[](http://www.google.co.il/url?sa=i&source=images&cd=&cad=rja&docid=SJUliebdClSAlM&tbnid=XbmgYwpASkG82M:&ved=0CAgQjRwwAA&url=http://www.nirshamim.co.il/%D7%9E%D7%95%D7%A1%D7%93%D7%95%D7%AA/colleges/thumbs/abc/%D7%9B%D7%95%D7%9C%D7%9D/2&ei=JMxyUtv_M4GAhAeatYCIBw&psig=AFQjCNH3EqCTGYwTaw1rOc_yYm-k5p3uIA&ust=1383341476901169)

פרויקט גמר

2019

4X4 – אלגוריתם לפתרון קובייה הונגרית בזיכרון מינימלי

המחלקה למדעי המחשב מכללת ספיר

מנחה: ד"ר אמיר ספיר

מגישים

אורן זליגמן אייל אילן

**תוכן עניינים**

1. **הקדמה......................................................................................3**
   1. **על המסמך...................................................................3**
   2. **רקע.............................................................................4**
   3. **רקע תשתיתי................................................................5**
   4. **מגבלות.......................................................................5**
2. **מצב קיים – סקר שוק..................................................................6**
3. **היישום......................................................................................7**
   1. **רקע............................................................................7**
   2. **ייצוג הקובייה...............................................................7**
   3. **קוד האלגוריתם............................................................8**
4. **סיכום והרחבות..........................................................................11**
5. **ביבליוגרפיה..............................................................................11**
6. **הקדמה**
   1. **על המסמך**

* רקע – נדבר על הסיבות והצורך באלגוריתם זה ומדוע בחרנו בקובייה הונגרית
* רקע תשתיתי – באילו טכנולוגיות השתמשנו וכלים מיוחדים שעזרו בדרך
* מגבלות – בעיות בהן נתקלנו, כאלו שהתגברנו עליהן וכאלו שגרמו לנו להסתכל על דברים מנקודת מבט שונה
* מצב קיים – כיצד האלגוריתם שלנו עומד מול אחרים בשוק
* היישום – כיצד פועל האלגוריתם מבחינת הקוד ,האופטימיזציה ומבני הנתונים
* סיכום והרחבות – לאן הגענו ומה אפשר לשפר ולשנות
  1. **רקע**

בעולמנו כיום קיימות בעיות רבות הדורשות קבלת החלטות, בין אם אלו בעיות פשוטות הנפתרות בזמן קצר ובין אם בעיות הדורשות משאבים רבים לקבלת פתרון, ואפילו בעיות להן אין פתרון מוגדר ולהן נרצה למצוא פתרון מיטבי.

בתחום מדעי המחשב נוכל להגדיר כל בעיה בעלת מספר אפשרויות איתן נוכל להתקדם כעץ אפשרויות, ונרצה למצוא אלגוריתם ,ואפילו מימוש תוכנתי, אשר פותר את הבעיה בין אם פתרון מושלם אם קיים או פתרון מיטבי לאדם הפותר.

קיימות מספר בעיות אשר עלולים להיתקל בהן בדרך למציאת אלגוריתם שכזה: האלגוריתם המוצע לוקח זמן ארוך ביותר להגיע לפתרון , יש לבדוק אם פתרון שקיבלנו אכן מיטבי ,מימוש האלגוריתם מוגבל ע"י המערכת אשר תריץ אותו ועוד ועוד...

נרצה להתמקד בבעיה מסוימת והיא מגבלת הזיכרון החומרתית של מערכות המריצות אלגוריתמי קבלת החלטות. כידוע בעץ בו כל צומת מהווה החלטה ולכל החלטה ישנן מספר אפשרויות, מספר האפשרויות הסופי יגדל בצורה מערכית לגובה העץ. בהינתן בעיה קשה מספיק יכול להיות מצב בו מערכת הבאה לפתור אותה תכשל למצוא פתרון כלשהו לא בגלל שהאלגוריתם אינו יכול לפתור אותה בתאוריה אלא בגלל שהאלגוריתם הנבחר לא מתאים לדרישות החומרה, נקודתית כמות הזיכרון מוגבלת במערכות אמתיות ואלגוריתמים רבים המקובלים בעולם מדעי המחשב אינם מתחשבים בזאת.

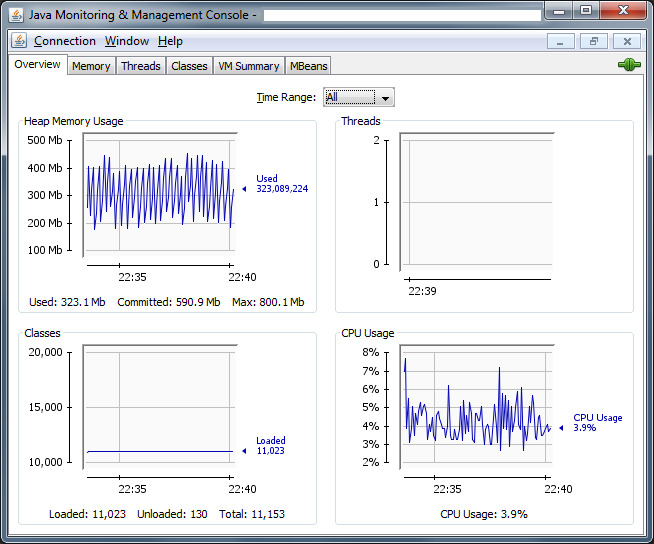
בעיית פתרון הקובייה ההונגרית מהווה מצע מצוין להדגמה שגם בעיה קשה כזאת ניתנת לפתרון בעזרת אלגוריתם מתאים. לא ידוע כיום על חסם מלרע למספר הפעולות הדורשות לפתרון כל קובייה הונגרית כלשהי בגודל 4X4 אך חסם שכזה מוערך להיות בסביבות ה30 מהלכים. כלאמר גובה העץ יהיה ~30 ומספר האפשרויות בכל צומת 24 ,כמספר ההזזות בקובייה כזו, ולכן הזכרון הדרוש גדול משכל מערכת קיום יכולה להכיל.

אלגוריתם הפתרון שנציע יפתור את הקובייה בעזרת זיכרון מוגבל בהרבה ותוך זמן ריצה הגיוני.

* 1. **רקע תשתיתי**

**”**אלגוריתם הוא דרך שיטתית וחד-משמעית לביצוע של משימה מסוימת, במספר סופי של צעדים**“.**

את מימוש האלגוריתם החלטנו לכתוב בשפת **JAVA**, אך מהיותו של הפרויקט אלגוריתמי והחלק הארי בו הוא הרעיון, מימוש האלגוריתם יכול היה להתבצע בכל שפת תכנות עילית מונחת עצמים.השימוש בjava נתן לנו יכולת לשמור על פשטות בחלקים שפחות נוגעים לאלגוריתם,למשל מימוש הקובייה, ובאותו הזמן סיפק לנו יכולת מעקב אחר שימוש בזיכרון בעזרת JConsole , וכלים נוספים בהם השתמשנו ליצור את האלגוריתם,למשל Reflection.



JConsole

* 1. **מגבלות**

בשלב התכנון עוד לפני שהתחלנו לכתוב קוד חשבנו שדרך העבודה תהיה שנצליח לפתור את הקובייה, ובפרט נצליח להגיע לכל שלביע הביניים שהצבנו לעצמנו, ולבסוף נסתכל על כמות הזיכרון בה השתמשנו וננסה להפחיתה.

המציאות הייתה שונה ומהר מאוד הבנו שנאלץ לחשוב על דרכי האופטימיזציה תוך כדי המעבר על הקוד,נתקלנו בשגיאות של חוסר זיכרון במערכת ולאחר שסידרנו זאת נתקלנו בבעיות אשר לוקחות זמן לא סביר לפתרון.

בהמשך נפרט כיצד התגברנו על בעיות אלו בפירוט רב יותר.

1. **מצב קיים – סקר שוק**

לפתרון קובייה הונגרית 4 על 4 לא קיימים אלגוריתמים אופטימליים לפתרון הבעיה אבל קיימים אלגוריתמים המנסים לקרב לפתרון אופטימלי.

**A star –** אלגוריתם חיפוש המוצא פתרון אופטימלי המשתמש בהיוריסטיקה. האלגוריתם משתמש בזיכרון רב יותר זיכרון ואי אפשר להריצו על מטלות הדורשות מספר פעולות רב כמו קובייה הונגרית 4 על 4.

**BFS -** אלגוריתם חיפוש לרוחב "טיפש" המבטיח פתרון אופטימלי אך רץ בסיבוכיות זמן ומקום מערכית.

**IDDFS –** אלגוריתם המבטיח אופטימלי ורץ בסיבוכיות זמן מערכית וסיבוכיות מקום ליניארית, חיפוש לעומק כאשר בכל איטרציה העומק המקסימלי גדל, בודק את אותה כמות הצמתים כמו BFS.

**אלגוריתם 4X4 –** דומה לIDDFS אך בעל אופטימיזציות גבוהות של ניהול הזיכרון ומותאם ספציפית לקובייה ההונגרית.

1. **היישום**
   1. **רקע:**

האלגוריתם שלנו כתוב כולו בשפת JAVA ,תחילה היה עלינו ליצור ייצוג לקובייה ההונגרית וליישם את התנועות האפשריות שלה. לאחר מכן התחלנו לכתוב את אלגוריתם הפתרון עצמו אשר משתמש במצבי הביניים לפי שיטת הרדוקציה לפתרון קובייה הונגרית 4 על 4, ומכיל את האופטימיזציות לזיכרון. לבסוף המיין המריץ את הקובייה והפתרון שלה ומציג למסך את השלבים התנועות הדרושות מהקובייה ההתחלתית ועד לקובייה הפתורה.

* 1. **ייצוג הקובייה:**

מחלקת cube מכילה את הייצוג המלא של הקובייה ואת השיטות המייצגות תנועות חוקיות עליה.

הצבעים של הקובייה נשמרים בשישה מערכים דו ממדיים של STRING הממוספרים 1-6.

כל מערך מייצג צבע אחר בקובייה הפתורה ולכן יש חשיבות לסדר שלהם בשיטות ההזזה.

כך למשל הפאה האדומה תהיה חייבת להיות מנוגדת לפאה הכתומה, הירוקה מנוגדת לכחולה והלבנה מנוגדת לצהובה, בנוסף קיימת חשיבות לאיזו מן הפאות המנוגדות נמצאת היכן.

קיימות 24 שיטות הזזה חוקיות על הקובייה : 4 בכל שלושת המישורים כפול 2 לכיוון ההפוך.

שיטות אלו נשמרות במערך בסדר כך שהאינדקס של כל שיטה משלים לאינדקס של השיטה בכיוון ההפוך לה, למשל הזזת שורה 1 בכיוון עם השעון תהיה באינדקס 0 והזזת שורה 1 בכיוון נגד השעון תהיה באינדקס 23.

קיים גם מערך קטן יותר של שיטות הזזה המסודר באותו האופן אך מכיל שיטות המזיזות פאות חיצוניות בלבד וזאת למען מצבי ביניים בהם ידוע כי רק אלו נדרשות .

קיימת גם שיטת הדפסה המציגה למסך את כל פאות הקובייה לפי סדר מסוים.

כמו כן רשימה של כל פונקציות ההזזה שנעשו על הקובייה מהתחלת הפתרון ועד לזמן מסוים, ושיטת הדפסה של פונקציות אלו.

עשינו שימוש שיטת REFLECTION אשר נותנת לנו גישה לשיטות של אובייקטים בזמן ריצה, כך אנו מוצאים כל שיטה לפי שמה ומכניסים לתוך המערך בעת פעולת הקונסטרקטור והקופי קונסטרקטור.

בקונסטרקטור נאתחל את רשימת השיטות שבוצעו עם שיטת ROOT ריקה לייצוג ההתחלה.

* 1. **קוד האלגוריתם:**

במימוש האלגוריתם שני חלקים עיקריים:

החלק הראשון מקבל קובייה כלשהי ומנסה להגיע לקוביית יעד המפורטת על-פי שיטות שלבי הביניים, בחלק זה נוצר עץ אפשרויות בו כל צומת נוצרת על ידי ביצוע אחת או יותר משיטות ההזזה של הקובייה ובשלב זה גם ממומשות אופטימיזציות הזיכרון שהכנסנו למערכת.

החלק השני מכיל את כל שלבי הביניים, חלקם התקבלו מידית על פי שיטת הרדוקציה לפתרון קובייה הונגרית של 4 על 4, וחלקם נאלצנו לגלות תוך כדי העבודה ואלו נוצרו כשלבי ביניים נוספים כאשר לא היה די זיכרון למערכת כדי להשלים שלב ביניים מלא ולמצוא את הקובייה הרצויה.

שני החלקים הללו נמצאים במחלקת solve המכילה בתוכה גם קונסטרקטור המאתחל את רשימת שלבי הביניים באופן דומה לזה שהוצג ברשימות שנמצאות בייצוג הקובייה במחלקת cube, כמו כן נשמרת קובייה אליה נעביר את פרטי קוביית היעד בכל שלב לאחר שנמצאה כמו פרטי התנועות שנעשו עליה עד כה, וכן שיטת ערבוב בה נשתמש בכדי למצוא קובייה מעורבבת רנדומלית עליה נחיל את הפתרון. הערבוב נעשה בצורה בה נלקחת קובייה פתורה ועליה מבוצעות 25 תנועות הזזה הנבחרות רנדומלית, דבר שעל פי הספרות שמצאנו מבטיח ערבוב טוב מספיק, לאחר מכן תנועות אלו נמחקות מהמערכת וכך נקבל קובייה מעורבבת לחלוטין ומוכנה לפתרון.

* + 1. חלק ראשון - בניית וסריקת העץ:

בחלק זה אנו מקבלים שיטת ביניים המייצגת מטרה אשר נרצה לוודא שהיא מתקיימת ובמידה וכן הרי הגענו לצומת יעד ונשמור את פרטי הקובייה.

ראשית נאתחל שני תורים queue3 , queue4 הראשון יאחסן רשימות של תנועות הזזה המייצגות את המעבר מהקובייה האחרונה שמצאנו עד לקוביית צומת אותה נרצה לבדוק, אך רק ברמה מסוימת של העץ. בעת שנגיע לרשימה שכזו נבצע את התנועות שבה על עותק של הקובייה השמורה אצלנו ולאחר מכן נבדוק התאמתה לדרישות היעד ,במידה והדרישה התקיימה הגענו למצב מטרה, נשמור את פרטי הקובייה ונמחק את הזיכרון בו השתמשנו עד כה, במידה זהו אינו מצב מטרה נייצר רשימות חדשות שכל אחת מכילה את איברי הרשימה העכשווית אך נוסיף לכל אחת תנועת הזזה שונה בסוף , חלק מהרמה הבאה בעץ, את רשימות אלה נכניס לתור השני. כאשר התור הראשון התרוקן נעביר את כל הרשימות מהתור השני לתוך הראשון, שיטה זו תורמת לכך שבמידה והגענו למצב מטרה באמצע הרמה חסכנו את שמירת הזיכרון של הרמה הבאה עד כדי חסכון בזיכרון של רמה שלמה. השימוש ברשימות של תנועות הזזה במקום בשמירת קוביות כצמתים מפחית משמעותית את הזיכרון הדרוש.

יעילות נוספת אותה מימשנו בחלק זה היא החסכון בהכנסת רשימות בעלות תנועות לא הגיוניות לעץ, למשל תנועות מנוגדות(בעלות אינדקסים משלימים במערך התנועות) או אם קיימת רשימה בה אותה התנועה קורית 3 פעמים ברצף (שווה ערך לתנועה המנוגדת פעם אחת) או אותה התנועה פעמיים ברצף על חצי מהתנועות (שווה ערך לפעמיים אותה התנועה בחצי השני של התנועות), יעילות זו מקטינה את גורם הגדילה של העץ החל מהרמה השנייה.

* + 1. חלק שני – שלבי הביניים:

שלבים אלו המוגדרים על פי שיטת הרדוקציה לפתרון קובייה הונגרית 4 על 4 מיוצגים על ידי שיטות בשם stepcheck, כל שיטה כזו מקבלת קובייה ומחזירה ערך בוליאני באם תנאי מסוים מתקיים. כל תנאי כולל שהדרישות של התנאים הקודמים מתקיימים אף הם.

השיטה במיקום האפס בודקת אם במערך face3 במקום ה[1][1] נמצא צבע אדום.

השיטה הבאה בודקת אם במיקום לידו [1][2] נמצא גם צבע אדום.

השיטות השנייה והשלישית משלימות את הריבוע הפנימי האדום על פאה face3.

4 השיטות הבאות מבצעות בדיקה לקבלת ריבוע פנימי כתום בפאה המנוגדת face4.

לאחר מכן נחפש ריבוע ירוק על פאה face1, ריבוע צהוב על פאה face 6, ריבוע כחול על פאה face5, הפאה הנותרת תהיה בהכרח בעלת ריבוע לבן.

כעת נחפש 12 בלוקים המורכבים כל אחד מ2 חתיכות זהות וממוקמים באמצע כל שורה .

נתחיל בפאה האחורית:

stepCheck20 יבדוק שהצבע בפאה האחורית face6 במיקום [3][1] זהה לזה שבמיקום שלידו [3][2], וגם שהצבע בפאה הצדדית face5 במיקום [3][1] זהה לזה שבמיקום שלידו [3][2], כעת קיבלנו את הבלוק הראשון שלנו.

שלושת השיטות הבאות ימצאו בקלות את 3 הבלוקים הנותרים בפאה האחורית עם הפאות הסמוכות לה.

כעת נחפש בלוקים בפאה הקדמית, לצורך הגעה לבלוק כלשהו בפאה זו יכולות להידרש כ11 תנועות, מספר שבעץ ידרוש זיכרון שהמערכת לא יכולה לספק, לכן נאלצנו להוסיף שלב ביניים כהכנה למציאת הבלוק, הוספה זו תדרוש במקרה הגרוע ביותר 5 מהלכים ובשלב מציאת הבלוק שלאחריה ידרשו 6 מהלכים מקסימום.

בעזרת שיטות אלו נצליח למצוא את כל 4 הבלוקים בפאה הקדמית face1.

כעת נותרו 4 בלוקים על הפאות הצדדיות אותם נמצא באותה השיטה.

במצב זה אפשר להתייחס לקובייה ככזו של 3 על 3 בה ובעזרת הגדרת מספר תנועות קטן מחדש נוכל להמשיך ולפתור את הקובייה בקלות רבה הרבה יותר.

1. **סיכום והרחבות:**

באגים ידועים:

* קיימים 2 שלבי ביניים בהם המערכת יכולה להיתקע ולא להגיע למצב המטרה כנראה עקב אחת האופטימיזציות בשילוב עם בחירה של סדר תנועות ספציפיות.
* לא סיימנו לבנות את המערכת ובזמן כתיבת מסמך זה היא לא מגיעה למצב של פתרון שלם של הקובייה, בשלב זה המערכת נתקעת בחיפוש אחר הבלוק האחרון, דרושה מציאת שלב ביניים נוסף בכדי להגיע אליו אך זהו מצב פתיר ולאחר הגעה אליו שאר הפתרון אמור להיות די מהיר ופשוט.

אפשרויות להרחבה:

* ניתן לנסות ולממש את האלגוריתם בעזרת שלבי ביניים שונים ממה שניסינו, למשל בשיטת השכבות במקום שיטת הרדוקציה.
* לא ידוע כיום על היוריסטיקה מושלמת עבור קובייה של 4 על 4, אך קיימות מספר היוריסטיקות שגם הן לא מושלמות עבור קובייה של 3 על 3.
* לא ידוע כיום על מספר אלוהי עבור קובייה של 4 על 4 אך הוא מוערך להיות בין 30 – 50.
* ניתן לשפר במקצת את אופטימיזציות ביטול התנועות המנוגדות כך שתהיה אפשרות לביטול של תנועה משלב קודם.

1. **ביבליוגרפיה:**
   1. <http://www.rubiksplace.com/cubes/4x4/> - באתר זה מפורטים שלבי הביניים לפתירת קובייה הונגרית על פי שיטת הרדוקציה.
   2. <http://cubezzz.dyndns.org/drupal/?q=node/view/541> - המספר האלוהי עבור קובייה של 4 על 4.
   3. <http://theoryofprogramming.com/2018/01/14/iterative-deepening-depth-first-search-iddfs/> - הסבר על IDDFS.
   4. <http://ceur-ws.org/Vol-1885/57.pdf> - הסבר על היוריסטיקות של קובייה הונגרית.