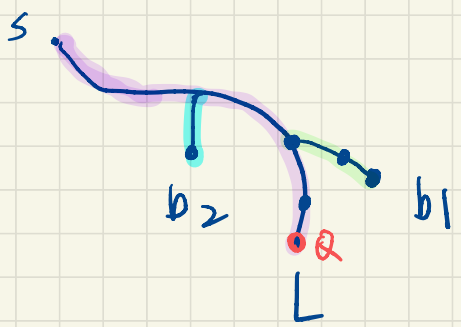
P2

[Think] 依序找出最長 branch (L)，2nd 長 branch (b_1)，3rd ..... (b_2)

則 ANS = 0 , L , $L+b_1$, $L+b_1+b_2$, ...



backup = { }
(存備選 branch)

① 先用 dfs 從任意點開始找到最長 path 的端點 Q

② 以 Q 為起點，再做一次 dfs.
過程中，每個 Node 都要記錄"從這個 Node 最遠可以走多遠"。
就像 DP， $DP[u] = \max_i (DP[Node.v_i] + w_{u,v})$

較小的 branch 值要保留到 backup 裡面。

全部做完之後，sort backup，則 backup = { $b_1, b_2, \ldots$ }

③ 由②的 dfs 可得 L，則依序 pop backup 中的 branch 可得
0 , L , $L+b_1$, $L+b_1+b_2$, ... ☆

P3

[Think] ① 把所有可以形成 MST 的边收集成一個新的 Graph G'

② 在 G' 中跑 Dijkstra 即為所求。

<Case 1> 所有 weight 皆相異：MST 唯一

① 做一次 Kruckal 得 MST = G' → Group 2

②

<Case 2> 有相同 weight：MST 不唯一

設把 edge weight 排序後為 a, b, c_1, c_2, c_3, d, ...

(c_1, c_2, c_3 weight 皆為 C 的 edge)

原本的 Kruskal：

```
for a edge in edges:
    如果和前面的边不形成 cycle:
        可以形成 MST → 加入 G'
```

⇒ check c_1, c_2, c_3 分別會不會和 a, b 形成 cycle，再一次加入 G'

變成：

```
for a edge in edges:
    如果 waited list 不為空:
        如果 waited list 裡面的 weight 和 a edge 不同:
            把 waited list 裡的 edge 都加進 G'
    如果和前面的边不形成 cycle:
        把 a edge 加入 waited list.
```

[NOTE] Check cycle by disjoint set. (union by rank)