



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

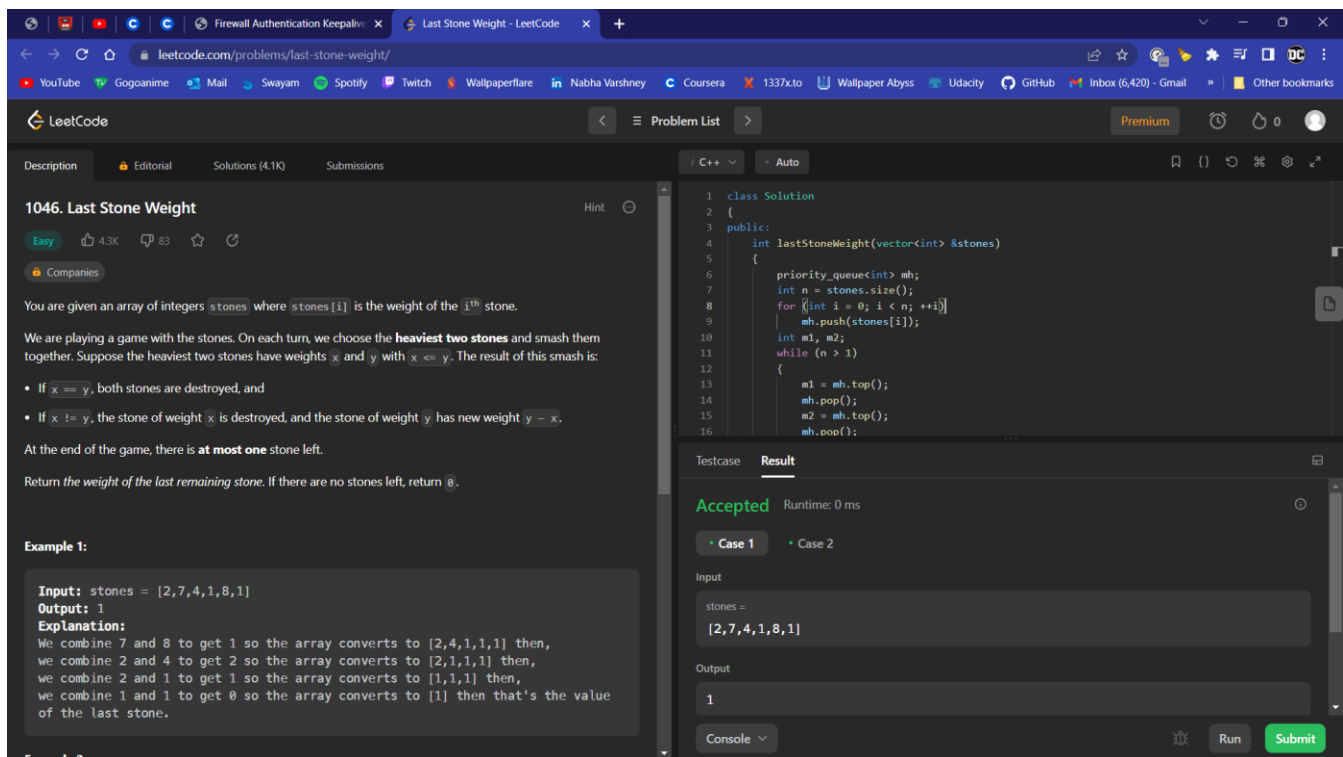
<https://leetcode.com/problems/last-stone-weight/>

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 -



1046. Last Stone Weight

Easy 4.3K 83

Companies

You are given an array of integers `stones` where `stones[i]` is the weight of the i^{th} 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 $x \leq y$. The result of this smash is:

- If $x == y$, both stones are destroyed, and
- If $x < y$, the stone of weight x is destroyed, and the stone of weight y has new weight $y - x$.

At the end of the game, there is **at most one** stone left.

Return the weight of the last remaining stone. If there are no stones left, return 0.

Example 1:

Input: `stones = [2,7,4,1,8,1]`
Output: 1
Explanation:
 We combine 7 and 8 to get 1 so the array converts to `[2,4,1,1,1]` then,
 we combine 2 and 4 to get 2 so the array converts to `[2,1,1,1]` then,
 we combine 2 and 1 to get 1 so the array converts to `[1,1,1]` then,
 we combine 1 and 1 to get 0 so the array converts to `[1]` then that's the value of the last stone.

Example 2:

Input: `stones = [1]`
Output: 1

Example 3:

Input: `stones = []`
Output: 0

Example 4:

Input: `stones = [5,4,3,2,1]`
Output: 2

Example 5:

Input: `stones = [2,2,4]`
Output: 0

Example 6:

Input: `stones = [1,2]`
Output: 1

Example 7:

Input: `stones = [1,3]`
Output: 2

Example 8:

Input: `stones = [1,1,2]`
Output: 0

Example 9:

Input: `stones = [1,2,3]`
Output: 0

Example 10:

Input: `stones = [1,2,3,4]`
Output: 0

Example 11:

Input: `stones = [1,2,3,4,5]`
Output: 1

Example 12:

Input: `stones = [1,2,3,4,5,6]`
Output: 1

Example 13:

Input: `stones = [1,2,3,4,5,6,7]`
Output: 1

Example 14:

Input: `stones = [1,2,3,4,5,6,7,8]`
Output: 1

Example 15:

Input: `stones = [1,2,3,4,5,6,7,8,9]`
Output: 1

Example 16:

Input: `stones = [1,2,3,4,5,6,7,8,9,10]`
Output: 1

Example 17:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11]`
Output: 1

Example 18:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12]`
Output: 1

Example 19:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13]`
Output: 1

Example 20:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14]`
Output: 1

Example 21:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15]`
Output: 1

Example 22:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16]`
Output: 1

Example 23:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17]`
Output: 1

Example 24:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18]`
Output: 1

Example 25:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]`
Output: 1

Example 26:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]`
Output: 1

Example 27:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21]`
Output: 1

Example 28:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22]`
Output: 1

Example 29:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23]`
Output: 1

Example 30:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24]`
Output: 1

Example 31:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]`
Output: 1

Example 32:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26]`
Output: 1

Example 33:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27]`
Output: 1

Example 34:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28]`
Output: 1

Example 35:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29]`
Output: 1

Example 36:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30]`
Output: 1

Example 37:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31]`
Output: 1

Example 38:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32]`
Output: 1

Example 39:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33]`
Output: 1

Example 40:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34]`
Output: 1

Example 41:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35]`
Output: 1

Example 42:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36]`
Output: 1

Example 43:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37]`
Output: 1

Example 44:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38]`
Output: 1

Example 45:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39]`
Output: 1

Example 46:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40]`
Output: 1

Example 47:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41]`
Output: 1

Example 48:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42]`
Output: 1

Example 49:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43]`
Output: 1

Example 50:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44]`
Output: 1

Example 51:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45]`
Output: 1

Example 52:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46]`
Output: 1

Example 53:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47]`
Output: 1

Example 54:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48]`
Output: 1

Example 55:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49]`
Output: 1

Example 56:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50]`
Output: 1

Example 57:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51]`
Output: 1

Example 58:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52]`
Output: 1

Example 59:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53]`
Output: 1

Example 60:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54]`
Output: 1

Example 61:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55]`
Output: 1

Example 62:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56]`
Output: 1

Example 63:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57]`
Output: 1

Example 64:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58]`
Output: 1

Example 65:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59]`
Output: 1

Example 66:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60]`
Output: 1

Example 67:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61]`
Output: 1

Example 68:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62]`
Output: 1

Example 69:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63]`
Output: 1

Example 70:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64]`
Output: 1

Example 71:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65]`
Output: 1

Example 72:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66]`
Output: 1

Example 73:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67]`
Output: 1

Example 74:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68]`
Output: 1

Example 75:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69]`
Output: 1

Example 76:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70]`
Output: 1

Example 77:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71]`
Output: 1

Example 78:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72]`
Output: 1

Example 79:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73]`
Output: 1

Example 80:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74]`
Output: 1

Example 81:

Input: `stones = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,`



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

2) Cheapest flights with K stops

<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];
    }
};
```



DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.
Output -

Screenshot of a web browser showing the LeetCode problem "787. Cheapest Flights Within K Stops".

Problem Description:

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`.

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:

```
graph TD
    0((0)) -- 100 --> 1((1))
    0((0)) -- 100 --> 2((2))
    1((1)) -- 100 --> 2((2))
    1((1)) -- 600 --> 3((3))
    2((2)) -- 200 --> 3((3))
```

Code Solution (C++):

```
1 class Solution {
2 public:
3     int findCheapestPrice(int n, vector<vector<int>>> &flights, int src, int dst, int k) {
4         vector<pair<int,int>> adj[n];
5         for(auto it : flights){
6             adj[it[0]].push_back({it[1],it[2]});
7         }
8         queue<pair<int,pair<int,int>>> pq;
9         pq.push({0,{src,0}});
10        vector<int> dist(n,1e9);
11        dist[src] = 0;
12        while(!pq.empty()){
13            auto front = pq.front();
14            pq.pop();
15            int stops = front.first;
16            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