Kth Smallest With Only 2, 3 As Factors (Medium)

Find the Kth smallest number s such that s = 2 ^ x * 3 ^ y, x >= 0 and y >= 0, x and y are all integers.

Assumptions

- K >= 1

Examples

- the smallest is 1
- the 2nd smallest is 2
- the 3rd smallest is 3
- the 5th smallest is 2 * 3 = 6
- the 6th smallest is 2 ^ 3 * 3 ^ 0 = 8

```
int kthSmallest23(int k){
    priority_queue<int, vector<int>, greater<int> > pq; // min heap
    unordered_map<int, int> umap;
    int minVal = -1;
    pq.push(1);
    for(int i = 0; i < k; i++){
        minVal = pq.top();
        pq.pop();
        int val1 = minVal * 2;
        int val2 = minVal * 3;

        if(umap.find(val1) == umap.end()){
            pq.push(val1);
            umap[val1] = 1;
        }

        if(umap.find(val2) == umap.end()){
            pq.push(val2);
            umap[val2] = 1;
        }
    }
    return minVal;
}
```

Time Complexity: O(1)

正确！Accepted！

时间复杂度分析不正确。STL 中 priority_queue 的 pop()和 push()操作时间复杂度均为 O(log(S))，S 为容器的大小。本算法复杂度上限 O(Klog(K))。

Super Ugly Number (Medium)

Write a program to find the nth super ugly number.

Super ugly numbers are positive numbers whose all prime factors are in the given prime list `primes` of size k. For example, `[1, 2, 4, 7, 8, 13, 14, 16, 19, 26, 28, 32]` is the sequence of the first 12 super ugly numbers given `primes = [2, 7, 13, 19]` of size 4.

Note:
(1) 1 is a super ugly number for any given `primes`.
(2) The given numbers in `primes` are in ascending order.
(3) $0 < k \leq 100$, $0 < n \leq 10^6$, $0 < primes[i] < 1000$.
(4) The nth super ugly number is guaranteed to fit in a 32-bit signed integer.

```
vector<int> superUglyNum(vector<int> primes, int n){
    priority_queue<int, vector<int>, greater<int> > pq; // min heap
    unordered_map<int, int> umap;
    vector<int> res;
    int minVal = -1;
    pq.push(1);
    for(int i = 0; i < n; i++){
        minVal = pq.top();
        res.push_back(minVal);
        pq.pop();
        vector<int> temp;
        for(int j = 0; j < primes.size(); j++){
            int val = minVal * primes[j];
            if(umap.find(val) == umap.end()){
                pq.push(val);
                umap[val] = 1;
            }
        }
    }
    return res;
}
```

Time Complexity: O(1)

正确！Accepted！

同样是复杂度分析的问题。上限 O(nlog(nK))

Kth Closest Point To <0,0,0> (Medium)

Given three arrays sorted in ascending order. Pull one number from each array to form a coordinate <x,y,z> in a 3D space. Find the coordinates of the points that is k-th closest to <0,0,0>.
We are using euclidean distance here.

Assumptions

- The three given arrays are not null or empty, containing only non-negative numbers
- K >= 1 and K <= a.length * b.length * c.length

Return

- a size 3 integer list, the first element should be from the first array, the second element should be from the second array and the third should be from the third array

Examples

- A = {1, 3, 5}, B = {2, 4}, C = {3, 6}
- The closest is <1, 2, 3>, distance is sqrt(1 + 4 + 9)
- The 2nd closest is <3, 2, 3>, distance is sqrt(9 + 4 + 9)

```
// define input as global parameter
vector<int> a = {1, 3, 5}, b = {2, 4}, c = {3, 6};

struct MyCmp{
    bool operator()(vector<int> loc1, vector<int> loc2) {
        int sum1 = pow(a[loc1[0]], 2) + pow(b[loc1[1]], 2) + pow(c[loc1[2]], 2);
        int sum2 = pow(a[loc2[0]], 2) + pow(b[loc2[1]], 2) + pow(c[loc2[2]], 2);
        return sum1 > sum2;
    }
};

vector<int> kthCloestPoint(vector<int> a, vector<int> b, vector<int> c, int k) {
    priority_queue< vector<int>, vector<vector<int> >, MyCmp> pq;
    map<vector<int>, int> Map;
    vector<int> loc(3, 0);
    pq.push(loc);
    for(int i = 0; i < k; i++){
        loc = pq.top();
        pq.pop();

        loc[0]++;
        if(loc[0] < a.size() && Map.find(loc) == Map.end()){
```

```
            pq.push(loc);
            Map[loc] = 1;
        }
        loc[0]--;

        loc[1]++;
        if(loc[1] < b.size() && Map.find(loc) == Map.end()){
            pq.push(loc);
            Map[loc] = 1;
        }
        loc[1]--;

        loc[2]++;
        if(loc[2] < c.size() && Map.find(loc) == Map.end()){
            pq.push(loc);
            Map[loc] = 1;
        }
        loc[2]--;
    }
    return { a[loc[0]], b[loc[1]], c[loc[2]]};
}
```

正确！Accepted！

参考代码：

```
// define input as global parameter
#include<bits/stdc++.h>
using namespace std;

vector<int> a = {1, 3, 5}, b = {2, 4}, c = {3, 6};

struct Node{
    int px, py, pz, i, j, k;
    bool operator<(const Node &r) const
    {
        return px * px + py * py + pz * pz > r.px * r.px + r.py * r.py + r.pz * r
.pz;
    }
    bool operator>(const Node &r) const
    {
        return px * px + py * py + pz * pz < r.px * r.px + r.py * r.py + r.pz * r
.pz;
    }
};
vector<int> kthCloestPoint(vector<int> a, vector<int> b, vector<int> c, int k) {
    priority_queue<Node> pq;
    pq.push({a[0], b[0], c[0], 0, 0, 0});
    Node x;
```

```
    set<pair<int, pair<int,int> > > st;
    while(k--)
    {
        x = pq.top();
        pq.pop();
        if(x.i + 1 < a.size())
        {
            Node p = x;
            p.i += 1;
            p.px = a[p.i];
            if(st.find({p.i, {p.j, p.k}}) != st.end()) continue;
            pq.push(p);
            st.insert({p.i, {p.j, p.k}});
        }
        if(x.j + 1 < b.size())
        {
            Node p = x;
            p.j += 1;
            p.py = b[p.j];
            if(st.find({p.i, {p.j, p.k}}) != st.end()) continue;
            pq.push(p);
            st.insert({p.i, {p.j, p.k}});
        }
        if(x.k + 1 < c.size())
        {
            Node p = x;
            p.k += 1;
            p.pz = c[p.k];
            if(st.find({p.i, {p.j, p.k}}) != st.end()) continue;
            pq.push(p);
            st.insert({p.i, {p.j, p.k}});
        }
    }
    return {x.px, x.py, x.pz};
}
int main()
{
    auto t = kthCloestPoint(a, b, c, 2);
    cout << t[0] << ' ' << t[1] << ' ' << t[2];
}
```

时间复杂度 O(KlogK)

这题想参考一下老师的代码。

Kth Smallest Number In Sorted Matrix (Medium)

Given a matrix of size N x M. For each row the elements are sorted in ascending order, and for each column the elements are also sorted in ascending order. Find the Kth smallest number in it.

Assumptions

- the matrix is not null, N > 0 and M > 0
- K > 0 and K <= N * M

Examples

{ {1, 3, 5, 7},
{2, 4, 8, 9},
{3, 5, 11, 15},
{6, 8, 13, 18} }

- the 5th smallest number is 4
- the 8th smallest number is 6

```
// define input as global parameter
vector< vector<int> > matrix = {{1, 3, 5, 7},
                                {2, 4, 8, 9},
                                {3, 5, 11, 15},
                                {6, 8, 13, 18}};

struct MyCmp2 {
    bool operator()(pair<int, int> a, pair<int, int> b) {
        int val1 = matrix[a.first][a.second];
        int val2 = matrix[b.first][b.second];
        return val1 > val2;
    }
};

int kthSmallest(vector<vector<int>> matrix, int k) {
    priority_queue< pair<int, int>, vector<pair<int, int> >, MyCmp2> pq;
    map<pair<int, int>, int> Map;
    pair<int, int> pos(0, 0);
    pq.push(pos);
    for(int i = 0; i < k; i++){
        pos = pq.top();
        pq.pop();

        pos.first++;
        if(pos.first < matrix.size() && Map.find(pos) == Map.end()){
            pq.push(pos);
            Map[pos] = 1;
```

```
        }
        pos.first--;

        pos.second++;
        if(pos.second < matrix[0].size() && Map.find(pos) == Map.end()){
            pq.push(pos);
            Map[pos] = 1;
        }
        pos.second--;
    }
    return matrix[pos.first][pos.second];
}
```

Accepted!

也可以用大根堆实现，堆里维护 K 个元素。

问题与上体类似， 这题想参考一下老师的代码。

```
class Solution {
public:
    int kthSmallest(vector<vector<int>>& matrix, int k) {
        priority_queue<int> pq;
        int n = matrix.size(), m = matrix[0].size();
        for(int i = 0; i < n; i++)
        {
            if(pq.size() == k && pq.top() < matrix[i][0]) break;
            for(int j = 0; j < m; j++)
            {
                if(pq.size() < k)
                {
                    pq.push(matrix[i][j]);
                }
                else
                {
                    if(matrix[i][j] > pq.top()) break;
                    pq.push(matrix[i][j]);
                    if(pq.size() > k) pq.pop();
                }
            }
        }
        return pq.top();
    }
};
```

时间复杂度 O(n*m*logk)

1. 在用 BFS-2 求 kth…的题目中，如何计算时间复杂度？

2. 十二次课已经学完了基本的数据结构， 想问老师后面的还有多少课程，课程安排大概是什么样的？ 我想用一两周时间做一做 Leetcode 上的题目，总结总结前面学的内容。

3. 另外，下周开始我要开始实习了，之后进度想保持一周两次课。