Esercizio 12* del Foglio 3

Dennis Parolin 2113203, Tommaso Ceron 2101045, Lorenzo Soligo 2101057

Marzo 2025

1 Modello Matematico

Il codice simula un gioco stocastico in cui pedine competono per raggiungere una posizione target (19) attraverso movimenti determinati dalla somma di due dadi. Il calcolo delle probabilità è ottenuto seguendo la regola di Montecarlo, che approssima una soluzione rigoroso ad una basata sulla simulazione. Di seguito la formalizzazione matematica. Le diverse probabilità sono infatti calcolate come semplice rapporto tra casi favorevoli e casi totali.

1.1 Dinamica del Gioco

- Sia P=p1,p2,. . . ,p12P=p1,p2,. . . ,p12 l'insieme delle pedine, inizializzate a posizione 00.
- Ad ogni mossa, si lanciano due dadi a 6 facce. La somma D2,3,. . . ,12D2,3,. . . ,12 determina la pedina da muovere.
- La probabilità P(D=k)P(D=k) per k2,. . . ,12k2,. . . ,12 è nota (Tabella **??**).
- Se pk1¡19pk1¡19, la pedina corrispondente a kk avanza di 1.
- Il gioco termina quando una pedina raggiunge 19. Sia TT il numero di mosse necessarie.

1.2 Probabilità di Vittoria (Punti A e B)

Le probabilità che arrivi prima la pedina in posizione 7 rispetto alla 8 (A) e viceversa (B) è equivalente al calcolo della probabilità che arrivi per prima la 7 (A) o prima la 8 (B) in quanto il gioco si interrompe una volta che una pedina raggiunge la fine. Viene quindi effettuato un conteggio di quante tra le partite simulate terminano con il numero 6 (caso A) o il numero 7 (caso B) e viene divisa per il numero di simulazioni totali.

1.3 Durata del Gioco (Punti C, D, E)

Lo stesso principio viene applicato anche in questo caso conteggiando i casi e dividendo per le simulazioni totali.

2 Pseudocodice

```
import random
import matplotlib.pyplot as plt

def simulate_game():
    # Inizializzazione la scacchiera
    pedine = [0] * 12
    mosse = 0

    while True:
        # Lancia dadi
        dadi = random.randint(1, 6) + random.randint(1, 6)

        # Mossa della pedina scelta
        if 2 <= dadi <= 12:
            colonna = dadi
            if pedine[colonna - 1] < 19:  # Se la pedina non è arrivata alla
                                            fine, la muovo
                pedine[colonna - 1] += 1

        mosse += 1

        # Se almeno una pedina ha raggiunto la fine, termina il gioco
        if any(pedina == 19 for pedina in pedine):
            break

    return pedine.index(19), mosse  # Ritorna la pedina vincente

# Simulazione di più giochi
simulazioni = 200000
risultati = [simulate_game() for _ in range(simulazioni)]

PUNTO A-------------------------------------------------------------------------#

# Calcolo probabilità che la pedina 7 arrivi prima della 8
arrivo_colonna_7 = sum(1 for risultato in risultati if risultato[0] == 6) / simulazioni
arrivo_colonna_8 = sum(1 for risultato in risultati if risultato[0] == 7) / simulazioni

print(f"Probabilità che la pedina della colonna 7 arrivi prima
        di quella della colonna 8: {arrivo_colonna_7}")
print(f"Probabilità che la pedina della colonna 8 arrivi prima
        di quella della colonna 7: {arrivo_colonna_8}")

PUNTO B-------------------------------------------------------------------------#
```

```
# Calcolare la probabilità che la pedina k arrivi per prima
probabilità_colonne = [sum(1 for risultato in risultati if risultato[0] == k) / simulazioni
print("Probabilità che la pedina della colonna k arrivi per prima:", probabilità_colonne)

PUNTO C-------------------------------------------------------------------------#

# Calcolare la probabilità che il gioco duri esattamente N mosse (con N tra 1 e 200)
durata_esatta = [0] * 200  # Lista del numero di volte in cui un gioco
                                        e' durato esattamente N mosse
giochi_piu_di_200_mosse = 0  # Contatore dei giochi con durata maggiore di 200 mosse

# Conta le durate
for _, mosse in risultati:
    if 1 <= mosse <= 200:
        durata_esatta[mosse - 1] += 1
    elif mosse > 200:
        giochi_piu_di_200_mosse += 1

# Probabilita' di durata esatta per ogni N
probabilità_durata_esatta = [count / simulazioni for count in durata_esatta]

for N in range(1, 201):
    print(f"Probabilità che il gioco duri esattamente {N} mosse:
                {probabilità_durata_esatta[N - 1]}")

PUNTO D-------------------------------------------------------------------------#

# Calcolare probabilita' dei giochi durati piu' di 100 mosse
giochi_piu_di_100_mosse = sum(1 for _, mosse in risultati if mosse > 100)
probabilità_piu_di_100_mosse =
            (giochi_piu_di_100_mosse + giochi_piu_di_200_mosse) / simulazioni
print(f"Probabilità che il gioco duri più di 100 mosse: {probabilità_piu_di_100_mosse}")

PUNTO E-------------------------------------------------------------------------#

# Calcolare probabilita' dei giochi durati piu' di 200 mosse
probabilità_piu_di_200_mosse = giochi_piu_di_200_mosse / simulazioni
print(f"Probabilità che il gioco duri più di 200 mosse: {probabilità_piu_di_200_mosse}")
```

3 Grafici

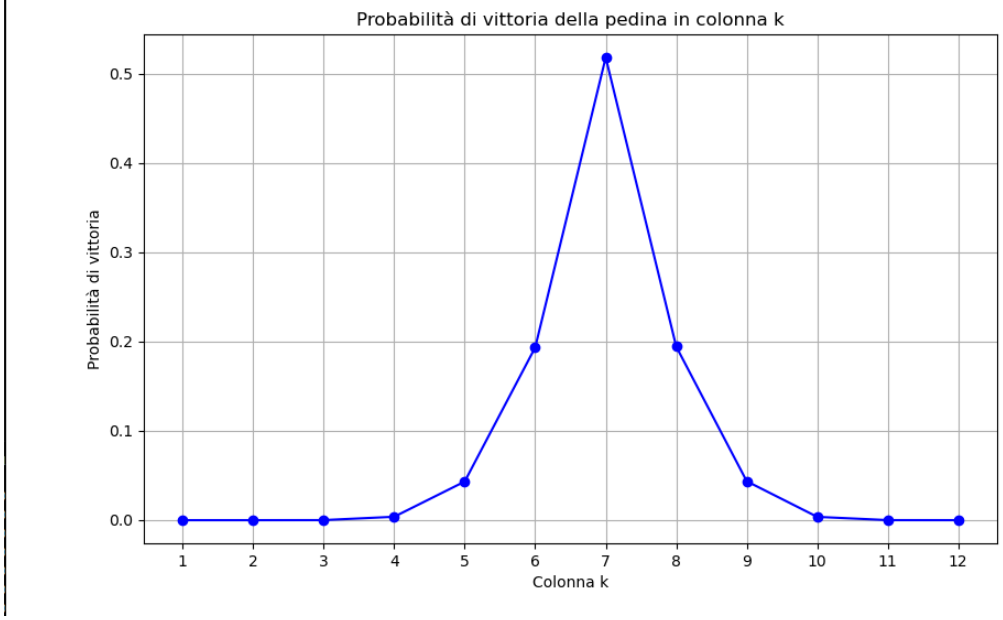

Figure 1: Probabilità di vittoria della k colonna

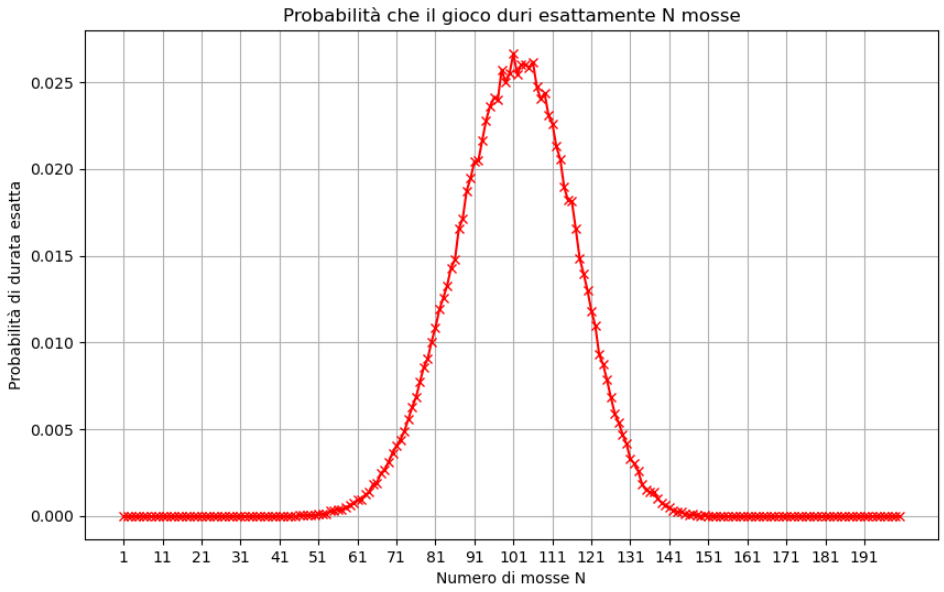

Figure 2: Probabilità di terminare in k mosse