福建师范大学第 25 届低年级程序设计竞赛简要题解

FJNU 低年级程序设计竞赛命题组
Fujian Normal University

日期：December 5, 2021

摘　　要

关键词： Algorithm

难度预估:

简单题:B、F、H

中档题:E、G、A、D

防穿题:C

彩蛋: 出题人的名字是白色隐藏在背景里的, 可以用鼠标选定一下, 遇到 Frank 出的题建议先跳过.

目录

1 B. 潘皇的迷宫

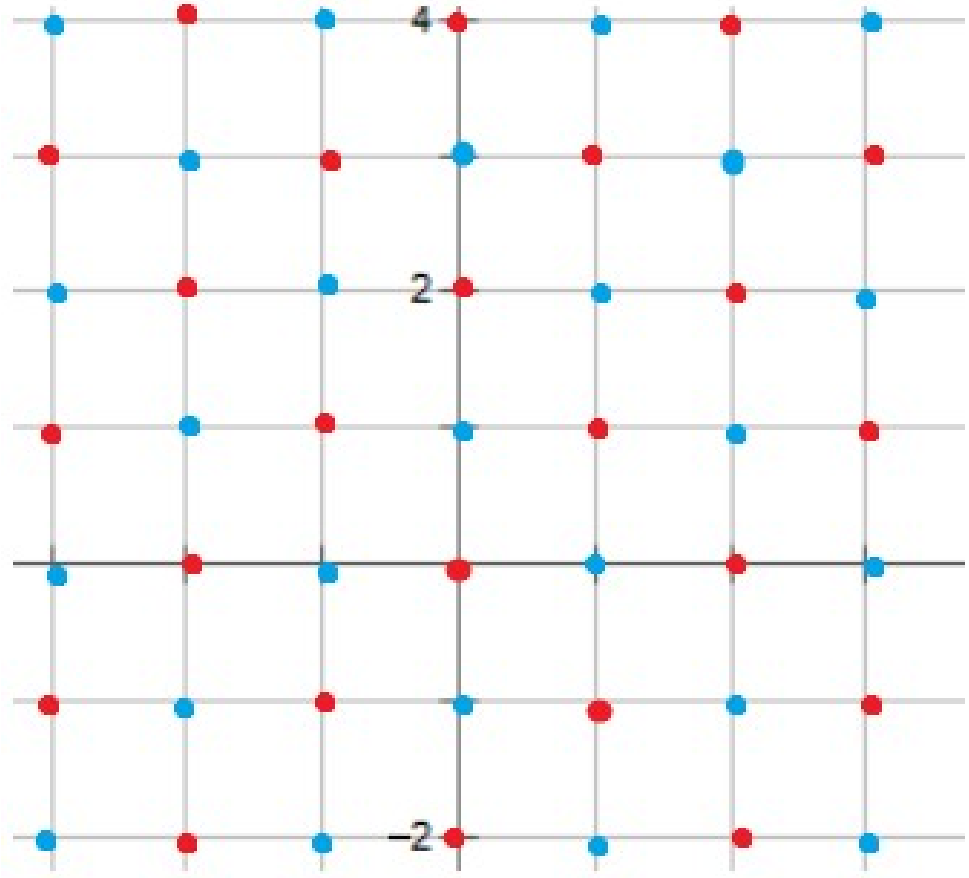
本场的签到题之一, 做法应该有挺多.

1.1 简要题意:

给定起点 (x,y) 和终点 (u,v), 只能沿着 k=ś1 的斜率变换坐标, 问若干次变换后能否从起点走到终点.

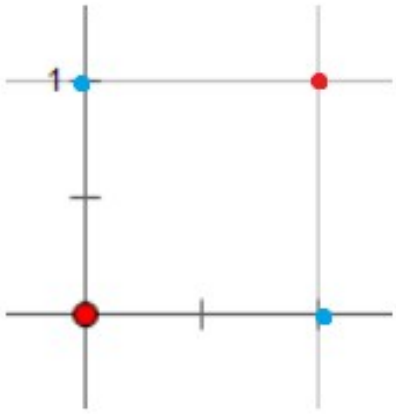
1.2 题解:

容易发现, 在这个移动规则下, 整个平面直角坐标系中的整数点可以被二染色:



同色点直接均可互相到达, 异色点之间不能到达.

不妨将所有坐标映射到原点范围内进行比较:



横纵坐标增加一个偶数, 点的颜色不发生改变. 因此可以先把坐标加一个很大的偶数, 使得所有坐标均为正数, 然后模 2 映射到这四个点当中:

(0,0) (1,1)　　(0,1) (1,0)

两个点能互相到达当且仅当两个点 x,y 坐标均相同或 x,y 坐标均不同时, 一个点可以到达另一个点. 其余情况均不能到达.

单个样例的时间复杂度 $\Theta(1)$.

ACcode

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5 + 10,INF = 1e8 + 10;
int main()
{
    int T;
    scanf("%d",&T);
    while(T--)
    {
        int a,b,c,d;
        scanf("%d %d %d %d",&a,&b,&c,&d);
        a += INF;b += INF;
        c += INF;d += INF;

        a %= 2;b %= 2;
        c %= 2;d %= 2;

        if(a==c && b==d) puts("Yes");
        else if(a!=c && b!=d) puts("Yes");
        else puts("No");
    }

    return 0;
}
```

2 F.Nekopara vol.pan

这题的具体证明不是很好想, 但是前 2 个小结论还是可以比较容易得到的. 最后一个在考场上可以适当猜一下.

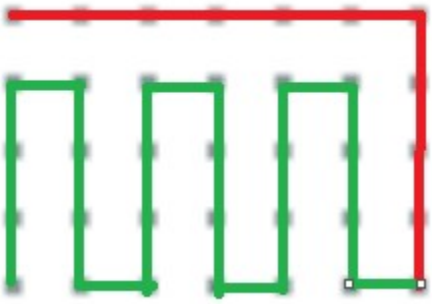
2.1 简要题意

给定一个 n 行 m 列的点阵图, 起点在左上, 终点在左下, 问是否存在一条无斜边的哈密顿路径.

2.2 题解

容易发现, 当 n 为偶数的时候, 这样的路径一定存在, 每两行为一个划分构造如图:

当 n 为奇数,m 为奇数的时候的时候, 这样的路径一定存在, 在走完一个折线之后, 剩下的点变为一个 n 为偶数的同类问题.



当 n 为奇数,m 为偶数的时候, 这样的路径一定不存在, 不妨在点阵左边加上一列点.

图中红色点为原来的起点和终点, 若在这样的图上构造一条哈密顿回路, 使得终点能到达起点, 则新加的点的连线方式必然如图:

要构成一个回路, 向左走的次数必然等于向右走的次数, 向下走的次数必然等于向上走的次数. 在右边的原图中, 一共走 $n * m - 1$ 次. 设在其中左右走的次数为 x 次, 向上走 y 次, 向下应为

$y + (n-1)$ 次.

则:$2x + 2y + (n-1) = nm - 1$.

左边为奇数, 右边为偶数.

因此这样的构造不存在.

时间复杂度 $\Theta(1)$.

ACcode

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e5+1;
char s1[N],s2[N];
int main()
{
    scanf("%s %s",s1 + 1,s2 + 1);
    int n = strlen(s1 + 1),m = strlen(s2 + 1);
    if((s1[n] - '0') % 2 == 1 && (s2[m] - '0') % 2 == 0) puts("Poor PanHuang!");
    else puts("PanHuang yyds!");

    return 0;
}
```

3 H. 斗牛

这题应该是思维最简单的题了, 就是题面有点长.

3.1 简要题意

给 5 张牌, 根据题意计算并比较每个玩家最终的点数大小.

3.2 题解

因为只有 5 张牌, 我们分别用变量 i,j,k 枚举数字, 判断这三张牌是否是 10 的倍数.

若是, 我们计算剩下两张牌的大小, 尝试更新这个玩家点数的最大值.

注意判断 i,j,k 是否两两不相等和一些细节.

时间复杂度 $\Theta(5^4)$

ACcode

```
#include<bits/stdc++.h>
using namespace std;
int a[7],b[7];
int cal(int a[]){
    int res = 0;
    for(int i=1;i<=5;++i)
        for(int j=1;j<=5;++j){
            if(i==j) continue;
            for(int k=1;k<=5;++k){
                if(j==k || i==k) continue;
                if((a[i] + a[j] + a[k]) % 10) continue;
                int tmpans = 0;
                for(int u=1;u<=5;++u){
                    if(u==i || u==j || u==k) continue;
                    tmpans += a[u];
                }
                if(tmpans % 10 == 0 ) return 10;
                tmpans %= 10;
                res = max(res,tmpans);
            }
        }
    return res;
}
int main(){
    for(int i=1;i<=5;++i)
        scanf("%d",&a[i]);
    for(int i=1;i<=5;++i)
        scanf("%d",&b[i]);
    int ansa = cal(a),ansb = cal(b);
    if(ansa == ansb) puts("draw");
    else if(ansa > ansb) puts("ahuinb");
    else puts("kknb");
    return 0;
}
```

4 E. 小蔡时代

不是很好想的思维题, 不适合 ckk 这种学了很久但依旧没有思维的人, 但是可以筛选出有思维的人.

4.1 简要题意

给定一个含有 n(偶数) 个点 m 条边的无向图, 点, 边均有权值. 玩家轮流选择未被选择的点, 获得它们的点权. 如果一条边的两个点均被玩家选择, 则这条边的边权也归属于这名玩家. 求两人采用最优方案时, 最后所获得的权值的差.

4.2 题解

因为最终所求得的答案是个差值, 因此我们可以把边权除以 2 分摊到 2 个点上. 这样, 如果一个玩家在最终的答案中选择了这两个点, 则这条边权的答案也会被计算入他所获得的总权值当中, 反之如果这两个点被不同玩家所占领, 则他们最终答案的差值不变.

在此基础上, 我们把最终每个点的点权从大到小排序, 两名玩家依次选取

时间复杂度 $\Theta(nlogn)$.

ACcode

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 1e4 + 10;
int n,m;
ll ans,p[N];
int main()
{
    scanf("%d %d",&n,&m);
    for(int i=1;i<=n;++i)
    {
        int x;scanf("%d",&x);
        p[i] += 2 * x;
    }
    while(m--)
    {
        int a,b,x;
        scanf("%d %d %d",&a,&b,&x);
        p[a] += x;p[b] += x;
    }
    sort(p+1,p+1+n);
    for(int i=n;i>=1;--i)
    {
        if(i&1) ans -= p[i];
        else ans += p[i];
    }
    printf("%lld\n",ans / 2);
    return 0;
}
```

5 G. 寒冰射手

ckk 不会写这题, 但是验题人说很简单就继续放这里给有思维的人写吧, 这里放一个出题人的题解.

5.1 简要题意

在三棱锥中计算射线总的通过路径, 计算方法参考具体题目.

5.2 题解

首先我们水平入射, 由平行四边形的定义可知, 轨迹一定成一个同旁内角为 60° 和 120° 的平行四边形. 我们再将正三角形抽象成一个平行四边形. 易得棱长为 1 的菱形的贡献为一个正三角形, 因为经过两次反射后会回到原点, 同理根据形似的性质可知, 棱长为 n 的菱形的贡献为 1 个边长为 n 的正三角形, 即贡献为 3n. 一般地对于任意边长为 n,m 的平行四边形, 我们进行若干次反射, 可以转化成边长为 n,m % n 的平行四边形, 假设 n<m 可知贡献即为 $3*(n*|\frac{m}{m}|)$. 由拓展欧几里得定理及数学归纳法, 可得经过若干次转化一定能将一个非菱形的平行四边形转化成一个菱形. 所以最终答案为:

$3*(N-GCD(N,N-K))$.

时间复杂度 $\Theta(logn)$.

ACcode

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
ll gcd(ll a,ll b) //求最大公约数
{
    return b ? gcd(b,a % b) : a;
}
int main()
{
    ll n,x;
    scanf("%lld %lld",&n,&x);
    printf("%lld\n",3 * (n - gcd(n,x)));
    return 0;
}
```

6 A.《光明记忆：无限》

比较简单的区间 DP, 用于筛出学过一段时间算法的人.

6.1 简要题意

给定 n 个机关和相应的复杂度与电磁干扰强度, 消除每个机关所需要消耗相应的复杂度和相邻两侧机关的电磁干扰强度之和. 消除一个机关后, 其余的机关会紧缩在一起.

6.2 题解

每个机关的复杂度都会且只会被计算一次, 因此直接在最终答案加上即可. 考虑计算电磁干扰强度, 容易发现这是一个区间 DP.

设 $dp[L][R]$ 表示将 $[L, R]$ 这个区间的机关全部消灭所要耗费的最小代价, 我们枚举在这个区间内最后一个消灭的机关 k.

则:

dp[L][R] = min(dp[L][R],dp[L][k-1] + dp[k+1][R] + 消灭 k 所要的代价).

表示要将这个区间全部消灭, 则最后必然消灭其中的最后一个机关 k. 那么取将 $[L, k-1]$ 和 $[k+1, R]$ 消灭的最小代价加上消灭 k 所需要的最小代价即为最终所要的代价可能值. 一开始我们枚举区间长度 len, 和起点 L, 则 $R = L + len1$;

最终枚举 k 即可得到答案, 最终的代价即为 $dp[1][n]$, 再加上复杂度的代价即可.

时间复杂度 $\Theta(n^3)$.

ACcode

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 405;
const ll INF = 1ll << 60;
int n,b[N];
ll ans,dp[N][N];
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;++i)
    {
        int x;scanf("%d",&x);
        ans += x;
    }

    for(int i=1;i<=n;++i) scanf("%d",&b[i]);
    for(int len=1;len<=n;++len)
    {
        for(int L=1;L+len-1<=n;++L)
        {
            int R = L + len - 1;
            dp[L][R] = INF; //这个区间的初始值设为正无穷.
            for(int k=L;k<=R;++k)
```

```
            {
                ll val1 = 0,val2 = 0;
                if(k-1>=L) val1 = dp[L][k-1]; //注意边界
                if(k+1<=R) val2 = dp[k+1][R];
                dp[L][R] = min(dp[L][R],val1 + val2 + b[L-1] + b[R+1]);
            }
        }
    }

    printf("%lld\n",ans + dp[1][n]);
    return 0;
}
```

7 D. りょういきてんかい

一道比较思维的题, 用于筛出学过一段时间算法, 且比较有想法的同学.

7.1 简要题意

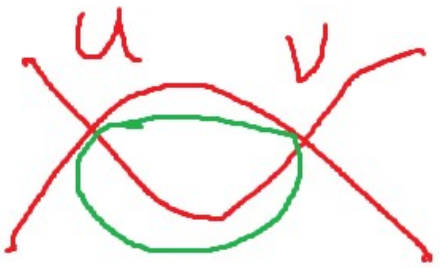
给一个 n 个点 m 条边的无向图, 每条边边权均为 1. 删除最多的边以使得从其中一个起点 s_1 到终点 t_1 所需要的时间小于等于 l_1; 从另一个一个起点 s_2 到终点 t_2 所需要的时间小于等于 l_2. 求最多能删除的边的数量.

7.2 题解

因为点权均为 1, 所以我们可以对每个点 BFS 求出这个点到其它点的最短距离.

若这两条路径不相交, 则最终要保留的边即为两个路径的最短距离 dist[u][v].

否则若这两条路径相交, 则他们之间最多有 2 个交点, 否则如图（纯手画，有点丑）所示, 我们可以删除多余的边:



我们枚举交点 u,v.

则答案的可能值即为:

dist[s1][u] + dist[s2][u] + dist[u][v] + dist[v][t1] + dist[v][t2];

表示从 s1,s2 分别到 u, 然后从 u 到 v, 在从 v 到两个终点 t1,t2;

或

dist[s1][u] + dist[t2][u] + dist[u][v] + dist[v][t1] + dist[v][s2];

表示从 s1,t2 分别到 u, 然后从 u 到 v, 在从 v 到一个终点和另一个的起点 t1,s2;

最终可以删除的边数 = m- 最小要保留的边数.

时间复杂度 $\Theta(n^2)$. ACcode

```
#include<bits/stdc++.h>
using namespace std;
const int N = 3005,INF = 0x3f3f3f3f;
int n,m,s1,t1,l1,s2,t2,l2;;
int e[N<<1],ne[N<<1],head[N<<1],idx;
int dist[N][N],st[N];
queue<int> q;
void add(int a,int b)
{
    e[idx] = b;ne[idx] = head[a];head[a] = idx++;
}
```

```cpp
void bfs(int u)
{
    memset(st,0,sizeof(st));
    while(q.size()) q.pop();
    st[u] = 1;
    dist[u][u] = 0;
    q.push(u);
    while(q.size())
    {
        int t = q.front();q.pop();
        for(int i=head[t];~i;i=ne[i])
        {
            int j = e[i];
            if(st[j]) continue;
            st[j] = 1;
            dist[u][j] = dist[u][t] + 1;
            q.push(j);
        }
    }
}
int main()
{
    memset(head,-1,sizeof(head));
    scanf("%d %d",&n,&m);
    for(int i=1;i<=m;++i)
    {
        int a,b;scanf("%d %d",&a,&b);
        add(a,b);add(b,a);
    }
    scanf("%d %d %d %d %d %d",&s1,&t1,&l1,&s2,&t2,&l2);
    for(int i=1;i<=n;++i)
        bfs(i);

    if(dist[s1][t1] > l1 || dist[s2][t2] > l2)
        return puts("-1"),0;
    int ans = m - dist[s1][t1] - dist[s2][t2];
    for(int u=1;u<=n;++u){
        for(int v=2;v<=n;++v){
            int tmp,c1,c2;
            tmp = dist[s1][u] + dist[s2][u] + dist[u][v] + dist[v][t1] + dist[v][t2];
            c1 = dist[s1][u] + dist[u][v] + dist[v][t1];
            c2 = dist[s2][u] + dist[u][v] + dist[v][t2];
            if(c1 <= l1 && c2 <= l2) ans = max(ans,m - tmp);

            tmp = dist[s1][u] + dist[t2][u] + dist[u][v] + dist[v][t1] + dist[v][s2];
            c1 = dist[s1][u] + dist[u][v] + dist[v][t1];
            c2 = dist[t2][u] + dist[u][v] + dist[v][s2];
            if(c1 <= l1 && c2 <= l2) ans = max(ans,m - tmp);
        }
    }
    printf("%d\n",ans);
    return 0;
}
```

8 C. 潘皇锁难

贪心 +dp+ 线段树, 是每年惯例的防穿题.ckk 看出最后是要使得这两个序列的单调递增元素一样多, 然后他 P 都不会了.

有想学的建议直接去问出题人.

.