אוניברסיטת תל אביב - בית הספר למדעי המחשב
מבוא מורחב למדעי המחשב, חורף 2016-2017

תרגיל בית מספר 4 - להגשה עד 25 בדצמבר (יום ראשון) בשעה 23:55

קיראו בעיון את הנחיות העבודה וההגשה המופיעות באתר הקורס, תחת התיקייה assignments. חריגה מההנחיות תגרור ירידת ציון / פסילת התרגיל.

הגשה:

- תשובותיכם יוגשו בקובץ pdf ובקובץ py בהתאם להנחיות בכל שאלה.
- השתמשו בקובץ השלד skeleton4.py כבסיס לקובץ ה py אותו אתם מגישים.

לא לשכוח לשנות את שם הקובץ למספר ת"ז שלכם לפני ההגשה, עם סיומת py.

- בסה"כ מגישים שני קבצים בלבד. עבור סטודנטית שמספר ת"ז שלה הוא 012345678 הקבצים שיש להגיש הם hw4_012345678.pdf ו- hw4_012345678.py.
- בכל השאלות ניתן להניח כי הקלט לתכנית / לפונקציות הינו תקין, אלא אם צוין במפורש אחרת.
- תשובות מילוליות והסברים צריכים להיות תמציתיים, קולעים וברורים.

שאלה 1

בשאלה זו ננתח ונשווה את הסיבוכיות של מספר פונקציות רקורסיביות לחישוב מקסימום ברשימה.

את שתי הגרסאות הבאות, max1 ו max2, פגשנו בתרגול 6, ואף ניתחנו את סיבוכיות הזמן ואת עומק הרקורסיה שלהן:

```
 def max1(L):
   if len(L)==1:
        return L[0]
    return max(L[0], max1(L[1:]))

def max2(L):
    if len(L)==1:
        return L[0]
    l = max2(L[:len(L)//2])
    r = max2(L[len(L)//2:])
    return max(l,r)
```

א. להלן הפונקציה max11, שהיא גרסה הדומה ל- max1, אך הפעם ללא slicing. פונקציה זו מקבלת בנוסף לרשימה L שני אינדקסים אשר תוחמים את אזור החיפוש הרלבנטי. הקריאה הרקורסיבית הראשונה מתבצעת מתוך פונקצית המעטפת max_list11:

```
def max11(L,left,right):
    if left==right:
        return L[left]
    return max(L[left], max11(L,left+1,right))

def max_list11(L):
    return max11(L,0,len(L)-1)
```

השימוש במנגנון כזה של פונקצית מעטפת מקובל מאוד, ומאפשר למשתמש להעביר לפונקציה אך ורק את הקלט הדרוש (הרשימה עצמה), ולא פרמטרים נוספים שקשורים לאופן מימוש רקורסיבי כזה או אחר (גבולות שמאלי וימני).

נתחו את סיבוכיות הזמן ואת עומק הרקורסיה של max11. על הניתוח לכלול:

1. ציור סכמטי של עץ הקריאות הרקורסיביות עבור רשימה באורך n. העץ יכיל צומת עבור כל קריאה רקורסיבית. בתוך הצומת רישמו את אורך הרשימה עליה בוצעה הקריאה (מספר איברי הרשימה שנמצאים בין האינדקסים הרלוונטים, כלומר את left-right+1), ולצד הצומת רישמו חסם אסימפטוטי הדוק במונחי O(…) על סיבוכיות הזמן של אותו צומת כפונקציה של n.
2. ציון עומק עץ הרקורסיה כפונקציה של n.
3. ציון סיבוכיות הזמן (חסם אסימפטוטי הדוק במונחי O(…)) כפונקציה של n.

ב. השלימו בקובץ השלד את הפונקציה max22, שתעבוד באותו אופן כמו max2 (כלומר תבצע חלוקה של הבעיה בדיוק לאותן תתי בעיות), אבל הפעם ללא slicing. הקריאה הראשונה ל- max22 תהיה מתוך max_list22, כפי שמופיע בקובץ השלד:

```
def max_list22(L):
    return max22(L,0,len(L)-1)
```

ג. נתחו את סיבוכיות הזמן ואת עומק הרקורסיה של max22 בדומה לניתוח שביצעתם בסעיף א' 1-3 עבור max11.

ד. השלימו את הטבלה הבאה, המציינת את זמני הריצה של ארבע הפונקציות max1, max2, max_list11, max_list22, עבור רשימות בגודל n=1000, 2000, 4000. את זמני הריצה מדדו באמצעות מנגנון ה"סטופר" שהוצג בתרגיל בית 1, או באמצעות הפונקציה elapsed מההרצאות. זכרו שכדי לקבל תוצאות אמינות עדיף להריץ מספר גדול של הרצות ולמצע. מאחר שכברירת מחדל עומק הרקורסיה המקסימלי הינו 1000, חלק מהפונקציות לא יצליחו לרוץ. לכן יהיה עליכם לשנות את עומק הרקורסיה המקסימלי לנניח 5000 באמצעות הפקודה sys.setrecursionlimit(5000) (עקבו אחר ההנחיות בשקפים של הרצאה 10).

Function	n = 1000	n = 2000	n = 4000
max1			
max2			
max_list11			
max_list22			

ה. הסבירו **<u>בקצרה</u>** כיצד תוצאות המדידה מסעיף ד' מתיישבות עם סיבוכיות זמני הריצה מסעיפים א3, ג3 ומהתרגול (של max1 ו- max2). התייחסו לקצב הגידול של זמני הריצה כתלות בגודל הקלט.

שאלה 2

חמדני הוא בעל דירה בשטח n מ״ר בתל אביב, והוא מעוניין לחלק את דירתו כך שתניב לו רווח מירבי. חמדני חישב את המחיר בו יוכל למכור כל חלק של הדירה בהתאם לגודלה במ״ר. כלומר לכל גודל דירה $i = 1,2,\dots,n$ במ״ר יש מחיר מסויים (כל תת-דירה חייבת להיות בשטח מ״ר שלם, למשל לא ניתן לחלק ל- 1.5 מ״ר).

לדוגמה, להלן כל הדרכים לחלק דירה בגודל 4 מ״ר לדירות בגדלים 1, 2, 3, 4 עבור המחירים [1, 5 ,8 , 9] בהתאמה:

- מחיר כל הדירה בשטח 4 מ״ר: 9
- מחיר הדירה לאחר חלוקה לשטח 1,1,1,1 מ״ר: 4
- מחיר הדירה לאחר חלוקה לשטח 1,1,2 מ״ר: 7
- מחיר הדירה לאחר חלוקה לשטח 2,2 מ״ר: 10
- מחיר הדירה לאחר חלוקה לשטח 1,3 מ״ר: 9

א. עזרו לחמדני למכור את דירתו במחיר המירבי, והשלימו בקובץ השלד את הפונקציה הרקורסיבית `profit`, אשר מחשבת את המחיר המירבי עבור דירה בשטח $size$ ורשימת מחירים $value$. הערך $value[i]$ עבור $0 \le i < size$ הוא המחיר של דירה בשטח $i + 1$. הניחו כי הקלט תקין, ובפרט כי אורך הרשימה $value$ אינו קטן מ- $size$, וכן $size \ge 1$.

למשל:

```
>>> profit([1, 5, 8, 9], 4)
10
>>> profit([2, 3, 7, 8, 9], 5)
11
```

<u>הנחיה מחייבת</u>: יש להשתמש בלולאת for (יחידה) בתוך הפונקציה, כמתואר בתבנית שלהלן. הקריאות הרקורסיביות יתבצעו אך ורק בתוך הלולאה. בפרט, משתמע מכך שמספר הקריאות הרקורסיביות אינו קבוע בשלבים שונים של הרקורסיה (זאת בניגוד, למשל, לאלגוריתם כמו מיון מיזוג שראינו, בו מספר הקריאות הוא קבוע ושווה תמיד 2).

```
def profit(value, size):
    ...
    for i in range(size):
        ...
    return ...
```

ב. שפרו את הפונקציה מהסעיף הקודם כך שתשתמש בממואיזציה. השתמשו במנגנון של פונקציית מעטפת, ששמה הוא `profit2`, שקוראת לפונקציה הרקורסיבית `profit2_mem`, כאשר האחרונה מקבלת פרמטר נוסף d מסוג מילון:

```
def profit2(value, size):
def profit2_mem(value, size, d):
```

<u>הנחיה מחייבת</u>: הבדיקה אם חישוב שנעשה נרשם כבר במילון הממואיזציה תעשה אך ורק לפני הכניסה ללולאת ה- for:

```
def profit2_mem(value, size, d):
    ...
    #check dictionary d here to avoid the next loop if possible
    ...
    for i in range(size):
        ...
    return ...
```

(שימו לב כי באופן כללי, אפשר היה לבדוק את המילון גם בכל איטרציה של הלולאה, כדי למנוע גם חלק מהקריאות הרקורסיביות הבאות.)

ג. ציירו את עץ הרקורסיה של הפונקציות `profit` ו- `profit2` עבור $size = 5$ כאשר בכל צומת רישמו את ערכו של size בקריאה המיוצגת ע"י הצומת (שימו לב: אין צורך לרשום את value בעץ, רק את size).

שאלה 3

בשאלה זו נממש אלגוריתמים להשוואה בין מחרוזות.

לפתרון שלכם אסור להשתמש באופרטור ההשוואה בין מחרוזות של פייתון, אבל מותר להשוות תווים בודדים מתוך מחרוזת.

לדוגמה, מותר:

```
s1[i]==s2[i+1], s1[0]==s2[-1]
```

אסור:

```
s1==s2, s1[2:4]==s2[1:3], s1==""
```

כאשר המחרוזות מכילות רק אותיות קטנות (lowercase) מהא"ב האנגלי a-z , ההשוואה תיתן תוצאה זהה להשוואה של פייתון.

א. השלימו את הפונקציה הרקורסיבית comp (s1, s2) להשוואה בין שתי מחרוזות s1 ו s2. הפונקציה מחזירה True אם ורק אם s1==s2. קיראו את ההנחיות בסוף השאלה שמסבירות כיצד להשלים את הקוד בקובץ השלד.

דוגמאות הרצה:

```
>>> comp('ab','ab')
True
>>> comp('','')
True
>>> comp('a','ab')
False
```

כעת נממש גרסה מורחבת ל comp שבה נרשה למחרוזת s1 להכיל אחד מבין שני תווים מיוחדים נוספים (מלבד האותיות a-z).

1. s1 יכולה להכיל את התו '+' (0 או יותר פעמים), והוא מפורש כתו מיוחד, שמתאים לכל תו יחיד ב s2.

דוגמאות הרצה של הפונקציה המורחבת:

```
>>> comp_ext('+','a')
True
>>> comp_ext('+','')
False
>>> comp_ext('a+b','axb')
True
>>> comp_ext('a+b','axxb')
False
```

2. s1 יכולה להכיל את התו '*' (0 או יותר פעמים), והוא מפורש כתו מיוחד שמתאים לכל רצף של תווים באורך 1 או יותר ב s2.

דוגמאות הרצה נוספות של הפונקציה המורחבת:

```
>>> comp_ext("a*xyz","abcxyz")
True
>>> comp_ext("a*","a")
False
```

הנחיות:

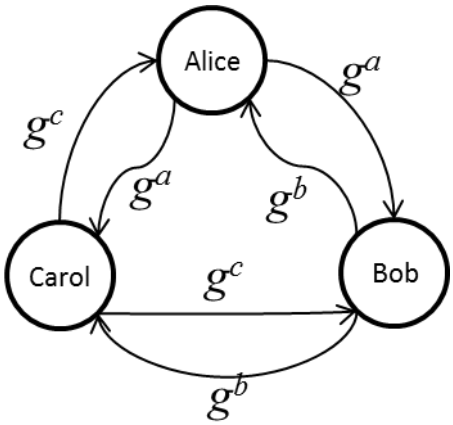
1. ניתן להניח כי s1 מכילה לכל היותר אחד מבין שני התווים המיוחדים, '+' או '*', אך לא את שניהם. s2 מכילה רק את האותיות a-z.
2. שתי הפונקציות comp, comp_ext חייבות להיות רקורסיביות.
3. חלק מהקוד של הפונקציות comp, comp_ext כבר נתון לכם בקובץ השלד, ואתם נדרשים להשלים בתוכו כמה ביטויים: בכל מקום בו מופיעה המחרוזת "replace string with code", החליפו אותה בקוד המתאים.
שימו לב: אין לשנות את מבנה הקוד, להוסיף בו שורות נוספות, או לשמות את הקוד הקיים. לאחר שתחליפו את כל המחרוזות קריאה לפונקציה comp או comp_ext צריכה לפעול כמתואר לעיל. כרגיל, ודאו כי שתי הפונקציות צולחות את הבדיקות בסוף הקובץ.

שאלה 4

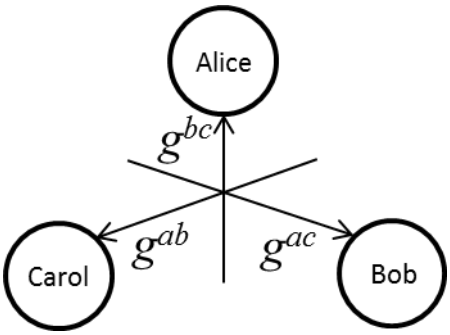
שאלה זו דנה בהרחבת פרוטוקול דיפי-הלמן להחלפת מפתח סודי, ליותר משני משתתפים. נסמן את מספר המשתתפים המעוניינים לייצר להם סוד משותף ב- N.

נזכיר, כי כל החישובים נעשים מודולו p ראשוני כלשהו שנבחר מראש. לשם פשטות, בשאלה זו לא נציין זאת בביטויים שבהמשך. כלומר, בכל מקום שכתוב למשל g^a הכוונה היא ל- $g^a \bmod p$.

א. נניח N=3. להלן תיאור פרוטוקול מורחב אפשרי. בשלב הראשון מתבצע הפרוטוקול כפי שלמדנו בכיתה, בין כל זוג משתמשים. לדוגמה, Alice מחשבת את g^a (פעם אחת), ושולחת זאת ל- Bob ול- Carol. תרשים הודעות שנשלחות:



כעת Alice ו- Bob חולקים את הסוד g^{ab}, Alice ו- Carol את הסוד g^{ac}, ואילו Bob ו- Carol את הסוד g^{bc}. כל זוג שולח את הסוד המשותף שלו למשתמש השלישי. למשל, Alice ו- Bob שולחים (אחד מהם או שניהם, לא משנה) ל- Carol את g^{ab}, ובדומה גם שאר הזוגות:



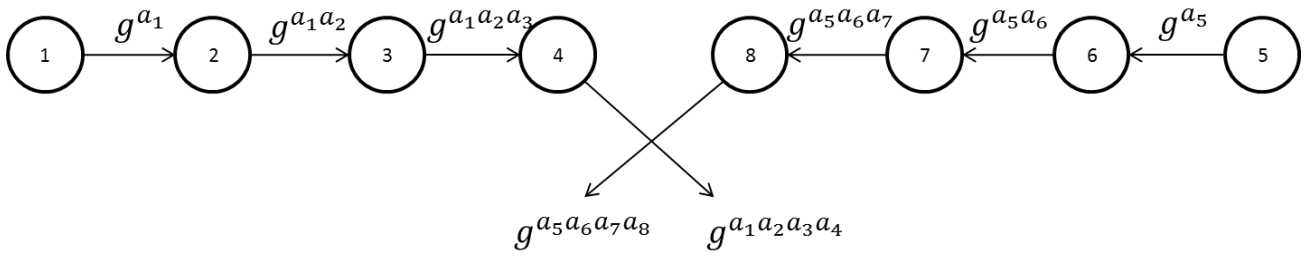
וכל משתמש יכול כעת לחשב את הסוד המשותף g^{abc}.

כמה פעולות modular exponentiation מבצע כל משתמש?

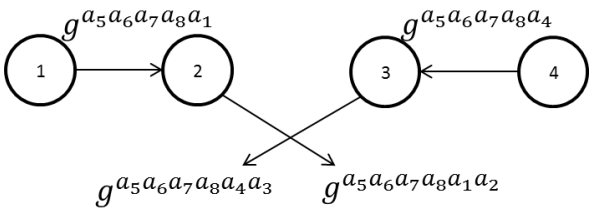
ב. בקובץ ה pdf תארו פרוטוקול דומה, בו כל משתמש מבצע **3** פעולות modular exponentiation בלבד. ציינו אילו הודעות נשלחות בכל שלב. אפשר להיעזר באיור בדומה לאיורים לעיל.

ג. להלן תיאור של פרוטוקול מורחב עבור N=8 משתתפים. נסמן את הסוד הפרטי של משתתף i ב-a_i .

<u>שלב 1</u> : מחלקים את המשתמשים ל- 2 קבוצות שוות, ושולחים את ההודעות הבאות :



שלב 2 : מחלקים כל קבוצה באופן דומה וחוזרים על התהליך, כאשר בכל קבוצה ההודעה ההתחלתית היא ההודעה שנשלחה בסוף השלב הקודם מהקבוצה השנייה. למשל, משתתפים 1, 2, 3 ו- 4 מחולקים שוב וחוזרים על התהליך, עם ההודעה ההתחלתית $g^{a_5a_6a_7a_8}$:



בקובץ ה pdf ענו על ארבע השאלות הבאות :

1) איזו הודעה ישלח משתתף 1 ל- 2 בשלב השלישי (והאחרון)?

2) איזו פעולה יעשה משתתף 2 לאחר השלב השלישי, כדי לחשב את הסוד המשותף?

3) מהו הסוד המשותף לכל 8 המשתתפים?

4) עבור N משתתפים, כמה פעולות modular exponentiation מבצע כל משתתף? יש לתת תשובה במונחים של O, הדוקה ככל שניתן.

שאלה 5

בשאלה זו עליכם לכתוב את הפונקציה הרקורסיבית choose_sets(lst, k).

הפונקציה מקבלת רשימה של איברים lst ומספר שלם k, ומחזירה רשימה המכילה את כל הרשימות השונות באורך k שניתן ליצור מאיברי lst, ללא חשיבות לסדר האיברים. כלומר, את כל האפשרויות לבחור k איברים מתוך הרשימה lst, ללא חשיבות לסדר הבחירה. ניתן להניח שאיברי הרשימה lst יחודיים, כלומר, שאין איברים שחוזרים על עצמם.

שימו לב:

- כאן אנו מעוניינים לייצר ממש את כל האפשרויות השונות לבחור k איברים, ולא רק למצוא כמה אפשרויות כאלו יש.
- הערך המוחזר הוא רשימה של רשימות, וכל אחת מהרשימות הללו הינה בדיוק באורך k.
- סדר הרשימות בתוך רשימת העל אינו חשוב.
- כאמור, הסדר הפנימי בכל תת-רשימה אינו חשוב, ואסור שיהיו כפילויות. לדוגמא, הרשימה [1,2,3] שקולה לרשימה [3,2,1].

הנחיה:

- ניתן לקבל את כל תת-הרשימות באורך k ע"י איסוף של כל תת-הרשימות שמכילות את האיבר הראשון ברשימה וכל תת-הרשימות שאינן מכילות את האיבר הראשון ברשימה.
- ניתן להניח כי הקלט תקין – אין חזרות של איברים ברשימת הקלט ו-$0 \le k \le n$ הוא מספר שלם, כאשר n הוא אורך הרשימה lst.
- 5 השורות הראשונות של הפונקציה choose_sets נתונות לכם בקובץ השלד <u>ואין לשנות אותן</u>.
- אין להשתמש בחבילות חיצוניות של פייתון בפתרון.

דוגמאות הרצה (המשך בעמוד הבא):

```
>>> choose_sets([1,2,3,4], 0)
[[]]
>>> choose_sets([1,2,3,4], 2)
[[4 ,3] ,[2 ,4] ,[2 ,3] ,[1 ,4] ,[1 ,3] ,[1 ,2]]
>>> choose_sets([1,2,3,4], 4)
[[4 ,3 ,2 ,1]]
>>> choose_sets(['a','b','c','d','e'], 4)
[['d', 'c', 'b', 'a'], ['e', 'c', 'b', 'a'], ['d', 'e', 'b', 'a'],
['c', 'd', 'e', 'a'], ['b', 'c','d', 'e']]
```

הנחיות הגשה:

השלימו את מימוש הפונקציה בקובץ השלד.

שאלה 6

שאלה זו עוסקת בנושא של אקראיות וסימולציה.

בשאלה זו נחקור אסטרטגיות משחק ב"משחק המטבעות", משחק בין שני שחקנים המתנהל באופן הבא:

בתחילת המשחק מאותחלות num_of_piles ערמות (num_of_piles הוא מספר חיובי שלם) להכיל כל אחת מספר חיובי זהה כלשהו של מטבעות זהים.

כל שחקן מושך בתורו מספר חיובי של מטבעות מערמה לא ריקה.

המנצח הוא זה שמושך את המטבעות האחרונים ומותיר את כל הערמות ריקות.

לדוגמה, נניח שהמשחק מאותחל כאשר num_of_piles=3, ומספר המטבעות ההתחלתי בשלוש הערמות הוא 5.

להלן מהלך משחק אפשרי:

סיבוב	ערמה 1	ערמה 2	ערמה 3	מהלך
0	5	5	5	מצב התחלתי
1	2	5	5	שחקן 1 משך 3 מטבעות מערמה 1
2	2	5	0	שחקן 2 משך 5 מטבעות מערמה 3
3	2	1	0	שחקן 1 משך 4 מטבעות מערמה 2
4	1	1	0	שחקן 2 משך 1 מטבעות מערמה 1
5	0	1	0	שחקן 1 משך 1 מטבעות מערמה 1
6	0	0	0	שחקן 2 משך 1 מטבעות מערמה 2 וניצח

א. השלימו בקובץ השלד את הפונקציה naive, שמממשת אסטרטגית משחק נאיבית, לפיה השחקן בוחר באופן רנדומלי ובהסתברות שווה אחת מבין הערמות הלא-ריקות, ומסיר מתוכה באופן רנדומלי ובהסתברות שווה מספר חוקי של מטבעות. לדוגמה, אם המשחק מתנהל עם שלוש ערמות אשר בשלב זה של המשחק מכילות [3,0,5] מטבעות, אז אחת מבין הערמות הראשונה והשלישית תבחרנה בהסתברות 0.5 כל אחד. אם נבחרה הערמה הראשונה, השחקן יסיר ממנה מטבע אחד / שני מטבעות / שלושה מטבעות, כל תרחיש בהסתברות 1/3, ובאופן דומה עבור הערמה השלישית.

הפונקציה מקבלת את הרשימה num_of_coins שמכילה את מספרי המטבעות בערמות (וארכה הוא num_of_piles, מספר הערמות). הפונקציה יוזמת מהלך משחק יחיד ומשנה את הרשימה בהתאם (כלומר מספר המטבעות באחת הערמות צריך לקטון). ניתן להניח כי בעת הקריאה לפונקציה נותרו מטבעות בלפחות אחת מהערמות.

דוגמאות הרצה:

```
>>> num_of_coins = [3,4,5]
>>> naive(num_of_coins)
>>> print(num_of_coins)
[3, 2, 5]
```

```
>>> naive(num_of_coins)
>>> print(num_of_coins)
[3, 2, 4]
>>> num_of_coins = [0,0,1]
>>> naive(num_of_coins)
>>> print(num_of_coins)
[0, 0, 0]
```

ב. השלימו בקובץ השלד את הפונקציה compete, שמקבלת את מספר הערמות num_of_piles, מספר התחלתי של מטבעות בכל ערמה init_num_of_coins, ושתי פונקציות אסטרטגיה s1 ו s2 (פרמטרים מטיפוס function), ומסמלצת משחק בין האסטרטגיות. המשחק יאותחל כך שכל ערמה תכיל את מספר המטבעות ההתחלתי. האסטרטגיה הפותחת תוגרל אף היא בהסתברות שווה, כך שכל אחת מהפונקציות s1 ו s2 תפתח בהסתברות 0.5. הפונקציה קוראת לסירוגין לפונקציות האסטרטגיה עד שאחת מהן מנצחת (כלומר מרוקנת את הערמות). הפונקציה מחזירה 0 אם s1 ניצחה, ו 1 אם s2 ניצחה.

דוגמאות הרצה:

```
>>> compete(3, 10, naive, naive)
1
>>> compete(3, 10, naive, naive)
1
>>> compete(3, 10, naive, naive)
0
```

ג. השלימו בקובץ השלד את הפונקציה compare, שמשווה בין שתי אסטרטגיות משחק. בדומה ל compete, הפונקציה מקבלת את מספר הערמות, את מספר המטבעות ההתחלתי בכל ערמה ושתי פונקציות אסטרטגיה. בנוסף, היא מקבלת את מספר המשחקים num_of_games. הפונקציה קוראת ל compete מספר פעמים ששווה ל- num_of_games עם הפרמטרים המתאימים, וסוכמת את מספר הפעמים שכל אסטרטגיה ניצחה. היא מחזירה tuple שהאבר הראשון שלו מכיל את מספר הנצחונות של s1, והשני את מספר הנצחונות של s2.

ד. בקובץ השלד מופיעה הפונקציה winning, שמממשת אסטרטגיה מוצלחת ש(כמעט תמיד) מנצחת (ראו בהמשך את סעיף הבונוס המתייחס לפרטי אסטרטגיה זו). השתמשו ב compare כדי להשוות בין ביצועי האסטרטגיות הנאיבית והמנצחת ובין כל אחת לבין עצמה. רשמו את התוצאות בקובץ ה pdf ותארו אותן בשניים-שלושה משפטים.

ה. המציאו אסטרטגיה חדשה, גרועה יותר מזו הנאיבית (אך חוקית, כלומר כזו שגורעת מטבעות מערמה לא-ריקה). השלימו בקובץ השלד את הפונקציה silly, שמממשת את האסטרטגיה שהמצאתם . השתמשו ב compare כדי להוכיח שהיא אכן גרועה יותר: רשמו את התוצאות בקובץ ה pdf ותארו אותן בשניים-שלושה משפטים. הוסיפו תיאור קצר של האסטרטגיה שהמצאתם.

ו. <u>שאלת בונוס</u>

האסטרטגיה המנצחת פועלת באופן הבא:

ראשית, היא ממירה את מספר המטבעות בכל ערימה לייצוג בינארי, ומחשבת bitwise XOR (כלומר XOR של כל ביט בנפרד) על הביטים של הייצוגים שהתקבלו. לדוגמה, אם הרשימה שמכילה את מספרי המטבעות בערימות היא [1,5,3,9], הפונקציה תחשב

1 -> 0001

5 -> 0101

3 -> 0011

9 -> 1001

BITWISE XOR ========

1110

הפונקציה תבצע מהלך שיהפוך את תוצאת ה bitwise XOR ל 0. במקרה הזה, אם נחסיר מהערמה שמכילה 9 מטבעות שני מטבעות, נוותר עם שבעה מטבעות, ונקבל את המבוקש:

1 -> 0001

5 -> 0101

3 -> 0011

7 -> 0111

BITWISE XOR ========

0000

אנו טוענים כי תמיד קיים מהלך משחק חוקי שיהפוך את ה bitwise XOR ל 0.

בנוסף, אנו טוענים כי שחקן שנוקט באסטרטגיה הזו ינצח, מלבד במקרה בו ה bitwise XOR הוא כבר 0 כאשר השחקן נוקט במהלך המשחק הראשון שלו (ואז מהלך המשחק תלוי גם באסטרטגיה של השחקן השני).

א. שימו לב כי במקרה שבו יש רק שתי ערמות (num_of_piles=2), האסטרטגיה המנצחת הופכת פשוטה במיוחד. תארו במילים את האסטרטגיה במקרה זה, והוכיחו כי אם שחקן שפועל לפיה מקבל מצב פתיחה שבו בשתי הערימות מספר שונה של מטבעות, הוא ינצח בוודאות.

ב. (בונוס על הבונוס): הסבירו מדוע האסטרטגיה המנצחת נוטה לנצח במקרה הכללי (num_of_piles >=2).

סוף