事先吐槽出题人！！题目难度根本不是递增的！A题最难不接受反驳

A无序组数

如果不算无序的话，很好做。

线性筛一发之后枚举每个数的因数有多少，

之后两边乘起来就是答案。

现在我们考虑无序。

当且仅当两个数同时是m,n的约数是会有影响，

所以在统计一下这两个数同时有的约数就好了……

考后ac代码

```
#include<bits/stdc++.h>
using namespace std;
int d[100010];
int T;
int main(){
    for(int i=1;i<100010;i++)
        for(int j=i;j<100010;j+=i)
            d[j]++;
    scanf("%d",&T);
    for(;T--;){
        int a,b;
        scanf("%d%d",&a,&b);
        printf("%d\n",d[a]*d[b]-d[__gcd(a,b)]*(d[__gcd(a,b)]-1)/2);
    }
    return 0;
}
```

复杂度$O(n+t)$

B路径数量

裸的$Folyed$传递闭包，

矩阵快速幂一下就好。

其实直接矩阵乘就能过

复杂度$O(n^3k)$或者$O(n^3logk)$

不知道的可以自行出门左转上度娘

```
//#include"suqingnian.h"

#include<iostream>
```

```cpp
#include<cstdio>
#include<algorithm>
#include<ctime>
#include<cstring>
#define int long long
using namespace std;
int n,k;
struct _Martix{
    int
      num[210][210];
    _Martix(){memset(num,0,sizeof(num));}
    friend _Martix operator*
      ( const _Martix &__a,const _Martix &__b)
        {
            _Martix __c;
            int _n=n;
            for( int _k = 1 ; _k <= _n ; _k ++ )
                for( int _i = 1 ; _i <= _n ; _i ++ )
                    for( int _j = 1; _j <= _n ; _j ++ )
                        __c.num[_i][_j] +=
                            __a.num[_i][_k] * __b.num[_k][_j] ;
            return __c;
        }
}q;

_Martix pow2(_Martix a,long long b)
{
    _Martix c;
    for(int i=1;i<=n;i++) c.num[i][i]=1;
    for(;b;b>>=1,a=a*a) if(b&1) c=c*a;
    return c;
}
signed main()
{
    scanf("%lld%lld",&n,&k);
    for(int i=1;i<=n;i++)
      for(int j=1;j<=n;j++)
        scanf("%lld",&q.num[i][j]);

    q=pow2(q,k);
    printf("%lld",q.num[1][n]);
    return 0;
}
```

C数列下标

看看这数据范围

n^2暴力可过

模拟一下就好

贴代码：

```
#pragma gcc optimize(2)
#include<bits/stdc++.h>
using namespace std;
int a[100010],n;
int main(){
    scanf("%d",&n);
    for(int i=1;i<=n;i++) scanf("%d",&a[i]);
    for(int i=1;i<=n;i++){
        int ans=0;
        for(int j=i+1;j<=n;j++)
            if(a[j]>a[i]){
                ans=j;
                break;
            }
        printf("%d ",ans);
    }
    return 0;
}
```

D星光晚餐

原题大作战$*1$，出门左拐洛谷开灯。

结论是，直接开根号下取整输出就好。

$O(1)$

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cmath>
#include<cstring>
using namespace std;
long long n;
int main()
{
    scanf("%lld",&n);
    long long ans=sqrt(n);
    printf("%lld",ans);
    return 0;
}
```

E括号序列

原题大作战$*2$，出门直走百度题面(但是窝忘了哪那道题了)

我们贪心的想每个右括号要不要换，尽量把左括号放到左边，右括号放到右边

直接说解法。用一个栈维护序列，如果是右括号，直接压栈

之后统计一发答案就行了

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<cstring>
#include<stack>
using namespace std;
char a[5000050];
stack<char> st;
int n;int cnt;
int main()
{
    scanf("%d",&n);
    cin>>a;
    for(int i=0;i<n;i++)
    if(a[i]=='(') st.push('(');
    else{
        if(st.empty()||st.top()==')') st.push(')');
        else st.pop(),cnt++;
    }
    printf("%d",(n/2-cnt+1)/2);
    return 0;
}
```

复杂度$O(n)$

F[假的数学游戏]

打表好题

先说一下第i组数据的输入就是i个7

普通的$java$，高精打表就不说了

这里讲个比较块的方法

首先我们想暴力

枚举$n \in [1,+\infty)$之后判断与x^x大小

然后，发现太大，所以日常套路一下。

两边取自然对数。

左边就变成

$\sum_{i=1}^{n} ln(i)$

右边变成

$xln(x)$

这样开$long\ double$就可以不用考虑精度问题了，复杂度O答案

然后常识告诉我们，第i组的答案的位数是i

然后就打一发表就行，然后最后一个点窝打了$5min$前面是秒出……

```
#include<iostream>
#include<cstdio>
#include<algorithm>
#include<ctime>
#include<cmath>
#include<cstring>
using namespace std;

long long n;
int main()
{
    scanf("%lld",&n);
    if(n==7ll) puts("10");
    if(n==77ll) puts("94");
    if(n==777ll) puts("892");
    if(n==7777ll) puts("8640");
    if(n==77777ll) puts("84657");
    if(n==777777ll) puts("834966");
    if(n==7777777ll) puts("8267019");
    if(n==77777777ll) puts("82052137");
    if(n==777777777ll) puts("815725636");
    if(n==7777777777ll) puts("8118965902");
    return 0;
}
```

这份题解会等出成绩之后做相应调整……

并且会贴上代码

如果窝爆零了就会删除

以上是窝考场上想出来的

ABC都有锅