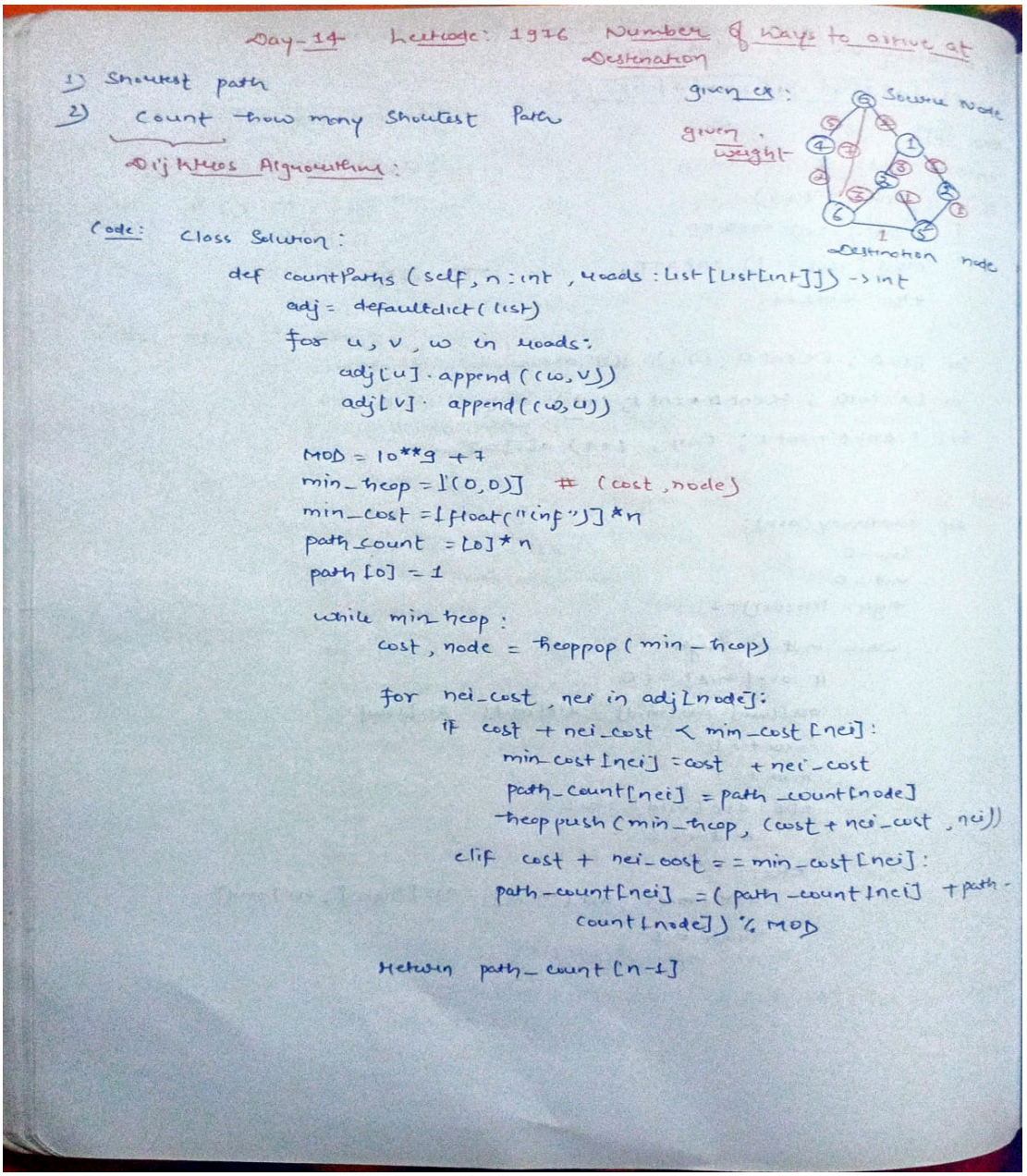
Day_14: DSA



Day-14 Leetcode: 1976 Number of ways to arrive at Destination

1) Shortest path
2) Count how many shortest Path

Dijkstras Algorithm:

given ex:

given weight

Code:

```
Class Solution:
    def countPaths(self, n:int, roads:List[List[int]]) -> int
        adj = defaultdict(list)
        for u, v, w in roads:
            adj[u].append((w,v))
            adj[v].append((w,u))

        MOD = 10**9 +7
        min_heap = [(0,0)]   # (cost, node)
        min_cost = [float("inf")]*n
        path_count = [0]*n
        path[0] = 1

        while min_heap:
            cost, node = heappop(min_heap)

            for nei_cost, nei in adj[node]:
                if cost + nei_cost < min_cost[nei]:
                    min_cost[nei] = cost + nei_cost
                    path_count[nei] = path_count[node]
                    heappush(min_heap, (cost + nei_cost, nei))
                elif cost + nei_cost == min_cost[nei]:
                    path_count[nei] = (path_count[nei] + path_
                        count[node]) % MOD

        Return path_count[n-1]
```

Q6: Stock Buy And Sell

ex: price = [7, 1, 5, 3, 6, 4]
output = 5
→ Buy in day 2 and sell on day 5

Brute:

```
def maxProfit(arr: List[int]) -> int:
    maxPro = 0
    n = len(arr)
    for i in range(n):
        for j in range(i+1, n):
            if arr[j] > arr[i]
                maxPro = max(arr[j] - arr[i], maxPro)
    return maxPro
```

Optimal: Soln

```
def maxProfit(arr)
    maxPro = 0
    minPrice = float('inf')
    for i in range(len(arr)):
        minPrice = min(minPrice, arr[i])
        maxPro = max(maxPro, arr[i] - minPrice)
    return maxPro
```

TC → O(N)