01包

int c[1100],w[1100],dp[1100],V;

//c[]：物品所占容量;w[]物品的价值;V为背包容量

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

for(int i = 0; i < N; ++i)//第i件物品

{

for(int j = V; j >= c[i]; j--)//填满空间j

{

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

}

}

完全包

int c[1100],w[1100],dp[1100],V;

//c[]：物品所占容量;w[]物品的价值;V为背包容量

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

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

{

for(int j = c[i]; j <= V; j++)

{

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

}

}

最长公共子序列

char s1[220],s2[220];

int dp[220][220];

//求串s1和串s2的公共子序列

int lcs(char \*s1,char \*s2)

{

int len1 = strlen(s1);

int len2 = strlen(s2);

for(int i = 0; i <= len1; ++i)

{

for(int j = 0; j <= len2; ++j)

{

if(i == 0 || j == 0)

dp[i][j] = 0;

else if(s1[i-1] == s2[j-1])

dp[i][j] = dp[i-1][j-1] + 1;

else

dp[i][j] = max(dp[i-1][j],dp[i][j-1]);

}

}

return dp[len1][len2];

}

最长递增子序列

int Arr[10010],List[10010]; int Stack[10010] ;int LIS(int \*Arr,int N)

{

int top = 0; Stack[top] = -1;

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

{

if(Arr[i] > Stack[top])

Stack[++top] = Arr[i];

else

{

int low = 1; int high = top;

while(low <= high)

{

int mid = (low + high)/2;

if(Arr[i] > Stack[mid]) low = mid + 1;

else high = mid - 1;

}

Stack[low] = Arr[i];

}

}

return top;

}

最大连续序列和

int a[110000],N,pos1,pos2,Start,End;//Start、End存储最大连续子序列的起点和终点

int MaxSubSum(int \*a)

{

int MaxSum = a[0],Sum = a[0]; pos1 = pos2 = Start = End = 0;

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

{

Sum += a[i]; if(Sum < a[i])

{

Sum = a[i]; pos1 = i; pos2 = i;

}

else

{

pos2 = i;

}

if(MaxSum < Sum)

{

MaxSum = Sum; Start = pos1; End = pos2;

} }

return MaxSum;

}

切钢条

#include<stdio.h>

#include<algorithm>

using namespace std;

const int INF = 0xffffff0;

int p[110],r[110];//r[n]来保存子问题

int BOTTOM\_UP\_CUT\_ROD(int n)

{

r[0] = 0;//长度为0的钢条没有收益

for(int j = 1; j <= n; j++)//对j=1,2,3,…,n按升序求解每个规模为j的子问题。

{

int q = -INF;

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

{

q = max(q,p[i]+r[j-i]);//直接访问数组r[j-i]来获得规模为j-i的子问题的解

}

r[j] = q;

}

return r[n];

}

int main()

{

int N;

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

{

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

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

int ans = BOTTOM\_UP\_CUT\_ROD(N);

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

}

return 0;

}

gcd

int gcd(int a,int b)

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

}

过山车（匈牙利）

#include<stdio.h>

#include<string.h>

int map[505][505]; //记录是否可以匹配，0表示不能，1表示能

int vis[505]; //用在不同队伍匹配是的标记作用 int pri[505];

int k,m,n; int find(int x)

{ int i;

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

{

if(vis[i]==0&&map[i][x])

{

vis[i]=1;

if(pri[i]==-1||find(pri[i]))

pri[i]=x; return 1;

} }

return 0;

}

int main()

{

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

{

if(k==0) break;

scanf ("%d%d", &m, &n); memset(map, 0, sizeof (map)); memset(pri, -1, sizeof (pri));

int i;

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

{

int a,b;

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

map[a][b]=1;

}

int output=0;

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

{

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

if(find(i))

output++;

}

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

} return 0;

}

块

#include<stdio.h>

#include<math.h>

int a[200005],belong[200005],L[200005],R[200005],ans[200005];

int min(int a,int b){

return a>b?b:a;

}

int max(int a,int b){

return a>b?a:b;

}

int main(){

int n,m,i,j;

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

int len=sqrt(n);

int num=n/len;

if(n%len!=0)num++;

for(i=1;i<=num;i++)ans[i]=0;

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

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

belong[i]=i/len;

if(i%len!=0)belong[i]++;

ans[belong[i]]=max(ans[belong[i]],a[i]);

}

for(i=1;i<=num;i++){

L[i]=R[i-1]+1;

R[i]=min(L[i]+len-1,n);

}

while(m--){

char s[2];

scanf("%s",s);

int x,y;

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

if(s[0]=='U'){

a[x]=y;

int bel=belong[x];

ans[bel]=0;

for(i=L[bel];i<=R[bel];i++){

ans[bel]=max(ans[bel],a[i]);

}

}

else{

int anss=0;

if(belong[x]==belong[y]){

for(i=x;i<=y;i++)anss=max(anss,a[i]);

}

else{

for(i=x;i<=R[belong[x]];i++)anss=max(anss,a[i]);

for(i=L[belong[y]];i<=y;i++)anss=max(anss,a[i]);

for(i=belong[x]+1;i<belong[y];i++)anss=max(anss,ans[i]);

}

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

} } } return 0; }

快速幂

long long int tt(int a,int b)

{

long long int r=1,c=a;

while(b)

{

if(b&1) r\*=c;//r=r\*c%X 如果有取余

c\*=c;//c=c\*c%X 如果有取余

b>>=1;

}

return r;

}

三分

#include<stdio.h>

#include<math.h>

double H,h,D;

double eps=0.000000001;

double lmid ; double rmid; double low,high;

double solve(double x)

{

double y=D\*(h-x)/(H-x)+x; return y;//表达式

}

double make() //解决模板

{

while(high-low>eps)

{

lmid=(low\*2+high)/3; rmid=(low+high\*2)/3;

if(solve(lmid)>solve(rmid))

{

high=rmid;

}

else

{

low = lmid;

} }

return solve(lmid); }

int main()

{ …………………………//输入输出

}

母函数

（假设有x1个字母A， x2个字母B,..... x26个字母Z，同时假设字母A的价值为1，字母B的价值为2,..... 字母Z的价值为26。那么，对于给定的字母，可以找到多少价值<=50的单词呢？单词的价值就是组成一个单词的所有字母的价值之和，比如，单词ACM的价值是1+3+14=18，单词HDU的价值是8+4+21=33。(组成的单词与排列顺序无关，比如ACM与CMA认为是同一个单词）。）

int n1[55],n2[55];

void GF(){

int i,j,k;

memset(n1,0,sizeof(n1));

memset(n2,0,sizeof(n2));

n1[0]=1;

for(i=1;i<=26;i++) // i每个东西的单位

{

for(j=0;j<=50;j++) //50为最大的总单位

{

for(k=0;k<=a[i] && i\*k+j<=50;k++) //套用中间条件需换

n2[i\*k+j]+=n1[j];

}

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

{

n1[j]=n2[j];

n2[j]=0;

} } }

i是代码每一个括号里面的多项式，j是代表（i - 1）个多项式乘完后的多项式指数(x ^ j) ，k\*i+j代表本次计算完的指数，根据物品i取k个，就是x^(i\*k) ，乘以之前算过的x^j ，等于 x^(i\*k+j)

即：x^(i\*k) \* x^j = x^(i\*k+j)。

n2数组是本次计算结果，n1数组是i-1个多项式的计算结果，最后把n2赋值给n1（保证循环的进行）

要多少n1就加到多少。

int a[10000000]; 筛法求素数

int isprime()

{

memset(a,1,sizeof(a));

int tmp;

for(int i=2;i<=10000000;i++)

{

if(a[i]!=0)

{tmp=i;

for(int j=2;j<10000000,j\*tmp<=10000000;j++)

{

a[j\*tmp]=0;

}}}

环形涂色 An=(m-1)^n+((-1)^n)\*(m-1) n为区域，m为颜色种类。

筛法求素数

1int sz[1000001];

int judge(int n)

{

int p,q,i,j,tmp,l,sum=0;

if(n==1) return 0;

if(n==2) return 1;

for(i=2;i<=sqrt(n);i++)

{

if(sz[i]==0)

continue;

else for(j=2;j\*i<=n;j++) sz[j\*i]=0;

}

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

{

if(sz[i]!=0)sum++;

}

return sum; }

向上取整

(a-1)/b+1或（a+b-1）/b

Kmp

int next[1000];

int KmpSearch(char\* s, char\* p)

{

int i = 0;

int j = 0;

int sLen = strlen(s);

int pLen = strlen(p);

while (i < sLen && j < pLen)

{

//①如果j = -1，或者当前字符匹配成功（即S[i] == P[j]），都令i++，j++

if (j == -1 || s[i] == p[j])

{

i++;

j++;

}

else

{

//②如果j != -1，且当前字符匹配失败（即S[i] != P[j]），则令 i 不变，j = next[j]

//next[j]即为j所对应的next值

j = next[j];

}

}

if (j == pLen)

return i - j;

else

return -1;

}

void GetNextval(char\* p, int next[])

{

int pLen = strlen(p);

next[0] = -1;

int k = -1;

int j = 0;

while (j < pLen - 1)

{

//p[k]表示前缀，p[j]表示后缀

if (k == -1 || p[j] == p[k])

{

++j;

++k;

//较之前next数组求法，改动在下面4行

if (p[j] != p[k])

next[j] = k; //之前只有这一行

else

//因为不能出现p[j] = p[ next[j ]]，所以当出现时需要继续递归，k = next[k] = next[next[k]]

next[j] = next[k];

}

else

{

k = next[k];

}

}

}

int main()

{

int t;

cin>>t;

char a[1000],b[1000];

while(t--)

{

scanf("%s%s",a,b);

GetNextval(b,next);

int bb=KmpSearch(a,b);

cout<<bb<<endl; }}