P1526 [NOI2003] 智破连环阵 题解

返回题目

本题目已不接受新题解提交

当题目的题解数量、做题思路已足够丰富，题目过于简单，或处于月赛保护期时，题解提交入口会被关闭。

不要把题解发布在题目讨论区。

题解仅供学习参考使用

抄袭、复制题解，以达到刷 AC 率/AC 数量或其他目的的行为，在洛谷是严格禁止的。

洛谷非常重视**学术诚信**。此类行为将会导致您成为**作弊者**。 具体细则请查看洛谷社区规则。

提交题解前请务必阅读《洛谷主题库题解规范》。

7 篇题解 默认排序

George1123 创建时间：2020-09-28 13:10:19 在 Ta 的

更木棒的阅读体验 /se → George1123

题面

> NOI2003 智破连环阵

> 有 m 个靶子 (ax_j, ay_j) 和 n 个箭塔 (bx_i, by_i)。每个箭塔可以射中距离在 k 以内的靶子。第 $i+1$ 只有第 i 个靶子被射中时才能被射中
> 个箭塔只能用一次，现在可以安排每个箭塔的射击顺序，求最少需要几个箭塔可以射光 m 靶子。

> 数据范围：$1 \le m, n \le 100$，$1 \le k \le 1000$，$1 \le ax_j, ay_j, bx_i, by_i \le 10000$。

蒟蒻语

爆搜神题，可惜题解都很晦涩，蒟蒻因为一个小错误折腾了一个晚上，现在拿到了最优解，于是准备写个逊逊的题解。

蒟蒻解

首先每个箭塔解决一个靶子区间。

所以可以爆搜每个区间和箭塔匹配，这很明显是个二分图匹配。

为了方便处理很多细节，设所有 i 为箭塔的下标，j 为靶子的下标。

设 bool $con_{i,j}$ 表示箭塔 i 与靶子 j 联通。

由于每个箭塔的每个负责区间只需用后缀就可以有解，所以记录 $nex_{i,j}$ 表示箭塔 i 在靶子 j 后面第一个射不到的靶子（即可用射到最右边的
$+1$）。

```
// 这是一个很显然的递推
R(i,0,n)L(j,0,m) con[i][j]&&(nex[i][j]=max(j+1,nex[i][j+1]));
```

为了后面 A* 做准备，还可以求出一个 mn_j 表示打到靶子 j 的剩余步数下限。

```
L(j,0,m)R(i,0,n) con[i][j]&&(mn[j]=min(mn[j],mn[nex[i][j]]+1));
```

然后就可以开始惊心动魄的 Dfs 了。

最直接的方法是先用 mn_j 来剪枝 A* 一下，然后用 $nex_{i,j}$ 枚举下一个区间端点，用过的箭塔打个标记，匹配一个没用过的箭塔。

前文说过这是个二分图匹配，所以有个野蛮操作（二分图优化）：每次区间找好后，直接匈牙利匹配看看能不能匹配得到箭塔。

这个操作时间复杂度比起原来操作是不增的。

但是这有什么用呢？要配上另一个骚操作：逆序枚举下一个区间开始端点。

由于用了匈牙利后完美匹配概率变高，所以就可以尽早找到优的答案，进一步 A* 剪枝。

然后就结束了，时限 $2s$ 的题跑得最慢的点 $4ms$，总时间 $31ms$。

注意 Dfs 回溯算法两个坑：回溯不彻底、回溯用了全局变量。

代码

```
#include <bits/stdc++.h>
using namespace std;
```

```cpp
//Start
typedef long long ll;
typedef double db;
#define mp(a,b) make_pair((a),(b))
#define x first
#define y second
#define be(a) (a).begin()
#define en(a) (a).end()
#define sz(a) int((a).size())
#define pb(a) push_back(a)
#define R(i,a,b) for(int i=(a),I=(b);i<I;i++)
#define L(i,a,b) for(int i=(b)-1,I=(a)-1;i>I;i--)
const int iinf=0x3f3f3f3f;
const ll linf=0x3f3f3f3f3f3f3f3f;

/*
注意： i 是箭塔，j 是靶子，s 是区间
*/

//Data
const int N=1e2;
int m,n,k;
pair<int,int> a[N],b[N];
bitset<N> con[N];
#define f(x) ((x)*(x))

//Dfs
bitset<N> e[N],vis;
int nex[N][N+1],mn[N+1],mat[N],ans;
bool match(int s){ // 匈牙利匹配
    R(i,0,n)if(e[s][i]&&!vis[i]){
        vis[i]=true;
        if(!~mat[i]||match(mat[i]))
            return mat[i]=s,true;
    }
    return false;
}
void dfs(int j,int s){
    if(ans<=s+mn[j]) return; //A*
    if(j==m) return void(ans=s);
    int cmat[N]; copy(mat,mat+n,cmat); // 这里的 cmat 你要是设为全局变量就死了，我在这里死了 2 个小时
    L(J,j+1,m+1){
        R(i,0,n) con[i][j]&&nex[i][j]>=J&&(e[s][i]=true);
        R(i,0,n) vis[i]=false; match(s)?dfs(J,s+1):void();
        R(i,0,n) con[i][j]&&nex[i][j]>=J&&(e[s][i]=false); //莫忘回溯
        copy(cmat,cmat+n,mat);
    }
}

//Main
int main(){
    ios::sync_with_stdio(0);
    cin.tie(0),cout.tie(0);
    cin>>m>>n>>k;
    R(j,0,m) cin>>a[j].x>>a[j].y;
    R(i,0,n) cin>>b[i].x>>b[i].y;
    R(i,0,n)R(j,0,m) con[i][j]=(f(a[j].x-b[i].x)+f(a[j].y-b[i].y)<=f(k));
    R(i,0,n) fill(nex[i],nex[i]+m+1,-1);
    R(i,0,n)L(j,0,m) con[i][j]&&(nex[i][j]=max(j+1,nex[i][j+1]));
    R(j,0,m) mn[j]=iinf;
    L(j,0,m)R(i,0,n) con[i][j]&&(mn[j]=min(mn[j],mn[nex[i][j]]+1));
    fill(mat,mat+n,-1),ans=min(n,m),dfs(0,0);
    // 夹杂点骚操作（正确性不保证，仅用来抢最优解：猜测最终 ans<=mn[0]+5），把 ans 的初始值和 mn[0]+5 取 min
    cout<<ans<<'\n';
    return 0;
}
```

祝大家学习愉快！

33 11 条评论

智破连环阵

这题是黑题~~但水的一批~~。

题目描述

> 给你 m 个武器的坐标、n 个炸弹的坐标，用炸弹炸范围半径为 k 内的武器，炸弹用完了就没了，可以改变炸弹顺序，求最少要多少炸弹武器炸完。

题目链接

首先发现每一个炸弹能炸的武器一定是编号连续的。

这是因为只有前一个武器被摧毁，后一个才能被摧毁，并且一个炸弹能持续5分钟，这注定了每个炸弹一定能将**能摧毁的都摧毁掉**。

用 dfs 的话，可以将 $1 \backsim m$ 个武器分为若干小块，每个小块由各自一个的炸弹炸。

那么枚举在哪里分段。

那么先维护出来每一个炸弹**能炸到的武器**，用你初中学过的勾股即可。

然后就可以通过**递推**方式求出每一个炸弹能炸到的**编号最大**的武器（正着枚举炸弹，倒着枚举武器，显然第 i 个炸弹炸到第 j 个武器时，能炸来这个炸弹能炸到的最远的武器）。

已经递推出每个炸弹能炸的区间了。~~就可以愉快的 dfs 子~~。

当然要剪枝：

设一个数组 g。

用来计：在炸弹可以无限重复用的情况下，从第 i 个武器到最后一个武器最少用多少个炸弹，即预估的还要用的最少炸弹数。

这样的话，当我们 dfs 到一个新的状态时，只需要提前判断当前已用的炸弹加上预估的还要的最少的炸弹数是否小于已有的答案，若否，则即可。

我们发现有有许多炸弹对应一段区间的情况。

用一个二分图最大匹配即可解决。

于是流程就是：处理处每个炸弹能炸的武器、地推出每个炸弹最远能炸多远、从每个位置炸到最后最少用的炸弹数、 dfs 、二分图最大匹配。

~~于是你就可以水掉他子~~。

代码注释见：

```
#include<bits/stdc++.h>
using namespace std;
int read ()
{
    int x = 0 , f = 1 ; char ch = getchar () ;
    while ( ! isdigit ( ch ) ) { if ( ch == '-' ) f = -1 ; ch = getchar () ; }
    while ( isdigit ( ch ) ) { x = ( x << 3 ) + ( x << 1 ) + ch - 48 ; ch = getchar () ; }
    return x * f ;
}

int m , n , k , ans , tmp [ 101 ] , f [ 101 ] [ 101 ] , g [ 101 ] , J [ 101 ] [ 101 ] ;
bool vis [ 101 ] , fl [ 101 ] [ 101 ] ;

struct node {
    int x , y ;
} a [ 101 ] , b [ 101 ] ;

int fang ( int x )
{
    return x * x ;
}
bool found ( int x ) // 二分图最大匹配
{
    for ( int i = 1 ; i <= n ; i ++ )
        if ( fl [ i ] [ x ] && vis [ i ] )
        {
            vis [ i ] = 0 ;
```

```
            if ( tmp [ i ] == -1 || found ( tmp [ i ] ) )
            {
                tmp [ i ] = x ;
                return 1 ;
            }
        }
    return 0 ;
}
void dfs ( int x , int y ) // 分到第 x 个武器，已经分了 y 个块
{
     if ( y + g [ x ] >= ans ) return ; // 提前预知不优，返回
     if ( x > m )
     {
        ans = y ; // 更新答案
        return ;
     }
     int tmpp [ 101 ] ;
     for ( int i = x ; i <= m ; i ++ ) // 枚举从哪里分开
     {
        for ( int j = 1 ; j <= n ; j ++ )
        {
            tmpp [ j ] = tmp [ j ] ; // 记录副本，方便还原现场
            if ( J [ j ] [ x ] && f [ j ] [ x ] >= i )
            fl [ j ] [ y + 1 ] = 1 ;
        }
        memset ( vis , 1 , sizeof ( vis ) ) ;
        if ( found ( y + 1 ) ) dfs ( i + 1 , y + 1 ) ; // 继续递归
        for ( int j = 1 ; j <= n ; j ++ ) // 还原现场
        {
            tmp [ j ] = tmpp [ j ] ;
            if ( J [ j ] [ x ] == 1 && f [ j ] [ x ] >= i )
                fl [ j ] [ y + 1 ] = 0 ;
        }
    }
}

int main ()
{
    m = read () ; n = read () ; k = read () ;
    for ( int i = 1 ; i <= m ; i ++ ) b [ i ] . x = read () , b [ i ] . y = read () ;
    for ( int i = 1 ; i <= n ; i ++ ) a [ i ] . x = read () , a [ i ] . y = read () ;
    for ( int i = 1 ; i <= m ; i ++ )
        for ( int j = 1 ; j <= n ; j ++ )
            if ( fang ( a [ j ] . x - b [ i ] . x ) + fang ( a [ j ] . y - b [ i ] . y ) <= fang ( k ) ) J [ j ] [ i ] = 1 ; // 判断能否炸
    for ( int i = 1 ; i <= n ; i ++ )
        for ( int j = m ; j >= 1 ; j -- )
            if ( J [ i ] [ j ] ) f [ i ] [ j ] = max ( j , f [ i ] [ j + 1 ] ) ; //递推求最大能炸到的武器
    memset ( g , 0x3f , sizeof ( g ) ) ;
    g [ m + 1 ] = 0 ;
    for ( int i = m ; i >= 1 ; i -- )
        for ( int j = 1 ; j <= n ; j ++ )
            if ( J [ j ] [ i ] ) g [ i ] = min ( g [ i ] , g [ f [ j ] [ i ] + 1 ] + 1 ) ; // 递推求从该点武器炸到最后武器最少用的炸弹
    ans = n ; // 最差要用 n 个炸弹，因此初始化为 n
    memset ( tmp , -1 , sizeof ( tmp ) ) ;
    dfs ( 1 , 0 ) ;  // dfs
    cout << ans << endl ;
    return 0 ;
}
```

后记

这是我第一次写题解，望通过 QwQ。

10 11 条评论

FelFa_1414666 创建时间：2022-07-06 13:51:21 在 Ta 的

传送门

根据勾股定理：

> 一个武器 (x,y) 能被炸弹 (u,v) 攻击到 $\Leftrightarrow (x-u)^2+(y-v)^2 \le k^2$

通过这个条件，对每个炸弹求出一个 m 位的 bitset 表示范围内武器集合。

我们考虑简化摧毁武器的过程，发现每次都是摧毁一段连续区间内的武器。即：将 m 个武器分成若干个连续区间，按顺序每次摧毁其中 1 个

那么就想到**搜索**。搜索区间的划分方案。

- $\mathrm{dfs}(i,cnt,lst)$ 表示考虑到第 i 个位置，当前区间左端点是 lst，lst 之前划分了 cnt 个区间的最小答案。
- 每次考虑当前是否开一个新区间，转移到 $\mathrm{dfs}(i+1,cnt,lst)$ 和 $\mathrm{dfs}(i+1,cnt+1,i+1)$。

考虑对于一个区间划分方案，每个区间都对应一个唯一的炸弹。那么可以将 n 个炸弹和 cnt 个区间看作节点。若炸弹可以摧毁一个区间，则
边，构造**二分图**。若：二分图最大匹配数 = cnt，那么这个区间划分方案就是合法的，使用炸弹个数 = cnt。

对于二分图的匹配，我们用**匈牙利算法**解决。~~这里就不解释匈牙利算法了，可以自行搜索。~~ 在搜索的同时，每次新开一个区间 $[lst,i]$，就用
法去找是否存在炸弹可以和这个区间匹配。如果是，就可以转移。最后答案就是最小区间个数。

以上是搜索的思路，在实现时还要加入优化。

1. 可以提前 $O(nm^2)$ 预处理,对于每个区间 $[l,r]$，包含 $[l,r]$ 的炸弹列表。若当前要将 $[lst,i]$ 划成一个区间，在二分图上就对应这个列表
炸弹，实时匈牙利算法 check 一下，如果 $\mathrm{check}(lst,i)=false$，那么就剪枝。因为当前 $[lst,i]$ 无法匹配，说明后面 $[lst,i+j],\forall 0<=j<=m-i$ 也一定无法匹配。不存在合法方案。
2. A* 剪枝，倒序预处理 $mn[1...m]$，表示从 i 开始，$[i,m]$ 这个区间内所有武器最小要用几个炸弹爆破（相同炸弹可以多次使用）。可以
暴力预处理 mn 数组，dfs 时若 $cnt+mn[lst]>=ans$，其中 ans 表示当前最优答案。说明答案不会更优，进行最优性剪枝。
3. 每个 dfs 状态，如果 $\mathrm{check}(lst,i)=true$，可以转移，那么先搜索 $\mathrm{dfs}(i+1,cnt,lst)$，再搜索 $\mathrm{dfs}(i+1,cnt+1,i+1)$，这样
搜到最优答案。

加上这些优化后就能高效地 ~~水过~~ 通过。

Code

```
#include <bits/stdc++.h>
#define ll long long
#define pb push_back
#define pii pair<int,int>
#define mp make_pair
#define F first
#define S second
using namespace std;
int n,m,K,x[205],y[205],to[105][105],mn[105],ans;//to表示二分图匹配中区间对应的炸弹
pii mch[105];//炸弹在二分图匹配中对应的区间
bitset<105> a[105];
vector<int> v[105][105];
bool used[105],link[105][105];
bool check(int l,int r,bool f=1)//匈牙利算法核心代码
{
    if (f)//初始化used数组
        fill(used,used+m,0);
    for(auto u:v[l][r])
        if (!used[u])
        {
            used[u]=1;
            if (mch[u].F==-1||check(mch[u].F,mch[u].S,0))//如果当前炸弹没有用过或其对应的区间可以用别的炸弹处理，则可以用这个炸弹处理当前[
            {
                mch[u]=mp(l,r);
                to[l][r]=u;
                return 1;
            }
        }
    return 0;
}
void dfs(int i,int cnt,int lst)
{
    if (i==n-1)
    {
        if (check(lst,i))
        {
            ans=min(ans,cnt+1);
```

```
            mch[to[lst][i]]=mp(-1,-1);
        }
        return;
    }
    if (cnt+mn[lst]>=ans)//A*最优性剪枝
        return;
    if (check(lst,i))//可以转移
    {
        mch[to[lst][i]]=mp(-1,-1);
        dfs(i+1,cnt,lst);//不新增区间
    }
    if (check(lst,i))
    {
        dfs(i+1,cnt+1,i+1);//新增区间
        mch[to[lst][i]]=mp(-1,-1);//回溯
    }
}
int main()
{
    ios::sync_with_stdio(false),cin.tie(nullptr);
    cin>>n>>m>>K;//代码n和m的定义与题面相反
    for(int i=0;i<n+m;i++)
        cin>>x[i]>>y[i];
    for(int i=0;i<m;i++)
        for(int j=0;j<n;j++)
            if ((x[n+i]-x[j])*(x[n+i]-x[j])+(y[n+i]-y[j])*(y[n+i]-y[j])<=K*K)
                a[i][j]=1;//处理bitset
    for(int i=0;i<m;i++)//预处理区间[l,r]可以对应的炸弹列表
    {
        for(int l=0;l<n;l++)
            for(int r=0;r<n;r++)
                link[l][r]=0;
        for(int l=0;l<n;l++)
            if (a[i][l])
            {
                link[l][l]=1;
                v[l][l].pb(i);
            }
        for(int r=1;r<n;r++)
            for(int l=r-1;l>=0;l--)
                if (link[l][r-1]&&a[i][r])
                {
                    link[l][r]=1;
                    v[l][r].pb(i);
                }
    }
    for(int i=n-1,j=n-1,cnt=0;i>=0;i--)//预处理mn数组
    {
        while(j>=i)
        {
            int p=j;
            while(p>=0&&!v[p][j].empty())//贪心，每次取可能大的区间覆盖当前位置
                --p;
            j=p;
            ++cnt;
        }
        mn[i]=cnt;
    }
    memset(mch,-1,sizeof(mch));
    memset(to,-1,sizeof(to));
    ans=m;
    dfs(0,0,0);
    cout<<ans<<endl;
    return 0;
}
```

5 3 条评论

Supor__Shoop 创建时间：2022-02-09 14:40:11 在 Ta 的

这道题放在黑题里面其实还比较简单，因为除了搜索和剪枝，并没有涉及到多少其它的黑题难度的知识点。所以说只要把搜索和剪枝掌握到这地步就能做对，更何况是我这样的初一蒟蒻呢？

翻译一下题目：有 n 个点需要被覆盖，你有 m 个半径为 k 的圆，每个圆覆盖的点是圆内（包括边界）编号连续的一些点，且每个圆只能使用
问至少要选择几个圆才能覆盖所有点。

首先我讲一下并不能过的普通搜索，想必我上战场的时候也最多达到这种地步。剪枝我借鉴了一下大佬的题解才对的。大家不要学习这种坏

其实本题可以分而治之，变成两个子问题。第一个是将 n 个点根据编号拆成 k 段，第二个则是判断 m 个圆与这 k 段的最大匹配是否是 k。

仔细看，发现第二个子问题相对来说比较好解决，就是用二分图完成，详细一点就是二分图的最大匹配，这个主要作用还是有一点的优化成

二分图详情

然而第一个子问题我们只能硬搜索。

不过这样的搜索时间复杂度可达 $O(2^m)$，可谓是相当劣质了。

这个代码我给大家品一品：

```
void dfs(int now,int sum)
{
    if(now>n)
    {
        ans=sum;
        return;
    }
    int C[105];
    int top;
    for(int i=n;i>=now;i--)
    {
        if(!have[now][i]) continue;
        top=0;
        for(int j=1;j<=m;j++)
        {
            if(mx[j][now]==i)
            {
                M[sum+1][j]=1;
                C[++top]=j;
            }
        }
        memset(vis,0,sizeof(vis));
        if(find(sum+1)) dfs(i+1,sum+1);
    }
}
```

难以置信的是没有超时的点。

搜索剪枝这个东西对我而言比较靠水平。

第一个剪枝我一下子就想到了动态，求出后面几个点至少需要几个圆覆盖，并以此来进行最优性剪枝。然后一边搜索，一边匹配判断此时的
开行即可。

可是我发现它并没有什么卵用。

于是就有了另一个的剪枝。

这里利用一下贪心思想：如果圆 a 覆盖了点 i 且包含点 $i+1$，那么一定存在一组最优解中 a 覆盖点 $i+1$。所以这个就可以用来当一个剪枝

然后就可以过了。

完整代码如下：

```
#include<bits/stdc++.h>
using namespace std;
int n,m,k;
int mx[105][105];
int ans;
int need[105];
int P[105];
bool ok[105][105],have[105][105],M[105][105],vis[105];
struct node
{
    int x,y;
}wq[105],zd[105];
inline int len(node u,node v)
{
```

```
    return (u.x-v.x)*(u.x-v.x)+(u.y-v.y)*(u.y-v.y);
}
bool find(int x)
{
    for(int i=1;i<=m;i++)
    {
        if(vis[i]||!M[x][i])   continue;
        vis[i]=1;
        if(!P[i]||find(P[i]))
        {
            P[i]=x;
            return 1;
        }
    }
    return 0;
}
void dfs(int now,int sum)
{
    if(need[now]+sum>ans)   return;//贪心的剪枝
    if(now>n)
    {
        ans=sum;
        return;
    }
    int C[105];
    int top;
    for(int i=n;i>=now;i--)
    {
        if(!have[now][i]) continue;
        top=0;
        for(int j=1;j<=m;j++)
        {
            if(mx[j][now]==i)
            {
                M[sum+1][j]=1;
                C[++top]=j;
            }
        }
        memset(vis,0,sizeof(vis));
        if(find(sum+1)) dfs(i+1,sum+1);//关键的搜索
        for(int j=1;j<=top;j++)
        {
            M[sum+1][C[j]]=0;
            if(P[C[j]]==sum+1) P[C[j]]=0;
        }
    }
}
int main()
{
    memset(need,0x3f,sizeof(need));
    cin>>n>>m>>k;
    k*=k;
    for(int i=1;i<=n;i++) cin>>wq[i].x>>wq[i].y;
    for(int i=1;i<=m;i++) cin>>zd[i].x>>zd[i].y;
    need[n+1]=0;
    for(int j=n;j>=1;j--)
    {
        for(int i=1;i<=m;i++)
        {
            if(len(zd[i],wq[j])<=k)
            {
                ok[i][j]=1;
                ok[i][j+1]?mx[i][j]=mx[i][j+1]:mx[i][j]=j;
                have[j][mx[i][j]]=1;
                need[j]=min(need[j],need[mx[i][j]+1]+1);
            }
        }
    }
    ans=min(n,m);
    dfs(1,0);
    //1就是now，0就是sum初始化
    cout<<ans;
    return 0;
}
```

应用
题库
题单
比赛
记录
讨论

4 2 条评论

不存在之人 创建时间：2018-06-13 20:47:52 在 Ta 的

感觉和之前百度之星那题是差不多套路吧，都是搜索套上一个二分图优化

很容易知道，每个炸弹都是炸掉连续的一段

然后你就搜索一下哪些段

然后二分图判定一下

注意要边搜索边判定，这样是一个强大的可行性剪枝，然后就可以跑过去了

其实在分段的时候也有技巧，就是肯定是在每个炸弹端点的时候分，但是我比较懒，没有加，反正也过了

因为这题数据随机

```
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<iostream>
#include<queue>
using namespace std;
const int N=105;
int m,n,k;//武器   炸弹   攻击范围
struct qq
{
    int x,y;
}a[N],b[N];
int dis (qq a,qq b)
{
    return (a.x-b.x)*(a.x-b.x)+(a.y-b.y)*(a.y-b.y);
}
int ans;
bool ok[N][N];
bool s[N][N];//能不能炸到
int tt[N][N];//这个点如果是true的话下一个是什么
int f[N];//这个炸弹配对了谁
bool vis[N];
bool find (int x)
{
    for (int u=1;u<=n;u++)
        if (ok[u][x]&&vis[u])
        {
            vis[u]=false;
            if (f[u]==-1||find(f[u]))
            {
                f[u]=x;
                return true;
            }
        }
    return false;
}
int dist[N];
void dfs (int y,int z)//划分到第几个点   划分了多少段
{
    if (z+dist[y]>=ans) return ;
    if (y>m)
    {
        ans=z;
        return ;
    }
    int ff[N];
    for (int i=y;i<=m;i++)//这一段划在哪里
    {
        for (int j=1;j<=n;j++)
        {
            ff[j]=f[j];
            if (s[j][y]==true&&tt[j][y]>=i)
                ok[j][z+1]=true;
        }
        memset(vis,true,sizeof(vis));
```

```
        if (find(z+1)) dfs(i+1,z+1);
        for (int j=1;j<=n;j++)
        {
            f[j]=ff[j];
            if (s[j][y]==true&&tt[j][y]>=i)
                ok[j][z+1]=false;
        }
    }
}
int main()
{
    scanf("%d%d%d",&m,&n,&k);
    for (int u=1;u<=m;u++)  scanf("%d%d",&a[u].x,&a[u].y);
    for (int u=1;u<=n;u++)  scanf("%d%d",&b[u].x,&b[u].y);
    memset(s,false,sizeof(s));
    for (int u=1;u<=n;u++)
        for (int i=1;i<=m;i++)
            if (dis(b[u],a[i])<=k*k)
                s[u][i]=true;
    for (int u=1;u<=n;u++)
        for (int i=m;i>=1;i--)
        {
            if (s[u][i]==false) continue;
            tt[u][i]=max(i,tt[u][i+1]);
        }
    memset(dist,127,sizeof(dist));
    dist[m+1]=0;
    for (int u=m;u>=1;u--)
    {
        for (int i=1;i<=n;i++)
        {
            if (s[i][u]==false) continue;
            dist[u]=min(dist[u],dist[tt[i][u]+1]+1);
        }
    }
    memset(f,-1,sizeof(f));
    memset(ok,false,sizeof(ok));
    ans=n;
    dfs(1,0);
    printf("%d\n",ans);
    return 0;
}
```

4 2 条评论

21002tyj 创建时间：2013-10-30 12:54:44 在 Ta 的

初步分析：

A国炸弹I可以炸到B国武器J的条件: (u[i]-x[j])2+(v[i]-y[j])2<=r2

结论：很难找到求最优解的多项式算法。

面对此类问题，一般只有搜索策略。

进一步分析：

每一颗炸弹必定炸掉B国武器中编号连续的一段。

5分钟只是表明每一颗炸弹可以炸掉任意多个编号连续的B国武器。

部分搜索：

此题使用部分搜索的算法需要一些转化：如果已经将B国武器根据编号分为x

段，[Si,Ti] (S1=1,Ti>=Si,Ti+1=Si+1)。然后判断是否可以从A国的N颗炸弹中选出x

颗，分别可以炸掉其中的一段。

其实我们把搜索分为了两部分，先通过搜索将B国武器根据编号分为x段，再

通过搜索判断是否可以从A国的N颗炸弹中选出x颗，分别可以炸掉其中的一段。

其实第二部分可以用匹配来解决。

C[T] 表示A国炸弹I是否可以炸到B国武器S,S+1..T-1,T。

C=((u-x)2+(v-y)2<=R2)

C[T]=C[T-1] & C[T][T] (S<T)

求C的时间复杂度为O(N3)。

建图：左边x个点，表示B国武器根据编号分为的x段，右边N个点，表示A国的N颗炸弹。左边第i个点到右边第j个点有边的条件即：C[Si][Ti][j]。

搜索的任务就是将B国武器根据编号划分为若干段+二分图匹配判断。

性能分析（1）：

搜索的基本框架已经建立，虽然数据是随机生成的，但是M个B国武器的划分方案还是非常多的，有时可能高达2m。时间上很难承受，如果使用卡时，正确性受到影响，效果不会很好。

只有4个数据可以在时限内出解，另外6个如果卡时，有1个也可以得到最优解。 优化：

优化可以通过可行性和最优性两方面分析。

优化一（最优性）：如果A国炸弹可以重复使用，设：

Dist[i]=炸掉B国武器i - m的最少使用炸弹数。可以用动态规划计算Dist值，状态转移方程如下：

Dist[m+1]=0。

Dist[i]=Min(Dist[j]+1 | Can[i][j-1]k) (1<=i<=n)

(i<j<=n+1)

求Dist的时间复杂度为O(N3)。

从而产生了一个最优性剪枝条件：

if 当前已经使用的炸弹数+Dist[当前已经炸掉的B国武器数+1]>=当前找到的最优解 then 剪枝；

优化二（可行性）：

此搜索方法一般都可以用两个效果很好的可行性优化：

(1)提前判断是否可以匹配成功，避免多余的搜索。

(2)每次匹配可以从以前的匹配开始扩展，不需要重新开始。

如果当前的划分方法已经无法匹配成功，就没有搜索下去的必要了，只要每搜索新的一段时立即通过匹配判断即可。

每次求匹配只要从原来的基础上扩展就可以了。

通过上述两个优化，程序的效率有了很大的提高。

性能分析（2）：

虽然通过上述两个优化，程序的效率较原来的搜索有了很大的提高。10个测试数据中有8个可以在时限内出解，另外2个如果卡时，有1个也可以得到最优解。

进一步优化：

优化二虽然排除了许多不必要的划分，但是在判断时浪费了不少时间。

因此，在枚举划分长度时，可以通过以前的划分和匹配情况（被匹配的边），用O(n2)的时间复杂度的宽度优先搜索计算出下一个划分的最大长度maxL，显然

应用
题库
题单
比赛
记录
讨论

下一个划分的长度在[1，maxL]都一定可以找到可行的匹配。这样既节省了判断的时间，又可以使每次划分长度从长到短枚举，使程序尽快逼近最优解，同时增强了剪枝条件一的效果。

这一部分的实现，首先需要求MaxT。

MaxT=炸弹i，从S开始炸，可以炸到的最大编号。

如果，炸弹i炸不到S，则MaxT=S-1。

求MaxT可以用动态规划的方法解决。状态转移方程为：

MaxT= 炸弹i炸不到S S-1

炸弹i炸得到S MaxT[i]

MaxT[i][m+1]=m

求MaxT的时间复杂度为O(N2)。

具体实现方法，考虑二分图右边的n个结点（n颗炸弹），如果结点i没有匹配，i被认为可以使用。对于一个已经匹配的结点i，如果从任何一个没有匹配的结点出发存在一条到达i，而且i为外点的交错路，i也被认为可以使用。

计算所有从没有匹配点出发的交错路（没有匹配点i出发的交错路没有被匹配点i一定为外点）所能到达的匹配的结点，只要从每一个没有匹配的结点出发，宽度优先搜索，只要O(N2)的时间。注意判断重复（如果一个已经匹配的结点已经被确定为可以使用，那么不需要对它再扩展一次，因为当把这个已经匹配的结点确定为可以使用的结点的时候，已经从这个结点扩展过，如果再扩展必将产生无谓的重复）

所以MaxL=Max(MaxT｜i可以使用)；

性能分析（3）：

通过以上的优化，所有数据都是瞬间出解，并且所有结果都是最优解。

甚至对n=200的随机数据，也可以在瞬间出解，可见程序的效率有了很大的提高。

精益求精：

另外，还有两个优化，但是有时效果不好，但也值得一提。

优化三：分支定界。这样可以增强剪枝条件一的效果，但是当最优解与Dist[1]相差比较远的时候，会浪费一定的时间。

优化四：优化一中Dist[i]的值有时并不是最优的，通过测试，发现如果Dist[i]的值与最优值相差1，特别是当i 小的时候，程序的速度都会有明显的影响。所以，可以通过同样的搜索来计算Dist[i]，（本题的答案就是Dist[1]）。这样做可以增强剪枝条件一的效果，但是同时对每个i都要搜索也浪费了一定的时间。

程序结果比较：

1 2 3 4 5 6 7 8 9 10

最简单的

搜索 0.00 0.00 0.50

Time Over 0.65 Time Over Time Over Time Over Time Over Time Over 优化的搜索 0.01 0.01 0.10

Time Over 0.50 0.80

Time Over 0.55 0.30 0.80

进一步优

化的搜索

0.01 0.01 0.02 0.03 0.00 0.02 0.02 0.01 0.02 0.02

此搜索方法一般都可以用两个效果很好的可行性优化：

(1)提前判断是否可以成功，避免多余的搜索。

(2)每次判断尽量多利用以前的判断结果。

楼教主的论文原文

```
var
  i,ans,n,m,d:longint;
  x1,x2,y1,y2:array[0..105] of longint;
  can:array[0..105,0..105,0..105] of boolean;
  res,from,reach,dis:array[0..105] of longint;
  ta,tb,max:array[0..105,0..105] of longint;
  v:Array[0..105] of boolean;
  q:array[0..105*105] of longint;
  g,bobo:array[0..105,0..105] of boolean;
procedure init;
var i,j,k,t:longint;
begin
  readln(m,n,d);
  for i:=1 to m do readln(x1[i],y1[i]);
  for i:=1 to n do readln(x2[i],y2[i]);
  for i:=1 to m do
   for j:=1 to n do
    if d*d>=sqr(x1[i]-x2[j])+sqr(y1[i]-y2[j]) then
     bobo[i,j]:=true;
  for i:=1 to m do
   begin
     for j:=1 to n do can[i,i,j]:=bobo[i,j];
     for j:=i+1 to m do
      for k:=1 to n do
       can[i,j,k]:=can[i,j-1,k]and bobo[j,k];
     for j:=1 to n do
     begin
      max[i,j]:=i-1;
      for k:=i to m do
      if can[i,k,j] then max[i,j]:=k;
     end;
   end;
  dis[m+1]:=0;
  for i:=m downto 1 do
   begin
   t:=i-1;
   for j:=1 to n do
    if max[i,j]>t then t:=max[i,j];
   dis[i]:=dis[t+1]+1;
   end;
end;
function change(x:longint):boolean;//匹配下
var i:longint;
begin
 for i:=1 to n do if (not v[i] )and(g[x,i])then
 begin
   v[i]:=true;
   if (from[i]=0)or(change(from[i])) then
   begin
     from[i]:=x;
     reach[x]:=i;
     exit(true);
   end;
 end;
 exit(false);
end;
```

```
procedure dfs(used,s:longint);
var x,maxl,i,h,l:longint;
begin
  if  used+dis[s]>=ans then exit;
  if s=m+1 then
  begin
   ans:=used;
   res:=reach;
  end;
  fillchar(v,sizeof(v),false);
  maxl:=s-1;
  l:=0;
  for i:=1 to n do if from[i]=0 then
  begin
   v[i]:=true;inc(l);q[l]:=i;
  end;
  h:=0;
  repeat//bfs求出下一段最远到哪里
    inc(h);
    x:=q[h];
    if max[s,x]>maxl then maxl:=max[s,x];
    for i:=1 to used do
     if (g[i,x]) and(v[reach[i]]=false) then
     begin
      inc(l);
      q[l]:=reach[i];
      v[q[l]]:=true;
     end;
  until h>=l;
  if maxl=s-1 then exit;
  inc(used);
  ta[s]:=reach;tb[s]:=from;
  g[used]:=can[s,maxl];
  fillchar(v,sizeof(v),false);
  change(used);
  for i:=maxl downto s do
  begin
   g[used]:=can[s,i];
   dfs(used,i+1);
  end;
  reach:=ta[s];from:=tb[s];
end;
begin
 init;
 ans:=35535;
 dfs(0,1);
 writeln(ans);
 //for i:=1 to ans do  write(res[i],' ');
end.
```

6 8 条评论

RicardoShips 创建时间：2019-07-30 22:13:28 在 Ta 的

先去看了一下楼天城和朱泽园两位神仙的论文

被各种骚操作吊打得怀疑人生

首先每枚炸弹炸毁的武器应该是一段连续的区间

那么我们可以把武器划分成ans段

对于每一段武器，都用一枚炸弹去炸毁

现在我们考虑怎么样求出最小的ans

直接搜索肯定超时，考虑剪枝优化

把每段武器看成一个点，每枚炸弹看成一个点

搜索的时候用二分图优化即可

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

```
#define INF 0x3f3f3f3f
using namespace std;
int m,n,k,ans,u[101],v[101],x[101],y[101],f[101],vis[101],step[101],ok[101][101],to[101][101];
inline bool check (int a,int b)
{ return (u[a]-x[b])*(u[a]-x[b])+(v[a]-y[b])*(v[a]-y[b])<=k*k; }
inline bool find (int x) {
    for(register int i=1;i<=n;++i)
        if(!vis[i]&&ok[i][x]) {
            vis[i]=true;
            if(!f[i]||find(f[i])) {
                f[i]=x;
                return true;
            }
        }
    return false;
}
inline int read () {
    char ch=getchar(); int num=0;
    while(!isdigit(ch)) ch=getchar();
    while(isdigit(ch)) num=(num<<3)+(num<<1)+(ch^'0'),ch=getchar();
    return num;
}
inline void dfs (int now,int sum) {
    if(ans<=sum+step[now]) return ;
    if(now>m) { ans=sum; return ; }
    int g[101]; memcpy(g,f,sizeof(g));
    for(register int i=now;i<=m;++i) {
        for(register int j=1;j<=n;++j)
            if(check(j,i)&&to[j][now]>=i) ok[j][sum+1]=true;
        memset(vis,false,sizeof(vis));
        if(find(sum+1)) dfs(i+1,sum+1);
        for(register int j=1;j<=n;++j)
            if(check(j,i)&&to[j][now]>=i) ok[j][sum+1]=false;
        memcpy(f,g,sizeof(f));
    }
}
int main () {
    m=read(),n=read(),k=read(); ans=n;
    for(register int i=1;i<=m;++i) x[i]=read(),y[i]=read();
    for(register int i=1;i<=n;++i) u[i]=read(),v[i]=read();
    for(register int i=1;i<=n;++i)
        for(register int j=m;j;--j)
            if(check(i,j)) to[i][j]=max(j,to[i][j+1]);
    for(register int i=1;i<=m;++i) step[i]=INF;
    for(register int i=m;i;--i)
        for(register int j=1;j<=n;++j)
            if(check(j,i)) step[i]=min(step[i],step[to[j][i]+1]+1);
    dfs(1,0); printf("%d\n",ans);
    return 0;
}
```

2 0 条评论

在洛谷，
享受 Coding 的欢乐

关于洛谷 | 帮助中心 | 用户协
小黑屋 | 陶片放逐 | 社区规
Developed by the Luo
2013-
增值电信业务经营许可证 沪
沪ICP备18008322号 All rig