Stat342-Lab1

Ben Liu & Blayne Downer

2025-02-21

Question 1

```
import math
import random

random.seed(21)

def run_simulation_question1(num_games, totalMoney):

    playerAwins = 0
    playerBwins = 0
    for i in range(num_games):
        if play_game(totalMoney):
            playerAwins += 1
        else:
            playerBwins += 1

    print(f"Player A wins {playerAwins} times")
    print(f"Player B wins {playerBwins} times")

    print(f"Probability that player A wins: {playerAwins/num_games}")


def play_game(totalMoney):
    n = totalMoney
    i = n/2
    p = 0.49
    while(i > 0 and i < n):
        if play_round(p):
            i += 1
        else:
            i -= 1

    return i == n # true if player A wins, false if player B wins


def play_round(prob): # i = player 1's money, j = player 2's money

    # return true if player 1 wins,
    # return false if player 2 wins
    # generate a random number between 0 and 1
    r = random.random()

    if r <= prob:
        return True
    else:
        return False



if __name__ == "__main__":

    run_simulation_question1(1000, 20)
    run_simulation_question1(1000, 100)
    run_simulation_question1(1000, 200)
```

```
## Player A wins 388 times
## Player B wins 612 times
## Probability that player A wins: 0.388
## Player A wins 113 times
## Player B wins 887 times
## Probability that player A wins: 0.113
## Player A wins 11 times
## Player B wins 989 times
## Probability that player A wins: 0.011
```

The simulated/emperical probability that A wins the game given values of N is:

$N = 20 : 0.388$

$N = 100 : 0.113$

$N = 200 : 0.011$

Question 2

The probability that A wins the game given values of N is approximately:

$N = 20 : 0.4013$

$N = 100 : 0.1192$

$N = 200 : 0.01798$

As N approaches infinity, it seems that the probability that A wins the game approaches zero. Here's a simulation with $N = 1000$

```
import math
import random

random.seed(21)

def run_simulation_question1(num_games):

    playerAwins = 0
    playerBwins = 0
    for i in range(num_games):
        if play_game():
            playerAwins += 1
        else:
            playerBwins += 1

    print(f"Player A wins {playerAwins} times")
    print(f"Player B wins {playerBwins} times")

    print(f"Probability that player A wins: {playerAwins/num_games}")


def play_game():
    n = 1000
    i = n/2
    p = 0.49
    while(i > 0 and i < n):
        if play_round(p):
            i += 1
        else:
            i -= 1

    return i == n # true if player A wins, false if player B wins


def play_round(prob): # i = player 1's money, j = player 2's money

    # return true if player 1 wins,
    # return false if player 2 wins
    # generate a random number between 0 and 1
    r = random.random()

    if r <= prob:
        return True
    else:
        return False



if __name__ == "__main__":

    run_simulation_question1(1000)
```

```
## Player A wins 0 times
## Player B wins 1000 times
## Probability that player A wins: 0.0
```

Player *A* didn't win a single game!

Question 3

```
import math
import random

random.seed(21)

def run_simulation_q3(num_games, totalMoney, probWin, playAStartingAmount):
    playerAwins = 0
    playerBwins = 0
    for i in range(num_games):
        if play_game_q3(totalMoney, probWin, playAStartingAmount):
            playerAwins += 1
        else:
            playerBwins += 1

    print(f"Player A wins {playerAwins} times")
    print(f"Player B wins {playerBwins} times")
    print(f"Probability that player A wins: {playerAwins/num_games}")

def play_game_q3(totalMoney, probWin, playAStartingAmount):
    n = totalMoney
    i = playAStartingAmount
    p = probWin
    while(i > 0 and i < n):
        won = play_round_q3(p)
        if won == 0:
            i += 1
        elif won == 1:
            i -= 1
        else:
            continue
    return i == n # if player A wins = true, if player B wins = false
def play_round_q3(p):

    r = random.random()
    if r <= p:
        return 0   # Player A wins
    elif r > p and r <= 2*p:
        return 1   # Player B wins
    elif r > 2*p:
        return 2   # Tie

if __name__ == "__main__":

    print(f"N = 20, P(tie) = .02, i = N/2")
    run_simulation_q3(1000, 20, .49, 10)
    print(f"N = 200, P(tie) = .02, i = N/2")
    run_simulation_q3(1000, 200, .49, 100)
    print(f"N = 20, P(tie) = .2, i = N/2")
    run_simulation_q3(1000, 20, .40, 10)
    print(f"N = 20, P(tie) = .8, i = 3N/4")
    run_simulation_q3(1000, 20, .49, 15)
```

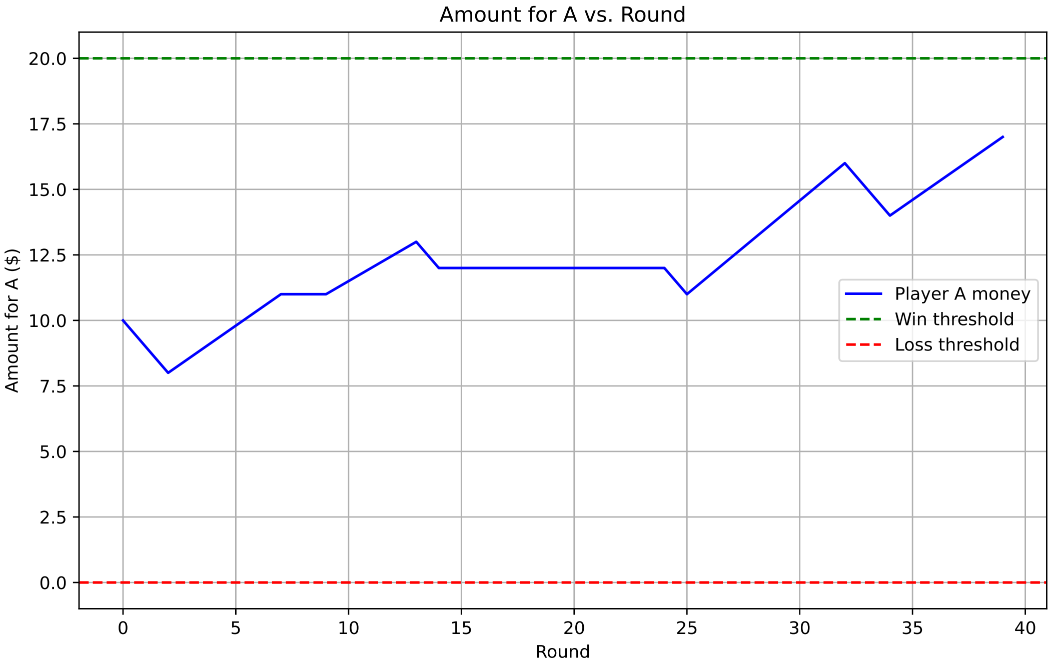
```
## N = 20, P(tie) = .02, i = N/2
## Player A wins 488 times
## Player B wins 512 times
## Probability that player A wins: 0.488
## N = 200, P(tie) = .02, i = N/2
## Player A wins 532 times
## Player B wins 468 times
## Probability that player A wins: 0.532
## N = 20, P(tie) = .2, i = N/2
## Player A wins 507 times
## Player B wins 493 times
## Probability that player A wins: 0.507
## N = 20, P(tie) = .8, i = 3N/4
## Player A wins 756 times
## Player B wins 244 times
## Probability that player A wins: 0.756
```

The code above messes around with the total money (N), the chance of a tie, and player A's starting amount (i). For cases when $i = N/2$, the probability of A winning was about .5 (which is equal to i/N), regardless of the size of N and the probability of a tie. When $i = 3N/4$, the probability of A winning was about .75, as expected.

Question 4

Part a:

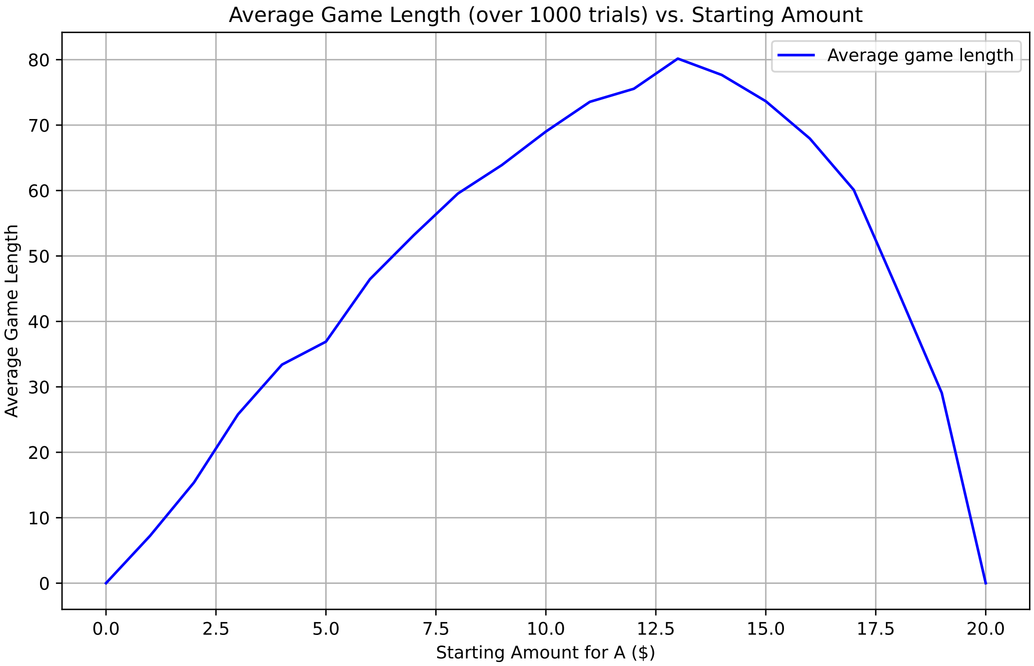
```python
import math
import random
import matplotlib.pyplot as plt

def run_simulation(num_games):

    playerAwins = 0
    playerBwins = 0
    for i in range(num_games):
        if play_game(initial_amount=10, total_dollars=20, p=0.49):
            playerAwins += 1
        else:
            playerBwins += 1

    print(f"Player A wins {playerAwins} times")
    print(f"Player B wins {playerBwins} times")

    print(f"Probability that player A wins: {playerAwins/num_games}")


def play_game(initial_amount, total_dollars, p):
    n = total_dollars
    i = initial_amount
    rounds = 0

    while(i > 0 and i < n):
        rounds += 1
        if play_round(p):
            i += 1
        else:
            i -= 1


    return i == n, rounds  # true if player A wins, false if player B wins


def play_round(prob): # i = player 1's money, j = player 2's money

    # return true if player 1 wins,
    # return false if player 2 wins
    # generate a random number between 0 and 1
    r = random.random()

    if r <= prob:
        return True
    else:
        return False


def q4a_plot():
    n=20
    i=n/2
    p=0.5
    rounds = 0

    i_values = [i]
    round_numbers = [0]

    while(i > 0 and i < n):
        rounds += 1
        if play_round(p):
            i += 1
        else:
            i -= 1
            i_values.append(i)
            round_numbers.append(rounds)

    plt.figure(figsize=(10, 6))
    plt.plot(round_numbers, i_values, '-b', label='Player A money')
    plt.axhline(y=n, color='g', linestyle='--', label='Win threshold')
    plt.axhline(y=0, color='r', linestyle='--', label='Loss threshold')
    plt.xlabel('Round')
    plt.ylabel('Amount for A ($)')
    plt.title('Amount for A vs. Round')
    plt.legend()
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    q4a_plot()
```



Part b:

```
import math
import random
import matplotlib.pyplot as plt

def run_simulation(num_games):

    playerAwins = 0
    playerBwins = 0
    for i in range(num_games):
        if play_game(initial_amount=10, total_dollars=20, p=0.49):
            playerAwins += 1
        else:
            playerBwins += 1

    print(f"Player A wins {playerAwins} times")
    print(f"Player B wins {playerBwins} times")

    print(f"Probability that player A wins: {playerAwins/num_games}")


def play_game(initial_amount, total_dollars, p):
    n = total_dollars
    i = initial_amount
    rounds = 0
```

```python
    while(i > 0 and i < n):
        rounds += 1
        if play_round(p):
            i += 1
        else:
            i -= 1


    return i == n, rounds  # true if player A wins, false if player B wins


def play_round(prob): # i = player 1's money, j = player 2's money

    # return true if player 1 wins,
    # return false if player 2 wins
    # generate a random number between 0 and 1
    r = random.random()

    if r <= prob:
        return True
    else:
        return False

def q4b_plot():
    n=20
    p=0.44
    startingAmounts = [i for i in range(21)]
    average_game_length = []
    for j in startingAmounts:
        total_length = 0
        num_games = 1000
        for i in range(num_games):
            _, rounds = play_game(j,n, p)
            total_length += rounds
        average_game_length.append(total_length/num_games)

    plt.figure(figsize=(10, 6))
    plt.plot(startingAmounts, average_game_length, '-b', label='Average game length')
    plt.xlabel('Starting Amount for A ($)')
    plt.ylabel('Average Game Length')
    plt.title('Average Game Length (over 1000 trials) vs. Starting Amount')
    plt.legend()
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    q4b_plot()
```


Average Game Length (over 1000 trials) vs. Starting Amount
Average game length
Average Game Length
80
70
60
50
40
30
20
10
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
Starting Amount for A ($)