小米 Online Judge 18年12月常规赛题解

本题解仅供大家参考，欢迎大佬们提供更为巧妙的方法~

小米兔跳格子

题解

解法一：动态规划

> 维护一个一维数组dp,dp[i]表示到达i位置时剩余的步数，当前位置的剩余步数和当前位置最多能跳的步数的较大的数决定了从当前位置能跳的最远的距离，而下一个位置的剩余步数就等于这个较大值减去1，所以状态转移方程为：dp[i] = max(dp[i-1], nums[i-1])-1，初始状态dp[0]=0。如果某一个时刻dp[i]为负数，则说明不能到达i位置，即也不能到达最后一个位置，可直接返回false。最后判断dp的最后一个位置的值是否为负数，为负数说明不能到达最后一个位置，否则可以到达。

解法二：贪婪算法（推荐）

> 因为此题要判断米兔是否能到达最后一个位置，所以我们关心的只是米兔是否能到达最后一个位置，不需要去关心每一个位置上剩余的步数。我们定义一个变量maxDistance，表示当前米兔最远能到达的位置，初始值为0，然后遍历每一个格子，当maxDistance小于当前的格子下标，则说明米兔不能到达当前这个格子，更不能到达最后一个格子吃到胡萝卜；当maxDistance大于等于一个格子到最后一个格子的距离，说明米兔可以到达；否则更新maxDistance的值，maxDistance=max(maxDistance, i+nums[i]),(i为当前格子下标，nums[i]表示此时米兔最多可以跳的格子数)。

标程

```
public class Main {
    public String solution(int[] nums) {
        int maxDistance = 0;
        for(int i = 0; i < nums.length; i++) {
            if(i > maxDistance || maxDistance >= nums.length) break;
            maxDistance = Math.max(i + nums[i], maxDistance);
        }
        return (maxDistance >= nums.length-1) + "";
    }
}
```

数数字游戏

题解

> 典型的 `SG` 函数问题。可选步数为一系列不连续的数（斐波那契数），可用 `getSG` 函数求得，最终结果是两个目标数字的所有 `SG` 值异或的结果 。

标程

```
#include<stdio.h>
#include<string.h>
int sg[10005],vis[10005];
int get_sg(int n){
    int f[21]={0,1,2,3,5,8,13,21,34,55,89,144,233,
               377,610,987,1597,2584,4181,6765,10946};
    memset(sg,0,(n+1)*sizeof(int));
    for(int i=1; i<=n; i++){
        memset(vis,0,(n+1)*sizeof(int));
        for(int j=1; f[j]<=i; j++)
            vis[sg[i-f[j]]]=1;
        for(int j=0; j<=n; j++)
            if(vis[j]==0){
                sg[i]=j;
                break;
            }
    }
    return sg[n];
}
int main(){
    int num1,num2,m;
    while(scanf("%d%d",&num1,&num2)!=-1){
        m=0;
        m^=get_sg(num1);
        m^=get_sg(num2);
        printf(m?"Xiaoai Win\n":"Xiaobing Win\n");
    }
    return 0;
}
```

MMD

题解

> 每条线段的固定收益可以直接求和.
>
> 然后可以有一万种方法求出线段的交.把线段看作点,交看作边,得到一张有向图.
>
> 观察到由高点权到低点权且有交的限制构成了一张DAG, 考虑如何最大化被选择的路径上的边权和.把每个点拆成x和x', 若原图上存在边$x \to y$, 则连边$x \to y'$, 权值为收益, 这样就可以合并以x为终点, 以y为起点的路径, 并获得边权的额外收益.
>
> 由于二分图匹配的特性, 原图上每个点只能进入一次, 离开一次, 符合题目要求, 跑一个二分图最大权匹配就可以了.
>
> 需要注意的是, 如果使用了最大费用最大流(EK)算法, 需要在增广出负的费用时退出, 因为在本题中我们要最大化权值, 而不是最大化流量.

标程

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

using namespace std;

typedef double db;
typedef long double ldb;
typedef long long ll;

const int maxn = 2005;
const ldb eps = 1e-9;
const int inf = 0x3f3f3f3f;

char line[1000001];

struct Vector
{
    ldb x, y;
    Vector(ldb xx = 0, ldb yy = 0):x(xx),y(yy){}
    Vector operator + (Vector rhs)
    {
        return Vector(x + rhs.x, y + rhs.y);
    }
    Vector operator - (Vector rhs)
    {
        return Vector(x - rhs.x, y - rhs.y);
    }
    ldb operator * (Vector rhs)
    {
        return x * rhs.x + y * rhs.y;
    }
    friend Vector operator * (Vector lhs, ldb rhs)
    {
        return Vector(lhs.x * rhs, lhs.y * rhs);
    }
    ldb operator ^ (Vector rhs)
    {
        return x * rhs.y - y * rhs.x;
    }
    bool operator < (const Vector &rhs) const
    {
        return (x == rhs.x) ? (y < rhs.y) : x < rhs.x;
    }
};
ldb norm(Vector a)
{
    return sqrt(a * a);
}
typedef Vector Point;

struct Line
{
```

```
    Point P;
    Point V;
    Line(Point P_ = Point(), Point V_ = Point()):P(P_), V(V_)
    {}
}l[maxn];

inline int DCMP(ldb x)
{
    if(x > -eps && x < eps) return 0;
    else return (x > 0) ? 1 : -1;
}
Point intersection(Line a, Line b)
{
    if(!DCMP(a.V ^ b.V)) return Point(-inf, -inf);
    return b.P + b.V*((a.V ^ (a.P - b.P))/(a.V ^ b.V));
}
ldb dist(Point p, Point q)
{
    return norm(p - q);
}
inline bool legal(Line a, Line b)
{
    Point p = intersection(a, b);
    //printf("%Lf %Lf\n", p.x, p.y);
    ldb mnax = min(a.P.x, a.P.x + a.V.x), mxax = max(a.P.x, a.P.x + a.V.x);
    ldb mnay = min(a.P.y, a.P.y + a.V.y), mxay = max(a.P.y, a.P.y + a.V.y);
    ldb mnbx = min(b.P.x, b.P.x + b.V.x), mxbx = max(b.P.x, b.P.x + b.V.x);
    ldb mnby = min(b.P.y, b.P.y + b.V.y), mxby = max(b.P.y, b.P.y + b.V.y);
    if(DCMP(p.x - mnax) >= 0 && DCMP(p.x - mxax) <= 0
    && DCMP(p.y - mnay) >= 0 && DCMP(p.y - mxay) <= 0
    && DCMP(p.x - mnbx) >= 0 && DCMP(p.x - mxbx) <= 0
    && DCMP(p.y - mnby) >= 0 && DCMP(p.y - mxby) <= 0)
        return true;
    else return false;
}

//computational geometry over

struct edge
{
    int to, nxt, w, c;
}e[maxn * maxn << 1];
int n, s, t, ptr, lnk[maxn], dis[maxn], pre[maxn], id[maxn];
int val[maxn], vis[maxn];

inline void add(int bgn, int end, int val, int cost)
{
    e[ptr].to = end; e[ptr].nxt = lnk[bgn]; e[ptr].w = val; e[ptr].c = cost; lnk[bgn] =
ptr; ++ptr;
    e[ptr].to = bgn; e[ptr].nxt = lnk[end]; e[ptr].w = 0; e[ptr].c = -cost; lnk[end] =
ptr; ++ptr;
}
inline bool spfa()
```

```
{
    //memset(dis, 0x3f, sizeof dis);
    for(int i = s; i <= t; ++i) dis[i] = -0x3f3f3f3f;
    dis[s] = 0; queue<int> q; q.push(s); vis[s] = 1;
    while(!q.empty())
    {
        int u = q.front(); q.pop(); vis[u] = 0;
        for(int p = lnk[u]; ~p; p = e[p].nxt)
        {
            int y = e[p].to;
            if(e[p].w && dis[y] < dis[u] + e[p].c)
            {
                dis[y] = dis[u] + e[p].c;
                pre[y] = u;
                id[y] = p;
                if(!vis[y]) vis[y] = 1, q.push(y);
            }
        }
    }
    return dis[t] > -0x3f3f3f3f;
}
inline int costflow()
{
    int flow = 0, cost = 0, aug;
    while(spfa())
    {
        if(dis[t] < 0) break;
        aug = 0x3f3f3f3f;
        for(int p = t; p != s; p = pre[p])
            aug = min(aug, e[id[p]].w);
        //if(aug < 0) break;
        flow += aug; cost += aug * dis[t];
        for(int p = t; p != s; p = pre[p])
            e[id[p]].w -= aug, e[id[p]^1].w += aug;
    }
    return cost;
}

int main()
{
    //freopen("in.txt", "r", stdin);
    //freopen("out.txt", "w", stdout);
    while (scanf("%d", &n) != EOF) {
        ptr = 0;
        memset(lnk, -1, sizeof lnk);int sum = 0;
        s = 0; t = n * 2 + 1;
        for(int i = 1; i <= n; ++i)
        {
            int px, py, qx, qy;
            scanf("%d%d%d%d%d", &px, &py, &qx, &qy, &val[i]);
            sum += val[i];
            l[i] = Line(Point(px, py), Point(qx - px, qy - py));
        }
```

```
        for(int i = 1; i <= n; ++i)
            for(int j = 1; j <= n; ++j)
            {
                if(i == j || val[i] <= val[j]) continue;
                if(legal(l[i], l[j]))
                    //cout << i << " to " << j << endl,
                    add(i, j + n, 1, (val[i] + val[j]) * ((i + j) / 2));
            }
        for(int i = 1; i <= n; ++i) add(s, i, 1, 0), add(i + n, t, 1, 0);
        printf("%d\n", costflow() + sum);
    }
    return 0;
}
```