Millionaire (2008 APAC local onsites C)

题目

玩家持有X元钱，进行M轮赌博游戏。每一轮可以将所持的任意一部分钱作为赌注（赌注为0元表示这一轮不押），赌注可以是小数的，不是一定要整数。每一轮 赢的概率为P，赢了赌注翻倍，输了赌注就没了。如果你最后持有至少1000000元钱的话，就可以把钱全部带走。要求计算在采取最优策略时，获得至少 1000000元钱的概率。

数据范围：

- 0<=P<=1
- 1<=X<=1000000
- 1<=M<=15

解题思路

由于题目中允许有小数的赌注，因此无法穷举搜索，但我们思考最后一轮，假如给定如下条件M=1,P=0.5，则有以下三种结论

- 当前持有1000000元以上的钱，则可以不参加博弈，有1的概率带钱回家
- 当前持有500000元以上的钱，则全部投入该次博弈，有0.5的概率带钱回家
- 如果持有钱不到500000，则带钱回家的概率为0

通过上述分析，可以清晰的看到，带钱回家的概率是分阶段的，某一范围内带钱回家的概率是相同的，因此我们化连续为离散，此外，博弈的场次之间明显的迭代性，因此考虑动态规划，我们如下定义状态

> dp[i][j] 表示第i轮赌博时，手中持有的钱数为第j阶段的采取最优策略后带钱回家的概率

根据我们的状态定义，可得到迭代的公式为

> dp[i+1][j] = max(P*dp[i][j+k]+(1-P)*dp[i][j-k]) 其中 0<=k<=min(j,n-j),n为依据赌博次数定下的总阶段数

对于上述等式的理解，我们第i+1场赌博中所持金钱在第j阶段的带钱回家最大概率是第i场赌博中手中所持金钱在j阶段之前的情况中我们赌赢了，即P*dp[i][j+k]的概率加上第i场赌博中我们输了，但存在如果所持金钱在j-k阶段时满足带钱回家的概率，即(1-P)*dp[i][j-k]

解题代码

```
//这里由于第i+1只和第i阶段有关联，因此直接使用一维数组保存中间结果
int M,X;
double P;
double dp[2][(1 << MAX_N) + 1];
int main()
{
    //freopen("input.txt","r",stdin);
    //freopen("output.txt","w",stdout);
    ios_base::sync_with_stdio(0);cin.tie(0);
    int n = 1 << M;
    double *pre = dp[0],*nxt = dp[1];
    memset(pre,0,sizeof(double) * (n + 1));
    pre[n] = 1.0;
    for(int r = 0;r < M;r++)
    {
        for(int i = 0;i <= n;i++)
        {
            int jub = min(i,n - i);
            double t = 0.0;
            for(int j = 0;j <= jub;j++)
                t = max(t,pre[i + j] * P + pre[i - j] * (1 - P));
            nxt[i] = t;
        }
        swap(pre,nxt);
    }
    int i = (LL)X * n / 1000000;
    printf("%.6f\n",pre[i]);
    return 0;

}
```