## hdoj 1241

### 1 原题目

**油藏**

**时间限制：2000/1000 MS（Java /其他）内存限制：65536/32768 K（Java /其他）  
提交总数：55624接受提交：31890**

**问题描述** <http://acm.hdu.edu.cn/showproblem.php?pid=1241>

GeoSurvComp地质勘测公司负责检测地下油藏。GeoSurvComp一次处理一个大矩形区域的土地，并创建一个将土地分成许多正方形图的网格。然后，它使用传感设备分别分析每个地块，以确定该地块是否包含油。包含油的地块称为矿穴。如果两个凹坑相邻，则它们是同一油藏的一部分。积油可能很大，可能包含许多凹穴。您的工作是确定网格中包含多少种不同的油藏。

**输入项**

输入文件包含一个或多个网格。每个网格均以包含m和n的行开始，该行包含网格中的行数和列数，并用单个空格分隔。如果m = 0，则表示输入结束；否则，输入0。否则为1 <= m <= 100和1 <= n <=100。紧随其后的是m行，每行n个字符（不计算行尾字符）。每个字符对应一个地块，要么是代表无油的“ \*”，要么是代表油囊的“ @”。

**输出量**

对于每个网格，输出不同的油藏数量。如果两个不同的油藏在水平，垂直或对角线上相邻，则它们是同一油藏的一部分。积油最多可容纳100个口袋。

**样本输入**

1 1

\*

3 5

\* @ \* @ \*

\*\* @ \*\*

\* @ \* @ \*

1 8

@@ \*\*\*\* @ \*

5 5

\*\*\*\* @

\* @@ \* @

\* @ \*\* @

@@@ \* @

@@ \*\* @

0 0

**样本输出**

0

1

2

2

### 2 题目分析

给你一个地图，找出油藏的数量（上下左右，对角线相邻都视为同一片油藏）

### 3 基础知识

#### 3.1数据结构

无。

#### 3.2算法

深度优先搜索（解法一用到）

### 4 解法

#### 4.1 解法1（C语言已过）

我的思路是将字符型的地图转换成整型存储起来，1代表油藏，0代表无油，然后遍历整个地图，找到油藏之后在当前位置开始深度优先遍历，将不同部分的油藏分别涂成不同的颜色，就可以计算出不同油藏的数量。

本人代码：

#include <stdio.h>

#include <stdlib.h>

int map[105][105];//地图

int next[8][2]= {-1,0,1,0,0,-1,0,1,-1,-1,-1,1,1,-1,1,1};//下一步如何走（上下左右，左上，右上，左下，右下）

int m,n;

void dfs(int x,int y,int color)

{

int k,xx,yy;

map[x][y] = color;//地图中油藏标记成不同的数字（代表不同的油藏区域）

for(k = 0; k <= 7; k++)//下一步可能走到的位置

{

xx = x + next[k][0];

yy = y + next[k][1];

if(xx >= 0 && xx < m && yy >= 0 && yy < n && map[xx][yy] == 1)//如果是油藏且在地图范围内

{

dfs(xx,yy,color);//继续深度优先遍历

}

}

}

int main()

{

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

{

getchar();//吸收回车

if(m == 0)

break;

int i,j;

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

{

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

{

char temp;

scanf("%c",&temp);//读入地图

if(temp == '@')//将地图转成整型存储

map[i][j] = 1;//1代表油藏

if(temp == '\*')

map[i][j] = 0;//0代表无油

}

getchar();//吸收回车

}

int num = 0;//num记录油藏数量

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

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

if(map[i][j] == 1)//遍历数组找到油藏的位置

{

num--;

dfs(i,j,num);//深度优先遍历

}

printf("%d\n",-num);

}

return 0;

}

## hdoj 2544

### 1 原题目

**最短路**

**Time Limit: 5000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 112804    Accepted Submission(s): 48566**

**Problem Description** <http://acm.hdu.edu.cn/showproblem.php?pid=2544>

在每年的校赛里，所有进入决赛的同学都会获得一件很漂亮的t-shirt。但是每当我们的工作人员把上百件的衣服从商店运回到赛场的时候，却是非常累的！所以现在他们想要寻找最短的从商店到赛场的路线，你可以帮助他们吗？

**Input**

输入包括多组数据。每组数据第一行是两个整数N、M（N<=100，M<=10000），N表示成都的大街上有几个路口，标号为1的路口是商店所在地，标号为N的路口是赛场所在地，M则表示在成都有几条路。N=M=0表示输入结束。接下来M行，每行包括3个整数A，B，C（1<=A,B<=N,1<=C<=1000）,表示在路口A与路口B之间有一条路，我们的工作人员需要C分钟的时间走过这条路。  
输入保证至少存在1条商店到赛场的路线。

**Output**

对于每组输入，输出一行，表示工作人员从商店走到赛场的最短时间

**Sample Input**

2 1

1 2 3

3 3

1 2 5

2 3 5

3 1 2

0 0

**Sample Output**

3

2

### 2 题目分析

给你N个街口，和每两个街口之间走完的时间，让你求解从第一个街口出发到第N个街口的最短时间。

### 3 基础知识

#### 3.1数据结构

无向图（解法一用到）

#### 3.2算法

弗洛伊德算法（解法一用到）

### 4 解法

#### 4.1 解法1（C++已过）

弗洛伊德算法的模板题，没什么好说的。

本人代码：

#include <iostream>

#include <climits>

using namespace std;

long long int map[105][105];//地图的构建

long long int length[105][105];//记录从i到j的距离

void floyd(int n)//弗洛伊德算法

{

int i,j,k;

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

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

length[i][j] = map[i][j];

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

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

for(j = 1; j <= n; j++)//从i经K到j的一条路径更短

if(length[i][k] + length[k][j] < length[i][j])

length[i][j] = length[i][k] + length[k][j];

}

int main()

{

int N,M;

while(cin>>N>>M)

{

if(N == 0 && M == 0)

break;

int i,j;

for(i = 0; i < 105; i++)//地图的初始化

for(j = 0; j < 105; j++)

map[i][j] = INT\_MAX;

int a,b,c;

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

{

cin>>a>>b>>c;//读入地图

map[a][b] = c;

map[b][a] = c;

}

floyd(N);

cout<<length[1][N]<<endl;

}

return 0;

}

## hdoj 2066

### 1 原题目

一个人的旅行

**Time Limit: 1000/1000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 61325    Accepted Submission(s): 20444**

**Problem Description** <http://acm.hdu.edu.cn/showproblem.php?pid=2066>

虽然草儿是个路痴（就是在杭电待了一年多，居然还会在校园里迷路的人，汗~),但是草儿仍然很喜欢旅行，因为在旅途中 会遇见很多人（白马王子，^0^），很多事，还能丰富自己的阅历，还可以看美丽的风景……草儿想去很多地方，她想要去东京铁塔看夜景，去威尼斯看电影，去阳明山上看海芋，去纽约纯粹看雪景，去巴黎喝咖啡写信，去北京探望孟姜女……眼看寒假就快到了，这么一大段时间，可不能浪费啊，一定要给自己好好的放个假，可是也不能荒废了训练啊，所以草儿决定在要在最短的时间去一个自己想去的地方！因为草儿的家在一个小镇上，没有火车经过，所以她只能去邻近的城市坐火车（好可怜啊~）。

**Input**

输入数据有多组，每组的第一行是三个整数T，S和D，表示有T条路，和草儿家相邻的城市的有S个，草儿想去的地方有D个；  
接着有T行，每行有三个整数a，b，time,表示a,b城市之间的车程是time小时；(1=<(a,b)<=1000;a,b 之间可能有多条路)  
接着的第T+1行有S个数，表示和草儿家相连的城市；  
接着的第T+2行有D个数，表示草儿想去地方。

**Output**

输出草儿能去某个喜欢的城市的最短时间。

**Sample Input**

6 2 3

1 3 5

1 4 7

2 8 12

3 8 4

4 9 12

9 10 2

1 2

8 9 10

**Sample Output**

9

### 2 题目分析

给你一些两城镇之间的路程所花费的时间，再给你一些起点，和一些终点，让你求从任意起点到任意终点的最短时间（一条）

### 3 基础知识

#### 3.1数据结构

链式前向星（解法一用到）

<https://blog.csdn.net/acdreamers/article/details/16902023>

#### 3.2算法

SPFA算法（解法一用到）

<http://keyblog.cn/article-21.html>

### 4 解法

#### 4.1 解法1（C++已过）

算是SPFA算法的模板题，只不过每个起点都要调用一遍SPFA算法，然后选出一个最短的时间。

本人代码：

#include <iostream>

#include <climits>

#include <queue>

#include <cstring>

using namespace std;

const int MAXN = 10010;

struct Edge//链式前向星

{

int next;//与第i条边同一起点的上一条边的位置

int to;//此边的终点

int len;//边的权值

} edge[MAXN];

int head[MAXN];//以i为起点的第一条边存储的位置（实际是最后输入的那个编号）

int index = 0;

bool vis[MAXN];//用来标记点是否在队列中

int cnt[MAXN];//每个点的入队列次数

int length[MAXN];//记录从起点到点n的路径长

void add(int start,int end,int len)//链式前向星的加边函数

{

edge[index].len = len;

edge[index].to = end;

edge[index].next = head[start];

head[start] = index++;

}

bool SPFA(int start,int n)//最短路SPFA算法

{

int i,j;

memset(vis,false,sizeof(vis));

memset(cnt,0,sizeof(cnt));

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

{

length[i] = INT\_MAX;

}

queue<int>que;

que.push(start);//入队

vis[start] = true;//标记此点入队列

length[start] = 0;//源点到源点的长度为0

cnt[start]++;//入队次数加1

while(!que.empty())

{

int u = que.front();

que.pop();//出队

vis[u] = false;//标记此点已经不在队列中

for(j = head[u]; j != -1; j = edge[j].next)//遍历与u联通的点

{

int v = edge[j].to;

if(length[v] > length[u] + edge[j].len)

{

length[v] = length[u] + edge[j].len;//路径更新

if(vis[v] == 0)//如果v点不在队列中

{

que.push(v);//入队

vis[v] = 1;//标记

cnt[v]++;//入队次数加1

if(cnt[v] > n)//如果这个点加入超过n次，说明存在负环回路，直接返回

return false;

}

}

}

}

return true;

}

int main()

{

int T,S,D;

while(cin>>T>>S>>D)

{

int i,j,a,b,time,start[MAXN],end[MAXN],minn = INT\_MAX;

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

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

{

cin>>a>>b>>time;

add(a,b,time);//构建链式前向星

add(b,a,time);

}

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

cin>>start[i];//读入相邻城市

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

cin>>end[i];//读入目标城市

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

{

SPFA(start[i],T);//SPFA算法

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

minn = min(minn,length[end[j]]); //找到目标城市花费的最短时间

}

cout<<minn<<endl;

}

return 0;

}

## hdoj 3038

### 1 原题目

有多少答案是错误的

**时间限制：2000/1000 MS（Java /其他）内存限制：32768/32768 K（Java /其他）  
总提交：23680已接受提交：8044**

**问题描述** <http://acm.hdu.edu.cn/showproblem.php?pid=3038>

TT和FF是...的朋友。呃...非常非常好的朋友-\_\_\_\_\_\_\_\_- b  
  
FF是个坏男孩，他总是向TT求爱，和他一起玩以下游戏。这是一个非常单调的游戏。首先，TT应该写下一个整数序列-\_- !!（无聊）。



然后，FF可以从中选择一个连续的子序列（例如，从第三个整数到第五个整数（包括端值）的子序列）。此后，FF将询问TT他选择的子序列的总和是多少。接下来，TT将回答FF的问题。然后，FF可以重做此过程。最后，FF必须计算出整数的整个序列。  
  
无聊~~无聊~~一个非常无聊的游戏！！！TT根本不想玩FF。为了惩罚FF，她经常告诉FF错误的答案。  
  
这个坏男孩不是一个傻瓜。FF检测到某些答案不兼容。当然，这些矛盾使计算序列变得困难。  
  
但是，TT是个好可爱的女孩。她没有FF的硬心。为了节省时间，她保证如果确实没有逻辑错误，那么答案是正确的。  
  
而且，如果FF发现答案有误，在判断下一个答案时，他将忽略该答案。  
  
但是会有如此多的问题，可怜的FF无法马上确定当前答案是对还是错。因此，他决定编写程序来帮助他解决此问题。该计划将收到来自FF的一系列问题以及FF从TT收到的答案。该程序的目的是找出错误的答案。FF只有忽略错误的答案，才能得出整数的整个序列。可怜的FF没有时间做这项工作。现在他正在寻求您的帮助〜（为什么要为自己找麻烦~~坏男孩）

**输入值**

第1行：两个整数N和M（1 <= N <= 200000，1 <= M <= 40000）。意味着TT写了N个整数，而FF问了她M个问题。  
  
行2..M + 1：行i + 1包含三个整数：Ai，Bi和Si。意味着TT回答FF，从Ai到Bi的总和是Si。保证0 <Ai <= Bi <=N。  
  
您可以假定子序列的总和都适合32位整数。

**输出量**

带有整数的单行表示有多少个答案是错误的。

**样本输入**

10 5

1 10 100

7 10 28

1 3 32

4 6 41

6 6 1

**样本输出**

1

### 2 题目分析

给你一些数字的区间段（数字排序是随便的）并告诉你这些数字的和，让你从中判断出错误区间段的个数（比如先告诉你1到20号这些数字的和是100，又告诉你1到10号数字之和是200，那么这二者中肯定有一个错误的）

### 3 基础知识

#### 3.1数据结构

带权并查集（解法一用到）

#### 3.2算法

无。

### 4 解法

#### 4.1 解法1（C++已过）

参考链接：<https://www.cnblogs.com/liyinggang/p/5327055.html>

本人代码：

#include <iostream>

#include <cstring>

using namespace std;

int father[200005];//存储当前点的父节点

int sum[200005];//存储当前点到根节点的数字之和

int Find(int x)

{

if(x != father[x])

{

int temp = father[x];

father[x] = Find(father[x]);

sum[x] = sum[x] + sum[temp];

}

return father[x];

}

int main()

{

int N,M,a,b,s,i;

while(cin>>N>>M)

{

for(i = 0; i <= N ; i++)//数组初始化

{

father[i] = i;

sum[i] = 0;

}

int ra,rb,num = 0;

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

{

cin>>a>>b>>s;

a--;//因为判断时是靠区间相加（题目的区间要求是左闭右闭），所以左区间减一（通俗点说你知道了区间1-5，和6-10的和，如果你不把6减1,你如何经行下面的更新呢？）

ra = Find(a);

rb = Find(b);

if(ra == rb)//如果两点根节点相同

{

if(sum[a] != sum[b] + s)//满足这个条件就说明是错误的

num++;

}

else

{

father[ra] = rb;//根节点进行更新

sum[ra] = s + sum[b] - sum[a];//距离进行更新

}

}

cout<<num<<endl;

}

return 0;

}

## hdoj 1166

### 1 原题目

敌兵布阵

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 162460    Accepted Submission(s): 67166**

**Problem Description** <http://acm.hdu.edu.cn/showproblem.php?pid=1166>

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

### 2 基础知识

#### 2.1数据结构

树状数组（解法一用到）

#### 2.2算法

无。

### 3 解法

#### 3.1 解法1（G++已过）

树状数组的单点更新，区间查询的模板运用。

推荐两个树状数组讲解不错的链接：<https://www.cnblogs.com/findview/archive/2019/08/01/11281628.html>

<https://www.cnblogs.com/xenny/p/9739600.html>

本人代码：

#include <iostream>

#include<algorithm>

#include <cstring>

#include <cstdio>

#include <cstdlib>

#include <cmath>

using namespace std;

int a[50005],c[50005],N;//对应原数组和树状数组

int lowbit(int x)

{

return x&(-x);

}

int getsum(int i)//求a[1]到a[i]的和

{

int res = 0;

while(i > 0)

{

res += c[i];

i -= lowbit(i);

}

return res;

}

void updata(int i,int k)//在i位置加上k（原始数组和树状数组的值都会更改）

{

a[i] += k;

while(i <= N)

{

c[i] += k;

i += lowbit(i);

}

}

int main()

{

int T,num = 1;

cin>>T;

while(T--)

{

cout<<"Case "<<num++<<":"<<endl;

int i,temp;

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

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

cin>>N;

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

{

cin>>temp;

updata(i,temp);//输入初值的时候，也相当于更新了值

}

string str;

int j,k;

while(cin>>str && str[0] != 'E')

{

cin>>j>>k;

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

updata(j,k);

else if(str[0] == 'S')

updata(j,-k);//减去操作，即为加上相反数

else if(str[0] == 'Q')

cout<<getsum(k) - getsum(j - 1)<<endl;//x-y区间和也就等于1-y区间和减去1-(x-1)区间和

}

}

return 0;

}

## hdoj 1285

### 1 原题目

确定比赛名次

**Time Limit: 2000/1000 MS (Java/Others)    Memory Limit: 65536/32768 K (Java/Others)  
Total Submission(s): 52010    Accepted Submission(s): 19469**

**Problem Description**

有N个比赛队（1<=N<=500），编号依次为1，2，3，。。。。，N进行比赛，比赛结束后，裁判委员会要将所有参赛队伍从前往后依次排名，但现在裁判委员会不能直接获得每个队的比赛成绩，只知道每场比赛的结果，即P1赢P2，用P1，P2表示，排名时P1在P2之前。现在请你编程序确定排名。

**Input**

输入有若干组，每组中的第一行为二个数N（1<=N<=500），M；其中N表示队伍的个数，M表示接着有M行的输入数据。接下来的M行数据中，每行也有两个整数P1，P2表示即P1队赢了P2队。

**Output**

给出一个符合要求的排名。输出时队伍号之间有空格，最后一名后面没有空格。  
其他说明：符合条件的排名可能不是唯一的，此时要求输出时编号小的队伍在前；输入数据保证是正确的，即输入数据确保一定能有一个符合要求的排名。

**Sample Input**

4 3

1 2

2 3

4 3

**Sample Output**

1 2 4 3

### 2 基础知识

拓扑排序+BFS+优先队列

### 3 解法

#### 3.1 解法1（C++已过）

直接利用拓扑排序+BFS就可以，但需要注意的是，题目要求多种情况时返回序号小的在前，所以在BFS的时候我们就不能利用普通的队列，要利用优先队列，这样题目就很容易出来了，模板题而已。拓扑排序不懂的，去百度百科找。

本人代码：

|  |
| --- |
| #include <iostream>  #include<algorithm>  #include <cstring>  #include <cstdio>  #include <cstdlib>  #include <cmath>  #include <vector>  #include <queue>  using namespace std;  struct qnode  {      int c;      qnode(int \_c=0):c(\_c) {}      bool operator <(const qnode &r)const      {          return c>r.c;//先返回序号小的      }  };  int main()  {      int N,M;      while(scanf("%d %d",&N,&M) != EOF)      {          int p1,p2;          // 存储有向图          vector<vector<int> > edges;          // 存储每个节点的入度          vector<int> indeg;          // 存储答案          vector<int> result;          edges.resize(N + 1);          indeg.resize(N + 1);          for(int i = 0; i < M; i++)          {              scanf("%d %d",&p1,&p2);              edges[p1].push\_back(p2);              indeg[p2]++;          }          //优先队列存储入度为0的点（即为正确的排名顺序）          priority\_queue<qnode> q;          for(int i = 1; i < indeg.size(); i++)          {              if(indeg[i] == 0)//入度为0的点先存进队列中                  q.push(i);          }          while(!q.empty())          {              int u = q.top().c;              q.pop();              result.push\_back(u);              for(int i = 0; i < edges[u].size(); i++)  //u点为去除掉的点，所以要将从u点来的入度减1              {                  indeg[edges[u][i]]--;                  if(indeg[edges[u][i]] == 0)                      q.push(edges[u][i]);              }          }          printf("%d",result[0]);          for(int i = 1; i < result.size(); i++)              printf(" %d",result[i]);          printf("\n");      }      return 0;  } |

## hdoj 3068

### 1 原题目

**最长回文**

**Time Limit: 4000/2000 MS (Java/Others)    Memory Limit: 32768/32768 K (Java/Others)  
Total Submission(s): 40820    Accepted Submission(s): 15006**

**Problem Description** <http://acm.hdu.edu.cn/showproblem.php?pid=3068>

给出一个只由小写英文字符a,b,c...y,z组成的字符串S,求S中最长回文串的长度.  
回文就是正反读都是一样的字符串,如aba, abba等

**Input**

输入有多组case,不超过120组,每组输入为一行小写英文字符a,b,c...y,z组成的字符串S  
两组case之间由空行隔开(该空行不用处理)  
字符串长度len <= 110000

**Output**

每一行一个整数x,对应一组case,表示该组case的字符串中所包含的最长回文长度.

**Sample Input**

aaaa

abab

**Sample Output**

4

3

### 2 基础知识

Manacher马拉车算法（解法一用到）

### 3 解法

#### 3.1 解法1（C++已过）

马拉车算法的模板题目。

关于此算法参考参考以下两个链接：

1. <https://blog.csdn.net/Form_/article/details/79766986>

2. <https://www.cnblogs.com/czsharecode/p/9705358.html>

本人代码：

|  |
| --- |
| #include <iostream>  #include<algorithm>  #include <cstring>  #include <cstdio>  #include <cstdlib>  #include <cmath>  #include <vector>  using namespace std;  #define MAXN 110005  int Manacher(char s[],int len)  {      char t[MAXN \* 2];      int l = 0;      //开头加'$'是为了防止越界      t[l++] = '$';      t[l++] = '#';      //在原字符串开头结尾以及字符之间插入一个字符(未在串中出现过的)      for(int i = 0; i < len; i++)      {          t[l++] = s[i];          t[l++] = '#';      }      t[l++] = '\0';      //p[i] - 1就是t串中以i为中点的回文长度      vector<int> p(MAXN \* 2,0);      //mx(某回文串延伸到的最右边下标),id(mx所属回文串中心下标)      int mx = 0,id = 0;      //bit（结果最大回文串中心下标）,maxlen（最大回文长度）      int maxlen = 0;      for(int i = 1; i < l; i++)      {          //算法核心          p[i] = mx > i ? min(p[2 \* id - i],mx - i) : 1;          //一个个进行比较          //t数组最左边 = '$',最右边 = '\0'，无需判断边界          while(t[i + p[i]] == t[i - p[i]])              p[i]++;          //当t[i]匹配的 右边界超过mx时,mx和id就更新          if(i + p[i] > mx)          {              mx = i + p[i];              id = i;          }          //更新结果数据          if(p[i] > maxlen)          {              maxlen = p[i];          }      }      return maxlen - 1;  }  int main()  {      char s[MAXN];      while(scanf("%s",s) != EOF)      {          int len=strlen(s);          printf("%d\n",Manacher(s,len));      }      return 0;  } |

## hdoj

### 1 原题目

### 2 基础知识

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |

## hdoj

### 1 原题目

### 2 基础知识

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |

## hdoj

### 1 原题目

### 2 基础知识

### 3 解法

#### 3.1 解法1（C++已过）

本人代码：

|  |
| --- |
|  |