Day - 19 12-09-2021

leet - medium

Q1 Kth smallest element in a sorted matrix.

Input: matrix = [[1,5,9],[10,11,13],[12,13,15]], k = 8
output: 13

Brute force

// traverse the matrix and store it in a vector mat. and print kth small. element.

// optimal by max-heap.

```
int KthSmallest(vector<vector<int>>& matrix, int k) {
priority_queue<int> pq;
for(int i=0; i<matrix.size(); i++)
  {
    for(int j=0; j<matrix.size(); j++)
      {
        pq.push(matrix[i][j]);
        if(pq.size()>k)
        {
          pq.pop();
        }
      }
  }
return pq.top();
}
```

```
//by binary search
    int n = matrix.size();
    int low = matrix[0][0];
    int high = matrix[n-1][n-1];
    while (low < high) {
    int mid = low + (high-low)/2;
    int cnt = 0;
    for (int i=0; i<n; i++) {
        cnt += upper_bound(matrix[i].begin(), matrix[i].end(),
            mid) - matrix[i].begin();
    }
        if (cnt < k)
            {
                low = mid+1;
            }
        else {
                high = mid};
    }
    return low;
}
```

Leet Medium

Q2 Eliminate maximum number of monsters.

Input: dist = [1,3,4], speed = [1,1,1]
Output: 3

// find time = d/s

// check arrival and actual time.

```
int eliminateMaximum(vector<int>& dist, vector<int>& speed)
{
    int n = dist.size();
    vector<long double> time;
    for(int i=0; i<n; i++) {                    ⟶ (n)

        long double d = dist[i], s = speed[i];
        long double t = d/s;
        time.push_back(t);
}
    sort(time.begin(), time.end());             ⟶ nlogn
    int k=0, ans=0;
    for(int i=0; i<n; i++)                      ⟶ n
    {
        if(time[i]>k)
            ans++;
        else
            break;
    }
    return ans;
}
```

doubt in TC:
O(nlogn + 2n)