CS/ECE 374 ✧ Spring 2017

Homework 9 Question 2

Due Wednesday, April 19, 2017 at 10am

Ho Yin Au (hoyinau2), Lanxiao Bai(lbai5), Renheng Ruan(rruan2)

- Question 2
  To compute such spanning tree, we can first use Kruskal's algorithm on only the set of vertices $V(G)-U$, where G is the provided undirected graph, and $U \subset V(G)$ is the subset of vertices of unreliable nodes. Then, base on the output spanning tree T of $V(G)-U$, use Kruskal's algorithm on G that with vertices in U and consider only the edges connecting vertices in U and vertices not in U to make sure vertices in U are leaves of the tree.
  To prove the strategy is correct, we need to prove the tree returned is minimal while satisfying the constraint vertices in U are leaves. As tree returned by Kruskal's algorithm is MST, which is proved in lecture, we can prove that the tree returned is minimal while satisfying the constraint using Kruskal's algorithm as black box.
  Prove by contradiction: Assume Kruskal's algorithm will give a MST with provided set of vertices and edges without considering constraint. Assume the above strategy build a optimum tree T, and there exists a better tree T' that has smaller total edge weight and has 1 edge differ from T, and it still satisfy the vertices in U are leaves constraint. Then, let $e = (u, v) \in E(T) - E(T')$, and $e' = (u', v')inE(T') - E(T)$. From the assumption, $e \neq e'$, and $w(e') < w(e)$ where $w(e)$ is the weight of edge e. There are 3 case to consider the edge e:

  - case $e = (u, v)$ that $u, v \notin U$:
    If $u, v \notin U$, e is picked by Kruskal's algorithm while considering $V(G) - U$. Edge $e' = (u', v')$ must also connect $u', v' \notin U$ as we already connect all vertices in U as leaves in a tree, and we cannot connect leaves to leaves. As $w(e') < w(e)$, e' will be picked first by Kruskal's algorithm. As Kruskal's algorithm should give tree T' instead of T, this contradict Kruskal's algorithm always give MST and that T (instead of T') is the tree output from the strategy.
  - case $e = (u, v)$ that $u, v \in U$:
    This means T violate the vertices in U are leaves constraint, contradict assumption T follow the constraint.
  - case $e = (u, v)$ that $u \in U, v \notin U$:
    The edge e is picked when we run Kruskal's algorithm with tree already connecting all vertices not in U, and consider only the edges connecting vertices in U and vertices not in U. As we consider that type of edges, delete edge e makes the vertex u isolated. We need to connect u by edge e', so $u = u'$ and $v' \notin U$ as we still cannot connect leaves to leaves. As $w(e') < w(e)$, e' must be picked first by Kruskal's algorithm. As Kruskal's algorithm should give tree T' instead of T, this contradict Kruskal's algorithm always give MST and that T (instead of T') is the tree output from the strategy.

  As we arrive contradiction in those 3 cases, we can conclude that the above strategy will build a optimum tree T satisfy the constraint. As we use Kruskal's algorithm as a black box, we can also know that if there is no tree possible from the Kruskal's algorithm.

```
MSTWithVerticesInUAreLeaves(Graph G, VertexSet U) :

    VertexSet goodVertex ← V(G) − U
    tree ← empty graph

    tree ← Kruskal(goodVertex, E(G), tree)

    if tree = None
        return None

    EdgeSet leavesEdge ← {(u, v) for u ∈ U, v ∉ U}

    return Kruskal(V(G), leavesEdge, tree)
```

```
Kruskal(VertexSet V, EdgeSet E, Graph tree) :

    E ← (u, v) ∈ E where u, v ∈ V
    E ← (u, v) ∈ E order by weight(u, v), increasing

    for (u, v) ∈ E
        if u ∉ V(tree) or v ∉ V(tree)
            V(tree) ← V(tree) ∪ {u, v}
            E(tree) ← E(tree) ∪ {(u, v)}

    return tree
```