# 夜莺杯补题报告

## E.HearthStone!

### 思路

因为xym的所有怪物只有1的攻击和血量,所以剩余血量取决过程中怪物的溢出量
通过贪心的策略可以找到怪物剩余血量的最大值与最小值
若设最大剩余血量为 **a**，最小剩余血量为 **b**
在模拟最小剩余血量过程中 我们可以通过调整攻击顺序让怪物溢出使最后剩余血量为 **b+1**
依次递推,我们可以得到怪物的可能血量在 **a~b**区间上

所以求出怪物剩余血量的最大值与最小值即可.

### 代码

```
const int max_n = 1e6+10;

long long T,limit,a_n,b_x,b_y,a_z,a_i;
long long sit;
struct card{
    int z,num;
};
card node[max_n];

bool cmp(const card &l_value,const card &r_value){
        return l_value.z < r_value.z;
}
long long solve(){
    long long y_max = b_y,y_min = b_y,sit_t = sit;
    std::sort(node+1,node+a_n+1,cmp);
    int i = 1;
    if(node[i].z == 0){
        sit_t += node[i].num;
        y_max -= node[i].num;
        i = 2;
    }
    for(i;i <= a_n;i++){
        if(sit_t+1 < node[i].z){
            long long overflow = node[i].z - sit_t - 1LL;
            if(node[i].num >= overflow){
                y_max -= overflow * ((sit_t+1LL) + (sit_t+overflow)) / 2LL +
overflow;
                y_max -= (node[i].num - overflow) * (node[i].z+1LL);
            }
            else{
                y_max -= node[i].num * ((sit_t+1LL) + (sit_t+node[i].num)) / 2LL +
node[i].num;
            }
        }
```

```
            else
                y_max -= node[i].num * (node[i].z+1LL);
            sit_t += node[i].num;
        }
        sit_t = sit;
        for(int i = 1;i <= a_n;i++){
            if(node[i].z == 0)
                continue;
            y_min -= node[i].num;
            sit_t = std::max(0LL,sit_t - (node[i].z-1LL)*node[i].num);
        }
        y_min -= (limit - sit_t);
        return y_min-y_max+1LL;
    }
    int main(){
        scanf("%d",&T);
        while(T-- > 0){
            scanf("%lld %lld %lld %lld",&limit,&a_n,&b_x,&b_y);
            sit = limit;
            for(int i = 1;i <= a_n;i++){
                scanf("%lld %lld",&a_z,&a_i);
                node[i].z = a_z;
                node[i].num = a_i;
                sit -= a_i;

            }
            if(b_x == 0)
                printf("1 \n");
            else
                printf("%lld \n",solve());
        }
        return 0;
    }
```

# F.Awa玩游戏

## 思路

尝试模拟过程可以知道几点

- 当场上只剩一个人的时候下一轮退出
- 第k轮手牌为k或m-k+1的人将退出
- 场上的人数与剩余可能范围内的数相同时所有人都能退出

将手牌按大小排序后,依次验证上三点并更新结果即可得到最后答案

## 代码

```
const int max_n = 1e6+10;
int T,n,m,num_awa,pos_awa,nums[max_n],ans;
```

```
int solve(){
    std::sort(nums+1,nums+n+1);
    pos_awa = std::find(nums+1,nums+n+1,num_awa) - nums;
    if(num_awa == 1 || num_awa == m || n == 1){
        ans = 1;
        return ans;
    }
    //1.
    ans = std::min(num_awa,m-num_awa+1);
    //2. n_now == 1
    if(pos_awa == 1){
        ans = std::min(ans,m-nums[2]+1+1);
    }
    else if(pos_awa == n){
        ans = std::min(ans,nums[n-1]+1);
    }
    else{
        ans = std::min(ans,std::max(nums[pos_awa-1],m-nums[pos_awa+1]+1)+1);
    }
    //3. r-l == n_now
    int pos_r = pos_awa,pos_l = pos_awa;
    while(pos_l > 1 && nums[pos_l-1] == nums[pos_l]-1)
        pos_l--;
    while(pos_r < n && nums[pos_r+1] == nums[pos_r]+1)
        pos_r++;
    ans = std::min(ans,std::max(nums[pos_l],m-nums[pos_r]+1));
    return ans;
}

int main(){
    scanf("%d",&T);
    while(T-- > 0){
        scanf("%d %d",&n,&m);
        for(int i = 1;i <= n;i++)
            scanf("%d",&nums[i]);
        num_awa = nums[1];
        printf("%d\n",solve());
    }

    return 0;
}
```

# H.和生蚝一起做乘法吧

## 思路

可以知道给定所有数的和相等(位交换在同位)
所以有 **a+b=c** 当 **a-b** 最大时 **a*b** 最小
所以需要找到交换后 **1** 的最大数量
统计二进制位数后按最多位数 依次取出数可以保证 **a-b** 最大

找到位数最多为 **k**
而 **1** 的数量可以知道最多为 **r-l+1-k** (当通过位交换使有>**k**的**1**的数量时,可以得到乘积会变大)

## 代码

```
using ll = long long;

const long long _mod = 1e9+7;
ll l,r,ans = 1ll;
int t;

struct bit_node{
    long long bit,cnt;
    bit_node(long long a,long long b){bit = a;cnt = b;}
    bit_node(){}
};

bit_node bit[64];
bool cmp(const bit_node &l , const bit_node &r){
    return l.cnt > r.cnt;
}

long long fast_pow(long long a,long long b){
    long long res = 1ll;
    for(;b;b >>= 1){
        if(b & 1)
            (res *= a) %= _mod;
        (a *= a) %= _mod;
    }
    return res;
}

ll bit_cnt(ll bit, ll in){
    if(!in) return 0;
    ll len = (1ll << bit) , a_p = (1ll << (bit+1));
    // a_p << 1 == a_p - len ???
    ll cnt = in / a_p * len;
    ll mod_in = in % a_p;
    if(mod_in >= len)
        cnt += mod_in - len + 1;
    return cnt;
}

long long solve(){

    for(int i = 0;i < 60;i++){
        bit[i].bit = i; bit[i].cnt = 0;
        bit[i].cnt = bit_cnt(i,r) - bit_cnt(i,l-1);
        //printf("%lld\n",bit[i].cnt);
    }

    ll k_max = 0;
```

```
    for(int i = 1;i < 60;i++)
       k_max = std::max(k_max,bit[i].cnt);
    ll t = (r - l + 1ll) - k_max;
    bit[0].cnt -= t;
    std::sort(bit,bit+60,cmp);

    ans = 1ll;
    ll n = 0;
    for(int i = 0; i < 60;i++){
        (n += (1ll << bit[i].bit)) %= _mod;
        if(bit[i].cnt != bit[i+1].cnt){
            ans = ans * fast_pow(n,bit[i].cnt - bit[i+1].cnt) % _mod;
           // printf("%lld %lld\n",n,bit[i].cnt - bit[i+1].cnt);
        }
    }
    return ans;
}

int main(){
    scanf("%d",&t);
    while(t--){
        scanf("%lld %lld",&l,&r);
        printf("%lld\n",solve());
    }

    return 0;
}
```

# K.新手训练

## 思路

暴力枚举按题意验证即可

## 代码

```
const int max_n = 1e6+10;
int l1,l2,r1,r2;
bool nums_flag [11];
int num1[11],num2[11];
struct num{
    int z1,z2,m1,m2;
    num(int a,int b,int c,int d){z1 = a;z2 = c;m1 = b;m2 = d;}
    num(){}
}ans[max_n];

int gcd(int a,int b){
    return b ? gcd(b,a%b) : a;
}

bool cmp(const num &l,const num &r){
```

```
    return l.z2 * r.m2 < l.m2 *r.z2 || (l.z2 * r.m2 == l.m2 *r.z2 && l.z1 <r.z1);
}

bool contain_judge(int m,int n){
    *num1 = *num2 = 0;
    while(m){
        num1[++ *num1] = m%10;
        m /= 10;
    }
    while(n){
        num2[++ *num2] = n%10;
        n /= 10;
    }
    for(int a = *num1,b = *num2;a && b; a--){
        if(num1[a] == num2[b])
            if(!(--b))
                return 1;
    }
    return 0;
}

bool value_check(int fz ,int fz_c,int fm,int fm_c){
    memset(num1,0,sizeof(num1));
    memset(num2,0,sizeof(num2));
    while(fz)
        ++num1[fz%10],fz /= 10;
    while(fz_c)
        --num1[fz_c%10],fz_c /= 10;
    while(fm)
        ++num2[fm%10],fm /= 10;
    while(fm_c)
        --num2[fm_c%10],fm_c /= 10;
    for(int i = 0;i < 10;i++){
        if(num1[i] != num2[i])
            return 0;
    }
    return 1;
}

bool wro_check(int a,int b){
    memset(nums_flag,0,sizeof(nums_flag));
    while(a){
        nums_flag[a%10] = 1;
        a /= 10;
    }
    while(b){
        if(nums_flag[b%10])
            return 0;
        b /= 10;
    }
    return 1;
}

void solve(){
```

```
int ans_num = 0;
    for(int fz = l1; fz <= r1;fz++)
        for(int fm = l2; fm <= r2;fm++){
            int fs_gcd = gcd(fz,fm);
            if(fs_gcd == 1){
                if(wro_check(fz,fm))
                    ans[ans_num++] = num(fz,fm,fz,fm);
                continue;
            }
            int fz_c = fz/fs_gcd , fm_c = fm/fs_gcd;
            if(wro_check(fz_c,fm_c) && contain_judge(fz,fz_c) &&
contain_judge(fm,fm_c) && value_check(fz,fz_c,fm,fm_c))
                ans[ans_num++] = num(fz,fm,fz_c,fm_c);
        }
    std::sort(ans,ans+ans_num,cmp);
    printf("%d\n",ans_num);
    for(int i = 0;i < ans_num;i++)
         printf("%d/%d=%d/%d\n",ans[i].z1,ans[i].m1,ans[i].z2,ans[i].m2);
    return;
}

int main(){
    scanf("%d %d",&l1,&r1);
    scanf("%d %d",&l2,&r2);
    solve();
    return 0;
}
```