Experiment-1.3

Student Name: Nabha Varshney **UID: 20BCS4995**

Branch: CSE **Section/Group: 20BCS-DM-704 (A)**

Semester: 6th **Date of Performance:01st Mar 2023**

Subject Name: Competitive Coding II **Subject Code: 20CSP- 351**

Aim – To demonstrate the concept of Heap Model

Objective-

- The objective is to build problem solving capability and to learn the basic concepts of data structures.
- The implementation of Last Stone Weight which shows and brushes up the concept of Heap and can be solved through various approaches.
- The implementation of priority queue which is max heap by default in C++.

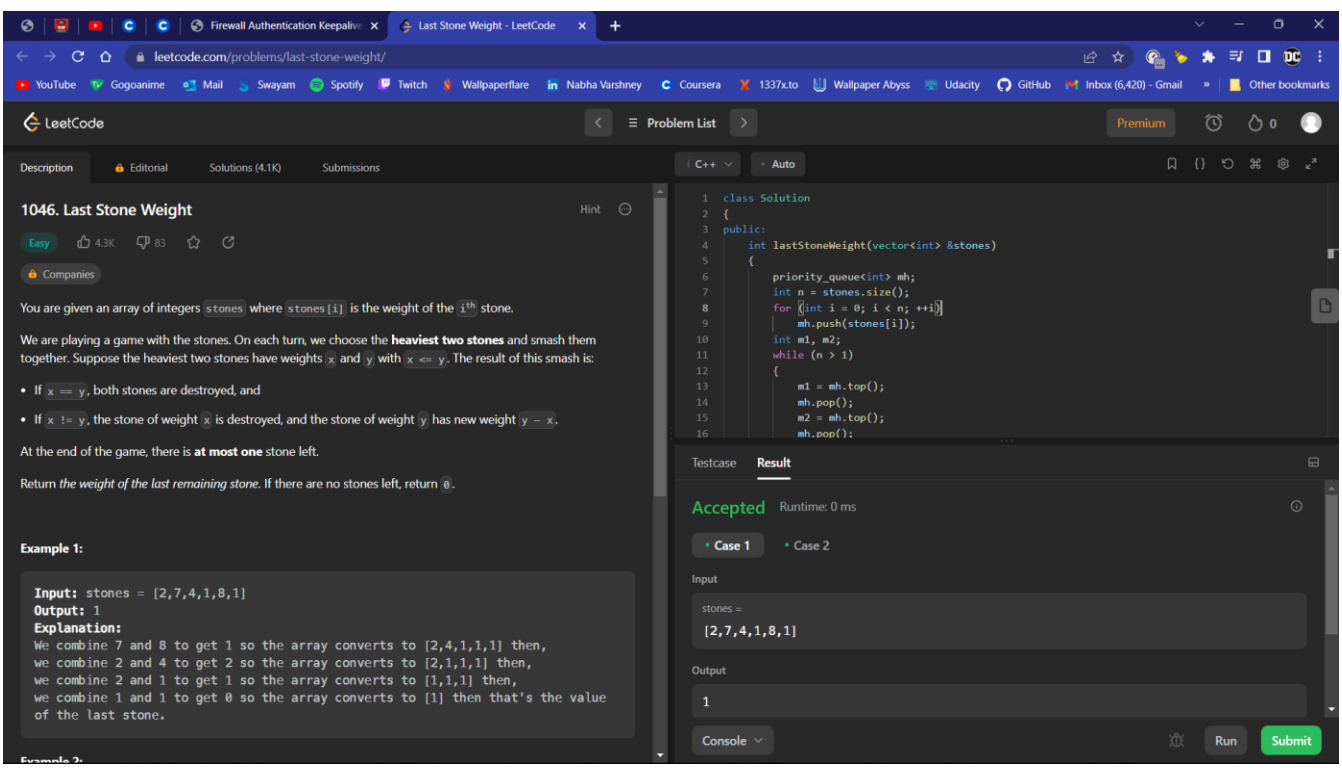
1) Last Stone Weight
https://leetcode.com/problems/last-stone-weight/

Code –

```
class Solution
{
public:
   int lastStoneWeight(vector<int> &stones)
   {
      priority_queue<int> mh;
      int n = stones.size();
      for (int i = 0; i < n; ++i)
         mh.push(stones[i]);
      int m1, m2;
      while (n > 1)
```

```
        {
            m1 = mh.top();
            mh.pop();
            m2 = mh.top();
            mh.pop();
            n -= 2;
            if (m1 - m2 > 0)
            {
                mh.push(m1 - m2);
                n += 1;
            }
        }
        if (!mh.empty())
            return mh.top();
        return 0;
    }
};
```

Output -



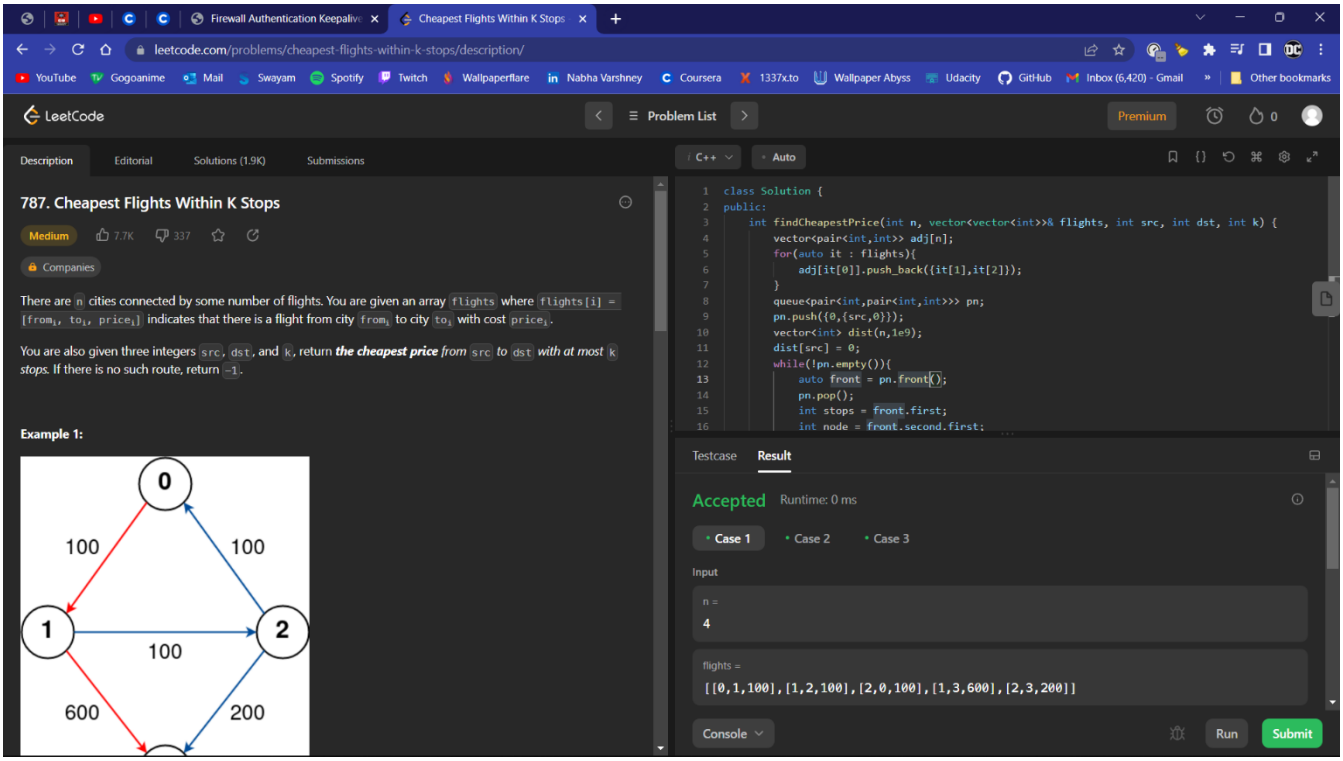
2) Cheapest flights with K shops

https://leetcode.com/problems/cheapest-flights-within-k-stops/

Code -

```
class Solution {
public:
    int findCheapestPrice(int n, vector<vector<int>>& flights, int src, int dst, int k) {
        vector<pair<int,int>> adj[n];
        for(auto it : flights){
            adj[it[0]].push_back({it[1],it[2]});
        }
        queue<pair<int,pair<int,int>>> pn;
        pn.push({0,{src,0}});
        vector<int> dist(n,1e9);
        dist[src] = 0;
        while(!pn.empty()){
            auto front = pn.front();
            pn.pop();
            int stops = front.first;
            int node = front.second.first;
            int distance = front.second.second;
            if(stops>k)continue;
            for(auto it:adj[node]){
                int adjnode = it.first;
                int d = it.second;
                if(distance + d<dist[adjnode]&&stops<=k){
                    dist[adjnode] = distance + d;
                    pn.push({stops+1,{adjnode,distance+d}});
                }
            }
        }
        if(dist[dst]==1e9)return -1;
        return dist[dst];
    }
};
```

Output -


Firewall Authentication Keepalive
Cheapest Flights Within K Stops
leetcode.com/problems/cheapest-flights-within-k-stops/description/
LeetCode
Problem List
Premium
Description
Editorial
Solutions (1.9K)
Submissions
C++
Auto
787. Cheapest Flights Within K Stops
Medium
7.7K
337
Companies
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:
0
1
2
100
100
100
600
200
class Solution {
public:
int findCheapestPrice(int n, vector<vector<int>>& flights, int src, int dst, int k) {
vector<pair<int,int>> adj[n];
for(auto it : flights){
adj[it[0]].push_back({it[1],it[2]});
}
queue<pair<int,pair<int,int>>> pn;
pn.push({0,{src,0}});
vector<int> dist(n,1e9);
dist[src] = 0;
while(!pn.empty()){
auto front = pn.front();
pn.pop();
int stops = front.first;
int node = front.second.first;
Testcase
Result
Accepted
Runtime: 0 ms
Case 1
Case 2
Case 3
Input
n =
4
flights =
[[0,1,100],[1,2,100],[2,0,100],[1,3,600],[2,3,200]]
Console
Run
Submit