電機三 B08901158 吳詩昀

Algorithm 2021, Programming Assignment #3

Graph Structure

- **Weight Matrix**: An n x n matrix. For element $[i, j]$, the value represent the weight of edge from i to j. $\Longrightarrow$ Total space: $\theta(n^2)$
- **Edge List**: An m x 3 matrix. Store all edges with weight by using a size 3 vector $[i, j, W(i, j)]$. $\Longrightarrow$ Total space: $\theta(3m)$
- **Removed Edge List**: Store removed edges by using a size 3 vector $[i, j, W(i, j)]$
- **Removed Weight**: The accumulated weight removed.

Undirected Graph

> Rough idea:
>
> 1. Use Kruskal algorithm to find maximum spanning tree.
> 2. Remove all other edges that are not in the spanning tree.

Remove by Kruskal: First, create disjoint set for all vertices. Then sort all edges by its weight in decreasing order. Pick the edges in order and union until all vertices are visited and in same set. For edge (u,v) that `FindSet(u) == FindSet(v)` add to the removed list and add its weight to the removed weight.

$\Longrightarrow$ Total time: $\theta(mlgm + n)$

- **Disjoint Set:** Implement by using path compression.
- **Sort Edges:** Using merge sort.

Directed Graph

> Rough idea:
>
> 1. find ~~max~~ cycle by DFS → seems like it doesn't matter if it is max
> 2. remove min cost edge
> 3. repeat until no cycle
> 4. Remove all negative redundant edges.

Remove by DFS: While existing a cycle, it will continue to do DFS to find cycle, using the idea that if back edge exist then there is cycle. When the cycle is found, it will search for the valid minimum−cost edge and removed it. To check if the edge can be removed, verify the connectivity after the removal by performing another DFS. However, this time see all edges as undirected. Lastly, remove redundant negative edge in order of increasing order such that the cost can be reduced and the graph is still weakly connected.