高精度算法

2021年1月14日 12:17

高精度加法：

```
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
	string num1,num2;
	int a[10500] = { 0 }, b[10500] = { 0 }, c[10500] = { 0 };
	cin >> num1 >> num2;
	int m = num1.length();
	int n = num2.length();

	for (int i = 0; i < m; i++)
		a[m - i - 1] = num1[i]-'0';//逆序存储数字
	for (int i = 0; i < n; i++)
		b[n - i - 1] = num2[i]-'0';

	m = max(m, n);//取较长的位数作为运算次数
	m++;//防止结果溢位

	for (int i = 0; i < m; i++)
	{
		int v = a[i] + b[i];
		if (c[i] + v < 10)//如果当前位无进位
			c[i] += v;
		else//若有进位
		{
			c[i + 1] += (c[i] + v) / 10;
			c[i] = (c[i] + v) % 10;
		}
	}

	if (c[m - 1] == 0)m--;//若有前导0
	for (int i = m - 1; i >= 0; i--)
		cout << c[i];


}
```

高精度减法：

```
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
	string num1,num2;
	int a[10500] = { 0 }, b[10500] = { 0 }, c[10500] = { 0 };
	int flag = 1;
	cin >> num1 >> num2;
	if (num1 < num2 && num1.length() == num2.length() || num1.length() < num2.length())
	{
		swap(num1, num2);
		flag = -1;//若第一个数小于第二个数，说明结果为负
	}
	int m = num1.length();
	int n = num2.length();

	for (int i = 0; i < m; i++)
		a[m - i - 1] = num1[i]-'0';//逆序存储数字
	for (int i = 0; i < n; i++)
		b[n - i - 1] = num2[i]-'0';

	m = max(m, n);//取较长的位数作为运算次数，这里不需要递增m防溢位，因为两数相减
	位数不可能增多

	for (int i = 0; i < m; i++)
	{
		if (a[i] < b[i])
		{
			a[i + 1]--;
			a[i] += 10;
		}//借位
		c[i] = a[i] - b[i];
	}

	while (c[m - 1] == 0)m--;//忽略前导0
	if (m < 1)cout << 0;//如果结果为0，直接输出0（不做这个判断的话会被当做前导0删光）
	(也可以直接在上面那个while里加上m>0)
	else
	{
		if (flag == -1)cout << '-';//若为负数则输出负号
		for (int i = m - 1; i >= 0; i--)
			cout << c[i];
	}


}
```

高精度乘法：

```
#include<iostream>
#include<algorithm>
using namespace std;
int main()
{
	string num1,num2;
	int a[10500] = { 0 }, b[10500] = { 0 }, c[10500] = { 0 };
	cin >> num1 >> num2;
	int m = num1.length();
	int n = num2.length();
	for (int i = 0; i < m; i++)
		a[m - i - 1] = num1[i]-'0';//逆序存储数字
	for (int i = 0; i < n; i++)
		b[n - i - 1] = num2[i]-'0';



	for(int i=0;i<m;i++)
		for (int j = 0; j < n; j++)
		{
			int v = a[i] * b[j];
			if (c[i + j] + v < 10)//注意两位相乘后对应到结果的位数，列个竖式可以验证这
			个关系
				c[i + j] += v;
			else
			{
				c[i + j + 1] += (c[i + j] + v) / 10;
				c[i + j] = (c[i + j] + v) % 10;
			}
		}
	int k = m + n - 1;
	while (k > 0 && c[k] == 0)k--;
	for (int i = k; i >= 0; i--)
		cout << c[i];

}
```

高精度乘低精度：

```
#include<iostream>
using namespace std;
void swapStr(char* s)
{
	int k = strlen(s);
	for (int i = 0; i < k / 2; i++)
	{
		swap(s[i], s[k - i - 1]);
	}
}
void mul(char* s, int t)
{

	int m = strlen(s);
	int cp = 0;//进位
	int tmp = 0;
	int i = 0;
	swapStr(s);
	for (i = 0; i < m; i++)
	{
		tmp = (s[i] - '0') * t + cp;
		s[i] = tmp % 10 + '0';
		cp = tmp / 10;
	}
	while (cp)//进位没用完
	{
		s[i++] = cp % 10 + '0';
		cp /= 10;
	}
	while (s[i - 1] == '0' && i > 1)i--;
	s[i] = '\0';//此前我们一直使用的是string，但这里说一下，如果用char*，一定要加这句
	swapStr(s);
}
int main()
{
	char a[1000];
	int t;
	cin >> a >> t;
	mul(a, t);
	for (int i = 0; i < strlen(a); i++)
	{
		cout << a[i];
	}
}
```

高精度除以低精度：

```
#include<bits/stdc++.h>
using namespace std;
using ll = long long;
string divid(string a,ll b)
{
   string ans;
	ll cp = 0;
	for (ll i = 0; i < a.length(); i++)
	{
		ll v = (cp * 10 + (a[i] - '0')) / b;
		ans += v + '0';
		cp= (cp * 10 + (a[i] - '0')) % b;
	}
	ll k=0;
	while(ans[k]=='0'&&k<ans.length()-1)
	{
	   k++;
	}
	ans=ans.substr(k);
	return ans;
}

int main()
{
	string a;
	ll b;
	cin >> a >> b;
	cout<<divid(a,b)<<endl;
}
```

高精度对低精度取模：

```
int mod(string a, int b)//高精度a除以单精度b
{
	int d=0;
```

```
    for(int i=0;i<a.size();i++)   d=(d*10+(a[i]-'0'))%b;   //求出余数
    return d;
}
```

来自 <https://blog.csdn.net/liangzhaoyang1/article/details/50927517>

brute force

2021年7月15日　　15:53

https://codeforces.com/problemset/problem/1550/C

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include <string>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 2e5 + 5;
const int mod = 10007;
const double eps = 1e-4;
ll n;
ll p[N];
int cmp(pll&a,pll&b)
{
    return a.first<b.first;
}
int isGood(ll l,ll r)
{
    for(ll i=l;i<=r;i++)
    {
        for(ll j=i+1;j<=r;j++)
        {
            for(ll k=j+1;k<=r;k++)
            {
                if(p[j]>=min(p[i],p[k])&&p[j]<=max(p[i],p[k]))
                {
                    return 0;
                }
            }
        }
    }
    return 1;
}
void solve()
{
    cin>>n;
    for(ll i=1;i<=n;i++)
    {
        cin>>p[i];
    }
    ll ans=0;
    for(ll i=1;i<=n;i++)
    {
        for(ll j=i;j<=min(i+3,n);j++)//满足条件的子串的长度不会超过4，直接暴力即可
        {
            if(isGood(i,j))
            {
                ans++;
            }
        }
    }
    cout<<ans<<endl;
}
int main()
{
    IOS;
    ll _;
    cin>>_;
    while(_--)
    {
        solve();
    }
}
```

m进制转n进制

2021年3月20日　　18:19

```
#include<bits/stdc++.h>
#define ll long long
#define eps 1.0E-8
#define zero(x) (((x)>0?(x):-(x))<eps)
#define INF 0x3f3f3f3f
#define PI 3.1415926535
using namespace std;
const ll N = 1e5 + 5;
ll turn = 1;
string trans(string num, ll m, ll n)
{
    ll dig=0;
    for (ll i = 0;i<num.length();i++)
    {
        if (num[i] <= '9')dig = dig * m + num[i] - '0';
        else dig = dig * m + num[i] - 'a' + 10;
    }
    string res;

    do
    {
        if (dig % n <= 9)
            res += dig % n + '0';
        else
            res += dig % n - 10 + 'a';
        dig /= n;
    } while (dig);//使用do-while是为了防止dig一开始就为0
    reverse(res.begin(), res.end());
    return res;
}
void solve()
{
    string a;
    ll m, n;
    while (cin >> a >> m >> n)
    {
        cout << trans(a, m, n)<<endl;
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    /* ll _;
     cin >> _;
     while (_--)*/
    solve();
}
```

高精度版本：
https://www.acwing.com/problem/content/126/

```
#include<iostream>
#include<map>
#include<algorithm>
#include<cstring>
#define int long long
using namespace std;
int indig,outdig;
string num;
int ch_num(char ch)
{
   if(ch>='0'&&ch<='9')
   {
      return ch-'0';
   }
   else if(ch>='A'&&ch<='Z')
   {
      return ch-'A'+10;
   }
   else if(ch>='a'&&ch<='z')
   {
      return ch-'a'+36;
   }
}
char num_ch(int num)
{
   if(num>=0&&num<=9)
   {
      return num+'0';
   }
   else if(num>=10&&num<=35)
   {
      return num-10+'A';
   }
   else if(num>=36&&num<=61)
   {
      return num-36+'a';
   }
}
string add(string s,int n)
{
         reverse(s.begin(),s.end());
         int cp=n;
         string res;
         for(int i=0;i<(int)s.length();i++)
         {
            int tmp=ch_num(s[i])+cp;
            res+=num_ch(tmp%10);
            cp=tmp/10;
         }
         while(cp)
```

```
		{
		   res+=num_ch(cp%10);
		   cp=cp/10;
		}
		int i=res.length()-1;
	while(res[i]=='0'&&i>0)i--;
	res=res.substr(0,i+1);
	reverse(res.begin(),res.end());
	return res;
}
string mul(string s,int n)
{
	reverse(s.begin(),s.end());
	int cp=0;
	string res;
	int tmp;
	for(int i=0;i<(int)s.length();i++)
	{
	   int cur=ch_num(s[i]);
	   tmp=cur*n+cp;
	   res+=num_ch(tmp%10);
	   cp=tmp/10;
	}
	while(cp)
	{
	   res+=num_ch(cp%10);
	   cp/=10;
	}
	int i=res.length()-1;
	while(res[i]=='0'&&i>0)i--;
	res=res.substr(0,i+1);
	reverse(res.begin(),res.end());
	return res;
}
char mod(string a,int b)
{
	int res=0;
	for(int i=0;i<(int)a.length();i++)
	{
	   res=(res*10+ch_num(a[i]))%b;
	}
	return num_ch(res);
}
string divid(string a,int b)
{
	string ans;
		int cp = 0;
		for (int i = 0; i < (int)a.length(); i++)
		{
			int v = (cp * 10 + ch_num(a[i])) / b;
			ans += num_ch(v);
			cp= (cp * 10 + ch_num(a[i])) % b;
		}
		int k=0;
		while(ans[k]=='0'&&k<(int)ans.length()-1)
		{
		   k++;
```

```
        }
        ans=ans.substr(k);
        return ans;
}
string trans(string num, int m, int n)
{
    string dig="0";
    for (int i = 0;i<(int)num.length();i++)
    {
        dig=add(mul(dig,m),ch_num(num[i]));
    }
    string res;

    do
    {
        res+= mod(dig,n);
        dig = divid(dig,n);
    } while (dig!="0");//使用do-while是为了防止dig一开始就为0
    reverse(res.begin(), res.end());
    return res;
}

void solve()
{
    cin>>indig>>outdig>>num;
    cout<<indig<<' '<<num<<endl;
    cout<<outdig<<' '<<trans(num,indig,outdig)<<endl<<endl;
}
signed main()
{
    int t;
    cin>>t;
    while(t--)
    {
        solve();
    }
}
```

快速幂

2021年1月19日　　14:38

https://blog.csdn.net/qq_19782019/article/details/85621386

```
#include<iostream>
using namespace std;
long long quickPow(long long base, long long power,long long mod)
{
	long long res = 1;
	while (power > 0)
	{
		if (power & 1)
			res = res *base% mod;
		power >>= 1;
		base = (base * base) % mod;
	}
	return res%mod;
}
int main()
{
	long long a, b,c;
	cin >> a >> b>>c;
	cout << a << '^' << b << ' ' << "mod" << ' ' << c << '=' << quickPow(a, b, c);
}
```

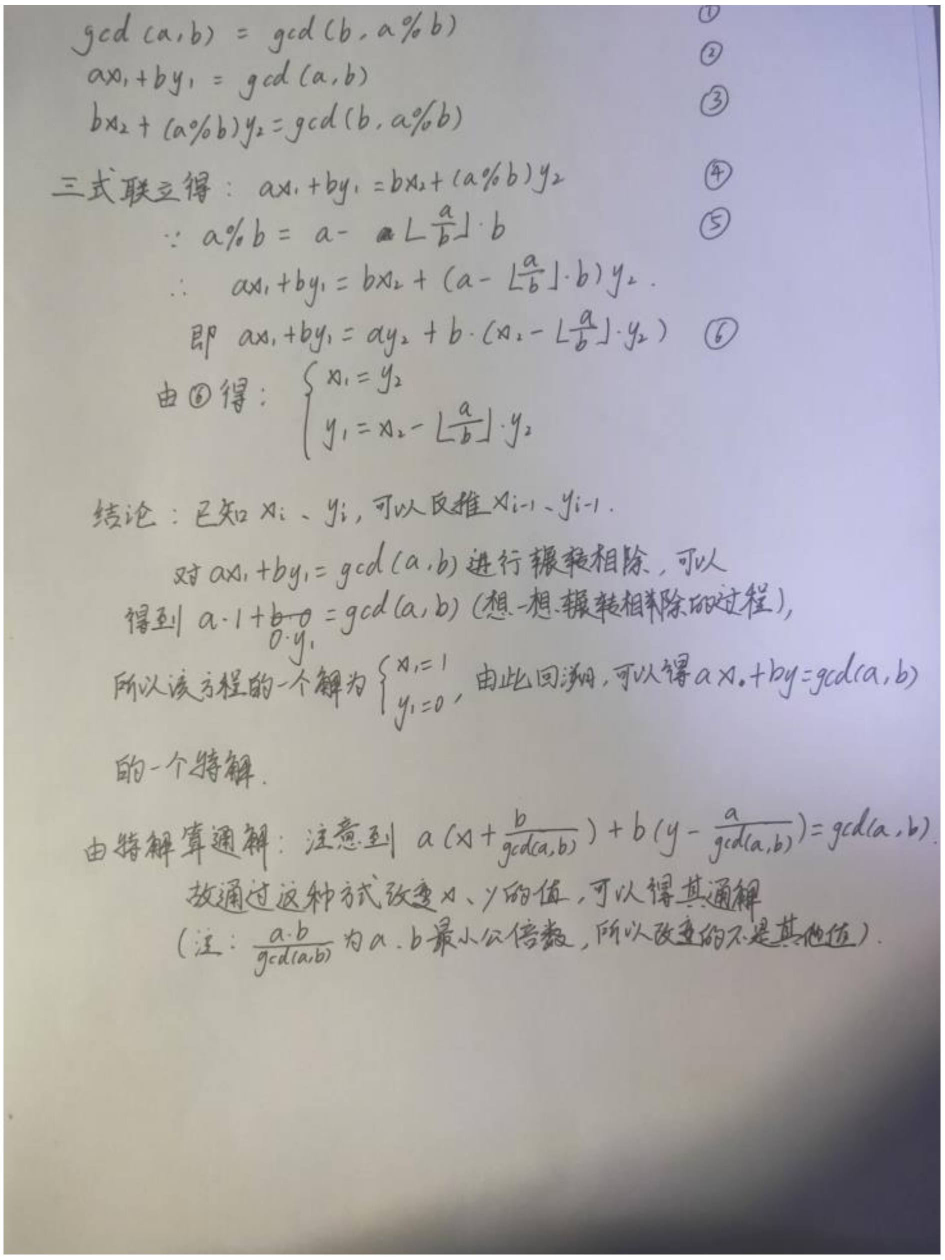
//注意：最后quickPow函数返回值为res%mod而非res,博客里这一点是错的

//如果直接返回res，数据1 0 1会wa

拓展欧几里得

2021年1月19日 15:52

https://blog.csdn.net/weixin_39645344/article/details/83615901
拓展欧几里得算法用于求方程ax+by=gcd(a,b)的整数解



$gcd(a,b) = gcd(b, a\%b)$ ①

$ax_1+by_1 = gcd(a,b)$ ②

$bx_2+(a\%b)y_2 = gcd(b, a\%b)$ ③

三式联立得：$ax_1+by_1=bx_2+(a\%b)y_2$ ④

$\because a\%b = a-\cancel{a}\lfloor\frac{a}{b}\rfloor\cdot b$ ⑤

$\therefore ax_1+by_1 = bx_2+(a-\lfloor\frac{a}{b}\rfloor\cdot b)y_2.$

即 $ax_1+by_1 = ay_2+b\cdot(x_2-\lfloor\frac{a}{b}\rfloor\cdot y_2)$ ⑥

由⑥得：$\begin{cases} x_1=y_2 \\ y_1=x_2-\lfloor\frac{a}{b}\rfloor\cdot y_2 \end{cases}$

结论：已知 x_i、y_i，可以反推 x_{i-1}、y_{i-1}.

对 $ax_1+by_1=gcd(a,b)$ 进行辗转相除，可以

得到 $a\cdot 1+\underset{0\cdot y_1}{\cancel{b\cdot 0}}=gcd(a,b)$ (想一想辗转相除的过程)，

所以该方程的一个解为 $\begin{cases} x_1=1 \\ y_1=0 \end{cases}$，由此回溯，可以得 $ax_0+by=gcd(a,b)$

的一个特解.

由特解算通解：注意到 $a(x+\frac{b}{gcd(a,b)})+b(y-\frac{a}{gcd(a,b)})=gcd(a,b)$.

故通过这种方式改变x、y的值，可以得其通解

(注：$\frac{a\cdot b}{gcd(a,b)}$ 为a、b最小公倍数，所以改变的不是其他值).

```
#include<iostream>
using namespace std;
int exgcd(int a, int b, int& x, int& y)
{
	if (b == 0)
	{
		x = 1;
		y = 0;
		return a;
	}
	else
	{
		int d = exgcd(b, a % b, y, x);
		y -= a / b * x;
		return d;
	}
}
int main()
{
	int a, b;
	cin >> a >> b;
	int x, y;
	cout << exgcd(a, b, x, y) << endl;
	cout << x << ' ' << y << endl;
}
```

费马小定理

2021年1月20日 1:21

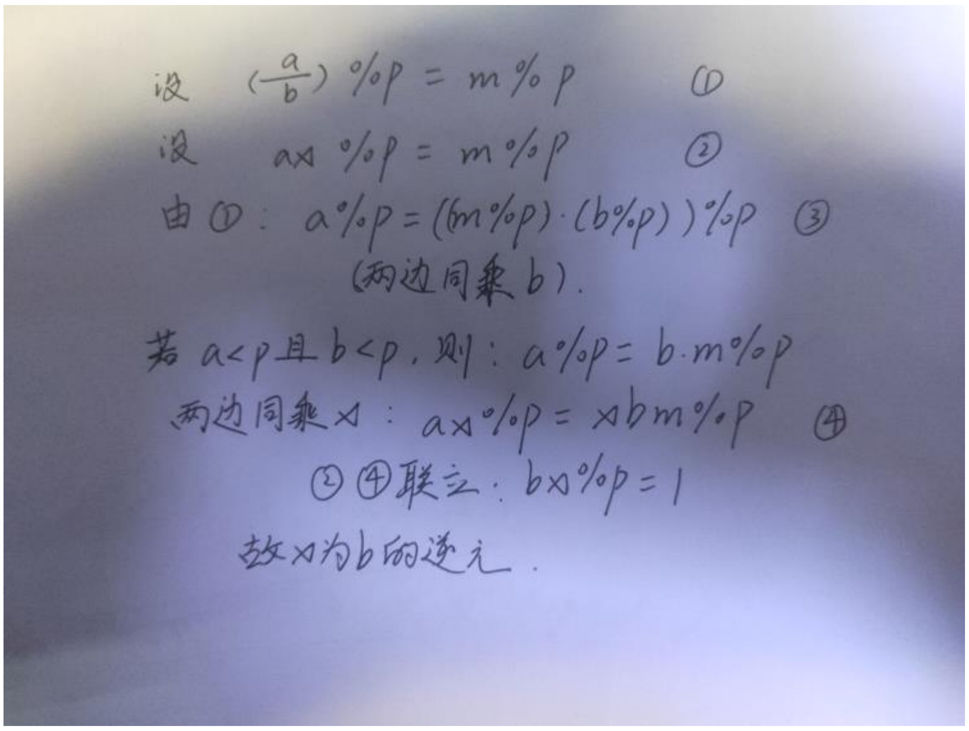
若p为质数且gcd(a,p)=1,则有a^(p-1)=1(mod p)

组合数

2021年1月19日 15:35

https://www.zybuluo.com/ArrowLLL/note/713749

其中关于将模运算除法通过逆元转化为乘法的部分疑似有误，更正如下：



设 $(\frac{a}{b})\%p = m\%p$ ①

设 $ax\%p = m\%p$ ②

由①：$a\%p = ((m\%p)\cdot(b\%p))\%p$ ③

(两边同乘 b).

若 $a<p$ 且 $b<p$，则：$a\%p = b\cdot m\%p$

两边同乘 x：$ax\%p = xbm\%p$ ④

②④联立：$bx\%p = 1$

故 x 为 b 的逆元.

利用费马小定理求组合数：

```
#include<iostream>
#include<cmath>
#include<algorithm>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 2e5 + 5;
const ll mod = 1e9+7;
const double eps = 1e-4;
ll fac[N];
ll inv[N];
ll mul(ll a,ll b)
{
    return ((a%mod)*(b%mod))%mod;
}
ll quick_pow(ll base, ll power)
```

```
{
    ll res = 1;
    while (power)
    {
        if (power & 1)
        {
            res = mul(res,base);
        }
        power >>= 1;
        base=mul(base,base);
    }
    return res % mod;
}
ll comb(ll m, ll n)
{
    if (n > m||n<0)return 0;
    return mul(fac[m],mul(inv[n],inv[m-n]));
}
int main()
{
    IOS;
    fac[0] = 1;
    inv[0] = 1;
    for (ll i = 1; i < N; i++)
    {
        fac[i] = mul(fac[i - 1],i);//预处理，数组fav[i]保存数i的阶乘
        inv[i]=  quick_pow(fac[i],mod-2);
    }
    ll a,b;
    cin>>a>>b;
    cout<<comb(a,b)<<endl;
}
```

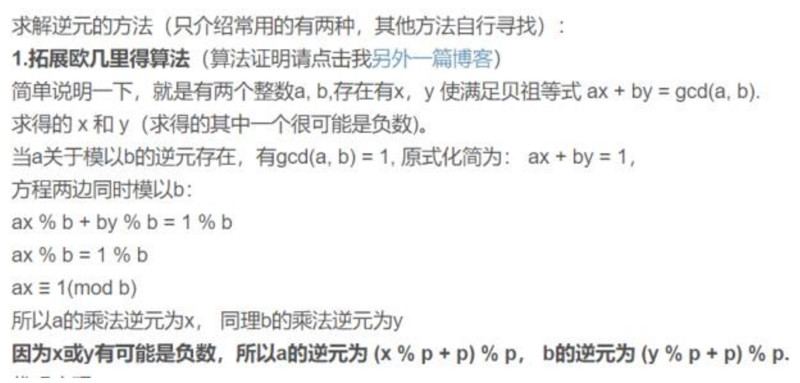
注意，数据范围必须开long long，否则求快速幂的时候来不及取模就爆了

如果不用long long 的话，需要将mul函数改成：

```
int mul(int a,int b)
{
    return ((a%mod)*1ll*(b%mod))%mod;
}
```

利用拓展欧几里得求组合数：

这个更清楚一点：


求解逆元的方法（只介绍常用的有两种，其他方法自行寻找）：

1.拓展欧几里得算法（算法证明请点击我另外一篇博客）

简单说明一下，就是有两个整数a, b,存在有x，y 使满足贝祖等式 ax + by = gcd(a, b).

求得的 x 和 y（求得的其中一个很可能是负数)。

当a关于模以b的逆元存在，有gcd(a, b) = 1, 原式化简为： ax + by = 1,

方程两边同时模以b：

ax % b + by % b = 1 % b

ax % b = 1 % b

ax ≡ 1(mod b)

所以a的乘法逆元为x， 同理b的乘法逆元为y

因为x或y有可能是负数，所以a的逆元为 (x % p + p) % p， b的逆元为 (y % p + p) % p.

其中gcd(a,b)=1是因为一般要求对素数取模，所以b为素数

```
#include<iostream>
#define ll long long
using namespace std;
const ll mod = 1e9+7;
const ll maxn = 1e5 + 5;
ll fac[maxn];
void exgcd(ll a, ll b, ll& x, ll& y)    //拓展欧几里得算法
{
	if (!b) x = 1, y = 0;
	else
	{
		exgcd(b, a % b, y, x);
		y -= x * (a / b);
	}
}
ll inv(ll a)   //求a对mod取模的逆元
{
	ll x, y;
	exgcd(a, mod, x, y);
	return (x + mod) % mod;
}
ll c(ll m, ll n)
{
	if (n > m)return 0;
	return (fac[m] * inv(fac[n]) % mod *inv(fac[m-n]) % mod) % mod;//利用乘法逆元化简组合数
	公式
}
int main()
{
	ll m, n;
	cin >> m >> n;//从m中取出n个元素
	fac[0] = 1;
	for (ll i = 1; i < maxn; i++)
	{
		fac[i] = fac[i - 1] * i % mod;//预处理，数组fav[i]保存数i的阶乘
	}
	cout << c(m, n) << endl;
}
```

也可以直接通过递推求阶乘逆元：

参考：https://blog.csdn.net/qq_40861916/article/details/82928080 看第三部分

例题：http://acm.hdu.edu.cn/showproblem.php?pid=5698

```
#include<iostream>
#define ll long long
using namespace std;
ll const maxn = 3e5 + 100;
ll const mod = 1000000007;
ll fac[maxn];
ll inv[maxn];
ll quick_pow(ll base, ll pow)
{
	ll result = 1;
	while (pow)
	{
		if (pow & 1)
		{
			result = result * base % mod;
		}
		base = base * base % mod;
		pow >>= 1;
	}
	return result;
}
void preLoad()
{
	fac[0] = 1;
	for (ll i = 1; i <= maxn; i++)
	{
		fac[i] = fac[i - 1] * i % mod;
	}
	inv[maxn] = quick_pow(fac[maxn], mod - 2);
	for (ll i = maxn - 1; i >= 0; i--)
	{
		inv[i] = inv[i + 1] * (i + 1) % mod;//利用递推预处理阶乘逆元
	}
}
ll c(ll m, ll n)
{
	return (fac[m] * inv[m-n] % mod * inv[n] % mod) % mod;
}
int main()
{
	ll n, m;
	preLoad();
	while (cin >> n >> m)
	{
		cout << c(n,m) << endl;
	}
}
```

注意：若题目并不要求取模，那么一般的组合数算法可能无法成立（比如说，即使是100的阶乘也爆了long long），这时可能需要考虑使用高精度，或者使用其他解法

不取模情况下，组合数的解法：

在不取模的情况下，组合数有两种递推解法；

第一种，利用组合数递推公式求解：C(m,n)=C(m-1,n-1)+C(m-1,n)

```
#include<iostream>
using namespace std;

long long comb(int m, int n)
{
	if (n == 0) return 1;
	if (n == 1) return m;
	if (n > m / 2) return comb(m, m - n);
	if (n > 1) return (comb(m - 1, n - 1) + comb(m - 1, n));
	return 0;
}

int main()
{
	int m, n;
	while (cin >> m >> n)
		cout << comb(m, n) << endl;
}
```

实际上用递归求解是很慢的，改成递推要快很多（以求解卡特兰数为例）：

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e2 + 5;
const int mod = 1e9;
const double eps = 1e-4;
ll c[N][N];
int main()
{
    IOS;
    ll n;
    cin >> n;

    c[0][0]=1;
    c[1][0]=c[1][1]=1;
    for (int i = 2; i <= 2e3; i++)
    {
        c[i][0]=1;
        for (int j = 1; j <= i; j++)
        {
            c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
        }
    }
    cout << c[2*n][n]-c[2*n][n-1] << endl;//卡特兰数
}
```

第二种，利用对数将组合数公式展开，即令log(C(m,n))=log(m!/(m-n)!)-log(n!)
=log(m-n+1)+log(m-n+2)+...+log(m)-[log(1)+log(2)+...+log(n)];

```
#include<iostream>
#include<math.h>
using namespace std;
double comb_log(int m, int n)//对数求法
{
	int i;
	if (n > m - n) n = m - n;
	double s1 = 0.0, s2 = 0.0;
	for (i = m - n + 1; i <= m; ++i) s1 += log(i); //求 log( m!/(m-n)!)
	for (i = 1; i <= n; ++i) s2 += log(i); // 求log n!
	return exp(s1 - s2);
}
int main()
{
	int m, n;
	while (cin >> m >> n) {
		cout << (long long)(comb_log(m, n)) << endl;
	}
	return 0;
}
```

二维计算几何

2021年10月23日　15:27

```
#include<bits/stdc++.h>
using namespace std;
struct Point{
   double x,y;
   Point(double x=0,double y=0):x(x),y(y){}
};
typedef Point Vector;//从程序实现上说，Vector只是Point的别名
//向量+向量=向量，点+向量=点
Vector operator+(Vector  A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
//点-点=向量
Vector operator-(Vector  A,Vector B){return Vector(A.x-B.x,A.y-B.y);}
//向量*数=向量
Vector operator*(Vector A,double p){return Vector(A.x*p,A.y*p);}
//向量/数=向量
Vector operator/(Vector A,double p){return Vector(A.x/p,A.y/p);}

bool operator<(const Point&a,const Point&b){
   return a.x<b.x||(a.x==b.x&&a.y<b.y);
}

const double eps=1e-10;
int dcmp(double x)
{
   if(fabs(x)<eps)return 0;else return x<0?-1:1;
}
bool operator==(const Point&a,const Point&b)
{
   return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
}
//C标准库中的atan2函数可以用来求向量极角，如向量(x,y)的极角就是atan2(y,x),单位是弧度
//点积等于两向量模长相乘再乘以二者夹角的余弦值
double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return acos(Dot(A,B)/Length(A)/Length(B));}
//两向量的叉积等于两向量组成的三角性的有向面积的两倍
//cross(v,w)>0,当w在v左边；cross(v,w)<0，当w在v右边
double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;}
double Area2(Point A,Point B,Point C){return Cross(B-A,C-A);}
/*使用点积和叉积可以判断两个向量的相对位置
假设有两向量v和w，以v作为坐标系的x轴正方向，那么对于不同的w位置，点积和叉积的符号是：
 （以下第一个符号是点积，第二个符号是叉积）
x轴正方向:(+,0)
第一象限：(+,+)
```

```
y轴正方向：(0,+)
第二象限：(-,+)
x轴负方向：(-,0)
第三象限：(-,-)
y轴负方向：(0,-)
第四象限：(+,-)
*/
//将向量逆时针旋转rad度（弧度）
Vector Rotate(Vector A,double rad)
{
   return Vector(A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
}
//计算向量的单位法线，即左转九十度后将长度归一化
//注意该向量不能是零向量
//另外，若两向量平行，则叉积为0；若两向量垂直，则点积为0
Vector Normal(Vector A)
{
   double L=Length(A);
   return Vector(-A.y/L,A.x/L);
}
//直线的参数方程:
//已知直线上一点P0和方向向量v，则直线上所有点满足P=P0+t*v
//射线和线段的方程形式也是一致的
//求P+tv和Q+tw两条直线的交点，使用前需确定两直线不平行
Point GetLineIntersection(Point P,Vector v,Point Q,Vector w)
{
   Vector u=P-Q;
   double t=Cross(w,u)/Cross(v,w);
   return P+v*t;
}
//点P到直线AB距离，用平行四边形面积除以底
double DistanceToLine(Point P,Point A,Point B)
{
   Vector v1=B-A,v2=P-A;
   return fabs(Cross(v1,v2)/Length(v1));
}
//点到线段距离，注意不是到投影点的距离
double DistanceToSegment(Point P,Point A,Point B)
{
   if(A==B)return Length(P-A);
   Vector v1=B-A,v2=P-A,v3=P-B;
   if(dcmp(Dot(v1,v2))<0)return Length(v2);
   else if(dcmp(Dot(v1,v3))>0)return Length(v3);
   else return fabs(Cross(v1,v2))/Length(v1);
}
//求点P到直线AB的投影点
Point GetLineProjection(Point P,Point A,Point B)
{
   Vector v=B-A;
   return A+v*(Dot(v,P-A)/Dot(v,v));
```

```
}
//判断线段规范相交(相交点不允许是端点)
bool SegmentProperIntersection(Point a1,Point a2,Point b1,Point b2)
{
   double c1=Cross(a2-a1,b1-a1),c2=Cross(a2-a1,b2-a1),
            c3=Cross(b2-b1,a1-b1),c4=Cross(b2-b1,a2-b1);
   return dcmp(c1)*dcmp(c2)<0&&dcmp(c3)*dcmp(c4)<0;
}
/*
如果允许两线段在端点相交，那么：
如果c1和c2均为0，那么两线段共线，可能有部分重叠；
如果不都为0，那么唯一的相交办法就是其中一条线段的端点在另一条线段上
*/
//判断一个点是否在一条线段上（不含端点）
bool OnSegment(Point P,Point a1,Point a2)
{
   return dcmp(Cross(a1-P,a2-P))==0&&dcmp(Dot(a1-P,a2-P))<0;
}
//n个顶点多边形的有向面积，将其划分为n-2个三角形即可（如果多边形非凸，那么多余的面
积会正负抵消）
double ConvexPolygonArea(Point *p,int n)
{
   double area=0;
   for(int i=1;i<n-1;i++)
   {
      area+=Cross(p[i]-p[0],p[i+1]-p[0]);
   }
   return area/2;
}
Point getD(Point A,Point B,Point C)
{
   Vector v1=C-B;
   double a1=Angle(A-B,v1);
   v1=Rotate(v1,a1/3);

   Vector v2=B-C;
   double a2=Angle(A-C,v2);
   v2=Rotate(v2,-a2/3);
   return GetLineIntersection(B,v1,C,v2);
}
int main()
{
   int t;
   cin>>t;
   while(t--)
   {
      Point A,B,C,D,E,F;
      cin>>A.x>>A.y;
      cin>>B.x>>B.y;
      cin>>C.x>>C.y;
      D=getD(A,B,C);
      E=getD(B,C,A);
      F=getD(C,A,B);
      printf("%.6lf %.6lf %.6lf %.6lf %.6lf %.6lf\n",D.x,D.y,E.x,E.y,F.x,F.y);
```

```
    }
}
```

模数的处理

2021年9月9日 22:41

两个数做模意义减法，如果最后的结果是负数，要不断加上模数直到结果为正

筛法

2021年1月14日　　21:48

埃氏筛：
从n个数中筛出素数，每筛出一个，其倍数必不为素数
例题：
输入$n(n\leqslant 100)$ 个不大于 100000 的整数。要求全部储存在数组中，去除掉不是质数的数字，依次输出剩余的质数。

来自 <https://www.luogu.com.cn/problem/P5736>

```
#include<iostream>
#define ll long long
using namespace std;
const  ll maxn = 1e5 + 5;
int is_prime[maxn] = { 0 };
void sieve()//埃氏筛法
{
	for (int i = 0; i < maxn; i++)
		is_prime[i] = 1;//一开始默认每个数都是素数
	is_prime[0] = is_prime[1] = 0;
	for (int i = 2; i <= maxn; i++)
	{
		if (is_prime[i]) {
			for (int j = i * 2; j <= maxn; j += i)//筛倍数
			{
				is_prime[j] = 0;
			}
		}
	}
}
int main() {
	int n;
	cin >> n;
	sieve();
	for (int i = 0; i < n; i++)
	{
		int t;
		cin >> t;
		if (is_prime[t])cout << t << ' ';
	}


}
```

欧拉筛（线性筛）：
由于埃氏筛法中每个合数可能被筛多次，所以在欧拉筛中，对每个合数，只由其最小质因数筛去，以保证其只筛一次
例：https://www.luogu.com.cn/problem/P3383

给出范围n与q次询问，每次询问第k大的素数

```
#include<iostream>
#include<vector>
using namespace std;
const int maxn = 1e8+1;
vector<int>prime;
int vis[maxn] = { 0 };
int sieve(int b)//欧拉筛
{
	int cnt = 0;
	for (int i = 2; i <= b; i++)
	{
		if (!vis[i]) {
			prime.push_back(i);
			cnt++;
		}
		for (int j = 0; j <= cnt && i * prime[j] <= b; j++)
		{
			vis[i * prime[j]] = 1;
			if (i % prime[j] == 0)break;//如果i为prime[j]的倍数即i=k*prime[j],那么在筛下一
			个时i*prime[j+1]=prime[j]*k*prime[j+1],在i=k*prime[j+1]时prime[j]就会被重复
			筛一次，所以这里要跳出
		}
	}
	return cnt;
}
int main()
{
	int n, q;
	std::ios::sync_with_stdio(0);
	cin >> n >> q;
	int cnt=sieve(n);
	for (int i=0;i<q;i++)
	{
		int index;
		cin >> index;
		cout << prime[index-1] << endl;
	}
}
```

另一种我觉得更容易理解的写法：

```
#include<bits/stdc++.h>
#define ll long long
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f3f3f3f3f
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll mod=998244353;
const ll N=1e6+5;
const ll M=5e3+5;
ll a,b;
ll v[N];
vector<ll>prime;
void sieve(ll n)
{
    for(ll i=2;i<=n;i++)
    {
        if(v[i]==0)
        {
            v[i]=i;
            prime.push_back(i);
        }
        for(auto j:prime)
        {
            if(j>v[i]||j>n/i)break;
            v[i*j]=j;
        }
    }
}
int main()
{
    sieve(1000000);
}
```

https://www.luogu.com.cn/problem/P1835

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=1e6+5;
const ll mod=1e9+7;
ll l,r;
ll vis[N];
vector<ll>prime;
void sieve(ll b)
{
    ll cnt=0;
    for(ll i=2;i<=b;i++)
    {
        if(vis[i]==0)
        {
            prime.push_back(i);
            cnt++;
        }
        for(ll j=0;j<cnt&&i*prime[j]<=b;j++)
        {
            vis[i*prime[j]]=1;
            if(i%prime[j]==0)break;
        }
    }
}
int main()
{
    IOS;
    cin>>l>>r;
    sieve(100000);
    if(l==1)l++;
    memset(vis,0,sizeof vis);
    for(auto i:prime)
    {
        ll j;
        if(i>r)break;
        if(l%i==0&&l!=i)
        {
            j=l;
        }
        else if(l>=i){
            j=l/i*i+i;
        }
        else{
            j=i*2;
        }
        //cout<<"start from "<<j<<endl;
        for(;j<=r;j+=i)
        {
            //cout<<"delete "<<j<<endl;
            vis[j-l]=1;
        }
    }
    ll ans=0;
    for(ll i=0;i<=r-l;i++)
    {
        if(!vis[i])ans++;
        //cout<<vis[i]<<' ';
    }
    cout<<ans<<endl;
}
```

迪杰斯特拉

2021年1月19日　　15:44

https://pintia.cn/problem-sets/994805046380707840/problems/994805073643683840

```
#include<iostream>
#include<cstring>
#include<vector>
#define ll long long
#define INF 0x3f3f3f3f
using namespace std;
ll const maxn = 1000;
ll graph[maxn][maxn];//邻接矩阵
ll teamN[maxn];//每个城市的队伍数
ll dis[maxn];//源点到每个点的最短距离
ll v[maxn];//是否访问过
ll final_teamN[maxn];//记录最终到达每个城市可积累的最大队伍数
vector<ll> road[maxn];//记录源点到每个城市最短路所经过的节点
ll roadN[maxn];//储存到每个城市最短路径的条数
void init(ll n,ll s,ll e)
{
        memset(dis, 127 / 3, sizeof(dis));
        for (ll i = 0; i < maxn; i++)
                road[i].clear();
        memset(final_teamN, 0, sizeof(final_teamN));
        memset(roadN, 0, sizeof(roadN));
        memset(v, 0, sizeof(v));
        for (ll i = 0; i < n; i++) {
                if (graph[s][i] != INF) {
                        dis[i] = graph[s][i];
                        road[i].push_back(s);
                        road[i].push_back(i);//最开始，源点到每个城市经过的节点就是"源点->节点"
                        final_teamN[i] = teamN[s] + teamN[i];
                        roadN[i] = 1;
                }
        }
        v[s] = 1;
        dis[s] = 0;
        roadN[s] = 1;
}
ll dijkstra(ll n, ll s,ll e)
{


        for (ll i = 0; i < n; i++)
        {
                ll min = INF;
                ll k = s;
                for (ll j = 0; j < n; j++)
                {
                        if (!v[j] && ( dis[j] < min))
                {
                    k = j;
                    min = dis[j];
                }
            }
            v[k] = 1;
            for (ll j = 0; j < n; j++)
            {
                if (!v[j] && graph[k][j]!=INF)
                {
                    if (dis[k] + graph[k][j] < dis[j])
                    {
                        roadN[j] = roadN[k];
                        dis[j] = dis[k] + graph[k][j];
                        road[j].clear();
                        for (auto z : road[k])
                        {
                            road[j].push_back(z);
                        }
                        road[j].push_back(j);
                        final_teamN[j] =final_teamN[k]+ teamN[j];
                    }
                    else if(dis[k] + graph[k][j] == dis[j])
                    {
                        roadN[j] += roadN[k];
                        if (final_teamN[k] + teamN[j] > final_teamN[j])
                        {
                            road[j].clear();
                            for (auto z : road[k])
                            {
                                road[j].push_back(z);
                            }
                            road[j].push_back(j);
                            final_teamN[j] = final_teamN[k] + teamN[j];
                        }
                    }
                }
            }
        }
        return final_teamN[e];
}
int main()
{
        ll n, m, s, d;//城市数，道路数，开始地，到达地
        while (cin >> n >> m >> s >> d)
        {
            for (ll i = 0; i < n; i++)
            {
                cin >> teamN[i];
            }
            for (ll i = 0; i < n; i++)
            {
                for (ll j = 0; j < n; j++)
                {
                    if (i != j)
                        graph[i][j] = INF;
                    else
                        graph[i][j] = 0;
                }
            }

            for (ll i = 0; i < m; i++)
            {
                ll c1, c2, len;
                cin >> c1 >> c2 >> len;
                graph[c1][c2] = len;
                graph[c2][c1] = len;//注意道路不是单向的
            }


            init(n, s, d);
            ll ans = dijkstra(n, s, d);
            cout << roadN[d] << ' ' << ans << endl;
            for (vector<ll>::iterator i = road[d].begin(); i != road[d].end(); i++)
            {
                if (i != road[d].end() - 1)
                    cout << (*i) << ' ';
                else
                    cout << (*i);
            }
        }
}
```

并查集

2021年1月19日　　15:58

```
#include<iostream>
using namespace std;
int pre[1000];
int input[1000];
int find(int i)//寻点
{
        if (i == pre[i])return i;
        return pre[i] = find(pre[i]);//含路径压缩
}
void merge(int x, int y)//合并
{
        int fx = find(x), fy = find(y);
        if (fx != fy)pre[fx] = fy;
}
int main()
{
        //暂无测试代码
}
//如果要计数的话，只需要数有多少个parent[i]==i的节点就行
```

连通块中点的数量：

https://www.acwing.com/problem/content/839/

```
#include<iostream>
using namespace std;
const int N=1e5+5;
int parent[N];
int psize[N];
int n,m;
int find(int i)
{
   if(parent[i]==i)return i;
   else return parent[i]=find(parent[i]);
}
void merge(int a,int b)
{
   int i=find(a);
   int j=find(b);
   if(i!=j)
   {
      parent[i]=j;
      psize[j]+=psize[i];
   }
}
int main()
{
   scanf("%d%d",&n,&m);
   for(int i=1;i<=n;i++){
      parent[i]=i;
```

```
        psize[i]=1;
    }
    while(m--)
    {
        char op[5];
        scanf("%s",op);
        if(op[0]=='C')
        {
            int a,b;
            scanf("%d%d",&a,&b);
            merge(a,b);
        }
        else
        {
            if(op[1]=='1')
            {
                int a,b;
                scanf("%d%d",&a,&b);
                if(find(a)==find(b))
                {
                    cout<<"Yes"<<endl;
                }
                else
                {
                    cout<<"No"<<endl;
                }
            }
            else if(op[1]=='2')
            {
                int a;
                scanf("%d",&a);
                cout<<psize[find(a)]<<endl;
            }
        }
    }
}
```

https://vjudge.net/problem/POJ-2236

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e3+5;
const ll mod=1e9+7;
ll n;
ll d;
ll pre[N];
```

```
pll a[N];
ll repaired[N];
ll find(ll i)
{
    if(pre[i]==i)return i;
    else return pre[i]=find(pre[i]);
}
void merge(ll a,ll b)
{
    ll pa=find(a),pb=find(b);
    if(pa!=pb)pre[pa]=pb;
}
double dis(pll a,pll b)
{
    double s=a.first-b.first;
    double r=a.second-b.second;
    return sqrt(s*s+r*r);
}
int main()
{
    IOS;
    cin>>n>>d;
    for(ll i=1;i<=n;i++)
    {
        cin>>a[i].first>>a[i].second;
        pre[i]=i;
    }
    char op;
    while(cin>>op)
    {
        if(op=='O')
        {
            ll p;
            cin>>p;
            repaired[p]=1;
            for(ll i=1;i<=n;i++)
            {
                if(dis(a[i],a[p])<=d&&repaired[i]&&i!=p)
                {
                    merge(i,p);
                }
            }
        }
        else{
            ll s,t;
            cin>>s>>t;
            if(find(s)==find(t))
            {
                cout<<"SUCCESS"<<endl;
            }
            else{
                cout<<"FAIL"<<endl;
            }
        }
    }
}
```

https://vjudge.net/problem/POJ-1611
```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e5+5;
const int mod=1e9+7;
ll pre[N];
ll vis[N];
ll a[N];
ll find(ll i)
{
    if(pre[i]==i)return i;
    else return pre[i]=find(pre[i]);
}
void merge(ll i,ll j)
{
    ll a=find(i);
    ll b=find(j);
    if(a!=b)
    {
        pre[a]=b;
    }
}
inline ll read() {
    ll x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch>'9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
int main()
{
    IOS;
    ll n,m;
    while(cin>>n>>m)
    {
        if(n==0&&m==0)break;
        for(ll i=0;i<n;i++)
        {
            pre[i]=i;
            vis[i]=0;
        }
        ll t=0;
        for(ll i=0;i<m;i++)
        {
            ll k;
            cin>>k;
            for(ll i=0;i<k;i++)
            {
                cin>>a[++t];
                if(i!=0)
                {
                    merge(a[t],a[t-1]);
                }
            }
        }
        ll ans=0;
        ll root=find(0);
        for(ll i=0;i<n;i++)
        {
            if(find(i)==root&&!vis[i])
            {
                vis[i]=1;
                ans++;
            }
        }
        cout<<ans<<endl;
    }
}
```

https://vjudge.net/problem/HDU-1213

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll n,m;
ll pre[N];
ll find(ll  i)
```

```
{
    if(pre[i]==i)return i;
    else return pre[i]=find(pre[i]);
}
void merge(ll i,ll j)
{
    ll a=find(i);
    ll b=find(j);
    if(a!=b)
    {
        pre[a]=b;
    }
}
int main()
{
    IOS;
    ll t;
    cin>>t;
    while(t--)
    {
        cin>>n>>m;
        for(ll i=1;i<=n;i++)
        {
            pre[i]=i;
        }
        while(m--)
        {
            ll a,b;
            cin>>a>>b;
            merge(a,b);
        }
        ll ans=0;
        for(ll i=1;i<=n;i++)
        {
            if(pre[i]==i)ans++;
        }
        cout<<ans<<endl;
    }
}
```

https://www.luogu.com.cn/problem/P1525

```
#include<iostream>
#include<cmath>
#include<vector>
#include<algorithm>
#include<cstring>
#include<queue>
#include<map>
#include<set>
#define INF 3e9
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 1e5 + 5;
const ll mod = 1e9+7;
const double eps = 1e-4;
ll n,m;
ll pa[N];
struct edge{
    ll u,v,w;
}e[N];
vector<ll>g[N];
int cmp(edge&a,edge&b)
{
    return a.w>b.w;
}
ll find(ll x)
{
    if(pa[x]==x)return x;
    else return pa[x]=find(pa[x]);
}
void merge(ll x,ll y)
{
    ll i=find(x);
    ll j=find(y);
    if(i!=j)
    {
        pa[i]=j;
    }
}
int main()
{
    IOS;
    cin>>n>>m;
    for(ll i=0;i<m;i++)
    {
        cin>>e[i].u>>e[i].v>>e[i].w;
    }
    for(ll i=1;i<=n;i++)
    {
        pa[i]=i;
    }
    sort(e,e+m,cmp);
    ll ans=0;
    for(ll i=0;i<m;i++)
    {
       ll x=e[i].u;
       ll y=e[i].v;
       if(find(x)==find(y))
       {
           ans=e[i].w;
           break;
       }
       else{
          for(auto j:g[x])
          {
              merge(j,y);
          }
          for(auto j:g[y])
          {
              merge(j,x);
          }
```

```
            g[x].push_back(y);
            g[y].push_back(x);
        }
    }
    cout<<ans<<endl;
}
```

https://www.luogu.com.cn/problem/P1621

```
#include<iostream>
#include<cmath>
#include<vector>
#include<algorithm>
#include<cstring>
#include<queue>
#include<map>
#include<set>
#define INF 3e9
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 1e5 + 5;
const ll mod = 1e9+7;
const double eps = 1e-4;
ll a,b,p;
vector<int>prime;
int vis[N] = { 0 };
ll pa[N];
int sieve(int b)//欧拉筛
{
    int cnt = 0;
    for (int i = 2; i <= b; i++)
    {
        if (!vis[i]) {
            prime.push_back(i);
            cnt++;
        }
        for (int j = 0; j <= cnt && i * prime[j] <= b; j++)
        {
            vis[i * prime[j]] = 1;
            if (i % prime[j] == 0)break;//如果i为prime[j]的倍数即i=k*prime[j],那
么在筛下一个时i*prime[j+1]=prime[j]*k*prime[j+1],在i=k*prime[j+1]时prime[j]就会
被重复筛一次，所以这里要跳出
        }
    }
    return cnt;
}
ll find(ll x)
{
    if(pa[x]==x)return x;
    else return pa[x]=find(pa[x]);
}
void merge(ll x,ll y)
{
    ll i=find(x);
    ll j=find(y);
    if(i!=j)
    {
        pa[i]=j;
    }
}
int main()
{
    IOS;
    cin>>a>>b>>p;
    for(ll i=a;i<=b;i++)
    {
        pa[i]=i;
    }
    sieve(b);
    ll i=0;
    while(prime[i]<p)i++;
    for(;i<prime.size();i++)
    {
        ll k=1;
        if(2*prime[i]>b)break;
        //cout<<"cur prime is "<<prime[i]<<endl;
        while(k*prime[i]<a)k++;
        ll root=k*prime[i];
        while(k*prime[i]<=b)
        {
            merge(root,k*prime[i]);
            //cout<<"merge "<<root<<' '<<k*prime[i]<<endl;
            k++;
        }
    }
    ll ans=0;
    for(ll i=a;i<=b;i++)
    {
        if(pa[i]==i)ans++;
    }
    cout<<ans<<endl;
}
```

https://www.luogu.com.cn/problem/P1955

```
#include<iostream>
#include<cmath>
#include<vector>
#include<algorithm>
#include<cstring>
#include<queue>
#include<map>
#include<set>
#define INF 3e9
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 1e6 + 5;
const ll mod = 1e9+7;
const double eps = 1e-4;
ll pa[N];
vector<ll>num;
ll n,m;
struct node{
    ll u,v,w;
}re[N];
ll find(ll x)
{
    if(pa[x]==x)return x;
    else return pa[x]=find(pa[x]);
}
void merge(ll x,ll y)
{
    ll i=find(x);
    ll j=find(y);
    if(i!=j)
    {
        pa[i]=j;
    }
}
int cmp(node&a,node&b)
{
    return a.w>b.w;
}
void solve()
{
    ll n;
    cin>>n;
    int f=1;
    num.clear();
   for(ll i=0;i<n;i++)
   {
      ll u,v,w;
      cin>>u>>v>>w;
      re[i].u=u;
      re[i].v=v;
      re[i].w=w;
     num.push_back(u);
     num.push_back(v);
   }
   sort(num.begin(),num.end());
   num.erase(unique(num.begin(),num.end()),num.end());
   for(ll i=0;i<n;i++)
   {
       re[i].u=lower_bound(num.begin(),num.end(),re[i].u)-num.begin();
       re[i].v=lower_bound(num.begin(),num.end(),re[i].v)-num.begin();
   }
   sort(re,re+n,cmp);
   for(ll i=0;i<2*n;i++)
   {
       pa[i]=i;
   }
   for(ll i=0;i<n;i++)
   {
       if(re[i].w==1)
       {
```

```
                merge(re[i].u,re[i].v);
            }
            else{
                if(find(re[i].u)==find(re[i].v))
                {
                    f=0;
                    break;
                }
            }
        }
        if(f)cout<<"YES"<<endl;
        else cout<<"NO"<<endl;

    }
    int main()
    {
         IOS;
        ll _;
        cin>>_;
        while(_--)
        {
            solve();
        }
    }
```

离散化

2021年7月20日　　20:43

https://www.luogu.com.cn/problem/P4305

```
#include<iostream>
#include<cmath>
#include<vector>
#include<algorithm>
#include<cstring>
#include<queue>
#include<map>
#include<set>
#define INF 3e9
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 1e6 + 5;
const ll mod = 1e9+7;
const double eps = 1e-4;
int vis[N];
vector<ll>num;
vector<ll>ans;
void solve()
{
    ll n;
    cin>>n;
    ans.clear();
    num.clear();
    memset(vis,0,sizeof vis);
    for(ll i=0;i<n;i++)
    {
        ll a;
        cin>>a;
        num.push_back(a);
        ans.push_back(a);
    }
    sort(num.begin(),num.end());
    num.erase(unique(num.begin(),num.end()),num.end());
    for(ll i=0;i<ans.size();i++)
    {
        ans[i]=lower_bound(num.begin(),num.end(),ans[i])-num.begin();
    }
    for(ll i=0;i<ans.size();i++)
    {
        if(!vis[ans[i]])
        {
            vis[ans[i]]=1;
            cout<<num[ans[i]]<<' ';
        }
        }
        cout<<endl;

}
int main()
{
        IOS;
      ll _;
      cin>>_;
      while(_--)
      {
            solve();
      }
}
```

字符串哈希

2021年7月20日　　23:59

https://www.luogu.com.cn/problem/P3405

```
#include<iostream>
#include<cmath>
#include<vector>
#include<algorithm>
#include<cstring>
#include<queue>
#include<map>
#include<set>
#define INF 3e9
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 1e6 + 5;
const ll mod = 1e9+7;
const double eps = 1e-4;
ll city[1000][1000];
int main()
{
    IOS;
    ll n;
    cin>>n;
    ll ans=0;
    for(ll i=0;i<n;i++)
    {
        string s,na;
        cin>>na>>s;
        na.substr(0,2);
        ll t1=(ll)(s[0]-'A')*26+(ll)(s[1]-'A');//将字符串转换为一个26进制数(进制太小会导致哈希冲突)
        ll t2=(ll)(na[0]-'A')*26+(ll)(na[1]-'A');
        if(t1!=t2)
        {
            city[t1][t2]++;
            ans+=city[t2][t1];
        }
    }
    cout<<ans<<endl;

}
```

https://www.acwing.com/problem/content/140/

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define ull unsigned long long
```

```
const ll N=1e6+5;
string s;
int query;
ull h[N];
ull p=131;
ull fac[N];
void make_hash()
{
   h[0]=s[0]-'a'+1;
   for(int i=1;i<s.length();i++)
   {
      h[i]=h[i-1]*p+s[i]-'a'+1;
   }
}
ull get_hash(ll l,ll r)
{
   if(l==0)return h[r];
   return h[r]-h[l-1]*fac[r-l+1];
}
void init(ll n)
{
   fac[0]=1;
   fac[1]=p;
   for(int i=2;i<=n;i++)
   {
      fac[i]=fac[i-1]*p;
   }
}
void solve()
{
   cin>>s;
   cin>>query;
   init(s.length());
   make_hash();
   while(query--)
   {
      int l1,r1,l2,r2;
      cin>>l1>>r1>>l2>>r2;
      l1--;
      r1--;
      l2--;
      r2--;
      if(get_hash(l1,r1)==get_hash(l2,r2))
      {
         cout<<"Yes"<<endl;
      }
      else{
         cout<<"No"<<endl;
      }
   }
}
int main()
{
      solve();
}
```

https://www.acwing.com/problem/content/141/

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
#define ull unsigned long long
using namespace std;
const int N=1e6+5;
int p=131;
string s;
ull h[N];
ull rh[N];
ull fac[N];
int cas=1;
void init()
{
   fac[0]=1;
   fac[1]=p;
   for(int i=1;i<=N-1;i++)
   {
      fac[i]=fac[i-1]*p;
   }
}
ull get_hash(int l,int r)
{
   if(l==0)return h[r];
   else return h[r]-h[l-1]*fac[r-l+1];
}
ull get_rhash(int l,int r)
{
   if(r==s.length()-1)return rh[l];
   else return rh[l]-rh[r+1]*fac[r-l+1];
}
void solve()
{
   memset(h,0,sizeof h);
   memset(rh,0,sizeof rh);
   h[0]=s[0]-'a'+1;
   for(int i=1;i<s.length();i++)
   {
      h[i]=h[i-1]*p+s[i]-'a'+1;
   }
   rh[s.length()-1]=s[s.length()-1]-'a'+1;
   for(int i=s.length()-2;i>=0;i--)
   {
      rh[i]=rh[i+1]*p+s[i]-'a'+1;
   }
   int ans=0;
   //cout<<"odd"<<endl;
   for(int i=0;i<s.length();i++)
   {
      int l=0,r=min(i,(int)s.length()-i-1);
      while(l<r)
      {

         int mid=(l+r+1)/2;
         if(get_hash(i-mid,i+mid)==get_rhash(i-mid,i+mid))
         {
            l=mid;
         }
         else
         {
            r=mid-1;
         }
      }
      //cout<<"hash of "<<i-l<<' '<<i+l<<" is "<<get_hash(i-l,i+l)<<endl;
      //cout<<"rhash of "<<i-l<<' '<<i+l<<" is "<<get_rhash(i-l,i+l)<<endl;
      if(get_hash(i-l,i+l)==get_rhash(i-l,i+l)){
         ans=max(ans,(i+l)-(i-l)+1);
         //cout<<"found "<<i-l<<' '<<i+l<<endl;
      }
   }
   //cout<<"even"<<endl;
   for(int i=0;i<s.length();i++)
   {
      int l=0,r=min(i,(int)s.length()-i-1-1);
      while(l<r)
      {
         int mid=(l+r+1)/2;
         if(get_hash(i-mid,i+1+mid)==get_rhash(i-mid,i+1+mid))
         {
            l=mid;
         }
         else
         {
            r=mid-1;
         }

      }

      if(get_hash(i-l,i+1+l)==get_rhash(i-l,i+1+l)){
         ans=max(ans,(i+1+l)-(i-l)+1);
         //cout<<"found "<<i-l<<' '<<i+1+l<<endl;
      }
   }
   printf("Case %d: %d\n",cas++,ans);
}
int main()
{
   init();
   while(cin>>s)
   {
      if(s=="END")break;
      solve();
   }
}
```

LIS（最长上升子序列）

2021年1月19日 16:00

```
#include<iostream>
#include<algorithm>
using namespace std;
int num[1005] = { 0 };
int dp[1005] = { 0 };
int main()
{
	int n;
	cin >> n;
	for (int i = 0; i < n; i++)
	{
		cin >> num[i];
		dp[i] = 1;
	}
	for (int i = 0; i < n; i++)
	{
		for (int j = 0; j < i; j++)
		{
			if (num[i] > num[j])
				dp[i] = max(dp[i], dp[j] + 1);
		}
	}
	int ans = 0;
	for (int i = 0; i < n; i++)
	{
		ans = max(ans, dp[i]);
	}
	cout << ans << endl;
}
```

权值线段树维护版本，复杂度O(nlogn):

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N = 6e5 + 7;

int n, a[N], k, tree[4 * N];

#define m (l + r) / 2
#define lson 2 * node
#define rson 2 * node + 1

void update(int pos, int v, int l, int r, int node) {
   if (l == r) {
      tree[node] = v;
      return;
   }
   if (pos <= m) update(pos, v, l, m, lson);
   else update(pos, v, m + 1, r, rson);
   tree[node] = max(tree[lson], tree[rson]);
}

int query(int ql, int qr, int l, int r, int node) {
   if (ql <= l && qr >= r) {
      return tree[node];
   }
   int ans = 0;
   if (ql <= m) ans = max(ans, query(ql, qr, l, m, lson));
   if (qr > m) ans = max(ans, query(ql, qr, m + 1, r, rson));
   return ans;
}
int main() {
   scanf("%d %d", &n, &k);
   for (int i = 1; i <= n; i++) {
      scanf("%d", &a[i]);
   }
   int ans = 0;
   for (int i = 1; i <= n; i++) {
      int res = query(max(a[i] - k, 0), a[i] + k, 0, N, 1) + 1;//千万注意query传入的是大写N，不是小写
n
      ans = max(ans, res);
      update(a[i], res, 0, N, 1);
   }
   cout << ans << endl;

}
```

子集构造

2021年1月24日 15:06

例：输出0-n-1的所有子集

方法一：增量构造

```
#include<iostream>
using namespace std;
void sub_set(int n, int* a, int cur)
{
        for (int i = 0; i < cur; i++)
                cout << a[i] << ' ';
        cout << endl;
        int s = cur ? a[cur - 1] + 1 : 0;
        for (int i = s; i < n; i++)
        {
                a[cur] = i;
                sub_set(n, a, cur + 1);
        }
}
int main()
{
        int n;
        cin >> n;
        int* a = new int[n];
        sub_set(n, a, 0);

}
```

方法二：位向量法

```
#include<iostream>
using namespace std;
void sub_set(int n, int* a, int cur)
{
        if (cur == n)
        {
                cout << '{';
                for (int i = 0; i < n; i++)
                        if (a[i])cout << i << ' ';
                cout << '}';
                cout << endl;
                return;
        }
        a[cur] = 1;
        sub_set(n, a, cur + 1);
        a[cur] = 0;
        sub_set(n, a, cur + 1);
}
int main()
{
        int n;
        cin >> n;
        int* a = new int[n];
```

```
	sub_set(n, a, 0);

}
```

方法三：二进制法（集合元素过多时会越界）

```
#include<iostream>
using namespace std;
void sub_set(int n, int s)
{
	for (int i = 0; i < n; i++)
		if (s & (1 << i))cout << i << ' ';//按位比较元素
	cout << endl;
}
int main()
{
	int n;
	cin >> n;
	int* a = new int[n];
	for (int i = 0; i < (1 << n); i++)
		sub_set(n, i);//枚举各子集对应的二进制编码

}
```

弦长公式

2022年3月7日　　23:15

弦长＝2Rsina，R是半径，a是圆心角；弦长为连接圆上任意两点的线段的长度。

来自 <https://zhidao.baidu.com/question/755914039830903164.html>

gcd 性质

2021年11月15日 9:26

gcd(x,y)=gcd(x-y,y);

gcd(a)=gcd(a/b)*gcd(b)//a可以被b整除
=>gcd(a/b)=gcd(a)/gcd(b)//a和b可以代表集合

二维偏序

2021年11月15日 16:34

https://zhuanlan.zhihu.com/p/112504092

例题： https://ac.nowcoder.com/acm/contest/24115/D

```
#include<bits/stdc++.h>
#define ll long long
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define FOR(i,a,b) for(ll i=(a);i<=(b);i++)
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f
using namespace std;
const ll N=5e5+5;
const ll M=1e3+5;
const ll mod=1e9+7;
ll n;
ll v[4][N];
template<typename T>
struct BIT{
    const int n;
    vector<T>a;
    BIT(int n):n(n),a(n+1){}
    void update(int x,T k)
    {
        for(int i=x;i<=n;i+=(i&(-i)))
        {
            a[i]+=k;
        }
    }
    T sum(int x)//求[1,x]的和
    {
        T res=0;
        for(int i=x;i>0;i-=(i&(-i)))
        {
            res+=a[i];
        }
        return res;
    }
    T rangeSum(int l,int r)//求[l,r]的和
    {
        return sum(r)-sum(l-1);
    }

};
int main()
{
    IOS;
    cin>>n;
    for(int i=1;i<=3;i++)
    {
        for(int j=1;j<=n;j++)
        {
            cin>>v[i][j];
            v[i][j]+=v[i][j-1];
        }
    }
    vector<ll>arr;
    for(int i=1;i<=n;i++)
    {
        arr.push_back(v[2][i]-v[3][i-1]);
        arr.push_back(v[2][i-1]-v[1][i]-v[3][n]);
    }
    sort(arr.begin(),arr.end());
    arr.erase(unique(arr.begin(),arr.end()),arr.end());
    BIT<int> tree(arr.size());
    ll ans=0;
    for(int i=1;i<=n;i++)
    {
        ll j=lower_bound(arr.begin(),arr.end(),v[2][i-1]-v[1][i]-v[3][n])-arr.begin()+1;
        tree.update(j,1);
        j=lower_bound(arr.begin(),arr.end(),v[2][i]-v[3][i-1])-arr.begin()+1;
        ans=(ans+tree.sum(j))%mod;
    }
    cout<<ans<<endl;

}
```

贝祖定理

2021年1月29日 18:31

若a, b是整数, 且gcd(a, b)=d，那么对于任意的整数x, y, ax+by都一定是d的倍数，特别地，一定存在整数x, y，使ax+by=d成立。

它的一个重要推论是：a, b互质的充分必要条件是存在整数x, y使ax+by=1.

欧拉函数（待补）

2021年1月27日　　14:16

https://blog.csdn.net/weixin_43237242/article/details/97388834

容斥原理

2021年1月27日　　14:17

假设有n个集合，记每个集合为a[i]，那么这些集合的总元素为：

每个集合-每两个集合交+每三个集合交-...+(-1)^(n-1)每n个集合交

比如两个集合时：a[1]+a[2]-a[1]^a[2]

三个集合：a[1]+a[2]+a[3]-a[1]^a[2]-a[1]^a[3]-a[2]^a[3]

四个集合：a[1]+a[2]+a[3]+a[4]-a[1]^a[2]-a[1]^a[3]-a[1]^a[4]-a[2]^a[3]-a[2]^a[4]-a[3]^a[4]
+a[1]^a[2]^a[3]+a[1]^a[2]^a[4]+a[2]^a[3]^a[4]-a[1]^a[2]^a[3]^a[4]

总的元素数为2^n-1=c(n,1)+c(n,2)+...+c(n,n)

Sample: https://www.acwing.com/problem/content/892/

```
#include<iostream>
#include<algorithm>

using namespace std;

#define ll long long

const ll N=1e5+5;
ll p[N];
int main()
{
  ll n,m;
  cin>>n>>m;
  for(ll i=0;i<m;i++)
  {
    cin>>p[i];
  }
  ll ans=0;
  for(ll i = 1;i < 1 << m;i++)
  {
    ll t=1,cnt=0;
    for(ll j=0;j<m;j++)
    {
      if(i>>j&1)
      {
        cnt++;
        t*=p[j];
        if(t>n)
        {
          t=-1;
          break;
        }
      }
    }
    if(t!=-1)
    {
      if(cnt&1)
      {
        ans+=n/t;
      }
```

```
            else
            {
                ans-=n/t;
            }
        }
    }
    cout<<ans<<endl;
}
```

唯一分解定理

2021年2月1日 16:26

https://www.cnblogs.com/theshorekind/p/12704405.html
对于任何一个大于1的正整数,都存在一个标准的分解式: N=p1^a1 * p2^a2*···*pn^an;（其中一系列an为指数，pn为质数）

来自 <https://www.cnblogs.com/theshorekind/p/12704405.html>

树状数组

2021年2月3日 14:57

https://www.cnblogs.com/xenny/p/9739600.html
https://blog.csdn.net/bestsort/article/details/80796531

模板：

```
template<typename T>
struct BIT{
   const int n;
   vector<T>a;
   BIT(int n):n(n),a(n+1){}
   void update(int x,T k)
   {
      for(int i=x;i<=n;i+=(i&(-i)))
      {
         a[i]+=k;
      }
   }
   T sum(int x)//求[1,x]的和
   {
      T res=0;
      for(int i=x;i>0;i-=(i&(-i)))
      {
         res+=a[i];
      }
      return res;
   }
   T rangeSum(int l,int r)//求[l,r]的和
   {
      return sum(r)-sum(l-1);
   }

};
```

单点更新与区间查询：

洛谷3374： https://www.luogu.com.cn/problem/P3374

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 5e5 + 5;
int a[maxn], c[maxn];
int n,m;
int lowbit(int i)
{
        return i & (-i);//lowbit的作用是返回i二进制表示的最后一位1的位置的二进制表示
                          //i&(-i)=i&(~i+1)，因为负数以补码存储
}
void update(int i, int k)//在i的位置加上k
{
        while (i <= n)
```

```
	{
		c[i] += k;
		i += lowbit(i);
	}
}
int getsum(int i)//求[1,i]的区间和
{
	int res = 0;
	while (i > 0)
	{
		res += c[i];
		i -= lowbit(i);
	}
	return res;
}
int main()
{
	cin >> n>>m;
	for (int i = 1; i <= n; i++)//注意树状数组下标不能从0开始，因为0的lowbit永远是0
	{
		cin >> a[i];
		update(i, a[i]);
	}
	for (int i = 1; i <= m; i++)
	{
		int k, x, y;
		cin >> k >> x >> y;
		if (k == 1)//单点更新
		{
			update(x, y);
		}
		else//区间查询
		{
			cout << getsum(y) - getsum(x-1) << endl;
		}
	}
}
```

区间更新与单点查询:

洛谷3368： https://www.luogu.com.cn/problem/P3368

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 5e5 + 5;
int a[maxn], c[maxn];
int n, m;
int lowbit(int i)
{
	return i & (-i);
}
void update(int i, int k)//在i的位置加上k
{
	while (i <= n)
	{
		c[i] += k;
		i += lowbit(i);
```

```
        }
}
int getsum(int i)//求[1,i]的区间和
{
        int res = 0;
        while (i > 0)
        {
                res += c[i];
                i -= lowbit(i);
        }
        return res;
}
int main()
{
        cin >> n >> m;
        for (int i = 1; i <= n; i++)//注意树状数组下标不能从0开始，因为0的lowbit永远是0
        {
                cin >> a[i];
                update(i, a[i]-a[i-1]);//差分建树状数组
        }
        for (int i = 1; i <= m; i++)
        {
                int k;
                cin >> k;
                if (k == 1)//区间更新
                {
                        int x, y, v;//将[x,y]内的值加v
                        cin >> x >> y >> v;
                        update(x, v);
                        update(y + 1, -v);
                }
                else//单点查询
                {
                        int x;
                        cin >> x;//查询第x个节点的值
                        cout << getsum(x) << endl;
                }
        }
}
```

区间更新与区间查询：

Poj3468: http://poj.org/problem?id=3468

```
#include<iostream>
using namespace std;
#define int long long
const int maxn = 5e5 + 5;
int a[maxn], c1[maxn],c2[maxn];
int n, m;
int lowbit(int i)
{
        return i & (-i);
}
void update(int i, int k)//在i的位置加上k
{
	int x = i;
	while (i <= n)
	{
		c1[i] += k;
		c2[i] += (x - 1) * k;
		i += lowbit(i);
	}
}
int getsum(int i)//求[1,i]的区间和
{
	int res = 0;
	int x = i;
	while (i > 0)
	{
		res += x*c1[i]-c2[i];
		i -= lowbit(i);
	}
	return res;
}//修改与求和的部分对比求和公式会更好理解
signed main()
{
	ios::sync_with_stdio(0);
	cin.tie(0);
	cin >> n >> m;
	for (int i = 1; i <= n; i++)//注意树状数组下标不能从0开始，因为0的lowbit永远是0
	{
		cin >> a[i];
		update(i, a[i] - a[i - 1]);//差分建树状数组
	}
	for (int i = 1; i <= m; i++)
	{
		char k;
		cin >> k;
		if (k == 'C')//区间更新
		{
			int x, y, v;//将[x,y]内的值加v
			cin >> x >> y >> v;
			update(x, v);
			update(y + 1, -v);
		}
		else//区间查询
		{
			int x,y;
			cin >> x>>y;//查询[x,y]的区间和
			cout << getsum(y)-getsum(x-1) << endl;
		}
	}
}
```

https://codeforces.com/problemset/problem/1579/E2

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
#define int long long
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
```

```
#define pii pair<int,int>
#define ll long long
using namespace std;
const int N=2e5+5;
const int mod=1e9+7;
int n;
int a[N];
template<typename T>
struct BIT{
   const int n;
   vector<T>a;
   BIT(int n):n(n),a(n+1){}
   void update(int x,T k)
   {
      for(int i=x;i<=n;i+=(i&(-i)))
      {
         a[i]+=k;
      }
   }
   T sum(int x)//求[1,x]的和
   {
      T res=0;
      for(int i=x;i>0;i-=(i&(-i)))
      {
         res+=a[i];
      }
      return res;
   }
   T rangeSum(int l,int r)//求[l,r]的和
   {
      return sum(r)-sum(l-1);
   }

};
void solve()
{
   cin>>n;
   for(int i=0;i<n;i++)
   {
      cin>>a[i];
   }

   int v[n];
   copy(a,a+n,v);
   sort(v,v+n);

   int m=unique(v,v+n)-v;
   BIT<int> f(m);

   int ans=0;
   for(int i=0;i<n;i++)
   {
      int t=lower_bound(v,v+m,a[i])-v;
      t++;
      f.update(t,1);
      ans+=min(f.sum(t-1),f.rangeSum(t+1,m));
```

```
    }
    cout<<ans<<endl;


}
signed main()
{
    IOS;
    int t;
    cin>>t;
    while(t--)
    {
        solve();
    }
}
```

悬线法求最大子矩阵

2022年4月3日　14:45

https://www.acwing.com/problem/content/description/154/

```
#include<bits/stdc++.h>
#define ll int
#define ull unsigned long long
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define debug(x) cout<< #x <<" is "<< x <<endl
#define endl '\n'
#define pb push_back
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f3f3f3f3f
#define ld long double
using namespace std;
const ll N=2e6+5;
const ll M=2e3+5;
const ll mod=1e9+7;
ll n,m;
ll mp[M][M];
ll h[M][M],l[M][M],r[M][M];
int main()
{
  IOS;

  cin>>n>>m;
  for(ll i=1; i<=n; i++)
  {
    for(ll j=1; j<=m; j++)
    {
      char ch;
      cin>>ch;
      if(ch=='F')mp[i][j]=1;
      else mp[i][j]=0;
    }
  }
  ll ans=0;
  for(ll i=0; i<=m; i++)
  {
    l[0][i]=1;
    r[0][i]=m;

  }
  for(ll i=1; i<=n; i++)
  {
    for(ll j=1; j<=m; j++)
    {
      if(mp[i][j]==0)
      {
        h[i][j]=0;
        l[i][j]=j;
        r[i][j]=j;
```

```
                continue;
            }
            if(mp[i-1][j]==0)
            {
                h[i][j]=1;
                ll nl=j,nr=j;
                while(mp[i][nl]==1&&nl>=1)nl--;
                while(mp[i][nr]==1&&nr<=m)nr++;
                l[i][j]=nl+1;
                r[i][j]=nr-1;
            }
            else
            {
                h[i][j]=h[i-1][j]+1;
                ll nl=j,nr=j;
                while(mp[i][nl]==1&&nl>=1)nl--;
                while(mp[i][nr]==1&&nr<=m)nr++;
                l[i][j]=max(l[i-1][j],nl+1);
                r[i][j]=min(r[i-1][j],nr-1);
            }


            ans=max(ans,h[i][j]*(r[i][j]-l[i][j]+1));
        }
    }
    cout<<ans*3<<endl;

}
```

二维计算几何

2022年4月3日 23:16

```
#include<bits/stdc++.h>
using namespace std;
struct Point{
   double x,y;
   Point(double x=0,double y=0):x(x),y(y){}
};
typedef Point Vector;//从程序实现上说，Vector只是Point的别名
//向量+向量=向量，点+向量=点
Vector operator+(Vector  A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
//点-点=向量
Vector operator-(Vector  A,Vector B){return Vector(A.x-B.x,A.y-B.y);}
//向量*数=向量
Vector operator*(Vector A,double p){return Vector(A.x*p,A.y*p);}
//向量/数=向量
Vector operator/(Vector A,double p){return Vector(A.x/p,A.y/p);}

bool operator<(const Point&a,const Point&b){
   return a.x<b.x||(a.x==b.x&&a.y<b.y);
}

const double eps=1e-10;
int dcmp(double x)
{
   if(fabs(x)<eps)return 0;else return x<0?-1:1;
}
bool operator==(const Point&a,const Point&b)
{
   return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
}
//C标准库中的atan2函数可以用来求向量极角，如向量(x,y)的极角就是atan2(y,x),单位是弧度
//点积等于两向量模长相乘再乘以二者夹角的余弦值
double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return acos(Dot(A,B)/Length(A)/Length(B));}
//两向量的叉积等于两向量组成的三角性的有向面积的两倍
//cross(v,w)>0,当w在v左边；cross(v,w)<0，当w在v右边
double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;}
double Area2(Point A,Point B,Point C){return Cross(B-A,C-A);}
/*使用点积和叉积可以判断两个向量的相对位置
假设有两向量v和w，以v作为坐标系的x轴正方向，那么对于不同的w位置，点积和叉积的符号是：
（以下第一个符号是点积，第二个符号是叉积)
x轴正方向:(+,0)
第一象限：(+,+)
```

```
y轴正方向：(0,+)
第二象限：(-,+)
x轴负方向：(-,0)
第三象限：(-,-)
y轴负方向：(0,-)
第四象限：(+,-)
*/
//将向量逆时针旋转rad度（弧度）
Vector Rotate(Vector A,double rad)
{
   return Vector(A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
}
//计算向量的单位法线，即左转九十度后将长度归一化
//注意该向量不能是零向量
//另外，若两向量平行，则叉积为0；若两向量垂直，则点积为0
Vector Normal(Vector A)
{
   double L=Length(A);
   return Vector(-A.y/L,A.x/L);
}
//直线的参数方程:
//已知直线上一点P0和方向向量v，则直线上所有点满足P=P0+t*v
//射线和线段的方程形式也是一致的
//求P+tv和Q+tw两条直线的交点，使用前需确定两直线不平行
Point GetLineIntersection(Point P,Vector v,Point Q,Vector w)
{
   Vector u=P-Q;
   double t=Cross(w,u)/Cross(v,w);
   return P+v*t;
}
//点P到直线AB距离，用平行四边形面积除以底
double DistanceToLine(Point P,Point A,Point B)
{
   Vector v1=B-A,v2=P-A;
   return fabs(Cross(v1,v2)/Length(v1));
}
//点到线段距离，注意不是到投影点的距离
double DistanceToSegment(Point P,Point A,Point B)
{
   if(A==B)return Length(P-A);
   Vector v1=B-A,v2=P-A,v3=P-B;
   if(dcmp(Dot(v1,v2))<0)return Length(v2);
   else if(dcmp(Dot(v1,v3))>0)return Length(v3);
   else return fabs(Cross(v1,v2))/Length(v1);
}
//求点P到直线AB的投影点
Point GetLineProjection(Point P,Point A,Point B)
{
   Vector v=B-A;
   return A+v*(Dot(v,P-A)/Dot(v,v));
}
//判断线段规范相交(相交点不允许是端点)
bool SegmentProperIntersection(Point a1,Point a2,Point b1,Point b2)
{
   double c1=Cross(a2-a1,b1-a1),c2=Cross(a2-a1,b2-a1),
            c3=Cross(b2-b1,a1-b1),c4=Cross(b2-b1,a2-b1);
   return dcmp(c1)*dcmp(c2)<0&&dcmp(c3)*dcmp(c4)<0;
}
/*
如果允许两线段在端点相交，那么：
如果c1和c2均为0，那么两线段共线，可能有部分重叠；
如果不都为0，那么唯一的相交办法就是其中一条线段的端点在另一条线段上
*/
//判断一个点是否在一条线段上（不含端点）
bool OnSegment(Point P,Point a1,Point a2)
{
   return dcmp(Cross(a1-P,a2-P))==0&&dcmp(Dot(a1-P,a2-P))<0;
}
/*n个顶点多边形的有向面积，将其划分为n-2个三角形即可（如果多边形非凸，那么多余的面积会正负抵消）
*/
double ConvexPolygonArea(Point *p,int n)
{
   double area=0;
   for(int i=1;i<n-1;i++)
   {
      area+=Cross(p[i]-p[0],p[i+1]-p[0]);
   }
   return area/2;
}
```

分块（待补）

2021年2月4日 14:26

逆序数

2021年2月4日 13:42

树状数组求逆序对：
思路：将原数组的元素一一插入现数组，每插入一个元素，就去现数组寻找比它大的元素个数

C[i]为元素i出现的次数（树状数组维护），每插入一个元素a[i]都会递增c[i]，接着，由于这里的树状数组是用元素的值来进行划分的，故而对每个元素a[i],只需对c数组求sum(1,a[i]),即可求出值小于等于a[i]的元素个数，用当前插入的元素个数减去这个值，即可得到大于a[i]的元素个数。

优化：由于树状数组用值来划分，所以可能会导致数组开得很大但是有许多空间没有使用造成浪费，所以考虑离散化。
设数组a[1]=3,a[2]=2,a[3]=1,a[4]=5,a[5]=4,其逆序数为4
对其根据大小排序后得到：a[3]=1,a[2]=2,a[1]=3,a[5]=4,a[4]=5，观察其下标3,2,1,5,4发现逆序数也为4；
故有结论：离散化并排序后的数组，其下标逆序数与原数组逆序数相同

```
#include<iostream>
#include<algorithm>
using namespace std;
using ll = long long;
const ll maxn=5e5+5;
ll n;
ll c[maxn];
struct num {
        ll index;
        ll val;
}a[maxn];
ll lowbit(ll i)
{
        return i & (-i);
}
void update(ll i)
{
        while (i <= n)
        {
                c[i]++;
                i += lowbit(i);
        }
}
ll sum(ll i)
{
        ll res = 0;
        while (i)
        {
```

```
			res += c[i];
			i -= lowbit(i);
		}
		return res;
}
int cmp(num a, num b)
{
		if (a.val == b.val)return a.index < b.index;//两个元素值相同时，标号应保持升序以防止出现
		新逆序对
		return a.val < b.val;
}
int main()
{
		ios::sync_with_stdio(0);
		cin.tie(0);
		cin >> n;
		for (ll i = 1; i <= n; i++)
		{
			cin >> a[i].val;
			a[i].index = i;
		}
		sort(a + 1, a + n + 1, cmp);
		ll ans = 0;
		for (ll i = 1; i <= n; i++)
		{
			update(a[i].index);
			ans += i - sum(a[i].index);
		}
		cout << ans << endl;
}
```

归并排序求逆序对：

```
#include<iostream>
#define ll long long
using namespace std;
const ll N=1e6+5;
ll p[N],tmp[N];
ll n;
ll merge_sort(ll l,ll r)
{
   if(l>=r)return 0;
   ll res=0;
   ll mid=l+r>>1;
   res+=merge_sort(l,mid)+merge_sort(mid+1,r);
   ll i=l,j=mid+1;
   ll k=0;
   while(i<=mid&&j<=r)
   {
      if(p[i]<=p[j])tmp[k++]=p[i++];
      else
      {
         tmp[k++]=p[j++];
         res+=mid-i+1;
      }
   }
```

```
    while(i<=mid)tmp[k++]=p[i++];
    while(j<=r)tmp[k++]=p[j++];
    for(ll i=l,j=0;i<=r;i++,j++)p[i]=tmp[j];
    return res;
}
int main()
{
    scanf("%d",&n);
    for(ll i=0;i<n;i++)
    {
        scanf("%d",&p[i]);
    }
    printf("%lld",merge_sort(0,n-1));
}
```

矩阵快速幂

2022年2月18日　　20:43

https://www.acwing.com/problem/content/1219/

```
#include<bits/stdc++.h>
#define ll long long
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f3f3f3f3f
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll mod=1e9+7;
const ll N=2e5+5;
const ll M=1e3+5;
ll n,m;
bool st[50][50];
ll op[]={0,4,5,6,1,2,3};
ll mul(ll a,ll b)
{
    return ((a%mod)*(b%mod))%mod;
}
ll add(ll a,ll b)
{
    return ((a%mod)+(b%mod))%mod;
}
ll mul(ll res[][50],ll a[][50],ll b[][50])
{
    ll tmp[50][50]={0};
    for(ll i=1;i<=6;i++)
    {
        for(ll j=1;j<=6;j++)
        {
            for(ll k=1;k<=6;k++)
            {
                tmp[i][j]+=(a[i][k]*b[k][j])%mod;
                tmp[i][j]%=mod;
            }
        }
    }
    memcpy(res,tmp,sizeof tmp);
}
int main()
{
    cin>>n>>m;
    for(ll i=1;i<=m;i++)
    {
        ll a,b;
        cin>>a>>b;
        st[a][b]=st[b][a]=1;
    }
    ll f[50][50];
    ll a[50][50];
    for(ll i=1;i<=6;i++)
    {
        f[1][i]=4;
    }
    for(ll i=1;i<=6;i++)
    {
        for(ll j=1;j<=6;j++)
        {
            if(st[i][op[j]])a[i][j]=0;
            else a[i][j]=4;
        }
    }

    ll power=n-1;
    while(power)
    {
        if(power&1)
        {
            mul(f,f,a);
        }
        power>>=1;
        mul(a,a,a);
    }
    ll ans=0;
    for(ll i=1;i<=6;i++)
    {
        ans+=f[1][i];
        ans%=mod;
    }
    cout<<ans<<endl;
}
```

gcd

2022年2月20日 2:26

更相减损法：

```
int gcd(int a,int b)
{
   if(a==b)
      return a;
   if(a>b)
      return gcd(a-b,b);
   if(a<b)
      return gcd(b-a,a);
}
```

辗转相除法：

```
int gcd(int a,int b)
{
   if(b==0)
      return a;
   else
      return gcd(b,a%b);
}
```

利用改版的更相减损法，我们可以求出在序列：

p^k1,p^k2,...,p^kn中的最大p值（也就是取所有指数的最大公约数k，得到p^k）

https://www.acwing.com/problem/content/1225/

```
#include<bits/stdc++.h>
#define ll long long
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f3f3f3f3f
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll mod=1e9+7;
const ll N=2e5+5;
const ll M=1e3+5;
ll n;
vector<ll>x;
int check(vector<ll>arr,ll num)
{
   cout<<"num=="<<num<<endl;
   if(num==1)return 0;
   for(auto i:arr)
   {
      while(i%num==0&&i>0)
      {
         i/=num;
      }
      if(i!=1)return 0;
   }
   return 1;
}
ll gcd_sub(ll a,ll b)
{
   if(b>a) swap(a,b);
   if(b==1) return a;
```

```
    return gcd_sub(b,a/b);
}
int main()
{
    IOS;
    cin>>n;
    for(ll i=1;i<=n;i++)
    {
        ll a;
        cin>>a;
        x.push_back(a);
    }
    sort(x.begin(),x.end());
    x.erase(unique(x.begin(),x.end()),x.end());
    vector<ll>mo,son;
    for(ll i=0;i<x.size()-1;i++)
    {
        ll g=__gcd(x[i],x[i+1]);
        son.push_back(x[i+1]/g);
        mo.push_back(x[i]/g);
    }
    sort(son.begin(),son.end());
    sort(mo.begin(),mo.end());

    ll t1=son[0];
    for(ll i=1;i<son.size();i++)
    {
        t1=gcd_sub(t1,son[i]);
    }
    ll t2=mo[0];
    for(ll i=1;i<mo.size();i++)
    {
        t2=gcd_sub(t2,mo[i]);
    }
    ll g=__gcd(t1,t2);
    cout<<t1/g<<'/'<<t2/g<<endl;

}
```

同余定理

2021年8月13日 3:54

给定一个正整数m，如果两个整数a和b满足a-b能够被m整除，即(a-b)/m得到一个整数，那么就称整数a与b对模m同余，记作a≡b(mod m)。

来自 <https://baike.baidu.com/item/%E5%90%8C%E4%BD%99%E5%AE%9A%E7%90%86/1212360>

STL应用

2021年7月12日 3:46

https://www.luogu.com.cn/problem/P2234

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cmath>
#include<set>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-4;
ll n;
ll a[N];
void solve()
{
    cin>>n;
    for (ll i = 1; i <= n; i++)
    {
        cin>>a[i];
    }
    ll ans=0;
    multiset<ll>q;
    q.insert(-INF);
    q.insert(INF);
    for(ll i=1;i<=n;i++)
    {
        multiset<ll>::iterator l=q.insert(a[i]);//multiset支持返回插入位置，注意set不
支持
        l--;
        ll res=abs(*l-a[i]);
        l++;
        l++;
        res=min(res,abs(*(l)-a[i]));
        if(i==1)res=a[1];
        ans+=res;
    }
    cout<<ans<<endl;
}
int main()
{
    IOS;
    solve();

}
```

搜索

2021年7月12日 17:11

https://codeforces.com/contest/1547/problem/G

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cmath>
#include<set>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 4e5 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-4;
ll n,m;
vector<ll>g[N];
int color[N];
bool circle[N];
bool mul[N];
int ans[N];
bool vis[N];
void dfs_color(ll cur)
{
    color[cur]=1;
    for(auto i:g[cur])
    {
        if(color[i]==1)//若当前节点被第二次访问时其颜色是1，说明它是一个带环节点
        {
            circle[i]=1;
        }
        else if(color[i]==2)//若当前节点被第二次访问时其颜色是2，说明它是一个多路
径节点
        {
            mul[i]=1;
        }
        else if(color[i]==0)
        {
            dfs_color(i);
        }
    }
    color[cur]=2;
}
void dfs_mul(ll cur, int m)
{
    vis[cur]=1;
    if (mul[cur] == 1)
    {
        m = 1;
    }
    if(m==1)
    {
        ans[cur]=2;
    }

    //cout<<"cur is "<<cur<<" cir=="<<cir<<" m=="<<m<<" ans=="<<ans[cur]
<<" color=="<<color[cur]<<endl;
    for(auto i:g[cur])
    {
        if(!vis[i]){
            dfs_mul(i,m);
        }
    }
}
void dfs_cir(ll cur, ll cir)
{
    vis[cur]=1;
    if (circle[cur] == 1)
    {
        cir = 1;
    }
    if(cir==1)
    {
        ans[cur]=-1;
    }

    for(auto i:g[cur])
    {
        if(!vis[i]){
            dfs_cir(i,cir);
        }
    }
}
void dfs_reachable(ll cur)
{
    vis[cur]=1;
    ans[cur]=1;
    for(auto i:g[cur])
    {
        if(!vis[i]){
            dfs_reachable(i);
        }
    }
}
void solve()
```

```
{
    cin>>n>>m;
    for(ll i=1;i<=n;i++)
    {
        g[i].clear();
        color[i]=0;//每个节点的颜色，0为未访问，1为访问过但没遍历其所有后继节点，2
为完全访问了其所有后继节点
        circle[i]=0;//该节点是不是环路节点
        mul[i]=0;//该节点是不是多路径节点
        ans[i]=0;
        vis[i]=0;
    }
    for(ll i=0;i<m;i++)
    {
        ll a,b;
        cin>>a>>b;
        g[a].push_back(b);
    }
    dfs_color(1);
    memset(vis,0,sizeof vis);
    dfs_reachable(1);//检验每个节点的可达性
    /*为什么不把可达性、多路径、环放在一个dfs里检验呢?
    这是因为放在一个里面可能导致后面的结果覆盖前面的结果，而我们发现环的优先级大于
多路径优先级，多路径优先级又大于单路径优先级
    所以分为三个dfs，优先级大的就可以覆盖优先级小的，而不会造成混乱
    */
    memset(vis,0,sizeof vis);
    for (ll i = 1; i <= n; i++)
    {
        if (mul[i] == 1)
        {
            dfs_mul(i, mul[i]);//找出所有多路径节点的可达节点
        }
    }
    memset(vis,0,sizeof vis);
    for (ll i = n; i >=1; i--)
    {
        if (circle[i] == 1)
        {
            dfs_cir(i,circle[i]);//找出所有带环节点的可达节点
        }
    }

    for(ll i=1;i<=n;i++)
    {
        cout<<ans[i]<<' ';
    }
    cout<<endl;

}
int main()
{
    IOS;
    ll _;
    cin>>_;
    while(_--)
    solve();

}
```

https://www.luogu.com.cn/problem/P3916

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll int
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=1e6+5;
ll n,m;
vector<ll>g[N];
ll ans[N];
int st[N];
void dfs(ll cur,ll node)
{
    st[cur]=1;
    ans[cur]=max(ans[cur],node);
    for(auto i:g[cur])
    {
        if(!st[i])
        {
            st[i]=1;
            dfs(i,node);
        }
    }
}
int main()
{
    IOS;
    cin>>n>>m;
    while(m--)
    {
        ll u,v;
        cin>>u>>v;
        g[v].push_back(u);//反向建图，看每个点能到达的有哪些点
    }
    for(ll i=1;i<=n;i++)
    {
        ans[i]=i;
    }
    for(ll i=n;i>=1;i--)
    {
        dfs(i,i);//只需要遍历上次dfs没访问过的那些点就行
    }
    for(ll i=1;i<=n;i++)
    {
        cout<<ans[i]<<' ';
    }
    cout<<endl;

}
```

https://www.luogu.com.cn/problem/P1127

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=2e3+5;
string s[N];
vector<ll>g[N];
ll pre[N];
ll pos[N];
int vis[N];
ll n;
string ans;
int fa;
void dfs(ll cur,ll cnt,string res)
{
    if(fa)return;
    if(cnt==n)
    {
        ans=res;
        fa=1;
        return;
    }
    for(ll i=0;i<n;i++)
    {
        if(!vis[i]&&s[cur][s[cur].length()-1]==s[i][0])
        {
            vis[i]=1;
            string tmp=res;
            tmp+=".";
```

```
                tmp+=s[i];
                dfs(i,cnt+1,tmp);
                vis[i]=0;
            }
        }
}
int main()
{
    IOS;
    cin>>n;
    for(ll i=0;i<n;i++)
    {
        cin>>s[i];
        pre[s[i][0]]++;//记录开头字符的在开头的出现次数
        pos[s[i][s[i].length()-1]]++;//记录结尾字符在结尾的出现次数
    }
    sort(s,s+n);//对字符串排序以满足字典序最小
    int f=0;
    for(ll i=0;i<n;i++)
    {
        if(pre[s[i][0]]==pos[s[i][0]]+1)//如果当前字符串的开头字符在开头出现次数等
于在结尾出现次数加一，那么当前串一定可以作为开头
        {
            vis[i]=1;
            string t=s[i];
            dfs(i,1,t);
            f=1;
           if(fa)//找到答案就跳出
           {
               break;
           }
           else{//没找到就清空状态
               memset(vis,0,sizeof vis);
               f=0;
           }
        }
    }
    if(!f)//如果没有找到满足条件的开头串，从第一个串开始搜
    {
        string t=s[0];
        vis[0]=1;
        dfs(0,1,t);
    }
    if(!fa)
    {
        cout<<"***"<<endl;
    }
    else{
        cout<<ans<<endl;
    }
}
```

拓扑问题

2021年7月23日　　2:51

https://www.luogu.com.cn/problem/P1983

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll int
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=2e3+5;
/*
本题中，对于每一条线路s1->s2->s3->...->sn，[s1,sn]内所有非停靠站点的等级都小于停靠站点的等级
那么，对每一对节点(u,v)，若节点u的等级大于节点v，就由u向v引一条有向边；形成一张图后对其做拓扑排序，然后看这张图有多少层即可
另外，在多对多关系中，可以用超级源汇点的思想优化图结构
*/
bool vis[N];
vector<ll>g[N];
ll have[N][N];//用来判断重边，不判断会空间超限
ll ans[N];
ll deg[N];
ll se[N];
ll n,m;
void topo_sort()
{
    queue<ll>q;
    for(ll i=1;i<=n;i++)
    {
        if(deg[i]==0)
        {
            q.push(i);
            ans[i]=1;
        }
    }
    while(!q.empty())
    {
        ll t=q.front();
        q.pop();
```

```
            for(auto i:g[t])
            {
                deg[i]--;
                if(deg[i]==0)
                {
                    if(i>n)
                    {
                        ans[i]=ans[t];//如果是超级源汇点，答案不用增加
                    }
                    else{
                        ans[i]=ans[t]+1;
                    }
                    q.push(i);
                }
            }
        }
    }
    int main()
    {
        IOS;
        cin>>n>>m;
        ll idx=n+5;
       while(m--)
       {
           ll s;
           cin>>s;
           memset(vis,0,sizeof vis);
           for(ll i=0;i<s;i++)
           {
               cin>>se[i];
               vis[se[i]]=1;
           }
           for(ll i=0;i<s;i++)
           {
               g[se[i]].push_back(idx);//idx为超级源汇点
               deg[idx]++;
           }
           for (ll j = se[0]; j <= se[s - 1]; j++)
           {
               if (!vis[j] && !have[idx][j])
               {
                   have[idx][j] = 1;
                   g[idx].push_back(j);
                   deg[j]++;
               }
           }
           idx++;
        }
        topo_sort();
        ll mm=0;
        for(ll i=1;i<=n;i++)
        {
            mm=max(mm,ans[i]);
        }
        cout<<mm<<endl;
```

```
}
```

https://www.luogu.com.cn/problem/P1347

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=2e3+5;
vector<ll>g[N];
ll n,m;
ll idx=1;
ll vis[N];
ll deg[N];
ll d[N];
int f=0;
ll step=0;
vector<ll>ans;
ll lay[N];
void topo_sort()
{
    memcpy(d,deg,sizeof deg);
    memset(vis,0,sizeof vis);
    queue<ll>q;
    for(ll i=0;i<n;i++)
    {
        if(deg[i]==0)
        {
            q.push(i);
            lay[i]=1;
        }
    }
    while(!q.empty())
    {

        ll t=q.front();
        ans.push_back(t);
        vis[t]=1;
        q.pop();
        int cnt=0;
        for(auto i:g[t])
        {
            d[i]--;
            if(d[i]==0)
            {
                q.push(i);
                lay[i]=lay[t]+1;
                cnt++;
            }
        }
    }
    ll mm=0;
    for(ll i=0;i<n;i++)
    {
        mm=max(mm,lay[i]);
        if(!vis[i])//如果有边没访问到，说明存在环
        {
            f=1;
            ans.clear();
            return;
        }
    }
    if(mm!=n)//如果排序后的层数不为n，说明无法确定
    {
        ans.clear();
        f=0;
        return;
    }
    f=2;
    return;
}
int main()
{
   IOS;
   cin>>n>>m;
   while(m--)
   {
       string s;
       cin>>s;
       ll a=(ll)(s[0]-'A');
       ll b=(ll)(s[2]-'A');
       g[a].push_back(b);
       deg[b]++;
       if(!f){
        step++;
        topo_sort();
       }
   }
   if(f==0)
   {
       cout<<"Sorted sequence cannot be determined."<<endl;

   }
   else if(f==1)
   {
       cout<<"Inconsistency found after "<<step<<" relations."<<endl;
   }
   else if(f==2)
   {
       cout<<"Sorted sequence determined after "<<step<<" relations: ";
       for(auto i:ans)
       {
           cout<<char(i+'A');
       }
       cout<<".";
   }

}
```

前缀和

2021年7月15日 13:17

https://www.luogu.com.cn/problem/P2671

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include <string>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e5 + 5;
const int mod = 10007;
const double eps = 1e-4;
ll n,m;
struct node{
    ll num,col,ind;
}mem[N];
ll sum[N][2];
ll cnt[N][2];
/*
由于y-x=z-y => 2*y=x+z
所以只要x与z同为奇数或者同为偶数，就一定满足条件
将所有元素按颜色分组，每种颜色内再按奇数偶数分组
那么假设一组内元素的下标为y[i]，值为x[i]，组内有k个元素
那么在一组内，答案为:
ans=(x1+x2)(y1+y2)+(x1+x3)(y1+y3)+...+(x1+xk)(y1+yk)
+(x2+x3)(y2+y3)+(x2+x4)(y2+y4)+...+(x2+xk)(y2+yk)
+...
+(x[k-1]+xk)(y[k-1]+yk)
展开后为
x1((k-2)y1+sum(y))
+x2((k-2)y2+sum(y))
+...
```

```
+xk((k-2)yk+sum(y))
预处理每一组的sum(y)，就可以在O(n)时间内求出答案
*/
int main()
{
    IOS;
    cin>>n>>m;
    for(ll i=1;i<=n;i++)
    {
        cin>>mem[i].num;
        mem[i].ind=i;
    }
    for(ll i=1;i<=n;i++)
    {
        cin>>mem[i].col;
    }
    for(ll i=1;i<=n;i++)
    {
        cnt[mem[i].col][mem[i].ind%2]++;
        sum[mem[i].col][mem[i].ind%2]=(sum[mem[i].col][mem[i].ind%2]%
mod+mem[i].ind%mod)%mod;
    }
    ll ans=0;
    for(ll i=1;i<=n;i++)
    {
        ans=(ans%mod+(mem[i].num%mod*(((cnt[mem[i].col][mem[i].ind%2]-2)%
mod*mem[i].ind%mod)%mod+sum[mem[i].col][mem[i].ind%2]%mod)%mod)%mod)%mod;
    }
    cout<<ans<<endl;
}
```

线段树

2021年4月20日 23:05

https://zhuanlan.zhihu.com/p/106118909
https://www.luogu.com.cn/problem/P3372

```
#include<bits/stdc++.h>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0);
#define ll long long
#define int long long
#define pll pair<ll,ll>
#define pdd pair<double,double>
using namespace std;
const ll N = 1e5 + 5;
const ll mod = 1e9 + 7;
int tree[4*N];
int mark[4*N];
int num[N];
int n, m;
void push_down(int p, int len)
{
	mark[2 * p] += mark[p];
	mark[2 * p + 1] += mark[p];
	tree[2 * p] += mark[p] * (len - len / 2);
	tree[2 * p + 1] += mark[p] * (len / 2);
	mark[p] = 0;
}
void update(int l, int r, int d, int p, int curl, int curr)//区间更新，[l,r]为目标区间
{
	if (curr<l || curl>r)
	{
		return;
	}
	else if (curl >= l && curr <= r)
	{
		tree[p] += d * (curr - curl + 1);
		if (curr != curl)
		{
			mark[p] += d;
		}
	}
	else
	{
		int mid = (curl + curr) >> 1;
		push_down(p, curr - curl + 1);
		update(l, r, d, 2 * p, curl, mid);
		update(l, r, d, 2 * p + 1, mid + 1, curr);
		tree[p] = tree[2 * p] + tree[2 * p + 1];
	}
}
int query(int l, int r, int p, int curl, int curr)
{
	if (curr<l || curl>r)
	{
		return 0;
	}
	else if (curl >= l && curr <= r)
	{
		return tree[p];
	}
	else
	{
		int mid = (curl + curr) >> 1;
		push_down(p, curr - curl + 1);
		return query(l,r, 2 * p, curl, mid) + query(l, r, 2 * p + 1, mid + 1, curr);
	}
}

void build(int l, int r, int p)
{
	if (l == r) {
		tree[p] = num[l];
		return;
	}
	int mid = (l + r) >> 1;
	build(l, mid, 2 * p);
	build(mid + 1, r, 2 * p + 1);
	tree[p] = tree[2 * p] + tree[2 * p + 1];
}
void solve()
{
	cin >> n >> m;
	for (int i = 1; i <= n; i++)
	{
		cin >> num[i];
	}
	build(1,n,1);
	while (m--)
	{
		int op;
		cin >> op;
		if (op == 1)
		{
			int x, y, k;
			cin >> x >> y >> k;
			update(x, y, k, 1, 1, n);
		}
		else
		{
			int x, y;
			cin >> x >> y;
			cout << query(x, y, 1, 1, n) << endl;
		}
	}
}
signed main()
{
	IOS
	solve();
}
```

https://vjudge.net/problem/HDU-1166

单点更新与区间查询

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 5e4+5;
const ll mod=1e9+7;
struct seg{
    ll l,r,v;
}tree[4*N];
ll n;
ll a[N];
ll cas=1;
void build(ll p,ll l,ll r)
{
    tree[p].l=l;
    tree[p].r=r;
    if(l==r)
    {
        tree[p].v=a[l];
        return;
    }
    ll mid=(l+r)/2;
    build(p*2,l,mid);
    build(p*2+1,mid+1,r);
    tree[p].v=tree[p*2].v+tree[p*2+1].v;
}
void update(ll p,ll x,ll v)
{
    if(tree[p].l==tree[p].r)
    {
        tree[p].v+=v;
        return;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    if(x<=mid)update(p*2,x,v);
    else update(p*2+1,x,v);
    tree[p].v=tree[p*2].v+tree[p*2+1].v;
}
ll query(ll p,ll l,ll r)
{
    if(tree[p].l>=l&&tree[p].r<=r)
    {
        return tree[p].v;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    ll sum=0;
    if(l<=mid)sum+=query(p*2,l,r);
    if(r>mid)sum+=query(p*2+1,l,r);
    return sum;
}
int main()
{
    IOS;
    ll t;
    cin>>t;
    while(t--)
    {
        cin>>n;
        for(ll i=1;i<=n;i++)
        {
            cin>>a[i];
        }
        build(1,1,n);
        string op;
        cout<<"Case "<<cas++<<":"<<endl;
        while(cin>>op)
        {
            if(op=="End")break;
            if(op=="Add")
            {
                ll i,j;
                cin>>i>>j;
                update(1,i,j);
            }
            else if(op=="Sub")
            {
                ll i,j;
                cin>>i>>j;
                update(1,i,-j);
            }
            else{
                ll i,j;
                cin>>i>>j;
                cout<<query(1,i,j)<<endl;
            }
        }
    }
}
```

https://vjudge.net/problem/HDU-1754

求区间最值

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
```

```
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 2e5+5;
const ll mod=1e9+7;
struct seg{
    ll l,r,v;
}tree[N*4];
ll n,m;
ll a[N];
void build(ll p,ll l,ll r)
{
    tree[p].l=l;
    tree[p].r=r;
    if(tree[p].l==tree[p].r)
    {
        tree[p].v=a[l];
        return;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    build(p*2,l,mid);
    build(p*2+1,mid+1,r);
    tree[p].v=max(tree[p*2].v,tree[p*2+1].v);
}
void update(ll p,ll x,ll v)
{
    if(tree[p].l==tree[p].r)
    {
        tree[p].v=v;
        return;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    if(x<=mid)update(p*2,x,v);
    else update(p*2+1,x,v);
    tree[p].v=max(tree[p*2].v,tree[p*2+1].v);
}
ll query(ll p,ll l,ll r)
{
    if(tree[p].l>=l&&tree[p].r<=r)
    {
        return tree[p].v;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    ll res=-1e9;
    if(l<=mid) res=max(res,query(p*2,l,r));
    if(r>mid)  res=max(res,query(p*2+1,l,r));
    return res;
}
int main()
{
    IOS;
    while(cin>>n>>m)
    {
        for(ll i=1;i<=n;i++)
        {
            cin>>a[i];
        }
        build(1,1,n);
        while(m--)
        {
            char op;
            ll a,b;
            cin>>op;
            cin>>a>>b;
            if(op=='Q')
            {
                cout<<query(1,a,b)<<endl;
            }
            else{
                update(1,a,b);
            }
        }
    }
}
```

https://vjudge.net/problem/POJ-3468

区间更新与区间查询（带延迟标记）

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 3e5+5;
const ll mod=1e9+7;
struct seg{
	ll l,r,v;
	ll lazy;
}tree[4*N];
ll n,q;
ll a[N];
void build(ll p,ll l,ll r)
{
	tree[p].l=l;
	tree[p].r=r;
	tree[p].lazy=0;
	if(tree[p].l==tree[p].r)
	{
		tree[p].v=a[l];
		return;
	}
	ll mid=(tree[p].l+tree[p].r)/2;
	build(p*2,l,mid);
	build(p*2+1,mid+1,r);
	tree[p].v=tree[p*2].v+tree[p*2+1].v;
}
void push_down(ll p)
{
	if(tree[p].lazy)
	{
		tree[p*2].v+=tree[p].lazy*(tree[p*2].r-tree[p*2].l+1);
		tree[p*2+1].v+=tree[p].lazy*(tree[p*2+1].r-tree[p*2+1].l+1);
		tree[p*2].lazy+=tree[p].lazy;
		tree[p*2+1].lazy+=tree[p].lazy;
		tree[p].lazy=0;
	}
}
void update(ll p,ll l,ll r,ll v)
{
	if(tree[p].l>=l&&tree[p].r<=r)
	{
		tree[p].v+=v*(tree[p].r-tree[p].l+1);
		tree[p].lazy+=v;
		return;
	}
	push_down(p);
	ll mid=(tree[p].l+tree[p].r)/2;
	if(l<=mid)update(p*2,l,r,v);
	if(r>mid)update(p*2+1,l,r,v);
	tree[p].v=tree[p*2].v+tree[p*2+1].v;

}
ll query(ll p,ll l,ll r)
{
	if(tree[p].l>=l&&tree[p].r<=r)
	{
		return tree[p].v;
	}
	push_down(p);
	ll mid=(tree[p].l+tree[p].r)/2;
	ll sum=0;
	if(l<=mid)sum+=query(p*2,l,r);
	if(r>mid)sum+=query(p*2+1,l,r);
	return sum;
}

int main()
{
	IOS;
	cin>>n>>q;
	for(ll i=1;i<=n;i++)
	{
		cin>>a[i];
	}
	build(1,1,n);
	while(q--)
	{
		char op;
```

```
                cin>>op;
                if(op=='C')
                {
                        ll a,b,c;
                        cin>>a>>b>>c;
                        update(1,a,b,c);
                }
                else{
                        ll a,b;
                        cin>>a>>b;
                        cout<<query(1,a,b)<<endl;
                }
        }
}
```

https://vjudge.net/problem/POJ-2528

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e5+5;
const int mod=1e9+7;
struct seg{
    int l,r,v;
    int lazy;
}tree[4*N];
int n;
bool vis[N];
void build(int p,int l,int r)
{
    tree[p].l=l;
    tree[p].r=r;
    if(tree[p].l==tree[p].r){
        tree[p].v=0;
        tree[p].lazy=0;
        return;
    }
    int mid=(l+r)/2;
    build(p*2,l,mid);
    build(p*2+1,mid+1,r);
    tree[p].v=0;
    tree[p].lazy=0;
}
```

```
void push_down(int p)
{
    if(tree[p].lazy)
    {
        tree[p*2].v=tree[p].lazy;
        tree[p*2+1].v=tree[p].lazy;
        tree[p*2].lazy=tree[p].lazy;
        tree[p*2+1].lazy=tree[p].lazy;
        tree[p].lazy=0;
    }
}
void update(int p,int l,int r,int v)
{

    if(tree[p].l>=l&&tree[p].r<=r)
    {
        tree[p].v=v;
        tree[p].lazy=v;
        return;
    }
    push_down(p);
    int mid=(tree[p].l+tree[p].r)/2;
    if(l<=mid)update(p*2,l,r,v);
    if(r>mid)update(p*2+1,l,r,v);
    tree[p].v=-1;//-1表示该区间内的值不是单一的
}
int query(int p,int l,int r)
{
    if(tree[p].l>=l&&tree[p].r<=r)
    {
        if(tree[p].v!=-1)
        {
            if(!vis[tree[p].v]&&tree[p].v!=0){//有可能当前海报已经计算过，所以用vis数
组记录是否计算过
                vis[tree[p].v]=1;
                //cout<<"we meet "<<tree[p].l<<' '<<tree[p].r<<' '<<tree[p].v<<endl;
                return 1;
            }
            else{
                return 0;
            }
        }
    }
    push_down(p);
    int mid=(tree[p].l+tree[p].r)/2;
    int val=0;
    if(l<=mid)val+=query(p*2,l,mid);
    if(r>mid)val+=query(p*2+1,mid+1,r);
    return val;
}
vector<int>num;
int a[N];
int b[N];
int bin_search(int v,int l,int r)
{
    while(l<r)
    {
        int mid=(l+r)/2;
        if(num[mid]>=v)r=mid;
        else l=mid+1;
    }
    return l;
}
int main()
{
    IOS;
    int t;
    cin>>t;
    while(t--)
    {
        cin>>n;
        memset(vis,0,sizeof vis);
        num.clear();
        for(int i=1;i<=n;i++)
        {
            cin>>a[i]>>b[i];
            num.push_back(a[i]);
            num.push_back(b[i]);
        }
        //由于数据范围过大所以要进行离散化
        sort(num.begin(),num.end());
        num.erase(unique(num.begin(),num.end()),num.end());//排序并去重
        for(int i=1;i<n;i++)
        {
            if(num[i]-num[i-1]>1)num.push_back(num[i]-1);
            /*
            对于间隔大于1的区间，离散化后它可能覆盖原来的区间，比如
            1 4
            6 10
            离散化后为1 2 3 4,显然，5被覆盖了，所以我们要在间隔大于1的区间中插入一个
数来模拟真实的区间间隔
            原序列变成1 3 4 5 6 9 10
            离散化后变成1 2 3 4 5 6 7,然后我们对[1 4][6 10]染色也就是对[1 3][5 7]染
色，我们会发现这样离散化后答案是正确的
            */
        }
        sort(num.begin(),num.end());
        num.erase(unique(num.begin(),num.end()),num.end());
        build(1,0,num.size());
        for(int i=1;i<=n;i++)
        {
            int x=bin_search(a[i],0,num.size()-1);
            int y=bin_search(b[i],0,num.size()-1);
            //cout<<"x=="<<x<<' '<<"y=="<<y<<' '<<"i=="<<i<<endl;
            update(1,x,y,i);
        }
        cout<<query(1,0,num.size())<<endl;
    }
}
```

https://vjudge.net/problem/HDU-1698

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e5+5;
const int mod=1e9+7;
ll cas=1;
struct seg{
    ll l,r,v;
    ll lazy;
}tree[4*N];
void build(ll p,ll l,ll r)
{
    tree[p].l=l;
    tree[p].r=r;
    tree[p].v=0;
    tree[p].lazy=0;
    if(tree[p].l==tree[p].r)
    {
        tree[p].v=1;
        return;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    build(p*2,l,mid);
    build(p*2+1,mid+1,r);
}
void push_down(ll p)
{
    if(tree[p].lazy)
    {
        tree[p*2].v=tree[p].lazy;
        tree[p*2+1].v=tree[p].lazy;
        tree[p*2].lazy=tree[p].lazy;
        tree[p*2+1].lazy=tree[p].lazy;
        tree[p].lazy=0;
    }
}
void update(ll p,ll l,ll r,ll v)
{
    if(l<=tree[p].l&&r>=tree[p].r)
    {
        tree[p].v=v;
        tree[p].lazy=v;
        return;
    }
    push_down(p);
    ll mid=(tree[p].l+tree[p].r)/2;
    if(l<=mid)update(p*2,l,r,v);
    if(r>mid)update(p*2+1,l,r,v);
    tree[p].v=-1;
}
ll query(ll p,ll l,ll r)
{
    if(l<=tree[p].l&&r>=tree[p].r)
    {
        if(tree[p].v!=-1)
        {
            return tree[p].v*(tree[p].r-tree[p].l+1);
        }
    }
    push_down(p);
    ll mid=(tree[p].l+tree[p].r)/2;
    ll val=0;
    if(l<=mid)val+=query(p*2,l,r);
    if(r>mid)val+=query(p*2+1,l,r);
    return val;
}
int main()
{
    IOS;
    ll t;
    cin>>t;
    while(t--)
    {
        ll n,q;
        cin>>n>>q;
        build(1,1,n);
        while(q--)
        {
            ll a,b,z;
            cin>>a>>b>>z;
            update(1,a,b,z);
        }
        cout<<"Case "<<cas++<<": The total value of the hook is ";
        cout<<query(1,1,n)<<"."<<endl;
    }
}
```

https://vjudge.net/problem/POJ-3264

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
```

```
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e5+5;
const int mod=1e9+7;
ll a[N];
struct srg{
    ll l,r,maxn,minn;
    ll lazy;
}tree[4*N];
void build(ll p,ll l,ll r)
{
    tree[p].l=l;
    tree[p].r=r;
    tree[p].lazy=0;
    if(tree[p].l==tree[p].r)
    {
        tree[p].maxn=a[l];
        tree[p].minn=a[l];
        return;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    build(p*2,l,mid);
    build(p*2+1,mid+1,r);
    tree[p].maxn=max(tree[p*2].maxn,tree[p*2+1].maxn);
    tree[p].minn=min(tree[p*2].minn,tree[p*2+1].minn);
}
ll queryMax(ll p,ll l,ll r)
{
    if(tree[p].l>=l&&tree[p].r<=r)
    {
        return tree[p].maxn;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    ll res=-1;
    if(l<=mid)res=max(res,queryMax(p*2,l,r));
    if(r>mid)res=max(res,queryMax(p*2+1,l,r));
    return res;
}
ll queryMin(ll p,ll l,ll r)
{
    if(tree[p].l>=l&&tree[p].r<=r)
    {
        return tree[p].minn;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    ll res=1e9;
    if(l<=mid)res=min(res,queryMin(p*2,l,r));
    if(r>mid)res=min(res,queryMin(p*2+1,l,r));
    return res;
}
inline ll read() {
    ll x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch>'9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
int main()
{
    IOS;
    ll n,q;
    n=read();
    q=read();
    for(ll i=1;i<=n;i++)
    {
        a[i]=read();
    }
    build(1,1,n);
    while(q--)
    {
        ll a,b;
        a=read();
        b=read();
        cout<<queryMax(1,a,b)-queryMin(1,a,b)<<endl;
    }
}
```

只执行一次查询的版本（注意两个版本的时间复杂度差不多，甚至实测这个更慢一点）

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e5+5;
const int mod=1e9+7;
ll a[N];
struct srg{
    ll l,r,maxn,minn;
    ll lazy;
}tree[4*N];
void build(ll p,ll l,ll r)
{
    tree[p].l=l;
    tree[p].r=r;
    tree[p].lazy=0;
    if(tree[p].l==tree[p].r)
    {
        tree[p].maxn=a[l];
        tree[p].minn=a[l];
        return;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    build(p*2,l,mid);
    build(p*2+1,mid+1,r);
    tree[p].maxn=max(tree[p*2].maxn,tree[p*2+1].maxn);
    tree[p].minn=min(tree[p*2].minn,tree[p*2+1].minn);
}
pll query(ll p,ll l,ll r)
{
    if(tree[p].l>=l&&tree[p].r<=r)
    {
        return {tree[p].maxn,tree[p].minn};
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    pll p1(-1,1e9),p2(-1,1e9);
    if(l<=mid)p1=query(p*2,l,r);
    if(r>mid)p2=query(p*2+1,l,r);
    pll p3(max(p1.first,p2.first),min(p1.second,p2.second));
    return p3;
}
inline ll read() {
    ll x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch>'9') {
        if (ch == '-')
            f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
int main()
{
    IOS;
    ll n,q;
    n=read();
    q=read();
    for(ll i=1;i<=n;i++)
    {
        a[i]=read();
    }
    build(1,1,n);
    while(q--)
    {
        ll a,b;
        a=read();
        b=read();
        pll t=query(1,a,b);
        cout<<t.first-t.second<<endl;
    }
}
```

https://vjudge.net/problem/HDU-4027

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
struct seg {
    ll l, r, v;
    ll lazy;
}tree[4 * N];
int n, m;
ll e[N];
int cas = 1;
void build(ll p, ll l, ll r)
{
    tree[p].l = l;
    tree[p].r = r;
    tree[p].lazy = 0;
    if (tree[p].l == tree[p].r)
    {

        scanf("%lld", &tree[p].v);
        //cout << "scanf function called" << ' '<<l<<' '<<r<<endl;
        return;
    }
    ll mid = (tree[p].l + tree[p].r) / 2;
    build(p * 2, l, mid);
    build(p * 2 + 1, mid + 1, r);
    tree[p].v = tree[p * 2].v + tree[p * 2 + 1].v;
}
void update(ll p, ll l, ll r)//修改单点更新函数，对区间内每个点进行更新，但每个点更新的值不同
{
    if (tree[p].v == (tree[p].r - tree[p].l + 1))return;//当区间值等于区间长度，说明
区间内每个点都为1，不用更新
    if (tree[p].l == tree[p].r)
    {
        if (tree[p].l >= l && tree[p].r <= r)
            tree[p].v = sqrt(tree[p].v);
        return;
    }
    ll mid = (tree[p].l + tree[p].r) / 2;
    if (l <= mid)
        update(p * 2, l, r);
    if (r > mid)
        update(p * 2 + 1, l, r);
    tree[p].v = tree[p * 2].v + tree[p * 2 + 1].v;
}
ll query(ll p, ll l, ll r)
{
    if (tree[p].l >= l && tree[p].r <= r)
    {
        return tree[p].v;
    }
    ll mid = (tree[p].l + tree[p].r) / 2;
    ll val = 0;
    if (l <= mid)val += query(p * 2, l, r);
    if (r > mid)val += query(p * 2 + 1, l, r);
    return val;
}
int main()
{
    while (scanf("%d",&n)!=EOF)
    {
        build(1, 1, n);

        printf("Case #%d:\n", cas++);
        scanf("%d", &m);
        while (m--)
        {
            ll a, b, c;
            scanf("%lld%lld%lld", &a, &b, &c);
            if (b > c)swap(b, c);//注意b可能大于c
            if (!a)
            {
                update(1, b, c);
            }
            else {
                printf("%lld\n", query(1, b, c));
            }
        }
        printf("\n");//注意每个testcase之后要有空行
    }
}
```

https://codeforces.com/contest/1547/problem/F

```
#include<iostream>
#include<algorithm>
```

```
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cmath>
#include<set>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 3e5 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-4;
/*
第一次操作后b[1]=gcd(a[1],a[2])
第二次操作后b[1]=gcd(a[1],a[2],a[3])
...
第n-1次操作后b[1]=gcd(a[1],a[2],a[3],…,a[n])
显然，最多n-1次操作后所有元素的值会变成一样的
那么可以二分答案将最小值问题转化为判定问题
对于当前步数mid，若能在mid步内完成转化，则mid可以继续缩小，否则mid要放大
对于区间gcd的求解，可以利用线段树完成
（哇我之前从没想过线段树还能这样用）
*/
ll n;
ll a[N];
struct seg{
    ll l,r,v;
}tree[N*4];
ll gcd(ll a,ll b)
{
    if(!b)return a;
    else return gcd(b,a%b);
}
void build(ll p,ll l,ll r)
{
    tree[p].l=l;
    tree[p].r=r;
    if(tree[p].l==tree[p].r)
    {
        tree[p].v=a[l];
        return;
    }
    ll mid=(l+r)/2;
    build(p*2,l,mid);
    build(p*2+1,mid+1,r);
    tree[p].v=gcd(tree[p*2].v,tree[p*2+1].v);
}
ll query(ll p,ll l,ll r)
{
    if(tree[p].l>=l&&tree[p].r<=r)
    {
        return tree[p].v;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    ll res=0;
    if(l<=mid){
        if(res==0)
            res=query(p*2,l,r);
        else
            res=gcd(res,query(p*2,l,r));
    }
    if(r>mid)
    {
        if(res==0)
            res=query(p*2+1,l,r);
        else
            res=gcd(res,query(p*2+1,l,r));
    }
    return res;
}
int ok(ll step)
{
    set<ll>q;
    for(ll i=1;i<=n;i++)
    {
        ll b;
        if(i+step<=n){
            b=query(1,i,i+step);
        }
        else{
            b=gcd(query(1,i,n),query(1,1,(i+step)%n));
        }
        q.insert(b);
    }
    return q.size()==1;
}
void solve()
{
    cin>>n;
    for(ll i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    build(1,1,n);
    ll l=0,r=n-1;
    while(l<r)
    {
        ll mid=(l+r)/2;
        if(ok(mid))r=mid;
        else l=mid+1;
    }
    cout<<l<<endl;

}
int main()
{
    IOS;
    ll _;
    cin>>_;
    while(_--)
    {
        solve();
    }

}
```

平衡树

2021年7月16日 4:06

https://www.acwing.com/problem/content/255/

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include <string>
#include <stdio.h>
//#define INF 0x3f3f3f3f
#define INF 2147483647
#define ll int
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e5 + 5;
const int mod = 10007;
const double eps = 1e-4;
struct treap{
    ll l,r;
    ll val,dat;//关键码与用于维护平衡的随机权值
    ll cnt,size;//副本数（相同关键码数），子树规模
}a[N];
ll tot,root;
ll New(ll val)
{
    a[++tot].val=val;
    a[tot].dat=rand();
    a[tot].cnt=a[tot].size=1;
    return tot;
}
void update(ll p)
{
    a[p].size=a[a[p].l].size+a[a[p].r].size+a[p].cnt;
}
void build()
{
    New(-INF),New(INF);
    root=1,a[1].r=2;
    update(root);
}
ll getRankByValue(ll p,ll val)
{
    if(p==0)return 0;
    if(val==a[p].val)return a[a[p].l].size+1;
    if(val<a[p].val)return getRankByValue(a[p].l,val);
    return getRankByValue(a[p].r,val)+a[a[p].l].size+a[p].cnt;
}
ll getValueByRank(ll p,ll rank)
{
    if(p==0)return INF;
    if(a[a[p].l].size>=rank)return getValueByRank(a[p].l,rank);
    if(a[a[p].l].size+a[p].cnt>=rank)return a[p].val;
    return getValueByRank(a[p].r,rank-a[a[p].l].size-a[p].cnt);
}
void zig(ll &p)//右旋
{
    ll q=a[p].l;
    a[p].l=a[q].r,a[q].r=p,p=q;
    update(a[p].r),update(p);
}
void zag(ll &p)//左旋
{
    ll q=a[p].r;
    a[p].r=a[q].l,a[q].l=p,p=q;
    update(a[p].l),update(p);
}
void insert(ll &p,ll val)
{
    if(p==0)
    {
        p=New(val);
        return;
    }
    if(val==a[p].val)
    {
        a[p].cnt++,update(p);
        return;
    }
    if(val<a[p].val)
    {
        insert(a[p].l,val);
        if(a[p].dat<a[a[p].l].dat)zig(p);
    }
    else{
        insert(a[p].r,val);
        if(a[p].dat<a[a[p].r].dat)zag(p);
    }
    update(p);
}
ll getPre(ll val)
{
    ll ans=1;
    ll p=root;
    while(p)
    {
        if(val==a[p].val)
        {
            if(a[p].l>0)
            {
                p=a[p].l;
                while(a[p].r>0)p=a[p].r;
                ans=p;
            }
            break;
        }
        if(a[p].val<val&&a[p].val>a[ans].val)ans=p;
        p=val<a[p].val?a[p].l:a[p].r;
    }
    return a[ans].val;

}
ll getNext(ll val)
{
    ll ans=2;
    ll p=root;
    while(p)
    {
        if(val==a[p].val)
        {
            if(a[p].r>0)
            {
                p=a[p].r;
                while(a[p].l>0)p=a[p].l;
                ans=p;
            }
            break;
        }
        if(a[p].val>val&&a[p].val<a[ans].val)ans=p;
        p=val<a[p].val?a[p].l:a[p].r;
    }
    return a[ans].val;
}
void remove(ll &p,ll val)
{
    if(p==0)return;
    if(val==a[p].val)
    {
        if(a[p].cnt>1)
        {
            a[p].cnt--,update(p);
            return;
        }
        if (a[p].l || a[p].r)
```

```
            {
                if (a[p].r == 0 || a[a[p].l].dat > a[a[p].r].dat)
                {
                    zig(p), remove(a[p].r, val);
                }
                else
                {
                    zag(p), remove(a[p].l, val);
                }
                update(p);
            }
            else
                p = 0;
            return;
        }
        val<a[p].val?remove(a[p].l,val):remove(a[p].r,val);
        update(p);

    }
    int main()
    {
        IOS;
        build();
        ll n;
        cin>>n;
        while(n--)
        {
            int op,x;
            cin>>op>>x;
            switch(op)
            {
                case 1:
                    insert(root,x);
                    break;
                case 2:
                    remove(root,x);
                    break;
                case 3:
                    cout<<getRankByValue(root,x)-1<<endl;
                    break;
                case 4:
                    cout<<getValueByRank(root,x+1)<<endl;
                    break;
                case 5:
                    cout<<getPre(x)<<endl;
                    break;
                case 6:
                    cout<<getNext(x)<<endl;
                    break;
            }
        }
    }
```

主席树

2021年4月21日　　0:15

https://www.luogu.com.cn/problem/solution/P3919

```
#include<bits/stdc++.h>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define ll int
#define pll pair<ll,ll>
#define pdd pair<double,double>
using namespace std;
const ll N = 1e8 + 10;//注意数组要开大
const ll mod = 1e9 + 7;
ll top = 0;
ll num[N];
ll root[N];//root数组用来保存历史版本的线段树根
ll n, m;
struct node
{
	ll lson, rson, val;
}tree[N];
ll clone(ll node)
{
	tree[++top] = tree[node];
	return top;
}
ll build(ll node, ll l, ll r)
{
	node = ++top;
	if (l == r)
	{
		tree[node].val = num[l];

	}
	else {
		ll mid = (l + r) >> 1;
		tree[node].lson = build(tree[node].lson, l, mid);
		tree[node].rson = build(tree[node].rson, mid + 1, r);

	}
	return node;
}
ll update(ll l, ll r, ll node, ll x, ll val)//单点更新
{
	node = clone(node);
	if (l == r)
	{
		tree[node].val = val;

	}
	else {
		ll mid = (l + r) >> 1;
		if (x <= mid)tree[node].lson = update(l, mid, tree[node].lson, x, val);
		else tree[node].rson = update(mid + 1, r, tree[node].rson, x, val);
	}
	return node;
}
ll query(ll l, ll r, ll node, ll x)
{
	if (l == r)
	{
		return tree[node].val;
	}
	ll mid = (l + r) >> 1;
	if (x <= mid)return query(l, mid, tree[node].lson, x);
	else return query(mid + 1, r, tree[node].rson,x);
}
void solve()
{
	cin >> n >> m;
	for (ll i = 1; i <= n; i++)
		cin >> num[i];
	root[0] = build(0, 1, n);
	ll i = 1;
	while (m--)
	{
		ll vi, op;
		cin >> vi >> op;
		if (op == 1)
		{
			ll loc, val;
			cin >> loc >> val;
			root[i++] = update(1, n, root[vi],loc, val);
		}
		else
		{
			ll loc;
			cin >> loc;
			cout << query(1, n, root[vi], loc) << endl;
			root[i++] = root[vi];
		}
	}
}
signed main()
{
	IOS;
	solve();
}
```

背包问题

2021年4月29日 9:52

01背包： https://www.acwing.com/problem/content/2/

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N=1e3+5;
int dp[N][N];//dp[i][j] = k means when consider the first i objects and the volume is j,the max value
of the current selection
int V[N],W[N];
int main()
{
   int n,v;
   cin>>n>>v;
   for(int i=1;i<=n;i++)
   {
      cin>>V[i]>>W[i];
   }
   int ans=0;
   for(int i=1;i<=n;i++)
   {
      for(int j=0;j<=v;j++)
      {
         if((v-j)>=V[i]){
             dp[i][j]=max(dp[i-1][j],dp[i-1][j+V[i]]+W[i]);
         }
         else
         {
            dp[i][j]=dp[i-1][j];
         }
        ans=max(ans,dp[i][j]);
      }
   }
   cout<<ans<<endl;
}
```

Optimization in space:

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N=1e3+5;
int dp[N];
int V[N],W[N];
int main()
{
   int n,v;
   cin>>n>>v;
   for(int i=1;i<=n;i++)
   {
      cin>>V[i]>>W[i];
   }
   int ans=0;
```

```
    for(int i=1;i<=n;i++)
    {
        for(int j=v;j>=V[i];j--)//note that you need to traverse j in reverse order,nor the dp[j-V[i]] will
                                 //equal to dp[i][j-V[i]] but not dp[i-1][j-V[i]]
        {
            dp[j]=max(dp[j],dp[j-V[i]]+W[i]);
        }
    }
    cout<<dp[v]<<endl;
}
```

完全背包问题： https://www.acwing.com/problem/content/3/
https://www.cnblogs.com/mfrank/p/10803417.html

```
#include<iostream>
using namespace std;
const int N=1e3+5;
int dp[N];
int V[N],W[N];
int main()
{
    int n,v;
    cin>>n>>v;
    for(int i=1;i<=n;i++)
    {
        cin>>V[i]>>W[i];
    }
    for(int i=1;i<=n;i++)
    {
        for(int j=V[i];j<=v;j++)//注意这里是顺序不是逆序
        {
            dp[j]=max(dp[j],dp[j-V[i]]+W[i]);
        }
    }
    cout<<dp[v]<<endl;
}
```

多重背包问题:

朴素做法：

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
```

```
#define ll long long
#define ull unsigned long long
#define PI 3.1415926
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define rep(i,a,b) for(ll i=a;i<=b;i++)
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=1e3+5;
const ll mod=1e9+7;
ll n,V;
ll v[N],w[N],s[N];
ll dp[N][N];
int main()
{
    IOS;
    cin>>n>>V;
    rep(i,1,n)
    {
        cin>>v[i]>>w[i]>>s[i];
    }
    rep(i,1,n)
    {
        rep(j,0,V)
        {
            rep(k,0,s[i])
            {
                if(j-k*v[i]>=0)
                {
                    dp[i][j]=max(dp[i][j],dp[i-1][j-v[i]*k]+w[i]*k);
                }
            }
        }
    }
    cout<<dp[n][V]<<endl;

}
```

二进制优化： https://www.acwing.com/activity/content/problem/content/1000/1/

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
```

```
#define ll long long
#define ull unsigned long long
#define PI 3.1415926
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define rep(i,a,b) for(ll i=a;i<=b;i++)
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=3e4+5;
const ll mod=1e9+7;
ll n,V;
ll v[N],w[N],s[N];
ll dp[N];
/*
二进制优化的核心思想是，将数量为s的物品分为若干组，每组分别为1,2,4,...,2^k,s-(1+2+4+...+
2^k)个物品
关于其详细讲解，请查阅《算法竞赛进阶指南》
*/
int main()
{
  IOS;
  cin>>n>>V;
  ll cnt=0;
  rep(i,1,n)
  {
    ll a,b,c;
    cin>>a>>b>>c;
    ll k=1;
    while(k<=c)
    {
      cnt++;
      v[cnt]=k*a;
      w[cnt]=k*b;
      c-=k;
      k*=2;
    }
    if(s>0)
    {
      cnt++;
      v[cnt]=c*a;
      w[cnt]=c*b;
    }
  }
  rep(i,1,cnt)
  {
    for(ll j=V;j>=v[i];j--)
    {
      dp[j]=max(dp[j],dp[j-v[i]]+w[i]);
    }
  }
  cout<<dp[V]<<endl;

}
```

https://www.acwing.com/problem/content/9/

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define PI 3.1415926
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define rep(i,a,b) for(ll i=a;i<=b;i++)
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=1e3+5;
const ll mod=1e9+7;
ll n,V;
ll v[N][N],w[N][N],s[N];
ll dp[N];
int main()
{
   IOS;
   cin>>n>>V;
   rep(i,1,n)
   {
      cin>>s[i];
      rep(j,1,s[i])
      {
         cin>>v[i][j]>>w[i][j];
      }
   }
   rep(i,1,n)
   {
      for(ll j=V;j>=0;j--)
      {
         rep(k,0,s[i])
         {
            if(j-v[i][k]>=0)
            {
               dp[j]=max(dp[j],dp[j-v[i][k]]+w[i][k]);
            }
         }
      }
   }
   cout<<dp[V]<<endl;

}
```

排序不等式

2021年8月21日 0:22

排序不等式表述如下，设有两组数a1,a2,……an和b1,b2,……bn，满足a1≤a2≤……≤an,b1≤b2≤……≤bn，c1,c2,……cn是b1,b2,……bn的乱序排列，则有a1bn+a2bn-1+……+anb1≤a1c1+a2c2+……+ancn≤a1b1+a2b2+……+anbn,当且仅当a1=a2=……=an或b1=b2=……=bn时等号成立。一般为了便于记忆，常记为：反序和≤乱序和≤顺序和.

最近点对问题

2021年8月21日　　13:22

求平面中距离最近的两个点的距离

https://www.acwing.com/problem/content/121/

解法： https://www.acwing.com/solution/content/15774/

```
#include<iostream>
#include<algorithm>
#include<cmath>
#include<vector>
#include<iomanip>
#define INF 1e9+5
#define eps 1.0e-6
#define pii pair<int,int>
using namespace std;
const int N=2e5+5;
/*
假设取一个中心点将平面一分为二，那么此时最短距离有三种情况
1.最短距离是中心点左边的点到中心点的距离
2.最短点是中心点右边的点到中心点的距离
3.最短点是中心点左边的点到中心点右边的点的距离
*/
struct node{
   double x,y;
   double type;
   bool operator <(node&ob)
   {
      return x<ob.x;
   }
}p[N],tmp[N];
double cur;
double dis(node a,node b)
{
   if(a.type==b.type)return INF;
   double x=a.x-b.x;
   double y=a.y-b.y;
   return sqrt(x*x+y*y);
}
double dfs(int l,int r)
{
   if(l==r)return INF;
   if(l==r-1)return dis(p[l],p[r]);
   int mid=(l+r)/2;
   double ans=min(dfs(l,mid),dfs(mid+1,r));

   for(int i=mid;i>=l;i--)
```

```
    {
        if(p[mid].x-p[i].x+eps>cur)break;
        for(int j=mid+1;j<=r;j++)
        {
            if(p[j].x-p[mid].x+eps>cur)break;
            ans=min(ans,dis(p[i],p[j]));
        }
    }
    cur=min(cur,ans);
    return ans;
}
void solve()
{
    int n;
    scanf("%d",&n);
    cur=INF;//当前最优答案，用于剪枝
    for(int i=1;i<=n;i++)
    {
        scanf("%lf%lf",&p[i].x,&p[i].y);
        p[i].type=0;
    }
    for(int i=n+1;i<=2*n;i++)
    {
        scanf("%lf%lf",&p[i].x,&p[i].y);
        p[i].type=1;
    }
    sort(p+1,p+1+2*n+1);
    cur=dis(p[1],p[2*n]);
    printf("%.3lf\n",dfs(1,2*n));
}
signed main()
{
    int _;
    scanf("%d",&_);
    while(_--)
    {
        solve();
    }
}
```

LCS(最长公共子序列，打印路径)

2021年3月9日 23:24

https://www.luogu.com.cn/problem/AT4527

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const ll maxn = 3e3 + 5;
ll dp[maxn][maxn];
int main()
{
	ios::sync_with_stdio(false);
	cin.tie(0);
	cout.tie(0);
	memset(dp, 0, sizeof(dp));
	string a;
	string b;
	string ans="";
	cin >> a >> b;
	ll m = a.length();
	ll n = b.length();
	for (ll i = 1; i <= m; i++)
	{
		for (ll j = 1; j <= n; j++)//注意要从一开始，0行0列的值要初始化为0
		{
			if (a[i-1] == b[j-1])
			{
				dp[i][j] = dp[i - 1][j - 1] + 1;
			}
			else
			{
				dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
			}
		}
	}
	while (dp[m][n])//逆序回溯打印路径
	{
		if (dp[m][n] == dp[m - 1][n])//说明串a第m个字母不选，下面的逻辑类似
		{
			m--;
		}
		else if (dp[m][n] == dp[m][n - 1])
		{
			n--;
		}
		else if(dp[m][n]>dp[m-1][n-1])
		{
			ans += a[m-1];//注意dp[m][n]对应a中第m-1个字母与b中第n-1个字母
			m--;
			n--;
		}
	}
```

```
        reverse(ans.begin(), ans.end());
        cout << ans << endl;

}
```

线性基

2022年4月15日 17:22

https://zhuanlan.zhihu.com/p/139074556
https://blog.csdn.net/a_forever_dream/article/details/83654397
https://www.cnblogs.com/scx2015noip-as-php/p/linear_basic.html
https://www.acwing.com/video/2274/

https://www.acwing.com/problem/content/212/

```
#include<iostream>
using namespace std;
#define ll long long
const ll N = 1e5 + 5;
ll n, m, t, T;
ll a[N];
int main()
{
	cin >> T;
	for(ll C = 1; C <= T; C ++ )
	{
		cin >> n;
		for(ll i = 1; i <= n; i ++ )cin >> a[i];
		int zero = 0;
		t = n;
		for(ll i = 1; i <= n; i ++ )
		{
			for(ll j = i + 1; j <= n; j ++ )
			{
				if(a[j] > a[i])swap(a[i],a[j]);
			}
			if(a[i] == 0)
			{
				zero = 1, t = i - 1;
				break;
			}
			for(ll k = 63; k >= 0; k -- )
			{
				if(a[i] >> k & 1)
				{
					for(ll j = 1; j <= n; j ++ )
					{
						if(i != j && (a[j] >> k & 1))
						{
							a[j] ^= a[i];
						}
					}
					break;
				}
			}
		}
		cin >> m;
		cout << "Case #" << C << ":\n";
```

```
        while(m -- )
        {
            ll k, ans = 0;
            cin >> k;
            if(zero) k -- ;
            if(k >= 1ll << t) cout << "-1" << endl;
            else
            {
                for(ll i = t - 1; i >= 0; i -- )
                {
                    if(k >> i & 1)
                    {
                        ans ^= a[t - i];
                    }
                }
                cout << ans << endl;
            }
        }
    }
}
```

https://www.acwing.com/problem/content/3167/

```
#include<iostream>
using namespace std;
#define ll long long
const ll N = 1e5 + 5;
ll n, m, t, T;
ll a[N];
int main()
{

        cin >> n;
        for(ll i = 1; i <= n; i ++ )cin >> a[i];
        int zero = 0;
        t = n;
        for(ll i = 1; i <= n; i ++ )
        {
            for(ll j = i + 1; j <= n; j ++ )
            {
                if(a[j] > a[i])swap(a[i],a[j]);
            }
            if(a[i] == 0)
            {
                zero = 1, t = i - 1;
                break;
            }
            for(ll k = 63; k >= 0; k -- )
            {
                if(a[i] >> k & 1)
                {
                    for(ll j = 1; j <= n; j ++ )
                    {
                        if(i != j && (a[j] >> k & 1))
                        {
```

```
                                        a[j] ^= a[i];
                                    }
                                }
                                break;
                            }
                        }
                    }
                    ll ans = 0;
                    for(ll i = 1; i <= t; i ++ )
                    {
                        ans ^= a[i];
                    }
                    cout << ans << endl;

}
```

https://atcoder.jp/contests/arc138/tasks/arc138_d

https://www.luogu.com.cn/problem/solution/P7949

```
#include<iostream>
using namespace std;
#define ll long long
const ll N = 1e5 + 5;
ll n,k,a[22],p[22],cnt;
void insert(ll x)
{
    ll tmp = x;
    for(ll i = n - 1; i >= 0; i -- )
    {
        if(x >> i & 1)
        {
            if(!p[i])
            {
                p[i] = x;
                a[cnt ++ ] = tmp;
                break;
            }
            else
            {
                x ^= p[i];
            }
        }
    }
    return;
}
ll popcount(ll i)
{
    ll res = 0;
    while(i > 0)
    {
        res ++ ;
        i -= i & (-i);
    }
    return res;
}
int main()
```

```
{
	cin >> n >> k;
	for(ll i = 0; i < (1 << n); i ++ )
	{
		if(popcount(i) == k)
		{
			insert(i);
		}
	}
	if(cnt < n)
	{
		cout << "No" << endl;
	}
	else
	{
		cout << "Yes" << endl;
		for(ll i = 0; i < (1 << n); i ++ )
		{
			ll g = i ^ (i >> 1),now = 0;
			for(ll j = 0; j < n; j ++ )
			{
				if(g >> j & 1)
				{
					now ^= a[j];
				}
			}
			cout << now << ' ';
		}
	}
}
```

格雷码

2022年4月16日　　0:42

https://cp-algorithms.com/algebra/gray-code.html

高斯消元

2022年4月15日　　17:44

https://www.acwing.com/problem/content/885/

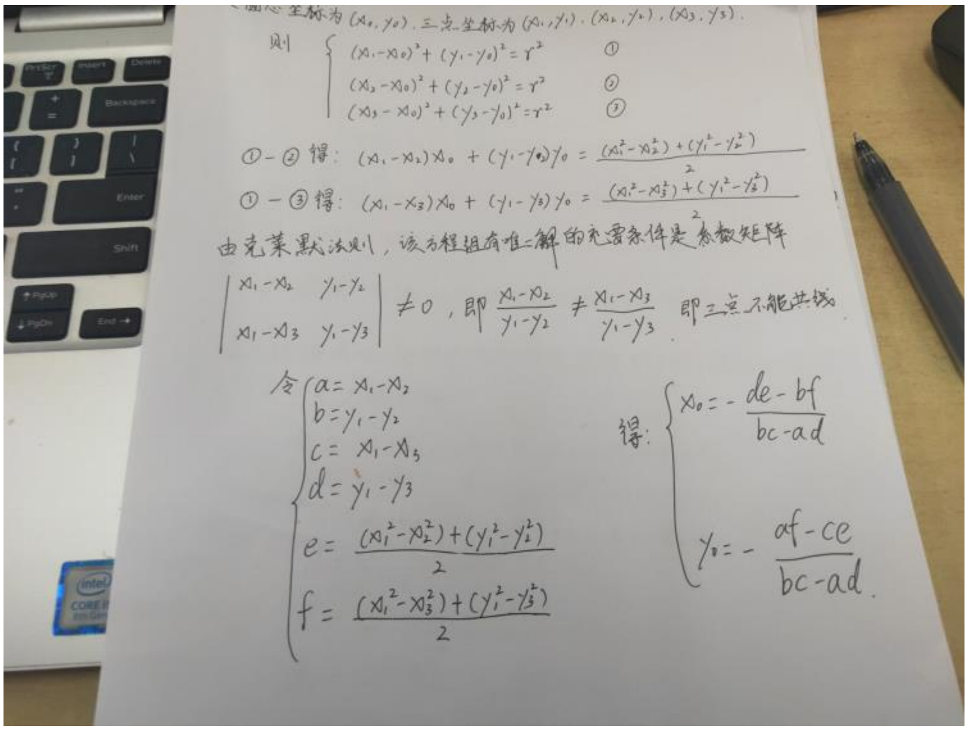
```
#include <iostream>
#include <algorithm>
#include <cmath>

using namespace std;

const int N = 110;
const double eps = 1e-6;

int n;
double a[N][N];


int gauss()
{
    int c, r;
    for (c = 0, r = 0; c < n; c ++ )
    {
        int t = r;
        for (int i = r; i < n; i ++ )
            if (fabs(a[i][c]) > fabs(a[t][c]))
                t = i;

        if (fabs(a[t][c]) < eps) continue;

        for (int i = c; i < n + 1; i ++ ) swap(a[t][i], a[r][i]);
        for (int i = n; i >= c; i -- ) a[r][i] /= a[r][c];
        for (int i = r + 1; i < n; i ++ )
            if (fabs(a[i][c]) > eps)
                for (int j = n; j >= c; j -- )
                    a[i][j] -= a[r][j] * a[i][c];

        r ++ ;
    }

    if (r < n)
    {
        for (int i = r; i < n; i ++ )
            if (fabs(a[i][n]) > eps)
                return 2;
        return 1;
    }

    for (int i = n - 1; i >= 0; i -- )
        for (int j = i + 1; j < n; j ++ )
            a[i][n] -= a[j][n] * a[i][j];

    return 0;
}
```

```
int main()
{
    cin >> n;
    for (int i = 0; i < n; i ++ )
        for (int j = 0; j < n + 1; j ++ )
            cin >> a[i][j];

    int t = gauss();

    if (t == 0)
    {
        for (int i = 0; i < n; i ++ ) printf("%.2lf\n", (fabs(a[i][n]) < eps) ? 0 : a[i][n]);
    }
    else if (t == 1) puts("Infinite group solutions");
    else puts("No solution");

    return 0;
}
```

最小圆覆盖

2021年3月10日 8:35

https://blog.csdn.net/commonc/article/details/52291822
注：其中三点定圆部分的推导如下图：



圆心坐标为 (x_0, y_0)，三点坐标为 (x_1, y_1)，(x_2, y_2)，(x_3, y_3).

则
$$\begin{cases} (x_1-x_0)^2+(y_1-y_0)^2=r^2 & ① \\ (x_2-x_0)^2+(y_2-y_0)^2=r^2 & ② \\ (x_3-x_0)^2+(y_3-y_0)^2=r^2 & ③ \end{cases}$$

①－② 得：$(x_1-x_2)x_0+(y_1-y_2)y_0=\frac{(x_1^2-x_2^2)+(y_1^2-y_2^2)}{2}$

①－③ 得：$(x_1-x_3)x_0+(y_1-y_3)y_0=\frac{(x_1^2-x_3^2)+(y_1^2-y_3^2)}{2}$

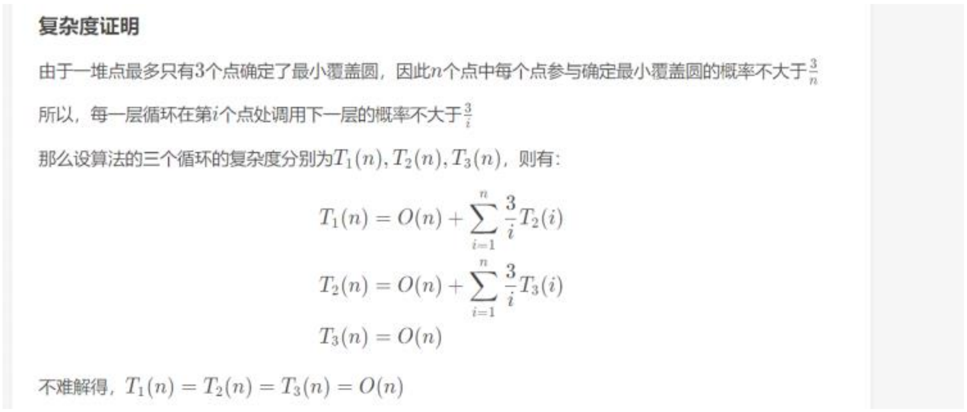
由克莱默法则，该方程组有唯一解的充要条件是系数矩阵

$\begin{vmatrix} x_1-x_2 & y_1-y_2 \\ x_1-x_3 & y_1-y_3 \end{vmatrix} \neq 0$，即 $\frac{x_1-x_2}{y_1-y_2} \neq \frac{x_1-x_3}{y_1-y_3}$. 即三点不能共线

令
$$\begin{cases} a=x_1-x_2 \\ b=y_1-y_2 \\ c=x_1-x_3 \\ d=y_1-y_3 \\ e=\frac{(x_1^2-x_2^2)+(y_1^2-y_2^2)}{2} \\ f=\frac{(x_1^2-x_3^2)+(y_1^2-y_3^2)}{2} \end{cases}$$

得：
$$\begin{cases} x_0=-\frac{de-bf}{bc-ad} \\ y_0=-\frac{af-ce}{bc-ad} \end{cases}$$

复杂度证明如下：



复杂度证明

由于一堆点最多只有3个点确定了最小覆盖圆，因此n个点中每个点参与确定最小覆盖圆的概率不大于$\frac{3}{n}$

所以，每一层循环在第i个点处调用下一层的概率不大于$\frac{3}{i}$

那么设算法的三个循环的复杂度分别为$T_1(n), T_2(n), T_3(n)$，则有：

$$T_1(n)=O(n)+\sum_{i=1}^{n}\frac{3}{i}T_2(i)$$

$$T_2(n)=O(n)+\sum_{i=1}^{n}\frac{3}{i}T_3(i)$$

$$T_3(n)=O(n)$$

不难解得，$T_1(n)=T_2(n)=T_3(n)=O(n)$

模板题：洛谷P1742: https://www.luogu.com.cn/problem/P1742

```
#include<bits/stdc++.h>
#define ll long long
#define eps 1.0E-8
using namespace std;
const ll maxn = 1e5 + 5;
struct node {
	double x;
	double y;
```

```
}point[maxn];
node centre;//圆心
double r;//半径
double dis(node a, node b)
{
	return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}
void get_cir(node p1,node p2,node p3)
{
	double a, b, c, d, e, f;
	a = p1.x - p2.x;
	b = p1.y - p2.y;
	c = p1.x - p3.x;
	d = p1.y - p3.y;
	e = ((p1.x * p1.x - p2.x * p2.x) + (p1.y * p1.y - p2.y * p2.y)) / 2;
	f = ((p1.x * p1.x - p3.x * p3.x) + (p1.y * p1.y - p3.y * p3.y)) / 2;
	centre.x = (b * f - d * e) / (b * c - a * d);
	centre.y = (c * e - a * f) / (b * c - a * d);
}
int main()
{
	ll n;
	cin >> n;
	for (ll i = 1; i <= n; i++)
	{
		cin >> point[i].x >> point[i].y;
	}
	random_shuffle(point + 1, point + 1 + n);
	centre = point[1];
	r = 0;
	for (ll i = 2; i <= n; i++)
	{
		if (dis(centre, point[i]) - r > eps)
		{
			centre = point[i];
			r = 0;
			for (ll j = 1; j < i; j++)
			{
				if (dis(centre, point[j]) - r > eps)
				{
					centre.x = (point[i].x + point[j].x) / 2;
					centre.y = (point[i].y + point[j].y) / 2;
					r = dis(centre, point[j]);
					for (ll k = 1; k < j; k++)
					{
						if (dis(centre, point[k]) - r > eps)
						{
							get_cir(point[i], point[j], point[k]);
							r = dis(centre, point[i]);
						}
					}
				}
			}
		}
	}
	cout << fixed << setprecision(10) << r << endl;
```

```
        cout << centre.x << ' ' << centre.y << endl;
}
```

浮点数误差

2021年3月10日 16:40

https://www.zhihu.com/question/36176935
在浮点数比较和取整时，一般设置一个极小常量作为误差，比如：

```
#define zero(x) (((x)>0?(x):-(x))<eps)//小于误差值的浮点数就被视为0
#define eps 1.0E-8

int double_cmp(double a)
{
        if (zero(a))
                return 0;
        return a > 0 ? 1 : -1;
}
```

比较两个浮点数大小时，可以作差然后将差传入这个函数看结果；
再比如取整时，假设要将double型的a取整存入int型的b：
b = (int)(a + eps);
这与一般写法的主要差别是，当a与整数极其接近，比如3.99999999,该写法b会得到4，而一般写法会得到3；

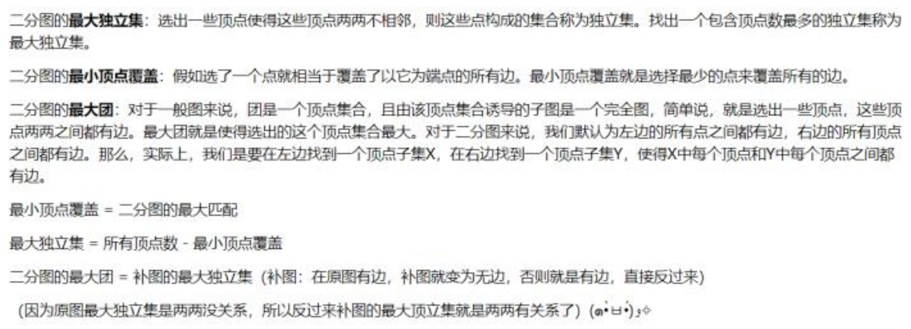
匈牙利算法

2021年3月12日　　16:44

https://blog.csdn.net/u013384984/article/details/90718287
https://zhuanlan.zhihu.com/p/96229700
模板题：洛谷p3386： https://www.luogu.com.cn/problem/P3386
解：

```
#include<bits/stdc++.h>
#define ll long long
#define eps 1.0E-8
#define zero(x) (((x)>0?(x):-(x))<eps)
using namespace std;
const ll N = 2e5 + 5;
ll mp[505][505] = { 0 };
ll ln, rn;//左、右集合数
ll e;//边数
ll vis[N] = { 0 };//右侧节点是否被访问过
ll p[N] = { 0 };//保存匹配结果，p[i]=j语义为：编号为i的右侧节点匹配编号为j的左侧节点
bool match(ll i)
{
	for (ll j = 1; j <= rn; j++)
	{
		if (mp[i][j]&&!vis[j])//有边且未访问
		{
			vis[j] = 1;
			if (p[j] == 0 || match(p[j]))//若暂无匹配或已匹配的点能找到其他匹配
			{
				p[j] = i;
				return true;
			}
		}
	}
	return false;
}
void solve()
{
	cin >> ln >> rn >> e;
	for (ll i = 1; i <= e; i++)
	{
		ll u, v;
		cin >> u >> v;
		mp[u][v] = 1;
	}
	ll cnt = 0;
	for (ll i = 1; i <= ln; i++)
	{
		memset(vis, 0, sizeof(vis));//每次遍历vis要重置
		if (match(i))cnt++;
	}
	cout << cnt << endl;
}
int main()
{
	ios::sync_with_stdio(false);
	cin.tie(0);
	cout.tie(0);
	solve();
}
```

此外，补充一些二分图相关的性质：


二分图的**最大独立集**：选出一些顶点使得这些顶点两两不相邻，则这些点构成的集合称为独立集。找出一个包含顶点数最多的独立集称为最大独立集。

二分图的**最小顶点覆盖**：假如选了一个点就相当于覆盖了以它为端点的所有边。最小顶点覆盖就是选择最少的点来覆盖所有的边。

二分图的**最大团**：对于一般图来说，团是一个顶点集合，且由该顶点集合诱导的子图是一个完全图，简单说，就是选出一些顶点，这些顶点两两之间都有边。最大团就是使得选出的这个顶点集合最大。对于二分图来说，我们默认为左边的所有点之间都有边，右边的所有顶点之间都有边。那么，实际上，我们是要在左边找到一个顶点子集X，在右边找到一个顶点子集Y，使得X中每个顶点和Y中每个顶点之间都有边。

最小顶点覆盖 = 二分图的最大匹配

最大独立集 = 所有顶点数 - 最小顶点覆盖

二分图的最大团 = 补图的最大独立集（补图：在原图有边，补图就变为无边，否则就是有边，直接反过来）

（因为原图最大独立集是两两没关系，所以反过来补图的最大顶立集就是两两有关系了）(๑•̀ㅂ•́)و✧

另外，如果是无向图，那么求出的最大匹配数要除以2

https://www.acwing.com/problem/content/863/

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define PI 3.1415926
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define rep(i,a,b) for(ll i=a;i<=b;i++)
#define rev(i,a,b) for(ll i=a;i>=b;i--)
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=1e6+5;
const ll mod=1e9+7;
ll n1,n2,m;
vector<ll>g[N];
ll p[N];
int st[N];
int match(ll x)
{
   for(auto i:g[x])
   {
      if (!st[i])
      {
         st[i] = 1;
         if (p[i] == 0 || match(p[i])) //如果当前点没有匹配或者其原来匹配的点可以找到其他匹
配
         {
            p[i] = x;

            return 1;
         }
      }
   }
   return 0;
}
int main()
{
   IOS;
   cin>>n1>>n2>>m;//左点集数、右点集数、边数
   while(m--)
   {
      ll u,v;
      cin>>u>>v;
      g[u].push_back(v);//左点集向右点集连一条边
   }
   ll res=0;
   rep(i,1,n1)//考虑左点集中每一个点
   {
      memset(st,0,sizeof st);//右点集的点是否被访问过
      if(match(i))
      {
         res++;
      }
   }
   cout<<res<<endl;

}
```

Dinic最大流算法

2021年9月27日 22:24

https://www.acwing.com/problem/content/2174/

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e5+5;
#define INF 0x3f3f3f3f

struct Dinic{
   int n,m,s,t;
   struct Edge{
      int from,to,cap,flow;
   };
   vector<int>G[N];//邻接表，G[i][j]表示节点i的第j条边在边表中的下标
   vector<Edge>edges;//边表
   bool vis[N];
   int d[N];//分层图的层次数组
   int cur[N];//当前弧下标
   void AddEdge(int from,int to,int cap)
   {
      edges.push_back((Edge){from,to,cap,0});
      edges.push_back((Edge){to,from,0,0});//反向边的容量为0
      m=edges.size();
      G[from].push_back(m-2);
      G[to].push_back(m-1);
   }
   bool BFS()//BFS构造分层图
   {
      memset(vis,0,sizeof vis);
      queue<int>Q;
      Q.push(s);
      d[s]=0;
      vis[s]=1;
      while(!Q.empty())
      {
         int x=Q.front();Q.pop();
         for(int i=0;i<(int)G[x].size();i++)
         {
            Edge& e=edges[G[x][i]];
            if(!vis[e.to]&&e.cap>e.flow)//寻找残量网络中的弧
            {
               vis[e.to]=1;
               d[e.to]=d[x]+1;
               Q.push(e.to);
            }
         }
      }
      return vis[t];
   }
```

```
    int DFS(int x,int a)//寻找增广路，a表示“目前为止所有弧的最小残量”
    {
        if(x==t||a==0)return a;
        int flow=0,f;
        for(int& i=cur[x];i<(int)G[x].size();i++)//从上次考虑的弧开始
        {
            Edge& e=edges[G[x][i]];
            if(d[x]+1==d[e.to]&&(f=DFS(e.to,min(a,e.cap-e.flow)))>0)
            {
                e.flow+=f;
                edges[G[x][i]^1].flow-=f;//更新反向边流量
                flow+=f;
                a-=f;
                if(a==0)break;
            }
        }
        return flow;
    }
    int MaxFlow(int s,int t)
    {
        this->s=s;this->t=t;
        int flow=0;
        while(BFS())
        {
            memset(cur,0,sizeof cur);
            flow+=DFS(s,INF);
        }
        return flow;
    }
};
int main()
{
    Dinic dinic;
    int n,m,s,t;
    cin>>n>>m>>s>>t;
    dinic.n=n;
    while(m--)
    {
        int u,v,w;
        cin>>u>>v>>w;
        dinic.AddEdge(u,v,w);
    }
    cout<<dinic.MaxFlow(s,t)<<endl;
}
```

https://vjudge.net/problem/Gym-101981I

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
using namespace std;
const int N=1e5+5;
int n,m,s,t;
```

```
struct Edge
{
   int from,to,cap,flow;
};
/*
3 5 2
4 1 2 3 5
2 2 5
2 1 2

5 10 2
2 3 10
5 1 3 4 6 10
5 3 4 6 8 9
3 1 9 10
5 1 3 6 7 10

3 5 2
1 2
2 2 3
3 1 4 5
*/
vector<int>G[N];//邻接表，G[i][j]表示节点i的第j条边在边表中的下标
vector<Edge>edges;//边表
bool vis[N];
int d[N];//分层图的层次数组
int cur[N];//当前弧下标
void AddEdge(int from,int to,int cap)
{
   edges.push_back((Edge)
   {
      from,to,cap,0
   });
   edges.push_back((Edge)
   {
      to,from,0,0
   });//反向边的容量为0
   m=edges.size();
   G[from].push_back(m-2);
   G[to].push_back(m-1);
}
bool BFS()//BFS构造分层图
{
   memset(vis,0,sizeof vis);
   queue<int>Q;
   Q.push(s);
   d[s]=0;
   vis[s]=1;
   while(!Q.empty())
   {
      int x=Q.front();
      Q.pop();
      for(int i=0; i<(int)G[x].size(); i++)
      {
            Edge& e=edges[G[x][i]];
            if(!vis[e.to]&&e.cap>e.flow)//寻找残量网络中的弧
            {
                vis[e.to]=1;
                d[e.to]=d[x]+1;
                Q.push(e.to);
            }
        }
    }
    return vis[t];
}
int DFS(int x,int a)//寻找增广路，a表示“目前为止所有弧的最小残量”
{
    if(x==t||a==0)return a;
    int flow=0,f;
    for(int& i=cur[x]; i<(int)G[x].size(); i++) //从上次考虑的弧开始
    {
        Edge& e=edges[G[x][i]];
        if(d[x]+1==d[e.to]&&(f=DFS(e.to,min(a,e.cap-e.flow)))>0)
        {
            e.flow+=f;
            edges[G[x][i]^1].flow-=f;//更新反向边流量
            flow+=f;
            a-=f;
            if(a==0)break;
        }
    }
    return flow;
}
int MaxFlow(int ss,int tt)
{
    s=ss;
    t=tt;
    int flow=0;
    while(BFS())
    {
        memset(cur,0,sizeof cur);
        flow+=DFS(s,INF);
    }
    return flow;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    int n,m,k;
    cin>>n>>m>>k;
    for(int i=1;i<=n;i++)
    {
        AddEdge(0,i,1);
        int t;
        cin>>t;
        for(int j=1;j<=t;j++)
        {
            int mons;
            cin>>mons;
            AddEdge(i,n+mons,1);
        }
    }
    for(int i=1;i<=m;i++)
    {
        AddEdge(n+i,1000,1);
    }
    AddEdge(0,1001,k);
    for(int i=1;i<=n;i++)
    {
        AddEdge(1001,i,1);
    }
    cout<<MaxFlow(0,1000)<<endl;

}
```

manacher

2021年10月14日 19:37

https://cp-algorithms.com/string/manacher.html#toc-tgt-4

https://www.luogu.com.cn/problem/P3805

这个程序奇数串和偶数串分开判断，会t几个点

```
#include<bits/stdc++.h>
#define ll long long
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f
using namespace std;
const ll N=2e7+5;
const ll mod=1e9+7;
string s;
ll d1[N];
ll d2[N];
void manacher(ll n)
{
   for(ll i=0,l=0,r=-1;i<n;i++)
   {
      ll k=(i>r)?1:min(d1[l+r-i],r-i+1);
      while(i-k>=0&&i+k<n&&s[i-k]==s[i+k])
      {
         k++;
      }
      d1[i]=k--;
      if(i+k>r)
      {
         l=i-k;
         r=i+k;
      }
   }
   for(ll i=0,l=0,r=-1;i<n;i++)
   {
      ll k=(i>k)?0:min(d2[l+r-i+1],r-i+1);
      while(i-k-1>=0&&i+k<n&&s[i-k-1]==s[i+k])
      {
         k++;
      }
      d2[i]=k--;
      if(i+k>r)
      {
         l=i-k-1;
         r=i+k;
      }
   }
}
void solve()
{
   cin>>s;
   manacher(s.length());
   ll ans=1;
```

```
    for(int i=0;i<s.length();i++)
    {
        ans=max(ans,(d1[i]-1)*2+1);
        ans=max(ans,d2[i]*2);
    }
    cout<<ans<<endl;
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cout.tie(0);
    solve();
}
```

在每个字符中间加入一个特殊字符，就可以只判奇数串

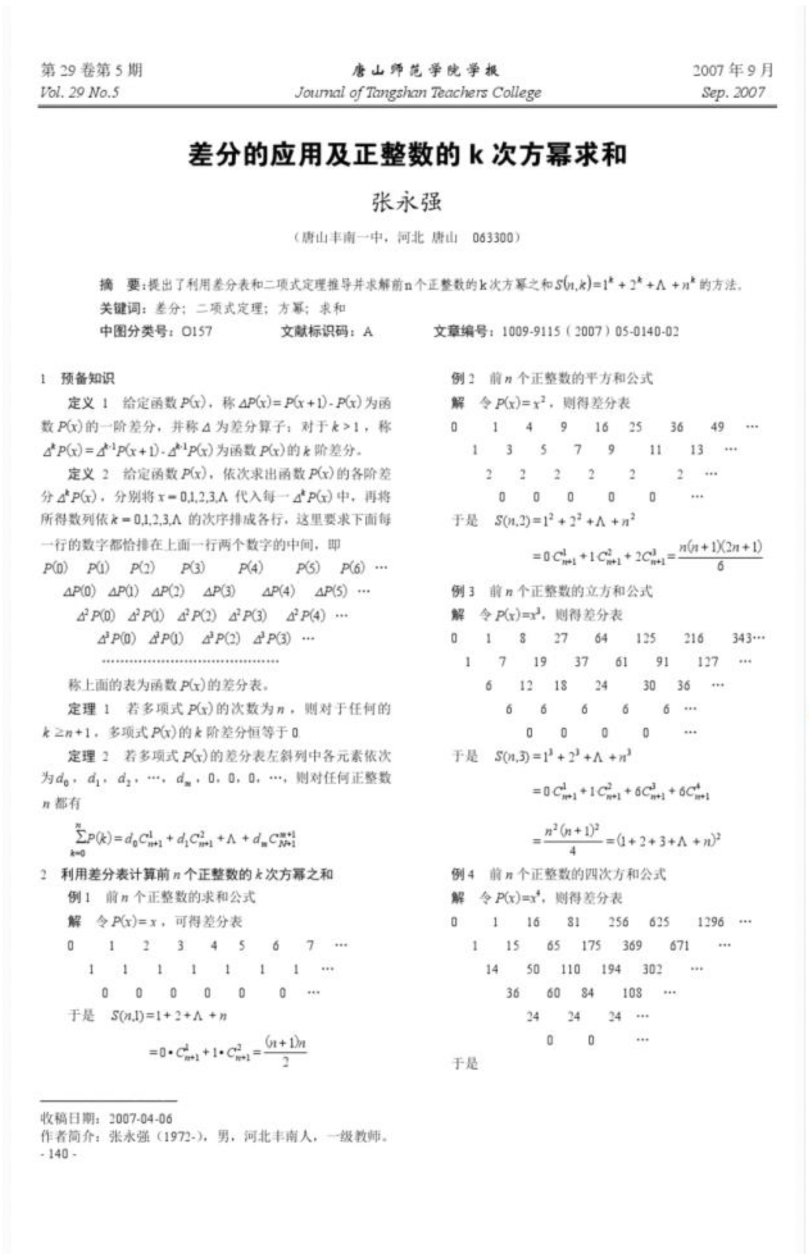
原版的manacher真的超慢，建议还是用这个

```
#include<bits/stdc++.h>
#define ll long long
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f
using namespace std;
const ll N=2e7+5;
const ll mod=1e9+7;
char s[N];
char t[N];
ll d1[N];
ll d2[N];
int main()
{
    scanf("%s",&s);
    ll n=strlen(s);
    int j=0;
    for(int i=0;i<n;i++)
    {
        t[j++]='#';
        t[j++]=s[i];
    }
    t[j++]='#';
    t[j]='\0';
    ll ans=1;
    n=strlen(t);
    for(ll i=0,l=0,r=-1;i<n;i++)
    {
        ll k=(i>r)?1:min(d1[l+r-i],r-i+1);
        while(i-k>=0&&i+k<n&&t[i-k]==t[i+k])
        {
            k++;
        }
        d1[i]=k--;
        ans=max(ans,(d1[i]-1)*2+1);
        if(i+k>r)
        {
            l=i-k;
            r=i+k;
        }
    }
    printf("%lld",ans/2);

}
```

差分求多项式和

2021年9月28日　　14:45

https://xueshu.baidu.com/s?wd=paperuri%3A%281d6cf25a905f1951b577cf287a82e1a3%29&filter=sc_long_sign&tn=SE_xueshusource_2kduw22v&sc_vurl=http%3A%2F%2Fwenku.baidu.com%2Fview%2Fef155ff1a26925c52dc5bf32.html&ie=utf-8&sc_us=12108961739379565888



第 29 卷第 5 期　　唐山师范学院学报　　2007 年 9 月
Vol. 29 No.5　　Journal of Tangshan Teachers College　　Sep. 2007

差分的应用及正整数的 k 次方幂求和

张永强

（唐山丰南一中，河北 唐山　063300）

摘　要：提出了利用差分表和二项式定理推导并求解前n个正整数的k次方幂之和 $S(n,k)=1^k+2^k+\Lambda+n^k$ 的方法.

关键词：差分；二项式定理；方幂；求和
中图分类号：O157　　**文献标识码**：A　　**文章编号**：1009-9115（2007）05-0140-02

1　预备知识

定义 1　给定函数 $P(x)$，称 $\Delta P(x)=P(x+1)-P(x)$ 为函数 $P(x)$ 的一阶差分，并称 Δ 为差分算子；对于 $k>1$，称 $\Delta^k P(x)=\Delta^{k-1}P(x+1)-\Delta^{k-1}P(x)$ 为函数 $P(x)$ 的 k 阶差分.

定义 2　给定函数 $P(x)$，依次求出函数 $P(x)$ 的各阶差分 $\Delta^k P(x)$，分别将 $x=0,1,2,3,\Lambda$ 代入每一 $\Delta^k P(x)$ 中，再将所得数列依 $k=0,1,2,3,\Lambda$ 的次序排成各行，这里要求下面每一行的数字都恰排在上面一行两个数字的中间，即

$$P(0)\quad P(1)\quad P(2)\quad P(3)\quad P(4)\quad P(5)\quad P(6)\ \cdots$$
$$\Delta P(0)\quad \Delta P(1)\quad \Delta P(2)\quad \Delta P(3)\quad \Delta P(4)\quad \Delta P(5)\ \cdots$$
$$\Delta^2 P(0)\quad \Delta^2 P(1)\quad \Delta^2 P(2)\quad \Delta^2 P(3)\quad \Delta^2 P(4)\ \cdots$$
$$\Delta^3 P(0)\quad \Delta^3 P(1)\quad \Delta^3 P(2)\quad \Delta^3 P(3)\ \cdots$$
$$\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots$$

称上面的表为函数 $P(x)$ 的差分表。

定理 1　若多项式 $P(x)$ 的次数为 n，则对于任何的 $k\geq n+1$，多项式 $P(x)$ 的 k 阶差分恒等于 0

定理 2　若多项式 $P(x)$ 的差分表左斜列中各元素依次为 d_0，d_1，d_2，…，d_m，0，0，0，…，则对任何正整数 n 都有

$$\sum_{k=0}^{n}P(k)=d_0C_{n+1}^1+d_1C_{n+1}^2+\Lambda+d_mC_{N+1}^{m+1}$$

2　利用差分表计算前 n 个正整数的 k 次方幂之和

例 1　前 n 个正整数的求和公式

解　令 $P(x)=x$，可得差分表

```
0    1    2    3    4    5    6    7    ···
   1    1    1    1    1    1    1    ···
      0    0    0    0    0    0    ···
```

于是　$S(n,1)=1+2+\Lambda+n$

$$=0\bullet C_{n+1}^1+1\bullet C_{n+1}^2=\frac{(n+1)n}{2}$$

例 2　前 n 个正整数的平方和公式

解　令 $P(x)=x^2$，则得差分表

```
0    1    4    9    16    25    36    49    ···
   1    3    5    7     9    11    13    ···
      2    2    2    2     2     2    ···
         0    0    0    0     0    ···
```

于是　$S(n,2)=1^2+2^2+\Lambda+n^2$

$$=0C_{n+1}^1+1C_{n+1}^2+2C_{n+1}^3=\frac{n(n+1)(2n+1)}{6}$$

例 3　前 n 个正整数的立方和公式

解　令 $P(x)=x^3$，则得差分表

```
0    1    8    27    64    125    216    343···
   1    7    19    37    61     91    127    ···
      6    12    18    24    30     36    ···
         6     6     6     6     6    ···
            0     0     0     0    ···
```

于是　$S(n,3)=1^3+2^3+\Lambda+n^3$

$$=0C_{n+1}^1+1C_{n+1}^2+6C_{n+1}^3+6C_{n+1}^4$$

$$=\frac{n^2(n+1)^2}{4}=(1+2+3+\Lambda+n)^2$$

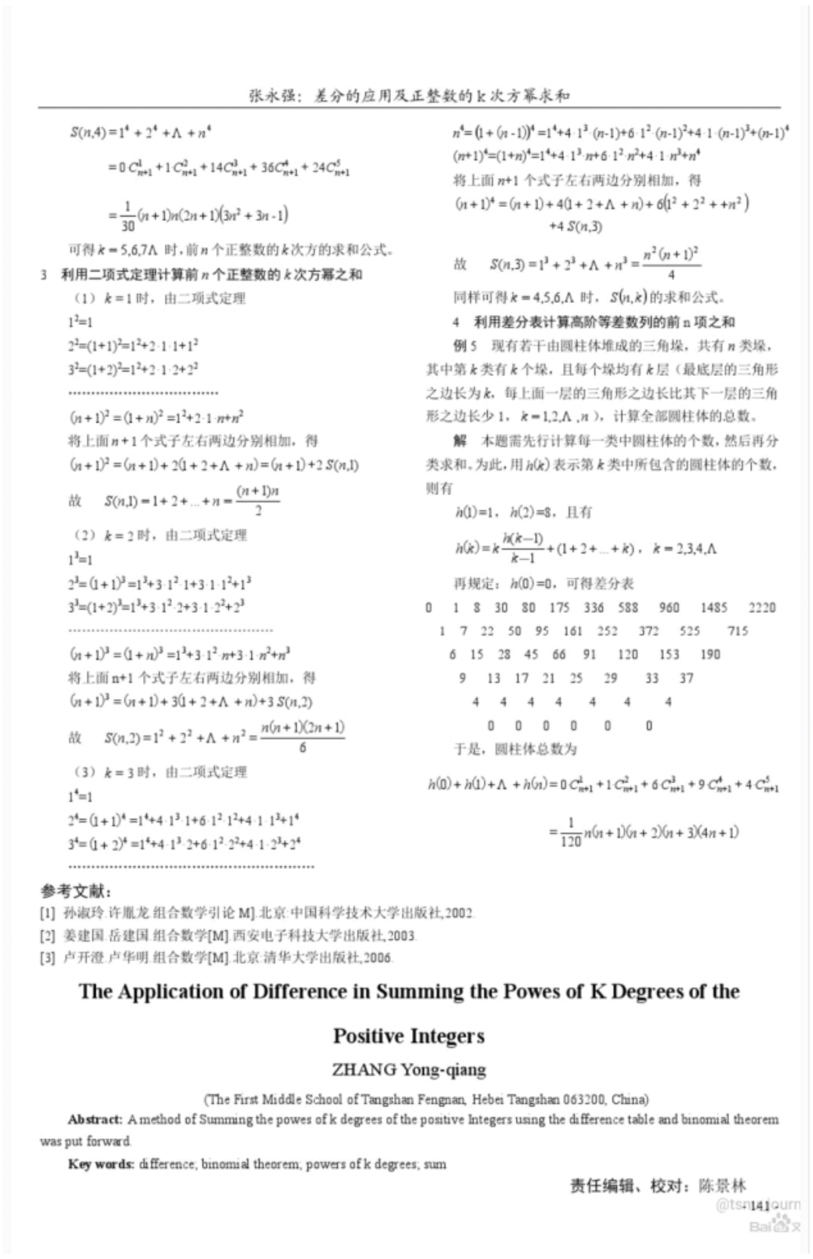
例 4　前 n 个正整数的四次方和公式

解　令 $P(x)=x^4$，则得差分表

```
0    1    16    81    256    625    1296    ···
   1    15    65    175    369    671    ···
      14    50    110    194    302    ···
         36    60    84    108    ···
            24    24    24    ···
               0     0    ···
```

于是

收稿日期：2007-04-06
作者简介：张永强（1972-），男，河北丰南人，一级教师。



$$S(n,4)=1^4+2^4+\Lambda+n^4$$
$$=0\,C_{n+1}^1+1\cdot C_{n+1}^2+14C_{n+1}^3+36C_{n+1}^4+24C_{n+1}^5$$
$$=\frac{1}{30}(n+1)n(2n+1)(3n^2+3n-1)$$

可得 $k=5,6,7\Lambda$ 时，前 n 个正整数的 k 次方的求和公式。

3 利用二项式定理计算前 n 个正整数的 k 次方幂之和

（1）$k=1$ 时，由二项式定理

$1^2=1$

$2^2=(1+1)^2=1^2+2\cdot1\cdot1+1^2$

$3^2=(1+2)^2=1^2+2\cdot1\cdot2+2^2$

………………………………

$(n+1)^2=(1+n)^2=1^2+2\cdot1\cdot n+n^2$

将上面 $n+1$ 个式子左右两边分别相加，得

$(n+1)^2=(n+1)+2(1+2+\Lambda+n)=(n+1)+2S(n,1)$

故 $S(n,1)=1+2+\ldots+n=\frac{(n+1)n}{2}$

（2）$k=2$ 时，由二项式定理

$1^3=1$

$2^3=(1+1)^3=1^3+3\cdot1^2\cdot1+3\cdot1\cdot1^2+1^3$

$3^3=(1+2)^3=1^3+3\cdot1^2\cdot2+3\cdot1\cdot2^2+2^3$

……………………………………………

$(n+1)^3=(1+n)^3=1^3+3\cdot1^2\cdot n+3\cdot1\cdot n^2+n^3$

将上面 n+1 个式子左右两边分别相加，得

$(n+1)^3=(n+1)+3(1+2+\Lambda+n)+3S(n,2)$

故 $S(n,2)=1^2+2^2+\Lambda+n^2=\frac{n(n+1)(2n+1)}{6}$

（3）$k=3$ 时，由二项式定理

$1^4=1$

$2^4=(1+1)^4=1^4+4\cdot1^3\cdot1+6\cdot1^2\cdot1^2+4\cdot1\cdot1^3+1^4$

$3^4=(1+2)^4=1^4+4\cdot1^3\cdot2+6\cdot1^2\cdot2^2+4\cdot1\cdot2^3+2^4$

……………………………………………………

$n^4=(1+(n-1))^4=1^4+4\cdot1^3\cdot(n-1)+6\cdot1^2\cdot(n-1)^2+4\cdot1\cdot(n-1)^3+(n-1)^4$

$(n+1)^4=(1+n)^4=1^4+4\cdot1^3\cdot n+6\cdot1^2\cdot n^2+4\cdot1\cdot n^3+n^4$

将上面 $n+1$ 个式子左右两边分别相加，得

$$(n+1)^4=(n+1)+4(1+2+\Lambda+n)+6(1^2+2^2++n^2)$$
$$+4S(n,3)$$

故 $S(n,3)=1^3+2^3+\Lambda+n^3=\frac{n^2(n+1)^2}{4}$

同样可得 $k=4,5,6,\Lambda$ 时，$S(n,k)$ 的求和公式。

4 利用差分表计算高阶等差数列的前 n 项之和

例 5 现有若干由圆柱体堆成的三角垛，共有 n 类垛，其中第 k 类有 k 个垛，且每个垛均有 k 层（最底层的三角形之边长为 k，每上面一层的三角形之边长比其下一层的三角形之边长少 1，$k=1,2,\Lambda,n$），计算全部圆柱体的总数。

解 本题需先行计算每一类中圆柱体的个数，然后再分类求和。为此，用 $h(k)$ 表示第 k 类中所包含的圆柱体的个数，则有

$h(1)=1$，$h(2)=8$，且有

$$h(k)=k\frac{h(k-1)}{k-1}+(1+2+\ldots+k)，\quad k=2,3,4,\Lambda$$

再规定：$h(0)=0$，可得差分表

```
0   1   8   30   80   175   336   588   960   1485   2220
  1   7   22   50   95   161   252   372   525   715
    6   15   28   45   66   91   120   153   190
      9   13   17   21   25   29   33   37
        4   4   4   4   4   4   4
          0   0   0   0   0   0
```

于是，圆柱体总数为

$$h(0)+h(1)+\Lambda+h(n)=0\,C_{n+1}^1+1\,C_{n+1}^2+6\,C_{n+1}^3+9\,C_{n+1}^4+4\,C_{n+1}^5$$
$$=\frac{1}{120}n(n+1)(n+2)(n+3)(4n+1)$$

参考文献：

[1] 孙淑玲.许胤龙.组合数学引论[M].北京:中国科学技术大学出版社,2002.

[2] 姜建国.岳建国.组合数学[M].西安电子科技大学出版社,2003.

[3] 卢开澄.卢华明.组合数学[M].北京:清华大学出版社,2006.

The Application of Difference in Summing the Powes of K Degrees of the Positive Integers

ZHANG Yong-qiang

(The First Middle School of Tangshan Fengnan, Hebei Tangshan 063200, China)

Abstract: A method of Summing the powes of k degrees of the positive Integers using the difference table and binomial theorem was put forward.

Key words: difference; binomial theorem; powers of k degrees; sum

责任编辑、校对：陈景林

模拟退火

2021年3月14日　　10:42

https://blog.csdn.net/AI_BigData_wh/article/details/77943787?locationNum=2&fps=1

最小球覆盖（模拟退火求法）

2021年9月26日 12:05

https://vjudge.net/problem/Gym-101981D

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
using namespace std;
const int N=1e2+5;
const double eps=1e-3;
const double start_t=2000;
const double rate=0.98;
struct node{
    double x,y,z;
}p[N];
int n;
double dis(node&a,node&b)
{
    return sqrt((a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y)+(a.z-b.z)*(a.z-b.z));
}
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++)
    {
        cin>>p[i].x>>p[i].y>>p[i].z;
    }
    double T=start_t;
    double ans=1e99;
    node ans_p={0,0,0};
    while(T>eps)
    {
        node max_p=p[1];
        for(int i=2;i<=n;i++)
        {
            if(dis(p[i],ans_p)>dis(max_p,ans_p))
            {
                max_p=p[i];
            }
        }
        ans=min(ans,dis(ans_p,max_p));
        ans_p.x+=(max_p.x-ans_p.x)*(T/start_t);
        ans_p.y+=(max_p.y-ans_p.y)*(T/start_t);
        ans_p.z+=(max_p.z-ans_p.z)*(T/start_t);
        T*=rate;
    }
    cout<<fixed<<setprecision(8)<<ans<<endl;
}
```

树的重心

2021年12月5日 20:31

https://www.luogu.com.cn/blog/nickelth/shu-di-zhong-xin

```
#include<iostream>
#include<cmath>
#include<vector>
#include<algorithm>
#include<cstring>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 2e5 + 5;
const ll mod = 1e9+7;
const double eps = 1e-4;
ll weight[N];
ll sz[N];
ll sum[N];
ll n;
vector<ll>tree[N];
void dfs(ll p,ll dep)
{
   sz[p]=weight[p];//在这道题中size要计入点权
   for(auto i:tree[p])
   {
      dfs(i,dep+1);
      sz[p]+=sz[i];
   }
   sum[1]+=weight[p]*dep;
}
void dp(ll p)
{
   for(auto i:tree[p])
   {
      sum[i]=sum[p]-sz[i]+sz[1]-sz[i];
      dp(i);
   }
}
int main()
{
   IOS;
   cin>>n;
   for(ll i=1;i<=n;i++)
   {
      ll w,u,v;
      cin>>w>>u>>v;
      if(u!=0)
```

```
            tree[i].push_back(u);
        if(v!=0)
            tree[i].push_back(v);
        weight[i]=w;
    }
    dfs(1,0);
    dp(1);
    ll ans=1e9;
    for(ll i=1;i<=n;i++)
    {
        ans=min(ans,sum[i]);
    }
    cout<<ans<<endl;

}
```

Tarjan算法（待补）

2021年3月20日　　12:27

kmp

2021年4月7日 21:02

https://blog.csdn.net/starstar1992/article/details/54913261

next[i]=j的含义为：str[1~j]==str[i-j+1~i];注意这两个序列的长度要在同区间中取最长，用自然语言描述其含义即：以下标i为终点的，与待匹配串前缀子串相匹配的待匹配串的后缀子串最长长度。如果前缀和后缀都等于整个串（即整个串的字符都相同），我们约定前缀不包括最后一个字符。

Sample: https://www.acwing.com/problem/content/833/

```
#include<iostream>
using namespace std;
const int N=1e6+5;
int n,m;
string s,p;
int ne[N];
int main()
{
   cin>>m>>p>>n>>s;
   //p is substring of s
   //calculate ne[N]
   ne[0]=-1;
   for(int i=1,j=-1;i<m;i++)
   {
      while(j>-1&&p[j+1]!=p[i])j=ne[j];
      if(p[i]==p[j+1])j++;
      ne[i]=j;
   }

   for(int i=0,j=-1;i<n;i++)
   {
      while(j>-1&&s[i]!=p[j+1])j=ne[j];
      if(s[i]==p[j+1])j++;
      if(j==m-1)
      {
         cout<<i-m+1<<' ';
         j=ne[j];
      }
   }
}
```

Z-function(EX-KMP)

2021年10月17日 1:27

https://cp-algorithms.com/string/z-function.html

https://www.luogu.com.cn/problem/P5410

```
#include<iostream>
#include<algorithm>
#include<cstring>
#define ll long long
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f
using namespace std;
const ll N=2e7+5;
const ll mod=1e9+7;
char s[N],t[N];
int z[N],ext[N];
inline void z_function(char *s,int n)
{
   z[0]=n;
   for(int i=1,l=0,r=0;i<n;i++)
   {
      if(i<=r)
      {
         z[i]=min(r-i+1,z[i-l]);
      }
      while(i+z[i]<n&&s[z[i]]==s[i+z[i]])
      {
         ++z[i];
      }
      if(i+z[i]-1>r)
      {
         l=i;
         r=i+z[i]-1;
      }
   }
}
inline void exkmp(char *a,int m,char*b,int n)
{
   int l=0,r=0;
   while(ext[0]<m&&b[ext[0]]==a[ext[0]])
   {
      ++ext[0];
   }
   if(ext[0]-1>r)
      r=ext[0]-1;
   for(int i=1;i<m;i++)
   {
      if(i<=r)
      {
         ext[i]=min(r-i+1,z[i-l]);
```

```
        }
        while(i+ext[i]<m&&b[ext[i]]==a[i+ext[i]])
        {
            ++ext[i];
        }
        if(i+ext[i]-1>r)
        {
            l=i;
            r=i+ext[i]-1;
        }
    }

}
int main()
{
    scanf("%s%s",&s,&t);
    ll m=strlen(s);
    ll n=strlen(t);
    z_function(t,n);
    exkmp(s,m,t,n);
    ll res=0;
    for(register int i=0;i<n;i++)
    {
        res^=((i+1)*(z[i]+1ll));
    }
    printf("%lld\n",res);
    res=0;
    for(register int i=0;i<m;i++)
    {
        res^=((i+1)*(ext[i]+1ll));
    }
    printf("%lld",res);
}
```

快速排序

2021年3月29日　　21:48

```
#include<iostream>
#define ll long long
using namespace std;
const ll N=1e6+10;
ll num[N];
void q_sort(ll l,ll r)
{
   if(l>=r)return;
   ll k=num[(l+r)>>1];
   ll i=l-1;
   ll j=r+1;
   while(i<j)
   {
      do i++;while(num[i]<k);
      do j--;while(num[j]>k);
      if(i<j)swap(num[i],num[j]);
   }
   q_sort(l,j);
   q_sort(j+1,r);
}
int main()
{
   ll n;
   scanf("%lld",&n);
   for(ll i=0;i<n;i++)
   {
      scanf("%lld",&num[i]);
   }
   q_sort(0,n-1);
   for(ll i=0;i<n;i++)printf("%lld ",num[i]);
}
```

第k小的数：
https://www.acwing.com/problem/content/788/

```
#include<iostream>
using namespace std;
const int N=1e5+5;
int n,k;
int p[N];
int quick_sort(int l,int r,int k)
{
   if(l==r)return p[l];
   int x=p[(l+r)>>1];
   int i=l-1;
   int j=r+1;
   while(i<j)
   {
      while(p[++i]<x);
      while(p[--j]>x);
```

```
        if(i<j)swap(p[i],p[j]);
    }
    int sl=j-l+1;
    if(k<=sl)return quick_sort(l,j,k);
    return quick_sort(j+1,r,k-sl);
}
int main()
{
    scanf("%d%d",&n,&k);
    for(int i=0;i<n;i++)
    {
        scanf("%d",&p[i]);
    }
    printf("%d",quick_sort(0,n-1,k));
}
```

Two pointer

2021年4月4日 21:48

最长连续不重复子序列：

```
#include<iostream>
using namespace std;
const int N=1e5+5;
int n;
int p[N];
int num[N];
int main()
{
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      scanf("%d",&p[i]);
   }
   int res=0;
   for(int i=0,j=0;i<n;i++)
   {
      num[p[i]]++;
      while(num[p[i]]>1)
      {
         num[p[j]]--;
         j++;
      }
       res=max(res,i-j+1);
   }
   printf("%d",res);
}
```

归并排序

2021年3月29日 22:58

```
#include<iostream>
using namespace std;
const int N=1e5+5;
int num[N];
int tmp[N];
void merge_sort(int l,int r)
{
   if(l>=r)return;
   int mid=(l+r)>>1;
   merge_sort(l,mid),merge_sort(mid+1,r);
   int k=0;
   int i=l,j=mid+1;
   while(i<=mid&&j<=r)
   {
      if(num[i]<num[j])tmp[k++]=num[i++];
      else tmp[k++]=num[j++];
   }
   while(i<=mid)tmp[k++]=num[i++];
   while(j<=r)tmp[k++]=num[j++];
   for(int i=l,j=0;i<=r;i++,j++)
   num[i]=tmp[j];
}
int main()
{
   int n;
   scanf("%d",&n);
   for(int i=0;i<n;i++)
   {
      scanf("%d",&num[i]);
   }
   merge_sort(0,n-1);
   for(int i=0;i<n;i++)printf("%d ",num[i]);
}
```

二分

2021年3月30日 0:27

二分对于区间划分需要考虑边界点（mid）是否满足左、右区间的性质;

模板1:

```
int mid=l + r + 1 >>1;//加一防止出现死循环,比如l=0,r=1时(以及注意加空格)
if(check(mid))l=mid;
else r=mid-1;
```

模板二:

```
int mid=l + r >>1;
if(check(mid))r=mid;
else l=mid+1;
```

二者的思考方式是，定义某种性质，第一种模板是右区间符合性质而左区间不符合；第二个模板是左区间符合性质而右区间不符合;

例： https://www.acwing.com/problem/content/description/791/

```
#include<iostream>
using namespace std;
const int N=1e5+5;
int p[N];
int n,q;
int main()
{
   scanf("%d%d",&n,&q);
   for(int i=0;i<n;i++)
   scanf("%d",&p[i]);
   while(q--)
   {
      int k;
      scanf("%d",&k);
      int l=0,r=n-1;
      int mid;
      while(l<r)
      {
         mid=l + r >>1;
         if(p[mid]>=k)r=mid;
         else l=mid+1;
      }
      if(p[l]!=k)
      {
         printf("-1 -1\n");
      }
      else
      {
         printf("%d ",l);
         l=0,r=n-1;
         while(l<r)
         {
            mid=l + r + 1 >>1;
            if(p[mid]<=k)l=mid;
            else r=mid-1;
        }
        printf("%d\n",l);
    }
  }
}
```
注意二分求得的是两种性质的分界点，这里第一遍二分将这个性质定义为“大于等于k”，得到的就是“小于k”与“大于等于k”的分界点；第二遍二分将性质定义为“小于等于k”，得到的就是“小于等于k”与“大于k”的分界点;

https://codeforces.com/contest/1538/problem/C
```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f
#define ll long long
#define pii pair<int,int>
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 2e5 + 5;
const int mod = 1e9 + 7;
ll n;
ll a[N];
ll l, r;
ll bin1(ll i, ll j, ll v)
{
	while (i<j)
	{
		ll mid = (i + j) >> 1;
		if (a[mid] >= v)j = mid;
		else i = mid+1;
	}
	return i;
}
ll bin2(ll i, ll j, ll v)
{
	while (i < j)
	{
		ll mid = (i + j+1) >> 1;
		if (a[mid] <= v)i=mid;
		else j=mid-1;
	}
	return i;
}
void solve()
{
	cin >> n>>l>>r;
	for (long long i = 1; i <= n; i++)
	{
		cin >> a[i];
	}
	sort(a + 1, a + 1 + n);
	ll ans = 0;
	for (ll i = 1; i <=n; i++)
	{
		ll t1 = bin1(i + 1, n, l - a[i]);
		ll t2 = bin2(i + 1, n, r - a[i]);
		if(t1>0&&t1<=n&&t2>0&&t2<=n&&a[i]+a[t1]>=l&&a[i]+a[t1]
		<=r&&a[i]+a[t2]>=l&&a[i]+a[t2]<=r)
		ans += (t2 - t1+1);
	}
	cout << ans << endl;
}
int main()
{
	IOS;
	ll t;
	cin >> t;
	while (t--)
	{
		solve();
	}
}
```

https://www.luogu.com.cn/problem/P1873

在设计二分函数时，要考虑到区间的单调性，即原序列是单调递增还是单调递减

首先对序列从大到小排序，然后二分查找高度，每找到一个高度，再二分查找高于等于这个高度的最后一颗树，然后用前缀和判断所有大于这个高度的树被锯掉的部分是否满足条件

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6+5;
const int mod=1e9+7;
ll n,m;
ll h[N];
ll pre[N];
int cmp(ll& a,ll& b)
```

```
{
    return a>b;
}
ll find(ll hei)
{
    ll l=1;
    ll r=n;
    while(l<r)
    {
        ll mid=(l+r+1)/2;
        if(h[mid]>=hei)l=mid;
        else r=mid-1;
        //cout<<"in find() now l=="<<l<<" r=="<<r<<endl;
    }
    //cout<<"we found "<<l<<endl;
    return l;
}
ll bin_search1()
{
    ll l=1;
    ll r=h[1];
    while(l<r)
    {
        ll mid=(l+r+1)/2;
        ll pos=find(mid);
        if(pre[pos]-mid*pos>=m)l=mid;
        else r=mid-1;
        //cout<<"now l=="<<l<<" r=="<<r<<endl;
    }
    //cout<<"res=="<<l<<endl;
    return l;
}
int main()
{
    IOS;
    cin>>n>>m;
    for(ll i=1;i<=n;i++)
    {
        cin>>h[i];
    }
    sort(h+1,h+1+n,cmp);
    for(ll i=1;i<=n;i++)
    {
        pre[i]=pre[i-1]+h[i];
    }
    ll ans=bin_search1();
    cout<<ans<<endl;
}
```

https://www.luogu.com.cn/problem/P1024

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
```

```
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps=1e-8;
double a,b,c,d;
double f(double x)
{
    return a*x*x*x+b*x*x+c*x+d;
}
double bin(double i,double j)
{
    //cout<<i<<' '<<j<<endl;
    double l=i,r=j;
    while(r-l>eps)
    {
        //cout<<l<<' '<<r<<endl;
        double mid=(l+r)/2;
        if(f(mid)*f(i)<0)
        {
            r=mid;
        }
        else{
            l=mid;
        }
    }
    return l;
}
int main()
{
    IOS;

    cin>>a>>b>>c>>d;
    double low=f(-100);
    double i=-99;
    for(;i<100;i++)
    {
        double k=f(i);
        if(low*f(i)<0)
        {
            break;
        }
    }
    double ans1=bin(-100,i);
    double t=f(ans1+1);
    for(i=ans1+2;i<=100;i++)
    {
        double k=f(i);
        if(t*k<0)
        {
            break;
        }
    }
    double ans2=bin(ans1+1,i);
    double ans3=bin(ans2+1,100);
    cout<<fixed<<setprecision(2)<<ans1<<' '<<ans2<<' '<<ans3<<endl;
}
```

https://www.luogu.com.cn/problem/P1678

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll m, n;
ll sch[N];
ll stu[N];
ll bin(ll x)
{
    ll l = 1;
    ll r = m;
    while (l < r)
    {
        ll mid = (l + r + 1) / 2;
        if (sch[mid] <= x)l = mid;
        else r = mid - 1;
    }
    ll res = abs(sch[l] - x);
    //cout<<"find min of "<<x<<' '<<sch[l]<<endl;
    l = 1;
    r = m;
    while (l < r)
    {
        ll mid = (l + r) / 2;
        if (sch[mid] >= x)r = mid;
        else l = mid + 1;
    }
    //cout<<"find min of "<<x<<' '<<sch[l]<<endl;
    res = min(res, abs(sch[l] - x));
    //cout<<"ans final res is"<<res<<endl;
    return res;
}
int main()
{
    IOS;
    cin >> m >> n;
    for (ll i = 1; i <= m; i++)
    {
        cin >> sch[i];
    }
    sort(sch + 1, sch + 1 + m);
    ll ans = 0;
    for (ll i = 1; i <= n; i++)
    {
        cin >> stu[i];
        ans += bin(stu[i]);
    }
    cout << ans << endl;

}
```

https://www.luogu.com.cn/problem/P2440

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll n,k;
ll a[N];
ll cal(ll x)
{
	ll sum=0;
	for(ll i=1;i<=n;i++)
	{
		sum+=a[i]/x;
	}
	return sum;
}
ll bin(ll i,ll j)
{
	ll l=i,r=j;
	ll t;
	while(l<r)
	{
		ll mid=(l+r+1)/2;
		t=cal(mid);
		if(t>=k)l=mid;
		else r=mid-1;
	}
	t=cal(l);
	if(t>=k)
	return l;
	else return 0;
}
int main()
{
	IOS;
	cin>>n>>k;
	for(ll i=1;i<=n;i++)
	{
		cin>>a[i];
	}
	sort(a+1,a+1+n);
	ll ans=bin(1,100000000);
	cout<<ans<<endl;

}
```

求最大化最小值或最小化最大值问题，参照《算法竞赛进阶指南》二分章节：

https://www.luogu.com.cn/problem/P2678

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll l,n,m;
ll a[N];
ll b[N];
int valid(ll size,ll num,ll g)
{
    ll group=1;
    ll cur=0;
    for(ll i=1;i<=num;i++)
    {
        if(cur<size)cur+=b[i];
        else group++,cur=b[i];
    }
    return group>=g&&cur>=size;
}
int main()
{
    IOS;
    cin>>l>>n>>m;
    for(ll i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    a[n+1]=l;
    for(ll i=1;i<=n+1;i++)
    {
        b[i]=a[i]-a[i-1];
    }
    ll i=0,j=1e9+5;
    while(i<j)
    {
        ll mid=(i+j+1)/2;
        if(valid(mid,n+1,n+1-m))//如果在最小距离为mid的情况下，n+1个区间能分为n+1-m以
上个组（也就是说挪去的石头小于等于m)
        {
            i=mid;//说明最小距离的值可以更大
        }
        else{
            j=mid-1;
        }
    }
    cout<<i<<endl;

}
```

https://www.luogu.com.cn/problem/P3853

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll l,n,k;
ll a[N];
ll b[N];
int valid(ll size,ll num,ll g)
{
    if(size==0)return 0;
    ll group=0;
    for(ll i=1;i<=num;i++)
    {
        if(b[i]>size)//如果当前区间大于size，就每隔size设一个路障
        {
            ll temp=b[i];
                while(temp>size)
                {
                    temp-=size;
                    group++;
                }
                group++;
        }
        else{
            group++;
        }
    }
    //cout<<"group=="<<group<<endl;
    return group<=g;
}
int main()
{
    IOS;
    cin>>l>>n>>k;
    for(ll i=0;i<n;i++)
    {
        cin>>a[i];
    }
    for(ll i=1;i<=n-1;i++)
    {
        b[i]=a[i]-a[i-1];
    }
    ll i=0,j=1e7+5;
    while(i<j)
    {
        ll mid=(i+j)/2;
        if(valid(mid,n-1,n-1+k))
        {
            j=mid;
        }
        else{
            i=mid+1;
        }
        //cout<<"i=="<<i<<' '<<"j=="<<j<<endl;
    }
    cout<<i<<endl;

}
```

https://www.luogu.com.cn/problem/P1182
```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll n,m;
ll a[N];
int valid(ll size)
{
    ll group=1;
    ll rest=size;
    for(ll i=1;i<=n;i++)
    {
        if(rest>=a[i])
        {
            rest-=a[i];
        }
        else{
            group++;
            rest=size-a[i];
            if(rest<0)return 0;
        }
    }
    return group<=m;
}
int main()
{
    IOS;
    cin>>n>>m;
    for(ll i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    ll l=0,r=1e9;
    while(l<r)
    {
        ll mid=(l+r)/2;
        if(valid(mid))
        {
            r=mid;
        }
        else{
            l=mid+1;
        }
    }
    cout<<l<<endl;

}
```

https://www.luogu.com.cn/problem/P1163

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
int main()
{
    IOS;
    double a,b,c;
    cin>>a>>b>>c;
    double l=0,r=10;
    while(r-l>=eps)
    {
        double mid=(l+r)/2;
        double sum=a;
        for(ll i=1;i<=c;i++)
        {
            sum=sum*(mid+1)-b;//模拟还钱的过程，每月需要还的钱数为总数*(利率+1)，每月
还的钱数为b
        }
        if(sum>=0)//判断是否还完
        {
            r=mid;
        }
        else{
            l=mid;
        }

    }
    cout<<fixed<<setprecision(1)<<l*100<<endl;
}
```

https://www.luogu.com.cn/problem/P3743
```cpp
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-4;
double a[N],b[N];
int main()
{
    IOS;
    ll n;
    double p;
    cin>>n>>p;
    double sum=0;
    for(ll i=1;i<=n;i++)
    {
        cin>>a[i]>>b[i];
        sum+=a[i];
    }
    if(sum-p<=0)
    {
        cout<<-1<<endl;
    }
    else{
        double l=0,r=2499975000+10;
        while(r-l>eps)
        {
            double mid=(l+r)/2;
            double cost=0;
            for(ll i=1;i<=n;i++)
            {
                if(mid*a[i]>b[i])
                {
                    cost+=mid*a[i]-b[i];
                }
            }
            if(cost-mid*p>eps)
            {
                r=mid;
            }
            else{
                l=mid;
            }
        }
        cout<<l<<endl;
    }
}
```

两个数不能组成的最大数字

2022年4月4日　　17:28

给出两个大于1的数n,m，求不能表示为c1*n+c2*m的最大数字
注意到能凑出来的数字一定是二者gcd的倍数，因此若gcd(n,m)>1，则所有不是gcd的倍数的数都无法凑出来
在互质的情况下，可以根据贝祖定理证明一定有解，解为(n-1)*(m-1)-1

二维前缀和

2021年4月1日 21:34

```
#include<iostream>
using namespace std;
const int N=1e3+5;
int pre[N][N]={0};
int n,m,q;
int  main()
{
   scanf("%d%d%d",&n,&m,&q);
   for(int i=1;i<=n;i++)
   {
      for(int j=1;j<=m;j++)
      {
         int k;
         scanf("%d",&k);
         pre[i][j]=pre[i-1][j]+pre[i][j-1]-pre[i-1][j-1]+k;
      }
   }
   while(q--)
   {
      int x1,y1,x2,y2;
      scanf("%d%d%d%d",&x1,&y1,&x2,&y2);
      printf("%d\n",pre[x2][y2]-pre[x1-1][y2]-pre[x2][y1-1]+pre[x1-1][y1-1]);
   }
}
```

差分矩阵

2021年4月1日 23:08

```
#include<iostream>
using namespace std;
const int N=1e3+5;
int s[N][N],t[N][N],u[N][N];
int n,m,q;
void update(int x1,int y1,int x2,int y2,int c)
{
   t[x1][y1]+=c;
   t[x1][y2+1]-=c;
   t[x2+1][y1]-=c;
   t[x2+1][y2+1]+=c;
}
int main()
{
   scanf("%d%d%d",&n,&m,&q);
   for(int i=1;i<=n;i++)
   {
      for(int j=1;j<=m;j++)
      {
         scanf("%d",&s[i][j]);
      }
   }
   for(int i=1;i<=n;i++)
   {
      for(int j=1;j<=m;j++)
      {
         update(i,j,i,j,s[i][j]);
      }
   }
   while(q--)
   {
      int x1,y1,x2,y2,c;
      scanf("%d%d%d%d%d",&x1,&y1,&x2,&y2,&c);
      update(x1,y1,x2,y2,c);
   }
   for(int i=1;i<=n;i++)
   {
      for(int j=1;j<=m;j++)
      {
        u[i][j]=u[i-1][j]+u[i][j-1]-u[i-1][j-1]+t[i][j];
        printf("%d ",u[i][j]);
      }
      printf("\n");
   }
}
```

https://www.luogu.com.cn/problem/P3397

```
#include <iostream>
#include <algorithm>
#include <queue>
```

```
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include <string>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e6 + 500;
const int mod = 10007;
const double eps = 1e-4;
ll n,m;
ll mp[1005][1005];
ll ans[1005][1005];
void insert(ll x1,ll y1,ll x2,ll y2)
{
    mp[x1][y1]+=1;
    mp[x1][y2+1]-=1;
    mp[x2+1][y1]-=1;
    mp[x2+1][y2+1]+=1;
}
int main()
{
    IOS;
    cin >> n>>m;
    for(ll i=1;i<=m;i++)
    {
        ll a,b,c,d;
        cin>>a>>b>>c>>d;
        insert(a,b,c,d);
    }
    for(ll i=1;i<=n;i++)
    {
        for(ll j=1;j<=n;j++)
        {
            ans[i][j]=ans[i-1][j]+ans[i][j-1]-ans[i-1][j-1]+mp[i][j];
        }
    }
    for(ll i=1;i<=n;i++)
    {
        for(ll j=1;j<=n;j++)
        {
            cout<<ans[i][j]<<' ';
        }
        cout<<endl;
    }
}
```

扫描线+线段树求矩形面积并

2022年4月5日 18:28

https://zhuanlan.zhihu.com/p/103616664

https://www.acwing.com/problem/content/description/1230/

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long
#define endl '\n'
#define debug(x) cout<< #x <<" is "<<(x)<<endl;
#define pll pair<ll,ll>
#define pb push_back
#define eps 1.0E-8
const ll mod=1e8+7;
const ll N=10000+5;
ll n;
struct node{
    ll l,r,y,t;
};
vector<node>arr;
vector<ll>tmp;
struct seg{
    ll l,r,col=0,len=0;//注意col是当前区间被完全覆盖的次数
}tree[4*N];
void build(ll p,ll l,ll r)
{
    tree[p].l=l;
    tree[p].r=r;
    if(l==r)
    {
        return;
    }
    ll mid=(l+r)/2;
    build(p*2,l,mid);
    build(p*2+1,mid+1,r);
}
void pushup(ll p)
{
    if(tree[p].col>0)tree[p].len=tree[p].r-tree[p].l+1;
    else if(tree[p].l==tree[p].r)tree[p].len=0;
    else
    {
        tree[p].len=tree[p*2].len+tree[p*2+1].len;
    }
}
void update(ll p,ll l,ll r,ll c)
{

    if(tree[p].l>=l&&tree[p].r<=r)
    {
        tree[p].col+=c;
        pushup(p);
        return;
    }
    ll mid=(tree[p].l+tree[p].r)/2;
    if(l<=mid)update(p*2,l,r,c);
    if(r>mid)update(p*2+1,l,r,c);
    pushup(p);
}
int main()
{
    scanf("%lld",&n);
    ll t1=0,t2=0;
    for(ll i=0;i<n;i++)
    {
        ll a,b,c,d;
        scanf("%lld%lld%lld%lld",&a,&b,&c,&d);
        arr.pb({a,c,b,0});
        arr.pb({a,c,d,1});
    }
    sort(arr.begin(),arr.end(),[&](node a,node b){
            return a.y<b.y;
         });
    build(1,0,N);
    ll ans=0;
    /*
    2
1 0 3 3
2 2 4 4
    */
    for(ll j=0;j<arr.size()-1;j++)
    {
        auto i=arr[j];
        ll l=i.l;
        ll r=i.r;
        ll t=i.t;

        if(t==0)
        {
            update(1,l,r-1,1);
        }
        else
        {
            update(1,l,r-1,-1);
        }

        ans+=(arr[j+1].y-arr[j].y)*(tree[1].len);

    }
    printf("%lld\n",ans);


}
```

去重的一般写法

2021年4月5日 12:39

```
#include<iostream>
#include<algorithm>
#include<vector>
using namespace std;
const int N=1e5+5;
vector<int>a;
int main()
{
    int n;
    cin>>n;
    while(n--)
    {
        int k;
        cin>>k;
        a.push_back(k);
    }
    sort(a.begin(),a.end());
    a.erase(unique(a.begin(),a.end()),a.end());
    for(auto i:a)
    cout<<i<<' ';
}
```

离散化

2021年4月5日　　13:36

离散化的一般步骤是去重+二分

https://www.acwing.com/problem/content/804/

```
#include<iostream>
#include<algorithm>
using namespace std;
#define pii pair<int,int>
const int N=3e5+5;
vector<pii>a;
vector<pii> q;
vector<pii> add;
int pre[N];
int find(int i)
{
   int l=0,r=a.size()-1;
   int mid;
   while(l<r)
   {
      mid=l+r >>1;
      if(a[mid].first>=i)r=mid;
      else l=mid+1;
   }
   return l;
}
int main()
{
   int n,m;
   scanf("%d%d",&n,&m);
   for(int i=0;i<n;i++)
   {
      int x,c;
      scanf("%d%d",&x,&c);
      add.push_back({x,c});
      a.push_back({x,0});
   }
   for(int i=0;i<m;i++)
   {
      int l,r;
      scanf("%d%d",&l,&r);
      q.push_back({l,r});
      a.push_back({l,0});
      a.push_back({r,0});
   }
   sort(a.begin(),a.end());
   a.erase(unique(a.begin(),a.end()),a.end());

   for(int i=0;i<n;i++)
   {
      int x=find(add[i].first);
      a[x].second+=add[i].second;
   }
```

```
    for(int i=0;i<a.size();i++)
    {
        if(i==0)
        {
            pre[i]=a[i].second;

            continue;
        }
        pre[i]=pre[i-1]+a[i].second;

    }
    for(int i=0;i<m;i++)
    {
        printf("%d\n",pre[find(q[i].second)]-pre[find(q[i].first)-1]);
    }
}
```

杂项

2021年4月7日 20:40

C++中变量名取为y1，next会报错，因为这两个名字在某个头文件中用过

在使用scanf读入字符时，建议写成
char op[2];
scanf("%s",op);
这样会自动忽略空格和换行
如果使用%c，则不会忽略空格和换行

对于互质的两个数p，q，px+py 不能表示的最大数为pq-p-q.

来自 <https://www.cnblogs.com/Yuzao/p/7074465.html>

在写sort的cmp函数时，尽量不要用大于等于或小于等于，否则在排序区间内所有元素均相等时可能会报错:invalid comparator
比如对1，1排序，如果用小于等于，sort函数会不断将这两个值相互交换，而不会停止

Memset(a,0x3f,sizeof a)是将数组的每一个字节置为0x3f，所以如果是int，那么每个元素的值就是0x3f3f3f3f(四字节),如果是long long，那就是0x3f3f3f3f3f3f3f3f(八字节)

如果函数名取为distance可能会报错，因为这是某个库函数的名字

如果想用memset将double初始化成无穷，不能用0x3f要用0x7f

a+b=(a&b)+(a|b)

如果用set维护结构体，重载小于号时参数需要加const，并重载成友元函数

set在判重时只对你重载的那个值判重，其他值是否相同是不考虑的

Suppose we have a set of numbers and we need to find minimal possible Xor over all the pairs. It is true that, if we sort the numbers in non-descending order, the answer will be equal to
minimal Xor over neighboring numbers.
Proof:
Let a,b and c be three numbers such that a<b<c. Let the first bit where a and c differ be the kth bit, then kth bit in a must be unset and it must be set in c. Then depending upon whether kth bit is set in b or not we get (b xor c) <(a xor c) or (b xor a) <(a xor c)

来自 <https://codeforces.com/blog/entry/99031>

三元运算符和位移运算符和按位与之类的一定要加括号

a+b=a^b+2*(a&b)

__lg(x)输出以2为底x的对数

两数的公共因数也是其差值的因数
证明：设x为a与b的公因数，即a=k1*x,b=k2*x;
则a-b=(k1-k2)*x;

PI=acos(-1)或2asin(1)

1/1+1/2+1/3+...+1/C的复杂度是O(logC)

如果你在使用超过32位的位运算，别忘了转成 long long,比如(1LL<<60)

位运算的分配律：
与运算和或运算满足分配律
(a & b) | c = (a | c) & (b | c)
(a | b) & c = (a & c) | (b & c)
有异或运算的部分不满足
(a ^ b) & c = (a & c) ^ (b & c)
(a & b) ^ c ≠ (a ^ c) & (b ^ c)
(a | b) ^ c ≠ (a ^ c) | (b ^ c)
(a ^ b) | c ≠ (a | c) ^ (b | c)

因子个数

范围	最大个数	哪个数的因子数最多
1e4	64	7560
1e5	128	83160
1e6	240	720720
1e7	448	8648640
1e8	768	73513440
1e9	1344	735134400

要求数组中长度大于 1、平均值最大的区间的平均值，只需在区间长度为 2 or 3 的区间找即可
https://www.cnblogs.com/olderciyuan/p/16181301.html

如果你需要寻找将平面上一堆点连接起来的最短路径，那么：
取所有点横坐标的中点

取所有点纵坐标的中点
将其他点和这个点连起来就行

如果要将位运算和加减乘除等混用，位运算符一定要用括号包裹

霍尔定理

2022年2月5日 16:46

对于一个二分图$G(V1,V2,E)(|V1|\leqslant|V2|)$$G(V1,V2,E)(|V1|\leqslant|V2|)$，
对于$\forall X\subseteq V1$$\forall X\subseteq V1$，定义$N(X)=vj|(vi,vj)\in E,vj\in V2,vi\in X$$N(X)=vj|(vi,vj)\in E,vj\in V2,vi\in X$。
其存在$V1$$V1$的完美匹配的充要条件为$\forall X\subseteq V1,|X|\leqslant|N(X)|$$\forall X\subseteq V1,|X|\leqslant|N(X)|$。

来自 <https://www.cnblogs.com/hzoi-fengwu/p/15170667.html>

状态压缩dp

2021年4月8日 17:21

https://www.acwing.com/problem/content/93/

```
#include<iostream>
#include<algorithm>
#include<cstring>
using namespace std;
#define INF 0x3f3f3f3f
const int N=20;
int mp[N][N];
int f[1<<N][N];
int n;
int main()
{

   cin>>n;
   for(int i=0;i<n;i++)
   {
      for(int j=0;j<n;j++)
      {
         cin>>mp[i][j];
      }
   }
   memset(f,INF,sizeof(f));
   f[1][0]=0;
   for(int i=0;i<(1<<n);i++)
   {
      for(int j=0;j<n;j++)
      {
         if((i>>j)&1){//若走过的点包括第j个点
         for(int k=0;k<n;k++)
         {
            if(((i-(1<<j))>>k)&1)//和上面相似，这次是判断k有没有走过，但注意这里枚举的是走到j
之前的那个点，所以j还没走到，要从状态中减去
            {
               f[i][j]=min(f[i][j],f[i-(1<<j)][k]+mp[k][j]);
               //f[i][j]表示以j为终点，走过的点状态为i的最小长度，比如说，若i为00001，则表示
只有第一个点走过了
            }
         }
         }
      }
   }
   cout<<f[(1<<n)-1][n-1]<<endl;
}
```

Trie tree

2021年4月13日 22:50

Sample: https://www.acwing.com/problem/content/837/

```
#include<iostream>
#include<cstring>
using namespace std;
const int N=1e5+5;
int son[N][26],cnt[N],idx;//the node whose index is 0,is root,also null node
char op[2];
char str[N];
void insert(char str[])
{
   int p=0;//start from root
   for(int i=0;str[i];i++)
   {
      int t=str[i]-'a';
      if(!son[p][t])
      {
         son[p][t]=++idx;
      }
      p=son[p][t];

   }
   cnt[p]++;
}
int query(char str[])
{
   int p=0;
   for(int i=0;str[i];i++)
   {
      int t=str[i]-'a';
      if(!son[p][t])
      {
         return 0;
      }
      p=son[p][t];
   }
   return cnt[p];

}

int main()
{
   int n;
   scanf("%d",&n);
   while(n--)
   {
     scanf("%s%s",op,str);
     if(strcmp(op,"I")==0)
     {
        insert(str);
     }
     else
```

```
        {
            cout<<query(str)<<endl;
        }
    }
}
```

https://www.acwing.com/problem/content/144/

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define _for(i,a,b) for(ll i=a;i<=b;i++)
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=1e6+5;
const ll mod=1e9+7;
ll n,m;
ll trie[N][26];
ll wend[N];//结尾标记
ll tot=1;
void insert(string s)
{
    ll p=1;
    for(ll i=0;i<s.length();i++)
    {
        ll ch=s[i]-'a';
        if(trie[p][ch]==0)trie[p][ch]=++tot;
        p=trie[p][ch];
    }
    wend[p]++;
}
ll search(string s)
{
    ll p=1;
    ll res=0;
    for(ll i=0;i<s.length();i++)
    {
        p=trie[p][s[i]-'a'];
        //if(p==0)return false;
        res+=wend[p];
    }
    return res;
}
void solve()
{
    cin>>n>>m;
    while(n--)
    {
        string str;
        cin>>str;
        insert(str);
    }
    while(m--)
    {
        string str;
        cin>>str;
        cout<<search(str)<<endl;
    }
}
int main()
{
    IOS;
    solve();
}
```

https://www.acwing.com/problem/content/145/

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define _for(i,a,b) for(ll i=a;i<=b;i++)
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=7e6+5;
const ll mod=1e9+7;
ll trie[N][2];
ll en[N];
ll tot=1;
void insert(ll num)
{
    ll p=1;
    for(ll i=31;i>=0;i--)
    {
        ll t=(num&(1<<i))==0?0:1;
        if(trie[p][t]==0)trie[p][t]=++tot;
        p=trie[p][t];
    }
    en[p]=1;
}
ll search(ll num)
{
    ll p=1;
    ll res=0;
    for(ll i=31;i>=0;i--)
    {
        ll t=(num&(1<<i))==0?0:1;
        if(trie[p][!t]!=0)//尽量选择二进制位不同的方向走
        {
            res=res*2+1;
            p=trie[p][!t];
        }
        else{
            res=res*2+0;
            p=trie[p][t];
        }
    }
    return res;
}
int main()
{
    IOS;
    ll n;
    cin>>n;
    ll ans=0;
    while(n--)
    {
        ll a;
        cin>>a;
        ans=max(ans,search(a));
        insert(a);
    }
    cout<<ans<<endl;
}
```

如何在有多组样例时做到边插入边初始化:

https://codeforces.com/contest/1658/problem/D2

```
#include<bits/stdc++.h>
#define ll long long
#define ull unsigned long long
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define debug(x) cout<< #x <<" is "<< x <<endl
#define endl '\n'
#define pb push_back
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f
```

```
#define ld long double
using namespace std;
const ll N=2e6+5;
const ll M=505;
const ll mod=998244353;
ll l,r;
ll a[N];
ll trie[N][2];
ll en[N];
ll tot=1;
/*
1
1 2
0 3
*/
void insert(ll x)
{
   ll p=0;

   for(ll i=16;i>=0;i--)
   {
      ll t= ((x&(1<<i))==0?0:1);
      if(trie[p][t]==0)
      {
         trie[tot][0]=trie[tot][1]=0;
         trie[p][t]=tot++;
      }
      p=trie[p][t];
   }
   en[p]=x;
}
ll qmm(ll x)
{
   ll p=0;
   for(ll i=16;i>=0;i--)
   {
      ll t= ((x&(1<<i))==0?0:1);
      if(trie[p][t^1])p=trie[p][t^1];
      else p=trie[p][t];
   }
   return en[p]^x;
}
ll qmi(ll x)
{
   ll p=0;
   for(ll i=16;i>=0;i--)
   {
      ll t= ((x&(1<<i))==0?0:1);
      if(trie[p][t])p=trie[p][t];
      else p=trie[p][t^1];
   }
   return en[p]^x;
}
void init()
{
   trie[0][0]=trie[0][1]=0;
   tot=1;
}
int main()
{
    IOS;
    ll _;
    cin>>_;
    while(_--)
    {
        cin>>l>>r;
        init();
        ll n=r-l+1;
        for(ll i=0;i<n;i++)
        {
            cin>>a[i];
            insert(a[i]);
        }
        ll ans=0;
        for(ll i=0;i<n;i++)
        {
            ll x=a[i]^l;
            //debug(qmm(x));
            //debug(qmi(x));
            if(qmm(x)==r&&qmi(x)==l)
            {
                ans=x;
                break;
            }
        }
        cout<<ans<<endl;

    }
}
```

堆

2021年4月14日 15:38

```
#include<iostream>
using namespace std;
const int N=1e5+5;
//小根堆
int n,m;
int h[N];
int hsize;//堆节点个数
void down(int i)//将下标为i的节点向下调整
{
   int t=i;
   if(i*2<=hsize&&h[i*2]<h[t])t=i*2;//若当前节点有左孩子（完全二叉树判断有无子树只需看其
子树下标是否小于等于总结点数）且左孩子更小
   if(i*2+1<=hsize&&h[i*2+1]<h[t])t=i*2+1;//若当前节点有右孩子且右孩子更小
   if(t!=i){
      swap(h[i],h[t]);//交换节点
      down(t);//已经交换的孩子节点继续向下调整
   }
}
int main()
{
   scanf("%d%d",&n,&m);
   for(int i=1;i<=n;i++)
   {
      scanf("%d",&h[i]);
   }
   hsize=n;
   for(int i=n/2;i>=1;i--)//完全二叉树初倒数第二层及以上的节点共有n/2个，所以将这n/2个节点
一个一个向下调整即可建堆，最后一层不用调（会在建堆过程中自动调好），这种建堆方式复
杂度为O(n)

      down(i);

   while(m--)
   {
      printf("%d ",h[1]);
      //删除堆的头节点的方法是，令其最后一个节点覆盖头节点，然后删除最后一个节
点，然后新头节点向下调整
      h[1]=h[hsize];
      hsize--;
      down(1);
   }
}
```

https://www.acwing.com/problem/content/841/

```
#include<iostream>
using namespace std;
```

```
const int N=1e5+5;
//小根堆
int h[N];
int hsize;
int ph[N],hp[N];//ph[k]=i表示第k个插入的元素在堆中位置为i,hp[i]=k表示堆中位置为i的元素是
第k个被插入的元素
void heap_swap(int a,int b)//交换元素时，不仅元素值要交换，元素的映射也要交换
{
   swap(ph[hp[a]],ph[hp[b]]);//假设ph[k]=a,hp[a]=k,则ph[hp[a]]=a,那么交换之后ph[hp[a]]=b;
   swap(hp[a],hp[b]);
   swap(h[a],h[b]);

}
void down(int i)//向下调整
{
   int t=i;
   if(i*2<=hsize&&h[i*2]<h[t])t=i*2;
   if(i*2+1<=hsize&&h[i*2+1]<h[t])t=i*2+1;
   if(t!=i)
   {
      heap_swap(t,i);
      down(t);
   }
}
void up(int i)//向上调整
{
   while(i/2&&h[i/2]>h[i])
   {
      heap_swap(i,i/2);
      i/=2;
   }
}
int main()
{
   int n;
   cin>>n;
   int cnt=0;
   while(n--)
   {
      string op;
      cin>>op;
      if(op=="I")//插入元素
      {
         int a;
         cin>>a;
         h[++hsize]=a;
         cnt++;
         ph[cnt]=hsize,hp[hsize]=cnt;//记录映射
         up(hsize);

      }
      else if(op=="PM")//输出最小值
      {
            cout<<h[1]<<endl;
        }
        else if(op=="DM")//删除最小值
        {
            heap_swap(1,hsize);
            hsize--;
            down(1);
        }
        else if(op=="D")//删除第k个元素
        {
            int k;
            cin>>k;
            k=ph[k];//找到第k个元素在堆中的位置
            heap_swap(k,hsize);
            hsize--;
            up(k),down(k);//二者中只会执行其中一个
        }
        else//修改第k个元素的值
        {
            int k,x;
            cin>>k>>x;
            k=ph[k];
            h[k]=x;
            up(k),down(k);
        }
    }
}
```

约数之和

2021年4月14日 20:26

https://www.acwing.com/problem/content/99/

对于一个整数A,由唯一分解定理，A=p1^k1*p2^k2*...*pn^kn（pi为质数)
下面来考虑A的所有约数之和
首先对于每一项pi^ki，其约数个数为ki+1　（1也包括在内)
由乘法原理，A的约数个数为(k1+1)*(k2+1)*...*(kn+1)
其所有约数之和为 (p1^0+p1^1+...+p1^k1)*(p2^0+p2^1+...+p2^k2)*...*(pn^0+pn^1+...+pn^kn),它可以化为(...)+(...)+...+(...)的形式，括号中的每一项是A的一个约数（每个约数是由n个质数相乘得到的，比如其中一个约数是p1,就是由p1=p1^1*p2^0*...*pn^0得到的）
接下来看乘式中单项怎么算：
设s(p,k)=p^0+p^1+...+p^k
先看项数为偶数的情况，此时k为奇数因为下标从0开始
S(p,k)=(p^0+p^1+...+p^(k/2))+(p^(k/2+1)+...+p^k)
　　=(p^0+p^1+...+p^(k/2))+p^(k/2+1)*(p^0+p^1+...+p^(k/2))
这里是从后面那个式子中提出了一个p^(k/2+1)，注意k为奇数所以除法向下取整，所以(k/2+1)+(k/2)=k（括号中的两项分别向下取整）
所以
　　S(p,k)=(p^0+p^1+...+p^(k/2))*(1+p^(k/2+1))
　　　　=s(p,k/2)*(1+p^(k/2+1))
如果项数为奇数，就先算最后一项，然后问题就转化为算偶数项了

树的直径

2021年4月17日 15:51

https://zhuanlan.zhihu.com/p/115966044?from_voters_page=true

从树上任意一点p开始dfs，找到距其最远的点q，然后从点q开始，找到距点q最远的点w，qw就是树的直径

性质：树上任意一点所能到的最远点，一定是树的直径的端点

Sample:

https://www.acwing.com/problem/content/description/1209/

```
#include<iostream>
#include<vector>
#define ll long long
using namespace std;
const ll N=1e5+5;
ll n;
struct node{
    ll to;
    ll w;
};
vector<node>tree[N];
int vis[N];
ll dis[N];
void dfs(ll cur)
{
    for(ll i=0;i<tree[cur].size();i++)
    {
        if(!vis[tree[cur][i].to])
        {
            vis[tree[cur][i].to]=1;
            dis[tree[cur][i].to]=dis[cur]+tree[cur][i].w;
            dfs(tree[cur][i].to);
        }
    }
}
int main()
{
    cin>>n;
    for(ll i=1;i<n;i++)
    {
        ll a,b,w;
        cin>>a>>b>>w;
        tree[a].push_back({b,w});
        tree[b].push_back({a,w});
    }
    vis[1]=1;
    dfs(1);
    ll m=0;
    ll midx=0;
    for(ll i=1;i<=n;i++)
    {
        if(dis[i]>m)
        {
            m=dis[i];
            midx=i;
        }
        vis[i]=dis[i]=0;
    }

    vis[midx]=1;
    dfs(midx);

    ll ans=0;
    for(ll i=1;i<=n;i++)
    {
        if(dis[i]>ans)ans=dis[i];
    }

    cout<<10*ans+ans*(ans+1)/2;

}
```

对数运算

2022年5月23日 23:19

c++为对数运算提供了以下函数：

log是以e为底

log10是以10为底

__lg是以2为底

nim游戏

2021年5月1日 10:58

公平组合游戏（ICG）：满足：
1.两名玩家交替行动
2.两名玩家可以进行的操作是一样的
3.无法进行操作的玩家判负
这样的游戏称为公平组合游戏

有向图游戏：给定一个有向无环图，图中有一个唯一的起点，在起点上放一枚棋子，两名玩家交替地将这枚棋子沿有向边进行移动，每次可以移动一步，无法移动者判负。
将公平组合游戏的每一个局面视为有向图游戏的一个节点，就可以将公平组合游戏转化为有向图游戏。

nim游戏：给定n堆棋子，每堆棋子有a[i]个，每名玩家轮流从其中一堆中拿走若干枚（不能是0枚），最后无法操作者判负

先手必胜状态：可以将当前状态变成某一种先手必败状态（因为先手操作完以后，后手相当于先手）
先手必败状态：当前状态无论如何不能转化成任何一种先手必败状态

有如下结论：
若这n堆石子的个数a[i]满足：a[1]^a[2]^...^a[n]=0（异或和），先手必败
否则，先手必胜

Mex运算：设S表示非负整数集合，Mex(S)为不在S内的最小的非负整数
SG函数：在有向图游戏种，设节点x有k个后继节点y[1],y[2],...,y[k]，那么
SG(x)=Mex{SG(y1),SG(y2),...,SG(yk)};
没有后继节点的节点的SG值定义为0，即失败局面的SG值为0
若SG(x)=0，则先手必败，反之先手必胜
若SG(x)!=0，说明当前节点一定可以到达先手必败态（就是对手必败态），反之说明无法到达；
如果说有向图不止一张，可以将所有图起点处的SG值异或起来，若为0，必败，否则必胜
SG函数一般可以用记忆化搜索来计算。
例： https://www.acwing.com/activity/content/problem/content/963/1/

```
#include<iostream>
#include<cstring>
#include<unordered_set>
using namespace std;
const int N=1e4+5;
```

```
int f[N];//储存节点的SG值
int s[N];
int h[N];
int n,m;

int sg(int x)
{
   if(f[x]!=-1)return f[x];
   unordered_set<int>suc;
   for(int i=0;i<m;i++)
   {
      if(s[i]<=x)
      {
         suc.insert(sg(x-s[i]));
      }
   }
   for(int i=0;;i++)
   {
      if(suc.count(i)==0)return f[x]=i;
   }
}

int main()
{
   memset(f,-1,sizeof f);
   cin>>m;
   for(int i=0;i<m;i++)
   {
      cin>>s[i];
   }
   cin>>n;
   for(int i=0;i<n;i++)
   {
      cin>>h[i];
   }

   int res=0;
   for(int i=0;i<n;i++)
   {
      res^=sg(h[i]);//将其视为n张图上的nim游戏
   }
   if(res==0)cout<<"No"<<endl;
   else cout<<"Yes"<<endl;
}
```

nim博弈变形： https://vjudge.net/problem/HDU-6892

欧拉图的判别

2021年11月1日　　18:07

经过每条边恰好一次的回路称为欧拉回路，具有欧拉回路的图称为欧拉图

欧拉图没有奇度顶点

一张无向图是欧拉图，当且仅当其连通且无奇度顶点

一张有向图是欧拉图，当且仅当其所有顶点属于同一个强连通分量且每个点的出度和入度相同

区间dp

2021年5月3日 19:36

区间dp有一个比较常用的模板，请查阅下方题目

其核心思路是，欲求一段区间内的最值，可以先求其子区间的最值，然后将子区间的最值（按一定规则，具体题目具体分析）进行合并，就可以得到这一段区间的最值

https://vjudge.net/problem/LightOJ-1422

```
#include<bits/stdc++.h>
#pragma optimize (2)
#define ll long long
#define pll pair<ll,ll>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e2 + 5;
ll n;
ll c[N];
ll dp[N][N];
int t = 1;
void solve()
{
        cin >> n;
        for (ll i = 1; i <= n; i++)
        {
                cin >> c[i];
        }
        for (ll i = 1; i <= n; i++)
                dp[i][i] = 1;
        for (ll len = 2; len <= n; len++)//枚举区间长度
        {
                for (ll i = 1; i + len - 1 <= n;i++)//枚举区间起点
                {
                        ll l = i;
                        ll r = i + len - 1;
                        dp[l][r] = dp[l][r-1] + 1;//先假设一定穿衣服
                        for (ll k = l; k < r; k++)//枚举分割点
                        {
                                if(c[k]==c[r])
                                        dp[l][r] = min(dp[l][r], dp[l][k] + dp[k + 1][r-1]);//如果第k天所穿
                                        //衣服与区间最后一天相同，则这件可以一直不脱
                                        }
                }
        }
        cout << "Case " << t++ << ": ";
        cout << dp[1][n] << endl;
}
int main()
{
	IOS;
	ll _;
	cin >> _;
	while (_--)
	{
		solve();
	}
}
```

https://vjudge.net/problem/POJ-2955

```
#include<iostream>
#include<cstring>
#include<algorithm>
#define ll long long
#define pll pair<ll,ll>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e2 + 5;
string s;
ll dp[N][N];
int main()
{
	IOS;
	while (cin >> s)
	{
		if (s == "end")break;
		memset(dp, 0, sizeof dp);
		for (ll i = 0; i < s.length() - 1; i++)
		{
			if (s[i] == '(' && s[i + 1] == ')' || s[i] == '[' && s[i + 1] == ']')
			{
				dp[i][i + 1] = 2;
			}
		}
		for (ll len = 2; len <= s.length(); len++)
		{
			for (ll i = 0; i + len - 1 < s.length(); i++)
			{
				ll l = i;
				ll r = i + len - 1;
				if (l + 1 <= r - 1) {
					if (s[l] == '(' && s[r] == ')' || s[l] == '[' && s[r] == ']')
						dp[l][r] = dp[l + 1][r - 1] + 2;
				}
				for (ll k = l; k <= r; k++)
				{
					dp[l][r] = max(dp[l][r], dp[l][k] + dp[k + 1][r]);
				}
			}
		}
		/*for (ll l = 0; l <= s.length(); l++)
		{
			for (ll r = l; r <= s.length(); r++)
			{
				cout <<'('<< l<<' '<<r<<')'<<dp[l][r] << ' ';
			}
```

```
				cout << endl;
			}*/
			cout << dp[0][s.length() - 1] << endl;
		}
}
```
https://vjudge.net/problem/CodeForces-149D
```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
const ll N = 7e2 + 5;
const ll mod = 1e9 + 7;
ll dp[N][N][3][3] = { 0 };
ll match[N] = { 0 };
/*
dp[l][r][x][y]表示区间[l,r]之间、左括号颜色为x且右括号颜色为y的括号涂色方案数
match[i]表示与括号i相匹配的括号的下标

*/
int main()
{

	string s;
	cin >> s;
	ll n = s.length();
	stack<ll>st;
	stack<ll>num;
	for (ll i = 0; i < n; i++)
	{
		if (st.empty())
		{
			st.push(s[i]);
			num.push(i);
		}
		else if (s[i] == ')' && st.top() == '(')
		{
			ll x = i;
			ll y = num.top();
			match[x] = y;
			match[y] = x;
			st.pop();
			num.pop();

		}
		else
		{
			st.push(s[i]);
			num.push(i);
		}
	}
	for (ll i = 0; i < s.length() - 1; i++)
	{
		if (s[i] == '(' && s[i + 1] == ')')
		{
			ll y = i;
			ll x = i + 1;
			dp[y][x][0][1] = 1;
```

```
                dp[y][x][0][2] = 1;
                dp[y][x][1][0] = 1;
                dp[y][x][2][0] = 1;
            }
        }
        for (ll i = s.length() - 1; i >= 0; i--)
        {
            for (ll j = i + 1; j < s.length(); j++)
            {
                if (match[i] == j) {//当前区间形如：(...)
                    for (ll x = 1; x <= 2; x++)//左括号涂色右括号不涂色
                    {
                        for (ll t = 0; t <= 2; t++) {
                            for (ll r = 0; r <= 2; r++) {
                                if (x != t && i + 1 < j - 1)
                                    dp[i][j][x][0] += dp[i + 1][j - 1][t][r] % mod;
                                //cout << '*'<<dp[0][s.length() - 1][x][0] <<' '<<x<<'
                                '<<0<< ' ' << t << ' ' << r <<' '<<dp[i+1][j-1][t][r]<< endl;
                            }
                        }
                    }

                    for (ll x = 1; x <= 2; x++)//左括号涂色右括号不涂色
                    {
                        for (ll t = 0; t <= 2; t++) {
                            for (ll r = 0; r <= 2; r++) {
                                if (x != r && i + 1 < j - 1)
                                    dp[i][j][0][x] += dp[i + 1][j - 1][t][r] % mod;
                            }
                        }
                    }


                }
                else//当前区间形如：(...( or )...( or )...(
                {
                    ll k = match[i];//分成[i,k],[k,j]两部分
                    if (match[i] > j || match[i] < i || match[j]<i || match[j]>j)
                    {
                        continue;
                    }
                    for (ll t1 = 0; t1 <= 2; t1++)
                    {
                        for (ll t2 = 0; t2 <= 2; t2++)
                        {
                            for (ll c1 = 0; c1 <= 2; c1++)
                            {
                                for (ll c2 = 0; c2 <= 2; c2++)
                                {
                                    if (c1 != c2 || (c1 == 0 && c2 == 0)) {
                                        dp[i][j][t1][t2] += ((dp[i][k][t1][c1] % mod) *
                                        (dp[k + 1][j][c2][t2] % mod)) % mod;
                                    }
                                    //cout << '*' << ' ' << dp[i][j][t1][t2] << ' ' << c1 << ' '
                                    << c2 << endl;
```

```
                                        }
                                    }
                                }
                            }

                        }
                    }
        }

        ll res = 0;
        for (ll t1 = 0; t1 <= 2; t1++)
        {
                for (ll t2 = 0; t2 <= 2; t2++)
                {
                        //cout << dp[0][s.length() - 1][t1][t2]<<' '<<t1<<' '<<t2<< endl;
                        res += dp[0][s.length() - 1][t1][t2] % mod;
                }
        }

        cout << res % mod << endl;

}
```
https://vjudge.net/problem/POJ-1651
```
#include<iostream>
#include<algorithm>
#define ll long long
using namespace std;
const ll N = 1e3 + 5;
ll dp[N][N];
/*
dp[l][r] means the minimum score in [l,r]

*/
ll c[N];
ll n;
int main()
{
        cin >> n;
        for (ll i = 1; i <= n; i++)
        {
                cin >> c[i];
        }

        for (ll i = n; i >= 1; i--)
        {
                for (ll j = i; j <= n; j++)
                {
                        if (i + 1 < j - 1) {
                                /*ll takel = dp[i + 1][j - 1] + c[i + 1] * c[i] * c[j - 1] + c[i] * c[j] * c[j - 1];
                                ll taker = dp[i + 1][j - 1] + c[j - 1] * c[i + 1] * c[j] + c[i] * c[i + 1] * c[j];
                                dp[i][j] = min(takel, taker);*/
                                dp[i][j] = 1e9;
                                for (ll k = i + 1; k <= j - 1; k++)
                                {
                                        ll t = c[k] * c[i] * c[j];
                                        dp[i][j] = min(dp[i][j], dp[i][k] + t + dp[k][j]);
                                }
```

```
				}
				else if (i + 1 == j - 1)
				{
					dp[i][j] = c[i] * c[i + 1] * c[j];
				}


			}
		}
		cout << dp[1][n] << endl;

}
```

https://vjudge.net/problem/HDU-4283

```
#include<iostream>
#include<algorithm>
#include<cstring>
#define ll long long
using namespace std;
const ll N = 1e2 + 5;
ll dp[N][N];
ll n;
ll d[N];
ll pre[N];
/*
dp[l][r] 表示区间[l,r]内的最小值
栈的操作可以改变一些元素的相对顺序
接下来考虑如何划分子区间
考虑第一个人的出场情况，他可以是第一个、第二个、。。。第n个出场，
那么假设第一个人第k个出场，那么他之后的2到k-1个人一定在他之前上场，无论区间[k-1,2]内
部元素的相对位置是什么
所以根据这个事实，可以将区间[l,r]划分成[l,k-1],k,[k+1,r]
*/
ll t = 1;
int main()
{
	ll testcase;
	cin >> testcase;
	while (testcase--)
	{
		cin >> n;
		memset(dp, 0, sizeof dp);
		memset(pre, 0, sizeof pre);
		memset(d, 0, sizeof d);
		for (ll i = 1; i <= n; i++)
		{
			cin >> d[i];
			pre[i] = pre[i - 1] + d[i];
		}

		for (ll s=n;s>=1;s--)
		{
			for (ll t=s;t<=n;t++)
			{
				if (s == t) {
					dp[s][t] = 0;
					continue;
				}
				dp[s][t] = dp[s + 1][t] + pre[t] - pre[s];
				//dp[s][t] = min(dp[s][t], dp[s + 1][t] + d[s] * (t - s));
				//当当前区间第一个人是当前区间第一个时要特判一下
				for (ll k = 2; k <= t-s+1; k++)
				{
					ll v = dp[s+1][s+k-1] + (pre[t] - pre[s+k-1]) * k + (k-1) * d[s] + dp[s+k][t];
					dp[s][t] = min(dp[s][t], v);
				}
			}
		}

		cout << "Case #" << t++<<": ";
		cout << dp[1][n] << endl;
	}
}
```
https://vjudge.net/problem/HDU-2476
```
#include<iostream>
#include<algorithm>
#include<cstring>
#define ll long long
using namespace std;
const ll N = 1e2 + 5;
ll dp[N][N];
ll ans[N];
/*
本题可以用三维dp做，也可以分解为两次dp
首先考虑从一个空字符串转换为b的最小转换次数，设dp[l][r]为区间[l,r]内空字符串转换为b的最小转换次数
不考虑其他情况，直接进行转换时，dp[l][r]=dp[l+1][r]+1,即，将整个区间先染成a[l]，然后加上子区间的最小转换次数
假设存在两端字符相同的子区间，即存在l<k<r使得a[l]==a[k]，那么此时[l,k]这部分字符串就可以单独拿出来涂，即dp[l][r]=dp[l+1][k]+dp[k+1][r]
然后定义ans[i]为区间[0,i]内字符串a转换为字符串b的最小转换次数，只需要按位比较字符决定是否转换即可

*/
int main()
{
	string a, b;
	while (cin >> a >> b)
	{
		memset(dp,0, sizeof dp);
		memset(ans, 0, sizeof ans);
		ll n = a.length()-1;
		for (ll i = 0; i <= n; i++)dp[i][i] = 1;
		for (ll i = n; i >= 0; i--)
		{
			for (ll j = i+1; j <= n; j++)
```

```
                        {

                                dp[i][j] = dp[i + 1][j] + 1;
                                for (ll k = i + 1; k <= j; k++)//k是可以等于j的，此时相当于区间两边字符本
                                来就相等
                                {
                                        if(b[k]==b[i])
                                        dp[i][j] = min(dp[i][j], dp[i+1][k] + dp[k + 1][j]);
                                }
                        }
                }
                for (ll i = 0; i <= n; i++)
                {
                        ans[i] = dp[0][i];
                }
                for (ll i = 0; i <= n; i++)
                {
                        if (a[i] == b[i])
                        {
                                ans[i] = min(ans[i], ans[i - 1]);
                        }
                        else
                        {
                                for (ll j = 0; j < i; j++)
                                {
                                        ans[i] = min(ans[i], ans[j] + dp[j+1][i]);
                                }
                        }
                }
                cout << ans[n] << endl;
        }
}
```

最短路问题与差分约束系统

2021年5月16日　　11:22

对于单源最短路，若所有边权均为正数，可以使用朴素dijkstra算法(稠密图适用)，复杂度为O(v^2);或者堆优化dijkstra算法（稀疏图适用），复杂度为O(elogv)；若存在负权边，可以使用bellman-ford算法，复杂度为O(ve)，或者spfa算法，一般情况下复杂度为O(v)，最坏情况下为O(ve);
对于多源汇最短路，有floyd算法，复杂度为O(v^3)

1、朴素dijkstra：
https://www.acwing.com/problem/content/851/

```
#include<iostream>
#include<algorithm>
#include<cstring>
using namespace std;
const int N=505;
int mp[N][N];
int dis[N];
int st[N];
int n,m;
int dijkstra()
{
   memset(dis,0x3f,sizeof dis);
   dis[1]=0;
   for(int i=0;i<n;i++)
   {
      int t=-1;
      for(int j=1;j<=n;j++)
      {
         if(!st[j]&&(t==-1||dis[t]>dis[j]))
         {
            t=j;
         }
      }
      st[t]=1;
      for(int j=1;j<=n;j++)
      {
         dis[j]=min(dis[j],dis[t]+mp[t][j]);
      }
   }
   if(dis[n]==0x3f3f3f3f)return -1;
   else return dis[n];
}
int main()
{
   cin>>n>>m;
   memset(mp,0x3f,sizeof mp);
   while(m--)
   {
      int x,y,z;
      cin>>x>>y>>z;
      mp[x][y]=min(mp[x][y],z);
```

```
    }
    int t=dijkstra();
    cout<<t<<endl;
}
```

2、堆优化dijkstra：
https://www.acwing.com/problem/content/852/

```
#include<iostream>
#include<algorithm>
#include<cstring>
#include<queue>
#include<vector>
#define pii pair<int,int>
const int N = 1e6 + 10;
using namespace std;

vector<pii>g[N];
int dis[N];
int st[N];
int n, m;


int heap_dijkstra()
{
    memset(dis, 0x3f, sizeof dis);
    dis[1] = 0;
    priority_queue<pii, vector<pii>, greater<pii> >heap;
    heap.push({0,1});//注意小根堆根据第一关键词排序，故必须距离在前标号在后
    while (heap.size())
    {
        auto t = heap.top();
        heap.pop();
        int ver = t.second;
        if (st[ver])continue;
        st[ver] = 1;
        for(auto i:g[ver])
        {
            if(dis[i.first]>dis[ver]+i.second)
            {
                dis[i.first]=dis[ver]+i.second;
                heap.push({dis[i.first],i.first});
            }
        }


    }

    if (dis[n] == 0x3f3f3f3f)return -1;
    else return dis[n];
}
int main()
{
    cin >> n >> m;
    while (m--)
    {
        int x, y, z;
        cin >> x >> y >> z;
        g[x].push_back({y,z});
    }
    cout << heap_dijkstra() << endl;
}
```
用数据模拟邻接表会快很多：
```
#include<iostream>
#include<algorithm>
#include<cstring>
#include<queue>
#include<vector>
#define pii pair<int,int>
const int N = 1e6 + 10;
using namespace std;

int dis[N];
int st[N];
int n, m;

int h[N], ne[N], e[N], w[N], idx;

void add(int x, int y, int z)
{
    e[idx] = y;
    w[idx] = z;
    ne[idx] = h[x];
    h[x] = idx++;
}
int heap_dijkstra()
{
    memset(dis, 0x3f, sizeof dis);
    dis[1] = 0;
    priority_queue<pii, vector<pii>, greater<pii> >heap;
    heap.push({ 0,1 });
    while (heap.size())
    {
        auto t = heap.top();
        heap.pop();
        int ver = t.second;
        if (st[ver])continue;
        st[ver] = 1;
        for (int i = h[ver]; i != -1; i = ne[i])
        {
            int j = e[i];
            if (dis[j] > dis[ver] + w[i])
            {
                dis[j] = dis[ver] + w[i];
                heap.push({dis[j],j});
            }
        }

    }

    if (dis[n] == 0x3f3f3f3f)return -1;
    else return dis[n];
}
```

```
int main()
{
   ios::sync_with_stdio(false);
   cin.tie(0);
   cin >> n >> m;
   memset(h, -1, sizeof h);
   while (m--)
   {
      int x, y, z;
      cin >> x >> y >> z;
      add(x, y, z);
   }
   cout << heap_dijkstra() << endl;
}
```

3、Bellman-ford 算法：

https://www.acwing.com/problem/content/855/

```
#include<iostream>
#include<cstring>
#include<algorithm>
const int N=1e5+5;
using namespace std;
int dis[N];
int backup[N];
int n,m,k;
struct node{
   int a,b,w;

}edge[N];
int bellman_ford()
{
   memset(dis,0x3f,sizeof dis);
   dis[1]=0;
   for(int i=0;i<k;i++)//循环k次表示寻找边数不超过k的最短路径
   {
      memcpy(backup,dis,sizeof dis);//防止串联更新（即前面的边更新后影响后面更新的边）
      for(int j=0;j<m;j++)
      {
         int a=edge[j].a;
         int b=edge[j].b;
         int w=edge[j].w;
         dis[b]=min(dis[b],backup[a]+w);
      }
   }
   if(dis[n]>0x3f3f3f3f/2)return -1;//存在负边时，可能存在两条边之间距离为INF，却被更新成INF-x
的情况
   else return dis[n];
}
int main()
{
   cin>>n>>m>>k;
   for(int i=0;i<m;i++)
   {
      int x,y,z;
```

```
        cin>>x>>y>>z;
        edge[i]={x,y,z};

    }
    int t=bellman_ford();
    if(t==-1)cout<<"impossible"<<endl;
    else cout<<t<<endl;
}
```

4、spfa算法：

https://www.acwing.com/problem/content/description/853/

```
#include<iostream>
#include<algorithm>
#include<cstring>
#include<queue>
#define pii pair<int,int>
using namespace std;

const int N=1e5+5;

vector<pii>g[N];
int dis[N];
int vis[N];
int n,m;
int spfa()
{
    memset(dis,0x3f,sizeof dis);
    dis[1]=0;
    queue<pii>que;
    que.push({0,1});
    vis[1]=1;
    while(!que.empty())
    {
        auto t=que.front();
        que.pop();
        vis[t.second]=0;
        for(auto i:g[t.second])
        {
            if(dis[i.first]>dis[t.second]+i.second)
            {
                dis[i.first]=dis[t.second]+i.second;
                if(!vis[i.first])
                {
                    que.push({dis[i.first],i.first});
                    vis[i.first]=1;
                }
            }
        }
    }
    if(dis[n]==0x3f3f3f3f)return -1;
    else return dis[n];
}
int main()
{
   cin>>n>>m;
   for(int i=0;i<m;i++)
```

```
  {
     int a,b,c;
     cin>>a>>b>>c;
     g[a].push_back({b,c});
  }
  int t=spfa();
  if(t==-1)cout<<"impossible"<<endl;
  else cout<<t<<endl;
}
```

5、spfa判负环：
https://www.acwing.com/problem/content/854/

```
#include<iostream>
#include<algorithm>
#include<cstring>
#include<queue>
#define pii pair<int,int>
using namespace std;

const int N=1e5+5;

vector<pii>g[N];
int dis[N];
int vis[N];
int cnt[N];//记录最短路径的边数，由于1个点到第n个点的最短路径最多经过n-1条边，如果迭代中某一次边数等于n，说明存在负环，
int n,m;
int spfa()
{
   memset(dis,0x3f,sizeof dis);
   dis[1]=0;//其实不需要初始化dis数组，因为如果存在负环，dis数组一定会被更新成负无穷（如果不中断的话）
   queue<pii>que;
   for(int i=1;i<=n;i++)//负环不一定和节点1相连，所以要把所有点放进队列
   {
      que.push({0,i});
      vis[i]=1;
   }
   while(!que.empty())
   {
      auto t=que.front();
      que.pop();
      vis[t.second]=0;
      for(auto i:g[t.second])
      {
         if(dis[i.first]>dis[t.second]+i.second)
         {
            dis[i.first]=dis[t.second]+i.second;
            cnt[i.first]=cnt[t.second]+1;
            if(cnt[i.first]>=n)return 0;
            if(!vis[i.first])
            {
               que.push({dis[i.first],i.first});
               vis[i.first]=1;
            }
         }
      }
   }
  return 1;
}
int main()
{
  cin>>n>>m;
  for(int i=0;i<m;i++)
  {
     int a,b,c;
     cin>>a>>b>>c;
     g[a].push_back({b,c});
  }
  int t=spfa();
  if(t==1)cout<<"No"<<endl;
  else cout<<"Yes"<<endl;
}
```

6、floyd:

https://www.acwing.com/problem/content/856/

```
#include<iostream>
#include<algorithm>
using namespace std;
const int N=205;
const int INF = 0x3f3f3f3f;
int d[N][N];
int n,m,q;
void floyd()
{
   for(int k=1;k<=n;k++)//从i到j只经过前k个点的最短路径
   {
      for(int i=1;i<=n;i++)
      {
         for(int j=1;j<=n;j++)
         {
            d[i][j]=min(d[i][j],d[i][k]+d[k][j]);
         }
      }
   }
}
int main()
{
   cin>>n>>m>>q;
   for(int i=1;i<=n;i++)
   {
      for(int j=1;j<=n;j++)
      {
         if(i==j)d[i][j]=0;
         else d[i][j]=INF;
      }
   }
   for(int i=0;i<m;i++)
   {
      int a,b,c;
      cin>>a>>b>>c;
```

```
        d[a][b]=min(d[a][b],c);
    }
    floyd();
    while(q--)
    {
        int a,b;
        cin>>a>>b;
        if(d[a][b]>INF/2)cout<<"impossible"<<endl;
        else cout<<d[a][b]<<endl;
    }
}
```

例题：

https://vjudge.net/problem/POJ-2387

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#define ll long long
#define pll pair<ll,ll>
using namespace std;
const ll N = 1e5 + 5;
ll t, n;
ll dis[N];
ll st[N];
vector<pll>g[N];
ll heap_dijkstra()
{
        memset(dis, 0x3f, sizeof dis);
        dis[1] = 0;
        priority_queue<pll, vector<pll>, greater<pll> >heap;
        heap.push({0,1});
        while (!heap.empty())
        {
                pll t = heap.top();
                heap.pop();
                ll ver = t.second;
                if (st[ver] == 1)continue;
                st[ver] = 1;
                for (vector<pll>::iterator i=g[ver].begin();i!=g[ver].end();i++)
                {
                        ll d = (*i).first;
                        ll v = (*i).second;
                        if (dis[v] > dis[ver] + d)
                        {
                                dis[v] = dis[ver] + d;
                                heap.push({ dis[v],v });
                        }
                }

        }
        return dis[n];

}
int main()
{
        cin >> t >> n;
        while (t--)
        {
                ll a, b, w;
                cin >> a >> b >> w;
                g[a].push_back({ w,b });
                g[b].push_back({ w,a });

        }
        cout << heap_dijkstra() << endl;
}
```

https://vjudge.net/problem/POJ-2253

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#define ll long long
#define pll pair<ll,ll>
#define pdd pair<double,double>
using namespace std;
const ll N = 2e2 + 5;
/*
题意为，求从石块1到石块2所有可能路径中最长边的最小值
*/
ll n;
pdd point[N];
double g[N][N];
int cas = 1;
double distance(pdd& a, pdd& b)
{
        return sqrt((a.first - b.first) * (a.first - b.first) + (a.second - b.second) * (a.second - b.second));
}
void floyd()
{
        for (ll k = 1; k <= n; k++)
        {
                for (ll i = 1; i <= n; i++)
                {
                        for (ll j = 1; j <= n; j++)
                        {
```

```
                        g[i][j] = min(g[i][j], max(g[i][k],g[k][j]));
                        //注意这里g[i][j]的意义和原始floyd不同，其意义为i到j所有通路中最长边的
                        最小值
                        //所以在状态转移时，先寻找i到k、k到j中所有通路的最长边最小值中较大
                        的那个作为整条通路的最长边，再与当前答案作比较，取较小的那个
                    }
                }
            }
}
int main()
{
            while (cin >> n)
            {
                        if (n == 0)break;
                        memset(g, 0, sizeof g);
                        for (ll i = 1; i <= n; i++)
                        {
                                    cin >> point[i].first >> point[i].second;
                        }
                        for (ll i = 1; i <= n; i++)
                        {
                                    for (ll j = 1; j <= n; j++)
                                    {
                                                double k = sqrt((point[i].first - point[j].first) * (point[i].first - point[j].first) +
                                                (point[i].second - point[j].second) * (point[i].second - point[j].second));
                                                g[i][j] = k;
                                                g[j][i] = k;

                                    }
                        }
                        floyd();
                        cout << "Scenario #" << cas++ << endl;
                        cout << fixed << setprecision(3) << "Frog Distance = " << g[1][2]<< endl<<endl;
            }
}
```

https://vjudge.net/problem/POJ-1797

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdd pair<double,double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e3 + 5;
ll n, m;
ll dis[N];
ll mp[N][N];
ll st[N];
ll cas = 1;
/*
就是上一题反过来，dis[i]定义为从1到i所有路径最小边最大值
*/
void dijkstra()
{
	for (ll i = 1; i <= n; i++)
	{
		dis[i] = mp[1][i];
	}
	dis[1] = 0;
	for (ll i = 0; i < n; i++)
	{
		ll t = -1;
		for (ll j = 1; j <= n; j++)
		{
			if (!st[j] && (t == -1 || dis[t] < dis[j]))//注意这里要找最大值
			{
				t = j;
			}
		}
		//cout << "t==" << t << endl;
		st[t] = 1;
		for (ll j = 1; j <= n; j++)
		{
			if (dis[j] < min(dis[t], mp[t][j])&&!st[j])
			{
				dis[j] = min(dis[t], mp[t][j]);
			}
			//cout << dis[j] << ' '<<"1-->" << j << endl;
		}
	}
}
int main()
{
	IOS;
	ll t;
	cin >> t;
	while (t--) {
		cin >> n >> m;
		memset(st, 0, sizeof st);
		memset(mp, 0, sizeof mp);
		for (ll i = 0; i < m; i++)
		{
			ll a, b, c;
			cin >> a >> b >> c;
			mp[a][b] = c;
			mp[b][a] = c;
		}
		dijkstra();
		cout << "Scenario #" << cas++ << ":"<<endl;
		cout << dis[n] << endl<<endl;
	}

}
```
https://vjudge.net/problem/POJ-3268

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
//#define INF 4557430888798830399
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdd pair<double,double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e3 + 5;
ll n, m, x;
ll g[N][N];
ll dis[N];
ll dis2[N];
ll st[N];
/*
首先注意是有向图，unidirection的意思是有向的，前缀是uni-
首先正向跑一遍dijkstra，可以求出x到其他点的最短距离，这是回程时的距离
然后将所有边倒过来跑一遍dijkstra，就可以求出所有点到x的最短距离

*/
void dijkstra()
{
	memset(dis, 0x3f, sizeof dis);
	memset(st, 0, sizeof st);
	dis[x] = 0;
	priority_queue<pll, vector<pll>, greater<pll> >heap;
	heap.push({ 0,x });
	while (!heap.empty())
	{
		pll t = heap.top();
		heap.pop();
		ll ver = t.second;
		if (st[ver])continue;
		st[ver] = 1;
		for (ll i = 1; i <= n; i++)
		{
			if (dis[i] > dis[ver] + g[ver][i])
			{
				dis[i] = dis[ver] + g[ver][i];
				heap.push({ dis[i],i });
			}
		}
	}
	//cout << endl;
}
void dijkstra2()
{
	memset(dis2, 0x3f, sizeof dis2);
	memset(st, 0, sizeof st);
	dis2[x] = 0;
	priority_queue<pll, vector<pll>, greater<pll> >heap;
	heap.push({ 0,x });
	while (!heap.empty())
	{
		pll t = heap.top();
		heap.pop();
		ll ver = t.second;
		if (st[ver])continue;
		st[ver] = 1;
		for (ll i = 1; i <= n; i++)
		{
			if (dis2[i] > dis2[ver] + g[ver][i])
			{
				dis2[i] = dis2[ver] + g[ver][i];
				heap.push({ dis2[i],i });
			}
		}
	}
}
void trans()
{
	for (ll i = 1; i <= n; i++)
	{
		for (ll j = i; j <= n; j++)
		{
			swap(g[i][j], g[j][i]);
		}
	}
}
int main()
{
	IOS;
	cin >> n >> m >> x;
	memset(g,0x3f,sizeof g);
	for (ll i = 1; i <= n; i++)
	{
		g[i][i] = 0;
	}
	for (ll i = 0; i < m; i++)
	{
		ll a, b, x;
		cin >> a >> b >> x;
		g[a][b] = x;
	}
	dijkstra();
	trans();
	dijkstra2();
	ll ans = 0;
	for (ll i = 1; i <= n; i++)
	{
		//cout << "dis[" << i << "]==" << dis[i] << endl;
		//cout << "dis2[" << i << "]==" << dis2[i] << endl;
		if(dis[i]!=INF&&dis2[i]!=INF)
		ans = max(ans, dis[i] + dis2[i]);
	}
	cout << ans << endl;
}
```

https://vjudge.net/problem/POJ-1860

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
//#define INF 4557430888798830399
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdd pair<double,double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e3 + 5;
ll n, m, s;
double v;
double dis[N];
ll vis[N];
ll cnt[N];
struct node {
	double rate;
	double comm;
};
vector<pair<node, ll> >g[N];
/*
以货币为点，以汇率、佣金为边权，观察到只要存在正权环路，就可以利用这个环路将资金增加到无限大

*/
int spfa()
{
	memset(dis, 0, sizeof dis);
	queue<ll>q;
	q.push(s);
	vis[s] = 1;
	dis[s] = v;
	while (!q.empty())
	{
		ll t = q.front();
		q.pop();
		vis[t] = 0;
		for (vector<pair<node, ll> >::iterator i = g[t].begin(); i != g[t].end(); i++)
		{
			ll ver = (*i).second;
			node edge = (*i).first;
			if ((dis[t] - edge.comm) * edge.rate > dis[ver])
			{
				dis[ver] = (dis[t] - edge.comm) * edge.rate;
				cnt[ver] = cnt[t] + 1;
				if (!vis[ver])
				{
					vis[ver] = 1;
					q.push(ver);
				}
```

```
                    if (cnt[ver] >= n)return 1;
                    //cout << "dis[ver]==" << dis[ver] << ' ' << "ver==" << ver << endl;
                }
            }
        }
        return 0;
}
int main()
{
        IOS;
        cin >> n >> m >> s >> v;
        for (ll i = 0; i < m; i++)
        {
            ll t1, t2;
            double r1, c1, r2, c2;
            cin >> t1 >> t2 >> r1 >> c1 >> r2 >> c2;
            node tmp = { r1,c1 };
            g[t1].push_back({ tmp,t2 });
            tmp = { r2,c2 };
            g[t2].push_back({ tmp,t1 });
        }
        if (spfa())cout << "YES" << endl;
        else cout << "NO" << endl;
}
```
https://vjudge.net/problem/POJ-3259
```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
//#define INF 4557430888798830399
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e3 + 5;
ll n, m, w;
vector<pll>g[N];
ll cnt[N];
ll dis[N];
ll vis[N];

/*
与上一题差不多，如果我们可以找到一个负权回路，就可以利用这个负权回路将用时回溯到负无穷

*/
int spfa()
{
        queue<ll>q;
        for (ll i = 1; i <= n; i++)
        {
```

```
			q.push(i);//由于题目中没有规定起点，所以将所有点入队
			vis[i] = 1;
		}
		while (!q.empty())
		{
			ll t = q.front();
			q.pop();
			vis[t] = 0;
			for (vector<pll>::iterator i = g[t].begin(); i != g[t].end(); i++)
			{
				ll ver = (*i).second;
				ll d = (*i).first;
				if (dis[ver] > dis[t] + d)
				{
					dis[ver] = dis[t] + d;
					cnt[ver] = cnt[t] + 1;
					if (cnt[ver] >= n)return 1;
					if (!vis[ver])
					{
						vis[ver] = 1;
						q.push(ver);
					}
				}
			}
		}
		return 0;
	}
	int main()
	{
		IOS;
		ll f;
		cin >> f;
		while (f--)
		{
			cin >> n >> m >> w;
			for (ll i = 1; i <= n; i++)
			{
				g[i].clear();
			}
			memset(cnt, 0, sizeof cnt);
			for (ll i = 0; i < m; i++)
			{
				ll a, b, w;
				cin >> a >> b >> w;
				pll tmp(w,b);
				g[a].push_back(tmp);
				tmp = { w,a };
				g[b].push_back(tmp);
			}
			for (ll i = 0; i < w; i++)
			{
				ll a, b, w;
				cin >> a >> b >> w;
				pll tmp(-w,b);
				g[a].push_back(tmp);
			}
		if (spfa())cout << "YES" << endl;
		else cout << "NO" << endl;
	}
}
```
https://vjudge.net/problem/POJ-1502
```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 2e5 + 5;
ll n;
ll st[N];
ll dis[N];
ll ans = 0;
vector<pll>g[N];
/*
找从第一个点到其他点的最短路径的最大值即可
*/
void heap_dijkstra()
{
	priority_queue<pll, vector<pll>, greater<pll> >q;
	memset(dis, 0x3f, sizeof dis);
	dis[1] = 0;
	q.push({ 0,1 });
	while (!q.empty())
	{
		pll t = q.top();
		q.pop();
		ll ver = t.second;
		if (st[ver] == 1)continue;
		st[ver] = 1;
		for (vector<pll>::iterator i = g[ver].begin(); i != g[ver].end(); i++)
		{
			ll v = (*i).second;
			ll d = (*i).first;
			if (dis[v] > dis[ver] + d)
			{
				dis[v] = dis[ver] + d;
				if (!st[v])
				{
					q.push({ dis[v],v });
				}
			}
		}
	}
	for (ll i = 1; i <= n; i++)
	{
		//cout << "dis[" << i << "]==" << dis[i] << endl;
		ans = max(ans, dis[i]);
	}
}
ll trans(string s)
{
	ll num = 0;
	for (ll i = 0; i < s.length(); i++)
	{
		num = num * 10 + s[i] - '0';
	}
	return num;
}
int main()
{
	IOS;
	cin >> n;
	for (ll i = 1; i <= n; i++)
	{
		for (ll j = 1; j < i; j++)
		{
			string ch;
			cin >> ch;
			if (ch != "x") {
				ll num = trans(ch);
				g[i].push_back({ num,j });
				g[j].push_back({ num,i });
			}
		}
	}
	heap_dijkstra();
	cout << ans << endl;
}
```

https://vjudge.net/problem/POJ-3660

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#define INF 0x3f3f3f3f
#define ll int
#define pll pair<ll,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e2 + 5;
ll n, m;
ll g[N][N];
/*
将胜负关系建成有向图，对于一个点，所有可以到达它的点和所有它可以到达的点的数量合起来若为n-1，那么这个点的排名就可以确定
*/
void floyd()
{
	for (ll k = 1; k <= n; k++)
	{
		for (ll i = 1; i <= n; i++)
		{
			for (ll j = 1; j <= n; j++)
			{
				g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
			}
		}
	}
}
int main()
{
	IOS;
	cin >> n >> m;
	memset(g, 0x3f, sizeof g);
	for (ll i = 1; i <= n; i++)
	{
		g[i][i] = 0;
	}
	for (ll i = 0; i < m; i++)
	{
		ll a, b;
		cin >> a >> b;
		g[a][b] = 1;
	}
	floyd();
	ll ans = 0;
	for (ll i = 1; i <= n; i++)
	{
		ll cnt = 0;
		for (ll j = 1; j <= n; j++)
		{
			if (i == j)continue;
			if (g[i][j] < 0x3f3f3f3f)
			{
				cnt++;
			}
		}

		for (ll j = 1; j <= n; j++)
		{
			if (i == j)continue;
			if (g[j][i] < 0x3f3f3f3f)
			{
				cnt++;
			}
		}
		if (cnt == n - 1)ans++;
	}
	cout << ans << endl;
}
```

https://vjudge.net/problem/POJ-2240

```
#include<iostream>
#include<algorithm>
#include<queue>
```

```
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e2 + 5;
ll n, m;
vector<pdl>g[N];
map<string, int>currency;
int cas = 1;
double dis[N];
ll vis[N];
ll cnt[N];
int spfa()
{
	memset(dis, 0, sizeof dis);
	memset(vis, 0, sizeof vis);
	memset(cnt, 0, sizeof cnt);
	queue<ll>q;
	for (ll i = 1; i <= n; i++)
	{
		q.push(i);
		vis[i] = 1;
	}
	dis[1] = 1;
	vis[1] = 1;
	while (!q.empty())
	{
		ll t = q.front();
		q.pop();
		vis[t] = 0;
		for (vector<pdl>::iterator i = g[t].begin(); i != g[t].end(); i++)
		{
			ll ver = (*i).second;
			double d = (*i).first;
			if (dis[ver] < dis[t] * d)
			{
				dis[ver] = dis[t] * d;
				cnt[ver] = cnt[t] + 1;
				//cout << "dis[" << ver << "]==" << dis[ver] << endl;
				if (cnt[ver] >= n)return 1;
				if (!vis[ver])
				{
					vis[ver] = 1;
					q.push(ver);
				}
			}
		}
	}
	return 0;
}
int main()
{
	IOS;
	while (cin >> n)
	{
		if (n == 0)break;
		for (ll i = 1; i <= n; i++)
		{
			g[i].clear();
		}
		for (ll i = 1; i <= n; i++)
		{
			string s;
			cin >> s;
			currency[s] = i;
		}
		cin >> m;
		for (ll i = 1; i <= m; i++)
		{
			string s, t;
			double ex;
			cin >> s >> ex >> t;
			g[currency[s]].push_back({ ex,currency[t] });
		}
		cout << "Case " << cas++ << ": ";
		if (spfa())cout << "Yes" << endl;
		else cout << "No" << endl;
	}
}
```

https://vjudge.net/problem/POJ-1511
这题比较坑，即使用关闭缓冲同步的cin也会超时，必须用scanf或者快读

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 2e6 + 5;
ll n, p, q;
ll h[N], ne[N], e[N], idx;
ll w[N];
ll st[N];
ll dis[N];
struct node {
	ll from, to, val;
}edge[N];
void add(ll a, ll b, ll c)
{
	e[idx] = b;
	w[idx] = c;
	ne[idx] = h[a];
	h[a] = idx++;

}
void heap_dijkstra()
{
	memset(st, 0, sizeof st);
	memset(dis, 0x3f, sizeof dis);
	priority_queue<pll, vector<pll>, greater<pll> >que;
	que.push({ 0,1 });
	dis[1] = 0;
	while (!que.empty())
	{
		pll t = que.top();
		que.pop();
		ll ver = t.second;
		if (st[ver])continue;
		st[ver] = 1;
		for (ll i = h[ver]; i != -1; i=ne[i])
		{
			ll j = e[i];
			ll v = w[i];
			if (dis[j] > dis[ver] + v)
			{
				dis[j] = dis[ver] + v;
				que.push({ dis[j],j });
			}
		}
	}
}

int main()
{
	scanf("%lld", &n);
	while (n--)
	{
		scanf("%lld%lld", &p, &q);
		memset(h, -1, sizeof h);
		idx = 0;
		for (ll i = 1; i <= q; i++)
		{
			//cin >> edge[i].from >> edge[i].to >> edge[i].val;
			scanf("%lld%lld%lld", &edge[i].from, &edge[i].to, &edge[i].val);
			add(edge[i].from, edge[i].to, edge[i].val);
		}
		heap_dijkstra();
		ll ans = 0;
		for (ll i = 1; i <= p; i++)
		{
			ans += dis[i];
		}
		memset(h, -1, sizeof h);
		idx = 0;
		for (ll i = 1; i <= q; i++)
		{
			add(edge[i].to, edge[i].from, edge[i].val);
		}
		heap_dijkstra();
		for (ll i = 1; i <= p; i++)
		{
			ans += dis[i];
		}
		cout << ans << endl;
	}
}
```

用scanf还是比较慢，再补一个用快读的版本

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 2e6 + 5;
ll n, p, q;
ll h[N], ne[N], e[N], idx;
ll w[N];
ll st[N];
ll dis[N];
struct node {
	ll from, to, val;
}edge[N];
void add(ll a, ll b, ll c)
{
	e[idx] = b;
	w[idx] = c;
	ne[idx] = h[a];
	h[a] = idx++;

}
void heap_dijkstra()
{
	memset(st, 0, sizeof st);
	memset(dis, 0x3f, sizeof dis);
	priority_queue<pll, vector<pll>, greater<pll> >que;
	que.push({ 0,1 });
	dis[1] = 0;
	while (!que.empty())
	{
		pll t = que.top();
		que.pop();
		ll ver = t.second;
		if (st[ver])continue;
		st[ver] = 1;
		for (ll i = h[ver]; i != -1; i=ne[i])
		{
			ll j = e[i];
			ll v = w[i];
			if (dis[j] > dis[ver] + v)
			{
				dis[j] = dis[ver] + v;
				que.push({ dis[j],j });
			}
		}
	}
}
inline int read() {
	int x = 0, f = 1;
	char ch = getchar();
	while (ch < '0' || ch>'9') {
		if (ch == '-')
			f = -1;
		ch = getchar();
	}
	while (ch >= '0' && ch <= '9') {
		x = (x << 1) + (x << 3) + (ch ^ 48);
		ch = getchar();
	}
	return x * f;
}
int main()
{
	n = read();
	while (n--)
	{
		p = read();
		q = read();
		memset(h, -1, sizeof h);
		idx = 0;
		for (ll i = 1; i <= q; i++)
		{
			//cin >> edge[i].from >> edge[i].to >> edge[i].val;
			//scanf("%lld%lld%lld", &edge[i].from, &edge[i].to, &edge[i].val);
			edge[i].from = read();
			edge[i].to = read();
			edge[i].val = read();
			add(edge[i].from, edge[i].to, edge[i].val);
		}
		heap_dijkstra();
		ll ans = 0;
		for (ll i = 1; i <= p; i++)
		{
			ans += dis[i];
		}
		memset(h, -1, sizeof h);
		idx = 0;
		for (ll i = 1; i <= q; i++)
		{
			add(edge[i].to, edge[i].from, edge[i].val);
		}
		heap_dijkstra();
		for (ll i = 1; i <= p; i++)
		{
			ans += dis[i];
		}
		cout << ans << endl;
	}
}
```

https://vjudge.net/problem/POJ-2502

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 5e2 + 5;
ll n;
const double fspeed = 10;
const double sspeed = 40;
struct node {
	double x, y;
}p[N];
double g[N][N];
double dist(node& a, node& b)
{
	double r = fabs(a.x - b.x);
	double s = fabs(a.y - b.y);
	return sqrt(r * r+s * s)/1000;
}
void floyd()
{
	for (ll k = 1; k <= n; k++)
	{
		for (ll i = 1; i <= n; i++)
		{
			for (ll j = 1; j <= n; j++)
```

```
                        {
                                g[i][j] = min(g[i][j], g[i][k] + g[k][j]);
                        }
                }
        }
}
int main()
{
        IOS;
        cin >> p[1].x >> p[1].y;
        cin >> p[2].x >> p[2].y;
        for (ll i = 1; i <= 500; i++)
        {
                for (ll j = 1; j <= 500; j++)
                {
                        if (i != j)
                                g[i][j] = 1e9;
                        else
                                g[i][j] = 0;
                }
        }
        n = 2;
        ll i = 0;
        ll a, b;
        while (cin >> a >> b)
        {
                if (a == -1 && b == -1)
                {
                        i = 0;
                        continue;
                }
                p[++n].x = a, p[n].y = b;
                i++;
                if (i == 1)
                        continue;
                double cost = dist(p[n], p[n - 1]) / sspeed;
                g[n][n - 1] = g[n - 1][n] = cost;
                /*
                注意这里，只有相邻的站点可以用地铁速度计算时间，因为地铁的路线可能不是直
                线；举个例子，如果路线是折线，此时用
                起点站和终点站的欧几里得距离计算时间显然是错的
                */

        }
        for (ll s = 1; s <= n; s++)
        {
                for (ll t = 1; t <= n; t++)
                {
                                double cost = dist(p[t], p[s]) / fspeed;
                                g[s][t] = min(g[s][t],cost);
                                g[t][s] = min(g[s][t],cost);
                }
        }

        floyd();
```

```
	cout << (long long)(g[1][2]*60+0.5)<<endl;
}
```

关于差分约束系统的建立和求解，可以参考这篇文章：
https://zhuanlan.zhihu.com/p/104764488

例题： https://vjudge.net/problem/POJ-3159

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 2e6 + 5;
ll n, m;
ll h[N], ne[N], e[N], w[N], idx;
ll st[N];
ll dis[N];
void add(ll a, ll b, ll c)
{
	e[idx] = b;
	w[idx] = c;
	ne[idx] = h[a];
	h[a] = idx++;
}
void heap_dijkstra()
{
	memset(st, 0, sizeof st);
	memset(dis, 0x3f, sizeof dis);
	priority_queue<pll, vector<pll>, greater<pll> >que;
	que.push({ 0,1 });
	dis[1] = 0;
	while (!que.empty())
	{
		pll t = que.top();
		que.pop();
		ll ver = t.second;
		if (st[ver])continue;
		st[ver] = 1;
		for (ll i = h[ver]; i != -1; i=ne[i])
```

```
				{
					ll j = e[i];
					ll v = w[i];
					if (dis[j] > dis[ver] + v)
					{
						dis[j] = dis[ver] + v;
						que.push({ dis[j],j });
					}
				}
		}
}
inline ll read()
{
		ll x=0, f=1;
		char ch = getchar();
		while (ch < '0' || ch>'9')
		{
				if (ch == '-')f = -1;
				ch = getchar();
		}
		while (ch >= '0' && ch <= '9')
		{
				x = (x << 1) + (x << 3) + (ch ^ 48);
				ch = getchar();
		}
		return x * f;
}
int main()
{
		IOS;
		n = read(), m = read();
				memset(h, -1, sizeof h);
				for (ll i = 1; i <= m; i++)
				{
					ll a, b, w;
					a = read();
					b = read();
					w = read();
					add(a, b, w);
				}
				heap_dijkstra();
				ll maxn = 0;
				ll minn = 1e9;
				for (ll i = 1; i <= n; i++)
				{
					maxn = max(maxn, dis[i]);
					minn = min(minn, dis[i]);
				}
				cout << maxn-minn << endl;
}
```

https://vjudge.net/problem/POJ-1062

```
#include<iostream>
#include<algorithm>
#include<queue>
```

```
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e2 + 5;
const ll mod=1e9+7;
ll m,n;
ll dis[N];
ll st[N];
ll max_level[N];
ll min_level[N];
struct node{
    ll pri,lev;
    vector<pll>adj;
}mem[N];
ll myabs(ll num)
{
    if(num<0)return -num;
    else return num;
}
void dijkstra()
{
    memset(dis,0x3f,sizeof dis);
    memset(max_level,0,sizeof max_level);
    memset(min_level,0x3f,sizeof min_level);
    priority_queue<pll,vector<pll>,greater<pll> >heap;
    dis[1]=mem[1].pri;
    max_level[1]=mem[1].lev;
    min_level[1]=mem[1].lev;
    heap.push({dis[1],1});
    while(heap.size())
    {
        pll t=heap.top();
        heap.pop();
        ll ver=t.second;
        if(st[ver])continue;
        st[ver]=1;
        for(vector<pll>::iterator i=mem[ver].adj.begin();i!
=mem[ver].adj.end();i++)
        {
            ll w=(*i).first;
            ll v=(*i).second;
            if(dis[v]>dis[ver]-mem[ver].pri+w+mem[v].pri&&myabs(mem[v].lev-
min_level[ver])<=m&&myabs(mem[v].lev-max_level[ver])<=m)
            {
                dis[v]=dis[ver]-mem[ver].pri+w+mem[v].pri;
                heap.push({dis[v],v});
                min_level[v]=min(min_level[ver],mem[v].lev);
                max_level[v]=max(max_level[ver],mem[v].lev);
            }
            //cout<<"dis["<<ver<<"]=="<<dis[ver]<<endl;
        }
    }
}
int main()
{
    IOS;
    cin>>m>>n;
    for(ll i=1;i<=n;i++)
    {
        cin>>mem[i].pri>>mem[i].lev;
        ll x;
        cin>>x;
        for(ll j=0;j<x;j++)
        {
            ll a,b;
            cin>>a>>b;
            mem[i].adj.push_back({b,a});
        }
    }
    dijkstra();
    ll ans=1e9;
    for(ll i=1;i<=n;i++)
    {
        ans=min(ans,dis[i]);
    }
    cout<<ans<<endl;
}
```

https://vjudge.net/problem/POJ-1847

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e2 + 5;
const ll mod=1e9+7;
ll n,a,b;
ll dis[N];
vector<ll>adj[N];
ll st[N];
void dijkstra()
{
    memset(dis,0x3f,sizeof dis);
    priority_queue<pll,vector<pll>,greater<pll> >heap;
    dis[a]=0;
    heap.push({dis[a],a});
    while(heap.size())
    {
        pll t=heap.top();
        heap.pop();
        ll ver=t.second;
        if(st[ver])continue;
        st[ver]=1;
        for(vector<ll>::iterator i=adj[ver].begin();i!=adj[ver].end();i++)
        {
            ll j=(*i);
            ll k=0;
            if(i==adj[ver].begin())
            {
                k=0;
            }
            else
            {
                k=1;
            }
            if(dis[j]>dis[ver]+k)
            {
                dis[j]=dis[ver]+k;
                heap.push({dis[j],j});
                //cout<<"dis["<<j<<"]=="<<dis[j]<<endl;
            }
        }
    }
}
int main()
{
    IOS;
    cin>>n>>a>>b;
    for(ll i=1;i<=n;i++)
    {
        ll k;
        cin>>k;
        for(ll j=1;j<=k;j++)
        {
            ll to;
            cin>>to;
            adj[i].push_back(to);
        }
    }
    dijkstra();
    if(dis[b]>=INF)cout<<-1<<endl;
    else
    cout<<dis[b]<<endl;
}
```

https://vjudge.net/problem/LightOJ-1074

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
```

```
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e5 + 5;
const ll mod=1e9+7;
ll n;
ll busy[N];
ll h[N],ne[N],e[N],idx;
ll dis[N];
ll st[N];
ll cnt[N];
ll color[N];
ll cas=1;
void add(ll a,ll b)
{
    e[idx]=b;
    ne[idx]=h[a];
    h[a]=idx++;
}
void dfs(ll cur)
{
    for(ll i=h[cur];i!=-1;i=ne[i])
    {
        ll j=e[i];
        if(!color[j]){
        color[j]=1;
        dfs(j);
        }
    }
}
void spfa()
{
    memset(dis,0x3f,sizeof dis);
    memset(st,0,sizeof st);
    memset(cnt,0,sizeof cnt);
    memset(color,0,sizeof color);
    queue<pll>q;
    dis[1]=0;
    st[1]=1;
    q.push({0,1});
    while(!q.empty())
    {
        pll t=q.front();
        q.pop();
        ll ver=t.second;
        st[ver]=0;
```

```
            for(ll i=h[ver];i!=-1;i=ne[i])
            {
                ll j=e[i];
                ll k=busy[j]-busy[ver];
                if(dis[j]>dis[ver]+k*k*k&&!color[j])
                {
                    dis[j]=dis[ver]+k*k*k;
                    cnt[j]=cnt[ver]+1;
                    if(cnt[j]>=n)
                    {
                        dfs(j);
                        continue;
                    }
                    if(!st[j])
                    {
                        q.push({dis[j],j});
                        st[j]=1;
                    }
                }
            }
        }
    }
    int main()
    {
        IOS;
        ll t;
        cin>>t;
        while(t--)
        {
            memset(h,-1,sizeof h);
            idx=0;
            cin>>n;
            for(ll i=1;i<=n;i++)
            {
                cin>>busy[i];
            }
            ll m;
            cin>>m;
            while(m--)
            {
                ll a,b;
                cin>>a>>b;
                add(a,b);
            }
            spfa();
            ll q;
            cin>>q;
            cout<<"Case "<<cas++<<":"<<endl;
            while(q--)
            {
                ll k;
                cin>>k;
                if(dis[k]!=INF&&dis[k]>=3&&color[k]==0)
                    cout<<dis[k]<<endl;
                else cout<<"?"<<endl;
            }
        }
}
```

https://vjudge.net/problem/HDU-4725

注：超级源汇点的思想十分巧妙，需要认真理解

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f
#define ll long long
#define pii pair<int,int>
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 8e5 + 5;
const int mod = 1e9 + 7;
int n, m, c;
int h[N], e[N], ne[N], w[N], idx;
int dis[N];
int st[N];
int cas = 1;
/*
朴素方法建图会超时。对于每一层，设定一个超级源点，然后各层的节点直接连向相邻层的超级源点
*/
inline void add(int a, int b, int c)
{
        e[idx] = b;
        w[idx] = c;
        ne[idx] = h[a];
        h[a] = idx++;
}
void heap_dijkstra()
{
        memset(dis, 0x3f, sizeof dis);
        memset(st, 0, sizeof st);
        priority_queue<pii, vector<pii>, greater<pii> >heap;
        dis[1] = 0;
        heap.push({ 0,1 });
        while (!heap.empty())
        {
                pii t = heap.top();
                heap.pop();
                int ver = t.second;
                if (st[ver])continue;
                st[ver] = 1;
                for (int i = h[ver]; i != -1; i = ne[i])
```

```
			{
				int j = e[i];
				int v = w[i];
				if (dis[j] > dis[ver] + v)
				{
					dis[j] = dis[ver] + v;
					heap.push({ dis[j],j });
				}
			}



		}
}
int main()
{
		int t;
		scanf("%d", &t);
		while (t--)
		{
			scanf("%d%d%d", &n, &m, &c);
			memset(h, -1, sizeof h);
			idx = 0;
			for (int i = 1; i <= n; i++)
			{
				int lay;
				scanf("%d", &lay);
				add(n + lay, i, 0);//每一层的超级源点到该层其他节点距离为0
				if (lay > 1)
				{
					add(i, n + lay - 1, c);
				}
				if (lay < n)
				{
					add(i, n + lay+ 1, c);
				}
			}

			while (m--)
			{
				int u, v, w;
				scanf("%d%d%d", &u, &v, &w);
				add(u, v, w);
				add(v, u, w);
			}
			heap_dijkstra();
			printf("Case #%d: ", cas++);
			if (dis[n] < INF)
				printf("%d\n", dis[n]);
			else printf("-1\n");
		}

}
```

https://vjudge.net/problem/HDU-4370

```
#include<iostream>
#include<algorithm>
```

```
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e3 + 5;
const ll mod=1e9+7;
ll n;
ll g[N][N];
ll dis[N];
ll st[N];
ll cir=INF;
/*
有两种情况满足题目所给的条件：一是存在一条从1到n的最短路径
二是存在1经过至少1个点回到1的闭环，和n经过至少一个点回到n的自环
两种情况取最小即可
*/
void dijkstra(ll s)
{
    memset(dis,0x3f,sizeof dis);
    memset(st,0,sizeof st);
    cir=INF;
    dis[s]=0;
    for(ll i=0;i<n;i++)
    {
        ll t=-1;
        for(ll j=1;j<=n;j++)
        {
            if(!st[j]&&(t==-1||dis[t]>dis[j]))
            {
                t=j;
            }
        }
        st[t]=1;
        for(ll j=1;j<=n;j++)
        {
            if(dis[j]>dis[t]+g[t][j])
            {
                dis[j]=dis[t]+g[t][j];

            }
            if(j==s&&t!=s)
            {
                cir=min(cir,dis[t]+g[t][j]);
            }
        }
    }
}
int main()
{
    IOS;
    while(cin>>n)
    {
        for(ll i=1;i<=n;i++)
        {
            for(ll j=1;j<=n;j++)
            {
                cin>>g[i][j];
            }
        }
        dijkstra(1);
        ll ans=dis[n];
        ll c=cir;
        dijkstra(n);
        c+=cir;
        ans=min(ans,c);
        cout<<ans<<endl;
    }
}
```

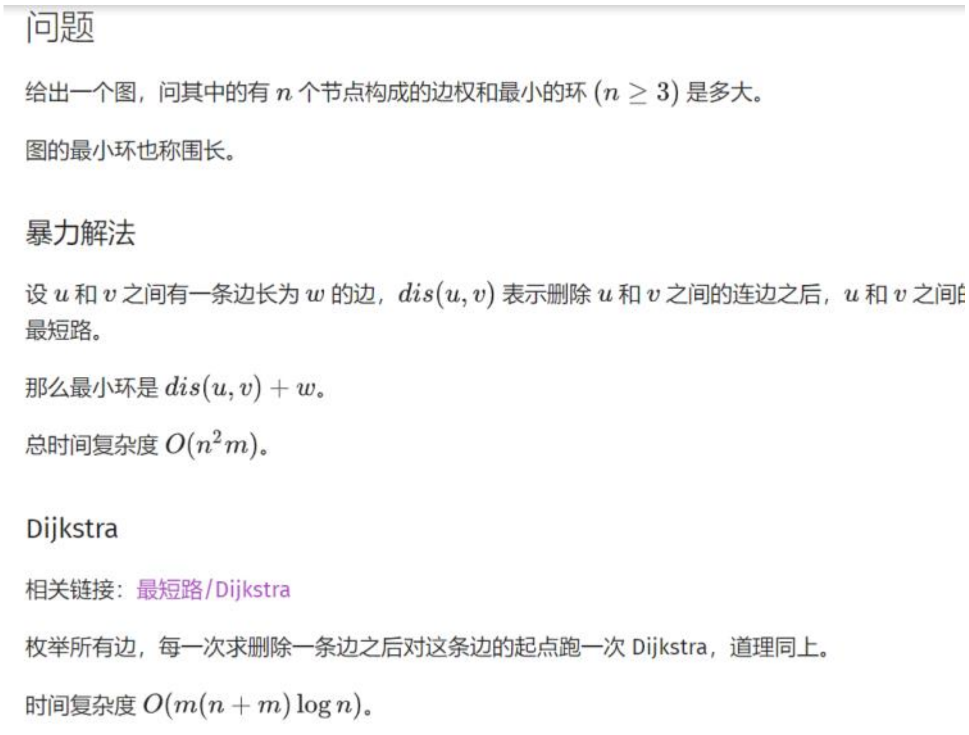
https://vjudge.net/problem/POJ-3169

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e5 + 5;
const ll mod=1e9+7;
/*
差分约束系统，注意不等关系的变形
*/
ll n,ml,md;
ll dis[N],vis[N];
ll cnt[N];
ll h[N],e[N],w[N],ne[N],idx;
void add(ll a,ll b,ll c)
{
    e[idx]=b;
    w[idx]=c;
    ne[idx]=h[a];
    h[a]=idx++;
}
int spfa()
{
    memset(dis,0x3f,sizeof dis);
    dis[1]=0;
    queue<ll>q;
    q.push(1);
    vis[1]=1;
    while(!q.empty())
    {
        ll ver=q.front();
        q.pop();
        vis[ver]=0;
        for(ll i=h[ver];i!=-1;i=ne[i])
        {
            ll j=e[i];
            ll v=w[i];
            if(dis[j]>dis[ver]+v)
            {
                dis[j]=dis[ver]+v;
                cnt[j]=cnt[ver]+1;
                if(cnt[j]>=n)return -1;
                if(!vis[j])
                {
                    vis[j]=1;
                    q.push(j);
                }
            }
        }
    }
    return 1;
}
int main()
{
    IOS;
    cin>>n>>ml>>md;
    memset(h,-1,sizeof h);
    while(ml--)
    {
        ll a,b,c;
        cin>>a>>b>>c;
        add(a,b,c);//b-a<=k等价于b<=a+k，所以从a到b建一条边
    }
    while(md--)
    {
        ll a,b,c;
        cin>>a>>b>>c;
        add(b,a,-c);//b-a>=k可以转化为a-b<=-k
    }
    if(spfa()==-1)//如果有负环，无解
    {
        cout<<-1;
    }
```

```
        else{
            if(dis[n]==INF)//如果dis[n]==INF，说明点1和点n没出现在不等式组中，也就是说不连通
            {
                cout<<-2<<endl;
            }
            else{
                cout<<dis[n]-dis[1]<<endl;
            }
        }
    }
```

dijkstra求最小环

2021年11月23日　　18:16



问题

给出一个图，问其中的有 n 个节点构成的边权和最小的环 $(n \geq 3)$ 是多大。

图的最小环也称围长。

暴力解法

设 u 和 v 之间有一条边长为 w 的边，$dis(u, v)$ 表示删除 u 和 v 之间的连边之后，u 和 v 之间
最短路。

那么最小环是 $dis(u, v) + w$。

总时间复杂度 $O(n^2 m)$。

Dijkstra

相关链接：最短路/Dijkstra

枚举所有边，每一次求删除一条边之后对这条边的起点跑一次 Dijkstra，道理同上。

时间复杂度 $O(m(n + m) \log n)$。

例： https://codeforces.com/gym/103409/problem/E

```
#include<bits/stdc++.h>
#define ll long long
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f3f3f3f3f
using namespace std;
const ll N=2e3+5;
const ll M=5e3+20;
const ll mod=1e9+7;
ll n,m,c;
ll st[N];
ll dis[N];
vector<pll>g[N];
ll dijkstra(ll u)
{
  memset(dis,0x3f,sizeof dis);
  memset(st,0,sizeof st);
  dis[u]=0;
  priority_queue<pll,vector<pll>,greater<pll> >heap;
  heap.push({0,u});
  while(!heap.empty())
  {
    ll t=heap.top().second;
    heap.pop();
    if(st[t])continue;
    st[t]=1;
    for(auto i:g[t])
    {
            ll val=i.first;
            ll ver=i.second;
            if(ver==u&&dis[t]+val<=c)return 1;
            if(!st[ver]&&dis[t]+val<dis[ver])
            {
                dis[ver]=dis[t]+val;
                heap.push({dis[ver],ver});
            }
        }
    }
    return 0;
}
int main()
{
    scanf("%lld%lld%lld",&n,&m,&c);
    int f=0;
    for(int i=1;i<=m;i++)
    {
        ll u,v,p;
        scanf("%lld%lld%lld",&u,&v,&p);
        g[u].push_back({p,v});
        if(p<=c)
        {
            f=1;
        }
    }
    if(!f)
    {
        printf("0\n");
        return 0;
    }
    for(int i=1;i<=n;i++)
    {
        if(dijkstra(i))
        {
            printf("2\n");
            return 0;
        }
    }
    printf("1\n");
}
```

最小生成树

2021年8月4日 0:02

prim算法：朴素版时间复杂度为O(v^2)，堆优化版时间复杂度为O(elogv)

kruskal算法：时间复杂度为O(eloge)

朴素prim: https://www.acwing.com/problem/content/860/

```
#include<iostream>
#include<algorithm>
#include<cstring>
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
using namespace std;
const ll N=1e3+5;
ll g[N][N];
ll dis[N];
ll vis[N];
ll n,m;
ll prim()
{
   memset(dis,0x3f,sizeof dis);
   ll res=0;
   for(ll i=0;i<n;i++)
   {
      ll t=-1;
      for(ll j=1;j<=n;j++)
      {
         if(t==-1||(dis[j]<dis[t]&&!vis[j]))
         {
            t=j;//找到距离连通块距离最近的点
         }
      }
      if(i!=0&&dis[t]==INF)//不连通，无解
      {
         return INF;
      }
      if(i!=0)
         res+=dis[t];
      vis[t]=1;
      for(ll i=1;i<=n;i++)
      {
         if(!vis[i])
         dis[i]=min(dis[i],g[t][i]);//更新其他点到连通块的距离
      }


   }
   return res;
}
int main()
```

```
{
   memset(g,0x3f,sizeof g);
   cin>>n>>m;
   for(ll i=1;i<=n;i++)
   {
      g[i][i]=0;
   }
   while(m--)
   {
      ll u,v,w;
      cin>>u>>v>>w;
      g[u][v]=g[v][u]=min(g[u][v],w);
   }
   ll ans=prim();
   if(ans==INF)cout<<"impossible"<<endl;
   else cout<<ans<<endl;
}
```

kusakal: https://www.acwing.com/problem/content/861/

```
#include<iostream>
#include<algorithm>
#define ll long long
using namespace std;
const ll N=2e5+5;
struct node{
   ll a,b,w;
}edge[N];
ll n,m;
ll pa[N];
ll find(ll x)
{
   if(pa[x]==x)return x;
   else return pa[x]=find(pa[x]);
}
int main()
{
   cin>>n>>m;
   for(ll i=0;i<m;i++)
   {
      ll u,v,w;
      cin>>u>>v>>w;
      edge[i]={u,v,w};
   }
   sort(edge,edge+m,[&](node s,node t)->bool{return s.w<t.w;});
   for(ll i=1;i<=n;i++)
   {
      pa[i]=i;
   }
   ll res=0;
   ll cnt=0;
   for(ll i=0;i<m;i++)
   {
      ll s=find(edge[i].a);
      ll t=find(edge[i].b);
      if(s!=t)
```

```
        {
            res+=edge[i].w;
            cnt++;
            pa[s]=t;
        }
    }
    if(cnt<n-1)cout<<"impossible"<<endl;
    else cout<<res<<endl;
}
```

次小生成树

2021年8月15日 22:20

次小生成树有两种：
1.权值第二小的生成树，这种定义下，次小生成树的权值可能和最小生成树相同
2.权值严格大于最小生成树的生成树

求解方法：
1.求一遍最小生成树，然后枚举删除其中的每条边，每次重新求一遍最小生成树
O(mlogm+nm)
这种方法无法求严格次小生成树
2.先求最小生成树，枚举所有非树边，加入树中，同时去掉一条边，使其仍为树，不断枚举，一定可以求得次小生成树。

https://www.acwing.com/problem/content/1150/

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <vector>
#include <cstring>
#include <cmath>
#include <iomanip>
#include <map>
#include <stdio.h>
#include <stack>
#include <set>
#include <deque>
#define INF 0x3f3f3f3f
#define ll long long
#define rep(i, a, b) for (ll i = a; i <= b; i++)
#define rev(i, a, b) for (ll i = a; i >= b; i--)
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define DINF 1e20
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
#define eps 1.0E-12
using namespace std;
const ll N = 1e3+5;
const ll M=1e4+5;
const ll mod = 1000000007;
ll n,m;
vector<pll>g[N];
ll pre[N];
ll dis1[N][N];
ll dis2[N][N];
struct node{
    ll u,v,w;
    int f=0;
}edge[M];
ll find(ll x)
{
    if(pre[x]==x)return x;
    else return pre[x]=find(pre[x]);
}
void merge(ll a,ll b)
{
    ll pa=find(a),pb=find(b);
    if(pa!=pb)
    {
        pre[pa]=pb;
    }
}
void dfs(ll root,ll fa,ll cur,ll mm1,ll mm2)
{
    dis1[root][cur]=mm1;
    dis2[root][cur]=mm2;
    for(auto i:g[cur])
    {
        if(i.second!=fa)
        {

            if(i.first>mm1)
            {
                mm2=mm1;
                mm1=i.first;
            }
            else if(i.first<mm1&&i.first>mm2)
            {
                mm2=i.first;
            }
            dfs(root,cur,i.second,mm1,mm2);
        }
    }
}
void solve()
{
    cin >> n >> m;
    rep(i, 1, n)
    {
        pre[i] = i;
    }
    rep(i, 1, m)
    {
        cin>>edge[i].u>>edge[i].v>>edge[i].w;
    }
    sort(edge+1,edge+1+m,[&](node a,node b)->bool{return a.w<b.w;});
    ll res=0;
    rep(i,1,m)
    {
        ll u=edge[i].u,v=edge[i].v,w=edge[i].w;
        if(find(u)!=find(v))
        {
            merge(u,v);
            res+=w;
            g[u].push_back({w,v});
            g[v].push_back({w,u});
            edge[i].f=1;
        }
    }
    rep(i,1,n)//在最小生成树上，以点i为根寻找其距离其他每个点的路径最大边与次大边
    {
        dfs(i,i,i,0,0);
    }
    ll ans=1e18;
    rep(i,1,m)
    {
        if(!edge[i].f)//如果是非树边
        {
            ll u=edge[i].u,v=edge[i].v,w=edge[i].w;
            if(w>dis1[u][v])//求严格次小生成树必须有此限制
            {
                ans=min(ans,res+w-dis1[u][v]);//加上当前边，去掉原来两点间最大边
            }
            else if(w>dis2[u][v])
            {
                ans=min(ans,res+w-dis2[u][v]);
            }
        }
    }
    cout<<ans<<endl;

}
int main()
{
    IOS;
    solve();
}
```

倍增

2021年8月17日 4:23

https://www.acwing.com/problem/content/111/

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <vector>
#include <cstring>
#include <cmath>
#include <iomanip>
#include <map>
#include <stdio.h>
#include <stack>
#include <set>
#include <deque>
#define INF 0x3f3f3f3f
#define ll long long
#define rep(i, a, b) for (ll i = a; i <= b; i++)
#define rev(i, a, b) for (ll i = a; i >= b; i--)
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define DINF 1e20
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
#define eps 1.0E-12
using namespace std;
const ll N = 5e5+5;
const ll M=1e4+5;
const ll mod = 1000000007;
ll n,m,t;
ll a[N];
ll b[N];
ll tmp[N];
ll cal(ll l,ll mid,ll r)
{
    rep(i,mid,r-1)
    {
        b[i]=a[i];
    }
    sort(b+mid,b+r);//对新增的区间排序
    ll i=l,j=mid;
    ll k=0;
    while(i<mid&&j<r)//归并新区间和旧区间
    {
        if(b[i]<b[j])
        {
            tmp[k++]=b[i++];
        }
        else{
            tmp[k++]=b[j++];
        }
    }
    while(i<mid)tmp[k++]=b[i++];
    while(j<r)tmp[k++]=b[j++];
    ll res=0;
    ll cnt=0;
    i=0,j=k-1;
    while(i<j&&cnt<m)//计算当前区间答案
    {
        res+=(tmp[j]-tmp[i])*(tmp[j]-tmp[i]);
        i++;
        j--;
        cnt++;
    }
    return res<=t;
}
void solve()
{
    cin>>n>>m>>t;
    rep(i,1,n)
    {
        cin>>a[i];
    }
    ll ans=0;
    ll st=1,en=1;
    while(en<=n)
    {
        ll p=1;
        while(p)
        {
            if(en+p<=n+1&&cal(st,en,en+p))//[st,en+p)
            {
                en+=p;
                p*=2;
                rep(i,st,en-1)
                {
                    b[i]=tmp[i-st];
                }
            }
            else{
                p/=2;
            }
        }
        st=en;
        ans++;
    }
    cout<<ans<<endl;
}
int main()
{
    IOS;
    ll _;
    cin>>_;
    while(_--)
    {
```

```
        solve();
    }
}
```

克鲁斯卡尔重构树

2022年1月17日　　18:52

https://codeforces.com/gym/103446/problem/H

```
#include<bits/stdc++.h>
#pragma optimize(2)
#pragma optimize(3)
#define ll long long
#define pll pair<ll,ll>
#define INF 0x3f3f3f3f
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll mod=998244353;
const ll N=3e5+5;
const ll M=55;
ll n,m,q;
ll x,k;
ll a[N];
ll fa[N][30];
struct edge{
   ll u,v,w;
}e[N];
struct node{
   int leaf=0;
   ll val=0;
   ll sum=0;
   vector<ll>adj;
};
ll pre[N];
ll val[N];
vector<node>g;
ll tot;
ll find(ll x)
{
   if(pre[x]==x)return x;
   else return pre[x]=find(pre[x]);
}
void dfs(ll cur)
{
   for(ll i=1;i<=20;i++)
   {
      fa[cur][i]=fa[fa[cur][i-1]][i-1];
   }
   for(auto i:g[cur].adj)
   {
      fa[i][0]=cur;
      dfs(i);
   }
}
int main()
{
   scanf("%lld%lld%lld",&n,&m,&q);
   tot=n;
```

```
    g.resize(2*n+5);
    for(ll i=1;i<=n;i++)
    {
        scanf("%lld",&a[i]);
        pre[i]=i;
        g[i].leaf=1;
        g[i].val=a[i];
        g[i].sum=a[i];
    }
    for(ll i=1;i<=m;i++)
    {
        scanf("%lld%lld%lld",&e[i].u,&e[i].v,&e[i].w);
    }
    sort(e+1,e+1+m,[&](edge a,edge b){
            return a.w<b.w;
         });
    for(ll i=1;i<=m;i++)
    {
        ll t1=find(e[i].u);
        ll t2=find(e[i].v);
        if(t1!=t2)
        {
            g[++tot].val=e[i].w;
            pre[tot]=tot;
            pre[t1]=tot;
            pre[t2]=tot;
            g[tot].adj.push_back(t1);
            g[tot].adj.push_back(t2);
            g[tot].sum=g[t1].sum+g[t2].sum;
        }
    }
    dfs(tot);
    while(q--)
    {

        scanf("%lld%lld",&x,&k);
        ll step=0;
        while(step>=0)
        {
            ll anc=fa[x][step];
            if(anc!=0&&g[x].sum+k>=g[anc].val)
            {
                x=anc;
                step++;
            }
            else
            {
                step--;
            }
        }
        printf("%lld\n",g[x].sum+k);
    }


}
```

https://codeforces.com/problemset/problem/1131/F

```
#include<bits/stdc++.h>
#define ll long long
#define ull unsigned long long
#define debug(x) cout << #x << " = " << (x) << endl
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define INF 0x3f3f3f3f3f3f3f3f
#define pll pair<ll,ll>
#define ld long double
using namespace std;
const ll N = 3e5 + 5;
const ll M = 1005;
ll n;
vector<ll>g[N];
ll tot = 150000;
ll p[N];
ll find(ll x)
{
	if(x == p[x])return x;
	else return p[x] = find(p[x]);
}
vector<ll>ans;
void dfs(ll cur)
{
	if(g[cur].empty())
	{
		ans.push_back(cur);
		return;
	}
	for(auto i : g[cur])
	{
		dfs(i);
	}
}
int main()
{
	IOS;
	cin >> n;
	for(ll i = 1;i <= N - 1; i ++ )
	{
		p[i] = i;
	}
	for(ll i = 0; i < n - 1; i ++ )
	{
		ll x, y ;
		cin >> x >> y;
		ll t1 = find(x);
		ll t2 = find(y);
		p[t1] = p[t2] = ++ tot ;
		g[tot].push_back(t1);
		g[tot].push_back(t2);

	}
	dfs(tot);
	for(auto i : ans)
	{
			cout << i << ' ';
		}
		cout << endl;



	}
```

st表

2021年8月17日 19:45

https://www.luogu.com.cn/problem/P3865

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <vector>
#include <cstring>
#include <cmath>
#include <iomanip>
#include <map>
#include <stdio.h>
#include<cmath>
#include <stack>
#include <set>
#include <deque>
#define INF 0x3f3f3f3f
#define ll int
#define rep(i, a, b) for (ll i = a; i <= b; i++)
#define rev(i, a, b) for (ll i = a; i >= b; i--)
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define DINF 1e20
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
#define eps 1.0E-12
using namespace std;
const ll N = 1e5+5;
const ll M=1e4+5;
const ll mod = 1000000007;
ll n,m;
ll a[N];
ll f[N][30];//f[i][j]表示区间[i,i+2^j-1]内的最大值
void init()
{
    rep(i,1,n)
    {
        f[i][0]=a[i];
    }
    int t=log2(n);
    rep(j,1,t)
    {
        rep(i,1,n-(1<<j)+1)
        {
            f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
        }
    }
}
ll query(ll l,ll r)
{
    ll k=log2(r-l+1);
    return max(f[l][k],f[r-(1<<k)+1][k]);
}
inline int read()
{
    int x=0,f=1;char ch=getchar();
    while (ch<'0'||ch>'9'){if (ch=='-') f=-1;ch=getchar();}
    while (ch>='0'&&ch<='9'){x=x*10+ch-48;ch=getchar();}
    return x*f;
}
void solve()
{
    n=read();
    m=read();
    rep(i,1,n)
    {
        a[i]=read();
    }
    init();
    while(m--)
    {
        ll l,r;
        l=read();
        r=read();
        //cout<<query(l,r)<<endl;
        printf("%d\n",query(l,r));
    }
}
int main()
{
    solve();
}
```

染色法判断二分图

2021年8月4日　　1:33

一个图是二分图，当且仅当图中不含奇数环
用两种颜色对一张图做染色，每一条边的两个顶点染成不同颜色，如果染色过程中没有出现矛盾（同一条边的两个点染成同一个颜色），那么这张图就是二分图

https://www.acwing.com/problem/content/862/

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define PI 3.1415926
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define rep(i,a,b) for(ll i=a;i<=b;i++)
#define rev(i,a,b) for(ll i=a;i>=b;i--)
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=1e6+5;
const ll mod=1e9+7;
ll n,m;
ll col[N];
vector<ll>g[N];
int dfs(ll ver,ll c)
{
   col[ver]=c;
   for(auto i:g[ver])
   {
      if(!col[i])
      {
         if(!dfs(i,c==1?2:1))
         {
            return 0;
         }
      }
      else if(col[i]==col[ver]){
         return 0;
```

```
        }
    }
    return 1;
}
int main()
{
    IOS;
    cin>>n>>m;
    while(m--)
    {
        ll u,v;
        cin>>u>>v;
        g[u].push_back(v);
        g[v].push_back(u);
    }
    ll f=0;
    rep(i,1,n)
    {
        if(!col[i])
        {
            if(!dfs(i,1))
            {
                f=1;
                break;
            }
        }
    }
    if(!f)
    {
        cout<<"Yes"<<endl;
    }
    else{
        cout<<"No"<<endl;
    }
}
```

快读

2021年5月30日　23:27

```
inline int read() {
	int x = 0, f = 1;
	char ch = getchar();
	while (ch < '0' || ch>'9') {
		if (ch == '-')
			f = -1;
		ch = getchar();
	}
	while (ch >= '0' && ch <= '9') {
		x = (x << 1) + (x << 3) + (ch ^ 48);
		ch = getchar();
	}
	return x * f;
}
```

位运算的状态压缩（待补充）

2021年6月3日　　19:18

https://www.bttcacm.cn/problem.php?id=1011

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 2e6 + 5;
ll a[N];
inline ll read()
{
    ll x=0, f=1;
    char ch = getchar();
    while (ch < '0' || ch>'9')
    {
        if (ch == '-')f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9')
    {
        x = (x << 1) + (x << 3) + (ch ^ 48);
        ch = getchar();
    }
    return x * f;
}
int main()
{
    IOS;
    ll n;
    while(cin>>n)
    {
        memset(a,0,sizeof a);
        for(ll i=0;i<n;i++)
        {
            ll num;
            cin>>num;
            ll state=0;
            while(num)
```

```
                {
                    state|=1<<(num%10);//对每个数字，用二进制记录其中0-9是否出现
                    num/=10;
                }
                a[state]++;//记录每种状态的数量
            }
            ll ans=0;
            for(ll i=0;i<1024;i++)//枚举1-9的出现状况
            {
                for(ll j=i;j<1024;j++)
                {
                    if(i==j)//完全相同
                    {
                        ans+=(a[i]-1)*a[i]/2;
                    }
                    else if(i&j){//部分相同
                        ans+=a[i]*a[j];
                    }
                }
            }
            cout<<ans<<endl;
        }
    }
```

线性DP

2021年6月8日　　15:38

https://www.acwing.com/problem/content/description/273/

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 31;
const ll mod=1e9+7;
ll k;
ll num[N];
ll dp[N][N][N][N][N];
/*
dp[a][b][c][d][e]表示第k到第1排分别已经按排了a,b,c,d,e个人的方案数
从高到低按排每一个人，考虑最后一个人的按排情况，注意之前已经按排的人都比他高
那么，他或者可以直接排到第一排，因为第一排要求身高最低
或者如果有第i排的人数大于第i+1排的人数，那么他可以排到第i排，不会影响整体的单调性
*/
int main()
{
        while(cin>>k)
        {
                if(k==0)break;
                memset(num,0,sizeof num);
                memset(dp,0,sizeof dp);
                for(ll i=1;i<=k;i++)
                {
                        cin>>num[i];
                }
                dp[0][0][0][0][0]=1;
                for(ll i=0;i<=num[1];i++)
                {
                        for(ll j=0;j<=num[2];j++)
                        {
                                for(ll p=0;p<=num[3];p++)
                                {
                                        for(ll q=0;q<=num[4];q++)
                                        {
                                                for(ll r=0;r<=num[5];r++)
                                        {
                                                if(i>0&&i-1>=j)
                                                {
                                                        dp[i][j][p][q][r]+=dp[i-1][j][p][q][r];
                                                }
                                                if(j>0&&j-1>=p)
                                                {
                                                        dp[i][j][p][q][r]+=dp[i][j-1][p][q][r];
                                                }
                                                if(p>0&&p-1>=q)
                                                {
                                                        dp[i][j][p][q][r]+=dp[i][j][p-1][q][r];
                                                }
                                                if(q>0&&q-1>=r)
                                                {
                                                        dp[i][j][p][q][r]+=dp[i][j][p][q-1][r];
                                                }
                                                if(r>0)
                                                {
                                                        dp[i][j][p][q][r]+=dp[i][j][p][q][r-1];
                                                }
                                                //cout<<'*'<<dp[i][j][p][q][r]<<endl;
                                        }
                                }
                        }
                }
        }
        cout<<dp[num[1]][num[2]][num[3]][num[4]][num[5]]<<endl;
    }
}
```

https://vjudge.net/problem/HDU-1024

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll int
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 1e6+5;
const ll mod=1e9+7;
ll m,n;
/*
ll w[N][N];//w[i][j]表示在前i个数中选j个区间且第i个数必须选时得到的最大价值
ll b[N][N];//b[i][j]表示在前i个数中选j个区间且第i个数可以不选时得到的最大价值
对于w，转移方程为：w[i][j]=max(b[i-1][j-1]+a[i],w[i-1][j]+a[i]);
```

```
即考虑第i个数，第i个数可以单独成为一个区间，也可以和前一个区间合并
对于b，转移方程为：b[i][j]=max(b[i-1][j],w[i][j]);
观察转移方程，将w和b压成一维：
w[j]=max(b[j-1]+a[i],w[j]+a[i]);
b[j]=max(b[j],w[j]);
注意更新j时用到了j-1，所以j必须反过来迭代，否则更新j时j-1已经变了
咳……但实践表明这样还是t了，我们来尝试压掉另一维
注意到更新w[i][j]时用到了b[i-1][j-1]，这意味着更新w[i][j]时b[i-1][j-1]是不能动的，即，压
成一维时b[i-1]不能动
那么有三种办法，一种是定义一个临时数组，每次都保存上一次循环b（这里指一维数组）的值，然后
w更新时用临时数组的值更新，事实证明会超时
另一种是写两重循环，让w先更新，这一层的w都更新完以后再让b更新，事实证明也会超时
那么第三种方法就是，用一个变量来记录上一次循环b的值，每次迭代都更新这个变量就行，这和第一
种方法的思想相同，但省去了
数组间批量复制所耗用的大量时间，具体如何复制可以查看代码
*/
ll w[N];
ll b[N];
ll a[N];
ll tmp[N];
int main()
{
    IOS;
    while(cin>>m>>n)
    {
        for(ll i=1;i<=n;i++)
        {
            cin>>a[i];
        }
        memset(w,0,sizeof w);
        memset(b,0,sizeof b);
        /*for(ll i=1;i<=n;i++)//二维时的情况
        {
            for(ll j=1;j<=i;j++)
            {
                w[i][j]=max(b[i-1][j-1]+a[i],w[i-1][j]+a[i]);
                b[i][j]=max(b[i-1][j],w[i][j]);
            }
        }*/
        /*for(ll i=1;i<=n;i++)//压掉i这一维的情况
        {
            for(ll j=i;j>=1;j--)
            {
                w[j]=max(b[j-1]+a[i],w[j]+a[i]);
                b[j]=max(b[j],w[j]);
            }
        }*/
        /*for(ll j=1;j<=m;j++)//压掉j这一维的方案一
        {
            memcpy(tmp,b,sizeof b);
            for(ll i=j;i<=n;i++)
            {
                w[i]=max(tmp[i-1]+a[i],w[i-1]+a[i]);
                b[i]=max(b[i-1],w[i]);
            }
        }*/
        /*for(ll j=1;j<=m;j++)//压掉j这一维的方案二
        {
            for(ll i=j;i<=n;i++)
            {
                w[i]=max(b[i-1]+a[i],w[i-1]+a[i]);

            }
            for(ll i=j;i<=n;i++)
            {
                b[i]=max(b[i-1],w[i]);
            }
        }*/
        ll mm;
        for(ll j=1;j<=m;j++)
        {
            mm=-INF;
            for(ll i=j;i<=n;i++)
            {
                w[i]=max(b[i-1]+a[i],w[i-1]+a[i]);//此处b[i-1]是b[i-1][j-1]
                b[i-1]=mm;//此处b[i-1]是b[i-1][j]
                mm=max(mm,w[i]);//此处mm是b[i][j]
            }
        }
        cout<<mm<<endl;

    }
}
```

https://vjudge.net/problem/HDU-1069

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N = 100+5;
const ll mod=1e9+7;
ll n;
ll dp[N];//dp[i]=j表示以标号为i的方块处于最底层时可以达到的最大高度
ll cas=1;
struct node{
    ll x,y,z;
    int operator < (node&obj)
    {
        if(x==obj.x&&y==obj.y)return z>obj.z;
        if(x==obj.x)
        {
            return y<obj.y;
        }
        else
        {
            return x<obj.x;
        }
    }
};
vector<node>bk;
int main()
{
    IOS;
    while(cin>>n)
    {
        if(n==0)break;
        bk.clear();
        memset(dp,0,sizeof dp);
        ll mm=0;
        for(ll i=1;i<=n;i++)
        {
            ll a,b,c;
            cin>>a>>b>>c;
            //由于必须要求底面的两个坐标均小于下面的方块，所以不能简单地用面积表示底面的
属性
            bk.push_back({a,b,c});
            bk.push_back({b,a,c});
            bk.push_back({a,c,b});
            bk.push_back({c,a,b});
            bk.push_back({b,c,a});
            bk.push_back({c,b,a});
        }
        sort(bk.begin(),bk.end());
        ll ans=0;
        for(ll i=0;i<6*n;i++)
        {
            dp[i]=bk[i].z;
            for(ll j=i-1;j>=0;j--)
            {
                if(bk[j].x<bk[i].x&&bk[j].y<bk[i].y)
                {
                    dp[i]=max(dp[i],dp[j]+bk[i].z);
                }

            }
            ans=max(ans,dp[i]);
            //cout<<"dp["<<i<<"]=="<<dp[i]<<endl;
        }
        cout<<"Case "<<cas++<<": maximum height = ";
        cout<<ans<<endl;
    }
}
```

https://vjudge.net/problem/HDU-1087

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll n;
ll a[N];
ll dp[N];
int main()
{
	IOS;
	while(cin>>n)
	{
		if(!n)break;
		memset(dp,0,sizeof dp);
		for(ll i=1;i<=n;i++)
		{
			cin>>a[i];
		}
		for(ll i=1;i<=n;i++)
		{
			for(ll j=0;j<i;j++)
			{
				if(a[i]>a[j]){
					dp[i]=max(dp[i],dp[j]+a[i]);
				}
			}
		}
		ll ans=0;
		for(ll i=1;i<=n;i++)
		{
			ans=max(ans,dp[i]);
		}
		cout<<ans<<endl;
	}
}
```

https://codeforces.com/contest/1547/problem/E
```
#include<iostream>
#include<algorithm>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll int //注意要用int
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 3e5 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-4;
ll n,k;
ll l[N],r[N];//l[i]表示仅考虑i左边的各自时i的气温的最小值, r[i]相反
ll a[N];
ll pos[N];
void solve()
{
    cin>>n>>k;
    memset(pos,0,sizeof pos);
    memset(l,0,sizeof l);
    memset(r,0,sizeof r);
    for(ll i=1;i<=k;i++)
    {
        cin>>a[i];
    }
    for(ll i=1;i<=k;i++)
    {
        ll t;
        cin>>t;
        pos[a[i]]=t;
    }
    if(pos[1]==0)l[1]=INF;
    else l[1]=pos[1];
    for(ll i=2;i<=n;i++)
    {
        l[i]=min(l[i-1]+1,pos[i]!=0?pos[i]:INF);
    }
    if(pos[n]==0)r[n]=INF;
    else r[n]=pos[n];
    for(ll i=n-1;i>=1;i--)
    {
        r[i]=min(r[i+1]+1,pos[i]!=0?pos[i]:INF);
    }
    for(ll i=1;i<=n;i++)
    {
        cout<<min(l[i],r[i])<<' ';
    }
    cout<<endl;

}
int main()
{
    IOS;
    ll _;
    cin>>_;
    while(_--)
    {
        solve();
    }

}
```

https://www.luogu.com.cn/problem/P1044

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 4e5 + 5;
const int mod = 998244353;
const double eps = 1e-4;
ll dp[100][100];//dp[i][j]表示操作序列中还有i个数，栈中有j个数时可能的输出序列数
ll dfs(ll i,ll j)
{
    if(i==0)return 1;
    if(dp[i][j])return dp[i][j];
    if(j>0)dp[i][j]+=dfs(i,j-1);
    dp[i][j]+=dfs(i-1,j+1);
    return dp[i][j];
}
int main()
{
    IOS;
    ll n;
    cin>>n;
    dp[1][0]=1;
    cout<<dfs(n,0)<<endl;
}
```

https://www.luogu.com.cn/problem/P1990

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include<string>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e4;
const double eps = 1e-4;
ll n;
ll f[N];//覆盖到第i列时的方案数
ll g[N];//第i列只有一个方块涂色时的方案数（不失一般性，这里假定是上面的方块涂色）
int main()
{
    IOS;
    cin>>n;
    f[0]=1;
    f[1]=1;
    g[1]=0;
    for(ll i=2;i<=n;i++)
    {
        g[i]=(f[i-2]%mod+g[i-1]%mod)%mod;
        f[i]=(f[i-1]%mod+f[i-2]%mod+g[i-1]*2%mod)%mod;
    }
    cout<<f[n]<<endl;

}
```

https://www.luogu.com.cn/problem/P1115

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include <string>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
```

```
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e6 + 500;
const int mod = 10007;
const double eps = 1e-4;
ll n;
ll a[N];
ll dp[N];
int main()
{
	IOS;
	cin >> n;
	for (ll i = 1; i <= n; i++)
	{
		cin >> a[i];
	}
	for(ll i=1;i<=n;i++)
	{
		dp[i]=max(dp[i-1]+a[i],a[i]);
	}
	ll ans=-INF;
	for(ll i=1;i<=n;i++)
	{
		ans=max(ans,dp[i]);
	}
	cout<<ans<<endl;
}
```

https://codeforces.com/problemset/problem/1535/C

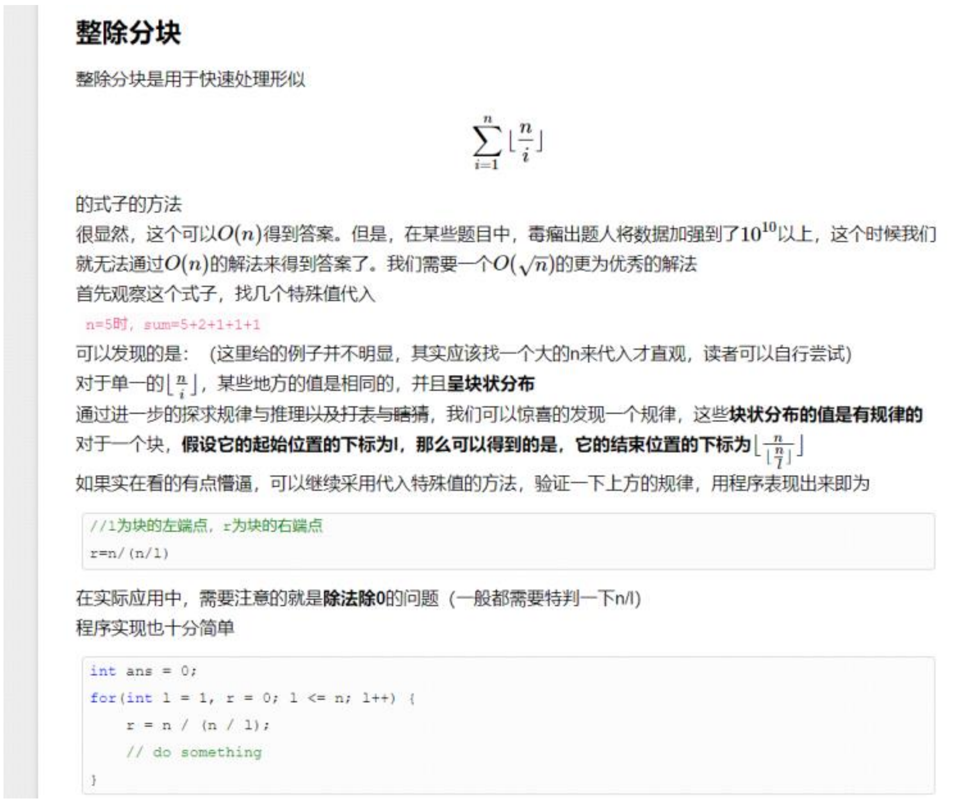
```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=2e5+5;
const ll mod=1e9+7;
ll dp[N][2];
void solve()
{
    string s;
    cin>>s;

    s="?"+s;
    ll ans=0;
    memset(dp,0,sizeof dp);
    for(ll i=1;i<s.length();i++)
    {
        if(s[i]=='1')dp[i][1]=dp[i-1][0]+1;
        else if(s[i]=='0')dp[i][0]=dp[i-1][1]+1;
        else{
            dp[i][1]=dp[i-1][0]+1;
            dp[i][0]=dp[i-1][1]+1;
        }
        ans+=max(dp[i][0],dp[i][1]);
    }
    cout<<ans<<endl;
}
int main()
{
    IOS;
    ll _;
    cin>>_;
    while(_--)
    {
        solve();
    }
}
```

https://codeforces.com/contest/1509/problem/C

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define ll long long
#define pll pair<ll,ll>
using namespace std;
const ll N=2e3+10;
const ll M=800;
const ll mod=998244353;
ll n;
ll a[N];
/*
考虑前i个元素，每次将第n-i+1大或n-i+1小的元素放在最后
将元素从小到大排序
比如对于最后一个位置，将第n个元素或第一个元素放在这里是最优，因为如果这两个元素在之前出现，那么它们之间的差一直会是最大
设dp[l][r]表示排序后的序列中子序列[l,r]的答案
转移方程是dp[l][r]=s[r]-s[l]+min(dp[l+1][r],dp[l][r-1])
*/
ll dp[N][N];
void solve()
{
    cin>>n;
    for(ll i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    sort(a+1,a+n+1);
    for(ll len=2;len<=n;len++)
    {
        for(ll i=1;i<=n-len+1;i++)
        {
            ll j=i+len-1;
            dp[i][j]=a[j]-a[i]+min(dp[i+1][j],dp[i][j-1]);
        }
    }
    cout<<dp[1][n]<<endl;
}
int main()
{
    IOS;
    solve();
}
```

整除分块

2021年9月8日 9:05



整除分块

整除分块是用于快速处理形似

$$\sum_{i=1}^{n}\lfloor\frac{n}{i}\rfloor$$

的式子的方法

很显然，这个可以$O(n)$得到答案。但是，在某些题目中，毒瘤出题人将数据加强到了10^{10}以上，这个时候我们就无法通过$O(n)$的解法来得到答案了。我们需要一个$O(\sqrt{n})$的更为优秀的解法

首先观察这个式子，找几个特殊值代入

`n=5时，sum=5+2+1+1+1`

可以发现的是：（这里给的例子并不明显，其实应该找一个大的n来代入才直观，读者可以自行尝试）

对于单一的$\lfloor\frac{n}{i}\rfloor$，某些地方的值是相同的，并且**呈块状分布**

通过进一步的探求规律与推理以及打表与瞎猜，我们可以惊喜的发现一个规律，这些**块状分布的值是有规律的**

对于一个块，**假设它的起始位置的下标为l，那么可以得到的是，它的结束位置的下标为**$\lfloor\frac{n}{\lfloor\frac{n}{l}\rfloor}\rfloor$

如果实在看的有点懵逼，可以继续采用代入特殊值的方法，验证一下上方的规律，用程序表现出来即为

```
//l为块的左端点，r为块的右端点
r=n/(n/l)
```

在实际应用中，需要注意的就是**除法除0**的问题（一般都需要特判一下n/l）

程序实现也十分简单

```
int ans = 0;
for(int l = 1, r = 0; l <= n; l++) {
    r = n / (n / l);
    // do something
}
```

```
int main()
{
    int t ;
    cin >> t ;
    while(t--)
    {
        int n ;
        cin >> n ;
        ll res = 0 ;
        for(int l = 1 , r ; l <= n ; l = r + 1)
        {
            r = n / (n / l) ;
            res += (r - l + 1 ) * (n / l );
        }
        //一维数论分块
        cout << res << endl;
    }
    return 0;
}
```

https://codeforces.com/problemset/problem/1561/D1

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
```

```cpp
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define ll long long
#define pll pair<ll,ll>
using namespace std;
const ll N=2e5+10;
const ll M=800;
const ll mod=998244353;
ll n,m;
ll dp[N];
ll pre[N];
/*
dp[x]:从x到1的走法数
dp[x]=dp[x-y](y from 1 to x-1)+dp[x/z](z from 2 to x)
第一项可以用前缀和处理
第二项至多有2*sqrt(x)个不同的值，因为一个数至多有2*sqrt(x)个因数
用整除分块计算第二项
*/
ll add(ll a,ll b)
{
    return ((a%m)+(b%m))%m;
}
ll mul(ll a,ll b)
{
    return ((a%m)*(b%m))%m;
}
ll div(ll x)
{
    ll res=0;
    for(int l=1,r=1;l<=x;l=r+1)
    {
        r=x/(x/l);
        res=add(res,mul((r-l+1),dp[x/l]));
    }
    return res-dp[x];
}
void solve()
{
    cin>>n>>m;
    dp[1]=1;
    pre[1]=1;
    for(int i=2;i<=n;i++)
    {
        dp[i]=add(pre[i-1],div(i));
        pre[i]=add(pre[i-1],dp[i]);
    }
    cout<<dp[n]<<endl;
}
int main()
{
    IOS;
    solve();
}
```

递推

2021年7月13日 23:03

https://www.luogu.com.cn/problem/P3612

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include<string>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e4;
const double eps = 1e-4;
string s;
ll n;
int main()
{
    IOS;
    cin>>s;
    cin>>n;
    ll m=s.length();
    ll t=m;
    while(t<n)
    {
        t*=2;
    }
    while(t>m)
    {
        //cout<<"t=="<<t<<' '<<"n=="<<n<<endl;
        if(n<=t/2)
        {
            t/=2;//若当前位置在上一个串的前半段，不用再进行转换
            continue;
        }
        t/=2;
        n-=t;
        n=(n-1)%t;//计算出当前位置在上一个串的位置
        if(n==0)n=t;
    }
    cout<<s[n-1]<<endl;

}
```

https://www.luogu.com.cn/problem/P1498

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include<string>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e4;
const double eps = 1e-4;
ll n;
char mp[3000][3000];
int main()
{
    IOS;
    cin>>n;
    memset(mp,' ',sizeof mp);
    ll h=2;
    ll w=4;
    mp[1][2]='/';
    mp[1][3]='\\';
    mp[2][1]='/';
    mp[2][2]='_';
    mp[2][3]='_';
    mp[2][4]='\\';
    for(ll i=2;i<=n;i++)
    {
        for(ll i=h+1;i<=h*2;i++)//原图案向下复制一份
        {
            for(ll j=1;j<=w;j++)
            {
                mp[i][j]=mp[i-h][j];
            }
        }
        for(ll i=h+1;i<=h*2;i++)//下面图案向右复制一份
        {
            for(ll j=w+1;j<=w*2;j++)
            {
                mp[i][j]=mp[i][j-w];
            }
        }
        for(ll i=1;i<=h;i++)//擦掉原来的图案
        {
            for(ll j=1;j<=w;j++)
            {
                mp[i][j]=' ';
            }
        }
        for(ll i=1;i<=h;i++)//下面的图案向上复制一份
        {
            for(ll j=w/2+1;j<=w/2*3;j++)
            {
                mp[i][j]=mp[i+h][j-w/2];
            }
        }
        h*=2;
        w*=2;
    }
    for(ll i=1;i<=h;i++)
    {
        for(ll j=1;j<=w;j++)
        {
            cout<<mp[i][j];
        }
        cout<<endl;
    }
}
```

https://www.luogu.com.cn/problem/P1228

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include<string>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e4;
const double eps = 1e-4;
ll k;
ll a,b;
void solve(ll k,ll cy,ll cx,ll x,ll y)//cy,cx表示当前方块左上角方块的坐标, x,y表示当
前方块中障碍的坐标
{
    if(k==0)
    {
        return;
    }
    ll xx = pow(2, k-1);//下一个小方块的长度
    ll yy = pow(2, k-1);
    if(x<cx+xx&&y<cy+yy)//如果障碍在左上角的小方块里
    {
        cout<<cy+yy<<' '<<cx+xx<<' '<<1<<endl;//人为在中央放一块地毯, 视作障碍, 这样
其他三个块也各有一块障碍
        solve(k - 1, cy, cx,x,y);
        solve(k - 1, cy, xx+cx,cx+xx,cy+yy-1);
        solve(k - 1, yy+cy, cx,cx+xx-1,cy+yy);
        solve(k - 1, yy+cy, xx+cx,cx+xx,cy+yy);
    }
    else if(x>=cx+xx&&y<cy+yy)
    {
        cout<<cy+yy<<' '<<cx+xx-1<<' '<<2<<endl;
        solve(k - 1, cy, cx,cx+xx-1,cy+yy-1);
        solve(k - 1, cy, xx+cx,x,y);
        solve(k - 1, yy+cy, cx,cx+xx-1,cy+yy);
        solve(k - 1, yy+cy, xx+cx,cx+xx,cy+yy);
    }
    else if(x<cx+xx&&y>=cy+yy)
    {
        cout<<cy+yy-1<<' '<<cx+xx<<' '<<3<<endl;
        solve(k - 1, cy, cx,cx+xx-1,cy+yy-1);
        solve(k - 1, cy, xx+cx,cx+xx,cy+yy-1);
        solve(k - 1, yy+cy, cx,x,y);
        solve(k - 1, yy+cy, xx+cx,cx+xx,cy+yy);
    }
    else if(x>=cx+xx&&y>=cy+yy)
    {
        cout<<cy+yy-1<<' '<<cx+xx-1<<' '<<4<<endl;
        solve(k - 1, cy, cx,cx+xx-1,cy+yy-1);
        solve(k - 1, cy, xx+cx,cx+xx,cy+yy-1);
        solve(k - 1, yy+cy, cx,cx+xx-1,cy+yy);
        solve(k - 1, yy+cy, xx+cx,x,y);
    }


}
int main()
{
    IOS;
    cin>>k>>a>>b;
    solve(k,1,1,b,a);
}
```

分解质因数

2021年6月11日　　10:25

https://www.acwing.com/problem/content/869/

```
#include<bits/stdc++.h>
using namespace std;
void divide(int x)
{
   for (int i = 2; i <= x / i; i++)
      if (x % i == 0)
      {
         int s = 0;
         while (x % i == 0) x /= i, s++;
         cout << i << ' ' << s << endl;
      }
   if (x > 1) cout << x << ' ' << 1 << endl;
   cout << endl;
}
int main()
{
   int x;
   cin >> x;
   divide(x);
}
```

https://codeforces.com/problemset/problem/1538/D

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define inf 0x3f3f3f3f
#define ll int
#define pii pair<int,int>
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 2e5 + 5;
const int mod = 1e9 + 7;
ll a, b, k;
ll gcd(ll a, ll b)
{
        if (a < b)swap(a, b);
        if (!b)return a;
        else return gcd(b, a % b);
}
vector<int>prime;
```

```
int vis[N] = { 0 };
int sieve(int b)//欧拉筛
{
	int cnt = 0;
	for (int i = 2; i <= b; i++)
	{
		if (!vis[i]) {
			prime.push_back(i);
			cnt++;
		}
		for (int j = 0; j <= cnt && i * prime[j] <= b; j++)
		{
			vis[i * prime[j]] = 1;
			if (i % prime[j] == 0)break;//如果i为prime[j]的倍数即i=k*prime[j],那么在筛下一
			个时i*prime[j+1]=prime[j]*k*prime[j+1],在i=k*prime[j+1]时prime[j]就会被重复
			筛一次，所以这里要跳出
		}
	}
	return cnt;
}

ll divide(int x)
{
	ll res = 0;
	for (auto i : prime)
	{
		if (i * i > x)break;//x至多包含一个大于sqrt(x)的质因子，若有两个以上，彼此间的乘
		积会大于x
		while (x % i == 0)res++,x/=i;
	}
	if (x > 1) res++;
	return res;

}
inline int read() {
	int x = 0, f = 1;
	char ch = getchar();
	while (ch < '0' || ch>'9') {
		if (ch == '-')
			f = -1;
		ch = getchar();
	}
	while (ch >= '0' && ch <= '9') {
		x = (x << 1) + (x << 3) + (ch ^ 48);
		ch = getchar();
	}
	return x * f;
}
void solve()
{
	scanf("%d%d%d", &a, &b, &k);

	if (a == b && k == 1)
	{
		cout << "NO" << endl;
		return;
	}

	ll m;
	if (a == b)
	{
		m = 0;
	}
	else if (a%b==0||b%a==0)
	{
		m = 1;
	}
	else
	{
		m = 2;
	}
	ll n = divide(a) + divide(b);
	if (m <= k && n >= k)
	{
		cout << "YES" << endl;
	}
	else
	{
		cout << "NO" << endl;
	}


}

int main()
{

	int t;
	scanf("%d", &t);
	sieve(1e5);
	while (t--)
	{
		solve();
	}
}
```

https://www.luogu.com.cn/problem/P1069

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define pll pair<ll,ll>
```

```
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=4e5+5;
const ll mod=1e9+7;
ll a[N];
ll n;
ll m1,m2;
bool vis[N];
vector<ll>prime;
map<ll,map<ll,ll> >mp;
void sieve()
{
    ll cnt=0;
    for(ll i=2;i<=300000;i++)
    {
        if(!vis[i])
        {
            prime.push_back(i);
            cnt++;
        }
        for(ll j=0;j<cnt&&i*prime[j]<=300000;j++)
        {
            vis[i*prime[j]]=1;
            if(i%prime[j]==0)break;
        }
    }
}
int main()
{
    IOS;
    sieve();
    cin>>n;
    cin>>m1>>m2;
    for(ll i=0;i<n;i++)
    {
        cin>>a[i];
    }
    ll ans=INF;
    for(ll i=0;i<n;i++)
    {
        ll k=0;
        ll tmp=m1;
        int f=0;
        for(auto j:prime)
        {
            ll cnt=0;
            ll cnt2=0;
            if(j*j>tmp)break;
            if(tmp%j!=0)continue;
            while(tmp%j==0)
            {
                tmp/=j;
                cnt++;
            }
            if(a[i]%j!=0)
            {
                f=1;
                break;
            }
            while(a[i]%j==0)
            {
                a[i]/=j;
                cnt2++;
            }
            k=max(k,(ll)ceil((double)m2*(double)cnt/(double)cnt2));
        }
        if(tmp>1)
        {
            if(a[i]%tmp!=0)
            {
                f=1;
            }
            else{
                ll cnt=0;
                while(a[i]%tmp==0)
                {
                    a[i]/=tmp;
                    cnt++;
                }
                k=max(k,(ll)ceil((double)m2/(double)cnt));
            }
        }
        if(f)continue;
        ans=min(ans,k);
    }
    if(ans==INF)
    {
        cout<<-1<<endl;
    }
    else{
        cout<<ans<<endl;
    }
}
```

生成指定范围内随机数

2021年7月27日 3:25

法一:

```
seed = time(0);
srand(seed);
const int MIN_VALUE = 10;
const int MAX_VALUE = 18;
number = rand() % (MAX_VALUE - MIN_VALUE + 1) + MIN_VALUE;
```

法二:

```
mt19937 eng(time(0));
int randint(int a, int b) // 生成a到b之间的随机数
{
    uniform_int_distribution<int> dis(a, b);
    return dis(eng);
}
```

构造

2021年6月19日　　10:35

https://codeforces.com/contest/1537/problem/C

将序列排序，找到差最小的一对数对a[i]和a[i+1]，然后将a[i+1]~a[n]放到前面，a[1]~a[i]放到后面

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
ll n;
ll h[N];
ll tmp[N];
void solve()
{
	cin >> n;
	for (ll i = 1; i <= n; i++)
	{
		cin >> h[i];
	}
	sort(h + 1, h + 1 + n);
	if (n == 2)
	{
		for (ll i = 1; i <= n; i++)
		{
			cout << h[i] << ' ';
		}
		cout << endl;
		return;
	}
	ll s = 0, t = 0;
	ll minn = 1e9;
	for (ll i = 2; i <= n; i++)
	{
			if (h[i] - h[i - 1] < minn)
			{
				minn = h[i] - h[i - 1];
				t = i;
				s = i - 1;
			}
		}
		for (ll i = t; i <= n; i++)
			cout << h[i] << ' ';
		for (ll i = 1; i <= s; i++)
		{
			cout << h[i] << ' ';
		}
		cout << endl;
}
int main()
{
		IOS;
		ll t;
		cin >> t;
		while (t--)
		{
			solve();
		}
}
```

涉及到位运算的构造：

https://codeforces.com/contest/1547/problem/D

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 2e5 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-4;
ll n;
ll x[N];
vector<ll>ans;
void solve()
{
    cin>>n;
    ans.clear();
    for(ll i=1;i<=n;i++)
    {
        cin>>x[i];
    }
    ans.push_back(0);
    ans.push_back(0);
    for(ll i=2;i<=n;i++)
    {
        ll t=ans[i-1] ^ x[i-1];
        /*
            对于t的每一位:
            若该位为0，则(ans[i]^x[i])的这一位可以取0或1，由于字典序最小所以
            ans[i]的这一位取0
            若该位为1，若x[i]的这一位为1，ans[i]这一位要取0，若x[i]这一位为0，ans[i]
这一位要取1
        */
        ll k=0;
        for(ll j=30;j>=0;j--)
        {
            if((t & (1<<j))!=0 && (x[i] & (1<<j))==0)//注意每个条件外面一定要包裹括
号，否则编译器会把&和&&搞混
            {
                k|=(1<<j);
            }
        }
        ans.push_back(k);

    }
    for(ll i=1;i<ans.size();i++)
    {
        cout<<ans[i]<<' ';
    }
    cout<<endl;

}
int main()
{
    IOS;
    ll t;
    cin>>t;
    while(t--)
    {
        solve();
    }

}
```

数学

2021年6月19日 10:40

https://codeforces.com/contest/1537/problem/D
一些结论：
奇数±奇数=偶数，偶数±偶数=偶数，奇数±偶数=奇数，偶数×偶数=偶数，奇数×偶数=偶数，奇数×奇数=奇数

也就是说，一个奇数的因数都是奇数

那么来考虑这道题，我们分三种情况：
n是奇数时。当n是奇数时，n的每一个因数都是奇数，又由于奇数加偶数等于奇数，所以n减去一个因数必然得到一个偶数(n-1)*Divisor,因为Divisor为奇数，所以此时得到的偶数必然不是2的次幂（2的次幂的所有因数都是偶数）。
n是偶数且不是2的次幂时。n必然存在奇数因子Divisor，那么(n-Divisor)必然是奇数。我们考虑到当n成为质数时，游戏会结束，而质数不是奇数就是2的次幂，所以最优的解法是每次令n成为一个奇数给对方，那么此时n要么是质数，要么可以再次被转化为不是2的次幂的偶数。
所以目前为止，第一种情况必输，第二种情况必胜。
第三种情况，n是偶数且是2的次幂时，我们有两种选择，一是将n变为原来的一半，一是将n变成不是2的次幂的偶数，第二种选择会令对方必胜，所以我们选第一种，在这种情况下，如果log2(n)为奇数，bob赢；反之alice赢；

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
ll n;
void solve()
{
    cin>>n;
    if(n%2==1)
    {
        cout<<"Bob"<<endl;
        return;
    }
    ll p=0;
    ll t=n;
    while(t%2==0)
    {
        t/=2;
        p++;
    }
    if(n%2==0&&t!=1)
    {
        cout<<"Alice"<<endl;
        return;
    }
    if(p%2!=0)
    {
        cout<<"Bob"<<endl;
    }
    else{
        cout<<"Alice"<<endl;
    }
}
int main()
{
    IOS;
    ll t;
    cin>>t;
    while(t--)
    {
        solve();
    }
}
```

https://www.luogu.com.cn/problem/P1072

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<set>
#include<stdio.h>
#include<ctime>
#include<random>
#define INF 0x3f3f3f3f
#define ll long long
#define ull unsigned long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=1e6+5;
const ll mod=1e9+7;
ll l,r;
ll gcd(ll a,ll b)
{
    if(!b)return a;
    else return gcd(b,a%b);
}
int main()
{
    IOS;
    ll a0,a1,b0,b1;
    ll n;
    cin>>n;
    while (n--)
    {
        cin >> a0 >> a1 >> b0 >> b1;
        ll ans = 0;
        for (ll x = 1; x*x <= b1; x++)
        {
            if(b1%x!=0)continue;
            if(x%a1==0&&gcd(x/a1,a0/a1)==1&&gcd(b1/b0,b1/x)==1)
            {
               ans++;
            }
            ll t=b1/x;
            if(t==x)continue;
            if(t%a1==0&&gcd(t/a1,a0/a1)==1&&gcd(b1/b0,b1/t)==1)
            {
               ans++;
            }
        }
        cout << ans << endl;
    }
}
```

计数问题

2021年6月11日 11:30

https://codeforces.com/contest/1538/problem/F

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#pragma optimize(2)
#define INF 0x3f3f3f3f
#define ll long long
#define pii pair<int,int>
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 2e5 + 5;
const int mod = 1e9 + 7;
ll n;
ll func(string num)
{
	ll res = 0;
	ll t = 0;
	for (ll i = 0; i < num.length(); i++)
	{
		res += num[i] - '0';

		if(i!=0)
		res += t*10;
		t = t * 10 + num[i] - '0';
	}
	return res;
}
void solve()
{
	string a, b;
	cin >> a >> b;
	cout << func(b) - func(a) << endl;
}
int main()
{
	IOS;
	ll t;
	cin >> t;
	while (t--)
	{
		solve();
	}
}
```
https://codeforces.com/contest/1541/problem/B
```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll a[N];
int main()
{
	IOS;
	ll t;
	cin>>t;
	while(t--)
	{
		ll n;
		cin>>n;
		map<ll,int>m;
		for(ll i=1;i<=n;i++)
		{
			cin>>a[i];
			m[a[i]]=1;
		}
		ll ans=0;
		for(ll i=1;i<=n;i++)
		{
			for(ll j=a[i]-i;j<=n;j+=a[i])
			{
				if(i<j&&i+j==a[i]*a[j])
				{
					ans++;
				}
			}
		}
		cout<<ans<<endl;

	}
}
```

https://codeforces.com/contest/1546/problem/D
```
#include<iostream>
#include<algorithm>
```

```
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cmath>
#include<set>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 4e5 + 5;
const int mod = 998244353;
const double eps = 1e-4;
ll n;
int a[N];
ll fac[N];
ll quick_pow(ll a, ll b)
{
    ll result = 1;
    while (b)
    {
        if (b & 1)
        {
            result = result * a % mod;
        }
        b >>= 1;
        a = a * a % mod;
    }
    return result % mod;
}
ll c(ll m, ll n)
{
    if (n > m)return 0;
    return (fac[m] * quick_pow(fac[n], mod - 2) % mod * quick_pow(fac[m - n], mod - 2
) % mod) % mod;//利用乘法逆元化简组合数公式
}
void solve()
{
    cin>>n;
    string s;
    cin>>s;
    ll a=0;
    ll b=0;
    for(ll i=0;i<s.length()-1;i++)
    {
        if(s[i]=='1'&&s[i+1]=='1')//将相邻的1看作整体，问题迎刃而解
        {
            a++;
            i++;
        }
        if(s[i]=='0')
        {
            b++;
        }
    }
    if(s[s.length()-1]=='0')b++;
    cout<<c(a+b,a)<<endl;
}
int main()
{
    IOS;
    ll _;
    cin>>_;
    fac[0] = 1;
    for (ll i = 1; i < N; i++)
    {
        fac[i] = fac[i - 1] * i % mod; //预处理，数组fav[i]保存数i的阶乘
    }
    while(_--)
    solve();

}
```

https://codeforces.com/problemset/problem/1550/D

```
#include<iostream>
#include<cmath>
#include<algorithm>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 2e5 + 5;
const ll mod = 1e9+7;
const double eps = 1e-4;
/*
首先考虑怎样构造数列a使:
1.数列a是good array
2.数列a的f(a)是所有长度为n的good array中最大的
首先由于数列a是good array，也就是a[i]!=i,我们不妨对每个元素设置一个相对其下标的偏移量k，即a[i]=i+k[i];
由于F(a)是a中满足a[i]+a[j]=i+j的数对的数量，而a[i]+a[j]=i+k[i]+j+k[j],所以对于满足条件的数对，有k[i]=-k[j];
那么很容易想到，如果我们要构造F(a)最大的good array,只需要令所有元素的偏移量的绝对值相同，一半拥有+k的偏移量,
另一半拥有-k的偏移量，那么两组元素就可以互相配对组成满足条件的数对，对于这样构造出的规模为n的数组,
其F(a)=(n/2)*(n/2),  (若n为奇数则其中一项向上取整)
```

```
接下来我们考虑k该怎样选，k首先必然大于等于1（否则等于没有偏移）
首先我们发现，如果k满足：
l<=i+k<=r  (1)
l<=i-k<=r  (2)
对所有1<=i<=n成立
即：
l-i<=k<=r-i (3)
i-r<=k<=i-l (4)
注意到-1e9<=l<=1,n<=r<=1e9
故(3)(4)两式中k的左界均不大于0
故最终k满足1<=k<=min(1-l,r-n)，那么此时对于所有1<=i<=n,i+k与i-k均在[l,r]的范围内
此时我们可以令a[1]~a[n]中任意一半数等于i+k,另外一半数等于i-k，所有a[i]都会在[l,r]内，
若n为偶数，总共有C(n,n/2)中选法，若n为奇数，共有C(n,n/2)+C(n,n/2+1)种选法
而对于k>min(1-l,r-n)的情况，如果随意指派a[i]=k+i或k-i，有些元素就有可能会超出范围
所以，我们只能令[1,max(1,l+k))范围内的元素为i+k，(min(n,r-k),n]范围内的元素为i-k
而对于在min(n,r-k)-max(1,l+k)+1之间的元素则没有限制可以随意选，我们仍然按照我们之前的
原则选就好
设lf=max(1,l+k),rg=min(n,r-k)，在这种情况下，如果n为偶数，总共有C(rg-lf+1,n/2-(lf-1))
种选法，如果n为奇数，总共有
C(rg-lf+1,n/2-(lf-1))+C(rg-lf+1,n/2+1-(lf-1))种选法
*/
ll fac[N];
ll inv[N];
ll mul(ll a,ll b)
{
    return ((a%mod)*(b%mod))%mod;
}
ll quick_pow(ll base, ll power)
{
    ll res = 1;
    while (power)
    {
        if (power & 1)
        {
            res = mul(res,base);
        }
        power >>= 1;
        base=mul(base,base);
    }
    return res % mod;
}
ll comb(ll m, ll n)
{
    if (n > m||n<0)return 0;
    return mul(fac[m],mul(inv[n],inv[m-n]));
}
void solve()
{
    ll n,l,r;
    cin>>n>>l>>r;
    ll k=min(1-l,r-n);
    ll ans=mul(k,comb(n,n/2));
    if(n&1)
    {
        ans+=mul(k,comb(n,n/2+1));
    }
    k++;
    while(1)
    {
        ll lf=max(1ll,l+k),rg=min(n,r-k);
        if(rg-lf+1<0)
        {
            break;
        }
        if (n % 2 == 0)
        {
            ans = (ans % mod + comb(rg - lf + 1, n / 2 - (lf - 1)) % mod) % mod;
        }
        else
        {
            ans = ((ans % mod + comb(rg - lf + 1, n / 2 - (lf - 1)) % mod) % mod + co
mb(rg - lf + 1, n / 2 + 1 - (lf - 1)) % mod) % mod;
        }
        k++;
    }
    cout<<ans<<endl;
}
int main()
{
    IOS;
    ll _;
    cin>>_;
    fac[0] = 1;
    inv[0] = 1;
    for (ll i = 1; i < N; i++)
    {
        fac[i] = mul(fac[i - 1],i);//预处理，数组fav[i]保存数i的阶乘
        inv[i]=  quick_pow(fac[i],mod-2);
    }
    while(_--)
    {
        solve();
    }
    /*ll a,b;
    cin>>a>>b;
    cout<<comb(a,b)<<endl;*/
}
```

https://www.luogu.com.cn/problem/P2822

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
```

```
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=4e3+5;
const ll mod=1e9+7;
ll c[N][N];
ll ans[N][N];
int main()
{
    IOS;
    ll t,k;
    cin>>t>>k;
    c[0][0]=1;
    c[1][0]=c[1][1]=1;
    for (int i = 2; i <= 2e3; i++)
    {
        c[i][0]=1;
        for (int j = 1; j <= i; j++)
        {
            c[i][j] = (c[i - 1][j]%k + c[i - 1][j - 1]%k)%k;
            ans[i][j]=ans[i-1][j]+ans[i][j-1]-ans[i-1][j-1];//二维前缀和保存答案
            if(c[i][j]%k==0)
            {
                ans[i][j]++;
            }
        }
        ans[i][i+1]=ans[i][i];//递推式中当i==j时要用到ans[i-1][j], 而对于任意ans[n]
[m], m大于n时只需要计算ans[n][n]
    }
    while(t--)
    {
        ll n,m;
        cin>>n>>m;
        if(n>=m)cout<<ans[n][m]<<endl;
        else cout<<ans[n][n]<<endl;
    }

}
```

https://www.luogu.com.cn/problem/P2789

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
```

```
#include<map>
#include<set>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const ll N=4e3+5;
const ll mod=1e9+7;
ll dp[N];
/*
注意题意不是n条直线相交最多能有多少条交点，而是能有多少种不同的交点数;
比如两条直线相交就可能没有交点（平行），或有一个交点;
对于n条直线，如果其中p条平行，那么剩余n-p条直线和这p条直线的交点数有p*(n-p)条
然后只需要考虑n-p条直线有多少种相交情况即可，这是一个子问题
*/
int vis[100000];
void solve(ll n,ll step)
{
    if(n==0)
    {
        vis[step]=1;
    }
    for(ll i=n;i>=1;i--)
    {
        solve(n-i,step+i*(n-i));
    }
}
int main()
{
    IOS;
    ll n;
    cin>>n;
    solve(n,0);
    ll ans=0;
    for(ll i=0;i<=1e5;i++)
    {
        if(vis[i])ans++;
    }
    cout<<ans<<endl;
}
```

https://atcoder.jp/contests/abc217/tasks/abc217_f

```
#include<bits/stdc++.h>
#define INF 0x3f3f3f3f
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
#define ll long long
#define pll pair<ll,ll>
using namespace std;
```

```
const ll N=1e3+5;
const ll M=800;
const ll mod=998244353;
ll n,m;
int good[M][M];
ll dp[M][M];
int vis[M][M];
/*
设dp[l][r]表示区间[l,r]中的答案数
设x为区间[l,r]中与l配对且区间[l+1,x-1]中的元素数为偶数的元素
那么为了计算dp[l][r]，我们需要先计算dp[l+1][x-1]和dp[x+1][r]
为了取光dp[l+1][x-1]中的元素，我们需要p=(x-1)-(l+1)+1=x-l-1次操作
为了取光dp[x+1][r]中的元素，我们需要q=r-(x+1)+1=r-x次操作
现在需要考虑两类操作所有的排列顺序：对于同一个区间的操作，其相互之间的相对顺序不变，
但是两种操作可以相互穿插
比如对第一个区间有操作 a1,a2,a3，对第二个区间有操作b1,b2,b3
我们可以将其这样排列：a1,b1,a2,a3,b2,b3
但是不能是a3,b1,b2,a1,a2,b3，因为a3不能在a1,a2之前
我们考虑将第一类操作看作小球，将第二类操作看作隔板
那么根据隔板法，我们要在p+q+1个空隙里插入隔板，总共的方式有C(p+q+1,q)种
总共的答案就是dp[l+1][x-1]*dp[x+1][r]*C(p+q+1,q)
注意特判,dp[l][l+1]=dp[l+1][l]*dp[l+2][l+1]*C(1,0)
故当r<l时，dp[l][r]=1
*/
ll fac[N];
ll inv[N];
ll mul(ll a,ll b)
{
    return ((a%mod)*(b%mod))%mod;
}
ll add(ll a,ll b)
{
    return ((a%mod)+(b%mod))%mod;
}
ll quick_pow(ll base, ll power)
{
    ll res = 1;
    while (power)
    {
        if (power & 1)
        {
            res = mul(res,base);
        }
        power >>= 1;
        base=mul(base,base);
    }
    return res % mod;
}
ll comb(ll m, ll n)
{
    if (n > m||n<0)return 0;
    return mul(fac[m],mul(inv[n],inv[m-n]));
}
ll dping(ll l,ll r)
{
    if(vis[l][r])
    {
        return dp[l][r];
    }
    if(l>r)
    {
        dp[l][r]=1;
        vis[l][r]=1;
        return 1;
    }
    for(ll x=l+1;x<=r;x++)
    {
        if(good[l][x]==1&&(x-l+1)%2==0)
        {
            ll p=(x-l-1)/2;
            ll q=(r-x)/2;
            dp[l][r]=add(dp[l][r],mul(mul(dping(l+1,x-1),dping(x+1,r)),comb(p+q+
1,q)));
        }
    }
    vis[l][r]=1;
    return dp[l][r];
}
void solve()
{
    cin>>n>>m;
    while(m--)
    {
        ll a,b;
        cin>>a>>b;
        good[a][b]=good[b][a]=1;

    }
    cout<<dping(1,2*n)<<endl;
}
int main()
{
    IOS;
    fac[0] = 1;
    inv[0] = 1;
    for (ll i = 1; i < N; i++)
    {
        fac[i] = mul(fac[i - 1],i);//预处理，数组fav[i]保存数i的阶乘
        inv[i]=  quick_pow(fac[i],mod-2);
    }
    solve();
}
```

树&图上DP

2021年7月22日 1:08

https://www.luogu.com.cn/problem/P4017

```
#include<iostream>
#include<cmath>
#include<vector>
#include<algorithm>
#include<cstring>
#include<queue>
#include<map>
#include<set>
#define INF 3e9
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const ll N = 1e6 + 5;
const ll mod = 80112002;
const double eps = 1e-4;
ll n,m;
vector<ll>g[N];
int vis[N];
ll ans;
ll deg[N];
ll outdeg[N];
ll dp[N];
void topo_sort()
{
    queue<ll>q;
    for(ll i=1;i<=n;i++)
    {
        if(deg[i]==0)
        {
            q.push(i);
            dp[i]=1;
        }
    }
    while(!q.empty())
    {
        ll t=q.front();
        q.pop();
        for(auto i:g[t])
        {
            deg[i]--;
            dp[i]=(dp[i]%mod+dp[t]%mod)%mod;
            if(deg[i]==0)
            {
                q.push(i);
            }
        }
    }
    for(ll i=1;i<=n;i++)
    {
        if(outdeg[i]==0)
        {
            ans=(ans%mod+dp[i]%mod)%mod;
        }
    }
}
int main()
{
    IOS;
    cin>>n>>m;
    while(m--)
    {
        ll a,b;
        cin>>a>>b;
        g[b].push_back(a);
        deg[a]++;
        outdeg[b]++;
    }
    topo_sort();
    cout<<ans<<endl;
}
```

卡特兰数

2021年7月13日　　2:04

卡特兰数有四个公式，只要记下面例子中这个就足够了

https://www.luogu.com.cn/problem/P1044

```
#include <iostream>
#include <algorithm>
#include <queue>
#include <stack>
#include <vector>
#include <cstring>
#include <cmath>
#include <set>
#include <iomanip>
#include <map>
#include <stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll, ll>
#define pdl pair<double, ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false), cin.tie(0), cout.tie(0)
using namespace std;
const int N = 1e2 + 5;
const int mod = 1e9;
const double eps = 1e-4;
ll c[N][N];
int main()
{
    IOS;
    ll n;
    cin >> n;
    for(ll i=1;i<=2*n;i++)
    {
        c[i][1]=c[i][i]=1;
    }
    for (int i = 3; i <= 2 * n; i++)
    {
        for (int j = 2; j < i; j++)
        {
            c[i][j] = c[i - 1][j] + c[i - 1][j - 1];
        }
    }
    cout << c[2*n][n]-c[2*n][n-1] << endl;//卡特兰数
}
```

推公式

2021年6月21日 16:55

https://codeforces.com/contest/1539/problem/A

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
int main()
{
    IOS;
    ll t;
    cin>>t;
    while(t--)
    {
        ll n,x,time;
        cin>>n>>x>>time;
        if(time<x)
        {
            cout<<0<<endl;
        }
        else{
            ll tmp=time/x;//一个区间内有多少开始的人
            if(n-tmp>0)
            cout<<(n-tmp)*tmp+tmp*(tmp-1)/2<<endl;//最后time/x个人在他们的区间内开始的人不够time/x，而是time/x-1,time/x-2,...,0个人
            else cout<<n*(n-1)/2<<endl;//如果总人数小于理论上区间开始的人数，直接算总人数的不满意度即可
        }
    }

}
```

排序问题

2021年6月21日 17:56

https://codeforces.com/contest/1539/problem/C

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<vector>
#include<cstring>
#include<cmath>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 1e6 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-8;
ll a[N];
vector<ll>inter;
int main()
{
    IOS;
    ll n,k,x;
    cin>>n>>k>>x;
    for(ll i=1;i<=n;i++)
    {
        cin>>a[i];
    }
    sort(a+1,a+1+n);
    ll ans=1;
    for(ll i=2;i<=n;i++)
    {
        if(a[i]-a[i-1]>x)
        {
            inter.push_back(a[i]-a[i-1]-1);
            ans++;
        }
    }
    sort(inter.begin(),inter.end());//区间从小到大排序，从最小的开始加
    for(auto i:inter)
    {
        ll num=i/x;
        if(k>=num)
        {
            k-=num;
            ans--;
        }
        else{
            break;
        }
    }
    cout<<ans<<endl;
}
```

https://codeforces.com/contest/1546/problem/C

```
#include<iostream>
#include<algorithm>
#include<queue>
#include<stack>
#include<vector>
#include<cstring>
#include<cmath>
#include<set>
#include<iomanip>
#include<map>
#include<stdio.h>
#define INF 0x3f3f3f3f
#define ll long long
#define pll pair<ll,ll>
#define pdl pair<double,ll>
#define pdd pair<double, double>
#define IOS ios::sync_with_stdio(false),cin.tie(0),cout.tie(0)
using namespace std;
const int N = 4e5 + 5;
const int mod = 1e9 + 7;
const double eps = 1e-4;
ll n;
ll a[N];
vector<ll>odd;
vector<ll>even;
/*
首先，对每个数，只有它经过偶数次交换时，其方向保持不变
其次，奇数位置的数交换偶数次还在奇数位置，偶数位置的数交换偶数次还在偶数位置
那么，我们可以奇数位置的数取出来进行排序，这个序列就是最终序列中奇数位置的数的分布情
况，偶数位置的数也如法炮制
两个数列合并，就是最终序列，如果这个序列不是非降的，那么答案是NO
*/
void solve()
{
    cin>>n;
    odd.clear();
    even.clear();
    for(ll i=1;i<=n;i++)
    {
        cin>>a[i];
        if(i%2==1)odd.push_back(a[i]);
        else even.push_back(a[i]);
    }
    sort(odd.begin(),odd.end());
    sort(even.begin(),even.end());
    ll j=0,k=0;
    for(ll i=1;i<=n;i++)
    {
        if(i%2==1)
        {
            a[i]=odd[j++];
        }
        else{
            a[i]=even[k++];
        }
    }
    for(ll i=1;i<n;i++)
    {
        if(a[i]>a[i+1])
        {
            cout<<"NO"<<endl;
            return;
        }
    }
    cout<<"YES"<<endl;
}
int main()
{
    IOS;
    ll _;
    cin>>_;
    while(_--)
    solve();

}
```

矩阵链乘法

2022年9月18日 22:26

```
#include<bits/stdc++.h>
using namespace std;
const int N = 2e3 + 5;
int str[N];
int cost[N][N];
int last[N][N];
int n;
/*
5
30 1 40 10 25
*/
void printTrace(int l, int r)
{
	if(r - l + 1 < 3)return;
	
	if(r - l + 1 == 3)
	{
		cout << "[" << l << "," << last[l][r] << "] [" << last[l][r] << "," << r << "]" << endl;
		return;
	}
	
	
	printTrace(l, last[l][r]);
	printTrace(last[l][r], r);
	
	cout << "[" << l << "," << last[l][r] << "] [" << last[l][r] << "," << r << "]" << endl;
}
int main()
{
	cin >> n;
	
	for(int i = 1; i <= n; i ++ )
	{
		cin >> str[i];
	}
	
	memset(cost, 0x3f, sizeof cost);
	
	for(int len = 2; len <= n; len ++ )
	{
		for(int L = 1; L + len - 1 <= n; L ++ )
		{
			int R = L + len - 1;
			
			if(R - L + 1 < 2) continue;
			else if(R - L + 1 == 2)
			{
				cost[L][R] = 0;
				last[L][R] = L;
			}
```

```
                else if(R - L + 1 == 3)
                {
                        cost[L][R] = (str[L] ) * (str[L + 1]) * (str[R]);
                        cout << "l == " << L << ' ' << "r == " << R << ' ' << "value == " << cost[L][R]
                        << endl;
                        last[L][R] = L + 1;
                }
                else{
                        for(int k = L; k <= R; k ++ )
                        {
                                if(cost[L][k] + cost[k][R] + (str[L]) * (str[k]) * (str[R]) < cost[L][R])
                                {
                                        cost[L][R] = cost[L][k] + cost[k][R] + (str[L]) * (str[k]) * (str[R]);
                                        last[L][R] = k;
                                }
                                cout << "k == " << k << endl;
                                cout << "l == " << L << ' ' << "r == " << R << ' ' << "value == " << cost[L]
                                [R] << endl;
                        }
                }

        }
    }

    cout << cost[1][n] << endl;

    printTrace(1,n);
}
```

二分查找

2021年1月27日 14:37

https://blog.csdn.net/qq_40160605/article/details/80150252

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 100000 + 10;
const int INF = 2 * int(1e9) + 10;
#define LL long long
int cmd(int a, int b) {
	return a > b;
}
int main() {
	int num[6] = { 1,2,4,7,15,34 };
	sort(num, num + 6);                      //按从小到大排序
	int pos1 = lower_bound(num, num + 6, 7) - num;   //返回数组中第一个大于或等于被查数的值
	int pos2 = upper_bound(num, num + 6, 7) - num;   //返回数组中第一个大于被查数的值
	cout << pos1 << " " << num[pos1] << endl;
	cout << pos2 << " " << num[pos2] << endl;
	sort(num, num + 6, cmd);                 //按从大到小排序
	int pos3 = lower_bound(num, num + 6, 7, greater<int>()) - num;  //返回数组中第一个小于或等于被查数的值
	int pos4 = upper_bound(num, num + 6, 7, greater<int>()) - num;  //返回数组中第一个小于被查数的值
	cout << pos3 << " " << num[pos3] << endl;
	cout << pos4 << " " << num[pos4] << endl;
	return 0;
}
```

元组

2021年1月26日 12:57

https://blog.csdn.net/m0_37809890/article/details/89367406

```
#include<iostream>
#include<utility>
using namespace std;
int main()
{
        int a = 2;
        double b = 3.14;
        string s = "qwe";
        tuple<int, double, string> t = { a,b,s };
        cout << get<0>(t) << ' ' << get<1>(t) << ' ' << get<2>(t) << endl;

        a = 3, b = 2.5, s = "www";
        tie(a, b, s) = t;
        cout << a << ' ' << b<< ' ' << s << endl;

        a = 3, b = 2.5, s = "www";
        tie(a, ignore, s) = t;
        cout << a << ' ' << b << ' ' << s << endl;
}
```

结构化绑定声明

2021年1月26日　　14:08

https://blog.csdn.net/janeqi1987/article/details/100065133

优先队列

2021年1月19日 15:37

//greater是从小到大排列

//less是从大到小排列

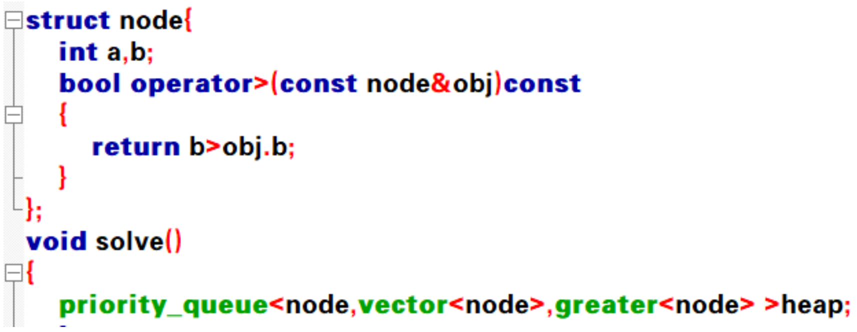
//例中的仿函数是从小到大排列

```
#include<iostream>
#include<queue>
using namespace std;
class func {//仿函数
public:
	bool operator()(int a, int b)
	{
		return a > b;
	}
};
int main()
{
	priority_queue<int,vector<int>,greater<int> > que;
	for (int i = 0; i < 5; i++)
		que.push(i);
	for (int i = 0; i < 5; i++)
	{
		cout << que.top() << ' ';
		que.pop();
	}
	cout << endl;

	priority_queue<int, vector<int>, less<int> > que2;
	for (int i = 0; i < 5; i++)
		que2.push(i);
	for (int i = 0; i < 5; i++)
	{
		cout << que2.top() << ' ';
		que2.pop();
	}
	cout << endl;



	priority_queue<int, vector<int>, func> que3;
	for (int i = 0; i < 5; i++)
		que3.push(i);

	for (int i = 0; i < 5; i++)
	{
		cout << que3.top() << ' ';
		que3.pop();
	}
	cout << endl;
```

}

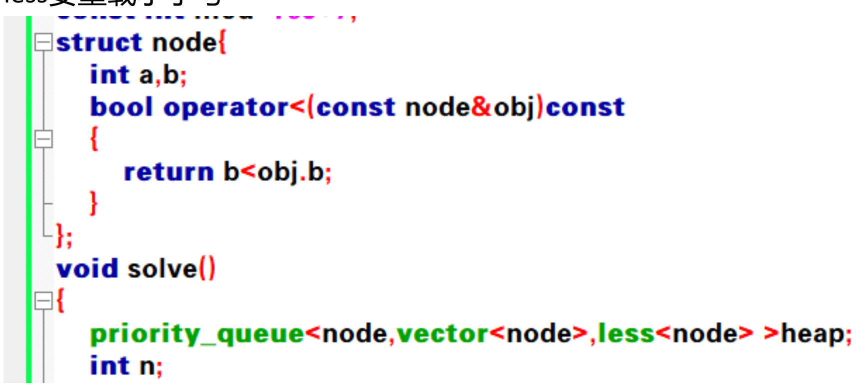
stl::set也可以用仿函数自定义比较函数

greater要重载大于号，注意const不能丢



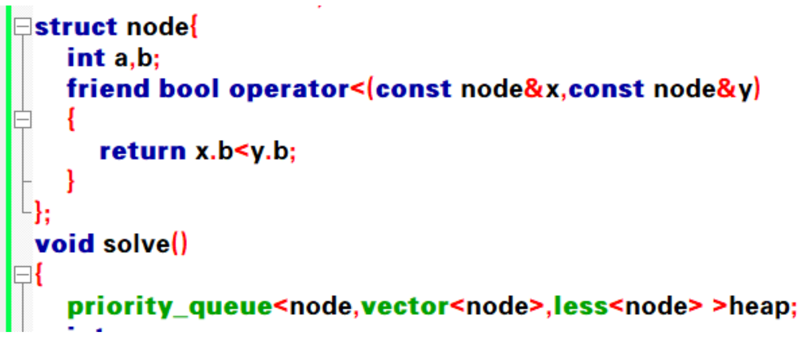
```cpp
struct node{
    int a,b;
    bool operator>(const node&obj)const
    {
        return b>obj.b;
    }
};
void solve()
{
    priority_queue<node,vector<node>,greater<node> >heap;
```

less要重载小于号



```cpp
struct node{
    int a,b;
    bool operator<(const node&obj)const
    {
        return b<obj.b;
    }
};
void solve()
{
    priority_queue<node,vector<node>,less<node> >heap;
    int n;
```

也可以这样重载成友元函数：



```cpp
struct node{
    int a,b;
    friend bool operator<(const node&x,const node&y)
    {
        return x.b<y.b;
    }
};
void solve()
{
    priority_queue<node,vector<node>,less<node> >heap;
```

vector

2022年2月28日 11:44

你可以用字符串来初始化vector：

```
vector<char>arr(s.begin(),s.end());
```

你可以用另一个vector来初始化vector：

```
vector<char>sub(arr.begin()+i+1,arr.end());
```

你可以在一个vector里插入另一个vector：

```
arr.insert(arr.end(),sub.begin(),sub.end());
```

迭代器的移动

2021年11月23日 8:20

prev(iter,2):返回一个在iter前两个单元的新迭代器

next(iter,2):返回一个在iter后两个单元的新迭代器

advance(iter,i):将iter向后或向前移动i个单元

如果上面三个函数的参数取负数则意味反向

distance(iter1,iter2):返回两个迭代器之间的距离

map

2021年1月19日 15:38

```
#include<iostream>
#include<map>
using namespace std;
int main()
{
	map<int, int> mp;
	int n;
	cin >> n;
	for (int i = 0; i < n; i++)
	{
		int h;
		cin >> h;
		mp[h]++;
	}
	for (map<int, int>::iterator i = mp.begin(); i != mp.end(); i++)
	{
		cout << i->first << ' ' << i->second << endl;
	}
	cout << mp.size()<<endl;
	for (int j = 0; j < mp.size(); j++)
		cout << mp[j] << endl;
}
```

accumulate

2021年1月27日 14:39

https://blog.csdn.net/qq_40803710/article/details/80273811

fill

2021年1月27日　　21:23

https://blog.csdn.net/ling_wang/article/details/79433479

cbrtl()

2021年2月18日　　12:08

开立方

copy

2021年9月30日 16:29

```
std::copy(start, end, std::back_inserter(container));
```

来自 <https://blog.csdn.net/a_ran/article/details/17385911>

tar

2021年2月22日　　19:02

压缩/解压命令
https://www.cnblogs.com/xioawu-blog/p/10999206.html
参数:
每条指令只能出现其中一个:

-c: 建立压缩档案

-x：解压

-t：查看内容

-r：向压缩归档文件末尾追加文件

-u：更新原压缩包中的文件

来自 <https://www.cnblogs.com/xioawu-blog/p/10999206.html>

每条指令可出现其中多个:

-z：有gzip属性的

-j：有bz2属性的

-Z：有compress属性的

-v：显示所有过程

-O：将文件解开到标准输出

参数-f是必须的

-f: 使用档案名字，切记，这个参数是最后一个参数，后面只能接档案名。

来自 <https://www.cnblogs.com/xioawu-blog/p/10999206.html>

例：tar -xf Python-3.8.1.tar.xz
作用：解压名为Python-3.8.1.tar.xz的包

./

2021年2月22日 19:06

意为：当前目录

以.或..开始的路径为相对路径，以/开头的路径为绝对路径

yum

2021年2月22日 19:08

在线安装命令
https://blog.csdn.net/whb15889740750/article/details/79055110

参数:

list 搜寻yum可以安装的所有的包名
search 搜索指定关键字的包名

-y 安装时自动回答yes

install 安装
update 更新，避免使用yum -y update指令，因为这将会更新系统的所有软件
remove 卸载包

gouplist 列出所有可用的软件组列表
groupinstall 软件组名 安装指定的软件组，组名可以由grouplist查出来
groupremove 软件组名 卸载指定的软件组

来自 <https://blog.csdn.net/whb15889740750/article/details/79055110>

make

2021年2月22日　　19:11

Makefile命令
http://www.80vps.com/new9058.htm
https://blog.csdn.net/electrocrazy/article/details/79348357

ls

2021年2月22日　　19:20

列出当前目录内容
详细说明:
https://www.cnblogs.com/zerotomax/p/7224927.html

pwd

2021年2月22日　　19:21

查看自己当前所在目录

https://www.cnblogs.com/zerotomax/p/7224927.html

cd

2021年2月22日 19:24

切换目录
https://www.cnblogs.com/zerotomax/p/7224927.html

mv

2021年2月22日　　19:27

文件改名/移动
https://blog.csdn.net/weixin_34413802/article/details/91884677

ln

2021年2月22日 19:31

创建连接命令
https://www.cnblogs.com/zerotomax/p/7232300.html

vi

2021年2月22日 19:34

编辑命令

https://www.cnblogs.com/shaosks/p/9167767.html

目录符号

2021年2月25日　　22:52

.	当前目录
..	上一层目录
-	前一个工作目录，上一次操作的目录 cd -　会切换都上次你操作的目录
~	当前【用户】所在的家目录
/	root的根目录

https://blog.csdn.net/weixin_33923148/article/details/94681449

find

2021年2月25日 23:07

查找命令

按文件名查找位置：从根目录查找：find / -name xxxx

从当前目录查找：find . -name xxxx

将文件名用'*　*'括起来就是模糊查找

https://blog.csdn.net/qq_38322527/article/details/102981561

whereis xxxx

2021年2月25日 23:15

查找目标文件位置

rm

2021年2月26日 20:32

删除指令

进程命令

2021年2月27日　　2:12

https://www.cnblogs.com/aipiaoborensheng/p/7676364.html

vim 指令

2021年8月11日 18:09

i 进入插入模式，esc退出

h左移，k上移，l右移，j下移

w移到下个单词的开头，b移到单词开头，e移到单词结尾

3w 按w三次

3 i go插入三个go

f w 找到下一个w的位置

3 f q 找到之后q出现第三次的位置

shift+5 跳到当前括号匹配括号的位置

0 跳到行首

$ 跳到行末

* 找到当前字符下一次出现的位置

# 找到当前字符上一次出现的位置

gg 跳到文件首

G 跳到文件末

2G 跳到第二行

/ text 搜索text出现的位置，n找到下一次出现位置，N找到上一次出现位置

o 插入新行

x 向左删除字符

r 覆盖当前字符

d 删除指令，如dw删除光标右边第一个单词并写到剪贴板，按p可以复制回去

d2e删除右边两个单词

. 重复上一次操作

v 进入visual模式，e选中当前单词，此时按h/j/k/l可以拖光标， d删除

:w保存

:q退出

:q!退出不保存

u撤销

x:删除光标所在字符

X:删除光标所在前字符

用 vim 写代码时，经常遇到这样的场景，复制多行，然后粘贴。 我现在这样做：
将光标移动到要复制的文本开始的地方，按v进入可视模式。
将光标移动到要复制的文本的结束的地方，按 y 复制。此时 vim 会自动将光标定位到选中文本的开始的地方，并退出可视模式。

我移动光标到文本结束的地方，按p粘贴。

来自 <https://ruby-china.org/topics/4266>

把光标放在要复制的行
esc
输入"y 行数 “，例如 “y10” （复制当前及紧跟的 10 行）
把光标挪到要粘贴的那一行
按 p
如果其他不变把 “y10” 变成 “10y” 就是复制第 10 行

来自 <https://ruby-china.org/topics/4266>

gg=G 对整个文档进行格式化

set nu 显示行号