

1. 解: 由斯特林公式, 得

$$\lim_{n \rightarrow \infty} \frac{n!}{\sqrt{2\pi n} \left(\frac{n}{e}\right)^n} = 1$$

即分子、分母为等价无穷大

$$\begin{aligned} \therefore \lim_{n \rightarrow \infty} \frac{\log(n!)}{\log(n^n)} &= \lim_{n \rightarrow \infty} \frac{\log(\sqrt{2\pi n} \left(\frac{n}{e}\right)^n)}{\log(n^n)} = \lim_{n \rightarrow \infty} \frac{\frac{1}{2} \log 2\pi + \frac{1}{2} \log n + n \log n - n \log e}{n \log n} \\ &= \lim_{n \rightarrow \infty} \left(\frac{\frac{1}{2} \log 2\pi}{n \log n} + \frac{\frac{1}{2}}{n} + 1 - \frac{1}{\ln n} \right) \end{aligned}$$

$$\therefore \log(n!) = \Theta(\log \Theta(n \log n)) = 1$$

2. 解: ① 因式分解:

$$210 = 3 \times 7 \times 2 \times 5$$

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

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

② 欧几里得算法:

$$588 \div 210 = 2 \cdots 168$$

$$210 \div 168 = 1 \cdots 42$$

$$168 \div 42 = 4 \cdots 0$$

$$\therefore \gcd(210, 588) = 42$$

3. 解: 由已知, 得

$$\text{设 } N = p \cdot q \quad \varphi(N) = (p-1)(q-1) = M$$

$$\therefore d < \varphi(N) < N, \text{ 且有 } ed \bmod M \equiv 1 \pmod{M}$$

$$ed = kM + 1 \quad \therefore d < M \quad \therefore k \text{ 的可能取值为 } 1 \text{ 或 } 2$$

则有 $k=1$ 时, 有

$$\text{① } M = ed - 1 = 3d - 1 = (p-1)(q-1)$$

$$\text{② } N = p \cdot q$$

$$\text{联立①②, 得 } \begin{cases} p = \frac{2N}{N+2-3d+\sqrt{3d-2-N}^2-4N} \\ q = \frac{2N}{N+2-3d+\sqrt{3d-2-N}^2-4N} \end{cases}$$

当 $k=2$ 时, 有

$$\text{③ } M = \frac{3d-1}{2} = (p-1)(q-1)$$

$$\text{④ } N = p \cdot q$$

$$\text{联立③④, 得 } \begin{cases} p = \frac{2N}{N-3(\frac{d-1}{2})+\sqrt{3(\frac{d-1}{2})-N}^2-4N} \\ q = \frac{2N}{N-3(\frac{d-1}{2})+\sqrt{3(\frac{d-1}{2})-N}^2-4N} \end{cases}$$

4、

算法思路:

采用动态规划的思想, 设置一个二维数组 $dp[n][n]$ ($n=A.size()$), 其代表的意义是:

假设存在 $A[i], A[j]$ 。那么 $dp[i][j]$ 代表的就是以 $A[i], A[j]$ 为结尾的斐波那契数列

的长度, 例如: $A=\{1, 2, 3, 5, 8\}$, 那么 $dp[3][4]=5$ 。同时, 如果我们存在 $A[k]=A[j]-$

$A[i]$ 且 $k < i$ ，显然存在 $dp[i][j] = dp[k][i] + 1$ 。所以我们用两层循环遍历一下所有的 i 和 j 的组合情况，保存最大值即可。

复杂度分析：

时间复杂度： $O(n^2)$ ，空间复杂度： $O(n^2)$ ， n 为 A 的长度。

代码：

```
class Solution {
public:
    int lenLongestFibSubseq(vector<int>& A) {
        unordered_map<int,int> hash;
        int n=A.size();
        if(n==0)
        {
            return 0;
        }
        for(int i=0;i<n;i++)
        {
            hash[A[i]]=i;
        }
        vector<vector<int>> dp(n,vector<int>(n,0));
        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                dp[i][j]=2;
            }
        }
        int temp=0;
        int answer=0;
        for(int i=0;i<n;i++)
        {
            for(int j=i+1;j<n;j++)
            {
                temp=A[j]-A[i];
                if(hash.count(temp))
                {
                    int temp1=hash[temp];
                    if(temp1<i)
                    {
                        dp[i][j]=dp[temp1][i]+1;
                    }
                }
            }
        }
        return answer;
    }
};
```

```

        answer=max(answer,dp[i][j]);
    }
}
}
return answer;
}
};

```

截图：

Execution Result: **通过** (Passed)

Runtime: 400 ms, beat 48.11% of C++ submissions

Memory: 61.7 MB, beat 33.33% of C++ submissions

提交时间	提交结果	执行用时	内存消耗
几秒前	通过	400 ms	61.7 MB
1 分钟前	解答错误	N/A	N/A

```

1 class Solution {
2 public:
3     int lenLongestFibSubseq(vector<int>& A) {
4         unordered_map<int,int> hash;
5         int n=A.size();
6         if(n==0)
7             return 0;
8         for(int i=0;i<n;i++)
9             hash[A[i]]=i;
10        vector<vector<int>> dp(n,vector<int>(n,0));
11        for(int i=0;i<n;i++)
12        {
13            for(int j=i+1;j<n;j++)
14            {
15                dp[i][j]=2;
16            }
17        }
18        int temp=0;
19        int answer=0;
20        for(int i=0;i<n;i++)
21        {
22            for(int j=i+1;j<n;j++)
23            {
24                temp=A[j]-A[i];

```

5、

算法思路：

解决本题有两种方法：第一种是比较投机取巧的，将链表中的元素放到数组中，然后对数组进行插入排序（或者其他排序），最后放回链表。不过本题旨在对链表进行插入排序的操作，故在此采用这种方法。

用 3 个指针：s, q, p。其中 s 是指向头指针的指针，q 是已经排好序的部分，p 是需要插入到合适位置的。我们对 p 进行操作，用一个 temp 指针从头遍历一下，找到 p 应该插入的位置，然后将 p 插入即可。最后返回 s->next 即可。

复杂度分析：

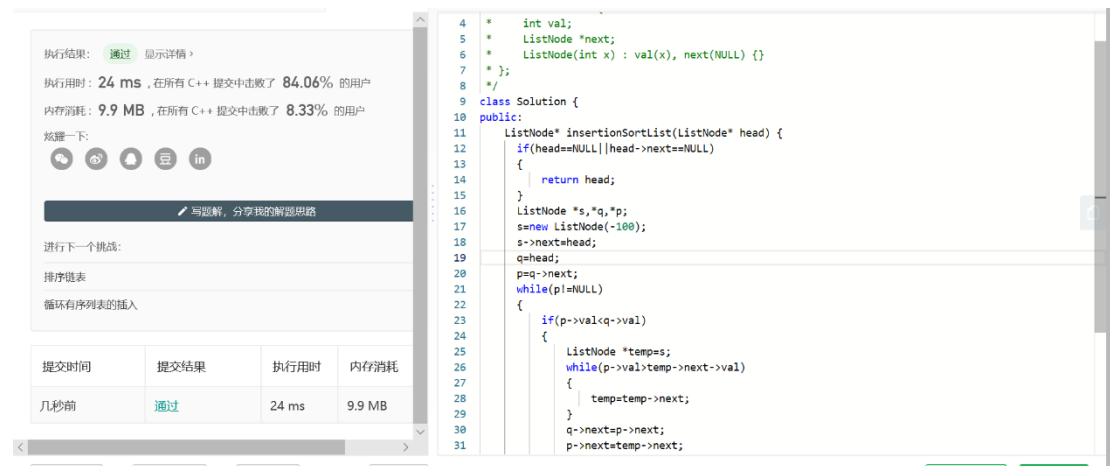
时间复杂度：O(nlogn)，空间复杂度：O(1)。n 为链表长度。

代码:

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
 *     ListNode(int x) : val(x), next(NULL) {}
 * };
 */
class Solution {
public:
    ListNode* insertionSortList(ListNode* head) {
        if(head==NULL || head->next==NULL)
        {
            return head;
        }
        ListNode *s,*q,*p;
        s=new ListNode(-100);
        s->next=head;
        q=head;
        p=q->next;
        while(p!=NULL)
        {
            if(p->val<q->val)
            {
                ListNode *temp=s;
                while(p->val>temp->next->val)
                {
                    temp=temp->next;
                }
                q->next=p->next;
                p->next=temp->next;
                temp->next=p;
                p=q->next;
            }
            else
            {
                q=q->next;
                p=p->next;
            }
        }
        return s->next;
    }
}
```

```
};
```

截图：



6、

算法思路：

写出两个链表合并的算法，然后将 lists 里面的链表逐个合并即可。关键在于两个链表如何合并。

从两个链表头开始，每次比较两个值的大小，然后选择较小者加入新链表，同时较小者所在的原链表指向下一个 next，直到有一个链表为空为止。最后将不为空的剩下的链表拼接到新链表的末尾即可。

复杂度分析：

时间复杂度： $O(k^2n)$ ，空间复杂度： $O(1)$ 。k 为 lists.size()-1，n 为 lists 里最长的链表的长度。

代码：

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     ListNode *next;
```

```

*     ListNode(int x) : val(x), next(NULL) {}
* };
*/
class Solution {
public:
    ListNode *TwoLists(ListNode *a,ListNode *b)
    {
        ListNode *head=new ListNode(0);
        ListNode *p=head;
        ListNode *pa=a,*pb=b;
        while(pa!=NULL&&pb!=NULL)
        {
            if(pa->val<pb->val)
            {
                head->next=pa,pa=pa->next;
            }
            else
            {
                head->next=pb,pb=pb->next;
            }
            head=head->next;
        }
        head->next=pa?pa:pb;
        return p->next;
    }
    ListNode* mergeKLists(vector<ListNode*> lists) {
        if(lists.size()==0)
        {
            return NULL;
        }
        if(lists.size()==1)
        {
            return lists[0];
        }
        int len=lists.size();
        ListNode *p=TwoLists(lists[0],lists[1]);
        for(int i=2;i<len;i++)
        {
            p=TwoLists(p,lists[i]);
        }
        return p;
    }
};

```

截图：

执行结果： 通过 显示详情 >

执行用时：180 ms，在所有 C++ 提交中击败了 20.47% 的用户

内存消耗：10.4 MB，在所有 C++ 提交中击败了 100.00% 的用户

炫耀一下：

写题解，分享我的解题思路

进行下一个挑战：

合并两个有序链表

丑数 II

提交时间	提交结果	执行用时	内存消耗
几秒前	通过	180 ms	10.4 MB
1 分钟前	执行出错	N/A	N/A

```
9 //
10 class Solution {
11 public:
12     ListNode *TwoLists(ListNode *a, ListNode *b)
13     {
14         ListNode *head=new ListNode(0);
15         ListNode *p=head;
16         while(p!=NULL&&q!=NULL)
17         {
18             if(p->val<q->val)
19             {
20                 head->next=p,p=p->next;
21             }
22             else
23             {
24                 head->next=q,q=q->next;
25             }
26             head=head->next;
27         }
28         head->next=p!=NULL?p:q;
29         return head;
30     }
31     ListNode* mergeKLists(vector<ListNode*> lists) {
32         if(lists.size()==0)
33         {
34             return NULL;
35         }
36         if(lists.size()==1)
```