HW12

1. *Let G = (V, E) be an undirected, weighted graph. Let U be a subset of the vertices. Give an algorithm to construct a minimum weight spanning tree T in which each vertex u ∈ U is a leaf. You may assume there is such a spanning tree. Your algorithm should have the same asymptotic running time as one of Prim's or Kruskal's MST algorithms (your choice). Justify your running time, and explain why your algorithm is correct.*

   Hint. If you remove U from T what do you obtain with respect to the vertex set V − U ?

   > If we want to make sure all of the vertex $u \in U$ is a leaf in the minmimum weight spanning tree T, there should be no edges connecting the vertices in U. For example, say $u_1, u_2 \in U, v_0 \in V - U$, if there is an edge (u_1, u_2), there may have a path in T that passes through these three vertices, like a path v_0, u_1, u_2, then u_1 is not a leaf any more since it has a child u_2.
   >
   > We construct a graph $H(V - U, E \cap ((V - U) \times (V - U)))$. We can get a minimum weight spanning tree T' first by H, then for each $u \in U$, connect u to T' with minimim distance.
   >
   > Here is a brief example, we use Prim's algorithm here. The difference is that, we enqueue the vertex $V - U$ first to construct a MST T' with vertices in $V - U$. Then we enqueue the subset U, to continue construct MST T by connecting the original MST T' with the vertices in subset U. To be noticed, all the edges are in subset $E \cap ((V - U) \times U)$.
   >
   > ```
   > EnQueue(Q, V-U, Dist);
   > while Q not empty do
   >   for each edge (u,v) with v ∈/ T and v in E∩((V-U)×(V-U))
   > .....
   > end while
   > // we enqueue the vertices in U here,
   > // so the MST by V-U can connect the vertices in U
   > EnQueue(Q, U, Dist);
   > while Q not empty do
   >   for each edge (u,v) with v ∈/ T and v in E∩((V-U)×U)
   > ....
   > end while
   > ```

2. *The following statements about MSTs of weighted undirected graph G = (V, E) are either true or false. For each statement, either prove it is true or give an example to show it is false.*
   a. Let e be the unique heaviest edge on a cycle C in G (i.e. every other edge on C has smaller weight than e). Then e cannot be part of any MST of G.

   > True. Assume there are n vertices in C, then $n - 1$ edges are needed to connect all the vertices in C. Acoording to Kruskal's algorithm, we always choose the lightest weight edges to construct a MST. For a circle, we only need to remove one edge in the circle. To make sure the sum of the weight is smallest, we always remove the heaviest one.

b. If e is part of some MST of G, then e must be a lightest edge across some cut of G (there may be other edges of the same weight as e crossing this cut).

> True. By cut lemma, the minimum weight edge accross a cut must be in the MST if the edge's weight is strictly smaller than all other edges accross the cut. So e mush be a lightest edge.

c. Prim's algorithm works correctly when there are negative weight edges.

> True. Because Prim's algorithm works by giving the smallest weight and do checks and union, negative weight edges does not affect the procedure.

d. Let U ⊂ V and let H = (U,E∩(U×U) be the subgraph of G induced by vertex set U, i.e. including all the edges between vertices in U. Suppose that H is connected. Let T = (V,D) be an MST of G. Then the forest F =(U,D∩(U×U), the portion of T joining vertices in U, is contained in some MST of H.

> True.
>
> If no edges e in $E \cap (U \times U)$ is used to construct MST of G, then $E \cap (U \times U) \cap D = \emptyset$, so $F = (U, D \cap (U \times U)) = (U, \emptyset)$, is contained in some MST of H, because $\emptyset$ is in any of MST.
>
> If there are edges e in $E \cap (U \times U)$ is used to construct MST of G. e must be the smallest weight edge during the cut in E, as $E \cap (U \times U) \subset E$, then it is also the smallest weight edge during the cut in $E \cap (U \times U)$. Since $e \in D$ and $e \in (U \times U)$, $e \in D \cap (U \times U)$, it is contained in some MST of H.