Q1

25 Points

For those of you who follow professional sports, you will know that there are many team sports that are clock-based, i.e., the game lasts a fixed amount of time and the winner is the team that scores the most points or goals during that fixed time.

In all of these sports (e.g. basketball, football, soccer, hockey), you will notice that near the end of the game, the team that is behind plays very aggressively (in order to catch up), while the team that is ahead plays very conservatively (a practice known as `stalling`, `stonewalling`, and `killing the clock`).

In this problem we will explain why this strategy makes sense, through a simplified game that can be solved using Dynamic Programming. This game lasts n rounds, and you start with 0 points. You have two fair coins, which we will call X and Y. The number n is known to you before the game starts.

In each round, you select one of the two coins, and flip it. If you flip coin X, you gain 1 point if it comes up Heads, and lose 1 point if it comes up Tails. If you flip coin Y, you gain 3 points if it comes up Heads, and lose 3 points if it comes up Tails. After n rounds, if your final score is $positive$ (i.e., at least 1 point), then you win the game. Otherwise, you lose the game.

All you care about is winning the game, and there is no extra credit for finishing with a super-high score. In other words, if you finish with 1 point that is no different from finishing with $3n$ points. Similarly, every loss counts the same, whether you end up with 0 points, -1 point, -2 points, or $-3n$ points.

Because you are a Computer Scientist who understands the design and analysis of optimal algorithms, you have figured out the best way to play this game to maximize your probability of winning. Using this optimal strategy, let $p_r(s)$ be the probability that you win the game, provided there are r rounds left to play, and your current score is s. By definition, $p_0(s) = 1$ if $s \geq 1$ and $p_0(s) = 0$ if $s \leq 0$.

Q1.1

3 Points

Clearly explain why $p_1(s) = 0$ for $s \leq -3$, $p_1(s) = \frac{1}{2}$ for $-2 \leq s \leq 1$, and $p_1(s) = 1$ for $s \geq 2$.

If s <= -3, we can pick up either X or Y ,because the score s will always be negative. So, the probability to win the game is 0.

If -2<=s<=1, we can pick up Y, then we will get 1<=s <=4 or -5 <= s <= -2, so, the probability to win the game is 1/2;

If s>=2, we can pick up X, so the score s always larger than 0, so, the probability to win the game is 1.

No files uploaded

Q1.2

2 Points

Explain why you must select X if s is 2 or 3, and you must select Y if s is -2 or -1.

If the score s = 2 or s = 3, we will get the result is 1, 2, 3, and 4 if we pick up X and all of them are positive score, so the probability is 1. But, we will get -1, 0, 5, and 6 if we pick up Y and the probability is 1/2. So, we choose to X.

If the score s = -2 or -1, we will get the result is -5, -4, 1 and 2 if we pick up Y, and so the probability is 1/2. But, we will get the result is -3, -2, -1 and 0 and the probability is 0 if we pick up X. So, we choose to Y.

No files uploaded

Q1.3

5 Points

For each possible value of s, determine $p_2(s)$. Clearly explain how you determined your probabilities, and why your answers are correct. (Hint: each probability will be one of $\frac{0}{4}$, $\frac{1}{4}$, $\frac{2}{4}$, $\frac{3}{4}$, or $\frac{4}{4}$.)

From the hints and question, we can know that P2(3) = 4/4, P2(2) = 3/4, P2(1) =3/4, P2(0)=2/4,P2(-1) = 2/4, P2(-2) = 1/4, P2(-3) = 1/4, P2(-4) = 1/4, P2(-5) = 1/4 and P2(-6) = 0, and if s less than -6, P2(s) = 0;

So, when s have different number of range ,we will get different probability.
s >= 3, then P2(s) = 4/4;
1<= s <= 2, then P2(s) = 3/4;
-1 <= s <= 0, then P2(s) = 2/4;
-5 <= s <= -2, then P2(s) = 1/4;
s <= -6, then P2(s) = 0
So, we can clearly explain those probabilities.

No files uploaded

Q1.4
10 Points

Find a recurrence relation for $p_r(s)$, which will be of the form $p_r(s) = \max(\frac{??+??}{2}, \frac{??+??}{2})$. Clearly justify why this recurrence relation holds.

From the question, we can apply for the recurrence and then we can get:
Pr(s) = max(Pr-1(s+1) + Pr-1(s-1) / 2 , Pr-1(s+3)+Pr-1(s-3) / 2)
We know that the probability for each round s is just based on the max value of X or Y.
So, if we can pick up X, the probability on the previous round result will be s +1 or s -1.
But, if we pick up the Y, we will use the same logic as we picked up X, so the the probability on the previous round result will be s + 3 or s - 3.
So, we can justify the recurrence relation.

No files uploaded

From your recurrence relation, explain why the optimal strategy is to pick X when you have certain positive scores (be conservative) and pick Y when you have certain negative scores (be aggressive).

To explain clearly, I'll take an example for s = 2 and r = 1;
then, we can apply for the recurrence relation and get:
P1(2) = max(P0(3) + P0(1)/2, P0(5)+P0(-1)/2);
= max(1+1/2, 1+0/2)
= 1;
So, when we pick up the X ,we will get the probability is 1, and for picking up Y, we will get p = 1/2;
So, when we have certain positive scores, the P(X) > P(Y), that said, we will pick up X.

Similar logic we can apply for the negative scores.
let s = -2 and r = 1;
then, we can get:
P1(-2) = max (P0(-3) + P0(-1)/2, P0(1) + P(-5)/2);
= max(0+0/2, 1+0/2)
= 1/2;
So, we can clearly get P = 0 when we pick up X, and P = 1/2 when we pick up Y.
So, when we have certain negative score, we will pick up Y.

No files uploaded

Q1.5

5 Points

Compute the probability $p_{100}(0)$, which is the probability that you win this game if the game lasts $n = 100$ rounds. Use Dynamic Programming to efficiently compute this probability.

The probability is 0.71. Please see my attached file for details.

▼ Q1.5.java Download

```
public class Main {
    public static void main(String[] args){
        // Drive the code
        System.out.println(dp(100,0));
    }

    public static double dp(int r, int s){
        double[] dp = new double[r*6+1];

```

```
        for(int i = 3*r+1; i < dp.length; i++){
            dp[i] = 1;
        }
        // new four variables for each of X and Y probability
        double winX, winY, loseX, loseY;
        for(int i = 1; i < r+1; i++){
            double[] tempdp = new double[r*6+1];

            for(int j = 0; j < dp.length; j++){
                if(j == dp.length - 1){
                    winX = 1;
                }else{
                    winX = dp[j+1];
                }

                if(j == 0){
                    loseX = 0;
                }else{
                    loseX = dp[j-1];
                }

                if(j >= dp.length - 3){
                    winY = 1;
                }else{
                    winY = dp[j+3];
                }

                if(j <= 2){
                    loseY = 0;
                }else{
                    loseY = dp[j-3];
                }

                double Xchoice = (loseX + winX) / 2, Ychoice = (loseY + winY) / 2;
                tempdp[j] = Math.max(Xchoice, Ychoice);
            }
            dp = tempdp;
        }
        return dp[s+3*r];
    }
}
```

Problem Set 8

GRADED

STUDENT

Kejian Tong

TOTAL POINTS

25 / 25 pts

QUESTION 1

(no title) **25** / 25 pts

1.1	(no title)	**3** / 3 pts
1.2	(no title)	**2** / 2 pts
1.3	(no title)	**5** / 5 pts
1.4	(no title)	**10** / 10 pts
1.5	(no title)	**5** / 5 pts