

$$1. \quad n! = n(n-1) \cdots 2 \cdot 1 \leq \underbrace{n \cdot n \cdots n \cdot n}_{n \uparrow n} = n^n$$

$$\therefore n! \leq n^n$$

$$\log(n!) \leq n \log n$$

$$\log(n!) = O(n \log n)$$

$$\because n! = n(n-1) \cdots 2 \cdot 1 \geq n(n-1) \cdots (n/2)$$

$$\geq \underbrace{(n/2) \cdot (n/2) \cdots (n/2)}_{n/2 \uparrow n/2}$$

$$= (n/2)^{n/2}$$

$$\therefore n! \geq (n/2)^{n/2}$$

$$\log(n!) \geq \frac{n}{2} \log\left(\frac{n}{2}\right)$$

$$\log(n!) \geq n \log n$$

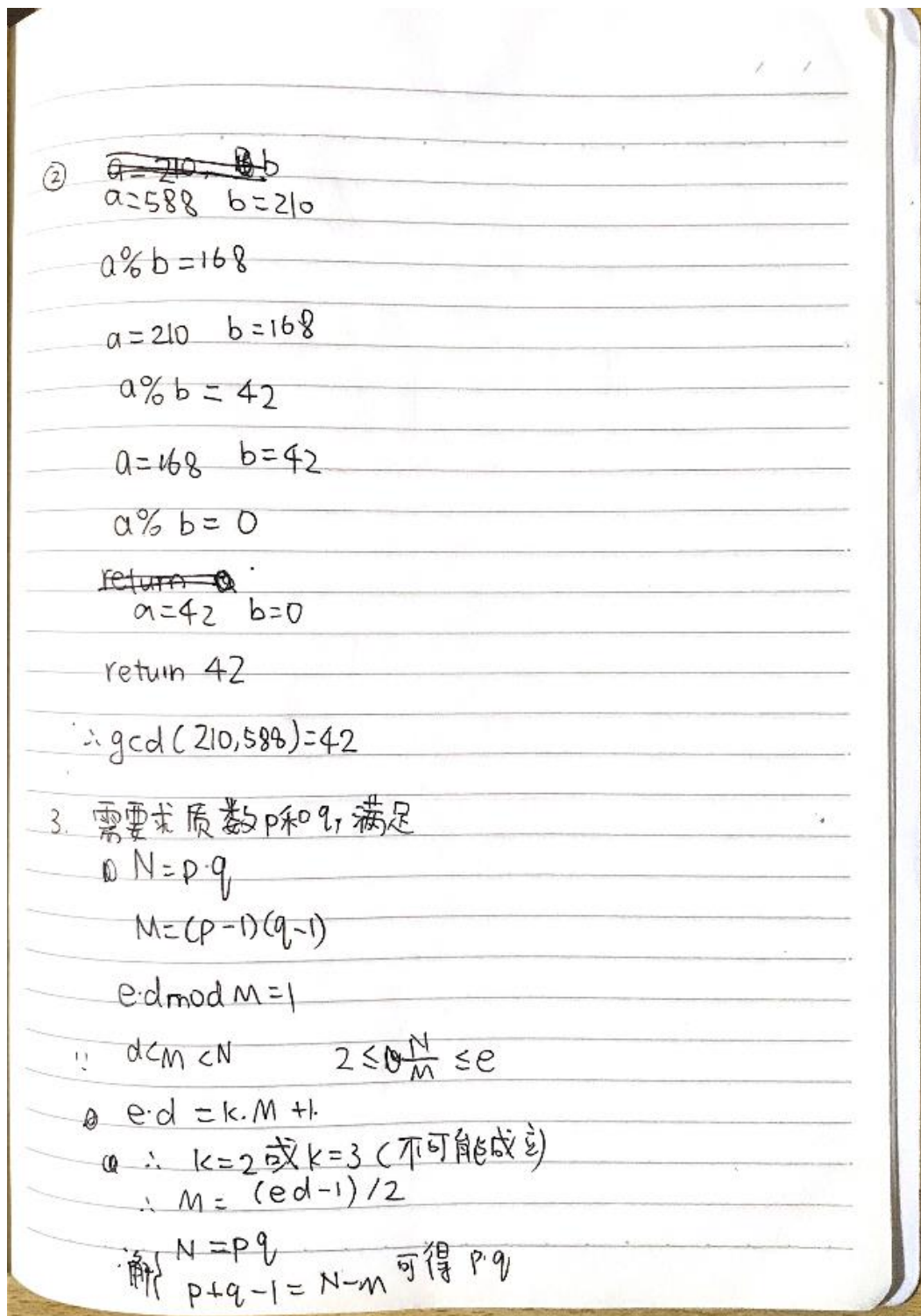
$$\therefore \log(n!) = \Omega(n \log n)$$

$$\therefore \log(n!) = \Theta(n \log n)$$

$$2. \quad (1) \quad 210 = 7 \times 5 \times 3 \times 2$$

$$588 = 7^2 \times 3 \times 2^2$$

$$\therefore \gcd(210, 588) = 7 \times 3 \times 2 = 42$$



1. 思路: 使用 c++ unordered_set 的暴力解法
 直接搜索两数之和是否在数组中

复杂度: $O(N^2 \log M)$

```
#include <unordered_set>
#include <vector>
```

```

using namespace std;

class Solution {
public:
    int lenLongestFibSubseq(vector<int>& A) {
        unordered_set<int> s(A.begin(), A.end());
        int r = 0;
        for (int i = 0; i < A.size(); i++) {
            for (int j = i + 1; j < A.size(); j++) {
                int a = A[j], b = A[i] + A[j];
                int l = 2;
                while (s.find(b) != s.end()) {
                    int c = b;
                    b += a;
                    a = c;
                    l++;
                }
                r = (l > r) ? l : r;
            }
        }
        return r >= 3 ? r : 0;
    }
};

```

Success Details >

Runtime: 188 ms, faster than 63.33% of C++ online submissions for Length of Longest Fibonacci Subsequence.

Memory Usage: 8.5 MB, less than 96.03% of C++ online submissions for Length of Longest Fibonacci Subsequence.

Next challenges: [Fibonacci Number](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
03/29/2021 23:41	Accepted	188 ms	8.5 MB	cpp

[Problems](#)
[Pick One](#)
[Prev](#) 873/1809 [Next](#)

[Run Code](#)
[Submit](#)

2. 思路：新建链表，对原链表每个元素插入新链表

复杂度： $O(N^2)$

```

class Solution {
public:

```

```

ListNode* insertionSortList(ListNode* head) {
    ListNode* list = new ListNode();
    ListNode* current = head;
    while( current!= nullptr ){
        ListNode* prev = list;
        while( prev->next!= nullptr && prev->next->val < current
->val ){
            prev = prev->next;
        }
        ListNode* next = current->next;
        current->next = prev->next;
        prev->next = current;
        current = next;
    }
    ListNode* l = list->next;
    delete list;
    return l;
}
};

```

Homework_1_1 x (4) Merge k Sorted Lists x (4) Insertion Sort List x Homework_1_1 x (4) Length of Linked List x (4) Length of Linked List x + -

leetcode.com/problems/insertion-sort-list/submissions/

力扣「中文社区」现已上线，全新「个人主页」更优体验，即刻加入先人一步攒积分！
新功能推荐：全新积分商城礼品 | 上万社区题解 | 企业题库 | 面试模拟 | 更多竞赛 | 轻松数据同步使用已有积分

Description Solution Discuss Submissions

Success Details >

Runtime: 44 ms, faster than 67.64% of C++ online submissions for Insertion Sort List.

Memory Usage: 9.6 MB, less than 39.34% of C++ online submissions for Insertion Sort List.

Next challenges: [Sort List](#) [Insert into a Sorted Circular Linked List](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory
03/29/2021 00:36	Accepted	44 ms	9.6 MB

```

1  // Definition of singly-linked list:
2  struct ListNode {
3      int val;
4      ListNode *next;
5      ListNode() : val(0), next(nullptr) {}
6      ListNode(int x) : val(x), next(nullptr) {}
7      ListNode(int x, ListNode *next) : val(x), next(next) {}
8  };
9
10 // Solution
11 class Solution {
12 public:
13     ListNode* insertionSortList(ListNode* head) {
14         ListNode* list = new ListNode();
15         ListNode* current = head;
16         while( current!= nullptr ){
17             ListNode* prev = list;
18             while( prev->next!= nullptr && prev->next->val < current->val ){
19                 prev = prev->next;
20             }
21             ListNode* next = current->next;
22             current->next = prev->next;
23             prev->next = current;
24             current = next;
25         }
26         return list->next;
27     }
28 };

```

Testcase Run Code Result Debugger

Accepted Runtime: 4 ms

Your input [4,2,1,3]

Output [1,2,3,4] [Diff](#)

Expected [1,2,3,4]

Console Use Example Testcases Run Code Submit

3. 思路：遍历各个链表，找到最小的元素，加在返回的链表后面

复杂度： $O(kN)$

```

class Solution {
public:
    ListNode* mergeKLists(vector<ListNode*> lists) {
        if (lists.size() == 0) {
            return nullptr;
        }
    }
};

```

```

    }
    vector<ListNode*> _lists = lists;
    ListNode* current = new ListNode;
    ListNode* head = current;
    while (true) {
        int m, j = 0;
        while (j < _lists.size() && _lists[j] == nullptr) {
            j++;
        }
        if (j == _lists.size()) {
            break;
        }
        m = j;
        for (int i = j; i < _lists.size(); i++) {
            if (_lists[i] != nullptr && _lists[i]->val < _lists[
m]->val) {

                m = i;
            }
        }
        ListNode* next = new ListNode(_lists[m]->val);
        current->next = next;
        current = current->next;
        _lists[m] = _lists[m]->next;
    }
    current = head->next;
    delete head;
    return current;
}
};

```

Success Details >

Runtime: 836 ms, faster than 5.01% of C++ online solutions for Merge k Sorted Lists.

Memory Usage: 13.9 MB, less than 26.44% of C++ online solutions for Merge k Sorted Lists.

Next challenges: [Merge Two Sorted Lists](#) [Ugly Number II](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory
03/29/2021 23:46	Accepted	836 ms	13.9 MB

Your previous code was restored from your local storage. [Reset to default](#)

Console [Contribute](#) [Run Code](#) [Submit](#)