תרגיל הגשה 1

מגישים:
עודד ועלי
ערן סטון

תשובות בעבור סעיף 2.1

1. תיאורטית האופטימליות לא מובטחת במקרה של אלגוריתם BFS ו-DFS
במקרה של אלגוריתם BFS נניח כי יש לנו את קודקוד המקור v_0 ואת הקודקודים v_1, v_2 כך ש-v_1 הוא קודקוד המטרה. נניח כי $(v_0, v_1), (v_0, v_2), (v_2, v_1) \in E$ כך ש-
$c(v_0, v_1) > c(v_0, v_2) + c(v_2, v_1)$ מדרך העבודה של אלגוריתם BFS בו אנחנו עוברים כל פעם על החוליה ה-i וכך בונים עץ רוחב. לעולם לא נוכל לקבל את המסלול (v_0, v_2, v_1) ולכן לא נוכל לקבל פתרון אופטימלי לבעיה.
במקרה של אלגוריתם DFS אפשר לקחת את אותה הדוגמא שהצגנו מקודם וגם במקרה של בניית עץ עומק, אם האלגוריתם ילך תחילה לקודקוד v_1 אנחנו לא נמצא את המסלול הקצר ביותר ולכן לא נמצא פתרון אופטימלי לבעיה בכלל.
2. אלגוריתם BFS עשה 3119 מעברי מצבים שונים בעוד שאלגוריתם DFS פתח רק 162 מצבים שונים. על פי בדיקה שלנו על ההגדרות שבהן הרצנו את המשחק הלוח שנוצר מהרצת 2 האלגוריתמים השונים היה זהה. וגם על פי בדיקה ידנית שלנו כנראה קיבלנו פתרון אופטימלי עם כיסוי של 17 תאים על ידי החלקים. אבל כדי להסיק שהתוצאה אופטימלית היינו צריכים לבדוק ידנית ולנסות ידנית ולא יכלנו להסיק ישירות מהלוח שהוצג לנו שזהו פתרון אופטימלי
3. התיאורטי אומר כי האלגוריתמים לא מבטיחים אופטימליות בעוד שהפתרון האמפירי לדעתנו הוא כן אופטימלי לבעיה שהרצנו. למה זה? יש סידורי מצבים וגרפים שונים שעלולים לגרום לאלגוריתמים להוציא לנו פתרונות אופטימליים לדוגמא אלגוריתם BFS מבטיח אופטימליות אם המחיר למעבר בין 2 קודקודים הוא 1. לדוגמא אם המסלול האופטימלי הוא הראשון שנבחר על ידי DFS מכיוון שהאלגוריתם נכנס לעומק, ובכל מצב אם הוא יבחר את הפנייה האופטימלית של המסלול הוא יקבל מסלול אופטימלי אבל זה לא מובטח תמיד.

תשובות בעבור סעיף 2.2

1. נסמן m - העומק המקסימלי של עץ הנוצר מהפעלת האלגוריתמים
נסמן b - פקטור הפיצול המקסימלי
נסמן d - עומק הפתרון הכי גבוה בעץ
מבחינת אלגוריתם BFS הזמן ריצה הוא: $O(b^{d+1})$ מכיוון שהאלגוריתם בונה עץ רוחב, אנחנו למעשה נעמיק בעץ d+1 פעמים עד שנגיע לפתרון שנמצא בעומק d. מכיוון שהאלגוריתם הולך שכבה שכבה מקודקוד המקור אנחנו מקבלים באופן ודאי שלא תהיה ריצה לעומק m ונגיע לכל היותר לעומק d+1

מבחינת אלגוריתם DFS הזמן ריצה הוא: $O(b^m)$ מכיוון שהאלגוריתם בעבור כל שלב בעץ מתפצל לכל היותר ל-b שכנים של החוליה ה-i ולכן אנחנו בעצם רצים אקספוננציאלית
2. על פי הרצת הבעיות שהוצגו בתרגיל קיבלנו:
אלגוריתם BFS התפצל סך הכל ל-3119 קודקודים שונים ובמשחק פקמן 269 קודקודים

אלגוריתם DFS התפצל סך הכל ל-162 קודקודים שונים ובמשחק פקמן 269 קודקודים
אנחנו שמים לב לחריגה משונה במספר הקודקודים שהאלגוריתמים הביאו בשני הבעיות
השונות. נרצה להסביר את השוני בהסתכלות על הבעיות עצמן.
בבעיית פקמן יש לנו דרגת פיצול מאוד נמוכה (4 אפשרויות שונות) בעוד שבעיית Blokus יש
לנו x חלקים וכל חלק יכול להיות מונח ב-4 מצבים שונים כלומר דרגת הפיצול שלנו היא $4x$
לן לדוגמא אלגוריתם BFS ביקר הרבה פחות קודקודים בדרך לפיתרון שכן דרגת הפיצול
הייתה קטנה משמעותית.
בבעיית פקמן אנחנו יכולים לראות את הבעייתיות המסוימת של אלגוריתם DFS הוא
אלגוריתם שיורה לחלל בתקווה למצוא. לפעמים הוא נכנס לעומק הגרף ומוצא מהר מאוד
פתרון (כמו במשחק בלוקוס) ולפעמים הוא מטייל המון עד שהוא מגיע לפתרון כמו
באלגוריתם פקמן

3. כמו שהראינו סעיף קודם.
זמן הריצה של DFS התיאורטי הוא לא בהכרח יציב ומתקיים בכל בעיה שקיימת, לדוגמא
במשחק בלוקוס זמן הריצה היה קצר באופן משמעותי מזמן הריצה התיאורטי ובמשחק
פקמן זמן הריצה התחיל להיראות יותר דומה לזמן הריצה התיאורטי
לעומת זאת זמן הריצה התיאורטי של BFS מתאים באופן מושלם לזמן הריצה בפועל בשני
הבעיות השונות. אנחנו ממש יכולים לראות את תוצאת החישוב וכל זה מובטח לנו
מהשיטתיות היציבה והקבועה של האלגוריתם בניסיון לפתור בעיות בלי קשר למהות
הבעיה.

תשובות לסעיף 5.1

1. ההצעה שלי לפונקציית יוריסטיקה בעבור בעית כיסוי הפינות במשחק ״בלוקוס״ אני מציע
שבהינתן מצב (לוח) v אזי: מספר הפינות הריקות $= f(v)$
2. נרצה להראות Admissible
יהי קודקוד v_0 ומסלול $v_0, \dots v_n$ כאשר $v_n \in G$
נניח כי $h(v_0) = t$ אזי על מנת שהמסלול $v_0, \dots v_n$ יתקיים יש צורך לכסות t פינות נוספות
אזי סכום הפעולות $\sum_{i=1}^{n} c\big((v_i, v_{i+1})\big) \geq t$ שכן מצב המטרה חייב להיות מצב בו כל הפינות
מכוסות כלומר כל t הפינות הנותרות מכוסות
לפיכך אנחנו מקיימים את התנאי של Admissible

תהא קשת $e = (v, w) \in E$ נניח בשלילה כי מתקיים $h(v) - h(w) > c(e)$
אם $h(v) = y$ ו-$h(w) = x$ אזי נקבל $x - y > c(e)$ אבל אנחנו יודעים שעלות פעולה זה
מספר התאים שמכסה הפעולה ובמקרה הזה על מנת לעבור ממצב של x פינות פתוחות ל-y
פינות פתוחות נדרש חתיכה בגודל $x - y$ כלומר נקבל $x - y > c(e) = x - y$ וזו סתירה
לפיכך מתקיים $h(v) < c(e) + h(w)$ כלומר פונקציית היוריסטיקה הינה עקבית

תשובות לסעיף 5.2

1. הפעם נציע פונקציית יוריסטיקה שונה
יהיה מצב (לוח) v נגדיר $h(v) = 15 \cdot h'(v)$ כאשר $h'(v)$ הינה הפונקציה שהצענו תשובה
בסעיף קודם

הרצנו את הבעיה

game.py -p tiny_set_2.txt -f astar -s 6 6 -z corners -H blokus_c orners_heuristic

וקיבלנו זמני ריצה טובים יותר המתבטאים בפחות התרחבות מצבים 9 מצבים לעומת 4998 מצבים של הפונקציה הקודמת.

2. נרצה להראות כי הפונקציה $h(v)$ שהגדרנו איננה Admissible נראה זאת באמצעות דוגמאת ההרצה.

היה שלב רגע לפני האחרון ש-3 פינות כוסו בלוח ונשאר לכסות את האחרונה.

על מנת לכסות את האחרונה האלגוריתם הדביק חתיכה מגודל 4

התחלנו ממצב לוח ריק ופונקציית היוריסטיקה הייתה $h(v_0) = 15 \cdot 4$ ויש לנו מסלול v_0, v_1, v_2, v_3 כאשר v_3 מצב רצוי לפתירת הבעיה. סכום כל גודלי החלקים שהונחו היה 17 כלומר עלות המסלול $\sum_{i=0}^{2} c\big((v_i, v_{i+1})\big) = 17$ בעוד שקיבלנו $h(v_0) = 15 \cdot 4$ כלומר נקבל $h(v_0) > \sum_{i=0}^{2} c\big((v_i, v_{i+1})\big)$ כלומר דוגמא למה הפונקציה שהצענו איננה Admissible