CHA²-OI ROUND 1 题解

SOL. MAKER：查茶 chala_tea（wjx）

T1　Malody

难度：入门 or 普及-
实则是一道非常简单的“数字 to 字符串”的模拟
关键点在于读题，是从上到下而节拍从大到小（模拟了 4k 谱面真实情况）

```
#include<bits/stdc++.h>
using namespace std;
char c[105][5];
int n,m;
int main(){
	for(int i=1;i<=100;i++){ //初始化，所有格都用空格填满
		for(int j=1;j<=4;j++){
			c[i][j]=' ';
		}
	}
	cin>>n>>m;
	for(int i=1;i<=n;i++){
		int a,k;
		cin>>a>>k;
		c[a][k]='_'; //将第 a 拍第 k 列用“_”代替
	}
	for(int i=m;i>=1;i--){//节拍倒序输出，保证从上到下节拍从大到小
		for(int j=1;j<=4;j++){
			cout<<c[i][j];
		}
		cout<<endl;
	}
	return 0;
}
```

T2　Phigros

难度：普及- ~ 普及
一道考验结构体运用、排序、计算的水题（别骂了别骂了我有用 SPJ）
虽说有点考验审题读题，但其实理清了就很简单。

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

struct song{
	string s;
	double level,acc,rks;
}a[1005];

int n;
double totalrks=0;

bool cmp(song x,song y){
	if(x.rks==y.rks)return x.level>y.level;
	return x.rks>y.rks;
}

int main(){
	cin>>n;

	for(int i=1;i<=n;i++){
		cin>>a[i].s>>a[i].level>>a[i].acc;
		if(a[i].acc==100){
			a[i].rks=a[i].level;
		}else{
			a[i].rks=((a[i].acc-55)/45)*((a[i].acc-
55)/45)*a[i].level;
		}
	}
```

```
        sort(a+1,a+n+1,cmp);
        for(int i=1;i<=10;i++)totalrks+=a[i].rks;
        totalrks/=10;
        cout<<floor((totalrks+0.005)*100)/100<<endl;
        for(int i=1;i<=n;i++){
                cout<<a[i].s<<'
'<<floor((a[i].rks+0.005)*100)/100<<endl;
        }
        return 0;
}
```

T3　Arcaea

难度：提高+/省选-

（说句实话，这道题是 LGR-(-13)（洛谷去年 CSP 初赛模拟赛）的原题）

思路：答案一定为 a*（x/2）+b 的形式，枚举 a=2i-j 而计算 b 的范围。

我们可以做循环，以 i 表示有 i 个点得了分。

我们还可以假设最多只有一个 x/2，因为两个 x/2 可以组成一个 x 和一个 0。所以就可以用一个 j 在 1~0 之间反复横跳来表示。

设一个区间 upper~lower，因为可以调整获得的是 x+1 还是 x，所以 upper 和 lower 可以分别表示 x+1 的上下限。表示在这个区间内的所有分都可以得到，所以答案加成即为 upper-lower+1。

而 last 则为当前最多能拿的分，对于无端直接加 upper-lower+1 进行约束；base 则是思路中“a”的体现。

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

using namespace std;

int n,m;

int main(){

        cin>>n>>m;

        if(n==m){//剪枝技巧，当全都是 m 时，就是 0 个得分，1 个得分，2 个得分……一直到 n 个得分，就相当于 n+1 种情况。

                cout<<n+1;

                return 0;

        }

        long long total=10000000;

        long long ans=1,last=0;//因为循环从 1 开始，未统计不得分的情况，所以 ans 初始为 1.

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

                for(int j=1;j>=0;--j){

                        int lower=max(0,i-(n-m));// x、x/2 最多只有 n-m 个，所以 x+1 最少有 i-(n-m) 次

                        int upper=i-j;

                        int base=(2*i-j)*total/(2*n);

                        ans+=upper-lower+1;

                        if(lower+base<=last) ans-=last-(lower+base)+1;

                        last=base+upper;
```

```
                }
        }
        cout<<ans;
        return 0;
}
```

T4　Maimai

难度：货真价实的普及+/提高（说实话，T3 比 T4 还难）

实际上是一道欧拉回路模板题。

可以随便选一个不是孤立点的点，一次 dfs 突突到底并且把答案压栈，最后倒序输出答案就完事了。但最后还需判断一下答案栈是否和边数相同。若不相同就说明没走完则输 NO。

剪枝技巧：玩过一笔画的都知道，只要有一个点的连接的边数是奇数，那么这张图就不可能有欧拉回路。所以，我们可以判断每个点所连上的边数是否为奇数，若有一个就 NO。

（虽然但是，这道题的所有数据答案都是 YES，所以直接刷 NO 的就请勿痴心妄想（逃））

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

struct edge{
	int num,to;
};

const int N=1e5+10;
vector<edge>g[1000005];
vector<int>ans;
int n,m;
int myroad[1000005];
bool been[2000005];

void dfs(int x){
	for(int i=0;i<g[x].size();i++){
		int c=abs(g[x][i].num);
		if(been[c]) continue;
		been[c]=1;
		dfs(g[x][i].to);
		ans.push_back(g[x][i].num);
	}
	return;
}
int main(){
```

```
	cin>>n>>m;
	for(int i=1;i<=m;i++){
		int x,y;
		cin>>x>>y;
		g[x].push_back(edge{i,y});
		g[y].push_back(edge{-i,x});
		myroad[x]++,myroad[y]++;
	}

	for(int i=1;i<=n;i++){
		if(myroad[i]%2){
			cout<<"NO";
			return 0;
		}
	}

	for(int i=1;i<=n;i++){
		if(myroad[i]){
			dfs(i);
			break;
		}
	}

	if(ans.size()!=m){
		cout<<"NO";
		return 0;
	}
	cout<<"YES"<<endl;
	for(int i=m-1;i>=0;i--)cout<<ans[i]<<' ';
	return 0;
}
```