板子

#### **1.gcd**

#### **long long gcd(long long a,long long b)**

#### **{**

#### **return b==0?a:gcd(b,a%b);**

#### **}**

#### **long long lcm(long long a,long long b)**

#### **{**

#### **return a/gcd(a,b)\*b;**

#### **}**

#### **2.扩展gcd )extend great common divisor**

ll exgcd(ll l,ll r,ll &x,ll &y){

if(r==0){x=1;y=0;return l;}

else

{

ll d=exgcd(r,l%r,y,x);

y-=l/r\*x;

return d;

}}

#### **3.求a关于m的乘法逆元**

ll mod\_inverse(ll a,ll m){

ll x,y;

if(exgcd(a,m,x,y)==1)//ax+my=1

return (x%m+m)%m;

return -1;//不存在}

#### **4.快速幂quick power**

ll qpow(ll a,ll b,ll m){

ll ans=1;

ll k=a;

while(b){

if(b&1)ans=ans\*k%m;

k=k\*k%m;

b>>=1;

}

return ans;}

#### **5.快速乘,直接乘会爆ll时需要它，也叫二分乘法。**

ll qmul(ll a,ll b,ll m){

ll ans=0;

ll k=a;

ll f=1;//f是用来存负号的

if(k<0){f=-1;k=-k;}

if(b<0){f\*=-1;b=-b;}

while(b){

if(b&1)

ans=(ans+k)%m;

k=(k+k)%m;

b>>=1;

}

return ans\*f;}

#### **7.筛素数，全局：int cnt,prime[N],p[N];**

void isprime(){

cnt = 0;

memset(prime,true,sizeof(prime));

for(int i=2; i<N; i++)

{

if(prime[i])

{

p[cnt++] = i;

for(int j=i+i; j<N; j+=i)

prime[j] = false;

}

}}

#### **8.快速计算逆元**

void inverse(){

inv[1] = 1;

for(int i=2;i<N;i++)

{

if(i >= M) break;

inv[i] = (M-M/i)\*inv[M%i]%M;

}}

#### **9.组合数取模**

ll fac[N]={1,1},inv[N]={1,1},f[N]={1,1};

ll C(ll a,ll b){

if(b>a)return 0;

return fac[a]\*inv[b]%M\*inv[a-b]%M;}void init(){//快速计算阶乘的逆元

for(int i=2;i<N;i++){

fac[i]=fac[i-1]\*i%M;

f[i]=(M-M/i)\*f[M%i]%M;

inv[i]=inv[i-1]\*f[i]%M;

}}

求C（n,m）%mod

#include <iostream>

#include <string.h>

#include <cmath>

#define ll long long

using namespace std;

const int maxn=100002;

ll n,m,p;

ll fac[maxn];

void getfac(ll p)//预处理阶层

{

fac[0]=1;

for(int i=1;i<=p;i++)

fac[i]=fac[i-1]\*i%p;

}

ll power(ll a,ll n,ll p)//快速幂运算

{

ll ans=1;

while(n)

{

if(n&1)

ans=ans\*a%p;

a=a\*a%p;

n/=2;

}

return ans;

}

ll lucas(ll n,ll m,ll p)

{

ll ans=1;

while(n&&m)

{

ll a=n%p;

ll b=m%p;

if(a<b) return 0;

ans=(ans\*fac[a]\*power(fac[b]\*fac[a-b]%p,p-2,p))%p;// fac[b]\*fac[a-b]后面别忘了%p，否则WA

n/=p;

m/=p;

}

return ans;

}

int main()

{

int t;cin>>t;

while(t--)

{

cin>>n>>m>>p;

getfac(p);

cout<<lucas(n+m,m,p)<<endl;

}

return 0;

}

#### Fibonacci

//#include<bits/stdc++.h>

#include <iostream>

#include <cstring>

#include <cstdio>

using namespace std;

typedef long long ll;

const int MOD=10000;

struct mat{

ll a[2][2];

};

mat mat\_mul(mat x,mat y){

mat res;

memset(res.a,0,sizeof(res.a));

for(int i=0;i<2;i++){

for(int j=0;j<2;j++){

for(int k=0;k<2;k++){

res.a[i][j]=(res.a[i][j]+x.a[i][k]\*y.a[k][j])%MOD;

}

}

}

return res;

}

void mat\_pow(int n){

mat c,res;

c.a[0][0]=c.a[0][1]=c.a[1][0]=1;

c.a[1][1]=0;

memset(res.a,0,sizeof(res.a));

for(int i=0;i<2;i++){

res.a[i][i]=1;

}

while(n){

if(n&1){

res=mat\_mul(res,c);

n--;

}

n=n/2;

c=mat\_mul(c,c);

}

printf("%I64d\n",res.a[0][1]);

}

int main(){

int n;

while(~scanf("%d",&n)){

if(n==-1)

break;

else

mat\_pow(n);

}

return 0;

}

Lele now is thinking about a simple function f(x).   
  
If x < 10 f(x) = x.   
If x >= 10 f(x) = a0 \* f(x-1) + a1 \* f(x-2) + a2 \* f(x-3) + …… + a9 \* f(x-10);   
And ai(0<=i<=9) can only be 0 or 1 .   
  
Now, I will give a0 ~ a9 and two positive integers k and m ,and could you help Lele to caculate f(k)%m.

**Input**

The problem contains mutiple test cases.Please process to the end of file.   
In each case, there will be two lines.   
In the first line , there are two positive integers k and m. ( k<2\*10^9 , m < 10^5 )  
In the second line , there are ten integers represent a0 ~ a9.

**Output**

For each case, output f(k) % m in one line.

**Sample Input**

10 9999

1 1 1 1 1 1 1 1 1 1

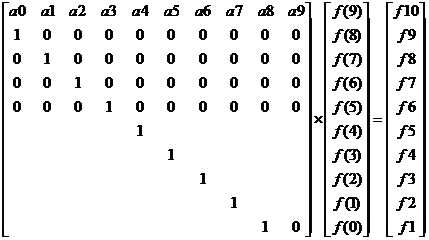
20 500

1 0 1 0 1 0 1 0 1 0

**Sample Output**

45

104



#include<iostream>

#include<string.h>

#include<algorithm>

#include<cstdio>

#define ll long long

using namespace std;

struct mat{

ll a[10][10];

};

ll k,m;

int aa[10];

int f[10];

void init(){

for(int i=0;i<10;i++){

f[i]=i;

}

}

mat mat\_mul(mat x,mat y){

mat res;

memset(res.a,0,sizeof(res.a));

for(int i=0;i<10;i++){

for(int j=0;j<10;j++){

for(int k=0;k<10;k++){

res.a[i][j]=(res.a[i][j]+x.a[i][k]\*y.a[k][j])%m;

res.a[i][j]=res.a[i][j]%m;

}

}

}

return res;

}

void mat\_pow(ll n){

mat c,res;

memset(c.a,0,sizeof(c.a));

for(int i=0;i<10;i++){

c.a[0][i]=aa[i];

}

for(int i=0;i<9;i++){

c.a[1+i][0+i]=1;

}

memset(res.a,0,sizeof(res.a));

for(int i=0;i<10;i++) {

res.a[i][i]=1;

}

while(n){

if(n&1) res=mat\_mul(res,c);

c=mat\_mul(c,c);

n=n>>1;

}

ll ans=0;

for(int i=0;i<10;i++){

ans=ans+(res.a[0][i]\*f[9-i])%m;

}

printf("%I64d\n",ans%m);

}

int main() {

while(~scanf("%lld %lld",&k,&m)){

init();

for(int i=0;i<10;i++){

scanf("%d",&aa[i]);

}

if(k<10){

printf("%lld\n",f[k]%m);

}

else{

mat\_pow(k-9);

}

}

return 0;

}

10.青蛙的约会

#include <iostream>

#define dnt long long

using namespace std;

dnt x, y;

dnt a, b, m, n, L;

dnt Exgcd( dnt a, dnt b, dnt &x, dnt &y ) {

if ( b == 0 ) {

x = 1;

y = 0;

return a;

}

dnt d = Exgcd(b, a%b, x, y), temp = x;

x = y;

y = temp-a/b\*y;

return d;

}

dnt solv( dnt a, dnt b, dnt c ) {

dnt d = Exgcd(a, b, x, y);

if ( c % d ) return -1;

x = x \* c / d;

y = y \* c / d;

x = (x % b + b) % b;

return x;

}

int main() {

cin >> a >> b >> m >> n >> L;

if ( solv(n-m, L, a-b) != -1 )

cout << solv(n-m, L, a-b) << endl;

else cout << "Impossible" << endl;

return 0;

}

/\*我们转换一下方程：

x - y = (n - m) \* t + L \* k

所以它是形如ax+by=c这种形式，我们直接用欧几里得求出一组解输出最小正解就好了。\*/

1. 深度优先遍历

递归实现

（1）访问顶点v；visited[v]=1；//算法执行前visited[n]=0

（2）w=顶点v的第一个邻接点；

（3）while（w存在）

           if（w未被访问）

                   从顶点w出发递归执行该算法；   
           w=顶点v的下一个邻接点；

1. **广度优先遍历**

（1）初始化队列Q；visited[n]=0；

（2）访问顶点v；visited[v]=1；顶点v入队列Q；

（3） while（队列Q非空）

              v=队列Q的对头元素出队；

              w=顶点v的第一个邻接点；

             while（w存在）

                     如果w未访问，则访问顶点w；

                     visited[w]=1；

                     顶点w入队列Q；

                     w=顶点v的下一个邻接点。

# HDU2717:Catch That Cow(BFS)

2013年06月03日 21:54:16

阅读数：8043

Problem Description

Farmer John has been informed of the location of a fugitive cow and wants to catch her immediately. He starts at a point N (0 ≤ N ≤ 100,000) on a number line and the cow is at a point K (0 ≤ K ≤ 100,000) on the same number line. Farmer John has two modes of transportation: walking and teleporting.  
  
\* Walking: FJ can move from any point X to the points X - 1 or X + 1 in a single minute  
\* Teleporting: FJ can move from any point X to the point 2 × X in a single minute.  
  
If the cow, unaware of its pursuit, does not move at all, how long does it take for Farmer John to retrieve it?

Input

Line 1: Two space-separated integers: N and K

Output

Line 1: The least amount of time, in minutes, it takes for Farmer John to catch the fugitive cow.

Sample Input

5 17

Sample Output

4

***Hint***

The fastest way for Farmer John to reach the fugitive cow is to move along the following path: 5-10-9-18-17, which takes 4 minutes.

#include <stdio.h>

#include <string.h>

#include <queue>

using namespace std;

const int N = 1000000;

int map[N+10];

int n,k;

struct node

{

int x,step;

};

int check(int x)

{

if(x<0 || x>=N || map[x])

return 0;

return 1;

}

int bfs(int x)

{

int i;

queue<node> Q;

node a,next;

a.x = x;

a.step = 0;

map[x] = 1;

Q.push(a);

while(!Q.empty())

{

a = Q.front();

Q.pop();

if(a.x == k)

return a.step;

next = a;

//每次都将三种状况加入队列之中

next.x = a.x+1;

if(check(next.x))

{

next.step = a.step+1;

map[next.x] = 1;

Q.push(next);

}

next.x = a.x-1;

if(check(next.x))

{

next.step = a.step+1;

map[next.x] = 1;

Q.push(next);

}

next.x = a.x\*2;

if(check(next.x))

{

next.step = a.step+1;

map[next.x] = 1;

Q.push(next);

}

}

return -1;

}

int main()

{

int ans;

while(~scanf("%d%d",&n,&k))

{

memset(map,0,sizeof(map));

ans = bfs(n);

printf("%d\n",ans);

}

return 0;

}

大数相加

#include<stdio.h>

#include<string.h>

int main()

{

int n,i,len1,len2,j,k,pi,t,flag;

char str1[1010],str2[1010];

scanf("%d",&n);

for(i=1;i<=n;i++)

{

int a[1200]={0};

flag=0;

printf("Case %d:\n",i);

scanf("%s%s",str1,str2);//以字符串形式读入

len1=strlen(str1);

len2=strlen(str2);

printf("%s + %s = ",str1,str2);

j=len1-1;

k=len2-1;

pi=0;

while(j>=0&&k>=0)//开始相加

{

if(a[pi]+(str1[j]-'0')+(str2[k]-'0')>=10)//相加后大于10的

{

a[pi]=a[pi]+(str1[j]-'0')+(str2[k]-'0')-10;

a[pi+1]++;

}

else

a[pi]=a[pi]+(str1[j]-'0')+(str2[k]-'0');

pi++;

k--;

j--;

}

if(j>=0)

{

for(t=j;t>=0;t--)

{

a[pi]=a[pi]+(str1[t]-'0');

pi++;

}

}

else if(k>=0)

{

for(t=k;t>=0;t--)

{

a[pi]=a[pi]+str2[t]-'0';

pi++;

}

}

else if(a[pi]!=0)//对于位数相同2个数加后最高位大于10的

pi++;

for(t=pi-1;t>=0;t--)

{

if(a[t]==0&&flag==0)//处理一次啊前导0，估计属于无用的一步

continue;

else

{

flag=1;

printf("%d",a[t]);

}

}

printf("\n");

if(i!=n)//对于2组之间加空行的情况

printf("\n");

}

return 0;

}

最短路

#include<bits/stdc++.h>

#define INF 0x3f3f3f3f

#define MST(a,b) memset(a,b,sizeof(a))

using namespace std;

const int maxn=1e3+7;

int n,m;

int dis[maxn],maz[maxn][maxn];

bool vis[maxn];

void init()

{

MST(dis,INF),MST(maz,INF),MST(vis,false);

}

int dijkstra(int st,int en)

{

for(int i=1;i<=n;i++)

dis[i]=maz[st][i];

dis[st]=0,vis[st]=true;

int t,pos;

for(int i=1;i<=n;i++){

t=INF;

for(int j=1;j<=n;j++)

if(!vis[j]&&dis[j]<t){

pos=j;

t=dis[j];

}

vis[pos]=true;

for(int j=1;j<=n;j++)

if(!vis[j]&&dis[j]>dis[pos]+maz[pos][j])

dis[j]=min(dis[j],dis[pos]+maz[pos][j]);

}

return dis[en];

//printf("%d\n",dis[en]);

}

int main()

{

//printf("%d\n",INF);

int a,b,c;

while(~scanf("%d%d",&n,&m)&&n+m){

init();

for(int i=0;i<m;i++){

scanf("%d%d%d",&a,&b,&c);

maz[a][b]=maz[b][a]=min(maz[a][b],c);

}

printf("%d\n",dijkstra(1,n));

}

return 0;

}

# pat L2-001. 紧急救援 dijkstra变形+记录路径

感觉这是特别好的一道题,这个最短路的变形有两个权值一个是点的权值一个是边之间的权值.而且要求我们记录很多东西,

注意：题中要求的输出的路径是保证在所能召集最多的救援队的前提下的.

对于记录路径：我们开一个path[]数组 记录当前点 i 的上一个点是什么;

数组 pathnum[]记录最短路径的条数

tol[]代表从起点到终点所能召集的最多救援队的数量

w[]数组是每个点的救援队的数量

剩下的就是dijkstra的模板了,但是注意如果当前路径能通过这点松弛的时候我们要更新其余能用它松弛点的

path,pathnum,tol,dis 否则的话 如果距离相等的话我们还要判断他的第二个权值就是tol[]（点的权值）是否需要更新,

也就是说要以路径最短为前提去寻找能召集的最多救援队的数量.

#include<stdio.h>

#include<string.h>

#include<math.h>

#include<algorithm>

#define inf 0x3f3f3f3f

using namespace std;

const int N=555;

int dis[N],path[N],w[N],pathnum[N],tol[N],book[N];

int mp[N][N];

int n,m,s,d;

void print(int x)//递归输出路径

{

if(path[x]==-1)

{

printf("%d",x);

return ;

}

print(path[x]);

printf(" %d",x);

return ;

}

void dijkstra()

{

memset(book,0,sizeof(book));

memset(tol,0,sizeof(tol));

int i,j;

for(i=0;i<n;i++)

dis[i]=inf;

dis[s]=0;

path[s]=-1;

tol[s]=w[s];//对起始点的初始化

pathnum[s]=1;//最短路径数为1

for(i=0;i<n;i++)

{

int u,minn=inf;

for(j=0;j<n;j++)

{

if(!book[j]&&dis[j]<minn)

{

u=j;

minn=dis[j];

}

}

book[u]=1;

for(j=0;j<n;j++)

{

if(dis[j]>dis[u]+mp[u][j])//松弛时各数组的更新

{

dis[j]=dis[u]+mp[u][j];

path[j]=u;

tol[j]=tol[u]+w[j];

pathnum[j]=pathnum[u];

}

else if(dis[j]==dis[u]+mp[u][j])//相等时的更新

{

pathnum[j]+=pathnum[u];//应该是二者最短路径的和

if(tol[j]<tol[u]+w[j])//点权值的更新

{

tol[j]=tol[u]+w[j];

path[j]=u;

}

}

}

}

}

int main()

{

int x,y,z;

scanf("%d%d%d%d",&n,&m,&s,&d);

for(int i=0;i<n;i++)

scanf("%d",&w[i]);

memset(mp,inf,sizeof(mp));

for(int i=0;i<m;i++)

{

scanf("%d%d%d",&x,&y,&z);

mp[x][y]=z;

mp[y][x]=mp[x][y];

}

dijkstra();

printf("%d %d\n",pathnum[d],tol[d]);

print(d);

return 0;

}

## SPFA 邻接表版

#include<iostream>

#include<queue>

#include<list>

#define inf 0x7fffffff/2

using namespace std;

queue<int> q;//队列优化

int m,n,s,t,cnt;

int d[1000000];

int flag[1000000];

int ss[1000000];

int h[1000000];

struct eg

{

int to,next,v;

}a[6000000];//邻接表

void Union(int x,int y,int v)

{

cnt++;

a[cnt]=(eg)

{

y,h[x],v

};

h[x]=cnt;

}//连边

int spfa(int v0)

{

for(int i=1;i<=n;i++)d[i]=inf;//初始化

d[v0]=0;

q.push(v0);

while(!q.empty())

{

int now=q.front();

q.pop();

flag[now]++;//now这个点被优化的次数增加

if(flag[now]==n)return 1;

/\*

存在负环无解，因为一共n个点，所以有n-1条边，若是正环，

走的越多，权值越大，这样最多也就可以是使now被优化n-1次，

然而存在负环时，由于这是环，所以走的越久，反而权值越小，

这样就会无限绕，now的优化次数就会大于n，如果这里不判断，

就会进入死循环，因为答案无限小

\*/

for(int i=h[now];i;i=a[i].next)//依次遍历与这个点相连的每一条边

{

int to=a[i].to;

if(d[to]>d[now]+a[i].v)//松弛操作

{

d[to]=d[now]+a[i].v;

if(ss[to]==0)//判断是否在队列中

{

q.push(to);

ss[to]=1;

}

}

}

}

return 0;

}

int main()

{

cin>>n>>m;

int tis,tit,tst;

for(int i=1;i<=m;i++)

{

cin>>tis>>tit>>tst;

Union(tis,tit,tst);

Union(tit,tis,tst);//建图

}

if(spfa(1)==0&&d[n]!=inf)cout<<d[n];

else cout<<"No Solution";

}

单源最短路条数：

void spfa(int st) {

memset(dis,INF,sizeof dis);

queue<int> q;q.push(st);

dis[st]=0;vis[st]=1;

while(!q.empty()) {

int cur = q.front();q.pop();vis[cur]=0;

int up = vv[cur].size();

for(int i = 0; i<up; i++) {

Edge now = vv[cur][i];

int to = now.to;

if(dis[to] > dis[cur] + now.w) {

dis[to] = dis[cur]+now.w;

dp[to] = dp[cur];

if(vis[to] == 0) vis[to]=1,q.push(to);

}

else if(dis[to] == dis[cur] + now.w) {

dp[to] += dp[cur];

}

}

}

}

并查集，并查集是一种树形结构，又叫“不相交集合”，保持了一组不相交的动态集合，每个集合通过一个代表来识别，代表即集合中的某个成员，通常选择根做这个代表。   
也就是说，并查集是用来处理不相交集合类型问题，如问不相交集合有几个。给定节点，找到该节点所在集合元素个数，当然这只是水题。并查集会与其他算法结合着考，如LCA中的tarjian算法。后续博客会整理。   
并查集，顾名思义，主要分三部分。   
一：合并：给出两点关系，如果属于同一集合，进行merge   
二：查：在合并时，需要先写出查，即找到该点的祖先点   
三：集:merge后，将新加入的点的祖先点更新   
然后，点集就因为共同的祖先点被分为不同的集合啦   
结合例题更容易理解   
裸题模板：   
hdu1232畅通工程   
畅通工程   
Time Limit: 4000/2000 MS (Java/Others) Memory Limit: 65536/32768 K (Java/Others)   
Total Submission(s): 52315 Accepted Submission(s): 27902

Problem Description   
某省调查城镇交通状况，得到现有城镇道路统计表，表中列出了每条道路直接连通的城镇。省政府“畅通工程”的目标是使全省任何两个城镇间都可以实现交通（但不一定有直接的道路相连，只要互相间接通过道路可达即可）。问最少还需要建设多少条道路？

Input   
测试输入包含若干测试用例。每个测试用例的第1行给出两个正整数，分别是城镇数目N ( < 1000 )和道路数目M；随后的M行对应M条道路，每行给出一对正整数，分别是该条道路直接连通的两个城镇的编号。为简单起见，城镇从1到N编号。   
注意:两个城市之间可以有多条道路相通,也就是说   
3 3   
1 2   
1 2   
2 1   
这种输入也是合法的   
当N为0时，输入结束，该用例不被处理。

Output   
对每个测试用例，在1行里输出最少还需要建设的道路数目。

Sample Input

4 2   
1 3   
4 3   
3 3   
1 2   
1 3   
2 3   
5 2   
1 2   
3 5   
999 0   
0

Sample Output

1   
0   
2   
998

就是说将所有独立的集合连接起来还需要几条路，那只要找到独立集合个数-1就可以啦

#include<stdio.h>int father[1005];int Find(int x)

{

while(x!=father[x])

x=father[x];

return x;

}

void Combine(int a,int b)

{

int fa=Find(a);

int fb=Find(b);

if(fa!=fb)

{

father[fa]=fb;

}

}int main()

{

int n,m;

int i;

int a,b;

while(~scanf("%d",&n))

{

if(n==0)

break;

scanf("%d",&m);

int sum=0;

for(i=1;i<=n;i++)

father[i]=i;

for(i=0;i<m;i++)

{

scanf("%d%d",&a,&b);

Combine(a,b);

}

for(i=1;i<=n;i++)

{

if(father[i]==i)

sum++;

}

printf("%d\n",sum-1);

}

return 0;

}

用C++实现二叉树的“先根遍历”存储。

用C++实现二叉树的“先根遍历”、“中根遍历”、“后根遍历”分别输出二叉树中结点的数据。

1 #include <iostream>

2 using namespace std ;

3

4 struct BiNode

5 {

6 char data ;

7 BiNode \*lchild , \*rchild ;

8 } ;

9 BiNode \*BiTree ;

10

11 int NodeID ;

12

13 BiNode \*CreateBiTree (char \*c , int n)

14 {

15 BiNode \*T ;

16 NodeID ++ ;

17 if (NodeID > n)

18 {

19 return (NULL) ;

20 }

21 if (c[NodeID] == '0')

22 {

23 return (NULL) ;

24 }

25 T = new BiNode ;

26 T -> data = c[NodeID] ;

27 T -> lchild = CreateBiTree (c , n) ;

28 T -> rchild = CreateBiTree (c , n) ;

29 return (T) ;

30 }

31

32 void PreOrderTraverse (BiNode \*T)

33 {

34 if (T)

35 {

36 cout << T -> data << " ";

37 PreOrderTraverse (T -> lchild) ;

38 PreOrderTraverse (T -> rchild) ;

39 }

40 }

41

42 void InOrderTraverse (BiNode \*T)

43 {

44 if (T)

45 {

46 InOrderTraverse (T -> lchild) ;

47 cout << T -> data << " ";

48 InOrderTraverse (T -> rchild) ;

49 }

50 }

51

52 void PostOrderTraverse (BiNode \*T)

53 {

54 if (T)

55 {

56 PostOrderTraverse (T -> lchild) ;

57 PostOrderTraverse (T -> rchild) ;

58 cout << T -> data << " ";

59 }

60 }

61

62 int main ()

63 {

64 int i , SampleNum ;

65 char c[100] ;

66 cin >> SampleNum ;

67 for (i = 1 ; i <= SampleNum ; i ++)

68 {

69 cin >> c[i] ;

70 }

71 NodeID = 0 ;

72 BiTree = CreateBiTree (c , SampleNum) ;

73 PreOrderTraverse (BiTree) ;

74 cout << endl ;

75 InOrderTraverse (BiTree) ;

76 cout << endl ;

77 PostOrderTraverse (BiTree) ;

78 return 0 ;

79 }

# **#103. 子串查找**

**内存限制：256 MiB时间限制：500 ms标准输入输出**

**题目类型：传统评测方式：文本比较**

**上传者： 匿名**

[提交](https://loj.ac/problem/103" \l "submit_code" \t "https://blog.csdn.net/zdc_8023/article/details/_blank)[提交记录](https://loj.ac/submissions?problem_id=103" \t "https://blog.csdn.net/zdc_8023/article/details/_blank)[统计](https://loj.ac/problem/103/statistics/fastest" \t "https://blog.csdn.net/zdc_8023/article/details/_blank)[讨论](https://loj.ac/discussion/problem/103" \t "https://blog.csdn.net/zdc_8023/article/details/_blank)[测试数据](https://loj.ac/problem/103/testdata" \t "https://blog.csdn.net/zdc_8023/article/details/_blank)

#### **题目描述**

这是一道模板题。

给定一个字符串 A A*A* 和一个字符串 B B*B*，求 B B*B* 在 A A*A* 中的出现次数。

A A*A* 中不同位置出现的 B B*B* 可重叠。

#### **输入格式**

输入共两行，分别是字符串 A A*A* 和字符串 B B*B*。

#### **输出格式**

输出一个整数，表示 B B*B* 在 A A*A* 中的出现次数。

#### **样例**

#### **样例输入**

zyzyzyz

zyz

#### **样例输出**

3

1≤A,B 1 \leq A, B1≤*A*,*B* 的长度 ≤106 \leq 10 ^ 6≤10​6​​，A A*A*、

B B

*B* 仅包含大小写字母。

KMP模板题

1. #include<cstdio>
2. #include<iostream>
3. #include<algorithm>
4. #include<cstring>
5. #define maxn 1000005
6. using namespace std;
7. char text[maxn], patten[maxn];
8. int net[maxn];
9. int main() {
10. scanf("%s%s",text,patten);
11. int la = strlen(text);
12. int lb = strlen(patten);
13. memset(net, 0, sizeof(net));
14. for (int i = 1; i < lb; ++i) {
15. int j = i;
16. while (j > 0) {
17. j = net[j];
18. if (patten[j] == patten[i]) {
19. net[i + 1] = j + 1;
20. break;
21. }
22. }
23. }
24. int cnt = 0;
25. for (int i = 0, j = 0; i < la; ++i) {
26. if (j < lb&&patten[j] == text[i]) {
27. j++;
28. }
29. else {
30. while (j > 0) {
31. j = net[j];
32. if (text[i] == patten[j]) {
33. j++;
34. break;
35. }
36. }
37. }
38. if (j == lb) cnt++;
39. }
40. printf("%d\n", cnt);
41. return 0;
42. }

# loj 108 多项式乘法

2018年04月28日 20:52:45

阅读数：5

[http://www.elijahqi.win/archives/3253](http://www.elijahqi.win/archives/3253" \t "https://blog.csdn.net/elijahqi/article/details/_blank)   
题目描述

这是一道模板题。

输入两个多项式，输出这两个多项式的乘积。

输入格式

第一行两个整数 n n n 和 m m m，分别表示两个多项式的次数。

第二行 n+1 n + 1 n+1 个整数，分别表示第一个多项式的 0 0 0 到 n n n 次项前的系数。

第三行 m+1 m + 1 m+1 个整数，分别表示第二个多项式的 0 0 0 到 m m m 次项前的系数。

输出格式

一行 n+m+1 n + m + 1 n+m+1 个整数，分别表示乘起来后的多项式的 0 0 0 到 n+m n + m n+m 次项前的系数。

样例

样例输入

1 2   
1 2   
1 2 1   
样例输出

1 4 5 2   
模板

ntt

#include<cstdio>#include<cctype>#include<algorithm>#define ll long long

using namespace std;

inline char gc(){

static char now[1<<16],\*S,\*T;

if (T==S){T=(S=now)+fread(now,1,1<<16,stdin);if(T==S) return EOF;}

return \*S++;

}

inline int read(){

int x=0,f=1;char ch=gc();

while(!isdigit(ch)) {if(ch=='-') f=-1;ch=gc();}

while(isdigit(ch)) x=x\*10+ch-'0',ch=gc();

return x\*f;

}

const int N=1e5+10;

const int g=3;

const int mod=998244353;

inline int ksm(ll b,int t){static ll tmp;

for (tmp=1;t;b=b\*b%mod,t>>=1) if (t&1) tmp=tmp\*b%mod;return tmp;

}int a[N<<2],b[N<<2],R[N<<2],n,m;ll invn;

inline int inc(int x,int v){return x+v>=mod?x+v-mod:x+v;}

inline int dec(int x,int v){return x-v<0?x-v+mod:x-v;}

inline void ntt(int \*x,int f){

for (int i=0;i<n;++i) if(i<R[i]) swap(x[i],x[R[i]]);

for (int i=1;i<n;i<<=1){

ll wn=ksm(g,f==1?(mod-1)/(i<<1):mod-1-(mod-1)/(i<<1));

for (int j=0;j<n;j+=i<<1){

ll w=1,t1,t2;

for (int k=0;k<i;++k,w=w\*wn%mod)

t1=x[j+k],t2=w\*x[j+k+i]%mod,x[j+k]=inc(t1,t2),x[j+k+i]=dec(t1,t2);

}

}

if (f==-1) for (int i=0;i<n;++i) x[i]=x[i]\*invn%mod;

}int main(){

freopen("loj108.in","r",stdin);

n=read();m=read();

for (int i=0;i<=n;++i) a[i]=read();

for (int i=0;i<=m;++i) b[i]=read();

m+=n;int t=0;for (n=1;n<=m;n<<=1,++t);

for (int i=0;i<n;++i) R[i]=R[i>>1]>>1|(i&1)<<t-1;

ntt(a,1);ntt(b,1);invn=ksm(n,mod-2);

for (int i=0;i<n;++i) a[i]=(ll)a[i]\*b[i]%mod;

ntt(a,-1);

for (int i=0;i<=m;++i) printf("%d ",a[i]);

return 0;

}

[LOJ#114]k 大异或和

**试题描述**

这是一道模板题。

给由 n 个数组成的一个可重集 S，每次给定一个数 k，求一个集合 T?S，使得集合 T 在 S 的所有非空子集的不同的异或和中，其异或和 T1 xor T2 xor … xor T|T| 是第 k 小的。

**输入**

第一行一个数 n。  
第二行 n 个数，表示集合 S。  
第三行一个数 m，表示询问次数。  
第四行 m 个数，表示每一次询问的 k。

**输出**

输出 m 行，对应每一次询问的答案，第 k 小的异或和。如果集合 S 的所有非空子集中，不同的异或和数量不足 k，输出 -1。

**输入示例**

31 2 351 2 3 4 5

**输出示例**

0123

-1

**数据规模及约定**

1≤n,m≤10?5??,0≤S?i??≤2?50??

题解

对线性基进行高斯消元，离散后对 k 进行二进制拆分。

#include <iostream>

#include <cstdio>

#include <cstdlib>

#include <cstring>

#include <cctype>

#include <algorithm>

using namespace std;

#define LL long long

const int BufferSize = 1 << 16;

char buffer[BufferSize], \*Head, \*Tail;

inline char Getchar() {

if(Head == Tail) {

int l = fread(buffer, 1, BufferSize, stdin);

Tail = (Head = buffer) + l;

}

return \*Head++;

}

LL read() {

LL x = 0, f = 1; char c = Getchar();

while(!isdigit(c)){ if(c == ‘-‘) f = -1; c = Getchar(); }

while(isdigit(c)){ x = x \* 10 + c - ‘0‘; c = Getchar(); }

return x \* f;

}

#define maxn 55

int n, cnt;

LL bit[maxn], cb[maxn];

bool has0;

int main() {

n = read();

for(int i = 1; i <= n; i++) {

LL a = read();

for(int j = maxn - 1; j; j--) if(a >> j-1 & 1) {

if(!bit[j]){ bit[j] = a; break; }

a ^= bit[j];

}

if(!a) has0 = 1;

}

for(int i = maxn - 1; i; i--)

for(int j = i - 1; j; j--) if(bit[i] >> j-1 & 1)

bit[i] ^= bit[j];

for(int i = 1; i < maxn; i++) if(bit[i]) cb[cnt++] = bit[i];

int q = read();

while(q--) {

LL k = read() - has0, ans = 0;

if(k > (1ll << cnt) - 1){ puts("-1"); continue; }

for(int i = cnt - 1; i >= 0; i--) if(k >> i & 1) ans ^= cb[i];

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

}

return 0;

}

# **LOJ #109. 并查集**

时间：2017-08-14 09:21:42      阅读：115      评论：0      收藏：0      [点我收藏+]

标签：[pen](http://www.bubuko.com/so/1/pen" \o "pen)   [bsp](http://www.bubuko.com/so/1/bsp" \o "bsp)   [content](http://www.bubuko.com/so/1/content" \o "content)   [\_id](http://www.bubuko.com/so/1/_id" \o "_id)   [ans](http://www.bubuko.com/so/1/ans" \o "ans)   [无向图](http://www.bubuko.com/so/1/%e6%97%a0%e5%90%91%e5%9b%be" \o "无向图)   [一行](http://www.bubuko.com/so/1/%e4%b8%80%e8%a1%8c" \o "一行)   [类型](http://www.bubuko.com/so/1/%e7%b1%bb%e5%9e%8b" \o "类型)   [mis](http://www.bubuko.com/so/1/mis" \o "mis)

内存限制：256 MiB时间限制：2000 ms标准输入输出

题目类型：传统评测方式：文本比较

上传者： 匿名

[提交](https://loj.ac/problem/109" \l "submit_code)[提交记录](https://loj.ac/submissions?problem_id=109)[统计](https://loj.ac/problem/109/statistics/fastest)[讨论](https://loj.ac/problem/109/discussion)

1

[测试数据](https://loj.ac/problem/109/testdata)

#### 题目描述

这是一道模板题。

维护一个 nnn 点的无向图，支持：

* 加入一条连接 uuu 和 vvv 的无向边
* 查询 uuu 和 vvv 的连通性

由于本题数据较大，因此输出的时候采用特殊的输出方式：用 000 或 111 代表每个询问的答案，将每个询问的答案一次从左到右排列，把得到的串视为一个二进制数，输出这个二进制数 mod 998244353\text{mod} ~ 998244353mod 998244353 的值。

#### 输入格式

第一行包含两个整数 n,mn,mn,m，表示点的个数和操作的数目。

接下来 mmm 行每行包括三个整数 op,u,v\text{op},u,vop,u,v。

* 如果 op=0\text{op} = 0op=0，则表示加入一条连接 uuu 和 vvv 的无向边；
* 如果 op=1\text{op} = 1op=1，则表示查询 uuu 和 vvv 的连通性。

#### 输出格式

一行包括一个整数表示答案。

#### 样例

#### 样例输入

3 6

1 1 0

0 0 1

1 0 1

1 1 2

0 2 1

1 2 1

#### 样例输出

5

#### 样例解释

答案串为 101101101。

#### 数据范围与提示

n≤4000000,m≤8000000n\le 4000000,m\le 8000000n≤4000000,m≤8000000

#### [显示分类标签](https://loj.ac/problem/109)

感觉这几天见鬼了。。

昨天写的旋转卡壳比暴力慢，

今天写的启发式合并比暴力合并慢，，

#include<iostream>

#include<cstdio>

#include<cstring>

#include<cmath>

using namespace std;

const int MAXN=8000001;

const int mod=998244353;

inline void read(int &n)

{

char c=‘+‘;bool flag=0;n=0;

while(c<‘0‘||c>‘9‘) c==‘-‘?flag=1,c=getchar():c=getchar();

while(c>=‘0‘&&c<=‘9‘) n=n\*10+c-48,c=getchar();

}

int fa[MAXN];

int size[MAXN];

int n,m;

string p;

int find(int x)

{return fa[x]==x?fa[x]:fa[x]=find(fa[x]);}

int query(int x,int y)

{return find(x)==find(y);}

void unionn(int x,int y)

{

int fx=find(x);int fy=find(y);

if(fx!=fy)

{

if(size[fx]>size[fy]) swap(fx,fy);

fa[fx]=fy; size[fy]+=size[fx];

//fa[fx]=fy;

}

}

int ans=0;

int main()

{

//freopen("a.in","r",stdin);

//freopen("a.out","w",stdout);

read(n);read(m);

for(int i=1;i<=n;i++) fa[i]=i;

for(int i=1;i<=m;i++)

{

int how;read(how);

if(how)// 询问

{

int x,y;read(x);read(y);

ans=(ans\*2+query(x,y))%mod;

}

else//连边

{

int x,y;read(x);read(y);

unionn(x,y);

}

}

printf("%d",ans);

return 0;

}

****在c++中，vector是一个十分有用的容器。****

****作用：它能够像容器一样存放各种类型的对象，简单地说，vector是一个能够存放任意类型的动态数组，能够增加和压缩数据。****

****vector在C++标准模板库中的部分内容，它是一个多功能的，能够操作多种数据结构和算法的**[模板类](http://baike.baidu.com/view/1923683.htm" \t "https://blog.csdn.net/duan19920101/article/details/50617190/_blank)**和函数库。****

****特别注意：****

****使用vector需要注意以下几点：****

****1、如果你要表示的向量长度较长（需要为向量内部保存很多数），容易导致内存泄漏，而且效率会很低；****

****2、Vector作为函数的参数或者返回值时，需要注意它的写法：****

****double Distance(vector<int>&a, vector<int>&b) 其中的“&”绝对不能少！！！****

****实例：vector<int>test;****

****//建立一个vector，int为数组元素的数据类型，test为动态数组名****

****简单的使用方法如下：****

****vector<int>test;//建立一个vector****

****test.push\_back(1);****

****test.push\_back(2);//把1和2压入vector，这样test[0]就是1,test[1]就是2****

****自己见到的实例：****

****vector<vector<Point2f> > points; //定义一个二维数组****

****points[0].size();  //指第一行的列数****

****1 、基本操作****

****(1)头文件#include<vector>.****

****(2)创建vector对象，vector<int> vec;****

****(3)尾部插入数字：vec.push\_back(a);****

****(4)使用下标访问元素，cout<<vec[0]<<endl;记住下标是从0开始的。****

****(5)使用迭代器访问元素.****

****vector<int>::iterator it;****

****for(it=vec.begin();it!=vec.end();it++)****

****cout<<\*it<<endl;****

****(6)插入元素：    vec.insert(vec.begin()+i,a);在第i+1个元素前面插入a;****

****(7)删除元素：    vec.erase(vec.begin()+2);删除第3个元素****

****vec.erase(vec.begin()+i,vec.end()+j);删除区间[i,j-1];区间从0开始****

****(8)向量大小:vec.size();****

****(9)清空:vec.clear();****

3、算法

****(1) 使用reverse将元素翻转：需要头文件#include<algorithm>****

****reverse(vec.begin(),vec.end());将元素翻转，即逆序排列！****

****(在vector中，如果一个函数中需要两个迭代器，一般后一个都不包含)****

****(2)使用sort排序：需要头文件#include<algorithm>，****

****sort(vec.begin(),vec.end());(默认是按升序排列,即从小到大).****

****可以通过重写排序比较函数按照降序比较，如下：****

****定义排序比较函数：****

****bool Comp(const int &a,const int &b)  
{  
    return a>b;  
}  
调用时:sort(vec.begin(),vec.end(),Comp)，这样就降序排序。****

****2.set中常用的方法****

****begin()     　　 ,返回set容器的第一个元素****

****end() 　　　　 ,返回set容器的最后一个元素****

****clear()   　　     ,删除set容器中的所有的元素****

****empty() 　　　,判断set容器是否为空****

****max\_size() 　 ,返回set容器可能包含的元素最大个数****

****size() 　　　　 ,返回当前set容器中的元素个数****

****rbegin　　　　 ,返回的值和end()相同****

****rend()　　　　 ,返回的值和rbegin()相同****

****count()**** 用来查找set中某个某个键值出现的次数。这个函数在set并不是很实用，因为一个键值在set只可能出现0或1次，这样就变成了判断某一键值是否在set出现过了。

****equal\_range()**** ，返回一对定位器，分别表示第一个大于或等于给定关键值的元素和 第一个大于给定关键值的元素，这个返回值是一个pair类型，如果这一对定位器中哪个返回失败，就会等于end()的值。具体这个有什么用途我还没遇到过~~~

****erase(iterator)  ,删除定位器iterator指向的值****

****erase(first,second),删除定位器first和second之间的值****

****erase(key\_value),删除键值key\_value的值****

****find()**** ，返回给定值值得定位器，如果没找到则返回end()。

****insert(key\_value);**** 将key\_value插入到set中 ，返回值是pair<set<int>::iterator,bool>，bool标志着插入是否成功，而iterator代表插入的位置，若key\_value已经在set中，则iterator表示的key\_value在set中的位置。

****inset(first,second);****将定位器first到second之间的元素插入到set中，返回值是void.

****lower\_bound(key\_value)**** ，返回第一个大于等于key\_value的定位器

****upper\_bound(key\_value)，****返回最后一个大于等于key\_value的定位器

# **[C++ 中queue（队列）的用法](https://www.cnblogs.com/yoke/p/6080092.html)**

#include <iostream>  
#include <queue>  
#include <assert.h>  
/\*  
调用的时候要有头文件： #include<stdlib.h> 或

#include<cstdlib> +

#include<queue>       #include<queue>  
详细用法:  
定义一个queue的变量     queue<Type> M  
查看是否为空范例        M.empty()    是的话返回1，不是返回0;  
从已有元素后面增加元素   M.push()  
输出现有元素的个数      M.size()  
显示第一个元素          M.front()  
显示最后一个元素        M.back()  
清除第一个元素          M.pop()

# C++中map用法详解

2016年09月20日 23:28:33

阅读数：41247

   Map是c++的一个标准容器，她提供了很好一对一的关系，在一些程序中建立一个map可以起到事半功倍的效果，总结了一些map基本简单实用的操作！  
1. map最基本的构造函数；  
   map<string , int >mapstring;         map<int ,string >mapint;  
   map<sring, char>mapstring;         map< char ,string>mapchar;  
   map<char ,int>mapchar;            map<int ,char >mapint；  
2. map添加数据；  
   map<int ,string> maplive;    
   1.maplive.insert(pair<int,string>(102,"aclive"));  
   2.maplive.insert(map<int,string>::value\_type(321,"hai"));  
   3, maplive[112]="April";//map中最简单最常用的插入添加！  
3，map中元素的查找：  
   find()函数返回一个迭代器指向键值为key的元素，如果没找到就返回指向map尾部的迭代器。          
   map<int ,string >::iterator l\_it;;   
   l\_it=maplive.find(112);  
   if(l\_it==maplive.end())  
                cout<<"we do not find 112"<<endl;  
   else cout<<"wo find 112"<<endl;  
4,map中元素的删除：  
   如果删除112；  
   map<int ,string >::iterator l\_it;;  
   l\_it=maplive.find(112);  
   if(l\_it==maplive.end())  
        cout<<"we do not find 112"<<endl;  
   else  maplive.erase(l\_it);  //delete 112;

Problem Description

C国的死对头A国这段时间正在进行军事演习，所以C国间谍头子Derek和他手下Tidy又开始忙乎了。A国在海岸线沿直线布置了N个工兵营地,Derek和Tidy的任务就是要监视这些工兵营地的活动情况。由于采取了某种先进的监测手段，所以每个工兵营地的人数C国都掌握的一清二楚,每个工兵营地的人数都有可能发生变动，可能增加或减少若干人手,但这些都逃不过C国的监视。  
中央情报局要研究敌人究竟演习什么战术,所以Tidy要随时向Derek汇报某一段连续的工兵营地一共有多少人,例如Derek问:“Tidy,马上汇报第3个营地到第10个营地共有多少人!”Tidy就要马上开始计算这一段的总人数并汇报。但敌兵营地的人数经常变动，而Derek每次询问的段都不一样，所以Tidy不得不每次都一个一个营地的去数，很快就精疲力尽了，Derek对Tidy的计算速度越来越不满:"你个死肥仔，算得这么慢，我炒你鱿鱼!”Tidy想：“你自己来算算看，这可真是一项累人的工作!我恨不得你炒我鱿鱼呢!”无奈之下，Tidy只好打电话向计算机专家Windbreaker求救,Windbreaker说：“死肥仔，叫你平时做多点acm题和看多点算法书，现在尝到苦果了吧!”Tidy说："我知错了。。。"但Windbreaker已经挂掉电话了。Tidy很苦恼，这么算他真的会崩溃的，聪明的读者，你能写个程序帮他完成这项工作吗？不过如果你的程序效率不够高的话，Tidy还是会受到Derek的责骂的.

Input

第一行一个整数T，表示有T组数据。  
每组数据第一行一个正整数N（N<=50000）,表示敌人有N个工兵营地，接下来有N个正整数,第i个正整数ai代表第i个工兵营地里开始时有ai个人（1<=ai<=50）。  
接下来每行有一条命令，命令有4种形式：  
(1) Add i j,i和j为正整数,表示第i个营地增加j个人（j不超过30）  
(2)Sub i j ,i和j为正整数,表示第i个营地减少j个人（j不超过30）;  
(3)Query i j ,i和j为正整数,i<=j，表示询问第i到第j个营地的总人数;  
(4)End 表示结束，这条命令在每组数据最后出现;  
每组数据最多有40000条命令

Output

对第i组数据,首先输出“Case i:”和回车,  
对于每个Query询问，输出一个整数并回车,表示询问的段中的总人数,这个数保持在int以内。

Sample Input

1 10 1 2 3 4 5 6 7 8 9 10 Query 1 3 Add 3 6 Query 2 7 Sub 10 2 Add 6 3 Query 3 10 End

Sample Output

Case 1: 6 33 59

虽然在大一寒假的时候曾经试过学习线段树，但是后来由于觉得麻烦所以放弃了，现在重新学习线段树，经过这么长时间的锻炼代码的风格也固定了下来，下面是全新的线段树写法，还是这个最经典最裸的线段树单点更新题

Problem Description

C国的死对头A国这段时间正在进行军事演习，所以C国间谍头子Derek和他手下Tidy又开始忙乎了。A国在海岸线沿直线布置了N个工兵营地,Derek和Tidy的任务就是要监视这些工兵营地的活动情况。由于采取了某种先进的监测手段，所以每个工兵营地的人数C国都掌握的一清二楚,每个工兵营地的人数都有可能发生变动，可能增加或减少若干人手,但这些都逃不过C国的监视。

中央情报局要研究敌人究竟演习什么战术,所以Tidy要随时向Derek汇报某一段连续的工兵营地一共有多少人,例如Derek问:“Tidy,马上汇报第3个营地到第10个营地共有多少人!”Tidy就要马上开始计算这一段的总人数并汇报。但敌兵营地的人数经常变动，而Derek每次询问的段都不一样，所以Tidy不得不每次都一个一个营地的去数，很快就精疲力尽了，Derek对Tidy的计算速度越来越不满:"你个死肥仔，算得这么慢，我炒你鱿鱼!”Tidy想：“你自己来算算看，这可真是一项累人的工作!我恨不得你炒我鱿鱼呢!”无奈之下，Tidy只好打电话向计算机专家Windbreaker求救,Windbreaker说：“死肥仔，叫你平时做多点acm题和看多点算法书，现在尝到苦果了吧!”Tidy说："我知错了。。。"但Windbreaker已经挂掉电话了。Tidy很苦恼，这么算他真的会崩溃的，聪明的读者，你能写个程序帮他完成这项工作吗？不过如果你的程序效率不够高的话，Tidy还是会受到Derek的责骂的.

Input

第一行一个整数T，表示有T组数据。

每组数据第一行一个正整数N（N<=50000）,表示敌人有N个工兵营地，接下来有N个正整数,第i个正整数ai代表第i个工兵营地里开始时有ai个人（1<=ai<=50）。

接下来每行有一条命令，命令有4种形式：

(1) Add i j,i和j为正整数,表示第i个营地增加j个人（j不超过30）

(2)Sub i j ,i和j为正整数,表示第i个营地减少j个人（j不超过30）;

(3)Query i j ,i和j为正整数,i<=j，表示询问第i到第j个营地的总人数;

(4)End 表示结束，这条命令在每组数据最后出现;

每组数据最多有40000条命令

Output

对第i组数据,首先输出“Case i:”和回车,

对于每个Query询问，输出一个整数并回车,表示询问的段中的总人数,这个数保持在int以内。

Sample Input

1 10 1 2 3 4 5 6 7 8 9 10 Query 1 3 Add 3 6 Query 2 7 Sub 10 2 Add 6 3 Query 3 10 End

Sample Output

Case 1: 6 33 59

虽然在大一寒假的时候曾经试过学习线段树，但是后来由于觉得麻烦所以放弃了，现在重新学习线段树，经过这么长时间的锻炼代码的风格也固定了下来，下面是全新的线段树写法，还是这个最经典最裸的线段树单点更新题

#include <stdio.h>

#include <string.h>

#include <algorithm>

using namespace std;

int sum,n;

struct node

{

int l,r,n;

} a[1000000];

void init()//新建一个线段树

{

int i,k;

for(k = 1; k<n; k<<=1);

for(i = k; i<2\*k; i++)

{

a[i].l = a[i].r = i-k+1;

a[i].n = 0;

}

for(i = k-1; i>0; i--)

{

a[i].l = a[2\*i].l;

a[i].r = a[2\*i+1].r;

a[i].n = 0;

}

}

void insert(int i,int x,int m)//线段树的插入

{

if(x>=a[i].l && x<=a[i].r)

a[i].n+=m;

if(a[i].l == a[i].r)

return ;

int mid = (a[i].l+a[i].r)/2;

if(x>mid)

insert(2\*i+1,x,m);

else

insert(2\*i,x,m);

}

void find(int x,int y,int i)//线段树的查询

{

if(a[i].l == x && a[i].r == y)

{

sum+=a[i].n;

return ;

}

if(a[i].l == a[i].r)

return ;

int mid = (a[i].l+a[i].r)/2;

if(x>mid)

find(x,y,2\*i+1);

else if(y<=mid)

find(x,y,2\*i);

else

{

find(x,mid,2\*i);

find(mid+1,y,2\*i+1);

}

}

int main()

{

int t,cas = 1,x,y,i,j,k;

char str[10];

scanf("%d",&t);

while(t--)

{

scanf("%d",&n);

init();

for(i = 1; i<=n; i++)

{

scanf("%d",&k);

insert(1,i,k);

}

printf("Case %d:\n",cas++);

while(scanf("%s",str) && str[0]!='E')

{

scanf("%d%d",&x,&y);

if(!strcmp(str,"Add"))

insert(1,x,y);

else if(!strcmp(str,"Sub"))

insert(1,x,-y);

else if(!strcmp(str,"Query"))

{

sum = 0;

find(x,y,1);

printf("%d\n",sum);

}

}

}

return 0;

}

# **Color the ball**

****Time Limit: 9000/3000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 8617    Accepted Submission(s): 4407****

**Problem Description**

N个气球排成一排，从左到右依次编号为1,2,3....N.每次给定2个整数a b(a <= b),lele便为骑上他的“小飞鸽"牌电动车从气球a开始到气球b依次给每个气球涂一次颜色。但是N次以后lele已经忘记了第I个气球已经涂过几次颜色了，你能帮他算出每个气球被涂过几次颜色吗？

**Input**

每个测试实例第一行为一个整数N,(N <= 100000).接下来的N行，每行包括2个整数a b(1 <= a <= b <= N)。  
当N = 0，输入结束。

**Output**

每个测试实例输出一行，包括N个整数，第I个数代表第I个气球总共被涂色的次数。

**Sample Input**

3 1 1 2 2 3 3 3 1 1 1 2 1 3 0

**Sample Output**

1 1 1 3 2 1

单点更新

#include<stdio.h>  
#include<algorithm>  
#include<string.h>  
#include<iostream>  
using namespace std;  
const int MAXN=50005;  
struct Node  
{  
 int l,r;  
 int nSum;  
}segTree[MAXN\*3];  
int num[MAXN];  
void Build(int i,int l,int r)  
{  
 segTree[i].l=l;  
 segTree[i].r=r;  
 if(l==r)  
 {  
 segTree[i].nSum=num[l];  
 return;  
 }   
 int mid=(l+r)>>1;  
 Build(i<<1,l,mid);  
 Build(i<<1|1,mid+1,r);  
 segTree[i].nSum=segTree[i<<1].nSum+segTree[i<<1|1].nSum;   
}   
void Add(int i,int t,int b)  
{  
 segTree[i].nSum+=b;  
 if(segTree[i].l==t&&segTree[i].r==t) return;  
 int mid=(segTree[i].l+segTree[i].r)>>1;  
 if(t<=mid) Add(i<<1,t,b);  
 else Add(i<<1|1,t,b);  
}   
int Query(int i,int l,int r)  
{  
 if(l==segTree[i].l&&r==segTree[i].r)  
 return segTree[i].nSum;  
 int mid=(segTree[i].l+segTree[i].r)>>1;  
 if(r<=mid) return Query(i<<1,l,r);  
 else if(l>mid) return Query(i<<1|1,l,r);  
 else return Query(i<<1,l,mid)+Query(i<<1|1,mid+1,r);  
}   
int main()  
{  
 int T;  
 int iCase=0;  
 int n,i;  
 char str[10];  
 int a,b;  
 scanf("%d",&T);  
 while(T--)  
 {  
 iCase++;  
 scanf("%d",&n);  
 for(i=1;i<=n;i++)  
 scanf("%d",&num[i]);  
 Build(1,1,n);  
 printf("Case %d:\n",iCase);  
 while(scanf("%s",&str))  
 {  
 if(strcmp(str,"End")==0) break;  
 scanf("%d%d",&a,&b);  
 if(strcmp(str,"Add")==0) Add(1,a,b);  
 else if(strcmp(str,"Sub")==0) Add(1,a,-b);  
 else printf("%d\n",Query(1,a,b));  
 }   
 }  
 return 0;   
}

#include<iostream>

#include<string>

#include<cstring>

#include<algorithm>

#include<cstdio>

#include<cmath>

#include<cctype>

#include<iomanip>

using namespace std;

const int inf=10000000;

int f[100005];

int main(){

int n;

while(scanf("%d",&n)!=EOF&&n){

memset(f,0,sizeof(f));

for(int i=0,a,b;i<n;++i){

scanf("%d%d",&a,&b);

f[a]++;

f[b+1]--;

}

cout<<f[1];

for(int i=2;i<=n;++i){

f[i]+=f[i-1];

cout<<' '<<f[i];

}

cout<<endl;

}

return 0;

}

**Description**

You have *N* integers, *A*1, *A*2, ... , *AN*. You need to deal with two kinds of operations. One type of operation is to add some given number to each number in a given interval. The other is to ask for the sum of numbers in a given interval.

**Input**

The first line contains two numbers *N* and *Q*. 1 ≤ *N*,*Q* ≤ 100000.  
The second line contains *N* numbers, the initial values of *A*1, *A*2, ... , *AN*. -1000000000 ≤ *Ai* ≤ 1000000000.  
Each of the next *Q* lines represents an operation.  
"C *abc*" means adding *c* to each of *Aa*, *Aa*+1, ... , *Ab*. -10000 ≤ *c* ≤ 10000.  
"Q *ab*" means querying the sum of *Aa*, *Aa*+1, ... , *Ab*.

**Output**

You need to answer all *Q* commands in order. One answer in a line.

**Sample Input**

10 5

1 2 3 4 5 6 7 8 9 10

Q 4 4

Q 1 10

Q 2 4

C 3 6 3

Q 2 4

**Sample Output**

4

55

9

15

区间更新

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<vector>

typedef long long LL;

using namespace std;

#define REPF( i , a , b ) for ( int i = a ; i <= b ; ++ i )

#define REP( i , n ) for ( int i = 0 ; i < n ; ++ i )

#define CLEAR( a , x ) memset ( a , x , sizeof a )

const int maxn=100000;

int num[maxn];

LL sum[maxn<<2],add[maxn<<2];

int N,Q;

void pushup(int rs)

{

sum[rs]=sum[rs<<1]+sum[rs<<1|1];

}

void pushdown(int rs,int l)

{

if(add[rs])

{

add[rs<<1]+=add[rs];

add[rs<<1|1]+=add[rs];

sum[rs<<1]+=add[rs]\*(l-(l>>1));

sum[rs<<1|1]+=add[rs]\*(l>>1);

add[rs]=0;

}

}

void build(int rs,int l,int r)

{

if(l==r)

{

scanf("%I64d",&sum[rs]);

return ;

}

int mid=(l+r)>>1;

build(rs<<1,l,mid);

build(rs<<1|1,mid+1,r);

pushup(rs);

}

void update(int c,int x,int y,int l,int r,int rs)

{

if(l>=x&&r<=y)

{

add[rs]+=c;

sum[rs]+=(LL)c\*(r-l+1);

return ;

}

pushdown(rs,r-l+1);

int mid=(l+r)>>1;

if(x<=mid) update(c,x,y,l,mid,rs<<1);

if(y>mid) update(c,x,y,mid+1,r,rs<<1|1);

pushup(rs);

}

LL query(int x,int y,int l,int r,int rs)

{

if(l>=x&&r<=y)

return sum[rs];

pushdown(rs,r-l+1);

int mid=(l+r)>>1;

LL ans=0;

if(x<=mid) ans+=query(x,y,l,mid,rs<<1);

if(y>mid) ans+=query(x,y,mid+1,r,rs<<1|1);

return ans;

}

int main()

{

int x,y,z;

std::ios::sync\_with\_stdio(false);

while(~scanf("%d%d",&N,&Q))

{

CLEAR(sum,0);

CLEAR(add,0);

build(1,1,N);

char str[2];

while(Q--)

{

scanf("%s",str);

if(str[0]=='C')

{

scanf("%d%d%d",&x,&y,&z);

update(z,x,y,1,N,1);

}

else

{

scanf("%d%d",&x,&y);

printf("%I64d\n",query(x,y,1,N,1));

}

}

}

return 0;

}

# **最大子矩阵**

**Time Limit: 30000/10000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 2901    Accepted Submission(s): 1454**

Problem Description

给你一个m×n的整数矩阵，在上面找一个x×y的子矩阵，使子矩阵中所有元素的和最大。

Input

输入数据的第一行为一个正整数T，表示有T组测试数据。每一组测试数据的第一行为四个正整数m,n,x,y（0<m,n<1000 AND 0<x<=m AND 0<y<=n），表示给定的矩形有m行n列。接下来这个矩阵，有m行，每行有n个不大于1000的正整数。

Output

对于每组数据，输出一个整数，表示子矩阵的最大和。

Sample Input

1

4 5 2 2

3 361 649 676

588 992 762 156

993 169 662 34

638 89 543 525

165 254 809 280

Sample Output

2474

1 /\*二维树状数组：同样不要忘记c的初始化，

2 modify 的功能是改变元素(x, y)，

3 sum的功能则是求从元素(1, 1)开始到(x, y)的总和，

4 同样，可以求出任意一个子矩阵内的所有元素之和，

5 即sum(x2, y2) - sum(x1-1, y2) - sum(x2, y1-1) + sum(x1-1, y1-1)

6 \*/

7

8

9 #include<iostream>

10

11 using namespace std;

12

13 int N,M;

14 int c[1005][1005];

15

16 int lowbit( int x )

17 {

18 return x & (-x);

19 }

20

21 void modify( int x, int y, int delta )

22 {

23 int i, j;

24 for(i=x; i<=N; i+=lowbit(i))

25 {

26 for(j=y; j<=M; j+=lowbit(j))

27 {

28 c[i][j] += delta;

29 }

30 }

31 }

32

33 int sum( int x, int y )

34 {

35 int res = 0, i, j;

36 for(i=x; i>0; i-=lowbit(i))

37 {

38 for(j=y; j>0; j-=lowbit(j))

39 {

40 res += c[i][j];

41 }

42 }

43 return res;

44 }

45

46 void init ()

47 {

48 int i,j;

49 for(i=0;i<=N;i++)

50 for(j=0;j<=M;j++)

51 c[i][j]=0;

52

53 }

54

55 int main()

56 {

57 int t;

58 scanf("%d",&t);

59 while(t--)

60 {

61 int x,y,i,j,k,temp;

62 int max = 0 ;

63 // memset(c,0,sizeof(c));

64 // c[1005][1005]= {{0}};

65 scanf("%d%d%d%d",&N,&M,&x,&y);

66 init ();

67

68 for(i=1;i<=N;i++)

69 for(j=1;j<=M;j++)

70 {

71 scanf("%d",&k);

72 modify(i,j,k);

73 }

74 for(i=1;i+x-1<=N;i++)

75 for(j=1;j+y-1<=M;j++)

76 {

77 temp=sum(i+x-1,j+y-1)-sum(i-1,j+y-1)-sum(i+x-1,j-1)+sum(i-1,j-1);

78

79 if(temp>max)max= temp ;

80 }

81 printf("%d\n",max);

82

83

84 }

85 return 0 ;

86

87 }

**Count Color**

|  |  |  |
| --- | --- | --- |
| ****Time Limit:**** 1000MS |  | ****Memory Limit:**** 65536K |
| ****Total Submissions:**** 40402 |  | ****Accepted:**** 12186 |

**Description**

Chosen Problem Solving and Program design as an optional course, you are required to solve all kinds of problems. Here, we get a new problem.   
  
There is a very long board with length L centimeter, L is a positive integer, so we can evenly divide the board into L segments, and they are labeled by 1, 2, ... L from left to right, each is 1 centimeter long. Now we have to color the board - one segment with only one color. We can do following two operations on the board:   
  
1. "C A B C" Color the board from segment A to segment B with color C.   
2. "P A B" Output the number of different colors painted between segment A and segment B (including).   
  
In our daily life, we have very few words to describe a color (red, green, blue, yellow…), so you may assume that the total number of different colors T is very small. To make it simple, we express the names of colors as color 1, color 2, ... color T. At the beginning, the board was painted in color 1. Now the rest of problem is left to your.

**Input**

First line of input contains L (1 <= L <= 100000), T (1 <= T <= 30) and O (1 <= O <= 100000). Here O denotes the number of operations. Following O lines, each contains "C A B C" or "P A B" (here A, B, C are integers, and A may be larger than B) as an operation defined previously.

**Output**

Ouput results of the output operation in order, each line contains a number.

**Sample Input**

2 2 4

C 1 1 2

P 1 2

C 2 2 2

P 1 2

**Sample Output**

2

1

**Source**

[POJ Monthly--2006.03.26](http://poj.org/searchproblem?field=source&key=POJ+Monthly--2006.03.26" \t "https://blog.csdn.net/qiqi_skystar/article/details/_blank),dodo

题目大意：一个长度为L的区间，最多有T种颜色，并且有O种操作，接下去有o行。

一共就两种操作：1、C a b c：表示的是将【a,b】这个区间染成颜色c。 2、P a b ：表示的是询问【a,b】这个区间有多少种颜色。

解题思路：这个题目需要注意的是不能一直更新到最下面，就更新到符合的区间即可，否则会超时。

详见代码。

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <cmath>

#include <queue>

using namespace std;

struct node

{

int l,r;

int num;

} s[400010];

int vis[35];

void InitTree(int l,int r,int k)

{

s[k].l=l;

s[k].r=r;

s[k].num=1;

int mid=(l+r)/2;

if (l==r)

return ;

InitTree(l,mid,2\*k);

InitTree(mid+1,r,2\*k+1);

}

void UpdataTree(int l,int r,int c,int k)

{

if(s[k].l==l&&s[k].r==r)

{

s[k].num=c;

return;

}

if (s[k].num==c)

return;

if (s[k].num!=-1)//如果所查询的区间不是多种颜色

{

s[2\*k].num=s[k].num;//更新区间的颜色

s[2\*k+1].num=s[k].num;

s[k].num=-1;//-1表示该区间有多种颜色

}

int mid=(s[k].l+s[k].r)/2;

if (l>mid)

UpdataTree(l,r,c,2\*k+1);

else if (r<=mid)

UpdataTree(l,r,c,2\*k);

else

{

UpdataTree(l,mid,c,2\*k);

UpdataTree(mid+1,r,c,2\*k+1);

}

}

void SearchTree(int l,int r,int k)

{

if (s[k].num!=-1)

{

vis[s[k].num]=1;

return;

}

int mid=(s[k].l+s[k].r)/2;

if (r<=mid)

SearchTree(l,r,2\*k);

else if (l>mid)

SearchTree(l,r,2\*k+1);

else

{

SearchTree(l,mid,2\*k);

SearchTree(mid+1,r,2\*k+1);

}

}

int main()

{

int l,t,o,ans;

while (~scanf("%d%d%d",&l,&t,&o))

{

InitTree(1,l,1);

while (o--)

{

char ch;

int a,b,c;

getchar();

scanf("%c",&ch);

if (ch=='C')

{

scanf("%d%d%d",&a,&b,&c);

if (a>b)

swap(a,b);

UpdataTree(a,b,c,1);

}

else

{

scanf("%d%d",&a,&b);

if (a>b)

swap(a,b);

memset(vis,0,sizeof(vis));

SearchTree(a,b,1);

ans=0;

for (int i=1; i<=t; i++)

if (vis[i]==1)

ans++;

printf ("%d\n",ans);

}

}

}

return 0;

}

# **Necklace**

****Time Limit: 15000/5000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 4003    Accepted Submission(s): 1330****

**Problem Description**

Mery has a beautiful necklace. The necklace is made up of N magic balls. Each ball has a beautiful value. The balls with the same beautiful value look the same, so if two or more balls have the same beautiful value, we just count it once. We define the beautiful value of some interval [x,y] as F(x,y). F(x,y) is calculated as the sum of the beautiful value from the xth ball to the yth ball and the same value is ONLY COUNTED ONCE. For example, if the necklace is 1 1 1 2 3 1, we have F(1,3)=1, F(2,4)=3, F(2,6)=6.  
  
Now Mery thinks the necklace is too long. She plans to take some continuous part of the necklace to build a new one. She wants to know each of the beautiful value of M continuous parts of the necklace. She will give you M intervals [L,R] (1<=L<=R<=N) and you must tell her F(L,R) of them.

**Input**

The first line is T(T<=10), representing the number of test cases.  
  For each case, the first line is a number N,1 <=N <=50000, indicating the number of the magic balls. The second line contains N non-negative integer numbers not greater 1000000, representing the beautiful value of the N balls. The third line has a number M, 1 <=M <=200000, meaning the nunber of the queries. Each of the next M lines contains L and R, the query.

**Output**

For each query, output a line contains an integer number, representing the result of the query.

**Sample Input**

2 6 1 2 3 4 3 5 3 1 2 3 5 2 6 6 1 1 1 2 3 5 3 1 1 2 4 3 5

**Sample Output**

3 7 14 1 3 6

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <vector>

using namespace std;

typedef long long LL;

#define lson rt << 1, l, mid

#define rson rt << 1|1, mid + 1, r

#define root 1, 1, N

const int MAXN = 5e4 + 5;

const int MAXM = 2e5 + 5;

const int MAXS = 1e6 + 5;

int N, M, T, pre[MAXS];

LL Ans[MAXM],Sum[MAXN << 2];

struct node {

int l, r, id;

bool operator < (const node & object) const {

return r < object.r;

}

} Node[MAXM];

void pushup(int rt) {

Sum[rt] = Sum[rt << 1] + Sum[rt << 1|1];

}

void build(int rt,int l,int r) {

if(l == r) {

scanf("%I64d", &Sum[rt]);

return;

}

int mid = (l + r) >> 1;

build(lson);

build(rson);

pushup(rt);

}

void update(int p,int rt, int l, int r) {

if(l == r) {

Sum[rt] = 0;

return;

}

int mid = (l + r) >> 1;

if(p <= mid) update(p, lson);

else update(p, rson);

pushup(rt);

}

LL query(int L, int R, int rt, int l, int r) {

if(L <= l && r <= R) {

return Sum[rt];

}

int mid = (l + r) >> 1;

LL ret = 0;

if(L <= mid) ret += query(L, R, lson);

if(R > mid) ret += query(L, R, rson);

return ret;

}

int main() {

//freopen("D://imput.txt","r",stdin);

scanf("%d", &T);

while(T --) {

scanf("%d", &N);

build(root);

scanf("%d", &M);

for(int i = 1 ; i <= M ; i ++) {

scanf("%d %d", &Node[i].l, &Node[i].r);

Node[i].id = i;

}

memset(pre, -1, sizeof(pre));

sort(Node + 1, Node + M + 1);

for(int i = 1,j = 1; i <= N; i++) {

int tmp = query(i, i, root);

if(tmp != 0 && pre[tmp] != -1) {//如果前面存在重复的数字则删除他

update(pre[tmp],root);

}

pre[tmp] = i;

while(j <= M && Node[j].r == i) {//如果右边的值等于当前的值，则进行求和，大家可以参考我博客里的教程

Ans[Node[j].id] = query(Node[j].l, Node[j].r, root);

j ++;

}

}

for(int i = 1; i <= M ; i ++) {

printf("%I64d\n", Ans[i]);

}

}

return 0;

}

题意：给定一个数组，每次询问一个区间[l, r]，求区间内所有数字的和，出现多次的数字只加一次

思路：莫队算法扫一遍即可，注意会爆int。我数组开小了不少RE而是TLE，蛋疼。。。

#include <iostream>

#include <cstdio>

#include <cstring>

#include <algorithm>

#include <cmath>

#include <set>

using namespace std;

typedef long long ll;

const int N = 50010, MOD = 1000000007;

struct node

{

int l, r, id;

} g[N\*4];

int n, m, unit;

int a[N], num[N\*20];

ll tmp, res[N\*4];

void add(int i)

{

if(num[i] == 0) tmp += i;

num[i]++;

}

void del(int i)

{

num[i]--;

if(num[i] == 0) tmp -= i;

}

void solve()

{

unit = (int)sqrt(1.0\*n);

memset(num, 0, sizeof num);

sort(g+1, g+1+m, [](node a, node b){return a.l/unit != b.l/unit ? a.l/unit < b.l/unit : a.r < b.r;});

int l = 1, r = 0;

tmp = 0;

for(int i = 1; i <= m; i++)

{

while(r < g[i].r) add(a[++r]);

while(r > g[i].r) del(a[r--]);

while(l < g[i].l) del(a[l++]);

while(l > g[i].l) add(a[--l]);

res[g[i].id] = tmp;

}

for(int i = 1; i <= m; i++) printf("%lld\n", res[i]);

}

int main()

{

int t;

scanf("%d", &t);

while(t--)

{

scanf("%d", &n);

for(int i = 1; i <= n; i++) scanf("%d", &a[i]);

scanf("%d", &m);

for(int i = 1; i <= m; i++) scanf("%d%d", &g[i].l, &g[i].r), g[i].id = i;

solve();

}

return 0;

}

# **[主席树：POJ2104 K-th Number (主席树模板题)](https://www.cnblogs.com/TenderRun/p/5202042.html)**

K-th Number

|  |  |  |
| --- | --- | --- |
| **Time Limit:** 20000MS |  | **Memory Limit:** 65536K |
| **Total Submissions:** 44952 |  | **Accepted:** 14951 |
| **Case Time Limit:** 2000MS | | |

Description

You are working for Macrohard company in data structures department. After failing your previous task about key insertion you were asked to write a new data structure that would be able to return quickly k-th order statistics in the array segment.   
That is, given an array a[1...n] of different integer numbers, your program must answer a series of questions Q(i, j, k) in the form: "What would be the k-th number in a[i...j] segment, if this segment was sorted?"   
For example, consider the array a = (1, 5, 2, 6, 3, 7, 4). Let the question be Q(2, 5, 3). The segment a[2...5] is (5, 2, 6, 3). If we sort this segment, we get (2, 3, 5, 6), the third number is 5, and therefore the answer to the question is 5.

Input

The first line of the input file contains n --- the size of the array, and m --- the number of questions to answer (1 <= n <= 100 000, 1 <= m <= 5 000).   
The second line contains n different integer numbers not exceeding 109 by their absolute values --- the array for which the answers should be given.   
The following m lines contain question descriptions, each description consists of three numbers: i, j, and k (1 <= i <= j <= n, 1 <= k <= j - i + 1) and represents the question Q(i, j, k).

Output

For each question output the answer to it --- the k-th number in sorted a[i...j] segment.

Sample Input

7 3

1 5 2 6 3 7 4

2 5 3

4 4 1

1 7 3

Sample Output

5

6

3

Hint

This problem has huge input,so please use c-style input(scanf,printf),or you may got time limit exceed.

#include <iostream>

#include <cstdio>

#include <algorithm>

using namespace std;

struct Node{

int a,b,rs,ls,sum;

}tr[2000010];

int a[100010],b[100010];

int rt[100010],pos,cnt;

void Build(int &node,int a,int b)

{

node=++cnt;

tr[node].a=a;

tr[node].b=b;

if(a==b)return;

int mid=(a+b)>>1;

Build(tr[node].ls,a,mid);

Build(tr[node].rs,mid+1,b);

}

void Insert(int pre,int &node)

{

node=++cnt;

tr[node].ls=tr[pre].ls;

tr[node].rs=tr[pre].rs;

tr[node].a=tr[pre].a;

tr[node].b=tr[pre].b;

tr[node].sum=tr[pre].sum+1;

if(tr[node].a==tr[node].b)return;

int mid=(tr[node].a+tr[node].b)>>1;

if(mid>=pos)Insert(tr[pre].ls,tr[node].ls);

else Insert(tr[pre].rs,tr[node].rs);

}

int Query(int pre,int node,int k)

{

if(tr[node].ls==tr[node].rs)return b[tr[node].a];

int cmp=tr[tr[node].ls].sum-tr[tr[pre].ls].sum;

if(cmp>=k)return Query(tr[pre].ls,tr[node].ls,k);

else return Query(tr[pre].rs,tr[node].rs,k-cmp);

}

int main()

{

int n,q;

scanf("%d%d",&n,&q);

for(int i=1;i<=n;b[i]=a[i],i++)

scanf("%d",&a[i]);

sort(b+1,b+n+1);

Build(rt[0],1,n);

for(int i=1;i<=n;i++)

{

pos=lower\_bound(b+1,b+n+1,a[i])-b;

Insert(rt[i-1],rt[i]);

}

int l,r,k;

for(int i=1;i<=q;i++)

{

scanf("%d%d%d",&l,&r,&k);

printf("%d\n",Query(rt[l-1],rt[r],k));

}

return 0;

}

## DQUERY - D-query

Given a sequence of n numbers a1, a2, ..., an and a number of d-queries. A d-query is a pair (i, j) (1 ≤ i ≤ j ≤ n). For each d-query (i, j), you have to return the number of distinct elements in the subsequence ai, ai+1, ..., aj.

### Input

* Line 1: n (1 ≤ n ≤ 30000).
* Line 2: n numbers a1, a2, ..., an (1 ≤ ai ≤ 106).
* Line 3: q (1 ≤ q ≤ 200000), the number of d-queries.
* In the next q lines, each line contains 2 numbers i, j representing a d-query (1 ≤ i ≤ j ≤ n).

### Output

* For each d-query (i, j), print the number of distinct elements in the subsequence ai, ai+1, ..., aj in a single line.

### Example

**Input**

5

1 1 2 1 3

3

1 5

2 4

3 5

**Output**

3

2

3

题意是查询区间内不同数字的种类。

莫队很经典的题，算每一个元素在区间内的贡献。

代码：

#pragma comment(linker, "/STACK:102400000,102400000")

#pragma warning(disable:4996)

#include <iostream>

#include <algorithm>

#include <cstring>

#include <vector>

#include <string>

#include <cstdio>

#include <cmath>

#include <queue>

#include <stack>

#include <set>

#include <map>

using namespace std;

#define INF 0x3fffffff

typedef long long ll;

const int mod = 1e9 + 7;

const int maxn = 1e6 + 5;

int n, q, bk;

int val[30005], res[200005], num[maxn];

struct no

{

int le;

int ri;

int id;

}qu[200005];

bool cmp(no n1, no n2)

{

if (n1.le / bk == n2.le / bk)

{

return n1.ri < n2.ri;

}

else

{

return n1.le / bk < n2.le / bk;

}

}

void input()

{

int i, u, v;

scanf("%d", &n);

for (i = 1; i <= n; i++)

{

scanf("%d", &val[i]);

}

scanf("%d", &q);

for (i = 1; i <= q; i++)

{

scanf("%d%d", &u, &v);

qu[i].le = u;

qu[i].ri = v;

qu[i].id = i;

}

}

void solve()

{

bk = sqrt(1.0\*n);

sort(qu + 1, qu + q + 1, cmp);

int i, j, id, ans = 0;

int pl = 1, pr = 0;

for (i = 1; i <= q; i++)

{

id = qu[i].id;

if (qu[i].le == qu[i].ri)

{

res[id] = 1;

continue;

}

if (pr < qu[i].ri)

{

for (j = pr + 1; j <= qu[i].ri; j++)

{

if (num[val[j]] == 0)

{

ans++;

}

num[val[j]]++;

}

}

else

{

for (j = pr; j > qu[i].ri; j--)

{

num[val[j]]--;

if (num[val[j]] == 0)

{

ans--;

}

}

}

pr = qu[i].ri;

if (pl < qu[i].le)

{

for (j = pl; j < qu[i].le; j++)

{

num[val[j]]--;

if (num[val[j]] == 0)

{

ans--;

}

}

}

else

{

for (j = pl - 1; j >= qu[i].le; j--)

{

if (num[val[j]] == 0)

{

ans++;

}

num[val[j]]++;

}

}

pl = qu[i].le;

res[id] = ans;

}

for (i = 1; i <= q; i++)

{

printf("%d\n", res[i]);

}

}

int main()

{

//freopen("i.txt","r",stdin);

//freopen("o.txt","w",stdout);

input();

solve();

return 0;

}

# **过山车**

****Time Limit: 1000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others) Total Submission(s): 14056    Accepted Submission(s): 6196****

**Problem Description**

RPG girls今天和大家一起去游乐场玩，终于可以坐上梦寐以求的过山车了。可是，过山车的每一排只有两个座位，而且还有条不成文的规矩，就是每个女生必须找个个男生做partner和她同坐。但是，每个女孩都有各自的想法，举个例子把，Rabbit只愿意和XHD或PQK做partner，Grass只愿意和linle或LL做partner，PrincessSnow愿意和水域浪子或伪酷儿做partner。考虑到经费问题，boss刘决定只让找到partner的人去坐过山车，其他的人，嘿嘿，就站在下面看着吧。聪明的Acmer，你可以帮忙算算最多有多少对组合可以坐上过山车吗？

**Input**

输入数据的第一行是三个整数K , M , N，分别表示可能的组合数目，女生的人数，男生的人数。0<K<=1000 1<=N 和M<=500.接下来的K行，每行有两个数，分别表示女生Ai愿意和男生Bj做partner。最后一个0结束输入。

**Output**

对于每组数据，输出一个整数，表示可以坐上过山车的最多组合数。

**Sample Input**

6 3 3

1 1

1 2

1 3

2 1

2 3

3 1

0

**Sample Output**

3

#include <stdio.h>

#include <string.h>

#include <math.h>

#include <stdlib.h>

#define N 1010

#define INF 0x3f3f3f3f

int G[N][N], used[N], vis[N], n;

bool Find(int u)

{

int i;

for(i = 1 ; i <= n ; i++)

{

if(!vis[i] && G[u][i])

{//vis[i]表示i男生还没有被别的女生选走

vis[i] = 1;

if(!used[i] || Find(used[i]))

{//如果女生used[i]没有选男生i作伴或者女生used[i]选了男生i，她放弃男生i并且另外找到了自己的伴

used[i] = u;//就让男生i与女生u作伴

return true;

}

}

}

return false;

}//该函数判断女生爱能不能找到伴

int main()

{

int k, m, i, a, b, ans;

while(scanf("%d", &k), k)

{

ans = 0;

memset(G, 0, sizeof(G));

scanf("%d%d", &m, &n);

while(k--)

{

scanf("%d%d", &a, &b);

G[a][b] = 1;//G[a][b]表示女生a愿意与男生b作伴

}//used[i]表示i男生与used[i]女生作伴

memset(used, 0, sizeof(used));

for(i = 1 ; i <= m ; i++)

{

memset(vis, 0, sizeof(vis));

if(Find(i))

ans++;

}

printf("%d\n", ans);

}

return 0;

}

AC自动机

Here you have a set of strings. A dominator is a string of the set dominating all strings else. The string SS is dominated by TT if SS is a substring of TT.

**Input**

The input contains several test cases and the first line provides the total number of cases.   
For each test case, the first line contains an integer NN indicating the size of the set.   
Each of the following NN lines describes a string of the set in lowercase.   
The total length of strings in each case has the limit of 100000100000.   
The limit is 30MB for the input file.

**Output**

For each test case, output a dominator if exist, or No if not.

**Sample Input**

3

10

you

better

worse

richer

poorer

sickness

health

death

faithfulness

youbemyweddedwifebetterworsericherpoorersicknesshealthtilldeathdouspartandpledgeyoumyfaithfulness

5

abc

cde

abcde

abcde

bcde

3

aaaaa

aaaab

aaaac

**Sample Output**

youbemyweddedwifebetterworsericherpoorersicknesshealthtilldeathdouspartandpledgeyoumyfaithfulness

abcde

No

简略题意：问是否存在一个串，其他串都是他的子串。

有可能成为目标串的一定是最长串，所有串建AC自动机，用最长串去匹配。如果匹配到了所有串，那么答案存在。   
具体做法就是每访问到一个节点，都看当前节点是否能匹配到某个串，以及这个节点一直在fail树上跳转到根部的这段路径上是否能匹配上某个串。   
这题卡自动机做法主要在两点：   
1. 卡读入，加了输入挂和不加输入挂相差500ms左右。   
2. 每次暴力在fail树跳转需要花费一定时间，并且会重复遍历。   
因为我们不需要知道每个字符串在最长串中出现的次数，因此对于遍历过的节点，不需要再重复遍历，加个标记即可。

#include <stdio.h>#include <algorithm>#include <iostream>#include <string.h>#include <queue>#define maxn 27using namespace std;

int cnt[100010];

int idx(char x) {

return x-'a';

}

struct Trie

{

int next[500010][maxn],fail[500010];

vector<int> end[500010];

int root,L;

int newnode()

{

for(int i = 0; i < maxn; i++)

next[L][i] = -1;

end[L++].clear();

return L-1;

}

void init()

{

L = 0;

root = newnode();

}

void insert(char \*buf, int len, int th)

{

int now = root;

for(int i = 0; i < len; i++)

{

if(next[now][idx(buf[i])] == -1)

next[now][idx(buf[i])] = newnode();

now = next[now][idx(buf[i])];

}

end[now].push\_back(th);

}

void build()

{

queue<int>Q;

fail[root] = root;

for(int i = 0; i < maxn; i++)

if(next[root][i] == -1)

next[root][i] = root;

else

{

fail[next[root][i]] = root;

Q.push(next[root][i]);

}

while( !Q.empty() )

{

int now = Q.front();

Q.pop();

for(int i = 0; i < maxn; i++)

if(next[now][i] == -1)

next[now][i] = next[fail[now]][i];

else

{

fail[next[now][i]]=next[fail[now]][i];

Q.push(next[now][i]);

}

}

}

bool vis[500010];

int query(string buf)

{

for(int i = 0; i < L; i++)

vis[i] = 0;

int len = buf.size();

int now = root;

int res = 0;

for(int i = 0; i < len; i++) {

now = next[now][idx(buf[i])];

int temp = now;

while( temp != root )

{

if(vis[temp]) break;

vis[temp] = 1;

for(int i = 0; i < end[temp].size(); i++) {

int v = end[temp][i];

cnt[v]++;

}

temp = fail[temp];

}

}

return res;

}

};

char s[1100000];

Trie ac;int n;int t;

int main() {

cin >> t;

while(t--) {

cin >> n;

for(int i = 1; i <= n; i++)

cnt[i] = 0;

int maxv = 0, mid;

string str;

ac.init();

for(int i = 1; i <= n; i++) {

scanf("%s", s);

int len = strlen(s);

if(len > maxv)

maxv = len, str = s;

ac.insert(s, len, i);

}

ac.build();

ac.query(str);

int sum = 0;

for(int i=1; i<=n; i++) {

if(cnt[i])

sum++;

}

for(int i = 0; i <= str.size(); i++)

s[i] = str[i];

if(sum != n) puts("No");

else cout<<str<<'\n';

}

return 0;

}

* 133

大家都会写假算法，这游戏越来越难玩了…

#include <bits/stdc++.h>#define maxn 27using namespace std;

string s[110000];

int main() {

cin.tie(0);

ios::sync\_with\_stdio(0);

int t, n;

cin >> t;

while(t--) {

cin >> n;

int maxv = 0;

int id;

for(int i = 1; i <= n; i++) {

cin >> s[i];

if(s[i].size() > maxv) {

maxv = s[i].size();

id = i;

}

}

int cnt = 0;

for(int i = 1; i <= n; i++) {

if(s[id].find(s[i]) != -1) {

cnt++;

}

}

if(cnt ==n) cout<<s[id]<<'\n';

else cout<<"No"<<'\n';

}

return 0;

}

# hdu 1251 统计难题

2014年02月12日 15:45:31

阅读数：1941

**Problem Description**

Ignatius最近遇到一个难题,老师交给他很多单词(只有小写字母组成,不会有重复的单词出现),现在老师要他统计出以某个字符串为前缀的单词数量(单词本身也是自己的前缀).

**Input**

输入数据的第一部分是一张单词表,每行一个单词,单词的长度不超过10,它们代表的是老师交给Ignatius统计的单词,一个空行代表单词表的结束.第二部分是一连串的提问,每行一个提问,每个提问都是一个字符串.  
  
注意:本题只有一组测试数据,处理到文件结束.

**Output**

对于每个提问,给出以该字符串为前缀的单词的数量.

**Sample Input**

banana band bee absolute acm ba b band abc

**Sample Output**

2 3 1 0

这是字典树的入门题目，较为简单，其实，字典树就是先建立树，然后查找，重要的是建立的过程，理解了快两天了，差不多算是理解了吧，希望大家都好好看看字典树，实践很重要

#include <bits/stdc++.h>

using namespace std;

struct Trie

{

int v;//v可以根据实际情况任意变化，在这里v是每个字母的次数;

Trie \*next[26];

};

Trie root;

void createTrie(char \*str)//建立字典树;

{

int len=strlen(str);

Trie \*p=&root,\*q;

for(int i=0;i<len;i++)

{

int id=str[i]-'a';

if(p->next[id]==NULL)

{

q=(Trie \*)malloc(sizeof(root));//申请一块新内存;

q->v=1;//v遇到新字母每一层都初始化为1;

for(int j=0;j<26;j++)

q->next[j]=NULL;

p->next[id]=q;

p=p->next[id];

}

else

{

p->next[id]->v++;//当第一个输入的字符串和后面又相等的时候，v++;

p=p->next[id];

}

}

}

int findTrie(char \*str)//在字典树里查询;

{

int len=strlen(str);

Trie \*p=&root;

for(int i=0;i<len;i++)

{

int id=str[i]-'a';

p=p->next[id];

if(p==NULL)

return 0;

}

return p->v;//相同的字母个数;

}

int main()

{

char str[15];

for(int i=0;i<26;i++)

root.next[i]=NULL;

while(gets(str)&&str[0]!='\0')

{

createTrie(str);

}

while(cin>>str)

{

int ans=findTrie(str);

cout<<ans<<endl;

}

return 0;

}

# hdoj 1248 寒冰王座 【完全背包】

2015年04月29日 13:17:39

阅读数：474

# **寒冰王座**

****Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 12612    Accepted Submission(s): 6397****

Problem Description

不死族的巫妖王发工资拉,死亡骑士拿到一张N元的钞票(记住,只有一张钞票),为了防止自己在战斗中频繁的死掉,他决定给自己买一些道具,于是他来到了地精商店前.  
  
死亡骑士:"我要买道具!"  
  
地精商人:"我们这里有三种道具,血瓶150块一个,魔法药200块一个,无敌药水350块一个."  
  
死亡骑士:"好的,给我一个血瓶."  
  
说完他掏出那张N元的大钞递给地精商人.  
  
地精商人:"我忘了提醒你了,我们这里没有找客人钱的习惯的,多的钱我们都当小费收了的,嘿嘿."  
  
死亡骑士:"......"  
  
死亡骑士想,与其把钱当小费送个他还不如自己多买一点道具,反正以后都要买的,早点买了放在家里也好,但是要尽量少让他赚小费.  
  
现在死亡骑士希望你能帮他计算一下,最少他要给地精商人多少小费.

Input

输入数据的第一行是一个整数T(1<=T<=100),代表测试数据的数量.然后是T行测试数据,每个测试数据只包含一个正整数N(1<=N<=10000),N代表死亡骑士手中钞票的面值.  
  
注意:地精商店只有题中描述的三种道具.

Output

对于每组测试数据,请你输出死亡骑士最少要浪费多少钱给地精商人作为小费.

Sample Input

2 900 250

Sample Output

0 50

模版题：

#include<stdio.h>

#include<string.h>

#include<math.h>

#include<stdlib.h>

#include<queue>

#include<stack>

#include<algorithm>

using namespace std;

int max(int x,int y)

{

return x>y?x:y;

}

int dp[10100];

int main()

{

int t,money;

int i,j;

int value[3]={150,200,350};

scanf("%d",&t);

while(t--)

{

scanf("%d",&money);

memset(dp,0,sizeof(dp));

for(i=0;i<3;i++)

{

for(j=value[i];j<=money;j++)

{

dp[j]=max(dp[j],dp[j-value[i]]+value[i]);

}

}

printf("%d\n",money-dp[money]);

}

return 0;

}

**一、01背包**

有N件物品和一个容量为V的背包。第i件物品的价格（即体积，下同）是w[i]，价值是c[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

这是最基础的背包问题，总的来说就是：选还是不选，这是个问题<(￣ˇ￣)/

相当于用f[i][j]表示前i个背包装入容量为v的背包中所可以获得的最大价值。

对于一个物品，只有两种情况

　　情况一: 第i件不放进去，这时所得价值为:f[i-1][v]

　　情况二: 第i件放进去，这时所得价值为：f[i-1][v-c[i]]+w[i]

状态转移方程为：f[i][v] = max(f[i-1][v], f[i-1][v-w[i]]+c[i])

一道裸01背包题↓\_↓

采药

题目描述 Description

辰辰是个天资聪颖的孩子，他的梦想是成为世界上最伟大的医师。为此，他想拜附近最有威望的医师为师。医师为了判断他的资质，给他出了一个难题。医师把他带到一个到处都是草药的山洞里对他说：“孩子，这个山洞里有一些不同的草药，采每一株都需要一些时间，每一株也有它自身的价值。我会给你一段时间，在这段时间里，你可以采到一些草药。如果你是一个聪明的孩子，你应该可以让采到的草药的总价值最大。”   
  
如果你是辰辰，你能完成这个任务吗？

输入描述 Input Description

输入第一行有两个整数T（1<=T<=1000）和M（1<=M<=100），用一个空格隔开，T代表总共能够用来采药的时间，M代表山洞里的草药的数目。接下来的M行每行包括两个在1到100之间（包括1和100）的整数，分别表示采摘某株草药的时间和这株草药的价值。

输出描述 Output Description

输出包括一行，这一行只包含一个整数，表示在规定的时间内，可以采到的草药的最大总价值。

样例输入 Sample Input

70 3  
  
71 100  
  
69 1  
  
1 2

样例输出 Sample Output

3

数据范围及提示 Data Size & Hint

【数据规模】  
  
对于30%的数据，M<=10；  
  
对于全部的数据，M<=100。

#include<stdio.h>

#include<algorithm>

using namespace std;

int f[1001][1001];

int main()

{

int T, n,c[10001], v[10001];

scanf("%d%d", &T, &n);

for(int i = 1; i <= n; i++)

scanf("%d%d", &v[i], &c[i]);

for(int i = 1; i <= n; i++)

{

for(int j = 0; j <= T; j++)

f[i][j] = f[i-1][j];

for(int j = 0; j+v[i] <= T; j++)

f[i][j] = max(f[i][j] + c[i], f[i-1][j+v[i]]);

}

int ans = 0;

for(int i = 0; i <= T; i++)

ans = max(ans, f[n][i]);

printf("%d", ans);

return 0;

}

**二、完全背包**

有N种物品和一个容量为V的背包，每种物品都有无限件可用。第i种物品的费用是w[i]，价值是c[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

完全背包和01背包十分相像， 区别就是完全背包物品有无限件。由之前的选或者不选转变成了选或者不选，选几件。√

和01背包一样，我们可以写出状态转移方程：f[i][v]=max(f[i-1][v-k\*c[i]]+k\*w[i]|0<=k\*c[i]<=v)

还有一个简单的优化↓\_↓

当一个物品的价值小于另一个物品的价值，但是价格高于另一个物品，我们就可以不去考虑这个物品。即若两件物品i、j满足c[i]<=c[j]且w[i]>=w[j]，则将物品j去掉，不用考虑。我们为什么要买一个又贵又难吃的东西呢(╯▽╰)

#include<stdio.h>

#include<algorithm>

using namespace std;

const int maxm=2001,maxn=101;

int n,m,v,i;

int c[maxn],w[maxn];

int f[maxm];

int main()

{

scanf("%d%d",&m,&n); //背包容量m和物品数量n

for(i=1;i<=n;i++)

scanf("%d%d",&w[i],&c[i]);

for(i=1;i<=n;i++)

for(v=w[i]; v<=m; v++) //设 f[v]表示重量不超过v公斤的最大价值

//这里是v++ 顺序 区别于01背包

f[v]=max(f[v-w[i]]+c[i], f[v]);

printf("%d\n", f[m]); // f[m]为最优解

return 0;

}

**三、多重背包**

有N种物品和一个容量为V的背包。第i种物品最多有n[i]件可用，每件费用是w[i]，价值是c[i]。求解将哪些物品装入背包可使这些物品的费用总和不超过背包容量，且价值总和最大。

这里又多了一个限制条件，每个物品规定了可用的次数。

同理，我们可以得出状态转移方程：f[i][v]=max(f[i-1][v-k\*w[i]]+ k\*c[i]|0<=k<=n[i])

一道例题↓\_↓

庆功会

【问题描述】

为了庆贺班级在校运动会上取得全校第一名成绩，班主任决定开一场庆功会，为此拨款购买奖品犒劳运动员。期望拨款金额能购买最大价值的奖品，可以补充他们的精力和体力。

【输入格式】

第一行二个数n(n<=500)，m(m<=6000)，其中n代表希望购买的奖品的种数，m表示拨款金额。 接下来n行，每行3个数，v、w、s，分别表示第I种奖品的价格、价值（价格与价值是不同的概念）和购买的数量（买0件到s件均可），其中v<=100，w<=1000，s<=10。

【输出格式】

第一行：一个数，表示此次购买能获得的最大的价值（注意！不是价格）。

【输入样例】

5 1000

80 20 4

40 50 9

30 50 7

40 30 6

20 20 1

【输出样例】

1040

#include<stdio.h>

#include<algorithm>

using namespace std;

int v[10001],w[10001];

int f[6001];

int n,m,n1;

int main()

{

scanf("%d%d",&n,&m);

for(int i=1;i<=n;i++)

{

int x,y,s,t=1;

scanf("%d%d%d",&x,&y,&s);

while (s>=t)

{

v[++n1]=x\*t;

w[n1]=y\*t;

s-=t;

t\*=2;

}

v[++n1]=x\*s;

w[n1]=y\*s; //把s以2的指数分堆：1，2，4，…，2^(k-1)，s-2^k+1,

}

for(int i=1;i<=n1;i++)

for(int j=m;j>=v[i];j--)

f[j]=max(f[j],f[j-v[i]]+w[i]);

printf("%d\n",f[m]);

return 0;

}

strtol 函数：

它的功能是将一个任意1-36进制数转化为10进制数，返回是long int型。

函数为long int strtol(const char \*nptr, char \*\*endptr, int base)

base是要转化的数的进制，非法字符会赋值给endptr，nptr是要转化的字符，例如：

char buffer[20]="10379cend$3";

char \*stop;

printf("%d\n",strtol(buffer, &stop, 8));

printf("%s\n", stop);

输出结果：

543

9cend$3

将一个8进制转化为10进制，读取1037，其他后面的为非法字符，转化结果以int型输出来。

另外，如果base为0，且字符串不是以0x(或者0X)开头，则按十进制进行转化。如果base为0或者16，并且字符串以0x（或者0X）开头，那么，x（或者X）被忽略，字符串按16进制转化。如果base不等于0和16，并且字符串以0x(或者0X)开头，那么x被视为非法字符。

最后，需要说明的是，对于nptr指向的字符串，其开头和结尾处的空格被忽视，字符串中间的空格被视为非法字符。

itoa函数：

它的功能是将一个10进制的数转化为n进制的值、其返回值为char型。（和上面的strtol效果相反）

例如：itoa(num, str, 2); num是一个int型的，是要转化的10进制数，str是转化结果，后面的值为目标进制。

#include<cstdlib>

#include<cstdio>

int main()

{

int num = 10;

char str[100];

itoa(num, str, 2);

printf("%s\n", str);

return 0;

}

输出结果：

1010

除了上面两个，c++中还有一些定向的转换：

std::bitset（转2进制），std::oct（转8进制），std::dec （转10进制），std::hex（转16进制）。

#include <bitset>

#include <iostream>

using namespace std;

int main()

{

cout << "36的8进制:" << std::oct << 36 << endl;

cout << "36的10进制" << std::dec << 36 << endl;

cout << "36的16进制:" << std::hex << 36 << endl;

cout << "36的2进制: " << bitset<8>(36) << endl;

return 0;

}

（这里bitset后面尖括号里代表输出的位数）

C++输入外挂

std::ios::sync\_with\_stdio(false);

cin.tie(0);

矩阵快速乘法

具体该怎么实现两个矩阵相乘呢?

一般会用O(n^3)的方法。。。配合剪枝【添条件，设门槛。。。】，如下：其实主要就是函数 MATRIX mat\_multiply (MATRIX a , MATRIX b , int n);

const int MOD=10000;

struct mat

{

int a[2][2];//这里数据范围就用小的示范

};

mat mat\_mul(mat x,mat y)//实现两个矩阵相乘，返回的还是一个矩阵。

{

mat res;//用来表示得到的新的矩阵；

memset(res.a,0,sizeof(res.a));

for(int i=0;i<2;i++)

for(int j=0;j<2;j++)

for(int k=0;k<2;k++)

{

res.a[i][j]+=x.a[i][k]\*y.a[k][j];

res.a[i][j]%=MOD;//这一步看题目具体需要了

}

return res;

}

下面来实现一个矩阵快速幂：

int pow(int n)//还是小范围数据来说吧,要不然返回值的类型自己定义

{

mat c,res;

memset(res.a,0,sizeof(res.a));

c.a[0][0]=1;//给矩阵赋初值

c.a[0][1]=1;

c.a[1][0]=1;

c.a[1][1]=0;

for(int i=0;i<n;i++) res.a[i][i]=1;//单位矩阵；

while(n)

{

if(n&1) res=mat\_mul(res,c);//这里看就要用到上面的矩阵相乘了；

c=mat\_mul(c,c);

n=n>>1;

}

return res.a[0][1];

}//时间复杂度log(n)

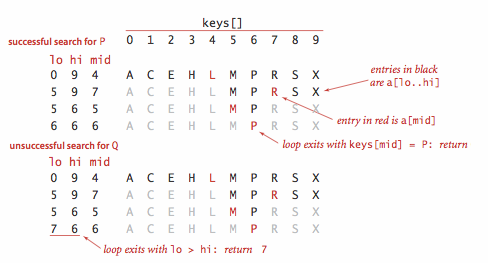
**对于矩阵乘法与递推式之间的关系：**

如：在斐波那契数列之中

f[i] = 1\*f[i-1]+1\*f[i-2]  f[i-1] = 1\*f[i-1] + 0\*f[i-2];

# **1 二分查找**

　　二分查找是一个基础的算法，也是面试中常考的一个知识点。二分查找就是将查找的键和子数组的中间键作比较，如果被查找的键小于中间键，就在左子数组继续查找；如果大于中间键，就在右子数组中查找，否则中间键就是要找的元素。



（图片来自《算法-第4版》）

[IMG_257](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

/\*\*

\* 二分查找，找到该值在数组中的下标，否则为-1

\*/static int binarySerach(int[] array, int key) {

int left = 0;

int right = array.length - 1;

// 这里必须是 <=

while (left <= right) {

int mid = (left + right) / 2;

if (array[mid] == key) {

return mid;

}

else if (array[mid] < key) {

left = mid + 1;

}

else {

right = mid - 1;

}

}

return -1;

}

[IMG_258](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

　　每次移动left和right指针的时候，需要在mid的基础上+1或者-1， 防止出现死循环， 程序也就能够正确的运行。

　　注意：代码中的判断条件必须是while (left <= right)，否则的话判断条件不完整，比如：array[3] = {1, 3, 5};待查找的键为5，此时在(low < high)条件下就会找不到，因为low和high相等时，指向元素5，但是此时条件不成立，没有进入while()中。

# **2 二分查找的变种**

　　关于二分查找，如果条件稍微变换一下，比如：数组之中的数据可能可以重复，要求返回匹配的数据的最小（或最大）的下标；更近一步， 需要找出数组中第一个大于key的元素（也就是最小的大于key的元素的）下标，等等。 这些，虽然只有一点点的变化，实现的时候确实要更加的细心。

　　二分查找的变种和二分查找原理一样，主要就是变换判断条件（也就是边界条件），如果想直接看如何记忆这些变种的窍门，请直接翻到本文最后。下面来看几种二分查找变种的代码：

## **2.1 查找第一个与key相等的元素**

　　查找第一个相等的元素，也就是说等于查找key值的元素有好多个，返回这些元素最左边的元素下标。

[IMG_259](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

// 查找第一个相等的元素static int findFirstEqual(int[] array, int key) {

int left = 0;

int right = array.length - 1;

// 这里必须是 <=

while (left <= right) {

int mid = (left + right) / 2;

if (array[mid] >= key) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

if (left < array.length && array[left] == key) {

return left;

}

return -1;

}

[IMG_260](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

## **2.2 查找最后一个与key相等的元素**

　　查找最后一个相等的元素，也就是说等于查找key值的元素有好多个，返回这些元素最右边的元素下标。

[IMG_261](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

// 查找最后一个相等的元素static int findLastEqual(int[] array, int key) {

int left = 0;

int right = array.length - 1;

// 这里必须是 <=

while (left <= right) {

int mid = (left + right) / 2;

if (array[mid] <= key) {

left = mid + 1;

}

else {

right = mid - 1;

}

}

if (right >= 0 && array[right] == key) {

return right;

}

return -1;

}

[IMG_262](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

## **2.3 查找最后一个等于或者小于key的元素**

　　查找最后一个等于或者小于key的元素，也就是说等于查找key值的元素有好多个，返回这些元素最右边的元素下标；如果没有等于key值的元素，则返回小于key的最右边元素下标。

[IMG_263](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

// 查找最后一个等于或者小于key的元素static int findLastEqualSmaller(int[] array, int key) {

int left = 0;

int right = array.length - 1;

// 这里必须是 <=

while (left <= right) {

int mid = (left + right) / 2;

if (array[mid] > key) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

return right;

}

[IMG_264](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

## **2.4 查找最后一个小于key的元素**

　　查找最后一个小于key的元素，也就是说返回小于key的最右边元素下标。

[IMG_265](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

// 查找最后一个小于key的元素static int findLastSmaller(int[] array, int key) {

int left = 0;

int right = array.length - 1;

// 这里必须是 <=

while (left <= right) {

int mid = (left + right) / 2;

if (array[mid] >= key) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

return right;

}

[IMG_266](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

## **2.5 查找第一个等于或者大于key的元素**

　　查找第一个等于或者大于key的元素，也就是说等于查找key值的元素有好多个，返回这些元素最左边的元素下标；如果没有等于key值的元素，则返回大于key的最左边元素下标。

[IMG_267](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

// 查找第一个等于或者大于key的元素static int findFirstEqualLarger(int[] array, int key) {

int left = 0;

int right = array.length - 1;

// 这里必须是 <=

while (left <= right) {

int mid = (left + right) / 2;

if (array[mid] >= key) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

return left;

}

[IMG_268](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

## **2.6 查找第一个大于key的元素**

　　查找第一个等于key的元素，也就是说返回大于key的最左边元素下标。

[IMG_269](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

// 查找第一个大于key的元素static int findFirstLarger(int[] array, int key) {

int left = 0;

int right = array.length - 1;

// 这里必须是 <=

while (left <= right) {

int mid = (left + right) / 2;

if (array[mid] > key) {

right = mid - 1;

}

else {

left = mid + 1;

}

}

return left;

}

[IMG_270](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

# **3 二分查找变种总结**

[IMG_271](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

// 这里必须是 <=while (left <= right) {

int mid = (left + right) / 2;

if (array[mid] ? key) {

//... right = mid - 1; }

else {

// ... left = mid + 1; }

}return xxx;

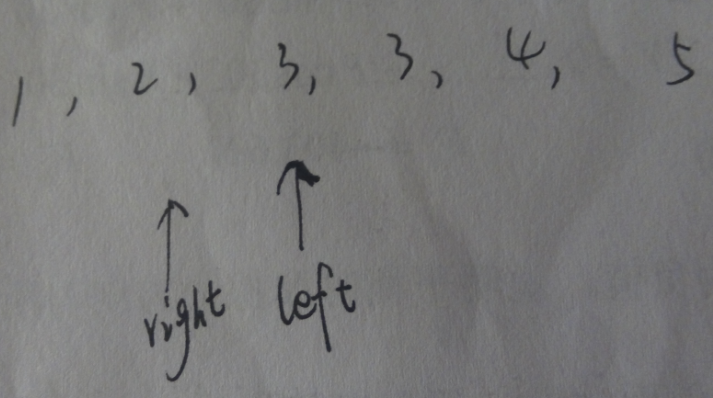
[IMG_272](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

　　二分查找变种较多，不过它们的“套路”是一样的，以上代码就是其套路，如何快速写出二分查找的代码，只需按照以下步骤即可：

#### ****1 首先判断出是返回left，还是返回right****

　　因为我们知道最后跳出while (left <= right)循环条件是right < left，且right = left - 1。最后right和left一定是卡在"边界值"的左右两边，如果是比较值为key，查找小于等于（或者是小于）key的元素，则边界值就是等于key的所有元素的最左边那个，其实应该返回left。

　　以数组{1, 2, 3, 3, 4, 5}为例，如果需要查找第一个等于或者小于3的元素下标，我们比较的key值是3，则最后left和right需要满足以下条件：



　　我们比较的key值是3，所以此时我们需要返回left。

#### ****2 判断出比较符号****

[IMG_274](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

int mid = (left + right) / 2;if (array[mid] ? key) {

//... right = xxx;}else {

// ... left = xxx;

}

[IMG_275](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

　　也就是这里的 if (array[mid] ? key) 中的判断符号，结合步骤1和给出的条件，如果是查找小于等于key的元素，则知道应该使用判断符号>=，因为是要返回left，所以如果array[mid]等于或者大于key，就应该使用>=，以下是完整代码

[IMG_276](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

// 查找小于等于key的元素int mid = (left + right) / 2;if (array[mid] >= key) {

right = mid - 1;

}else {

left = mid + 1;

}

[IMG_277](https://www.cnblogs.com/luoxn28/p/javascript:void(0);)

大数操作top1模板:

#include<cstdio>

#include<iostream>

#include<cmath>

#include<algorithm>

#include<cstring>

#include<string>

#include<vector>

using namespace std;

struct bigint{ // only positive number;

static const int BASE=100000000;

static const int WIDTH=8;

vector<int> s;

//value

bigint(long long num=0){ \*this = num;}

bigint operator = (long long num){

s.clear();

do{

s.push\_back(num%BASE);

num/=BASE;

}while(num>0);

return \*this;

}

bigint operator = (const string& str){

s.clear() ;

int x,len = (str.length()-1)/WIDTH + 1;

for(int i=0;i<len;i++){

int end = str.length() - i\*WIDTH;

int start = max(0,end - WIDTH);

sscanf(str.substr(start,end-start).c\_str(),"%d",&x);

s.push\_back(x);

}

return \*this;

}

//input&output

friend ostream& operator << (ostream &out, const bigint& x){

out << x.s.back();

for(int i=x.s.size()-2;i>=0;i--){

char buf[20];

sprintf(buf,"%08d",x.s[i]);

for(int j=0;j<strlen(buf);j++) out << buf[j];

}

return out;

}

friend istream& operator >>(istream &in, bigint& x){

string s;

if(!(in>>s)) return in;

x=s;

return in;

}

//compare

bool operator < (const bigint& b) const {

if(s.size()!=b.s.size()) return s.size() < b.s.size();

for(int i=s.size()-1;i>=0;i--) if(s[i]!=b.s[i]) return s[i] < b.s[i];

return false;//equal

}

bool operator > (const bigint& b) const {return b < \*this;}

bool operator <= (const bigint& b) const {return !(b < \*this);}

bool operator >= (const bigint& b) const {return !(\*this < b);}

bool operator != (const bigint& b) const {return b < \*this || \*this < b;}

bool operator == (const bigint& b) const {return !(b < \*this) && !(\*this < b);}

//calculate

bigint operator +(const bigint& b) const {

bigint c;

c.s.clear();

for(int i=0,g=0;;i++){

if(g==0 && i>=s.size() && i>=b.s.size()) break;

int x=g;

if(i<s.size()) x+=s[i];

if(i<b.s.size()) x+=b.s[i];

c.s.push\_back(x%BASE);

g = x/BASE;

}

return c;

}

bigint operator +=(const bigint& b){

\*this = \*this + b;

return \*this;

}

bigint operator -(const bigint& b) const {

bigint c;

c.s.clear();

for(int i=0,g=0;;i++){

if(g==0 && i>=s.size() && i>=b.s.size()) break;

int x=g;

if(i<s.size()) x+=s[i];

if(i<b.s.size()) x-=b.s[i];

x+=BASE;

c.s.push\_back(x%BASE);

g = x/BASE - 1;

}

return c;

}

bigint operator \* (const bigint& b) const {

bigint c;

c.s.clear();

bigint g=0;

for(int i=0;;i++){

if(g.s.size()==0 && i>=s.size()+b.s.size()-1) break;

bigint x;

x.s.clear() ;

for(int j=0;j<g.s.size();j++) x.s.push\_back(g.s[j]);

if(i<s.size()+b.s.size()-1){

for(int j = max(0 , i-(int)s.size()+1);j<=min(i,(int)b.s.size()-1);j++){

bigint t = (long long)b.s[j]\*s[i-j];

x += t;

}

}

c.s.push\_back(x.s[0]);

g.s.clear();

if(x.s.size()>1) for(int j=1;j<x.s.size();j++) g.s.push\_back(x.s[j]);

}

return c;

}

};

int main(){

bigint a,b,c,ji,g;

cin>>a;

b=1;

c=0;

g=3;

int t=0;

while(c<a){

t++;

c=c+b;

b=b\*3;

}

cout<<t<<endl;

}

求一个数的全部因子：

**int** i=2;

**while**(i<**sqrt**(maxn)+1){

**if**(maxn%i==0){

                su.insert(i);

                su.insert(maxn/i);

            }

            i++;

        }

        su.insert(maxn);

求一个区间含有6的数字个数：

emmmm这也算是个经典数学问题了

以前做过一个类似的题目，先从那个题目讲起可能比较好理解

原题是：从1开始的所有自然数1, 2, 3, 4,… 中，删掉含有数字2,3,5,7的数，变成1,4,6,8,9,10,11,14,16,18,19,30…

问新序列中的第k个数是多少

首先我们要把0,1,2,…,9这些数字看成符号，因为这里有10个符号，所以叫作十进制数，这10个符号替换成其他的任何符号，比如将用！替换1，@替换2，仍然不影响十进制的使用，此时1234就会写成！@34

现在要将其中的4个符号删掉，就相当于原来的10进制数变成了6进制数，删掉2之后，原来的符号3就称为了第二个符号，这个符号就不再代表我们思想中理解的三（他不再是第三个符号），而是我们理解中的二（他是第2个符号）。

十进制中的第17个数，就是把17表示成十进制（尽管现在他就是十进制数）后得到的结果

那么六进制中的第17个数，就是把17表示成6进制后得到的结果，即25

如果还不懂的话可以看看我以前写的博客Sichuan University Programming Contest 2018 Preliminary的D题

然后回到我们这个题，题目想问[l，r]中有多少个含有6的数字

利用前缀和的思想容易想到，就是求[0，r]中含有6的数字个数减去[0，l-1]中含有6的数字个数

那么怎么求含有6的数字个数呢？我们可以先求出有多少个不含有6的数字个数。

不含有6的数字个数怎么求呢？利用刚才那题的思想，其实就是将其理解为九进制，再将这个理解为的九进制数转换为十进制，就是其不含6的数字个数

比方说123，把它当成九进制数来理解，再转换为十进制数，即

这说明123中不含有6的数字个数为102个，反过来说，含有6的数字个数就是123-102=21个

类似的，如果是19，把它当成九进制数来理解，转换为十进制数。这里要注意的是，我们把它当做九进制数来理解的原因是我们把符号6删去了，导致有10个符号的十进制数变成了九进制数，所以这里19中的9不再是第九个符号，而是第8个符号

(19)伪9进制=1×9+8=17 (19)\_{伪9进制}=1 \times 9 + 8 = 17

(19)

伪9进制

​

=1×9+8=17

也就是说19中不含6的数字个数为17个，含有6的数字个数为2

但是还有一个问题，既然我们说理解为九进制是把6删了，那么如果这个数本身就有6怎么办呢？

这时候就要做一些处理了。比方说，16432中不含有6的数字个数，其实跟15999中不含有6的数字个数是相同的，对吧，那么现在15999就是一个不含有6的数字了。同理，如果有多个6，那么我们应该取最高位的6变成5，其后各位数全部变成9，比如1654625中不含6的数字个数和1599999相同

到这里解释就差不多了，可能对于相似的方法有更容易理解的解释，但是计算方法上基本是换汤不换药的

细节比较多，所以代码也可能不那么好懂

#include<bits/stdc++.h>

using namespace std;

long long f(long long x) {

long long p = 1, q = 1;

long long tmp = x;

tmp /= 10;

while(tmp) tmp /= 10, p \*= 10, q \*= 9;

long long ans = 0;

bool flag = false;

while(p) {

int t = x/p; x %= p;

if(!flag && t == 6) {

t = 5;

flag = true;

}

else {

if(flag) t = 8;

else if(t > 6) t--;

}

ans += t\*q;

p /= 10; q /= 9;

}

return ans;

}

long long count(long long n) {

if(n < 6) return 0;

return n - f(n);

}

int main() {

long long l, r; cin>>l>>r;

cout<<count(r) - count(l-1)<<endl;

return 0;

}

1.**常规方法**，先判断整体是否回文，若整体回文，可以在中间加一个数，直接返回YES。如果整体不是回文，依次去掉一个字符后判断剩下的字符串是否为回文串，时间复杂度O(n^2)。如abcddecba：1.去掉a，判断bcddecba；2.去掉b，判断acddecba；3.去掉c，判断abddecba；等等。一直进行下去，若发现回文即可停止返回YES，反之如果一直没发现回文串，返回NO。

2.**创意解法**，先找到不匹配的位置，然后提取出中间不匹配的的字符串，分别判断其去掉头和去掉尾的两个字符串是否为回文，其中一个为回文即可。时间复杂度O(n)。如abcddecba，取出dde，然后判断dd和de，发现其中有一个满足回文，即返回YES。反之返回NO。

3.**扩展解法**，先把原字符串逆序，然后计算两字符串的最长公共子序列长度，最后diff=字符串长度-最长公共子序列长度，diff即为如果可以形成回文串，原字符串需要添加的字符个数。用到这个题目里，如果diff<=1，即可。时间复杂度O(n^2)。**可以看到，这种解法可以解决一类问题：添加几个字符可以构成回文串？**

#### **创意解法**

#include <iostream>

#include <string>

#include <algorithm> //std::reverse

using namespace std;

//判断回文方法1

//bool IsHuiWen(string str)

//{

// string rstr=str;

// reverse(rstr.begin(),rstr.end());

// return str==rstr;

//}

//判断回文方法2

bool IsHuiWen(string str)

{

for(int i = 0; i<(int)str.size()/2;i++){

if(str[i] != str[str.size()-1-i])

return false;

}

return true;

}

int main()

{

string str;

while(cin>>str)

{

bool flag = false;

int i;

for(i = 0; i<(int)str.size()/2;i++)

{

if(str[i] != str[str.size()-1-i])

break;;

}

if(i == (int)str.size()/2)

flag = true;

else

flag = (IsHuiWen(str.substr(i+1,str.size()-2\*i-1)) || IsHuiWen(str.substr(i,str.size()-2\*i-1)));

cout<< ((flag==true) ? "YES" : "NO")<<endl;

};

return 0;

}

#### **扩展解法**

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

using namespace std;

int findLCS(string str1,string str2) {

// write code here

if(0 == str1.size() || 0 == str2.size())

return 0;

vector<vector<int> > array(str1.size()+1,vector<int>(str2.size()+1,0));

for(size\_t i=1; i<= str1.size(); ++i){//注意：是小于等于

for(size\_t j=1; j<= str2.size(); ++j){//注意：是小于等于

if(str1[i-1] == str2[j-1]){//前面填充了一行一列，因此判断i-1和j-1

array[i][j]=array[i-1][j-1]+1;

}

else

array[i][j]=max(array[i-1][j],array[i][j-1]);

}

}

return array[str1.size()][str2.size()];

}

int main(){

string str;

while(cin>>str){

string rstr=str;

reverse(rstr.begin(),rstr.end());

int diff=(int)str.size()-findLCS(str,rstr);

cout<<(diff<2?"YES":"NO")<<endl;

}

return 0;

}

广搜走迷宫拓展（优先队列）

精通程序设计的 Applese 双写了一个游戏。  
  
在这个游戏中，它被困在了一个 n×mn×m 的迷宫中，它想要逃出这个迷宫。  
  
在迷宫中，有一些方格是水池，只有当 Applese 处于水属性的时候才可以通过；有一些方格是岩浆，只有当 Applese 是火属性的时候可以通过；有一些方格是墙壁，无论如何都无法通过；另一些格子是空地（包括起点和终点），可以自由通过。

在一些空地上有神秘道具可以让 Applese 转换自己的属性（从水属性变为火属性或从火属性变为水属性，需要一个单位的时间）。

已知 Applese 在一个单位的时间内可以朝四个方向行走一格，且开始处于水属性，位于空地的道具拾取后只能在该处立即使用（或者不使用），且可以多次使用。求它走出迷宫需要的最少时间。

## **输入描述:**

第一行两个正整数 n, m 表示迷宫的大小。  
接下来 n 行，每行长度为 m 的字符串。描述地图。  
其中 'S' 表示起点，'T' 表示终点，'.' 表示空地，'w'表示岩浆，'~'表示水池，'@' 表示道具，'#'表示障碍。  
保证地图中的起点和终点只有一个，道具都位于空地。

## **输出描述:**

输出一个整数，表示 Applese 走出迷宫的最短时间。特别地，如果 Applese 走不出迷宫，输出 "-1"。

5 5

.w@..

.S#..

~w#..

.w..~

@w.~T

18

#include <bits/stdc++.h>

using namespace std;

const int maxn=105;

int dir[4][2]={{0,1},{0,-1},{1,0},{-1,0}};

int n,m;

char ma[maxn][maxn];

struct point{

int x,y;

};

bool visit[maxn][maxn][2];

struct node{

int x,y,state,step;

friend bool operator<(node a,node b)

{

return a.step>b.step;

}

};

bool flag=true;

void bfs(point st,point en){

node init;

init.x=st.x,init.y=st.y;

init.state=0,init.step=0;

visit[init.x][init.y][0]=1;

priority\_queue<node> que;

que.push(init);

while(!que.empty()){

node temp=que.top();

que.pop();

if(temp.x==en.x&&temp.y==en.y){

printf("%d\n",temp.step);

flag=false;

break;

}

for(int i=0;i<4;i++){

node temp1;

temp1.x=temp.x+dir[i][0];

temp1.y=temp.y+dir[i][1];

temp1.step=temp.step+1;

if(temp1.x>=0&&temp1.y>=0&&temp1.x<n&&temp1.y<m&&ma[temp1.x][temp1.y]!='#'){

if(ma[temp1.x][temp1.y]=='.'&&!visit[temp1.x][temp1.y][temp.state]){

temp1.state=temp.state;

que.push(temp1);

visit[temp1.x][temp1.y][temp.state]=1;

}

else if(ma[temp1.x][temp1.y]=='~'&&temp.state==0&&!visit[temp1.x][temp1.y][temp.state]){

temp1.state=temp.state;

que.push(temp1);

visit[temp1.x][temp1.y][temp.state]=1;

}

else if(ma[temp1.x][temp1.y]=='w'&&temp.state==1&&!visit[temp1.x][temp1.y][temp.state]){

temp1.state=temp.state;

que.push(temp1);

visit[temp1.x][temp1.y][temp.state]=1;

}

else if(ma[temp1.x][temp1.y]=='@'){

temp1.state=temp.state;

if(!visit[temp1.x][temp1.y][temp1.state]){

que.push(temp1);

visit[temp1.x][temp1.y][temp1.state]=1;

}

temp1.step=temp.step+2;

temp1.state=temp1.state^1;

if(!visit[temp1.x][temp1.y][temp1.state]){

que.push(temp1);

visit[temp1.x][temp1.y][temp1.state]=1;

}

}

}

}

}

return ;

}

int main() {

scanf("%d %d",&n,&m);

getchar();

for(int i=0;i<n;i++){

gets(ma[i]);

}

struct point st,en;

for(int i=0;i<n;i++){

for(int j=0;j<m;j++){

if(ma[i][j]=='S'){

st.x=i;

st.y=j;

ma[i][j]='.';

}

if(ma[i][j]=='T'){

en.x=i;

en.y=j;

ma[i][j]='.';

}

}

}

memset(visit,0,sizeof(visit));

bfs(st,en);

if(flag){

printf("-1\n");

}

return 0;

}

链接：[https://ac.nowcoder.com/acm/contest/332/A](https://ac.nowcoder.com/acm/contest/332/A" \t "_blank)  
来源：牛客网

## 题目描述

小B准备出模拟赛。  
她把题目按难度分为四等，分值分别为6,7,8,9。  
已知小B共出了m道题，共n分。  
求小B最少出了多少道6分题。

## **输入描述:**

两个正整数n,m

## **输出描述:**

一个数，表示答案。  
若无解，输出"jgzjgzjgz"。

这不是个小学数学题嘛？？？

把n划分成m个数，只能是6,7,8,9，那么显然当n<6m或者n>9m时不可能，否则一定可能

既然问最少出多少个6，那我们就尽量把6放少一点，最好是一个都不放

嗯，什么情况下可以一个6都不放呢？那就是n>=7m时啊（仍满足于n<=9m），这种情况我就直接在7,8,9中选就可以了

嗯，再讨论必须放6的情况，也就是说6m<=n<7m的情况。通过这个式子可以看出，这m个数的平均数在6到7之间，那么我一定有一种策略可以只用6和7（即不用8和9）凑出来。

如果想不通为什么一定可以只使用6和7，你可以想想，如果有一个8，已知最少有一个6，那么我们可以去掉6和8，换成2个7。如果有1个9，已知最少有1个6，那么可以去掉6和9，换成7和8，此时如果还有6，就可以再把6和8换成2个7，如果没有6了，就不符合我们的假设

嗯上面那段话感觉是废话

所以啊，m个数，只使用6和7，凑出一个n来，那里面有多少个6嘛？假设m个数全部是7，那么7m比n多出来的，就是6的个数。即6的个数为7m-n

嗯开long long，我不开long long挂了一发（我不会说我做这题之前还提醒了队友要开，结果自己忘了）

#include<bits/stdc++.h>

using namespace std;

int main() {

long long n, m; scanf("%lld%lld", &n, &m);

if(6\*m > n || 9\*m < n)

printf("jgzjgzjgz\n");

else {

if(n/m > 6) printf("0\n");

else printf("%lld\n", m\*7-n);

}

return 0;

}

A double blessing has descended upon Tom           一个双重的祝福已降临汤姆

alphabetical order                      字典序

alternately rise and fall                      交替上升和下降

ascending order                                                    升序

Assume            假设，认为

axis    axes                                                            轴

calculate         计算

clockwise order                      顺时针方向顺序

comma                          逗号

consistof             由....组成

contain              包含

coordinates                          坐标

corrupt                          腐烂，破坏

cubes                          立方

D is rounded to 2 decimal places                   D是精确到小数点后2位

Diagonal                           对角

dial                          钟面，拨打

digit                          位数  数字；手指；足趾

distinctvalues                                                        独一无二的值

Education is where it all begins               教育是在这一切开始

equation                        方程式 等式

followby             跟随，其后

horizontal or vertical direction                水平和垂直方向

I don't have anything planned                  我没有什么计划

I don't have the foggiest idea                  我是一点儿概念都没有

in the range of                    在....范围内

in the shape of a cross                   十字形

integer             整数

intersect                          相交

it may not execute in the allottedtime                    这可能不是在规定的时间段内执行

loop                               环

meadow                          草坪

matrix                          矩阵

minimalvolume                                                    最小体积

negative ，positive              负   ，正

odd and even                      奇和偶

parity property                      奇偶性

positive and negative integers                     正整数和负整数

present Error                      呈现错误

proceed                          运行

process a sequence of n distinct integers             处理一串n 个截然不同的整数

Rows and columns                                                行与列

runtime error:运行时错误                                                                        一般都是非法访问内存（数组越界、访问空指针、堆栈溢出）、                                                做除法时除以了0.

sequence                   一连串，有关联的一组事物

series              连续的同类事物，系列

specify                           指定

the course material                      课程内容

There are no special punctuation symbols or spacingrules        没有特殊标点符号或间距的规则

two adjacent sequence elements                           两个相邻的元素串

two-dimensional array                      二维数组

unique identifier                         唯一的标识符

vertex                          顶点

where did your drive come from?          你的动力是来自哪里呢？

wooden planks                         木板

duplicated 复制；打印的