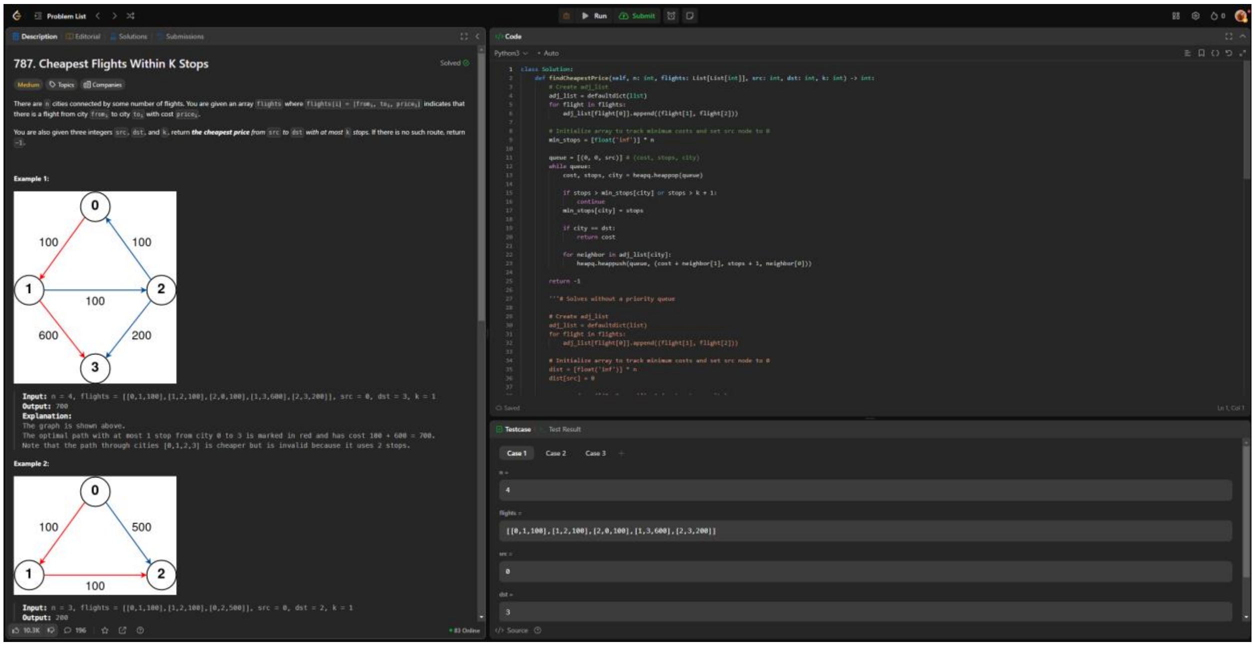
Nicholas Hersh (UNI: nh2693)

Week 1 Warmup

Problem 1

a) A web application that I've become quite familiar with over the last six months and one that I've come to appreciate over other similar platforms is www.leetcode.com. It is a platform that is accessible through a browser, and it allows users to practice questions similar to what software engineers will be asked in interviews.

b)



787. Cheapest Flights Within K Stops

There are `n` cities connected by some number of flights. You are given an array `flights` where `flights[i] = [from_i, to_i, price_i]` indicates that there is a flight from city `from_i` to city `to_i` with cost `price_i`.

You are also given three integers `src`, `dst`, and `k`, return ***the cheapest price*** *from* `src` *to* `dst` *with at most* `k` *stops.* If there is no such route, return `-1`.

Example 1:

```
Input: n = 4, flights = [[0,1,100],[1,2,100],[2,0,100],[1,3,600],[2,3,200]], src = 0, dst = 3, k = 1
Output: 700
Explanation:
The graph is shown above.
The optimal path with at most 1 stop from city 0 to 3 is marked in red and has cost 100 + 600 = 700.
Note that the path through cities [0,1,2,3] is cheaper but is invalid because it uses 2 stops.
```

Example 2:

```
Input: n = 3, flights = [[0,1,100],[1,2,100],[0,2,500]], src = 0, dst = 2, k = 1
Output: 200
```

```python
class Solution:
    def findCheapestPrice(self, n: int, flights: List[List[int]], src: int, dst: int, k: int) -> int:
        # Create adj_list
        adj_list = defaultdict(list)
        for flight in flights:
            adj_list[flight[0]].append((flight[1], flight[2]))

        # Initialize array to track minimum costs and set src node to 0
        min_stops = [float('inf')] * n

        queue = [(0, 0, src)] # (cost, stops, city)
        while queue:
            cost, stops, city = heapq.heappop(queue)

            if stops > min_stops[city] or stops > k + 1:
                continue
            min_stops[city] = stops

            if city == dst:
                return cost

            for neighbor in adj_list[city]:
                heapq.heappush(queue, (cost + neighbor[1], stops + 1, neighbor[0]))

        return -1

        '''# Solves without a priority queue

        # Create adj_list
        adj_list = defaultdict(list)
        for flight in flights:
            adj_list[flight[0]].append((flight[1], flight[2]))

        # Initialize array to track minimum costs and set src node to 0
        dist = [float('inf')] * n
        dist[src] = 0
```

n =

4

flights =

[[0,1,100],[1,2,100],[2,0,100],[1,3,600],[2,3,200]]

src =

0

dst =

3

c)

Heuristic 1: User Control and Freedom (Navigation)

Explanation: Leetcode is very simple to use. From the home page we can easily navigate to any problem we want. It offers us the ability to filter problems by name / topic / company name / difficulty / frequency and more. All one must do is filter on their desired parameter and then click into a problem. We're also able to very quickly exit a problem and move onto another one

Heuristic 2: Flexibility and Efficiency of Use

Explanation: For me, Leetcode excels here. The platform is visually appealing while making it very easy to use. We can select our programming language from a drop-down menu, all test cases are provided and we're able to execute our code without having to build out anything beyond the function we're solving for. Compared to some other platforms that force you to essentially write out a main function and provide your own test cases, the "accelerators" on the Leetcode platform allow you to quickly focus on what's important - solving the algorithms.

Heuristic 3: Aesthetic and Minimalist Design

Explanation: Leetcode, especially in dark mode, is vastly superior in terms of design compared to platforms like Hackerrank or GeeksforGeeks. With its simple design you can easily distinguish between the problem description, the test cases / test results and your code. In addition, where to “run” and “submit” your code is also clearly visible. For me at least, I find it significantly easier to use than its competitors.