D 求生者

出题人:梅子酒

考察点:概率DP

题意

每次给定僵尸血量 h_i，和离玉米投手的距离 x_i，玉米投手可能投出玉米粒（伤害 20）和黄油（伤害 40，定身 4 秒，可以叠加定身时间），投出的黄油的概率随僵尸的靠近提高。问僵尸靠近玉米投手之前就被击杀的概率。数据范围为：$h_i, x_i(1 \le h_i \le 4 * 10^3, 1 \le x_i \le 10^9)$。

思路

这是一个概率DP，我们从需要维护的状态来考虑。

1. 僵尸离玉米投手的距离，因为投出的黄油会随着这个数据变化所以一定要维护，并且这也是结束状态的一种（僵尸被击杀前就靠近玉米投手）。
2. 僵尸的血量，结束状态之一（靠近之前就被击杀）。
3. 僵尸被定住的状态，我们这里取被定住的时间来维护。

于是状态就知道了，第一维僵尸的剩余血量 h_i，第二维僵尸的位置 x_i（或者僵尸已经前进的距离），第三维被定住的时间 t_i，这样的状态意义就是 $f[i][j][k]$：当僵尸血量还剩下 i 的血量，已前进了 j 格，还剩下 k 秒定身时间的概率。但是从数据范围来看，我们的空间需要开到 $4 \times 10^3 \times 10^9 \times 300 = 1.2 \times 10^{15}$，时间复杂度也是这个量级，时空复杂度全面爆炸，我们来考虑如何优化。

我们注意到僵尸的血量只有 4×10^3，就算只投出玉米粒也只需要 $\frac{4 \times 10^3}{20} = 200$ 次就可以击杀，也就是说僵尸最多前进 200 格就一定会被击杀，所以不用考虑距离太远的僵尸的情况，第二维距离 10^9 的量级就降低到 200。

同样我们也注意到一次伤害至少也是 20 点，没必要将血量精确到个位数，我们可以将血量换算成最少需要几次玉米粒才能击杀（黄油的伤害相当于两个玉米粒），于是我们将第一维血量 10^3 的量级降低到 200。

而定身时间，最多 100 次黄油就一定能击杀任何僵尸，每次投出黄油到下一次投出黄油之间至少间隔一秒，所以定身时间最多为 $100 \times 4 - 100 = 300$。 于是时空复杂度就降低到 $200 \times 200 \times 300 = 1.2 \times 10^7$，这是我们可以接受的一个时间复杂度。

并且我们存在僵尸血量过低，或距离过远的情况，数据不一定能跑满，10 组极限数据的情况下也最多是 1.2×10^8 的量级，满足 1 秒的时间限制。具体如何转移，看代码和注释理解。

代码

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

#define ll long long

const int N = 210, mod = 998244353;

ll ksm(ll a, ll b){
```

```cpp
    ll res = 1;
    while(b){
        if(b & 1) res = res * a % mod;
        a = a * a % mod;
        b >>= 1;
    }
    return res;
}

ll f[N][N][310]; // 当僵尸血量还剩余需要 i 次攻击，已经前进 j 格，还剩下 k 秒的定身时间的概率
void add(ll& x, ll y){
    x = (x + mod + y) % mod;
}

void solve(){
    int h, x;
    cin >> h >> x;

    h = (h + 19) / 20; // 转变成至少需要的玉米粒攻击的次数

    if(h <= x){
        cout << "1\n";
        return ;
    }

    memset(f, 0, sizeof f);
    f[h][0][0] = 1;
    ll inv = ksm(100, mod - 2); // 100 的乘法逆元

    for(int i = h; i > 0; i --){
        for(int j = 0; j < x; j ++){
            for(int k = 0; k <= 300; k ++){
                if(!f[i][j][k]) continue ;
                // 概率
                ll p = min(99LL, (25LL + j / 5)) * inv % mod, q = (1 + mod - p) % mod; // 投出黄油和玉米粒的概率

                // 投出黄油的转移方程
                add(f[max(0, i - 2)][j][k + 3], f[i][j][k] * p % mod);

                // 投出玉米粒的方程转移
                if(k) add(f[i - 1][j][k - 1], f[i][j][k] * q % mod); // 减少一秒定身时间
                else add(f[i - 1][j + 1][k], f[i][j][k] * q % mod); // 没有被定身，前进一格
            }
        }
    }

    ll ans = 0;
    for(int j = 0; j <= 200; j ++){
        for(int k = 0; k <= 300; k ++){
            ans = (ans + f[0][j][k]) % mod;
        }
```

```
    }
    cout << ans << "\n";
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);

    int n;
    cin >> n;
    while(n --){
        solve();
    }
    return 0;
}
```