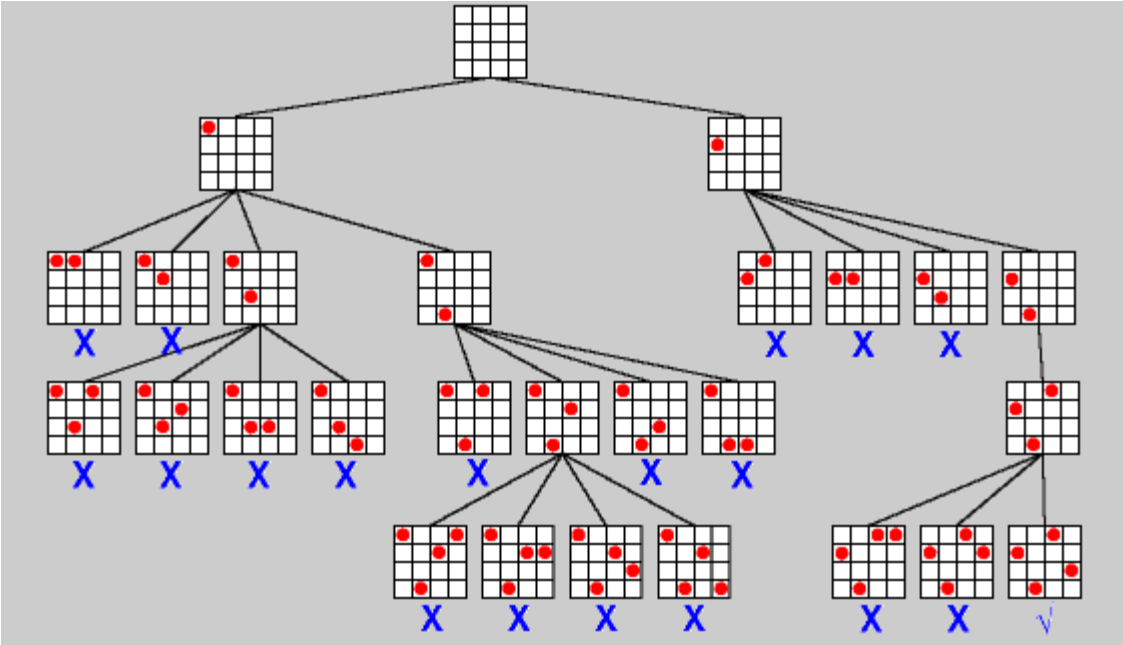
בית הספר להנדסה ומדעי המחשב ע״ש רחל וסלים בנין

קורס מבוא למדעי המחשב 67101

תרגיל 8 – Backtracking

להגשה עד *יום רביעי, 25 לדצמבר 2019, בשעה 22:00*.



בית הספר להנדסה ומדעי המחשב ע״ש רחל וסלים בנין

הקדמה

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

- בתרגיל זה יש להגיש קובץ אחד בשם ex8.zip, ובתוכו נמצא הקובץ **nonogram.py**.
- 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, ורק אם יש בכך צורך.
- בכל שאלה מפורט מה ניתן להניח על הקלט - אין צורך לבצע בדיקות תקינות נוספות מעבר למפורט.
- בתרגיל זה מומלץ לייבא ולהעזר בקובץ העזר **ex8_helper.py**. אין לייבא (לעשות import) לאף אחד מהספריות: numpy, itertools, re לכל ספריה אחרת – בקשו במפורש בפורום.
- אין להגיש קוד המשתמש בפונקציות מ ex8_helper.py – תצטרכו לכתוב פונקציות משלכם.
- 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.*

בית הספר להנדסה ומדעי המחשב ע״ש רחל וסלים בנין

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**).

הסבר על המשחק:

כדאי לקרוא על המשחק בויקיפדיה:

שחור ופתור

Nonogram

חלק א׳ – שפה משותפת ופונקציות עזר

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

רשימת אילוצים:

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 השורות.

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**.

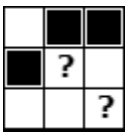
שימו לב שעבור 0 משבצות צבועות בשורה, האילוץ יהיה רשימה ריקה.

מטריצת המשחק:

את המשחק נייצג ע״י מערך דו מימדי: 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 שהיא בסימן שאלה (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]]

בית הספר להנדסה ומדעי המחשב ע״ש רחל וסלים בנין

*הערה: בין כל זוג "בלוקים" של אילוצים, **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'.*

לנוחיותכם, מצורף קובץ ex8_helper.py (**שאינו להגשה!**) עם הפונקציות הבאות:

פונקצית עזר נתונה #1 – print_board

כתבנו עבורכם פונקציה המקבלת לוח משחק (מערך דו מימדי) עם הערכים האפשריים 1, 0, -1, ומדפיסה ייצוג טקסטואלי שלה. תוכלו להשתמש בה על מנת לדבג את הקוד.

- חתימת הפונקציה: def print_board(board)
- ההדפסה למסך, לפי הפורמט הבא:
  - 0 (משבצת ריקה) יסומן ע״י _ (קו תחתון).
  - 1 (משבצת מושחרת) יסומן ע״י **X** (תו איקס באות גדולה, אחד).
  - -1 (לא ידוע) יסומן ע״י **?** (סימן שאלה)

דוגמה: מטריצה המיוצגת ע״י המערך הבא (הורדנו פסיקים לנוחות הצגה):

```
[[-1  1  1  1 -1]
 [-1  1  1 -1  0]
 [-1  1  0 -1 -1]
 [ 1  1  0  0  0]
 [-1  0  0 -1 -1]]
```

תוחזר באופן הבא:

```
? X X X ?
? X X ? _
? X _ ? ?
X X _ _ _
? _ _ ? ?
```

פונקצית עזר נתונה #2 – get_line_constraints

כתבנו עבורכם פונקציה המקבלת רשימה המייצגת שורה עם משבצות לבנות או שחורות, ומחזירה את מערך האילוצים שלה.

- חתימת הפונקציה: def get_line_constraints(lst)

דוגמה לשימוש בפונקציה:

get_line_constraints([0, 0, 1,1,1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1]) → [3, 1, 1]

חלק ב' – בניית עזרי פתרון

1. בדיקת אפשרויות הצביעה:

משימה: כתבו פונקציה אשר מקבלת שורה נתונה בלוח ואת האילוצים שלה (רשימת גדלי בלוקים) ומחזירה רשימה של כל האפשרויות לצביעת השורה, כך שהצביעה תעמוד באילוצים.

- חתימת הפונקציה: def get_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:

- get_row_variations([1, 1, -1, 0], [3]) → [[1, 1, 1, 0]]
- get_row_variations([-1, -1, -1, 0], [2]) → [[0, 1, 1, 0], [1, 1, 0, 0]]
- get_row_variations([-1, 0, 1, 0, -1, 0], [1,1]) → [[0, 0, 1, 0, 1, 0], [1, 0, 1, 0, 0, 0]]
- get_row_variations([-1, -1, -1] , [1]) → [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
- get_row_variations([0, 0, 0] , [1]) → [ ]
- get_row_variations([0, 0, -1, 1, 0] , [3]) → [ ]
- get_row_variations([0, 0, -1, 1, 0] , [2]) → [[0, 0, 1, 1, 0]]
- get_row_variations([0, 0, 1, 1, 0] , [2]) → [[0, 0, 1, 1, 0]]

הערות למימוש:

- ניתן להניח שהקלטים תקינים, ושמערך האילוצים אינו סותר את תוכן השורה ולהיפך.
- 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).

2. 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 צבועות בשחור / לבן (בהתאמה), וניטרלית היא משבצת בה אין דפוס חד משמעי.

- חתימת הפונקציה: def get_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:

get_intersection_row([[0, 0, 1], [0, 1, 1], [0, 0, 1]]) → [0, -1, 1]

get_intersection_row([[0, 0, 1], [0, 1, 1], [0, 0, -1]]) → [0, -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.

3. בדיקת הפונקציות:

בדקו תקינותן של הפונקציות הנ״ל שרשמתם. לא מומלץ להמשיך הלאה לפני שהן עובדות כנדרש.

4. הסקת מסקנות מאילוצים:

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.

משימה: כתבו פונקציה אשר מקבלת כקלט לוח משחק נכחי ומערך אילוצים (בהתאם לחלק א׳). הפונקציה תעבור על כל שורה בלוח ותצבע משבצות בשחור / לבן, בהתאם לצבע שהן חייבות להיצבע בו, לפי האילוצים וחפיפת האפשרויות. אם אין צבע חד משמעי, המשבצת תיצבע בלא ידוע (-1).

- חתימת הפונקציה: def conclude_from_constraints(board, constraints)
- קלט הפונקציה: לוח משחק נתון (עם הערכים 1 ,0 ,-1).
- פלט הפונקציה: הערך **None**.

הערות למימוש:

- שימו לב שברגע שהסקנו מסקנה על שורה מסויימת (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).
- ניתן להניח כי הקלט תקין (לוח ואילוצים לפי המוסכמה בסעיף א׳, ומותאם ללוח החלקי המתקבל).
- שימו לב שהפונקציה מחזירה את הערך **None**, ולכן עובדת על הקלט – כלומר משנה אותו.

בית הספר להנדסה ומדעי המחשב ע״ש רחל וסלים בנין

חלק ג׳ – פתרון למשחק

5. פתרון משחק Nonogram פשוט:

עבור לוחות משחק מסויימים ויחסית פשוטים, בעזרת הפונקציות 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.
במימוש זה, נניח כי הקלטים הם מהסוג שניתן לחזות ע״י השיטות הנ״ל.

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 את הפתרון למשחק (לוח מלא), לפי החוקים, בפורמט חלק א (בהנחה שהלוח ניתן לפתרון המתאים לתאור בסעיף 6 הנכחי).

הערות למימוש:

- הקלט constraints הינו מערך מהסוג המתואר בחלק א׳, וניתן להניח כי הינו תקין.
- 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 – ויופיעו בו סימני שאלה במקומות המתאימים. כלומר הפונקציה תחזיר לוח פתור חלקית – עד כמה שניתן.
- ממשו את הפונקציה בצורה יעילה (על אילו עמודות \ שורות מיותר לעבור, בכל איטרציה?)
- ניתן ורצוי להשתמש בפונקציית עזר משלכם.
- **השתמשו בפונקציות מסעיפים קודמים.**

6. פתרון מלא למשחק Nonogram (שווי סעיף: 3 נ׳ק):

שווי סעיף זה הוא 3 נ׳ק (בלעדיו הציון המקסימלי לתרגיל הוא 97). **אין קשר** בין שווי הסעיף לרמת הקושי שלו – שיקולנו הם דידקטיים בלבד.
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]]] = [

, 

]

בית הספר להנדסה ומדעי המחשב ע״ש רחל וסלים בנין

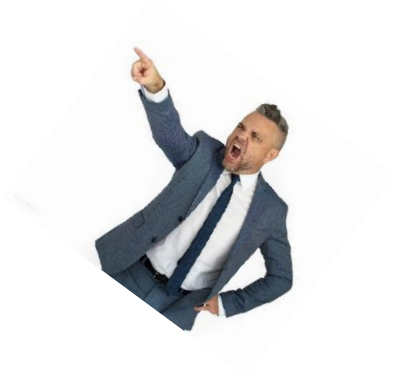
הערות למימוש:

- הקלט constraints הינו מערך מהסוג המתואר בחלק א׳, וניתן להניח כי הינו תקין.
- ניתן ורצוי להשתמש בפונקציית עזר משלכם.
- **השתמשו בפונקציות מסעיפים קודמים**, וממשו הפתרון בעזרת Backtracking.
- על הפונקציה להיות יעילה. נקצה זמן מסויים לפתרון לוחות ספציפיים בעזרת מימושכם – מימוש לא יעיל מספיק לא ינוקד באופן מלא (או בכלל).
- שימו לב שמגבלת 3 הנקודות היא על פתרון לוחות קשים, ותאורטית פתרון כל לוח. יש לוחות משחק שהפונקציה בסעיף 5 אינה יכולה לפתור, ופונקציה זו כן.

חלק ד׳ – שאלות תאורטיות

7. מענה על שאלות ב Ex8 - Quiz:

ענו על Ex8 – Quiz הנמצא במודל תחת סימניה Exercise 8.



חלק ה׳ – Code Review

8. סקר קוד \ Code 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 להתיחס 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!

בית הספר להנדסה ומדעי המחשב ע״ש רחל וסלים בנין

נהלי הגשה

בתרגיל זה, יש להגיש קובץ zip הנקרא ex8.zip המכיל בדיוק את הקובץ nonogram.py בו הפונקציות שמימשתם.

אין להגיש את ex8_helper.py. אין להגיש קוד המשתמש בפונקציות מ ex8_helper.py.

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

שימו לב ש Code Review הינו חובה.

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 1 3	16	16	1 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																		