1. **Priority Queue : Recap**
   If pair in priority queue : (x, y) if x equal than compare with y.
   Time Complexity of push & pop : O (log n)

2. **Optimized Dijkstra : Pseudocode & complexity {Sparce Graph: E<<<V}**
   Normal Dijkstra = O (n^2) → Optimized Dijksatra = O (n^2 Log n)

   While (!pq.empty()): O (E)
   Pick the node with minimum distance value from priority queue : pq.front();
   pq.pop(); **O (log E)**
   If visited[head] == 1: ignore
   If visited[head] == 0: set it 1 ;
   Take a reverse [Minimum] priority queue pq;
   pq.push({distance, src_node})

   After relaxation :
   pq.push({distance[adj_node], adj_node}) **O (log E)**

   Time Complexity :
   O(E Log E) + O(E Log E) = **O (|E| log |E|)** { |E| = Number of Edges }
   Worst Case : **E→V^2**

   So,
   O (|E| Log $|V^2|$)
   =>O (2|E| Log |V|)

   O (|E| Log |V|) => O(E Log V) [Base : 2]

   Optimization : O(n^2) → O (E Log V) {Sparse Graph}
   New Time Complexity : O (E Log V)
   Space Complexity : O (V) → O(E)

3. **Solving on Codeforces :** *https://codeforces.com/problemset/problem/20/C*