演算法 Homework 20191128
山下夏輝（Yamashita Natsuki）
R08922160

Mini HW #7

繳交作業

截止時間 星期四， 14:20 **總分** 1 **繳交** 線上輸入或者檔案上傳 **檔案類型** pdf
接受繳交時間 直到 12月5日 14:20

Wayne has decided to pour water into his N block farmland(index from 1 to N). There are two ways to irrigate a piece of land, to drink water from other farmland, or to build a reservoir of land. Connecting two pieces of land takes P_{ij}, construction of a reservoir takes W_i. Please design an algorithm to help Wayne calculate minimal required price in $O(N^2 \log N)$.

(Algorithms taught in class can be used without implementation, but you should briefly explain how you use them.)

```
MIN_PRIM_MOD(G,p,w)
        Sort G into non-decreasing order by w
        For u in G.V
                u.key = infinity max
                u.π = NIL
        r.key = 0
        Q = G.V
        Temp_cost=0
        While Q ≠ empty
                u = EXTRACT-MIN(Q)
                cost += temp_cost    //accumulate the min cost in adj edges
                for v in G.adj[u]        //find minimum(p_ij, w_j)
                        if v ∈ Q && p(u,v) < v.key
                                v.π = u
                                v.key = p(u,v)
                        elif v ∈ Q && w(v) < v.key
                                v.π = u
                                v.key = w(v)
                        temp_cost = Min(temp_cost, v.key)
        return cost
```

This problem is basically same as Minimal Spanning Tree Problem. The point which this problem is different from MST is that there would be multiple trees. For making multiple trees, we should distinguish which vertices have already calculated into minimal cost and which vertices have not yet. So the Prim's Algorithm is a good choice to be applied to this

problem because Prim’s Algorithm would check vertices one by one and extract vertex from the set to distinguish which vertices are still in the set or not.

The cost is accumulated after finding minimal p(u,v) or w(v) in the adjacent of u because the way to pour has to be picked one of two.

The time complexity of this is O(mlogn) with binary min-heap or O(m+nlogn) with Fibonacci heap when there is no other condition in this problem.