**Problem1** : [Flight Discount](https://cses.fi/problemset/task/1195)

**Approach** :

-> Its just a modification to classic dijkstra implementation.

-> **Priority Queue Entry** contains : {dist\_to\_node,{node\_id,coupon\_used\_or\_not}}

-> **minDist** contains smallest dist to a node from src 1) with index 2) without index

-> Here for every iteration we add distance to a node **from src with coupon & without coupon in the Heap** (if it’s better than what’s previously stored in minDist(with or without coupon) ).

-> The rest is explained in Implementation :

**Code :** <https://ideone.com/eStv84>

**Problem2 (better version):** [**https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/practice-problems/algorithm/shortest-path-revisited-9e1091ea/**](https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/practice-problems/algorithm/shortest-path-revisited-9e1091ea/)

**Approach :**

**-> implemented dijkstra with set ,so that only 1 entry for a node remains at any time in the queue.**(can also be done with priority queue).

**-> Unique State = dp[i][j] ,** means shortest path distance to node ‘i’ using ‘j’ coupons.

->In the set implementation,whenever a node 'u' is popped from queue with some 'distance', and any number of coupons,then it's guaranteed that now we can't reach that node with lesser distance .

REASON : bcoz we have ordered nodes based on distances, and the node popped right now e.g ‘u’,has the lowest distance currently from the source amongst all the other nodes visited with any number of coupons , in the set,then how can we ever reach ‘u’ with lesser distance with same number of coupons? **So when a node is popped for the first time , then the distance stored would be the answer for that node.**

**Full explanation :** [**https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/practice-problems/algorithm/shortest-path-revisited-9e1091ea/discussion/approach-and-code-1ee91232/**](https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/practice-problems/algorithm/shortest-path-revisited-9e1091ea/discussion/approach-and-code-1ee91232/)

**Time = O(ElogV)**

**Code :** [**https://ideone.com/FrdUgK**](https://ideone.com/FrdUgK)