חידת איסתרא

שלום לכולם,
לפניכם חידת איסתרא מאי 2023
שימו לב להוראות הגשה ואופן קביעת הזכייה בתחרות בסוף המסמך
בהצלחה לכולם :)

המשחק של מר לגינא - פיתוח אסטרטגיה לקבלת מקסימום רווח מובטח

במשחק הבא עליכם לכתוב אסטרטגיה שהרווח שלה במקרה הגרוע ביותר יהיה מקסימלי.

לפניכם שני כדים (ממוספרים 1,2) ובכל אחד מהם יש בדיוק N כדורים. באחד מהכדים (לא ידוע איזה) ישנם בדיוק K כדורים ירוקים ושאר הכדורים אדומים ואילו בכד האחר ישנים בדיוק K כדורים אדומים ושאר הכדורים ירוקים.
המשחק מתחיל כאשר נותנים לכם M אסימונים (ניתן להניח ש $M > 10N$) ועליכם לבצע סדרה של 2N ניחושים. בכל ניחוש אתם מחליטים על:

- הכד שמתוכו יוציא מר לגינא את הכדור הבא (מיוצג על ידי הערך is_first)
- מספר אסימונים חיובי v שאינו גדול ממספר האסימונים שנצברו בידיכם עד כה
- צבע הכדור (מיוצג על ידי הערך is_red)

אם צבעו של הכדור זהה לצבע שניחשתם תקבלו v אסימונים ממר לגינא, אחרת תאלצו להיפרד מ- v אסימונים ולהעבירם למר לגינא.

המשחק יסתיים כאשר יוצאו כל הכדורים משני הכדים או כאשר ייגמרו לכם האסימונים.

עליכם לכתוב פונקציית פייתון 3 (ייבדק על python 3.6.10) בשם bet אשר מקבלת כקלט את תקציר המשחק עד כה ואת כמות האסימונים שנותרה, ומחזירה את הניחוש כמתואר להלן:

הנחיות לכתיבת הפתרון:

- עליכם לממש את הפונקציה bet:

In [ ]:

```
# don't use  i m p o r t  in this file
def bet(N,K,m,g1,r1,g2,r2):
  """
      Returns a bet based on available tokens and balls fetched so
      far.

      Parameters
      ----------------
      N : int  - total number of balls in each urn
      K : int  - number of red balls in one urn and green balls in
                 the other urn
      m : int  - left number of tokens
      g1 : int - number of green balls fetched so far from first urn
      r1 : int - number of red balls fetched so far from first urn
      g2 : int - number of green balls fetched so far from second
                 urn
      r2 : int - number of red balls fetched so far from second urn

      Returns
      -----------
      (v, is_first, is_red) where:
      v: int          - Number of tokens for the bet (1 <= v <= m)
      is_first : bool - True if bet is on first urn and False
                        otherwise
      is_red : bool   - True if bet is on red ball and False
                        otherwise
  """
    # את הפונקציה הזאת אתם צריכים לממש
```

- חתימת הפונקציה (שם הפונקציה, שמות הפרמטרים וסדרם) חייבת להיות זהה להגדרה המופיעה לעיל.
- אפשר לכלול בקובץ פונקציות או מחלקות עזר אך ללא שימוש בספריות חיצוניות (import).
- ההימור שהפונקציה מחזירה חייב להיות חוקי (על כד שנותרו בו כדורים), אבל לא חייב להיות הגיוני (שימו לב שהפונקציה בדוגמא שבהמשך עלולה בהחלט להמר על צבע אדום כאשר ברור שכל הכדורים האדומים בכד כבר הוצאו).
- ניתן להניח שתקציר המשחק עד כה התקבל מקריאות קודמות לפונקציה זו ולכן אין צורך להתמודד עם מצבים שלא תגיעו אליהם. למשל אם האסטרטגיה בוחרת להוציא תמיד קודם מהכד הראשון היא לא חייבת להתמודד עם קלט שבו הוצאו כדורים קודם מן הכד השני.
- שימו לב שהאסטרטגיה נבחנת עפ"י ביצועיה במקרה הגרוע ביותר - עבור ערכים קטנים של N מומלץ לבדוק את כל הסידורים האפשריים .

דוגמא למימוש אפשרי:

In [ ]:

```
# don't use  i m p o r t  in this file
def bet(N,K,m,g1,r1,g2,r2):
  """
      Returns a bet based on available tokens and balls fetched so
      far.

      Parameters
      ----------------
      N : int  - total number of balls in each urn
      K : int  - number of red balls in one urn and green balls in
                 the other urn
      m : int  - left number of tokens
      g1 : int - number of green balls fetched so far from first urn
      r1 : int - number of red balls fetched so far from first urn
      g2 : int - number of green balls fetched so far from second
                 urn
      r2 : int - number of red balls fetched so far from second urn

      Returns
      -----------
      (v, is_first, is_red) where:
      v: int          - Number of tokens for the bet (1 <= v <= m)
      is_first : bool - True if bet is on first urn and False
                        otherwise
      is_red : bool   - True if bet is on red ball and False
                        otherwise
  """
    s=g1+r1+g2+r2
    if s < N:
        return (1, True, True) # Bet on 1 token that next ball
                               # from first urn will be red
    elif s < 2*N-1:
        return (1, False, True) # Bet on 1 token that next ball
                                # from second urn be red
    else:
        return(m, False, g2 == r1) # Choose red if g2==r1 and green
                                   # otherwise; Bet on m tokens that
                                   # next ball from second urn will
                                   # be of the chosen color.
```

סביבת הרצה:

In [ ]:

```
#no problem to import when testing your code
from importlib import reload
import math
import logging
import itertools
reload(logging)
LEVEL = logging.INFO
#LEVEL = logging.DEBUG

logging.basicConfig(level=LEVEL,
    format="%(levelname)s - %(asctime)s - %(message)s")

class Game:
    class Urn:
        def __init__(self,n,red_balls):
            self.size = n
            self.red_balls = red_balls
            self.next = 0
            self.count = {True:0,False:0}

        def get_next_ball(self):
            assert self.next < self.size, "out of range"
            is_red = self.next in self.red_balls
            self.next += 1
            self.count[is_red] += 1
            return is_red

    def __init__(self, total_tokens, N, K, red1, red2):
        self.tokens = self.total_tokens = total_tokens
        self.N = N
        self.K = K
        self.urn1 = self.Urn(N, red1)
        self.urn2 = self.Urn(N, red2)
        self.cur_iter = 0

    def get_urn(self, is_first):
        res_urn = self.urn1 if is_first else self.urn2
        return res_urn

    def get_ball(self, is_red, is_first, tokens):
        cur_urn = self.get_urn(is_first)
        assert tokens >=1 and tokens <= self.tokens , \
          "bad bet: tokens {} self.tokens {}".format(tokens,
                                                     self.tokens)
        return cur_urn.get_next_ball()
```

In [ ]:

```
def run(self):
    logging.debug("Starting tokens={} N={} K={}".format(self.tokens,
                                                        self.N,
                                                        self.K))
    while self.cur_iter < 2*self.N and self.tokens > 0:
        self.cur_iter += 1
        bet_tokens, is_first, bet_is_red =
            bet(self.N, self.K, self.tokens, self.urn1.count[False],
                self.urn1.count[True], self.urn2.count[False],
                self.urn2.count[True])
        cur_urn = self.get_urn(is_first)
        result_is_red = self.get_ball(bet_is_red, is_first,
                                      bet_tokens)
        bet_factor = 1 if result_is_red == bet_is_red else -1
        self.tokens += bet_factor*bet_tokens
        bet_urn="1" if is_first else "2"
        bet_color="r" if bet_is_red else "g"
        res="r" if result_is_red else "g"
        logging.debug(
            "{:-3}: [{},{}],[{},{}] {}=>{} ({}) left: {}".format(
            self.cur_iter, self.urn1.count[False],
            self.urn1.count[True], self.urn2.count[False],
            self.urn2.count[True], bet_color + bet_urn,
            res + bet_urn, bet_tokens, self.tokens))
    logging.debug("Finished with {}".format(self.tokens))
    return self.tokens

# Note that this function support small values of N as it runs over
# all combinations
def get_worst_case(M, N, K):
    assert N < 10, "only small values of N are supported"
    min_val = math.inf
    for sub1 in itertools.combinations(range(N),K):
        for sub2 in itertools.combinations(range(N),N-K):
            for pair in [(sub1,sub2), (sub2,sub1)]:
                min_val = min(min_val, Game(M, N, K, pair[0],
                                            pair[1]).run())

    logging.info(f"M:{M}, N:{N}, K:{K} - {min_val}")
    return min_val


get_worst_case(1000, 5, 3)
```

סביבת הרצה אופציונלית:

אם ברצונכם להתנסות בפתרון אונליין תוכלו להריץ דרך Repo Github
שימו לב, זאת סביבת הרצה בלבד, פורמט ההגשה מתואר למטה.

הוראות הגשה:

- החידה מיועדת לסטודנטים בטכניון בלבד
- את הפתרון יש לשלוח עד יום 23/05/2023 למייל: riddle@istraresearch.com
- כותרת המייל: Subject: Full Name
- בגוף המייל יש לציין שם מלא, מסלול לימודים ושנה
- הקוד יצורף למייל בקובץ Your_Name.py אשר כולל פונקציה bet כמתואר. אפשר לכלול בקובץ פונקציות או מחלקות עזר אך ללא שימוש בספריות חיצוניות (import).

קביעת הזוכה בתחרות:

- לכל קוד יחושב ציון ע"י תכנית בדיקה אשר תרוץ על מספר קלטים של הבעיה ובכל קלט (N,K,M) יחושב מספר האסימונים המינימלי שהאסטרטגיה השיגה כנגד מספר תרחישים. מספר האסימונים יחולק בחסם עליון המתאים לקלט הבעיה לקבלת ציון לכל קלט. הציון הכולל יחושב כסכום הציונים עבור כל הקלטים.
- קלטים קטנים ייבדקו כנגד כל התרחישים האפשריים ואילו קלטים גדולים ייבדקו גם כנגד אסטרטגית יריב (adversary) נגדית.

- זמן הריצה לכל אסטרטגיה יוגבל, והציון ייחשב רק עבור קלטים שיספיקו לעמוד במגבלת הזמן. שגיאה בריצה כלשהיא תגרום לציון 0 עבור הקלט בו קרתה השגיאה.

- במידה ויהיו מספר פתרונות בעלי ציונים זהים תתבצע הגרלה ביניהם.
- הכרזת הזוכה תתקיים ב 31/05/2023 - עקבו אחרינו- https://www.linkedin.com /company/istraresearch
- הפתרון הטוב ביותר יזכה בפרס: אוזניות קשת Sony -Ear-Over ANC B5XM-1000WH

אם יש לכם שאלות בואו לפגוש אותנו ביום רביעי, 17.5, בין השעות 12:30-14:30 בלובי טאוב (קומת הכניסה)
החל מהשעה 13:00 נקיים הרצאה "מבוא לאלגוטריידינג" בטאוב 3 (קומה 0)