Homework 3

Team: nogg[*†]

October 31, 2016

Exercise 1

1. **Definition 1** (Cut Lemma)**.** *Suppose edge set X is good, pick any vertex set $S \subseteq V$ s.t. there is no edge in X that goes from S to $V \backslash S$. Let $e \in E$ be the edge going from S to $V \backslash X$ with the cheapest weight, then $X \cup \{e\}$ is also good.*

   Proof.

   (1) If the cheapest edge e happens to be in the tree T, then the case is trivial.

   (2) If the cheapest edge e is not in the tree T, since T is already a tree, adding any edge to it will result in a circle and there must exist another edge e' which also goes from S to $V \backslash X$. If we remove this edge e', we will get another graph $T' = T \cup \{e\} - \{e'\}$. A sample graph is shown in Fig.1. Next, we are going to prove that it is also a minimum spanning tree.

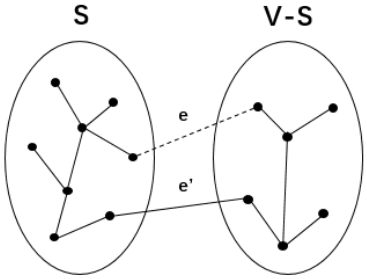


Figure 1: Picture to illustrate the cut lemma

   (a) First, we prove that T' is a tree. Since T is a tree, adding a edge to it will form a circle. Then we remove the edge e' from $T \cup \{e\}$ where e' is part of a

[*]E-mail: kimi.ysma@gmail.com
[†]Team member: Ma Yesheng, Zhao Ming, Hu Hu, Zou Yikai, Fan Minghua

circle and removing it will not disconnect the graph, hence $T' = T \cup \{e\} - e'$ is also connected. On the other hand, in the connected graph T', $|E| - |V| = 1$, therefore T' is a tree.

(b) Next, we prove that T' is a minimum spanning tree. Since substitute e' for e will not affect spanning property of minimum spanning tree, all we need to prove is it takes minimum weight. From the equation $weight(T') = weight(T) - w(e) + w(e')$, since e' is chosen to be the edge with minimum weight, thus $weight(T') < weight(T)$. Therefore T' is a minimum spanning tree.

Combine (1) and (2), cut lemma is proved.

□

2. *Proof.*

To prove this lemma, we construct a situation that meets the conditions of this lemma. And then we will prove that it must exist. In the following, we will illustrate that how we construct it and prove it must exit.

The construct process

(1) Construct the initial cut S
We divided X to different connected sets $X_1, X_2, ...X_n$. Then consider the following situations, we will construct a connected initial cut S which includes only one vertex of the given edge e and no edge from X crosses it.

a. If only one vertex of the given edge e is in a X_i, then we construct the initial cut S with vertice in X_i.

b. If two vertice of e are in X_i and X_j, then we can construct initial S with vertice in X_i or X_j.

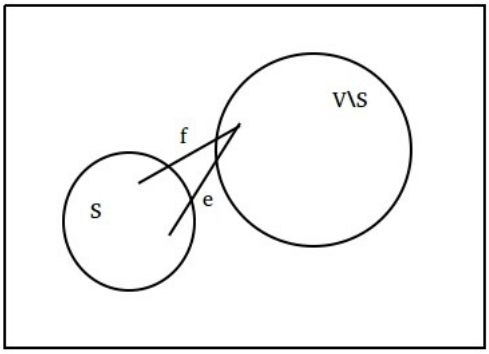
c. If none of vertice of e it in any X_i. Then choose any X_i, there must be a route g in MST connects e and X_i. Here we construct initial S with vertice in route g(include one vertex of e) and X_i. If g goes through any other X_j, add it to S.

(2) Update the cut S
We update the initial S constructed in (1), then we will ensure that e is the minimum weight edge of G crossing this cut.

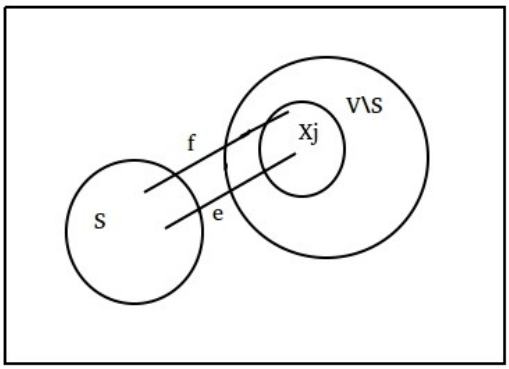
a. Find the minimum weight edge f crossing S to $V \backslash X$.

b. If f is not e, add the other vertex of f to S.If the other vertex of f is included in other X_j, add the whole vertice in X_j to S. Return to step a.
According to **Cut Lemma**, f must belongs to $E(T)$. And the S must be connected all the time we add vertex to it.
Here are 2 situations we should consider.

I. The other vertex of f is the other vertex of e. As the picture shown blow. Here f and e are all belong to $E(T)$, and vertice in S are connected by edge in $E(T)$, so it must be a cycle in $E(T)$, which is impossible.



II. The other vertex of e is in the X_j that include the other vertex of f. As the picture shown below. Here f adn e are all belong to $E(T)$, and vertice in S and X_j are connected by edge in $E(T)$, so it must be a cycle in $E(T)$, which is impossible.



c. If f is e, stop. Here, for cut S, there is no edge from X crosses it. And e is a minimum weight edge of G crossing this cut.

To sum up, we can always construct a cut S that meets the conditions in reverse cut lemma. Therefore, this lemma is proved.

□

Exercise 4

1.

2. **If:**
   Proof by contradiction. Assuming that u and v are connected in G_c, but not connected in T_c, which means two things:
   1. In T, there exists at least one edge with weight more than c in the route from u to v.
   2. In G_c, there is more than one routes from u to v with all edges' weight less than or equal to c.
   It's obvious that the special edge in 1 is not within G_c.Then according to the construction process of MST T using Kruskal algorithm, if there is no cycle, then we first add the lowest-weight edge into the MST. So since the special edge is of more weight than all the edges in the routes of Gc which connect u and v, then it shouldn't be added into T until one route from u to v is constructed. Then we can know that in MST T, u and v are connected.
   Only if:
   Since T_c is a subset of G_c, then if u,v are connected in T_c, then it is connected in G_c.

Exercise 7

1.

2. *Proof.* Considering Kruskal algorithm, we can prove this by induction on the lemma that if at one state of Kruskal algorithm two spanning trees have the same vertex set, then after adding all the edges with same length, the current vertex sets of the two "good" tree and the number of added edges are the same.

   First, we prove that the vertex sets at each stage are the same.

   (1) For the beginning state, the only vertex set is $\emptyset$, thus the lemma holds for initial state.

   (2) Suppose at one stage of Kruskal algorithm, there are two vertex sets $V_1 = V_2$. After adding all edges with the minimum weight a that will no lead to circles, we have vertex sets $V_1{}'$ and $V_2{}'$. Suppose $V_1{}'$ and and $V_2{}'$ are not the same, consider vertices in $V_1{}' - V_2{}'$, we can connect vertices in V_2 to $V_1{}' - V_2{}'$ via vertices in $V_1{}' \cap V_2{}'$. Thus $(V_1{}' - V_2{}') \cap V_2{}'$ is not empty, which is obviously contradictory.

   Next, we prove that the numbers of edges added to the same vertex set at each stage are the same. First, not considering the circles, we add all the edges with the least weight at current stage. Since the original "good" graph $\{V, E\}$ in the vertex set is a tree, adding edges of least weight to it will lead to same number of n circles. To remove n circles from a graph G, we need to remove n edges. Hence the total number of added edges at this stage are the same.

   Combine the above two, we can prove that two different minimum spanning trees has the same set of edge length and the numbers of edges with same weight are the same for different MSTs, which is a stronger result than the lemma. □

Exercise 8

Proof.
Supposed there are two MST T and T' of the same graph G.
Then:

$$\sum_{i=1}^{n-1} e_i = \sum_{i=1}^{n-1} e'_i \quad (e_i \in E(T), e'_i \in E(T'))$$

$$|E(T)| = |E(T')| = n - 1$$

According to the question, if no two edges of G have the same weight. Assumed T and T' is not the same tree.

$$T: \ e_1, e_2, ..., e_{n-1} \quad (|e_1| < |e_2| < ... < |e_n|)$$

$$T': \ e'_1, e'_2, ..., e'_{n-1} \quad (|e'_1| < |e'_2| < ... < |e'_n|)$$

Due to they are different and G has no same weight edge, there must be $e_i < e'_i$ and with the e and e_i less than e_i is all same. Given $c = e_i$ then $m_c(T) < m_c(T')$, which contradicts the lemma proved in the last exercise. So it is impossible.

Hence, T and T' is the same. This lemma is proved.

□

Exercise 10.

Solution.
There are two connected components, so we can just count the spanning tree number of each connected component, and the spanning forests number is the multiplication of every spanning tree's number.
Considering the first graph, there are 2 x 3 = 6 spanning trees.
Considering the second graph, there are 5 spanning trees.
Totally, there are 5 x 6 = 30 spanning forests.

□

Exercise 11.

Solution.
Firstly, let's recall the procedure of Kruskal algorithm. Step by step, we choose the most cheapest edges, and check whether adding it to the answer will cause a cycle. If not causing a cycle, we will add it to the final graph, which is a minimum spanning tree.
Considering the graph that has edges whose weights are the same. We will find that we must check every edge with same weight.
Based on the finding above, we can gather the edges with same weight as a new graph, and every time, we just add one of the spanning tree of this graph to the answer and check whether there is a tree. After trying every possible spanning tree, we will get several possible answers. And finally we will get many minimum spanning tree.

So according to the problem, we have the algorithm that can calculate the spanning tree number of multigraph in polynomial time. So every time we consider the sets of edges with same weight, we can calculate the number of spanning tree in polynomial time.
So in the worst case, we will need n * polynomial-time(n is the edge number). And it is still polynomial-time algorithm.
For example: we have the following weighted graph.

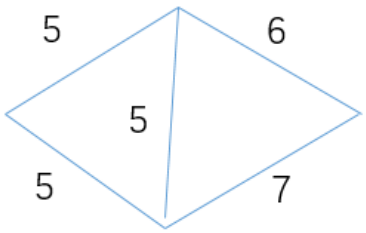


Figure 2: the weighted graph

At the first step, there are 3 edges with weight 5, and there are 3 spanning of this multigraph.

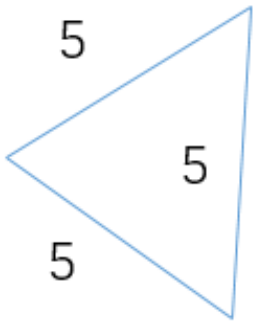


Figure 3: the multigraph of edge with weight 5

And we will add edge with weight 6. And we can get the spanning tree.
So we figure out that there are 3 minimum spanning tree.

□