

Experiment No. - 3

Student Name: Vivek Kumar

Branch: BE-CSE(LEET)

Semester: 6th

Subject Name: Competitive coding - II

UID: 21BCS8129

Section/Group: 20BCS-ST-801/B

Date of Performance: 28/02/2023

Subject Code: 20CSP-351

1. Aim/Overview of the practical:

Q.1 Kth Largest Element in a Stream.

<https://leetcode.com/problems/kth-largest-element-in-a-stream/>

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Queue
- To implement the concept of Heap.

4. Code:

```
class KthLargest {
private:
    int k;
    std::priority_queue<int, std::vector<int>, std::greater<int>> pq;
public:
    KthLargest(int k, std::vector<int>& nums) {
        this->k = k;
        for (int num : nums) {
            pq.push(num);
            if (pq.size() > k) {
                pq.pop();
            }
        }
    }

    int add(int val) {
        pq.push(val);
        if (pq.size() > k) {
            pq.pop();
        }
        return pq.top();
    }
};
```

SuccessDetails

Runtime: 38 ms. faster than 56.95% of C++ online submissions for Kth Largest Element in a Stream.

Memory Usage: 19.9 MB. less than 83.87% of C++ online submissions for Kth Largest Element in a Stream.

Next challenges:

Kth Largest Element in an Array

Finding MK Average

Sequentially Ordinal Rank Tracker

Show off your acceptance:

Time Submitted

Status

Runtime

Memory

Language

02/28/2023 10:48

Accepted

38 ms

19.9 MB

cpp

C++

Autocomplete

```
1 class KthLargest {
2 private:
3     int k;
4     std::priority_queue<int, std::vector<int>, std::greater<int>> pq;
5 public:
6     KthLargest(int k, std::vector<int>& nums) {
7         this->k = k;
8         for (int num : nums) {
9             pq.push(num);
10            if (pq.size() > k) {
11                pq.pop();
12            }
13        }
14    }
15
16    int add(int val) {
17        pq.push(val);
18        if (pq.size() > k) {
19            pq.pop();
20        }
21        return pq.top();
22    }
23 };
24
25 /**
26  * Your KthLargest object will be instantiated and called as such:
27  * KthLargest* obj = new KthLargest(k, nums);
28  * int param_1 = obj->add(val);
29  */
```

Testcase

Run Code Result

Debugger

Accepted

Runtime: 0 ms

Your input

[{"kthLargest", "add", "add", "add", "add", "add"}]

[{"[3, [4, 5, 8, 2]], [3], [5], [10], [9], [4]}]

Output

[null, 4, 5, 5, 8, 8]

Diff

Expected

[null, 4, 5, 5, 8, 8]

Console

Use Example Testcases

Run Code

Submit

Problems

Pick One

Prev

703/2577

Next

1. Aim/Overview of the practical:

Q.2 Cheapest Flights Within K Stops

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

2. Apparatus / Simulator Used:

- Windows 7 or above
- Google Chrome

3. Objective:

- To understand the concept of Vector.
- To implement the concept of Vector iteration.

4. Code:

```
class Solution {
public:
    int findCheapestPrice(int n, vector<vector<int>>> &flights, int src, int dst, int k) {
        vector<vector<pair<int, int>>> adj(n);
        for (auto& e : flights) {
            adj[e[0]].push_back({e[1], e[2]});
        }
        vector<int> dist(n, numeric_limits<int>::max());
        queue<pair<int, int>> q;
        q.push({src, 0});
        int stops = 0;

        while (stops <= k && !q.empty()) {
            int sz = q.size();
            while (sz-- > 0) {
                auto [node, distance] = q.front();
                q.pop();
                for (auto& [neighbour, price] : adj[node]) {
                    if (price + distance >= dist[neighbour]) continue;
                    dist[neighbour] = price + distance;
                    q.push({neighbour, dist[neighbour]});
                }
            }
            stops++;
        }
        return dist[dst] == numeric_limits<int>::max() ? -1 : dist[dst];
    }
};
```

5. Result/Output/Writing Summary:

Cheapest Flights Within K Stops

[Explore](#)
[Problems](#)
[Interview](#)
[Contest](#)
[Discuss](#)
[Store](#)

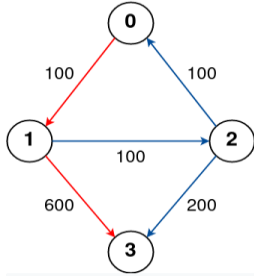
787. Cheapest Flights Within K Stops

Medium 7675 335 Add to List Share

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:



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.

```

1  class Solution {
2  public:
3      int findCheapestPrice(int n, vector<vector<int>>> &flights, int src, int dst, int k) {
4          vector<vector<pair<int, int>>> adj(n);
5          for (auto& e : flights) {
6              adj[e[0]].push_back({e[1], e[2]});
7          }
8          vector<int> dist(n, numeric_limits<int>::max());
9          queue<pair<int, int>> q;
10         q.push({src, 0});
11         int stops = 0;
12
13         while (stops <= k && !q.empty()) {
14             int sz = q.size();
15             while (sz-- > 0) {
16                 auto [node, distance] = q.front();
17                 q.pop();
18                 for (auto& [neighbour, price] : adj[node]) {
19                     if (price + distance >= dist[neighbour]) continue;
20                     dist[neighbour] = price + distance;
21                     q.push({neighbour, dist[neighbour]});
22                 }
23             }
24             stops++;
25         }
26         return dist[dst] == numeric_limits<int>::max() ? -1 : dist[dst];
27     }
28 };

```

Testcase

Run Code Result

Accepted

Runtime: 0 ms

Your input

4

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

Output

700

Diff

Expected

700

Run Code

Submit

Cheapest Flights Within K Stops

[Explore](#)
[Problems](#)
[Interview](#)
[Contest](#)
[Discuss](#)
[Store](#)

Success Details

Runtime: 24 ms, faster than 83.83% of C++ online submissions for Cheapest Flights Within K Stops.

Memory Usage: 13.1 MB, less than 86.85% of C++ online submissions for Cheapest Flights Within K Stops.

Next challenges:

Maximum Vacation Days Minimum Cost to Reach City With Discounts

Show off your acceptance:

Time Submitted	Status	Runtime	Memory	Language
03/07/2023 10:33	Accepted	24 ms	13.1 MB	cpp

```

1  class Solution {
2  public:
3      int findCheapestPrice(int n, vector<vector<int>>> &flights, int src, int dst, int k) {
4          vector<vector<pair<int, int>>> adj(n);
5          for (auto& e : flights) {
6              adj[e[0]].push_back({e[1], e[2]});
7          }
8          vector<int> dist(n, numeric_limits<int>::max());
9          queue<pair<int, int>> q;
10         q.push({src, 0});
11         int stops = 0;
12
13         while (stops <= k && !q.empty()) {
14             int sz = q.size();
15             while (sz-- > 0) {
16                 auto [node, distance] = q.front();
17                 q.pop();
18                 for (auto& [neighbour, price] : adj[node]) {
19                     if (price + distance >= dist[neighbour]) continue;
20                     dist[neighbour] = price + distance;
21                     q.push({neighbour, dist[neighbour]});
22                 }
23             }
24             stops++;
25         }
26         return dist[dst] == numeric_limits<int>::max() ? -1 : dist[dst];
27     }
28 };

```

Testcase

Run Code Result

Accepted

Runtime: 0 ms

Your input

4

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

Output

700

Diff

Expected

700

Run Code

Submit

Learning outcomes (What I have learnt):

- Learned the concept of cheapest flights within k stops.
- Learnt about Array in Vector and Its iteration.

Submitted By: Vivek Kumar