Experiment 1.3

Student Name: Rohan Jaiswal
UID: 21BCS2856
Branch: BE-CSE
Section/Group: KRG-CC-1-B
Semester: 6th
Date of Performance: 30-01-2024
Subject Name: Advance Programming-2
Subject Code: 21CSP-251

1. Aim:

- To Solve the Last Stone Weight.
- To Solve the Cheapest Flight Booking with K stops.

2. Objective:

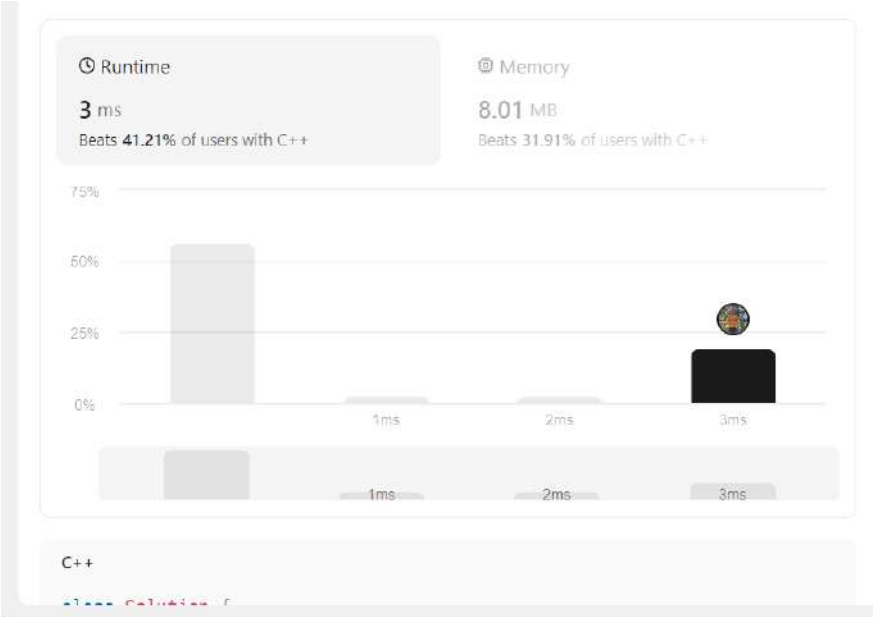
- You are given an array of integers stones where stones[i] is the weight of the ith stone. We are playing a game with the stones. On each turn, we choose the heaviest two stones and smash them together. Suppose the heaviest two stones have weights x and y with x <= y.
- There are n cities connected by some number of flights. You are given an array flights where flights[i] = [fromi, toi, pricei] indicates that there is a flight from city fromi to city toi with cost pricei.

3. Algo. /Approach and output:

1st:

```
class Solution {
public:
   int lastStoneWeight(vector<int>& stones)
   {
      int a,b;
      priority_queue<int>pq;
      for(int i=0;i<stones.size();i++)
      {
         pq.push(stones[i]);
      }
      while(pq.size()!=1)
      {
```

```
            a=pq.top();
            pq.pop();
            b=pq.top();
            pq.pop();
            int c=a-b;
            pq.push(c);
        }
        return pq.top();
    }
};
```



2nd:

```
class Solution {
public:
    int findCheapestPrice(int n, vector<vector<int>>& flights, int src, int dst, int k) {
        vector<vector<pair<int,int>>> adj(n,vector<pair<int,int>>{});
        for(auto x:flights){
            adj[x[0]].push_back({x[1],x[2]});
        }

        queue<pair<int,pair<int,int>>> q;
        vector<int> dist(n,1e9);
        dist[src]=0;
```

```
        q.push({0,{src,0}});

        while(!q.empty()){
            auto front=q.front();
            q.pop();
            int stops=front.first;
            int cost=front.second.second;
            int node=front.second.first;

            if(stops>k){continue;}

            for(auto it:adj[node]){
                if(cost+it.second < dist[it.first] && stops<=k){
                    dist[it.first]=cost+it.second;
                    q.push({stops+1,{it.first,dist[it.first]}});
                }
            }
        }
        if(dist[dst]!=1e9){return dist[dst];}
        return -1;
    }
};
```