm 2: find max matching edge from general graph

Input : bipartite graph G without constraints, given maximum matching
Output: Maximally matchable edges in $Graph.Edges$

```
Function general_case_find(G):
    E.lower_edge.maximally_matchable ← True;
    E.upper_edge.maximally_matchable ← False;
    G_u ← the perfect bipartite sub-graph from G;
    find_max_matchingEdge(G_u);
    for Each of the node from G.V.left as source do
        for Each of the node from G.V.right as sink do
            Construct directed graph H_L;
        end for
    end for
    for Each of the node from G.V.right as source do
        for Each of the node from G.V.left as sink do
            Construct directed graph H_R;
        end for
    end for
    while H_L.BFS()! = ∅ do
        Each visit mark the corresponding e' in G as maximally-reachable;
    end while
    while H_R.BFS()! = ∅ do
        Each visit mark the corresponding e' in G as maximally-reachable;
    end while
end
```

Time complexity

Since each operation in the algorithm before is linear, and for each edge and each node, it will consume linear time, the total time complexity is

$$\mathcal{O}(|V| + |E|)$$

References

- TUM, Edmond's Blossom Algorithm https://www-m9.ma.tum.de/graph-algorithms/matchings-blossom-algorithm/index_en.html
- Lszl Lovsz; M. D. Plummer (1986), Matching Theory, North-Holland
- Tamir T (2011), Finding all maximally-matchable edges in a bipartite graph, The Open University