Homework 3

Team: nogg[*,†]

November 8, 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 $V \backslash S$ 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 $V \backslash S$. 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.

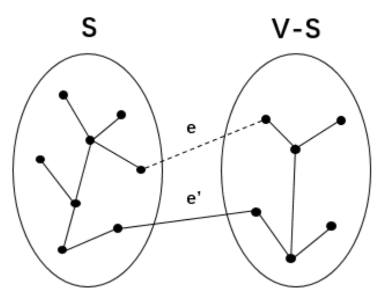


Figure 1: Picture to illustrate the cut lemma

   (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.

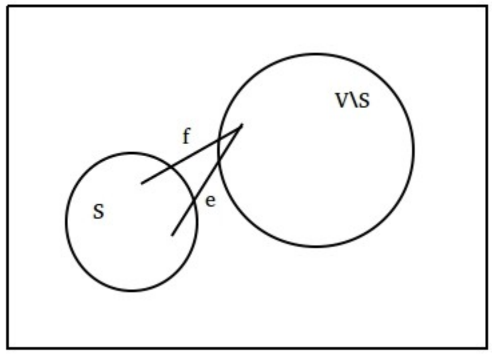
(2) Update the cut S

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

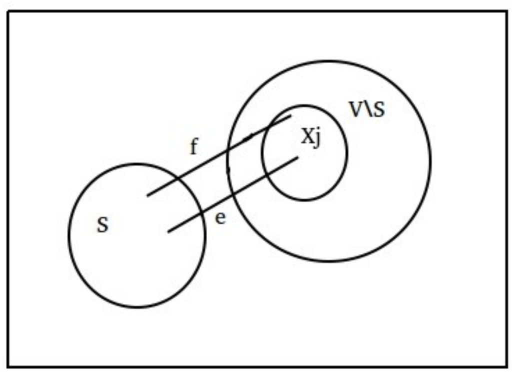
a. Find the minimum weight edge f crossing S to $V \backslash X$.

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.

$\square$

Exercise 4

1. Let G be a weighted graph as shown below, T is the MST of G. Assuming c is 2, then G_c and T_c are both shown as graph 3. So obviously they have the same connected components.

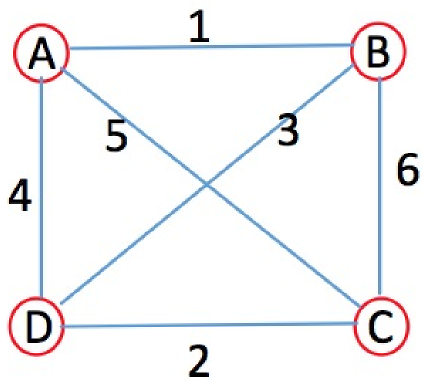


Figure 2: 4-1

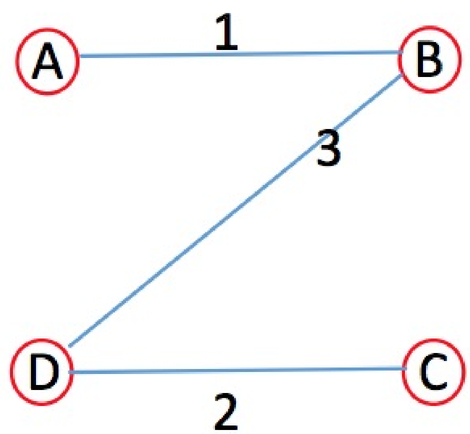


Figure 3: 4-2

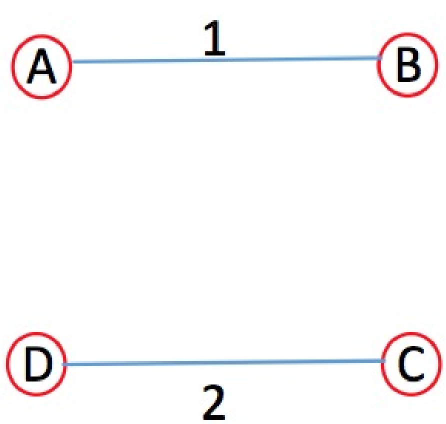


Figure 4: 4-3

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 at least one route from u to v with all edges' weight less than or equal to c.
Now let's assume that in 1 there is only one break edge(x,y) in the route(It's similar but more complex when there are multiple breaks, but after all they are similar). Define set S containing x and those which are connected to x in T_c. Then V/S contains y and those which are connected to y in T_c. According to cut-lemma, we know that in G, because edge(x,y) appears in MST T, then edge(x,y) must be the shortest edge from cut S to cut V/S. As a result, in G_c, there will be no edge from cut S to cut V/S, then u and v should not be connected. This is a contradiction.
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.

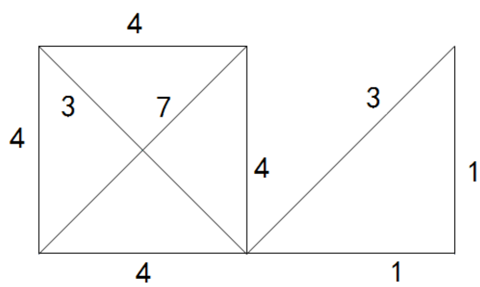


Figure 5: The original graph

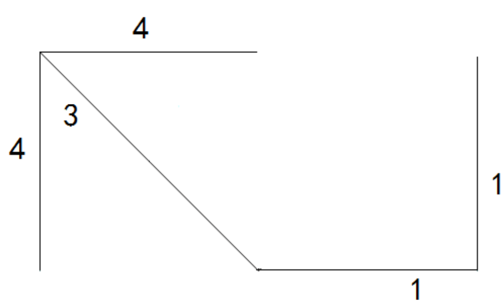


Figure 6: MST for Fig.2

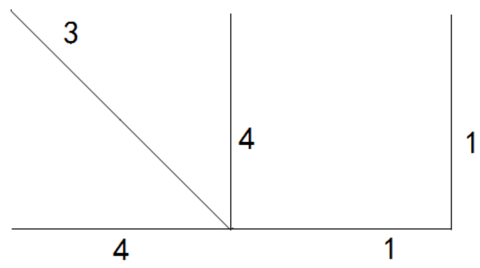


Figure 7: Another MST for Fig.2

2. *Proof.* We prove this lemma by contradiction. Suppose there exists a c s.t. $m_c(T) \neq m_c(T')$.
   First, we can prove that the value of $m_c(T)$ can only decrease when c is equal to some e_i in the graph since the edges in a MST are all edges in the original graph G. Therefore we can only consider the case that c is equal to some e_1 and W.L.O.G we can sort these edges of a graph by weight.
   Next, we prove that if we remove all the edges of weight c' that is previous to c in the sorted edge list, then $m_c(T) - m_{c'}(T) = m_{c'}(T') - m_{c'}(T')$. If this does not hold, we will immediately get that the number of connected components in $T'_{c'}$ and $T_{c'}$ are not equal since each removed edge in MST will lead to a new connected component. However, we have proved in lemma 3 that $T_{c'}$ and $G_{c'}$ have the same connected components and $T'_{c'}$ and $G_{c'}$ have the same connected components. Thus $T_{c'}$ and $T'_{c'}$ have the same number of connected components. These facts lead to a contradiction.
   Finally, by mathematical induction on the sorted edge list. we will get that $m_0(T) \neq m_0(T)$, which means that $0 \neq 0$, which is impossible.
   To conclude, there is no c s.t. $m_c(T) \neq m_c(T')$, i.e. $m_c(T) = m_c(T')$ for a graph G. □

Exercise 8

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

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

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

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 + 1 = 7 spanning trees.
Considering the second graph, there are 2 x 3 + 1 = 7 spanning trees.
Totally, there are 7 x 7 = 49 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.

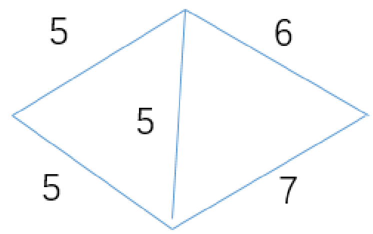


Figure 8: the weighted graph

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

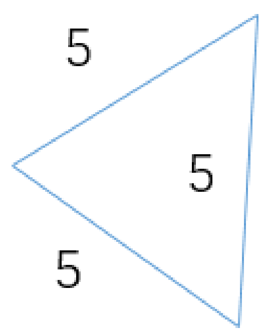


Figure 9: the multigraph of edge with weight 5

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.

□