מבוא לבינה מלאכותית

1. **<u>שאלות הבנה:</u>**

2.1 אופטימליות:

2.1.1 בבעיית <u>משחק הבלוקוס</u> שהרצנו, אנו מחפשים דרך למקם 5 חלקים על הלוח. לכן כל פתרון הממקם את חמשת החלקים על הלוח, הוא פתרון חוקי ואופטימלי.
כפי שראינו בתרגול, אלגוריתם ה-BFS שעובר על כל המצבים בעץ לרוחב, ימצא את המסלול הכי קצר. לכן, בבעיה זו שהעלות זהה בכל צעד, בהכרח נמצא פתרון חוקי, לכן <u>BFS ימצא פתרון אופטימלי.</u>
גם אלגוריתם ה-DFS ימצא בהכרח פתרון חוקי ולכן <u>גם הוא ימצא פתרון אופטימלי</u>. כפי שראינו בתרגול, אלגוריתם ה-DFS הוא complete אם אנו נמנעים ממעגלים ואם m- עומק העץ המקסימלי, הוא סופי. אנו נמנעים ממעגלים ע"י שמירה של כל המצבים בהם ביקרנו כבר, כך, ברגע שנגיע למצב שביקרנו בו כבר, לא נפתח אותו שוב. בנוסף, m הוא סופי, שכן גודל הלוח הוא סופי ומספר החלקים שממקמים על הלוח הוא סופי, לכן מספר המצבים יהיה סופי גם. במילים אחרות, כל מצב הוא קונפיגורציה של הלוח עם מצבים עליו, ולכן מספר המצבים הוא מספר הקונפיגורציות של הלוח, שהוא חסום על ידי מספר החלקים (שהוא סופי) כפול מספר אפשרויות ההנחה של כל חלק על הלוח (שהוא סופי), כלומר בסה"כ סופי.

בבעיית <u>משחק הפקמן</u> שהרצנו, אנו מחפשים את הדרך הקצרה ביותר להגיע לנקודת הסיום במסלול מנקודת ההתחלה, כאשר העלות זהה בכל צעד.
כאמור, אלגוריתם הBFS ימצא את המסלול הקצר ביותר (שכן הוא עובר על העץ לרוחב) לכן אלגוריתם הBFS ימצא את המסלול הקצר מנקודת ההתחלה לנקודת הסיום ומכאן <u>שBFS ימצא את הפתרון האופטימלי</u>.
לעומת זאת, אלגוריתם הDFS, שעובר על המסלולים בעץ לעומק, לא בהכרח ימצא את הפתרון הקצר ביותר (לדוגמא, יבקר קודם במסלול חוקי ארוך יותר ויחזיר אותו) ולכן <u>לא בהכרח יחזיר פתרון אופטימלי</u>.

2.1.2 כאשר בדקנו על מחשבי האוניברסיטה, התוצאות שקיבלנו היו:
<u>עבור משחק הבלוקוס game.py:</u>

	Expanded nodes	Score	Cost	Running time in sec
DFS	128	17	5	~0.190185
BFS	1655	17	5	~1.602868

ניתן לראות, שגם אלגוריתם הDFS וגם אלגוריתם הBFS, מגיעים לתוצאות זהות- score 17 וcost 5 -כלומר, מניחים 5 חלקים על הלוח שתופסים 17 משבצות על הלוח. ומכאן נוכל להסיק ששני הפתרונות הם אופטימליים (שכן, כפי שהסברנו בסעיף הקודם, כל פתרון שמצליח להניח 5 חלקים על הלוח יהיה אופטימלי).

עבור משחק הפקמן pacman.py:

	Expanded nodes	Score	Cost	Running time in sec
DFS	144	380	130	~0.0
BFS	267	442	68	~0.0

ניתן לראות שאלגוריתם הBFS מוצא פתרון בעלות של 68 ומגיע לניקוד 442 ואילו אלגוריתם הDFS מוצא פתרון בעלות 130 ומגיע לניקוד 380. נתייחס לעלות שהיא מספר הצעדים:
מאופן פעולתו, אלגוריתם ה-BFS יחפש את כל המסלולים בגודל 1, לאחר מכן ימשיך לחפש את כל המסלולים בגודל 2, וכך הלאה... לכן בהכרח כשיגיע לפתרון, הוא יגיע לפתרון הקצר ביותר, ויחזיר אותו. מכאן נוכל להסיק שBFS אכן מצא את הפתרון האופטימלי. כפי שהסברנו קודם, בבעיה זו, שכל צעד הוא בעל אותה עלות, BFS מוצא את המסלול הקצר ביותר- במקרה זה באורך 68. לעומת זאת הDFS מצא מסלול באורך 130, וזה לא המסלול האופטימלי.

2.1.3 כפי שניתן לראות בטבלאות ב2.1.2 , התוצאות שקיבלנו <u>תואמות</u> למה שציפינו לקבל.
<u>במשחק הבלוקוס</u>- ציפינו ששני האלגוריתמים יחזירו תוצאות אופטימליות, שכן פתרון אופטימלי בבעיה זו, הוא פתרון המניח את כל 5 החלקים על הלוח ושני האלגוריתמים עושים את זה. ואכן קיבלנו תוצאות זהות ,כלומר, הscore והcost של שני האלגוריתמים זהים - שניהם מצליחים להניח 5 חלקים על הלוח ושניהם תופסים 17 משבצות על הלוח.

<u>במשחק הפקמן</u>- ציפינו שאלגוריתם הBFS ימצא את הפתרון האופטימלי, שבבעיה זו, מדובר בדרך הקצרה ביותר לנקודת הסיום, כאשר עלות כל צעד היא זהה. זה אכן תואם את התוצאה שקיבלנו- בה אלגוריתם הBFS הגיע לעלות 68 ואילו אלגוריתם הDFS הגיע לעלות 130. כאמור ב-2.1.2, אלגוריתם ה-BFS ימצא את הפתרון האופטימלי, כלומר את המסלול הקצר ביותר מנקודת ההתחלה לנקודת הסיום. בהתאם לציפיותינו, אכן אלגוריתם ה-BFS החזיר פתרון אופטימלי ואילו הDFS עבר על מסלול מסוים לעומק, וכתוצאה מכך לא הגיע לפתרון האופטימלי. ניתן לראות זאת אף באנימציה של המשחק- ניתן לראות שבאלגוריתם הBFS הפקמן הולך במסלול האופטימלי אל נקודת הסיום, ואילו הDFS מוצא לו דרך אחרת, ארוכה יותר.

2.2 זמן ריצה:

2.2.1 תיאורטית אסימפטוטית, לBFS יש זמן ריצה טוב יותר משל DFS.
כפי שראינו בתרגול, זמן הריצה של BFS הוא $O(b^d)$ כאשר b הוא הbranching factor -רמת ההסתעפות המקסימלית של כל קודקוד בעץ החיפוש, וd הוא עומק הפתרון המינימלי. ואילו זמן הריצה של DFS הוא $O(b^m)$ כאשר m הוא העומק המקסימלי בעץ.
BFS עובר על העץ רמה רמה-לרוחב, ומוצא את הפתרון האופטימלי (כאשר העלות זהה בכל צעד), לכן נקבל שזמן הריצה הוא
$1 + b + b^2 + .. + b^d = O(b^d)$.

DFS עובר על העץ לעומק, לכן במקרה הגרוע ביותר, נעבור על כל הקודקודים בעץ ולכן $O(b^m)$ זהו זמן הריצה של אלגוריתם הDFS.
מתקיים $m \leq d$ ולכן $b^d \leq b^m$ ומכאן שBFS בעל זמן ריצה טוב יותר משל DFS.

2.2.2 בבעיית הבלוקוס- ניתן לראות בטבלאות שצירפנו ב2.1.2 שאלגוריתם ה DFS הרחיב 128 קודקודים, ואלגוריתם הBFS הרחיב 1655 קודקודים.
בבעיית הפקמן- ניתן לראות בטבלאות שצירפנו ב2.1.2 שאלגוריתם ה DFS הרחיב 144 קודקודים, ואלגוריתם הBFS הרחיב 267 קודקודים.
בשתי הבעיות, ניתן לראות שאלגוריתם הDFS הרחיב פחות קודקודים מאשר אלגוריתם הBFS, לכן הDFS מוצא פתרון מהר יותר.
אינטואיטיבית, בבעיית משחק הפקמן בה אנו מחפשים מסלול לנקודת הסיום, אלגוריתם הBFS עובר רמה אחר רמה בעץ, כך הוא עובר על כל המסלולים האפשריים בכל רמה, ומוצא את המסלול הקצר ביותר. ניתן לראות שאכן העלות של הBFS היא 68 וקטנה יותר מהעלות של הDFS. אלגוריתם הDFS לעומת זאת, עובר על המסלולים בעץ לעומק, ומסיים ברגע שהוא מוצא מסלול. כך, הDFS עשוי למצוא פתרון מהר יותר מאלגוריתם הBFS, אך פתרון זה לא בהכרח האופטימלי, כפי שקורה בהרצת משחק הפקמן- קיבלנו מסלול בעלות של 130 (וזה לא המסלול האופטימלי) אך ניתן לראות שהDFS הרחיב פחות קודקודים מהBFS כלומר הוא מצא פתרון מהר יותר.

2.2.3 ניתן לראות שהתוצאות שקיבלנו סותרות את התוצאות שציפינו לקבל. תיאורטית, ציפינו שזמן הריצה של הBFS יהיה קצר יותר מזמן הריצה של הDFS, אך קיבלנו שבפועל הDFS מוצא פתרון מהר יותר מהBFS. עם זאת, הפתרון שהDFS מוצא הוא לא בהכרח אופטימלי (כמו בעיית משחק הפקמן). זה יכול לקרות מאחר שאלגוריתם הDFS חוקר לעומק את העץ ויכול להגיע לפתרון בעומק העץ לפני שאלגוריתם הBFS, שעובר על כל המסלולים האפשריים בכל רמה של העץ, מספיק להגיע לפתרון.

5. היוריסטיקות אדמיסביליות VS לא אדמיסביליות

5.1 <u>היורסטיקה אדמיסבילית:</u>

5.1.1 נבחר את היוריסטיקה הלא-טריויאלית הבאה: לכל מצב, ההיוריסטיקה מחזירה את מספר הפינות על הלוח שלא מכוסות על ידי חלק כלשהו.

5.1.2 נראה כעת כי ההיוריסטיקה אדמיסבילית. נוכיח ראשית שההיוריסטיקה קבועה (קונסיסטנטית) ומכך אדמיסבילית. כפי שראינו בתרגול, עלינו להוכיח מההגדרה כי

$$\forall e = (v, v') \in E,\ \ h(v) \leq c(e) + h(v')\ \ and\ \ \forall v \in G\ , h(v) = 0$$

ראשית, נשים לב כי לכל מצב בקבוצת מצבי המטרה, ההיוריסטיקה (h(v)) היא אכן 0. שכן, אם אנחנו במצב מטרה כל הפינות מכוסות וההיוריסטיקה תחזיר 0 מאופן פעולתה.
כעת, תהי צלע $e = (v, v')$, כך ש-'v הינו successor של v. נשים לב כי מספר הפינות שכיסינו עד לצעד זה הוא $h(v)$, ומספר הפינות שנכסה לאחר הצעד הוא $h(v')$.

העלות של הצלע בבעיית corners, היא גודל החלק שמניחים במעבר בין מצב למצב (כלומר בין לוח בלי החלק ולוח עם החלק). כל חלק תופס לפחות כמספר הפינות שהוא מכסה, ולכן $h(v) - h(v') \le c(e)$. נוסיף לשני אגפי אי השיוון את $h(v')$, ונקבל
$\forall e = (v, v') \in E,\ \ h(v) \le c(e) + h(v')$, כנדרש.

5.2 היורסטיקה חדשה

5.2.1 נציע היורסטיקה חדשה, לא אדמיסבילית, המשתמשת במרחק מנהטן:

$$h(v) = \sum_{\substack{corner \,\in\, Corners \\ slot \,\in\, valid\ slots\ available}} manhattan(slot, corner) = \sum_{\substack{corner=(c_x,c_y) \,\in\, Corners \\ slot=(s_x,s_y) \,\in\, valid\ slots\ available}} |c_x - s_x| + |c_y - s_y|$$

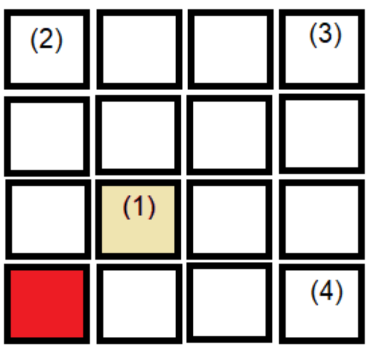
כאשר

$$\boldsymbol{Corners} = \{(0,0), (0, boardwidth - 1), (boardheight - 1,0), (boardheight - 1, boardwidth - 1)\}$$

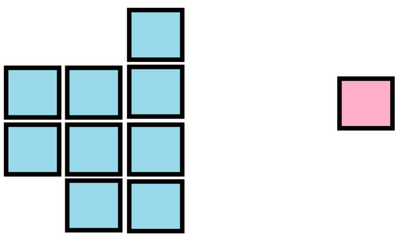
$$\boldsymbol{valid\ slots\ available} = \{(x,y) : there's\ a\ piece\ on\ the\ board, (x',y'), such\ that\ (x',y')\ is\ touching\ the\ corner\ of\ (x,y)\}$$

ההיוריסטיקה הזו סוכמת את כל מרחקי המנהטן מכל משבצת פנויה שאפשר להניח עליה חלק לפי חוקי המשחק, לכל פינה פנויה על הלוח. ההיורסטיקה הזו לא אדמיסבילית, שכן היא מעריכה יתר על המידה את העלות- נראה דוגמה בסעיף הבא.

5.2.2 נראה דוגמה שבה ההיוריסטיקה שהצענו נותנת קירוב טוב יותר מההיוריסטיקה הקודמת (הפתרון הנאיבי):
בהינתן המצב ההתחלתי הבא (רק המשבצת האדומה מונחת על הלוח) :



ובהינתן החלקים שנותרו:

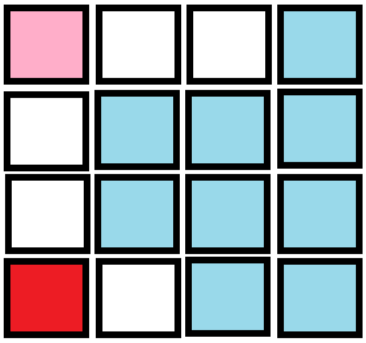


הפתרון הנאיבי יחזיר 3 (שכן, מספר הפינות שנותר לכסות הוא 3).
מאידך, ההיוריסטיקה החדשה שהצענו תחזיר 10, שכן:
המשבצת היחידה הפנויה שאפשר לעשות בה צעד היא המשבצת (1).
ההיוריסטיקה תחזיר:

$$h(v) = 3 + 4 + 3 = 10$$

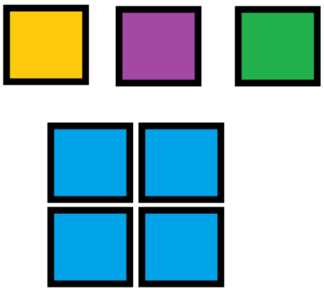
שכן, מרחק מנהטן מ(1) למשבצת (2) הוא 3 , מ(1) ל(3) הוא 4, ומ(1) ל(4) הוא 3.

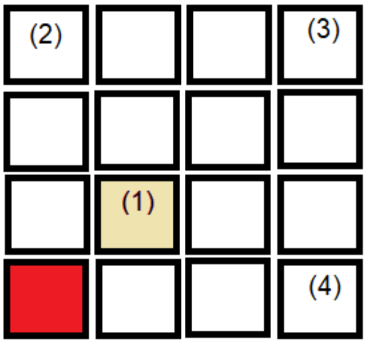
נשים לב, שהפתרון המינימלי לבעיה זו, ייראה כך:



כלומר, בפתרון זה כיסינו 10 משבצות (לא כולל החלק האדום שהיה במצב ההתחלתי), שזה **בדיוק** מה שההיוריסטיקה החדשה החזירה. ולכן היא נותנת קירוב טוב יותר.

כעת, נראה דוגמה נגדית לכך שההיוריסטיקה לא אדמיסבילית:
בהינתן מצב הלוח הבא, והחלקים הנותרים הבאים:



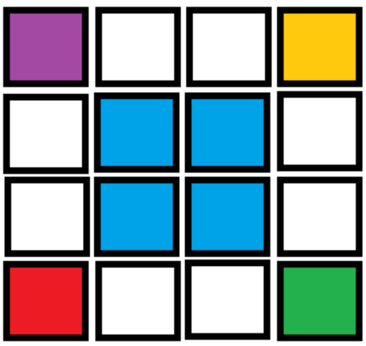


שוב, נשים לב כי המשבצת היחידה הפנויה שאפשר לעשות בה צעד היא המשבצת (1) . ההיוריסטיקה תחזיר:

$$h(v) = 3 + 4 + 3 = 10$$

שכן, מרחק מנהטן מ(1) למשבצת (2) הוא 3 , מ(1) ל(3) הוא 4, ומ(1) ל(4) הוא 3.

אבל נשים לב כי קיים פתרון יותר טוב:



נוכל להניח את החלק הכחול, ואז את החלק הסגול, הכתום והירוק, וכך נגיע למצב המטרה- כל הפינות מכוסות, ב 4 צעדים ובעלות של 7- יותר טוב ממה שההיוריסטיקה חזתה ולכן לא אדמיסבילית, כנדרש. (העלות היא 7 כיוון שכל שלושת החלקים שצירפנו על הלוח מכסים יחדיו 7 אריחים).