演算法Homework 20191128

山下夏輝（Yamashita Natsuki）

R08922160

A screenshot of a cell phone

Description automatically generated

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(pij, wj)

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.