Homework 3

Ji Jiabao

2020 年 4 月 3 日

Exer.4:

Obviously the edge connecting the left part of the graph and the right part must be included in a spanning tree. To construct a spanning tree, we now pay attention on these two subgraphs.

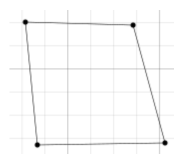


图 1: Simple graph

We first consider such a simple graph with 4 edges. Without any effort, it has 4 spanning trees, and this result can be applied to a graph with n edges.

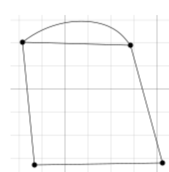


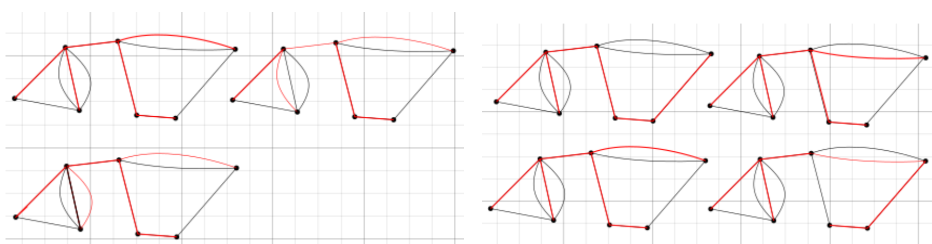
图 2: Simple graph

Then we consider a simple multi-graph case. If we remove one of the parallel edges in the multi-graph, it's exactly a simple graph discussed above. So obviously, we know it has $4 \times 2 = 8$ spanning trees.

(a) left part

(b) right part

Based on these observations, we consider the subgraphs in *Exer.*4. Left part has 3 parallel edges, which leads to $3 \times 3 = 9$ spanning trees, and right part has $2 \times 4 = 8$ spanning trees. Due to the simple counting method, there're $9 \times 8 = 72$ spanning trees for the graph.Below is some of them.



(a) left part examples

(b) right part examples

Exer.5.

We already know $Prim's$ and $Kruskal's$ algorithm to construct one of the MSTs of a graph, and only those edges having the same weight may lead to different spanning trees and we have a magic polynomial-time algorithm for cumputing the number of spanning trees. Based on these tools, we just need to find do some midifications for $Kruskal's$ algorithm.

We know that during the process of $Kruskal's$ algorithm, it gets the edges with different weights of an MST. Having these weights, we can get all other *candidate* edges for another MST as long as its weight is the same as one of the known MST's edge's. After that, we construct a subgraph g of graph G, and each spanning tree of g is a MST of G. Call the magic algorithm, we get the number of spanning tree of g, which is also the number of MSTs of G. The pseudocode is shown below.

Data: graph G
Result: number of MSTs of G
T = Kruskal(G);
$toMergeEdges = \emptyset$;
for $e \in T(E)$ **do**
  for $e_{tmp} \in G(E)$ **do**
    if $e.weight == e_{tmp}.weight$ **and** $e \neq e_{tmp}$ **then**
      $toMergeEdges = toMergeEdges \cup e_{tmp}$;
    end
  end
end
$T(E) = T(E) \cup toMergeEdges$;
$result$ = ComputeSpanningTreeNumber(T);
return $result$

Algorithm 1: Compute number of MST

The running time for this algorithm is really simple, since we know that $Kruskal$ and $ComputeSpanningTreeNumber$ are polynomial-time. And finding $toMergeEdges$ need at most $|E|^2$ comparisons,namely $O(|E|^2)$, the merge for $T(E)$ and $toMergeEdges$ is $O(|E|)$. In all running time is the sum of these polynomial subroutines, so in all the running time is polynomial.