# 第一章

## 1.1 第k++小数问题

实验室中有一台原始的神奇的计算机，它可以存储一个整数集合，具有两种操作：

（1）、插入操作 INS（x）：将一个整数 x 存入计算机中；

（2）、取数操作 GET：如果这是第 k（从 1 开始计数）次 GET 操作，则取出计算机内存

中按非降序排序后的第 k 个数。

现在，给你计算机的操作序列，请你输出相应的操作结果。 ★数据输入：

第一行两个整数 M(M<=30000)、N(N<= M)，表示有 M 个 INS 操作，有 N 个 GET 操作。第

二行有 M 个整数(均在 32 位整数范围内)，第 i（1<=i<=M）个整数为第 i 次 INS 操作的操作

数。第三行有 N 个整数(p1,p2,…,pN)，按非降序给出，表示当内存中有 pk(1<=k<=N)个整数

时，执行一次 GET 操作。下面的输入输出样例对应于问题描述中的数据。 ★结果输出：

输出每次 GET 操作取出的数，按样例格式输出。 输入示例 输出示例

7 4

4 2 -3 3 9 -99 3

1 2 6 6

4

4

2

3

#include<iostream>

#include<cstdio>

#include<cstring>

#include<queue>

using namespace std;

const int MAXN = 30000 + 5;

int data[MAXN];

struct mincmp

{

bool operator()(const int x,const int y)

{return x>y;}

};

int main()

{

priority\_queue<int,vector<int>, mincmp > minh;

priority\_queue<int,vector<int> > maxh;

int i,j=0,n,m,t,x;

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

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

{

scanf("%d",data+i);

}

j = 0;

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

{

scanf("%d",&t);

for(; j<t;++j)

{

if(maxh.empty())

minh.push(data[j]);

else

{

if(data[j]<maxh.top())

{ minh.push(maxh.top());

maxh.pop();

maxh.push(data[j]);

}else if(data[j]>=maxh.top())

{

minh.push(data[j]);

}

}

}

printf("%d\n",minh.top());

maxh.push(minh.top());

minh.pop();

}

return 0

## 1.2 符号平衡问题

给定 n 个整数 a1,a2,…，an组成的序列。序列中元素 ai 的符号定义为：

ai > 0 : 1

ai == 0 : 0

ai <0 : -1

符号平衡问题是求给定序列的最长符号平衡段的长度 L，即：

例如，当 n=10，相应序列为：1，1，-1，-2，0，1，3，-1，2，-1 时，L=9。

★编程任务

给定 n（1<=n<=3,000,000）个整数 a1,a2,…，an 组成的序列，试设计一个 O(n) 时间算法，计算

其最长符号平衡段的长度。 ★数据输入

输入数据的第 1 行是正整数 n，第 2 行是整数序列 a1,a2,…，an。 ★数据输出

将计算出的最长符号平衡段的长度输出。

Input:

10

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

Output:

9

#include<stdio.h>

#include<string.h>

const int N = 3000000;

int pos[N+N+10];

int max(int x,int y)

{

return x>y?x:y;

}

int main()

{

int n,x,i,ans=0,sum=0,tmp;

scanf("%d",&n);

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

{

scanf("%d",&x);

if(x>0)

{

x = 1;

}

else if(x<0)

{

x = -1;

}

sum+=x;

if(sum == 0)

{

max(ans,i);

}

else

{

if(sum>0)

tmp = sum+N;

else

tmp = -sum;

if(pos[tmp]>0)

{

ans = max(ans,i-pos[tmp]);

}

else

{

pos[tmp] = i;

}

}

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

return 0;

## 1.3 社团统计问题

#include<stdio.h>

#include<string.h>

const int N = 1000010;

struct union\_find

{

int parent[N];

bool root[N];

int root\_num;

void init(int \_size = N)

{

root\_num = \_size;

for(int i=1;i<=\_size;++i)

{

parent[i] = 1;

root[i] = true;

}

}

int find(int e)

{

int p = e;

int tmp;

while(!root[p]) p = parent[p];

while(e!=p)

{

tmp = parent[e];

parent[e] = p;

e = tmp;

}

return p;

}

int Union(int i,int j)

{

if(i == j)return i;

if(parent[i]<parent[j])

{

parent[j]+=parent[i];

root[i] = false;

--root\_num;

parent[i] = j;

return j;

}else

{

parent[i]+=parent[j];

root[j] = false;

--root\_num;

parent[j] = i;

return i;

}

}

int get\_root\_num()

{

return root\_num;

}

};

struct union\_find uf;

int main()

{

int x,y,n,m;

int i;

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

{

uf.init(n);

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

{

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

uf.Union(uf.find(x),uf.find(y));

}

printf("In this case,the result is %d.\n",uf.get\_root\_num());

}

return 0;

}

## 1.4 数字游戏

Winder 最近在玩一个数字游戏，该游戏是在一个 n\*m 的网格上进行的，每个格子上有

一个数字，代表这个格子的数值。玩家需要从网格的左上角的格子走到右下角的格子，每次

只能向右或者向下走，并且不能回头。玩家每经过一个格子可以选择分值是否加上该格子的

数值，每次游戏的初始分数都是 0。

Input:

6 5

1 2 3 4 5

6 7 8 9 10

11 12 13 14 15

16 17 18 19 20

21 22 23 24 25

26 27 28 29 30

Output:

195

#include<stdio.h>

const int N = 16;

int dp[N];

inline int max(int a,int b)

{

return a>b?a:b;

}

int main()

{

int n,m,x;

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

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

{

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

{

scanf("%d",&x);

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

dp[j]+= x>0?x:0;

}

}

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

return 0;

## 1.5 水箱问题

#include<stdio.h>

const int N= 50010;

int b[N],h[N],d[N],w[N],n;

int main()

{

int i,tot=0;

double v;

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

{

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

{

scanf("%d%d%d%d",b+i,h+i,d+i,w+i);

}

scanf("%lf",&v);

if(v-0<1e-9)

{

printf("0.00\n");

}

double left = 0,right = 2000000001,mid;

while(right-left>1e-3)

{

mid = (right+left)/2;

tot = 0;

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

{

if(mid<b[i])continue;

if(mid>=b[i]+h[i])

{

tot+= h[i]\*w[i]\*d[i];

}else

{

tot+= w[i]\*d[i]\*(mid-b[i]);

}

}

if(tot<v)

left=mid;

else

right = mid;

}

printf("%.2f",left);

}

return 0;

# 第二章 分治

## 2.1 条形图轮廓问题

★

在 x 轴上水平放置着 n 个条形图。条形图的轮廓是消去这 n 个条形图的隐藏线后得到的

图形，如图所示。

每个条形图由 3 元组（Li，Hi，Ri）表示。其中，Li 和 Ri 分别为条形图左右竖线的 x 坐标值，Hi 为条形图的高度。例如，上图的 8 个条形图表示为：（1,11,5），（2,6,7），（3,13,9），

（12,7,16），（14,3,25），（19,18,22），（23,13,29），（24,4,28）。条形图的轮廓可用轮廓向

量（V1，V2，…，Vm）表示。当 i 为奇数时，Vi 表示条形图轮廓中一条竖线的 x 坐标值：当

i 为偶数时，Vi 表示条形图轮廓中一条横线的高度。例如，上图的条形图轮廓向量为

（1,11,3,13,9,0,12,7,16,3,19,18,22,3,23,13,29,0）。

现在，对于给点的 n 个条形图，计算其条形图轮廓。 ★数据输入

第一行一个正整数 n，表示 n 个条形图（1 <= n <= 5000 0）。

接下来 n 行，每行有 3 个整数（Li，Hi，Ri），Li 和 Ri 分别为条形图左右竖线的 x 坐标

值，Hi 为条形图的高度（-30 0 00<= Li，Ri<= 300 00 ，1 <= Hi <= 100 00 ）。 ★数据输出

输出计算出的条形图轮廓向量。

8

1 11 5

2 6 7

3 13 9

12 7 16

14 3 25

19 18 22

23 13 29

24 4 28

Output: 1 11 3 13 9 0 12 7 16 3 19 18 22 3 23 13 29 0

#include <stdio.h>

#include <memory.h>

struct SS

{

short height;

short left;

};

void merge(SS\* c, SS\* d, int l, int m, int r)

{

int i = l, j = m+1, k = l;

while((i <= m) && (j <= r))

{

if(c[i].left == c[j].left)

{

if(c[i].height >= c[j].height)

d[k++] = c[i++];

else

d[k++] = c[j++];

continue;

}

if(c[i].left < c[j].left)

{

if(j == m+1) d[k++] = c[i++];

else

{

if(c[i].height >= c[j-1].height)

d[k++] = c[i++];

else

{

d[k].height = c[j-1].height;

d[k++].left = c[i++].left;

}

}

}

else

{

if(i == l) d[k++] = c[j++];

else

{

if(c[j].height >= c[i-1].height)

d[k++] = c[j++];

else

{

d[k].height = c[i-1].height;

d[k++].left = c[j++].left;

}

}

}

}

if(i > m) for(int q = j; q <= r; q++) d[k++] = c[q];

else for(int q = i; q <= m; q++) d[k++] = c[q];

}

void mergePass(SS\* a, SS\* b, int s, int n)

{

int i = 0;

while(i < n - 2\*s)

{

merge(a, b, i, i+s-1, i+2\*s-1);

i = i + 2 \* s;

}

if(i + s < n) merge(a, b, i, i+s-1, n-1);

else for(int j = i; j <= n-1; j++) b[j] = a[j];

}

void mergeSort(SS\* a, int n)

{

SS\* b = new SS[n];

int s = 2;

int i;

while(s < n)

{

mergePass(a, b, s, n);

s += s;

mergePass(b, a, s, n);

s +=s;

}

s = -1;

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

{

if(s != a[i].height)

{

s = a[i].height;

printf("%hd %hd ", a[i].left, a[i].height);

}

}

printf("\n");

delete [] b;

}

int main

{

int i, j;

int n;

SS\* p;

scanf("%d", &n);

p = new SS[2\*n];

for(i = 0, j = 0; i < n; i++, j+=2)

{

scanf("%hd%hd%hd", &(p[j].left), &(p[j].height), &(p[j+1].left));

p[j+1].height = 0;

}

mergeSort(p, 2\*n);

delete [] p;

return 0;

}

## 2.2 集合划分问题

问题描述：

n 个元素的集合{1,2,, n }可以划分为若干个非空子集。例如，当 n=4 时，集合{1，2，

3，4}可以划分为 15 个不同的非空子集如下：

{{1}，{2}，{3}，{4}}，

{{1，2}，{3}，{4}}，

{{1，3}，{2}，{4}}，

{{1，4}，{2}，{3}}，

{{2，3}，{1}，{4}}，

{{2，4}，{1}，{3}}，

{{3，4}，{1}，{2}}，

{{1，2}，{3，4}}，

{{1，3}，{2，4}}，

{{1，4}，{2，3}}，

{{1，2，3}，{4}}，

{{1，2，4}，{3}}，

{{1，3，4}，{2}}，

{{2，3，4}，{1}}，

{{1，2，3，4}}

´编程任务：

给定正整数 n，计算出 n 个元素的集合{1,2,, n }可以划分为多少个不同的非空子集。

´数据输入：5

´结果输出:

52

输入数据包括一行，代表正整数n(0<n<24)（即元素个数n）。

输出整数n的不同的非空子集数。（提示：结果采用64位int型）

#include <iostream>

using namespace std;

int main()

{

\_\_int64 array[24][24];

\_\_int64 ans;

int n;

ans=0;

scanf("%d",&n);

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

{

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

{

if(i==j||i==0) array[i][j]=1;

else array[i][j]=array[i][j-1]\*(i+1)+array[i-1][j-1];

}

}

for(int k=0;k<n;k++) ans+=array[k][n-1];

printf("%I64d",ans);

return 0;

}

## 2.3 最长公共子串问题

3 6 1 4 2

1 1 4 4 0

#include <stdio.h>

int ans;

int Max(int a,int b)

{

if(a>b)

return a;

return b;

}

int Min(int a,int b)

{

if(a<b)

return a;

return b;

}

void cmp(int l1,int l2,int r1,int r2)

{

ans = Max(ans,Min(r1,r2)-Max(l1,l2)+1);

}

void gao(int l1,int l2,int r1,int r2, int mid)

{

if((l1<=l2&&r2<=r1) || (l2<=l1&&r1<=r2))

{

cmp(l1,l2,r1,r2);

return;

}

int temp;

bool flag1,flag2;

flag1 = flag2 = true;

if(l1<=mid&&mid<=r1)

flag1 = false;

if(l2<=mid&&mid<=r2)

flag2 = false;

if(flag1&&flag2)//对应ppt情况1

gao(l1%mid,l2%mid,r1%mid,r2%mid,mid>>1);

else if(!flag1&&!flag2)//对应ppt情况3

{

cmp(l1,l2,r1,r2);

if(l1<mid&&r2>mid)

gao(l1,1,mid-1,r2%mid,mid>>1);

if(l2<mid&&r1>mid)

gao(l2,1,mid-1,r1%mid,mid>>1);

}

else//对应ppt情况2

{

if(flag1)

{

temp = l1;

l1 = l2;

l2 = temp;

temp = r1;

r1 = r2;

r2 = temp;

}

if(l1<mid)

gao(l1,l2%mid,mid-1,r2%mid,mid>>1);

if(r1>mid)

gao(1,l2%mid,r1%mid,r2%mid,mid>>1);

}

return;

}

int main()

{

int l1,l2,r1,r2;

while(scanf("%d%d%d%d",&l1,&r1,&l2,&r2)!=EOF)

{

ans = 0;

gao(l1,l2,r1,r2,(1<<30));

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

}

return 0;

}

## 2.4 树上路径

鹏给一棵树,每条边有权.求一条路径,权值和等于 K,且边的数量最小.

★数据输入：

第一行 两个整数 n, k (1<=n<=300000,0<=k<2^31-1)

第二..n 行 每行三个整数 表示一条无向边的两端和权值 (注意点的编号从 0 开

始)

★结果输出:

一个整数 表示最小边数量 如果不存在这样的路径 输出-1。

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

#define EPS 0x3fffffff

typedef \_\_int64 llong;

const int N = 300004;

int max(int aa,int bb)

{

return aa>bb?aa:bb;

}

int min(int aa,int bb)

{

return aa<bb?aa:bb;

}

typedef struct Node

{

int color,b;

llong d;

bool operator<(const Node& z)const

{

if(d ==z.d)return color<z.color;

return d<z.d;

}

}Node;

typedef struct Edge

{

int v,w;

}Edge;

Edge E[N+N];

int edge\_num;

int Next[N+N],head[N];

int n,k;

int max\_son[N],sum\_node[N];

Node dist[N];

int d\_num;

int visited[N];

int que[N],q\_n;

void add\_edge(int u,int v,int w)

{

E[edge\_num].v = v; E[edge\_num].w = w; Next[edge\_num] = head[u]; head[u] = edge\_num++;

E[edge\_num].v = u; E[edge\_num].w = w; Next[edge\_num] = head[v]; head[v] = edge\_num++;

}

void init()

{

edge\_num=0;

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

memset(visited,0,sizeof(visited));

}

void dfs4center(int now,int fa)

{

que[q\_n++] = now;

sum\_node[now] = 1;

max\_son[now] = 0;

int v;

for(int p=head[now];p!=-1;p=Next[p])

{

v= E[p].v;

if(v!=fa && !visited[v])

{

dfs4center(v,now);

sum\_node[now]+=sum\_node[v];

max\_son[now] = max(sum\_node[v],max\_son[now]);

}

}

}

int get\_center(int r,int fa)

{

q\_n = 0;

dfs4center(r,fa);

int ansn = q\_n,ansi=-1,tmp;

for(int i=0;i<q\_n;++i)

{

tmp = max(max\_son[que[i]],q\_n-sum\_node[que[i]]) ;

if(ansn > tmp)

{

ansn = tmp;

ansi = que[i];

}

}

return ansi;

}

void dfs4find(int now,int d,int fa,int c,int b)

{

dist[d\_num].b = b;

dist[d\_num].color = c;

dist[d\_num++].d = d;

int v;

for(int p=head[now];p!=-1;p=Next[p])

{

v= E[p].v;

if(v!=fa && !visited[v])

{

dfs4find(v,E[p].w+d,now,c,b+1);

}

}

}

int find(int r)

{

int ans=EPS,v;

d\_num = 0;

dist[d\_num].color = 0;

dist[d\_num].b = 0;

dist[d\_num++].d = 0;

visited[r]=1;

int p;

int c;

for(p=head[r],c=1;p!=-1;p=Next[p])

{

if(!visited[E[p].v])

{

dfs4find(E[p].v,E[p].w,r,c++,1);

}

}

std::sort(dist,dist+d\_num);

int i=0,j=d\_num-1,tt;

while(i<j)

{

while(i<j && dist[i].d+dist[j].d>k )--j;

tt = j;

while(i<tt && dist[i].d+dist[tt].d == k && dist[i].color!=dist[tt].color)

{

ans = min(ans,dist[i].b+dist[tt].b);

--tt;

};

++i;

}

for(p=head[r];p!=-1;p=Next[p])

{

v = E[p].v;

if(!visited[v])

{

v = get\_center(v,r);

ans=min(ans,find(v));

}

}

return ans;

}

void solve()

{

int u,v,w;

init();

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

{

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

add\_edge(u,v,w);

}

int ans = find(get\_center(1,-1));

if(ans == EPS)

printf("-1\n");

else

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

}

int main()

{

/\*

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

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

{

solve();

}

\*/

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

solve();

return 0;

}

## 2.5 最近点对

#include<iostream>

#include<algorithm>

#include<cmath>

#include<vector>

#include<fstream>

#include<ctime>

using namespace std;

const int MAXN = 10010;

struct Point {

double x, y;

Point(){}

Point(const double &\_x, const double &\_y):x(\_x), y(\_y) {}

bool operator<(const Point& z)

{

return x < z.x || x == z.x && y < z.y;

}

};

int n;

Point p[MAXN];

bool cmpx(const Point &a, const Point &b)

{

return a.x < b.x || a.x == b.x && a.y < b.y;

}

bool cmpy(const int &a, const int &b)

{

return p[a].y < p[b].y;

}

double get\_dist(Point a, Point b)

{

return ((a.x - b.x) \* (a.x - b.x) + (a.y - b.y) \* (a.y - b.y));

}

int p1[MAXN / 2 + 1], p2[MAXN / 2 + 1];

double find\_pair(int left, int right, Point p[])

{

double best = 1e20, temp;

if(right - left == 1) return get\_dist(p[left], p[right]);

if(right - left < 10) {

for(;left < right;++ left)

for(int j = left + 1;j <= right;++ j)

if(best > (temp = get\_dist(p[left], p[j]))) best = temp;

return best;

}

int mid = (left + right) >> 1;

double min1, min2;

min1 = find\_pair(left, mid, p);

min2 = find\_pair(mid, right, p);

best = (min1 < min2) ? min1 : min2;

int cnt1 = 0, cnt2 = 0, L = mid - 1, r = mid + 1, i, j, k;

while(L >= left && p[mid].x - p[L].x < best) p1[cnt1++] = L --;

while(r <= right && p[r].x - p[mid].x < best) p2[cnt2++] = r ++;

sort(p1, p1 + cnt1, cmpy);

sort(p2, p2 + cnt2, cmpy);

for(L = k = 0;L < cnt1;++ L) {

for(i = p1[L], r = k;r < cnt2;++ r) {

j = p2[r];

if(fabs(p[j].x - p[i].x) >= best) continue;

if(p[j].y - p[i].y >= best) break;

if(p[i].y - p[j].y >= best) { k = r; continue; }

temp = get\_dist(p[j], p[i]);

if(temp < best) best = temp;

}

}

return best;

}

double closest\_pair(int n, Point p[])

{

sort(p, p + n, cmpx);

return sqrt(find\_pair(0, n - 1, p)) ;

}

int main()

{

int i;

while(scanf("%d",&n)&& n!=0)

{

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

{

scanf("%lf%lf",&p[i].x,&p[i].y);

}

printf("%.2f\n",closest\_pair(n,p)/2);

}

return 0;

}

# 第三章 回溯

## 3.1 编辑距离

/\*

编辑距离

1. 删除

2. 插入

3. 修改

dp[i][j] = min ( dp[i][j-1] + 1, dp[i-1][j]+1,dp[i-1][j-1]+(a[i] == b[j])

abcab

abab

\*/

#include<stdio.h>

#include<string.h>

const int N = 2011;

int dp[N][N];

char s1[N],s2[N];

inline int min(int x,int y)

{return x<y?x:y;}

int main()

{

int i,j;

scanf("%s%s",s1,s2);

int n,m;

n = strlen(s1);

m = strlen(s2);

for(i=0;i<=n;++i)dp[i][0] = i;

for(j=0;j<=m;++j)dp[0][j] = j;

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

{

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

{

dp[i][j]= min(min(dp[i][j-1]+1,dp[i-1][j]+1),dp[i-1][j-1]+!(s1[i-1] == s2[j-1]));

}

}

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

return 0;

}

## 3.2 蛋糕

/\*

dp[i][j] 表示前i个放j个A后最多能放多少个B

dp[i][j] = max（dp[i-1][j]，dp[i-3][j-1] 当前 (i-3 ：i) 能放 A

max( dp[i-1][j], dp[i-3][j] + 1) 当前 （i-3：i）能放 B

dp[i-1][j] 不能放A或B

\*/

#include<iostream>

#include<cstring>

#include<cmath>

#include<algorithm>

using namespace std;

const int N = 1001;

char s[N];

int dp[N][N];

inline int max(int x,int y)

{

return x>y?x:y;

}

inline int min(int x,int y)

{

return x<y?x:y;

}

void solve()

{

int ans,i,j;

int n=strlen(s);

if(n<6)

{

printf("0\n");

return;

}

int cnt=0,t;

if(s[2]=='a' && s[1]=='c' && s[0] == 'b')

{

dp[2][0]=1;

}

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

{

if(s[i] == 'c' && s[i-1]=='b' && s[i-2] == 'a')

{

t=1;

}

else if(s[i]=='a' && s[i-1]=='c' && s[i-2] == 'b')

{

t=2;

}

else

{

t=0;

}

if(t==1)++cnt;

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

{

if(t==1)

{//放A或不放

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

if(j>0)

dp[i][j] = max(dp[i][j],dp[i-3][j-1]);

}else if(t==2)

{//放B或不放

dp[i][j]=max(dp[i-3][j]+1,dp[i-1][j]);

}else

{

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

}

}

}

ans = 0;

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

{

ans=max(ans,min(i,dp[n-1][i]));

}

printf("%d\n",ans\*3);

}

int main()

{

scanf("%s",s);

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

solve();

return 0;

}

## 3.3 最长递增子序列

/\*

dp[i] 表示以第i个数结尾的序列的最长递增子序列长度

dp[i] = max( dp[ t= 1:i-1 ] )+1 where a[t]<a[i]

复杂度O(n^2)

非dp算法 O(nlog(n))：

维护一个 有序数组，st[]，st[i] 为 长度为i的递增子序列的最末数的最小值。

对于a[i] , if(a[i]>st[len-1]) st[len++] = a[i]; else 二分查找 st 中 第一个 st[k] >= a[i] 的数，替换之。

4 5 6 3 1 2 3 5 6

\*/

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

const int N = 100001;

int st[N];

int main()

{

int n,x,p;

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

{

int len = 1;

scanf("%d",&st[0]);

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

{

scanf("%d",&x);

if(st[len-1]<x)

st[len++]=x;

else

{

p = lower\_bound(st,st+len,x)-st;

st[p] = x;

}

}

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

}

return 0;

}

## 3.4 消去问题

/\*

dp[i][j] 表示 前i个至少还要放入j个才能全部消除 的方案数

dp[i][j] = dp[i-1][j+1] + dp[i-1][j-k+1]\*(h-1) (j != k-1)

+ dp[i-1][j-k+1]\*h (j == k-1)

\*/

#include<iostream>

#include<cstring>

using namespace std;

const int N= 1001;

const int MOD = 1000000007;

typedef \_\_int64 llong;

int a[N];

int dp[2][N];

int n,m,h,k;

void solve()

{

int i;

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

{

scanf("%d",a+i);

}

int cnt=0;

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

{

cnt+=k-1;

while(i+1<=n && a[i]==a[i+1])

{

++i;

--cnt;

}

}

dp[0][cnt]=1;

int len = n+m;

int c=1;

llong hh=h;

for(i=n+1;i<len;++i,c=1-c)

{

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

{

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

if(j==k-1)

{

dp[c][j]= (dp[c][j]+dp[1-c][j-k+1]\*(hh) )%MOD;

}else if(j>k-1)

{

dp[c][j]=(dp[c][j] + dp[1-c][j-k+1]\*(hh-1))%MOD;

}

}

}

printf("%d\n",dp[1-c][1]);

}

int main()

{

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

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

solve();

return 0;

}

## 3.5 邮局问题

#include<stdio.h>

int d[1005];

int opt[1005][1005],cost[1005][1005],s[1005][1005];

const int inf=100000000;

int main()

{

int m,n,i,j,k,mid;

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

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

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

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

{

cost[i][i]=0;

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

{

mid=(i+j)/2;

cost[i][j]=cost[i][j-1]+(d[j]-d[mid]);

}

}

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

{

opt[i][1] = cost[1][i];

s[i][1]=0;

}

for(j=2;j<=m;j++)

{

s[n+1][j]=n;

for(i=n;i>j;i--)

{

opt[i][j]=inf;

for(k=s[i][j-1];k<=s[i+1][j];k++)

{

if(opt[i][j]>opt[k][j-1]+cost[k+1][i])

{

opt[i][j]=opt[k][j-1]+cost[k+1][i];

s[i][j]=k;

}

}

}

}

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

return 0; }

# 第四章 贪心

## 4.1 勤劳的Alice

#include<iostream>

#include<algorithm>

using namespace std;

int n,m,k;

int a[20];

bool cmp(int x,int y)

{return x>y;}

void solve()

{

int i,cnt = 0;

bool flag =false;

sort(a,a+n,cmp);

while(k--)

{

if(a[m-1] == 0)

{

for(i=0;i<n;++i)a[i]+=10;

++cnt;

}

for(i=0;i<m;++i)--a[i];

sort(a,a+n,cmp);

}

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

}

int main()

{

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

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

for(int i=0;i<n;++i)scanf("%d",a+i);

solve();

return 0;

}

## 4.2 懒惰的 Bob

Alice 和 Bob 在玩一个有趣的游戏。

Bob 拥有 n 个不同的数字，但每次只能拿一个数字在手上。按照时间顺序，Alice 依次

给出一个数，和 Bob 手中的数字相比较。若数字相同，Bob 必须进行切换，从 n 个数中选

一个不同的数。

Bob 是个很懒惰的人。每进行一次切换，他的心情就会变得更差。所以，他希望切换的

次数越少越好。为了让着 Bob，Alice 事先把自己所拿出数字序列告诉了 Bob，你能帮 Bob 计算出，他最少的切换次数吗？

Input： 2 6

1 3 2 3 2 2

Output：1

Input： 3 9

1 3 1 41 20 2 3 2 1

Output： 2

#include <iostream>

#include <vector>

using namespace std;

int main()

{

int n,m,i,t=0,sum=0,total=0;

bool flag=false;

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

vector<bool> b(n+1,0);

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

{

scanf("%d",&t);

if(t<=n)

{

if(b[t]==flag)

{

b[t]=!flag;

if(++sum==n)

{

flag=!flag;

total++;

b[t]=!flag;

sum=1;

}

}

}

}

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

return 0

## 4.3 郊游问题

#include<iostream>

#include<algorithm>

#include<vector>

#include<queue>

typedef \_\_int64 llong;

using namespace std;

const int N = 100001;

typedef struct Node{

int a,b;

llong t;

Node(){}

Node(int \_a,int \_b,llong \_t):a(\_a),b(\_b),t(\_t){}

bool operator<(const Node& z)const

{

return t>z.t || (t==z.t && b<z.b);

}

}Node;

bool cmp(const Node& node1,const Node& node2)

{

return node1.a<node2.a;

}

Node boxs[N];

Node min\_heap[N];

void solve()

{

int i,n,d;

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

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

scanf("%d%d%d",&boxs[i].a, &boxs[i].b, &boxs[i].t);

sort(boxs,boxs+n,cmp);

int top = 0;

min\_heap[top++] = Node(0,1,0);

int nowb;

llong nowt;

i=0;

while(i<n && top)

{

pop\_heap(min\_heap,min\_heap+top);

--top;

nowb = min\_heap[top].b;

nowt = min\_heap[top].t;

if(nowb>=d)

{

printf("%I64d\n",nowt);

return;

}

while(i<n && nowb>= boxs[i].a)

{

min\_heap[top++] = Node(boxs[i].a,boxs[i].b,boxs[i].t+nowt);

push\_heap(min\_heap,min\_heap+top);

++i;

}

}

while(top)

{

pop\_heap(min\_heap,min\_heap+top);

--top;

nowb = min\_heap[top].b;

nowt = min\_heap[top].t;

if(nowb>=d)

{

printf("%I64d\n",nowt);

return;

}

}

printf("-1\n");

}

int main()

{

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

int T;

scanf("%d",&T);

while(T--)

{

solve();

}

return 0;

}

## 4.4 旅游购物问题

小 F 最近去玩了，很不小心的跟了个旅游团。于是乎，导游带他们四处购物。所幸的是

导游所推荐的物品中有一些是小 F 感兴趣的东西，但是每件商品限购一件，买多了小 F 就没

兴趣了。对于每个物品，小 F 的都有一个价值的评价，这次出门小 F 带了 M 元的 RMB，他希

望能在 M 元内（包含 M 元）买到的物品的价值总和最大。★实验任务

小 F 希望你能帮他算出购买的物品的价值总和最大的值。 ★数据输入

输入第一行有一个整数 N，M（0<=N<=100,0<=M<=10000），N 代表有 N 件物品，M 代

表小 F 所携带的金钱。

输入第二行有 N 个整数，分别代表每个物品的价格。

输入第三行有 N 个整数，分别表示小 F 对每件物品的价值评价。 ★数据输出

每个测试数据输出一行，表示小 F 能买到的最大价值。

Input：

3 10

2 4 5

3 7 2

Output:

10

#include <iostream>

using namespace std;

const int MAX\_NUM = 100;

const int MAX\_WEIGHT = 10000;

int weight[MAX\_NUM];

int value[MAX\_NUM];

//进行空间压缩，使用一维数组

int dp[MAX\_WEIGHT];

void Solve( int n, int w )

{

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

{

//因为使用了一维数组，所有j要按照递减顺序

for( int j=w; j>=weight[i]; j-- )

{

if( dp[j-weight[i]] + value[i] > dp[j] )

dp[j] = dp[j-weight[i]] + value[i];

}

}

cout << dp[w] << endl;

}

int main()

{

int n, w;

int i;

cin >> n >> w;

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

{

cin >> weight[i];

}

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

{

cin >> value[i];

}

Solve( n, w );

return 0;

}

## 4.5 战旗

小 M 喜欢玩一款战棋游戏。游戏的规则很简单，小 M 有 N 个士兵，每个士兵都有个对应

的战斗值 Fi，电脑有 N 个怪物，每个怪物的战斗值为 Mi。每场战斗在一个士兵和一个怪物

间展开。若士兵的战斗值高于怪物的战斗值，小 M 的积分加 10；若士兵的战斗值低于怪物

的战斗值，小 M 的积分减 10，二者战斗值相等则积分不变。 ★实验任务

小 M 希望你能帮其安排士兵与怪物的对阵，从而获得最高的积分。 ★数据输入

输入第一行有一个整数 T 表示，有 T 组测试数据。

每组测试数据的第一行有一个整数 N（1<=N<=10^3），表示士兵的数量。

接下来有两行，第一行有 N 个整数 Fi 表示士兵的战斗值，第二行有 N 个整数 Mi 表示

怪物的战斗值（1<=Fi，Mi<=10^3）。 ★数据输出

每个测试数据输出一行，表示小 M 能得到的最高积分。

Input:

2

3

2 4 6

3 5 7

2

1 2

2 1

Output:

10

#include<stdio.h>

#include<algorithm>

#include<stdlib.h>

int cmp(const void \*a, const void \*b)

{

return(\*(int \*)a-\*(int \*)b);

}

int solution(){

int F[1001],M[1001];

int N;

int score,maxF,maxM,minF,minM,count,i;

scanf("%d",&N);

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

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

}

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

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

}

qsort(F,N,sizeof(int),cmp);

qsort(M,N,sizeof(int),cmp);

score=0;

maxF=maxM=N-1;

minF=minM=0;

count = 0;

while((count++)<N){

if(M[maxM]<F[maxF]){

score=score+10;

maxM=maxM-1;

maxF=maxF-1;

}

else if(M[maxM]>F[maxF]){

score=score-10;

minF=minF+1;

maxM=maxM-1;

}

else{

if(F[minF]>M[minM]){

score=score+10;

minF=minF+1;

minM=minM+1;

}

else{

if(F[minF]<M[maxM]){

score=score-10;

minF=minF+1;

maxM=maxM-1;

}

}

}

}

return score;

}

int main(){

int T;

int \* scores;

scanf("%d",&T);

scores = (int \*) malloc(sizeof(int) \* T);

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

scores[i] = solution();

}

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

printf("%d\n",scores[j]);

}

return 0;

}

# 第五章 回溯

## 5.1 覆盖问题

#include<iostream>

#include<cstdio>

#include<cstring>

#include<cmath>

#include<algorithm>

#define INF 0x3fffffff

using namespace std;

int n,r;

typedef struct Node

{

int x,y,c;

bool operator < (const Node& z)const

{return c>z.c;}

}Node;

Node ns[24];

bool covs[24][24];

bool g[24][24];

bool vis[24];

int ans;

int get\_dist(int i,int j)

{

return (ns[i].x-ns[j].x)\*(ns[i].x-ns[j].x) + (ns[i].y-ns[j].y)\*(ns[i].y-ns[j].y);

}

void dfs(int u,int cost,int cnt)

{

if(cost > ans)return;

if(cnt == n)

{

ans = ans<cost?ans:cost;

return;

}

int i,j,vist[24];

int t=cnt,t1=0;

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

{

vist[i] = vis[i];

}

for(int v=u+1;v<n;++v)

{

t=cnt;

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

{

if(!vis[i] && covs[v][i])++t;

}

if(t == n)

{

t1=cnt;

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

{

if(g[v][i] && !vis[i])

{

vis[i]=1;

++t1;

}

}

dfs(v,cost+ns[v].c,t1);

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

{

vis[i] = vist[i];

}

}

}

}

int main()

{

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

int i,j;

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

{

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

{

scanf("%d%d%d",&ns[i].x,&ns[i].y,&ns[i].c);

}

scanf("%d",&r);

sort(ns,ns+n);

memset(covs,0,sizeof(covs));

memset(g,0,sizeof(g));

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

{

g[i][i]=1;

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

{

if(get\_dist(i,j)<=r\*r)

g[i][j] = g[j][i] = 1;

}

}

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

{

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

{

if(g[i][j])

{

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

{

covs[k][j] = 1;

}

}

}

}

ans = INF;

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

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

{

int t=0;

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

{

if(covs[v][i])++t;

}

if(t==n)

{

int t1=0;

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

{

if( g[v][i])

{

vis[i] = 1;

++t1;

}

}

dfs(v,ns[v].c,t1);

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

}

}

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

}

return 0;

}

## 5.2 搬砖

/\*

搜索 + 剪枝

\*/

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

typedef int llong ;

int n;

int a[64];

llong sum;

llong maxn;

void dfs(int u,llong ns)

{

ns+=a[u];

if(ns > sum)return;

maxn = maxn>ns?maxn:ns;

if(maxn == sum)return;

if(ns+a[0]>sum)return;

for(int i=u-1;i>=0;--i)

{

dfs(i,ns);

}

}

int main()

{

scanf("%d",&n);

sum=0;

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

{

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

sum+=a[i];

}

llong tmp = sum;

sum/=2;

maxn = 0;

sort(a,a+n);

dfs(n,0);

//printf("%I64d %I64d\n",maxn,tmp-maxn);

printf("%d %d\n",maxn,tmp-maxn);

return 0;

}

/\*

4

100

90

200

100

3

1

1

1

\*/

## 5.3 带权任务调度问题

/\*贪心\*/

#include<iostream>

#include<algorithm>

#include<vector>

using namespace std;

typedef struct task{

int t;

int w;

}task;

bool cmp(task x,task y)

{

return x.t \* y.w <y.t \* x.w;

}

vector<task> t;

void main()

{

int n,i,s=0,c=0;

task tmp;

cin >> n;

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

{

cin >> tmp.t;

t.push\_back(tmp);

}

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

{

cin >> t[i].w;

}

sort(t.begin(),t.end(),cmp);

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

{

c += t[i].t;

s += t[i].w \* c;

}

cout << s << endl;

}

/\*回溯TE\*/

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

#define INF 0x3fffffff

typedef struct Node

{

int t,w;

bool operator<(const Node& z)

{

return t\*1.0/w < z.t\*1.0/z.w ;

}

}Node;

Node ns[10000];

int n;

int ans;

int mint,minw;

bool vis[10000];

void dfs(int t,int sum,int step)

{

if(sum>ans)return;

if(step == n)

{

ans = ans<sum?ans:sum;

return;

}

int k = n-step;

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

{

if(!vis[v] && sum+minw\*(k\*t+(k+1)\*k/2\*mint)<ans )

{

vis[v] =1;

dfs(t+ns[v].t,sum+(t+ns[v].t)\*ns[v].w,step+1);

vis[v] = 0;

}

}

}

int main()

{

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

scanf("%d",&n);

int i,j;

mint = INF;

minw = INF;

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

{

scanf("%d",&ns[i].t);

mint = mint<ns[i].t?mint:ns[i].t;

}

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

{

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

minw = minw<ns[i].w?minw:ns[i].w;

}

std::sort(ns,ns+n);

ans = INF;

dfs(0,0,0);

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

return 0;

}

## 5.4 最优旋转游戏问题

迭代加深搜索

#include<iostream>

using namespace std;

#define inf 0x3fffffff

int a[32];

int max\_depth;

int pos[8][7] = {

{0,2,6,11,15,20,22},

{1,3,8,12,17,21,23},

{10,9,8,7,6,5,4,},

{19,18,17,16,15,14,13},

{23,21,17,12,8,3,1},

{22,20,15,11,6,2,0},

{13,14,15,16,17,18,19},

{4,5,6,7,8,9,10}

};

int opposite[8] = {5,4,7,6,1,0,3,2};

int mid[8] = {6,7,8,11,12,15,16,17};

int rdir[8] = {5,4,7,6,1,0,3,2};

char cpath[9] = "ABCDEFGH";

int path[1000000];

bool suc;

inline int max(int x,int y)

{

return x>y?x:y;

}

int get\_h()

{

int cnt[4]={0,0,0,0};

for(int i=0;i<8;++i)

{

++cnt[a[mid[i]]];

}

return 8-max(max(cnt[1],cnt[2]),cnt[3]);

}

void roate(int d)

{

int tmp = a[pos[d][0]];

for(int i=0;i<6;++i)

{

a[pos[d][i]] =a[pos[d][i+1]];

}

a[pos[d][6]] = tmp;

}

int dfs(int depth,int dir)

{

int h,t,minn=inf;

h = get\_h();

if(h == 0)

{

suc = true;

return depth;

}

if(depth+h>max\_depth)return depth+h;

for(int i=0;i<8;++i)

{

if(i == dir)continue;

roate(i);

path[depth]=i;

t = dfs(depth+1,rdir[i]);

if(suc)return t;

minn = minn<t?minn:t;

roate(opposite[i]);

}

return minn;

}

int main()

{

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

int i,T;

scanf("%d",&T);

while(T--)

{

for(i=0;i<24;++i)scanf("%d",a+i);

max\_depth = get\_h();

if(max\_depth == 0)

{

printf("it is OK!\n%d\n",a[mid[0]]);

continue;

}

suc = false;

while(!suc)

{

max\_depth = dfs(0,-1);

}

for(i=0;i<max\_depth;++i)

{

printf("%c",cpath[path[i]]);

}

printf("\n%d\n",a[mid[0]]);

}

return 0;

}

## 5.5 与运算问题

/\*搜索 +剪枝\*/

#include<iostream>

#include<algorithm>

using namespace std;

typedef \_\_int64 llong;

llong minn;

int n,m;

llong a[40];

llong c[40];

void dfs(int cnt,int p,llong v)

{

minn = minn<v?minn:v;

if(cnt == m )return;

if(p<n)

{

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

{

if( (v & c[i]) < minn)

{

dfs(cnt+1,i+1,v&a[i]);

}

}

}

}

int main()

{

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

int i;

for(i=0;i<n;++i)scanf("%I64d",a+i);

sort(a,a+n);

c[n-1]=a[n-1];

for(i=n-2;i>=0;--i)

c[i] = a[i]&c[i+1];

minn = 1;

minn = minn << 62;

dfs(0,0,minn-1);

printf("%I64d\n",minn);

return 0;

}

# 第六章

## 6.1 色子游戏

/\*

算法实验作业6-1 色子游戏问题

?问题描述：

色子游戏是一个在 N\*M 的方格上进行的棋盘游戏。棋盘的每行从北向南被标记为 0 到

N-1，每列从西到东被标记为 0 到 M-1。如下图所示：

棋盘上的每一个方格可以用一个坐标(x，y)表示，x 是行标记，y 是列标记。

每一个色子都是一个立方体，它的六个面分别被标记为 1，2，3，4，5 和 6。其中标记

为 1，2 和 3 的面两两相邻，标记为 6 的面与标记为 1 的面相反，标记为 5 的面与标记为 2 的面相反，标记为 4 的面与标记为 3 的面相反。

色子游戏规则：

色子初始时被放在一个方格上，要移动色子，玩家可以水平或垂直滚动色子到相邻的方

格上。游戏的目的是要求用最少的移动步数，让色子从初始位置移动到目标位置。色子的状

态必须与目标位置的状态相一致。

?编程任务：

用最少的移动步数，让色子从初始位置移动到目标位置并且使得色子的状态与目标位置

的状态相一致。

?数据输入：

输入数据的第一行为两个正整数 N 和 M（1≤N，M≤100），以下 N 行，每行包含 M个字符，字符是'#'或者是'.'，如果是'#'，表示这个方格是一个障碍物，色子无法翻转到这个

方格，如果是'.'，表示这个方格是一个空格，色子可以翻转到这个方格。以上 N\*M 个字符

表示 N\*M 的方格棋盘，色子不能移动到棋盘之外。

以下一行有四个整数 x1，y1，x2，y2，(x1，y1)和(x2，y2)分别表示初始位置坐标和目

标位置坐标。初始位置坐标和目标位置坐标可以看成是'.'的方格。以下两行分别表示初始位

置时色子的状态和目标位置时色子的状态。色子的状态用色子的六个面来表示，每行的六个

数字分别表示色子的上，下，北，南，西，东面。

输入数据有多组，当 N=0 并且 M=0 时，表示输入数据结束。

?结果输出: 将求出的最少的移动步数输出，如果无法从初始点到目标点，输出“-1”。

\*/

#include<iostream>

#include<cstdio>

#include<vector>

#include<queue>

#include<cstring>

using namespace std;

const int N= 110;

const int M= 110;

#define INF 0x3fffffff

int n,m;

char g[N][M];

int gg[N][N][7][7];

typedef struct Node

{

int u,d,n,s,e,w,x,y,v;

bool operator<(const Node& z)const

{

return v>z.v;

}

}Node;

Node s\_node,t\_node;

Node change\_pos(int direct,Node s)

{

Node t = s;

if(direct == 0)

{

t.x = s.x-1;

t.u = s.s;

t.d = s.n;

t.n = s.u;

t.s = s.d;

t.e = s.e;

t.w = s.w;

}

if(direct == 1)

{

t.y = s.y+1;

t.u = s.w;

t.d = s.e;

t.n = s.n;

t.s = s.s;

t.e = s.u;

t.w = s.d;

}

if(direct == 2)

{

t.x = s.x+1;

t.u = s.n;

t.d = s.s;

t.n = s.d;

t.s = s.u;

t.e = s.e;

t.w = s.w;

}

if(direct == 3)

{

t.y = s.y-1;

t.u = s.e;

t.d = s.w;

t.n = s.n;

t.s = s.s;

t.e = s.d;

t.w = s.u;

}

return t;

}

void bfs()

{

int i,j,c,k;

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

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

for(c=1;c <= 6;++c)

for( k=1;k<=6;++k)

gg[i][j][c][k] = INF;

gg[s\_node.x][s\_node.y][s\_node.u][s\_node.n] = 0;

priority\_queue<Node> Q;

Node u,v;

s\_node.v = 0;

Q.push(s\_node);

while(!Q.empty())

{

u = Q.top();

Q.pop();

for(i=0;i<4;++i)

{

v = change\_pos(i,u);

if( g[v.x][v.y]!='#' && gg[v.x][v.y][v.u][v.n] > gg[u.x][u.y][u.u][u.n]+1 )

{

gg[v.x][v.y][v.u][v.n] = gg[u.x][u.y][u.u][u.n]+1;

v.v = gg[v.x][v.y][v.u][v.n];

Q.push(v);

}

}

}

}

void solve()

{

int i,j;

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

g[i][0] = g[i][m+1] = '#';

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

g[0][j] = g[n+1][j] = '#';

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

{

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

{

scanf(" %c",&g[i][j]);

}

}

scanf("%d%d%d%d",&s\_node.x,&s\_node.y,&t\_node.x,&t\_node.y);

s\_node.x+=1;s\_node.y+=1;

t\_node.x+=1;t\_node.y+=1;

scanf("%d%d%d%d%d%d",&s\_node.u,&s\_node.d,&s\_node.n,&s\_node.s,&s\_node.w,&s\_node.e);

scanf("%d%d%d%d%d%d",&t\_node.u,&t\_node.d,&t\_node.n,&t\_node.s,&t\_node.w,&t\_node.e);

bfs();

if(gg[t\_node.x][t\_node.y][t\_node.u][t\_node.n] >= INF)

printf("-1\n");

else

printf("%d\n",gg[t\_node.x][t\_node.y][t\_node.u][t\_node.n]);

}

int main()

{

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

while(scanf("%d%d",&n,&m) && n!=0 && m!=0)

{

solve();

}

return 0;

}

## 6.2 过河问题

一天,小明和他的朋友们在野外郊游。很快,他发现自己在一条河的前面。他

们立即决定去河对岸。幸运的是,有一个船到河边。但是这艘船能容纳人的总重

量最多 k 公斤，船是不会自己动的，也就是说船在载一批人过河后至少要有一个

人把船划回来。于是小明立即统计了他和小伙伴们的体重，并开始制定过河方案。 ★实验任务：

给你一艘载重为 K 公斤的船，以及 n 个人的体重，求出船至少要来回几次，

和把人都载到对案总的方案数。 ★数据输入：

第一行有两个正整数 n,k(1<=n<=50),1(1<=k<=5000)表示人的个数及船的最

大载重量。

第二行 n 个整数，表示 n 个人的体重（每个人的体重不是 50 就是 100 公斤）。 ★结果输出: 对每个测试项，输出两个整数。

第一个整数表示最少的来回次数。如果无解则输出-1. 第二个整数为方案数，是模 1000000007 的结果，如果没有任何一个方案行

Input ouput

3 100 5

50 50 100 2

2 50 -1

50 50 0

#include <stdio.h>

#include <iostream>

#include <string.h>

#include <queue>

using namespace std;

typedef \_\_int64 LL;

#define maxnum 55

const LL NUM=1000000007;

int n,k,sum1=0,sum2=0;

struct Point{

int num1;

int num2;

int s;

};

LL dp[maxnum][maxnum][2];

LL C[maxnum][maxnum];

int mintimes[maxnum][maxnum][2];

void Find(int sum1,int sum2,int value)

{

LL tmp;

int i,j,leftnum1,leftnum2;

queue<Point> q;

Point p,next;

p.num1=sum1;

p.num2=sum2;

p.s=0;

dp[sum1][sum2][0]=1;

mintimes[sum1][sum2][0]=0;

q.push(p);

while(!q.empty())

{

Point cur=q.front();

q.pop();

for(i=0;i<=cur.num1;i++)

{

if(i\*50>value)

break;

for(j=0;j<=cur.num2;j++)

{

if(i\*50+j\*100>value)

continue;

if(i\*50+j\*100<=0)

continue;

leftnum1=cur.num1-i;

leftnum2=cur.num2-j;

if(mintimes[sum1-leftnum1][sum2-leftnum2][!cur.s]==-1)

{

mintimes[sum1-leftnum1][sum2-leftnum2][!cur.s]=mintimes[cur.num1][cur.num2][cur.s]+1;

tmp=C[cur.num1][i]\*C[cur.num2][j]%NUM;

dp[sum1-leftnum1][sum2-leftnum2][!cur.s]+=(tmp\*dp[cur.num1][cur.num2][cur.s]%NUM);

dp[sum1-leftnum1][sum2-leftnum2][!cur.s]%=NUM;

next.num1=sum1-leftnum1;

next.num2=sum2-leftnum2;

next.s=!cur.s;

q.push(next);

}

else if(mintimes[sum1-leftnum1][sum2-leftnum2][!cur.s]==mintimes[cur.num1][cur.num2][cur.s]+1)

{

tmp=C[cur.num1][i]\*C[cur.num2][j]%NUM;

dp[sum1-leftnum1][sum2-leftnum2][!cur.s]+=(tmp\*dp[cur.num1][cur.num2][cur.s]%NUM);

dp[sum1-leftnum1][sum2-leftnum2][!cur.s]%=NUM;

}

}

}

}

}

int main()

{

int i,j,x;

C[0][0]=1;

for(i=1;i<=50;i++)

{

C[i][0] = 1;

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

{

C[i][j]=(C[i-1][j-1]+C[i-1][j])%NUM;

}

}

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

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

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

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

{

scanf("%d",&x);

if(x==50)

sum1++;

else

sum2++;

}

Find(sum1,sum2,k);

printf("%d\n%I64d\n",mintimes[sum1][sum2][1],dp[sum1][sum2][1]);

return 0;

}

## 6.3 骑士救公主1

#include<iostream>

#include<cstdio>

#include<cstring>

#include<queue>

#define INF 0x3fffffff

int n,m;

char g[222][222];

typedef struct Node

{

int x,y,v;

Node(){}

Node(int \_x,int \_y,int \_v):x(\_x),y(\_y),v(\_v){}

bool operator<(const Node& z)const

{

return v>z.v;

}

};

int xdir[4]={-1,0,1,0};

int ydir[4]={0,1,0,-1};

bool vis[222][222];

int gg[222][222];

int bfs()

{

int i,j;

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

{

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

{

vis[i][j] = 0;

gg[i][j] = INF;

}

}

std::priority\_queue<Node> Q;

gg[1][1] = 0;

Q.push(Node(1,1,0));

Node u,v;

while(!Q.empty())

{

u = Q.top();

Q.pop();

vis[u.x][u.y] = 1;

if(u.x == n && u.y == m)

return gg[u.x][u.y];

for(i=0;i<4;++i)

{

v.x = u.x+xdir[i];

v.y = u.y+ydir[i];

if( g[v.x][v.y] != '#' && !vis[v.x][v.y] && gg[v.x][v.y] > gg[u.x][u.y]+g[v.x][v.y]-'0'+1 )

{

gg[v.x][v.y] = gg[u.x][u.y] +g[v.x][v.y]-'0'+1;

v.v = gg[v.x][v.y];

Q.push(v);

}

}

}

return -1;

}

int main()

{

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

int i,j;

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

{

g[i][0] = g[i][m+1] = '#';

}

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

{

g[0][j] = g[n+1][j] = '#';

}

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

{

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

{

scanf(" %c",&g[i][j]);

if(g[i][j] == '.')g[i][j] = '0';

}

}

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

return 0;

}

## 6.4 骑士救公主2

骑士顺利从魔王城堡中救出公主后带着公主逃到了一片森林，然而他们还没

有安全。这片森林也是魔王掌控下的地盘，其中有许多怪物，而这些怪物都很聪

明，它们会尽一切的可能性阻止骑士走出这片森林。为了通过这片森林，骑士一

旦遇上怪物，就必须先杀死怪物才行。

为了简化问题，我们将森林看成一张 N \* M 的网格地图，骑士和怪物每个时

刻都可以选择相邻四个方向的格子移动或者不动。地图上有四种类型的符号：

‘0’ - ‘9’ 表示对应格子上有几只怪物

‘S’ 表示骑士的起点

‘E’ 表示森林出口

‘T’ 表示树，骑士和怪物都不能进入该位置 ★实验任务：

给你一张地图请输出骑士要通过这片森林至少打死几只怪物。 ★数据输入：

第一行有两个正整数 n(1<n<=1000),m(1<m<=1000)表示地图大小。

接下来有 n 行每行有 m 个字符。 ★结果输出: 对每个测试项，输出一个整数，表示打死多少只怪。

5 7 3

000E0T3

T0TT0T0

010T0T0

2T0T0T0

0T0S000

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct Node

{

int l;//层数

int x;

int y;

struct Node \*next;

};

struct Queue

{

struct Node \*front;

struct Node \*rear;

};

void EnQueue(Queue &Q,int px,int py,int pl)//入队

{

struct Node \*pe;

pe=(struct Node\*)malloc(sizeof(struct Node));

pe->l = pl;

pe->x = px;

pe->y = py;

pe->next = NULL;

Q.rear->next=pe;

Q.rear=pe;

}

void DeQueue(Queue &Q,struct Node &pe)//出队，数据存到pe中

{

//空队的时候，在调用时判断

struct Node \*p;

p=Q.front->next;

Q.front->next=p->next;

pe.l=p->l;

pe.x=p->x;

pe.y=p->y;

if(Q.rear==p)

Q.rear = Q.front;

}

int main()

{

int n,m;

int \*\*map;

int flag=0;

int sum=0;

char ch[1000];

int i,j;

int px,py,pl;

//初始化队列

Queue Q;

Q.front = Q.rear = (struct Node\*)malloc(sizeof(struct Node));

Q.front->next=NULL;

struct Node \*pre;

pre = (struct Node\*)malloc(sizeof(struct Node));

pre->next=NULL;

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

map = (int\*\*)malloc(n\*sizeof(\*map));//行

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

{

map[i] = (int\*)malloc(m\*sizeof(int));//列

}

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

{

scanf("%s",ch);

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

{

if('0'==ch[j])// 0

map[i][j]=0;

else if('T'==ch[j])// T

map[i][j]=-1;

else if('S'==ch[j])// S

map[i][j]=-2;

else if('E'==ch[j])// E

{

pre->x=i;

pre->y=j;

pre->l = 1;//第一层

map[i][j]=-1;

}

else //其他

map[i][j]=ch[j]-'0';

}//end j<m

}//end i<n

while(0==flag)

{

px=pre->x;

py=pre->y;

pl=pre->l;

if(px>0&&map[px-1][py]!=-1)//上

{

if(-2==map[px-1][py])

flag=1;

else sum+=map[px-1][py];

map[px-1][py]=-1;

if(flag!=1)

EnQueue(Q,px-1,py,pl+1);//入队

}

if(px<n-1&&map[px+1][py]!=-1)//下

{

if(-2==map[px+1][py])

flag=1;

else sum+=map[px+1][py];

map[px+1][py]=-1;

if(flag!=1)

EnQueue(Q,px+1,py,pl+1);

}

if(py>0&&map[px][py-1]!=-1)//左

{

if(-2==map[px][py-1])

flag=1;

else sum+=map[px][py-1];

map[px][py-1]=-1;

if(flag!=1)

EnQueue(Q,px,py-1,pl+1);

}

if(py<m-1&&map[px][py+1]!=-1)//右

{

if(-2==map[px][py+1])

flag=1;

else sum+=map[px][py+1];

map[px][py+1]=-1;

if(flag!=1)

EnQueue(Q,px,py+1,pl+1);

}

// }//end for i<3

if(1==flag)

{

break;//要不要记下层数

}

DeQueue(Q,\*pre);//出队

}//end while flag=0

//找到骑士了

while(Q.rear!=Q.front)//队列不为空

{

DeQueue(Q,\*pre);

if(pre->l>pl)

break;

px=pre->x;

py=pre->y;

pl=pre->l;

if(px>0&&map[px-1][py]!=-1)//上

{

sum+=map[px-1][py];

map[px-1][py]=-1;

}

if(px<n-1&&map[px+1][py]!=-1)//下

{

sum+=map[px+1][py];

map[px+1][py]=-1;

}

if(py>0&&map[px][py-1]!=-1)//左

{

sum+=map[px][py-1];

map[px][py-1]=-1;

}

if(py<m-1&&map[px][py+1]!=-1)//右

{

sum+=map[px][py+1];

map[px][py+1]=-1;

}

}

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

return 0;

}

## 6.5 机器人扫除问题

#include <iostream>

#include <cstdio>

#include <cstring>

#include <queue>

using namespace std;

const int MAX=106;

const int inf=0xffff;

int Move[4][2]={{0,1},{0,-1},{1,0},{-1,0}};

struct Node

{

int x,y,step;

};

struct Dirty

{

int x,y;

}Num[13];

int n,m,Min,k,Flag[13][13];

char Map[MAX][MAX];

bool flag,Used[13];

bool Visted[MAX][MAX];

queue<Node> Q;

int BFS( int x, int y, int x2, int y2 )

{

while( !Q.empty() )

Q.pop();

memset(Visted,0,sizeof(Visted));

Visted[x][y]=1;

Node p1;

p1.x=x;

p1.y=y;

p1.step=0;

Q.push(p1);

while( !Q.empty() )

{

Node Now=Q.front();

Q.pop();

for( int i=0; i<4; ++i )

{

Node Next;

Next.x=Now.x+Move[i][0];

Next.y=Now.y+Move[i][1];

Next.step=Now.step+1;

if( Next.x<0 || Next.y<0 || Next.x>=n || Next.y>=m

|| Map[Next.x][Next.y]=='x' )

continue;

if( !Visted[Next.x][Next.y] )

{

Visted[Next.x][Next.y]=1;

if( Next.x==x2 && Next.y==y2 )

{

return Next.step;

}

Q.push(Next);

}

}

}

return inf;

}

void DFS( int x, int sum, int len )

{

if( len>Min )

return;

if( sum>=k )

{

Min=Min<len?Min:len;

return;

}

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

{

if( Flag[x][i]!=inf && !Used[i] )

{

Used[i]=1;

DFS(i,sum+1,len+Flag[x][i]);

Used[i]=0;

}

}

}

int main()

{

int i,j;

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

{

if( !n && !m )

break;

k=0;

getchar();

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

{

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

{

scanf("%c",&Map[i][j]);

if( Map[i][j]=='o' )

{

Num[0].x=i;

Num[0].y=j;

Map[i][j]='.';

}

else if( Map[i][j]=='\*' )

{

Num[++k].x=i;

Num[k].y=j;

}

}

getchar();

}

Min=0;

flag=1;

//cout<<k<<endl<<endl;

if( k==0 )

{

printf("0\n");

continue;

}

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

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

{

int temp=BFS(Num[i].x,Num[i].y,Num[j].x,Num[j].y);

Flag[i][j]=Flag[j][i]=temp;

if( temp==inf )

{

flag=0;

break;

}

}

if( !flag )

{

printf("-1\n");

continue;

}

memset(Used,0,sizeof(Used));

Min=inf;

Used[0]=1;

DFS(0,0,0);

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

}

return 0;

}

# 第八章 网络流

## 8.1 排水沟

#include <iostream>

#include <cstdio>

using namespace std;

#define INF 0x3fffffff

const int N= 210;

int m,n;

int Cap[N][N];

int pre[N];

int vis[N];

inline int min(int x,int y)

{

return x<y?x:y;

}

int bfs(int s,int t)

{

int nf[N];

int que[N]; int qtop=0,qrear=0;

int u,v;

nf[s]= INF ; nf[t]=0; //nf[x] : flows in node x

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

que[qrear++]=s;

vis[s]=1;

while(qtop<qrear)

{

u = que[qtop++];

if(u == t)break;

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

{

if(!vis[v] && Cap[u][v]>0)

{

que[qrear++] = v;

vis[v] = 1;

pre[v] = u;

nf[v] = min(nf[u],Cap[u][v]);

}

}

}

return nf[t];

}

int FF(int s,int t) //Ford-Fulkerson

{

int flow=0,inc;

while(1)

{

inc = bfs(s,t);

if(inc==0)return flow;

flow+=inc;

for(int i=t;i!=s;i=pre[i])

{

Cap[i][ pre[i] ] += inc;

Cap[ pre[i] ][i] -= inc;

}

}

return flow;

}

int main()

{

int u,v,c;

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

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

{

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

Cap[u][v] += c;

}

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

return 0;

}

## 8.2 数字梯形

#include<iostream>

#include<cstdio>

#include<cstring>

#include<algorithm>

#include<queue>

const int N = 3000;

#define INF 0x3fffffff

int n;

typedef struct Edge

{

int v,next,c,w;

}Edge;

int ei;

Edge es[N+N+N+N+N];

int head [N];

int pre[N];

int pe[N];

int vis[N];

int g[64][64];

int gid[64][64];

int cost[N];

void addEdge(int u,int v,int c,int w)

{

es[ei].v = v; es[ei].c = c; es[ei].w = w; es[ei].next = head[u]; head[u] = ei++;

es[ei].v = u; es[ei].c = 0; es[ei].w = -w; es[ei].next = head[v]; head[v] = ei++;

}

int bfs(int s,int t,int& res)

{

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

{

cost[i] = INF;

}

std::queue<int > Q;

Q.push(s);

vis[s] = 1;

cost[s] = 0;

int u,v;

pre[t] = -1;

while(!Q.empty())

{

u = Q.front();

Q.pop();

vis[u] = 0;

for(int p = head[u];p!=-1;p=es[p].next)

{

v = es[p].v;

if(es[p].c>0 && cost[v]>cost[u]+es[p].w)

{

pre[v] = u;

pe[v] = p;

cost[v] = cost[u]+es[p].w;

if(!vis[v])

{

Q.push(v);

vis[v] =1;

}

}

}

}

if(pre[t] == -1)return 0;

for(v=t;v!=s;v=pre[v])

{

es[pe[v]].c -= 1;

es[pe[v]^1].c += 1;

}

res+=cost[t];

return 1;

}

int maxflow(int s,int t)

{

int res=0;

while(bfs(s,t,res));

return res;

}

int main()

{

int i,j,l,h,k;

int mx = -INF;

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

int cnt = 0;

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

{

for(j=0;j<i+l;++j)

{

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

gid[i][j] = cnt++;

mx = mx>g[i][j]?mx:g[i][j];

}

}

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

int s= cnt+cnt;

int ss = s+1;

int t= ss+1;

n = t+1;

addEdge(s,ss,k,0);

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

{

addEdge(ss,gid[0][j],INF,0);

}

for(i=0;i<h-1;++i)

{

for(j=0;j<l+i;++j)

{

addEdge(gid[i][j],gid[i][j]+cnt,1,mx-g[i][j]);

addEdge(gid[i][j],gid[i][j]+cnt,INF,mx);

addEdge(gid[i][j]+cnt,gid[i+1][j],1,mx);

addEdge(gid[i][j]+cnt,gid[i+1][j+1],1,mx);

}

}

for(j=0;j<h-1+l;++j)

{

addEdge(gid[i][j],gid[i][j]+cnt,1,mx-g[i][j]);

addEdge(gid[i][j],gid[i][j]+cnt,INF,mx);

addEdge(gid[i][j]+cnt,t,INF,0);

}

printf("%d\n",k\*mx\*(h+h-1) - maxflow(s,t));

return 0;

}

## 8.3 围棋

★问题描述

Alice和Bob一起下围棋，Alice下黑子，Bob下白子。给定n行字符串，每行字符串包含n 个字母，表示n\*n大小的棋盘。黑子用’x’表示，白子用’o’表示，空的格子用’.’表示。给定的

棋局中没有两个相邻的白子，Bob已经认输了，不会再继续下白子了。现在Alice想继续下一

些黑子，使得剩余的空格最多。每次Alice下完黑子，所有被围死的白子会被拿掉。白子被

围死是指其周围没有空的格子。在这个问题中围棋的规则跟正常的围棋规则有些不同，Alice 的黑子永远都不会被白子围死。问最多能得到的空格数是多少？ ★数据输入

输入第一行包含一个整数N (3<=N<=50)，N表示棋盘大小为N\*N。接下来输入N行字符串，

每行包含N个字符。 ★数据输出

输出一个整数，表示最多能得到的空格数。

3 5

o.o

.o.

o.o

#include<cstdio>

#include<cstring>

int n,cnt,s,t;

char str[51][51];

int head[3000],dis[3000],gap[3000],sign[51][51],p[]={-1,0,1,0},q[]={0,1,0,-1};

const int inf=1<<30;

struct EDGE

{

int to,f,nxt;

}edge[15000];

int min(int a,int b) { return a<b?a:b;}

void init()

{

cnt=0;

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

memset(gap,0,sizeof(gap));

memset(dis,0,sizeof(dis));

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

gap[0]=t;

}

void add(int x,int y,int c)

{

edge[cnt].to=y;

edge[cnt].f=c;

edge[cnt].nxt=head[x];

head[x]=cnt++;

}

void gao(int x,int y,int c)

{

add(x,y,c);

add(y,x,0);

}

int dfs(int x,int flow)

{

if(x==t)

return flow;

int temp=flow;

int pos=t-1;

int j;

for(j=head[x];j!=-1;j=edge[j].nxt)

{

int y=edge[j].to;

int c=edge[j].f;

if(c>0&&dis[x]==dis[y]+1)

{

int temp\_flow=dfs(y,min(temp,c));

temp-=temp\_flow;

edge[j].f-=temp\_flow;

edge[j^1].f+=temp\_flow;

if(temp==0||dis[s]==t)

return flow-temp;

}

if(c>0&&pos>dis[y])

pos=dis[y];

}

if(temp==flow)

{

if(!(--gap[dis[x]]))

dis[s]=t;

else

gap[dis[x]=pos+1]++;

}

return flow-temp;

}

int sap()

{

int maxflow=0;

while(dis[s]<t)

{

maxflow+=dfs(s,inf);

}

return maxflow;

}

int main()

{

int i,j,k,sum,num,key;

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

{

s=n\*n+1,t=s+1;

init();

sum=0; num=1;

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

gets(str[i]);

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

{

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

{

if(str[i][j]=='o')

{

sum++;

gao(s,num,1);

key=num;

sign[i][j]=num++;

for(k=0;k<4;k++)

{

if(i+p[k]>=0&&i+p[k]<n&&j+q[k]>=0&&j+q[k]<n&&str[i+p[k]][j+q[k]]=='.')

{

if(sign[i+p[k]][j+q[k]]==-1) sign[i+p[k]][j+q[k]]=num++;

gao(key,sign[i+p[k]][j+q[k]],10000000);

}

}

}

if(str[i][j]=='.') sum++;

}

}

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

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

{

if(str[i][j]=='.'&&sign[i][j]!=-1)

gao(sign[i][j],t,1);

}

printf("%d\n",sum-sap()); } return 0;}

## 8.4 神庙

Alex参观一个古老的寺庙。寺庙中有很多雕像，一些雕像是相同的，总共有不超过26 种类型的雕像。所有雕像排成一个N行M列的方阵。这些雕像需要被摆放成所有行列都相对于

中间位置对称，也就是说对于每一行和每一列，到中间位置距离相等的两个雕像要是同一种

类型。现给出所有雕像的当前摆放位置，问最少需要改变几个雕像的位置，使得所有行列都

相对于中间位置对称。 ★数据输入

输入第一行包含两个整数N,M (2<=N,M<=20)，表示雕像被摆放成N行M列，N和M都是偶

数 。接下来输入N行字符串，每行包含M个字符，表示雕像刚开始的摆放位置，这些字符为

小写字母,不同的字母表示不同类型的雕像。 ★数据输出

输出一个整数，最少需要改变几个雕像的位置，使得所有行列都相对于中间位置对称。

数据保证肯定能使得所有行列相对于中间位置对称。

4 4 2

abxa

xyyb

xyyx

abba

#include <stdio.h>

#include <string.h>

#define M 1000

#define inf 0x7fffffff

int link[M],lx[M],ly[M],slack[M];

int visx[M],visy[M],w[M][M];

char s[25][25],d[150];

int nx,ny;

int sign[26];

int DFS(int x)

{

visx[x] = 1;

for (int y = 1;y <= ny;y ++)

{

if (visy[y])

continue;

int t = lx[x] + ly[y] - w[x][y];

if (t == 0)

{

visy[y] = 1;

if (link[y] == -1||DFS(link[y]))

{

link[y] = x;

return 1;

}

}

else if (slack[y] > t)

slack[y] = t;

}

return 0;

}

int KM()

{

int i,j;

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

memset (ly,0,sizeof(ly));

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

for (j = 1,lx[i] = -inf;j <= ny;j ++)

if (w[i][j] > lx[i])

lx[i] = w[i][j];

for (int x = 1;x <= nx;x ++)

{

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

slack[i] = inf;

while (1)

{

memset (visx,0,sizeof(visx));

memset (visy,0,sizeof(visy));

if (DFS(x))

break;

int pp= inf;

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

if (!visy[i]&&pp > slack[i])

pp = slack[i];

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

if (visx[i])

lx[i] -= pp;

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

if (visy[i])

ly[i] += pp;

else

slack[i] -= pp;

}

}

int res = 0;

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

if (link[i] > -1)

res += w[link[i]][i];

return res;

}

int main()

{

int nn,mm,k1,k2,i,j,k,count;

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

{

nx=(mm\*nn)/4;ny=(mm\*nn)/4;

k1=0;k2=0;

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

scanf("%s",s[i]+1);

memset(sign,0,sizeof(sign));

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

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

sign[s[i][j]-'a']++;

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

if(sign[i])

{

while(sign[i])

{

d[++k2]='a'+i;

sign[i]-=4;

}

}

for(i=1;i<=nn/2;i++)

{

for(j=1;j<=mm/2;j++)

{

k1++;

for(k=1;k<=k2;k++)

{

count=0;

if(s[i][j]!=d[k]) count++;

if(s[i][mm-j+1]!=d[k]) count++;

if(s[nn-i+1][j]!=d[k]) count++;

if(s[nn-i+1][mm-j+1]!=d[k]) count++;

w[k1][k]=-count;

}

}

}

int ans = KM();

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

}

return 0;

}

## 8.5 奶牛

#include<iostream>

#include<queue>

#include<cstring>

#include<algorithm>

#include<cstdio>

using namespace std;

#define INF 0x3fffffff

const int N = 410;

typedef struct Edge

{

int v,next,c;

}Edge;

int ei;

Edge es[60000];

int head[N];

void addEdge(int u,int v,int c)

{

es[ei].v = v;es[ei].c = c;es[ei].next = head[u]; head[u]=ei++;

es[ei].v = u;es[ei].c = 0;es[ei].next = head[v]; head[v]=ei++;

}

int n,mf,md,nn;

int pre[N];

int pe[N];

int vis[N];

int bfs(int s,int t)

{

queue<int> Q;

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

Q.push(s);

vis[s]= 1;

int u,v;

pre[t] = -1;

while(!Q.empty())

{

u = Q.front();

Q.pop();

if(u == t)break;

for(int p = head[u];p!=-1;p=es[p].next)

{

v = es[p].v;

if(!vis[v] && es[p].c > 0)

{

pre[v]=u;

pe[v] = p;

Q.push(v);

vis[v]=1;

}

}

}

if(pre[t]==-1)return 0;

int inc = INF;

for(v=t;v!=s;v=pre[v])

{

inc = inc<es[pe[v]].c?inc:es[pe[v]].c;

}

for(v=t;v!=s;v=pre[v])

{

es[pe[v]].c -= inc;

es[pe[v]^1].c += inc;

}

return inc;

}

int maxflow(int s,int t)

{

int f = 0,inc;

while(1)

{

inc = bfs(s,t);

if(inc == 0)return f;

f+=inc;

}

return f;

}

int main()

{

int i,j,fi,di,x;

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

{

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

int s= 0,t=mf+n+n+md+1;

nn = t+1;

ei = 0;

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

{

addEdge(s,i,1);

}

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

{

scanf("%d%d",&fi,&di);

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

{

scanf("%d",&x);

addEdge(x,mf+i,1);

}

addEdge(i+mf,i+mf+n,1);

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

{

scanf("%d",&x);

addEdge(i+mf+n,mf+n+n+x,1);

}

}

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

addEdge(mf+n+n+i,t,1);

printf("%d\n",maxflow(s,t));

}

return 0;

}