אוניברסיטת בן גוריון

הפקולטה למדעי המחשב

מיני פרוייקט - נושאים במשחקי חשיבה – חידאתו

*מטרת ונושא הפרוייקט: להבין כיצד ניתן להשתמש בכלים מקורסים קודמים לפתרון המשחק חידאתו. במידת האפשר נתבונן גם במופעים אקראיים של הבעיה וננסה להבין את התנהגותם הטיפוסית.*

מרצה הקורס: פרופ' דניאל ברנד

מגישים: טל יצחק (204260533)

חיים סבן (308000371)

כלל הקוד בפרויקט נכתב בJava, וזמין ב-GitHub:

[github.com/talitz/Topics-in-Logic-Puzzle-Mini-Project-On-Hidato](https://github.com/talitz/Combinatorial-Optimization-MaxSAT)

סמסטר ב', תשע"ז, 2017

חידאתו – מבוא וכללי המשחק

חידאתו הינה חידה מתמטית בה נתון מערך , בו:

1. המספרים ו- ( מספר טבעי כלשהו) משובצים לתאים מסויימים במערך.
2. ייתכנו ערכים נוספים המשובצים לתאים במערך.
3. לא קיימים במערך 2 תאים עם מספר זהה.
4. קיימים תאים ריקים במערך, אותם פותר החידה צריך לשבץ עם מספרים.

נדרש למצוא שיבוץ של מספרים לתאים, כך שתיווצר שרשרת של מספרים עוקבים (שני מספרים הם מספרים עוקבים בשרשרת כאשר הם נמצאים על הלוח זה לצד זה, אנכית אופקית או באלכסון).

בהינתן מופע כלשהו של משחק החידאתו, ננסה לפתור את המשחק בעזרת רדוקציה לבעיות מוכרות במדעי המחשב.

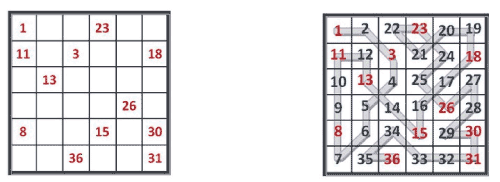
הערה: קיימות גירסאות של בעיית החידאתו בה המערך הוא לא בהכרח , אליהן לא נתייחס בפרויקט זה.

הפרויקט מתחלק ל2 חלקים:

1. רדוקציה לבעיית מסלול-המילטון-עם-אילוצים.
2. רדוקציה לSAT.

חלק 1: רדוקציה למסלול-המילטון-עם-אילוצים-על-הקודקודים

נשים לב שפיתרון של בעיית החידאתו מגדיר מסלול, שמתחיל בקודקוד עם הערך ההתחלתי בחידה (בדוגמא: 1), עובר דרך הריבועים עם הערכים 2,3,4, וכן הלאה עד הערך הגדול ביותר 36.

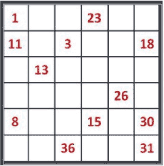


אם כן, בהגדרת הרדוקציה נרצה להשתמש בקופסה השחורה כאלגוריתם שבהינתן גרף, מחזיר את מסלול המילטון בגרף **עם-אילוצים-על-הקודקודים** אם קיים, ואם לא מחזיר "לא קיים".

נגדיר מספר מושגים שיעזרו לנו בהצגת האלגוריתם:

קלט לבעיה: מערך דו מימדי , כאשר כל תא במערך מייצג מספר נתון של בעיית החידאתו, או 1- כמקום ריק שעל השחקן למלא.

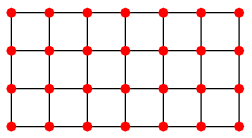
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| null | null | 23 | null | null | 1 |
| 18 | null | null | 3 | null | 11 |
| null | null | null | null | 13 | null |
| null | 26 | null | null | null | null |
| 30 | null | 15 | null | null | 8 |
| 31 | null | null | 36 | null | null |



דוגמא:

גרף סריג (grid/lattice graph): גרף דו-מימדי , אשר ניתן להצגה במערכת צירים אוקלידית , ויוצר אריח (סריג).

דוגמא: סריג 4 על 7:



נגדיר מסלול המילטון **עם אילוצים על הקודקודים**:

בהינתן גרף , כאשר לכל קודקוד קיים ערך (מספר טבעי המייצג את ערכו של הקודקוד, או null אם אין לקודקוד ערך) נגדיר מסלול המילטון עם אילוצים באופן הבא:

1. מסלול המילטון.
2. מקיים כי עבור המסלול , אם הקודקוד ה- במסלול הוא בעל ערך , אזי .

דוגמא:

עם הערכים:

*נקבל כי:*

1. הוא מסלול המילטון, אבל לא מסלול המילטון עם אילוצים על הקודקודים, משום שהקודקוד הוא הקודקוד השני במסלול ובעל ערך השווה ל- .
2. הוא מסלול המילטון עם אילוצים, משום שמתקיים:
   1. *הקודקוד הוא הראשון במסלול ועם ערך השווה ל-.*
   2. *הקודקוד הוא השני במסלול וללא ערך (null), ולכן אין תנאי עבורו.*
   3. *הקודקוד הוא הקודקוד השלישי במסלול ועם ערך השווה ל-.*
   4. *הקודקוד הוא הקודקוד הרביעי במסלול ועם ערך השווה ל-.*

*לכן, מסלול זה הוא מסלול המילטון עם אילוצים.*

אלגוריתם לפיתרון בעיית החידאתו בעזרת רדוקציה למסלול-המילטון-עם אילוצים-על-הקודקודים:

בהינתן קלט לבעיה, מערך דו מימדי :

1. בנה מהבעיה גרף סריג מכוון .
2. הסר את כל הקשתות הנכנסות לקודקוד שערכו .
3. הסר\_צלעות\_לא\_רלוונטיות\_בין\_קודקודים\_עם\_ערכים\_עוקבים().
4. אם קיים מסלול המילטון עם אילוצים בגרף :

4.1. אתחל *.*

4.1. יהי המסלול ההמילטוני המתקבל, לכל קודקוד :

4.1.1. אם לקודקוד לא נקבע ערך עדיין:

4.1.1.1. .

4.1.2. .

5. החזר את .

הסר\_צלעות\_לא\_רלוונטיות\_בין\_קודקודים\_עם\_ערכים\_עוקבים()

עבור על הגרף ולכל 2 קודקודים עם ערכים עוקבים (אנכית, אופקית או באלכסון):

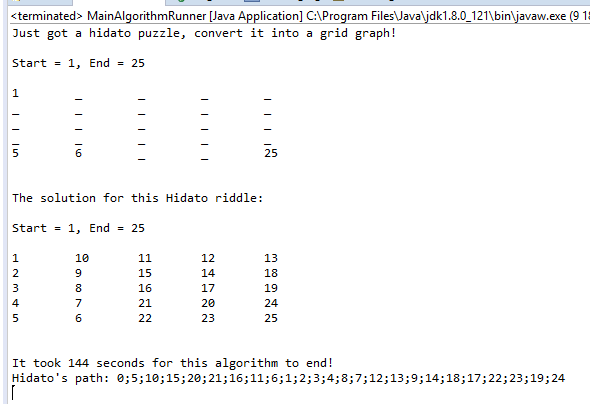
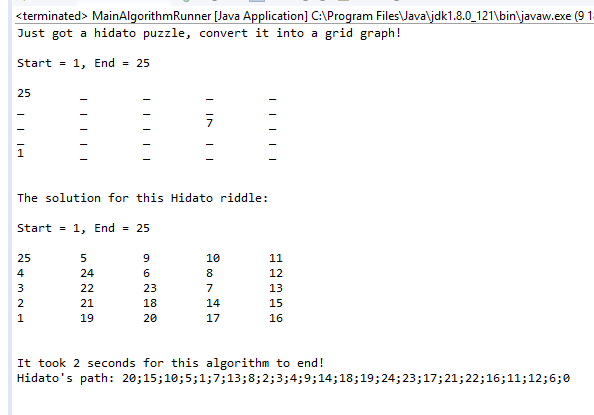
4.1. הסר את כל הצלעות הנכנסות ל- שהם לא .

4.2. הסר את (,).

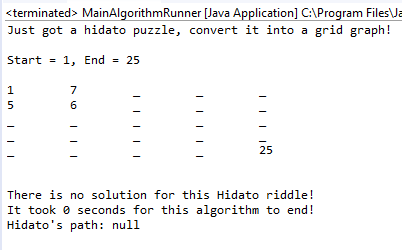
4.3. אם היא חלק ממסלול המילטון:

4.3.1. הסר את כל הצלעות היוצאות מ- שהם לא .

*מימשנו את האלגוריתם בJava, נדגים מספר הרצות שלו עבור קלטים שונים:*



*מקרה בו אנו מחזירים כי לא קיים פיתרון לחידאתו:*



חלק 2: רדוקציה לSAT

*כעת נראה איך אפשר לפתור את בעיית חידאתו בעזרת בעיית SAT.*

*כפי שבוצע בחלק 1, ראשית נבצע שוב רדוקציה למסלול המילטון, נבנה גרף סריג מכוון . נגדיר את n להיות מספר הקדקודים בגרף ו-p להיות ההסתברות של צלע להיות קיימת בגרף.*

*נגריל פרמוטציה מהקבוצה, כאשר התוצאה שנקבל תהיה הבסיס לגרף שנבנה (כלומר, הפרמוטציה תגדיר את מסלול ההמילטון בגרף זה).*

*נסמן:*

*V- מכיל קדקודים שהוגרלו מהפרמוטציה כאשר כל קודקוד מייצג את התא בבעיית החידאתו.*

*E - אם קיימת צלע בגרף השריג כך ש- , אזי ו הם שכנים במפת החידאתו. כלומר, נבנה את כל הצלעות המתקבלות מהפרמוטציה שקיבלנו כך שצלע מכוונת מקודקוד שהוגרל בפרמוטציה לקודקוד שבא אחריו בפרמוטציה. כך נבטיח מסלול המילטוני, כלומר פתרון אחד לפחות לבעיה.*

*לאחר מכן נגריל צלעות בגרף לפיp שנקבל בקלט. נשים לב שאין צלעות מקודקוד לעצמו.*

*דוגמא: בהינתן :*

*נניח ו- היא הפרמוטציה שהוגרלה, נקבל גרף (נייצג ע"י מטריצת שכנויות)*

*נראה כיצד נמיר את הגרף המייצג את בעיית החידאתו לידי ייצוג :*



*הגדרות:*

|  |  |
| --- | --- |
| *סימן* | *משמעות* |
|  | *פאזל של חידאתו (מייצג את התאים לפי אינדקסים)* |
|  | *פאזל מספרי (מייצג את המספרים בתוך התאים)* |
|  | *קבוצה המכילה את כל התאים ב-* |
|  | *קבוצה המכילה את כל התאים ב-* |
|  | *מייצג את התא ה-* |
|  | *שכן של תא ב* |
|  | *שכן התא המכיל את המספר העוקב לו* |
|  |  |

*נרצה שפסוקית ה- תקיים:*

1. *כל תא חייב להיות עם מספר בין עד ל-.*
2. *אין תא עם שני מספרים זהים.*
3. *אין שני תאים שיש להם את אותו המספר.*
4. *כל תא חוץ מהתא המכיל את חייב להכיל successor.*
5. *תאים שמכילים מספרים קבועים, לא יכולים להשתנות.*

*משתנים:*

*יהי H פאזל חידאתו עם ויהי מציין לקידוד אונארי כלומר את נוסחת ה- הסופית.*

*אנחנו נציין משתנה כאשר ) ו*

*לייצג את העובדה שהתא ה- מכיל מספר x. כתוצאה מכך ליטראל מציין את ההיפך. לפי הגדרה זאת אנחנו צריכים n משתנים בשביל כל תא . לכן הקבוצה של המשתנים של קוד האונראי מוגדרת כך:*

*ויש משתנים.*

*פסוקיות*

*מכיל חמישה קבוצות של פסוקיות המוגדרות באופן הבא, כאשר הקוד האונראי הינו הנוסחא:*

*מספר הפסוקיות:*

*כעת הרדוקציה של הגרףG1 שהגדרנו קודם היא:*

C3=

(¬C11 | ¬C21) ^ (¬C12 | ¬C22) ^ (¬C13 | ¬C23) ^ (¬C14 | ¬C24) ^ (¬C15 | ¬C25)

C4=

C5=

*פתירת ה-*

*נשתמש באלגוריתםDPLL : מימוש לאלגוריתם Backtracking: אלגוריתם כזה מנסה לבנות באופן הדרגתי פתרון לבעיה כלשהי, תוך הסתמכות על כך שהוא יכול לפעמים לזהות "באמצע" הבניה שמשהו התקלקל ואז להתחיל לחזור לאחור ולתקן את עצמו מבלי לבזבז עוד זמן על המשך הבניה המקולקלת עד הסוף. בשל כך, אלגוריתמי Backtracking הם יעילים יותר מאשר "סתם" חיפוש ממצה שעובר על כל ההשמות האפשריות.*

*ניתוח פתרון האלגוריתם*

*ניתוח 1- האם יש קשר ביןp , ההסתברות להגריל צלע בגרף לבין התוצאה שנקבל במקרה שלנו? האם התוצאה שנקבל היא הפרמוטציה שהגרלנו מלכתחילה?*

*ניתוח 2- בדומה לניתוח 1, רק ללא הפסוקית C5 כלומר האילוץ שהתא הראשון והאחרון יהיו חייבים להיות בתוצאה הסופית.*

*סטטיסטיקה של 1000 הרצות כל אחד.*

n=5

ניתוח 1

Example 1.1:

with probability 0.9 we get the initial path 25.4%

with probability 0.7 we get the initial path 50.9%

with probability 0.6 we get the initial path 66.4%

with probability 0.4 we get the initial path 86.8%

with probability 0.2 we get the initial path 97.2%

with probability 0.1 we get the initial path 98.1%

with probability 0.015 we get the initial path 99.8%

with probability 0.01 we get the initial path 99.7%

Example 1.2:

with probability 0.9 we get the initial path 0.8999999999999999%

with probability 0.7 we get the initial path 2.9000000000000004%

with probability 0.6 we get the initial path 5.4%

with probability 0.4 we get the initial path 17.5%

with probability 0.2 we get the initial path 39.6%

with probability 0.1 we get the initial path 60.3%

with probability 0.015 we get the initial path 80.4%

with probability 0.01 we get the initial path 81.5%

n=8

ניתוח 2

Example 2.1:

with probability 0.9 we get the initial path 0.2%

with probability 0.7 we get the initial path 1.0%

with probability 0.6 we get the initial path 5.3%

with probability 0.4 we get the initial path 28.1%

with probability 0.2 we get the initial path 85.5%

with probability 0.1 we get the initial path 97.3%

with probability 0.015 we get the initial path 99.4%

with probability 0.01 we get the initial path 99.5%

Example 2.2:

with probability 0.9 we get the initial path 0.0%

with probability 0.7 we get the initial path 0.3%

with probability 0.6 we get the initial path 0.1%

with probability 0.4 we get the initial path 1.2%

with probability 0.2 we get the initial path 16.400000000000002%

with probability 0.1 we get the initial path 50.6%

with probability 0.015 we get the initial path 83.5%

with probability 0.01 we get the initial path 86.4%