Univerza v Ljubljani
Fakulteta za matematiko in fiziko

Finančna matematika – 1. stopnja

Urška Komatar, Ian Lampič

Intransitive dice

Projekt pri predmetu Finančni praktikum

Ljubljana, 2023

Kazalo

1. Predstavitev osnovnega problema

Naj bo $p > \frac{1}{2}$ konstanta, A, B in C pa 6 - strane kocke z različnimi števili na svojih straneh, tako da kocka A premaga B z verjetnostjo vsaj p, B premaga C z verjetnostjo vsaj p in C premaga A z verjetnostjo vsaj p. Želimo poiskati maksimalni p tako, da bo obstajala trojica kock, za katere bo veljala dana lastnost. Kasneje si bova ogledala še primere s štirimi kockami, trostranimi kockami, štiristranimi kockami in kockami s petimi stranmi.

1.1. **Ideja reševanja.** Najprej bova sestavila pomožno funkcijo, ki bo za argumente prejela seznam treh seznamov. V vsakem seznamu bo 6 naravnih števil, ki bodo predstavljale števila na kocki. Če so A, B in C slučajne spremenljivke, ki povedo, koliko pokaže posamezna kocka, bo torej funkcija vračala $min(P(A > B), P(B > C), P(C > A))$. Nato bi naredila krovno funkcijo, ki bi sprejela naravno število n. Znotraj te funkcije bi generirala seznam seznamov z elementi prvih n naravnih števil in na vsakem takem seznamu poklicala pomožno funkcijo. Medtem bi beležila dobljene verjetnosti in na koncu vrnila maksimum teh vrednosti. Za vsa nadaljna vprašanja bova funkcijo prilagodila danim zahtevam in ponovila postopek.

1.2. **Primeri netranzitivnih kock.**

1.3. **Primer.**

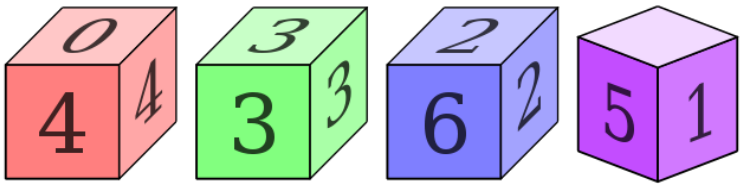
- A: 2, 2, 6, 6, 7, 7
- B: 1, 1, 5, 5, 9, 9
- C: 3, 3, 4, 4, 8, 8

$$P(A > B) = P(B > C) = P(C > A) = \frac{5}{9}$$

Primer, ko je $p = \frac{5}{9}$

1.3.1. *Efronove kocke.* Efronove kocke so set štirih netranzitivnih kock A, B, C in D z naslednjimi številkami na stranicah:

- A: 4, 4, 4, 4, 0, 0
- B: 3, 3, 3, 3, 3, 3
- C: 6, 6, 2, 2, 2, 2
- D: 5, 5, 5, 1, 1, 1



dice.png

Vsaka kocka je premagana s prejšnjo z verjetnostjo $\frac{2}{3}$:

$$P(A > B) = P(B > C) = P(C > D) = P(D > A) = \frac{2}{3}$$

B ima na vseh straneh enako število (B je konstantna). A ima 4 od 6 števil višja od B, torej verjetnost da A premaga B je $\frac{2}{3}$, medtem ko C pa ima 4 od 6 števil manjša od B, torej B premaga C z verjetnostjo $\frac{2}{3}$.
$P(C > D)$ izračunamo s seštevanjem pogojnih verjetnosti:

- C pokaže 6 ($P = \frac{1}{3}$); premaga D z verjetnostjo 1

- C pokaže 2 ($P = \frac{2}{3}$); premaga D le, če D pokaže 1 ($P = \frac{1}{2}$)

Torej verjetnost da C premaga D je naslednja:

$$(\frac{1}{3} \cdot 1) + (\frac{2}{3} \cdot \frac{1}{2}) = \frac{2}{3}$$

S podobnim izračunom dobimo še verjetnost, da D premaga A.

2. Reševanje osnovnega problema

2.1. **Primerjava kock.** V prvem koraku sva naredila funkcijo, ki sprejme dva seznama, ju razporedi po vrstnem redu ter šteje, koliko števil prvega seznama je večjih od števil iz drugega seznama. Seznama nam predstavljata dve kocki, števila v njih pa so števila na stranicah kock.

```
def comparison(sez1, sez2):
count = 0
p = []
sez1.sort()
sez2.sort()
for i in range(len(sez1)):
    for j in range(len(sez2)):
        if sez1[i] > sez2[j]:
            count += 1
    p.append(count)
    count = 0
return p
```

2.2. **intranzitivnost.** Opis

```
def intransitive(dice):
numb_of_dice = len(dice)
lst_of_probs = []
for i in range(numb_of_dice - 1):
    comparison(dice[i], dice[i+1])
    lst_of_probs.append((sorted(dice[i]), sorted(dice[i + 1]),
     sum(comparison(dice[i], dice[i+1]))))
lst_of_probs.append((sorted(dice[numb_of_dice-1]), sorted(dice[0]),
 sum(comparison(dice[numb_of_dice-1], dice[0]))))
prob = 100 #določena zgornja meja za minimum, ki ne bo presežena
for i in range(len(lst_of_probs)):
    prob = min(prob, lst_of_probs[i][2])
return prob, dice
```

2.3. **generator kocke.** Naslednja funkcija nam bo naredila seznam različnih naravnih števil, ki jih bo izbrala slučajno iz intervala $[1, m]$, za .. sides pa bo sprejela število stranic kocke. Torej dobili bomo seznam, ki bo predstavljal naključno kocko.

```
def generator_list(m, sides):
st = set()
while len(st) < sides:
    random_num = random.randint(1, m)
```

```
        st.add(random_num)
    return list(st)
```

2.4. **Končna funkcija.** Generira m ponovitev z *num_of_dice* kockami, ki imajo *sides* stranic, zgornja meja za generator seznama pa je n

```
def krovna(n, m, sides, num_of_dice):
lst = []
while len(lst) < m:
    dice = []
    for i in range(num_of_dice):
        dice.append(generator_list(n, sides))
    lst.append(intransitive(dice))
max_prob = 0
for i in range(len(lst)):
    if max_prob < lst[i][0]:
        max_prob = lst[i][0]
        lst_with_max = lst[i][1]
return max_prob, lst_with_max
```

3. Rezultati