

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Experiment-1.3

Student Name: Kanishk Soni
Branch: BE-CSE
Semester: 6th
Subject Code: 20CSP-351

UID: 20BCS9398
Section/Group: 20BCS_DM-708B
Subject Name: Competitive Coding-II

AIM: To demonstrate the concept of Heap model

Problem1: Last stone weight

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

Program Code:

```
class Solution {
    public int lastStoneWeight(int[] stones) {
        PriorityQueue<Integer> maxHeap = new PriorityQueue<>
(Comparator.reverseOrder());
        for (int stone : stones) {
            maxHeap.add(stone);
        }
        while (maxHeap.size() != 1) {
            int y = maxHeap.remove();
            int x = maxHeap.remove();
            if (x == y) maxHeap.add(0);
            if (x != y) maxHeap.add(y - x);
        }
        return maxHeap.peek();
    }
}
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Output:

The screenshot shows a LeetCode submission interface. On the left, the 'Submissions' tab is active, displaying a green 'Accepted' status for the problem '1047. Remove All Adjacent Duplicates In String'. Below this, there are suggestions for other problems: '992. Subarrays with K Different Integers', '1847. Closest Room', and '1053. Previous Permutation With One Swap'. The submission was made using Java. On the right, a detailed view of the submission is shown for user 'Kanishk' on 'Apr 06, 2023 22:15'. It includes a performance chart with 'Runtime 1 ms' (Beats 98.10%) and 'Memory 40.4 MB' (Beats 19.63%). A 'Notes' section is present with a placeholder 'Write your notes here'. Below the notes, there are 'Related Tags' and a 'Select tags' dropdown. The bottom part of the right panel shows the Java code for the solution, which uses a max heap to process the string.

Problem2: Cheapest flights with k stops

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

Program Code:

```
class Solution {  
    public int findCheapestPrice(int n, int[][] flights, int src, int dst, int k) {  
        int [] prev = new int[n];  
        Arrays.fill(prev,Integer.MAX_VALUE);  
        prev[src]=0;  
        for(int i=0;i<=k;i++){  
            int cur[]= new int[n];  
            for(int j=0;j<n;j++){
```

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

```
        cur[j]= prev[j];
    }

    for(int e[]:flights){
        int u=e[0],v=e[1],wt=e[2];
        if(prev[u] != Integer.MAX_VALUE && prev[u]+wt <cur[v]){
            cur[v] = prev[u]+wt;
        }
    }

    for(int j=0;j<n;j++){
        prev[j]=cur[j];
    }
}

return prev[dst]==Integer.MAX_VALUE ?-1:prev[dst];
}
}
```

Output:

The screenshot shows a LeetCode submission interface. On the left, the 'Submissions' tab is active, displaying 'Accepted' status and a 'Next question' button. Below this, there are links to '788. Rotated Digits', '568. Maximum Vacation Days', and '2093. Minimum Cost to Reach City With Discounts'. The submission details on the right include the user 'Kanishk', the problem name, and the solution code in Java. The code is a dynamic programming solution that uses two arrays, 'prev' and 'cur', to store the minimum cost to reach each city. It iterates through all flights and updates the 'cur' array based on the 'prev' array. The final result is returned as -1 if the destination is unreachable, or the minimum cost otherwise.

LeetCode

Problem List

Premium

Accepted

Next question

788. Rotated Digits

More challenges

568. Maximum Vacation Days

2093. Minimum Cost to Reach City With Discounts

All statuses

All languages

Accepted

a few seconds ago

Java

Kanishk

Apr 06, 2023 22:16

Details

+ Solution

Java

Runtime 4 ms

Beats 98.10%

Memory 43.9 MB

Beats 27.10%

Click the distribution chart to view more details

Notes

Write your notes here

Related Tags

Select tags

0/5

```
class Solution {
    public int findCheapestPrice(int n, int[][] flights, int src, int dst, int k) {
        int [] prev = new int[n];
        Arrays.fill(prev,Integer.MAX_VALUE);
        prev[src]=0;
        for(int i=0;i<=k;i++){
            int cur[] = new int[n];
            for(int j=0;j<n;j++){
                cur[j]= prev[j];
            }
            for(int e[]:flights){
                int u=e[0],v=e[1],wt=e[2];
                if(prev[u] != Integer.MAX_VALUE && prev[u]+wt <cur[v]){
                    cur[v] = prev[u]+wt;
                }
            }
            prev=cur;
        }
        return prev[dst]==Integer.MAX_VALUE ?-1:prev[dst];
    }
}
```