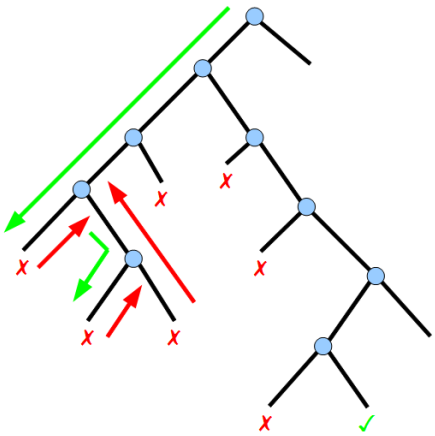
מבוא למדעי המחשב 67101 – סמסטר א' 2021
תרגיל 8 – Backtracking
להגשה בתאריך **17/12/2020** בשעה 22:00



הקדמה והערות חשובות

בתרגיל זה נתרגל שימוש ב **Backtracking**. אנא קראו את הדברים הבאים **לפני תחילת עבודה**:

- בתרגיל זה יש להגיש קובץ אחד בשם ex8.zip, ובתוכו נמצא הקובץ **nonogram.py** והקובץ **AUTHORS.**
- חתימות הפונקציות (שם הפונקציה והפרמטרים שלה) צריכות להיות זהות במדויק לחתימות המתוארת במשימות (היזהרו מ copy – paste מקובץ זה, כתבו בעצמכם).
- לפני מימוש פונקציה, קראו את כל הסעיף, וודאו הבנה של הדוגמאות המצורפות.
- ניתן להוסיף פונקציות נוספות וניתן להשתמש בשאלות מאוחרות יותר בפונקציות שמומשו קודם.
- סגנון: הקפידו על תיעוד נאות ובחרו שמות משתנים משמעותיים. הקפידו להשתמש בקבועים (שמות משתנים באותיות גדולות), על פי ההסברים שנלמדו, רק אם יש בכך צורך.
- בכל שאלה מפורט מה ניתן להניח על הקלט - אין צורך לבצע בדיקות תקינות נוספות מעבר למפורט.
- אין לייבא (לעשות import) לאף אחד מהספריות: numpy, itertools, re לכל ספריה אחרת – בקשו במפורש בפורום.
- בתרגיל זה יש חשיבות ליעילות מימוש הפונקציות. מימוש לא יעיל יגרור הורדת נקודות.
- שימו לב מתי אתם משנים קלט לפונקציה ומתי לא.
- **קראו את כל המסמך לפני תחילת עבודה!**
- **המלצת עבודה: לפני מימוש כל פונקציה, רשמו את אלגוריתם הפתרון שלכם על דף טיוטה.**

הגשה בזוגות

את חלק א'-ג' בתרגיל זה יש להגיש זוגות. שימו לב: עבודה בזוגות היא עבודה משותפת, **ולא חלוקת עבודה**. אנו נבדוק מעורבות כל סטודנט בחלקו בתרגיל. כמוכן אנו מבטיחים שחומר זה הינו קריטי להמשך. החלקים האחרים של התרגיל הינם לעבודה אישית.

מטרת התרגיל

בתרגיל זה נממש פתרון למשחק **שחור ופתור** (או בשמו הלועזי **Nonogram**) ללוח שחור לבן. על מנת לעשות כן, נשתמש ב**גישוש נסוג** (**Backtracking**).

<u>הסבר על המשחק:</u>

כדאי לקרוא על המשחק בויקיפדיה:

שחור ופתור

Nonogram

חלק א' – שפה משותפת

בחלק זה נגדיר את האופן בו אנו מייצגים את המשחק בפייתון. נשים לב שלוח משחק מוגדר ע"י גודלו, ומספר רצפים בסדר מסוים לכל שורה ולכל עמודה.

<u>רשימת אילוצים:</u>

עבור לוח משחק בגודל $n \times m$ (n שורות, m עמודות) נגדיר את **מערך אילוצי השורות** להיות מערך בגודל **n** כאשר בתא ה- **i** ישנו מערך המכיל את כל מספרי הרצפים (לפי הסדר) של הבלוקים המושחרים המופיעים בשורה ה **i**.

באותו האופן בדיוק נגדיר את **מערך אילוצי העמודות**.

את שני המערכים הנ"ל נקבץ למערך יחיד, כאשר **מערך אילוצי השורות** מופיע בתא ה 0, ו**מערך אילוצי העמודות** מופיע בתא ה 1.

דוגמה: את אילוצי לוח המשחק הבא:

					2	2
		0	2	1	2	2
	0					
	4					
	6					
2	2					
1	3					

נייצג ע"י המערך:

my_constraints=[[[], [4], [6], [2, 2], [1, 3]], [[], [2], [1], [2, 2], [2, 2]]]

כאשר בצבע הכתום מסומן **מערך אילוצי השורות** ובצבע הכחול מסומן **מערך אילוצי העמודות**.

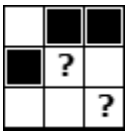
שימו לב שעבור 0 משבצות צבועות בשורה, האילוץ יהיה רשימה ריקה.

<u>מטריצת המשחק:</u>

את המשחק נייצג ע"י מערך דו ממדי: כלומר רשימה של רשימות, כאשר הרשימה ה-0 היא השורה הראשונה, והאיבר ה-0 ברשימה ה-0 הוא האיבר הראשון בעמודה הראשונה. כלומר האיבר ה-0,0 ממוקם שמאל עליון.

נסמן ב-0 משבצת לבנה, ב **1** משבצת מושחרת, וב **-1** משבצת שאינה צבועה (כלומר עדיין לא הוחלט לגביה שחור או לבן)

למשל את הלוח הבא:



נייצג ע"י הרשימה:

my_board=[[0, 1, 1], [1, -1, 0], [0, 0, -1]]

הערה: בין כל זוג "בלוקים" של משבצות מלאות, **חייבת** להופיע לפחות משבצת ריקה אחת. לדוגמה, שורה בת 5 משבצות שחורה לגמרי, תסומן באילוץ [5], ולא ב [3,2] או ב [1,4] וכו'.

חלק ב' – בניית עזרי פתרון

1. כל אפשרויות הצביעה עם אילוצים:

משימה: כתבו פונקציה אשר מקבלת אורך שורה n ורשימת אילוצים blocks, ומחזירה את כל אפשרויות הצביעה של שורה באורך n עם אילוצים blocks.

- חתימת הפונקציה:

def constraint_satisfactions (n, blocks)

- קלט הפונקציה:
  - מספר טבעי n.
  - הקלט blocks הינו מערך מספרים (גדלי הבלוקים בשורה לפי הסדר).

- פלט הפונקציה: רשימה של כל הצביעות האפשריות (רשימה של {רשימות באורך n} עם הערכים 0 או 1) לפי רשימת האילוצים blocks.

דוגמאות לקלטים ופלטים:

constraint_satisfactions(3, [1]) -> [[1, 0, 0], [0, 1, 0] , [0, 0, 1]]

constraint_satisfactions (3, [2]) -> [[1, 1, 0], [0, 1, 1]]

constraint_satisfactions (3, [1, 1]) -> [[1, 0, 1]]

constraint_satisfactions (4, [1, 1]) -> [[1, 0, 1, 0], [0, 1, 0, 1], [1, 0, 0, 1]]

constraint_satisfactions (5, [2, 1]) -> [[1, 1, 0, 1, 0] , [1, 1, 0, 0, 1], [0, 1, 1, 0, 1]]

הערות למימוש:

- ניתן להניח הקלטים תקינים (אם אין צביעה אפשרית שעומדת באילוצים, הקלט נחשב תקין).
- מספר טבעי אצלנו הוא שלם גדול מ 0.
- שימו לב **לא לשנות את רשימת הקלט.**
- לא חשוב סדר האיברים ברשימה המוחזרת.
- ממשו פונקציה זו בעזרת Backtracking.

2. בדיקת אפשרויות הצביעה:

משימה: כתבו פונקציה אשר מקבלת שורה נתונה בלוח ורשימת אילוצים (רשימת גדלי בלוקים) ומחזירה רשימה של כל האפשרויות להשלמת צביעת השורה, **כך שהצביעה תעמוד באילוצים**.

- חתימת הפונקציה:

```
def row_variations(row, blocks)
```

- קלט הפונקציה:
  - הקלט blocks הינו מערך מספרים (גדלי הבלוקים בשורה לפי הסדר).
  - הקלט row הינו רשימה המייצגת צביעה חלקית של שורה (רשימה עם הערכים 1-, 0, 1)
- פלט הפונקציה: רשימה של כל הצביעות האפשריות (רשימת רשימות עם הערכים 0 או 1)

דוגמאות לקלטים ופלטים:

```
row_variations([1, 1, -1, 0], [3]) -> [[1, 1, 1, 0]]
row_variations([-1, -1, -1, 0], [2]) -> [[0, 1, 1, 0], [1, 1, 0, 0]]
row_variations([-1, 0, 1, 0, -1, 0], [1,1]) -> [[0, 0, 1, 0, 1, 0], [1, 0, 1, 0, 0, 0]]
row_variations([-1, -1, -1] , [1]) -> [[1, 0, 0], [0, 1, 0], [0, 0, 1]  ]
row_variations([0, 0, 0] , [1]) -> [ ]
row_variations([0, 0, -1, 1, 0] , [3]) -> [ ]
row_variations([0, 0, -1, 1, 0] , [2]) -> [ [0, 0, 1, 1, 0]]
row_variations([0, 0, 1, 1, 0] , [2]) -> [ [0, 0, 1, 1, 0]]
```

הערות למימוש:

- ניתן להניח שהקלטים תקינים. שימו לב שיתכן ואין צביעות העומדות באילוצים.
- הפונקציה צריכה להשחיר רק משבצות שלא ידועות זהותן, כלומר עם הערך 1-. משבצות עם הערך 0 לא נשחיר. משבצות עם הערך 1, כלומר כבר מושחרות – בהן נצטרך להתחשב כחלק מבלוק כלשהו.
- שימו לב **לא לשנות את רשימת הקלט.**
- לא חשוב סדר האיברים ברשימה המוחזרת.
- **חישבו:** אילו צביעות ניתן לפסול מראש? אילו תנאים צריכה לקיים צביעה חוקית? אילו תנאים מקיימת צביעה לא חוקית? כדי לקבל אינטואיציה, נסו לכתוב על דף את כל אפשרויות הצביעה של הדוגמה הרביעית.
- ממשו פונקציה זו בעזרת Backtracking. מימוש לא יעיל עלול לאבד ניקוד בשאלה (יעיל יותר -> פחות פעולות ובדיקות מיותרות). שימו לב ששימוש בפונקציה הקודמת (קריאה אליה) הינה מימוש לא יעיל מבחינה חישובית. קראו שוב נקודה אחת למעלה.
- לפני מימוש הסעיף הנכחי, נמליץ רק לקרוא את הסעיף הבא.

3. השחרת/הלבנת משבצות משותפות:

נרצה לכתוב פונקציה שמקבלת אילוצים של כמה שורות, ומחזירה את האילוץ המשותף לכולן.

משימה: כתבו פונקציה אשר מקבלת רשימה של שורות ומחזירה "חיתוך השורות" – שורה בה ישנן משבצות שחורות, לבנות וניטרליות – כאשר משבצת שחורה/לבנה היא משבצת שכל שורות הקלט בה מסכימות על צביעה בשחור/לבן (בהתאמה), וניטרלית היא משבצת בה אין דפוס חד משמעי. במקרה שבו כל המשבצות הצבועות בשורות הקלט צבועות בשחור/לבן, אך המשבצת היא ניטרלית בחלק מהשורות, ניתן לבחור (באופן עקבי) אם משבצות כאלה יצבעו בשחור/לבן (בהתאמה) או שישארו ניטרליות. הצדיקו את הבחירה שלכם בהערה בראש הקובץ.

- חתימת הפונקציה:

```
def intersection_row(rows)
```

- קלט הפונקציה: רשימה של שורות באותו אורך.
- פלט הפונקציה: שורה בה צבועות בשחור/לבן רק משבצות שצבועות כך בכל אחת משורות הקלט. השאר ניטרליות.

דוגמה לקלט ופלט:

```
intersection_row([[0, 0, 1], [0, 1, 1], [0, 0, 1]]) -> [0, -1, 1]
intersection_row([[0, 1, -1], [-1, -1, -1]]) -> [0, 1, -1] or [-1, -1, -1]
```

הערות למימוש:

- אין לשנות את הקלט.
- ניתן להניח שהקלט תקין (רשימה לא ריקה, שכל הרשימות בתוכה הן באותו האורך).
- משבצת שחורה מסומנת ב-1, משבצת לבנה מסומנת ב 0, ומשבצת ניטרלית תסומן ב- 1-.

חלק ג' – פתרון למשחק

4. פתרון משחק Nonogram פשוט:

נכתוב פונקציה אשר מחקה שיטת פתרון אנושית לשחור ופתור: הפונקציה תבדוק אילו משבצות חייבות להיות שחורות/לבנות בכל שורה (לפי חפיפת כל האפשרויות), ותצבע אותם בהתאם. לדוגמה, עבור הלוח הבא:

0	2	1

אפילו מבלי לדעת את האילוצים על השורות, אנו יכולים להסיק כי המשבצת האמצעית תהיה שחורה – זה נובע מחפיפת כל האפשרויות לצביעת העמודה האמצעית.

עבור לוחות משחק מסוימים ויחסית פשוטים, בעזרת השיטה הנ"ל ניתן לפתור את המשחק.
כלומר ע"י הסקת מסקנות מהאילוצים בלבד, נוכל לצבוע משבצות בכל שורה, ולאחר מכן עם המידע החדש לעשות את אותה הפעולה על העמודות. עם המידע החדש, נחזור חלילה על הנ"ל עד שנתכנס ללוח, בתקווה, פתור.

משימה: כתבו פונקציה המקבלת אילוצי לוח משחק Nonogram (כמתואר בחלק א') ומחזירה את המשחק הפתור.

- חתימת הפונקציה:

```
def solve_easy_nonogram(constraints)
```

- פלט הפונקציה: הפונקציה תחזיר לוח משחק פתור ככל שניתן, או None אם הגעתם לסתירה במהלך הפתרון..

הערות למימוש:

- הקלט constraints הינו מערך מהסוג המתואר בחלק א', וניתן להניח כי הינו תקין. לא ניתן להניח שיש פתרון.
- אם יתקבל כקלט לוח משחק שלא ניתן לפתור לגמרי על ידי השיטות הנ"ל, הפונקציה תחזיר את הלוח המתקבל שממנו לא ניתן להסיק יותר מסקנות ישירות מחפיפת אילוצים – וישארו בו משבצות ניטרליות במקומות המתאימים. כלומר הפונקציה תחזיר לוח פתור חלקית – עד כמה שניתן.
- שימו לב שיש להסיק רק את מה שניתן להסיק על ידי בדיקת כל שורה או עמודה בפני עצמן.
- שימו לב שברגע שהסקנו מסקנה על שורה מסוימת (כלומר צבענו בה משבצות לפי האילוצים) אז נוסף מידע חדש על עמודה / עמודות שבהן צבענו את השורה, ועליהן צריך לבצע הסקת מסקנות נוספת.

- על הפונקציה לבצע הסקת מסקנות על השורות ועל העמודות, **ולחזור על התהליך** עד אשר אין מסקנות שניתן להסיק (לאחר סבב אחד על שורות ועמודות, עשוי להיווצר מידע חדש שניתן להשתמש בו להסקה נוספת).
- ממשו את הפונקציה בצורה יעילה (על אילו עמודות \ שורות מיותר לעבור, בכל איטרציה?)
- ניתן ורצוי להשתמש בפונקציית עזר משלכם. בנוסף, **השתמשו בפונקציות מסעיפים קודמים**.

5. פתרון מלא למשחק Nonogram:

בעזרת סעיפים קודמים, נוכל לממש פונקציה הפותרת כל משחק Nonogram בצורה יחסית מהירה (מבחינה פרקטית).

משימה: כתבו פונקציה המקבלת אילוצי לוח משחק Nonogram (כמתואר בחלק א') ומחזירה רשימה של פתרונות למשחק.

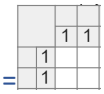
- חתימת הפונקציה:

```
def solve_nonogram(constraints)
```

- פלט הפונקציה: הפונקציה תחזיר רשימה של לוחות פתורים (כל הפתרונות האפשריים למשחק).

דוגמה לקלט ופלט:

עבור הלוח הבא:

[[[1] ,[1]] ,[[1] ,[1]]] = 

נקבל:

solve_nonogram([[[1], [1]], [[1], [1]]]) = [[[1, 0], [0, 1]], [[0, 1], [1, 0]]]= [ , ]

הערות למימוש:

- הקלט constraints הינו מערך מהסוג המתואר בחלק א', וניתן להניח כי הינו תקין.
- אם אין לוחות שמתאימים לאילוצים, יש להחזיר רשימה ריקה.
- ניתן ורצוי להשתמש בפונקציית עזר משלכם.
- **השתמשו בפונקציות מסעיפים קודמים**, וממשו הפתרון בעזרת Backtracking.
- על הפונקציה להיות יעילה. נקצה זמן מסוים לפתרון לוחות ספציפיים בעזרת מימושכם – מימוש לא יעיל מספיק לא ינוקד באופן מלא (או בכלל).
- יש לוחות משחק שהפונקציה בסעיף הקודם אינה יכולה לפתור, ופונקציה זו כן.

החלקים הבאים הינם לעבודה אישית ולא בזוגות:

חלק ד' – שאלות תאורטיות

6. מענה על שאלות ב Ex8 - Quiz:

ענו על Ex8 – Quiz הנמצא במודל תחת סימניה Exercise 8. **יינתן עליו ציון כחלק מהתרגיל**.

חלק ה' – Code Review

7. ביקורת עמיתים / Peer Review:

השבוע כל אחד מכם יקבל שני פתרונות של חבריכם לתרגיל 6 ותתבקשו לעשות להם code review (סקר קוד). סקר קוד הוא הליך מקובל מאד בחברות תוכנה בו מתכנת אחד קורא קוד של מתכנת אחר ומחווה את דעתו עליו. לסקר קוד מטרות רבות, ביניהן מציאת שגיאות, שיפור הקוד, תכנון קוד נכון יותר, למידה מניסיונם של מתכנתים אחרים ועוד.

במודל השבוע יהיה לינק ל-Code Review ובו תקבלו גישה לשני פתרונות לתרגיל 6. עליכם לקרוא את הפתרון, להבין אותו ולהעביר עליו ביקורת. בביקורת יש להתייחס להיבטים הבאים:

- תכנון הקוד וחלוקתו לפונקציות.
- קריאות הקוד - האם היו חלקים שהיו קשים להבנה?
- האם הקוד מודולרי וקל לשינוי? למשל, אילו היינו רוצים להוסיף אופציות לחלק מהתפריטים כמה שינויים היו נגררים בקוד.
- שגיאות מפורשות בפתרון התרגיל.

חשוב לציין שלא מספיק לכתוב מה נעשה לא נכון, אלא צריך גם לכתוב איך היה ניתן לפתור את הבעיה. לכל הערה יש לכתוב את מספר השורה ושם הפונקציה הרלוונטיים. כמו כן, כתבו על דברים שנעשו טוב בפתרון התרגיל ועל דברים שאתם למדתם מקריאתו. זאת אומרת שבכל מקרה יש לעבור על כל תרגיל ולהעיר הערות מפורטות על אילו חלקים היה צריך לשנות ואיך, ועל אילו חלקים נעשו היטב לדעתכם ולמה.

סעיף זה הינו חובה, וחלק מהתרגיל.

שימו לב: התרגיל אותו תבדקו ושעליו תתנו את דעתכם, לא יאבד ניקוד כתוצאה מהביקורת שלכם. לא תפגעו בציון חבריכם!

נהלי הגשה

בתרגיל זה, יש להגיש קובץ zip הנקרא ex8.zip המכיל את הקבצים הבאים:

- הקובץ nonogram.py בו הפונקציות שמימשתם.
- קובץ AUTHORS – כמו שהגשתם בעבר: קובץ ללא סיומת בשם AUTHORS, בו רשומים משתמשי ה CSE של המגישים (שמות משתמשים איתם אתם מתחברים למודל), מופרדים בפסיקים וללא רווחים.
  a. לדוגמה: user_1,user_2.
  b. אם תגישו את התרגיל לבד, יש לכתוב רק את שם המשתמש שלכם ללא פסיק.

שימו לב ש Ex8 – Quiz הינו חובה וחלק מציון התרגיל.

שימו לב ש Code Review הינו חובה וחלק מציון התרגיל.

שימו לב שאי הגשת התרגיל לפי הדרישות לעיל, עלולה לגרום הורדה בציון (ללא ערעורים). אנא עמדו בדרישות.

בהצלחה!

התנדבות לקהילה (לא חובה וללא ניקוד):

אם תצליחו לפתור את הלוח הבא, שתפו אותנו! העלו תמונה שלו לפורום !

(אם יש כמה פתרונות, העלו את הפתרון שלדעתכם התכוונו אליו 😊)

rows:16 / cols:18	1 8 1	1 2	1 6 1 4	1 1 2	1 4 1 1 1	1 1 1 1	1 2 1 1 1 2	1 1 1 1 3	16	16	1 1 1 1 3	1 2 1 1 1 2	1 1 1 1	1 4 1 1 1	1 1 2	1 6 1 4	1 2	1 8 1
1 1 1 1 2 1 1 1 1																		
2																		
1 1 1 1 2 1 1 1 1																		
1 1 1 6 1 1 1																		
1 1 1 2 1 1 1																		
1 1 10 1 1																		
1 1 2 1 1																		
1 14 1																		
1 2 1																		
18																		
2																		
1 2																		
5 2 1																		
3 4 5																		
1 6 3																		
8 1																		