Into to AI

**HW1**

Dana

Roey Graif 315111401

horizontal line

# 

# 

# 

# 

# 

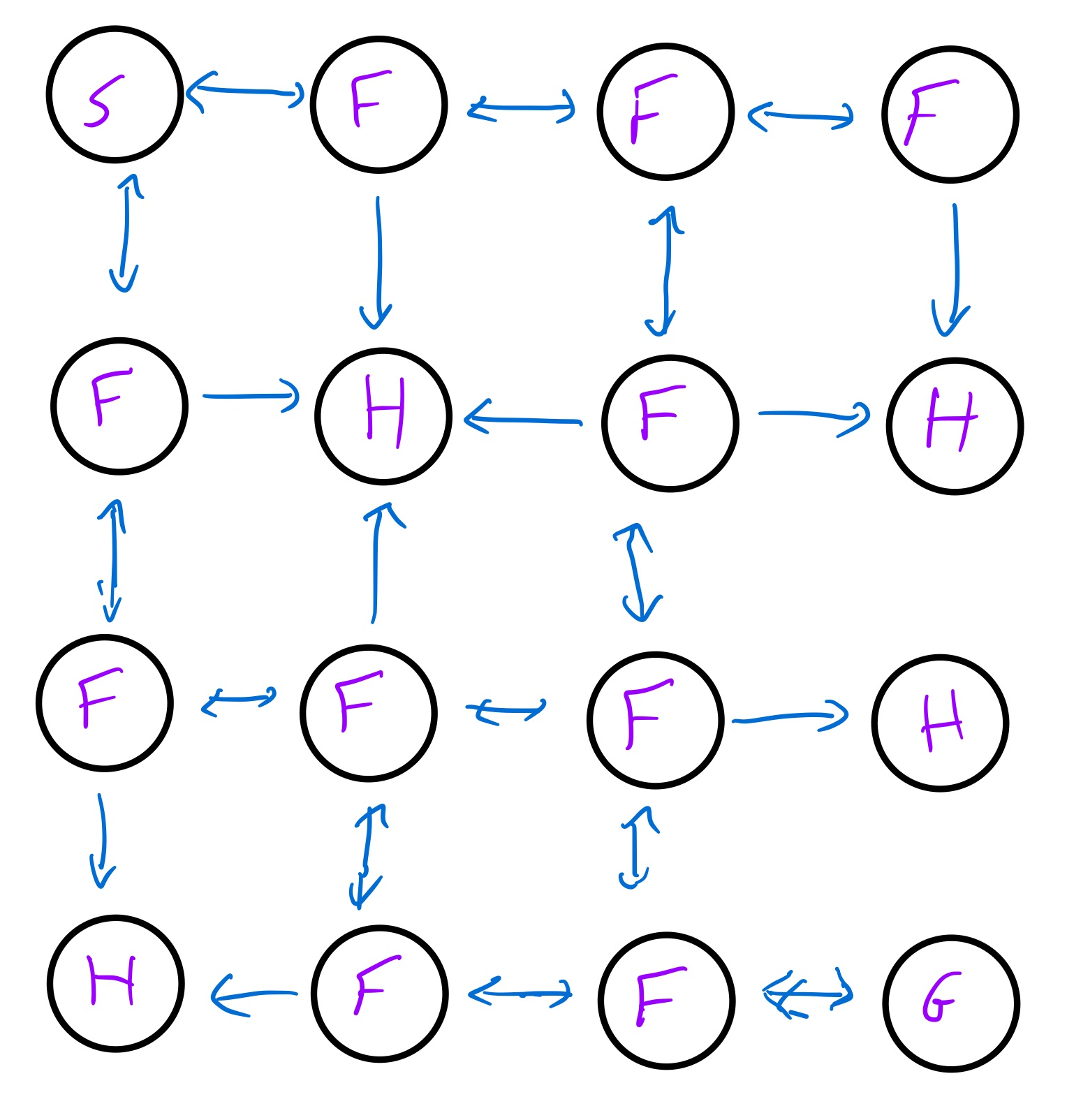
# 

# 

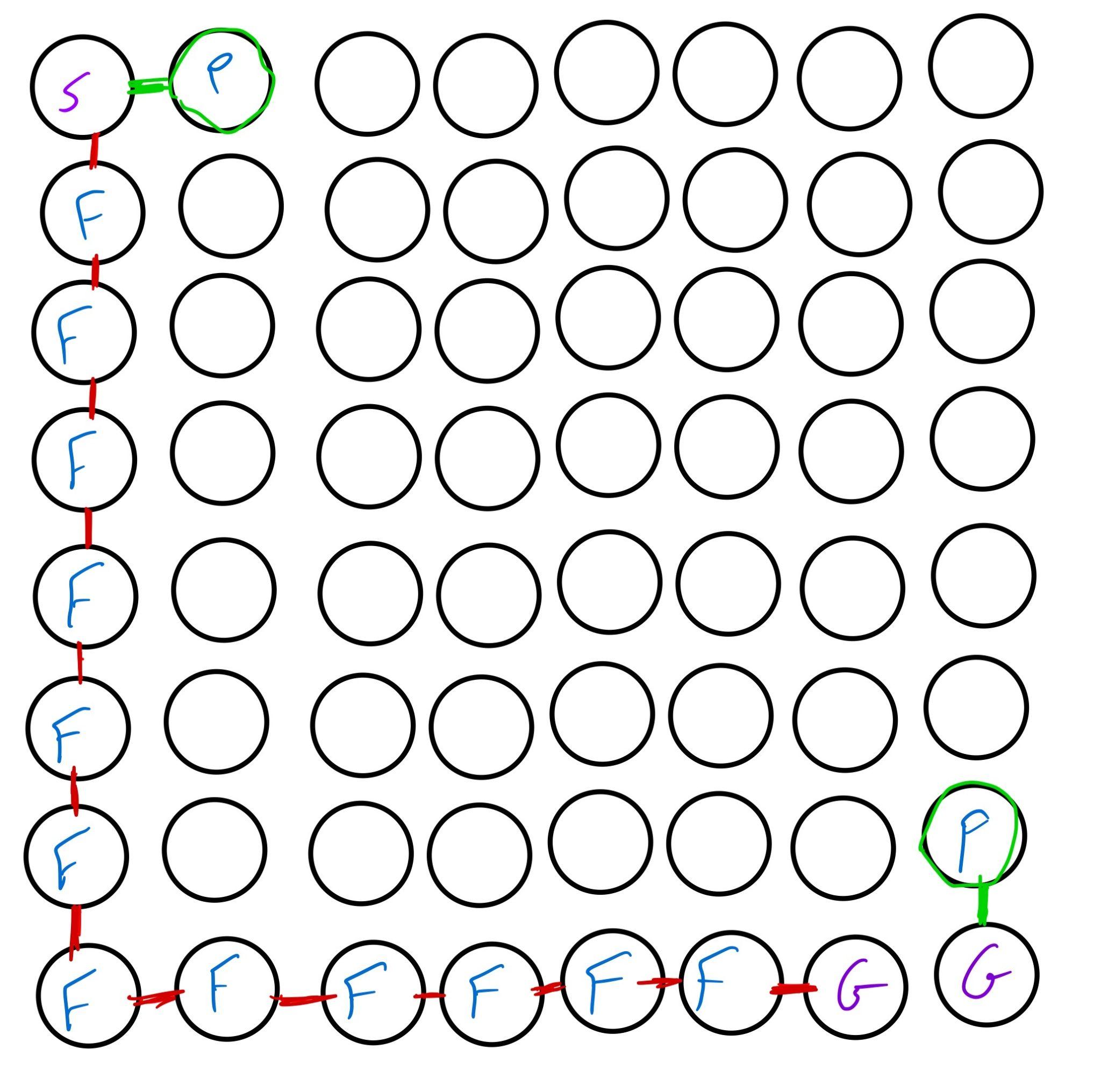
# 

# 

# שאלה 1

1. בוצע
2. מכיוון שS הקבוצה שהגדרנו בסעיף הקודם =|S| כלומר לכל מצב יש 5 אפשרויות בלוח של 64 משבצות.
3. הפונקציה domain על אופרטור down תחזיר לנו את כל המצבים **מלבד לשורה האחרונה**. מכיוון שאין אפשרות לרדת למטה בלוח בשורה האחרונה. הערה: אם הכוונה בתרגיל שאי אפשר להגיע לחור אז גם כל משבצת מעל חור **לא תוחזר**.
4. מכיוון שהוגדר לנו שהמצב התחלתי הוא למעלה מצד שמאל אז הפונקציה SUCC תחזיר את המצבים שנמצאים מימין ומלמטה למצב ההתחלתי.
5. מקדם הסעיוף הוא 4 כי לכל צומת יש 4 בנים מכיוון שיש רק 4 פעולות שניתן לבצע.
6. מכיוון שיש 4 פעולות כל מצב יכול לתת לנו לכל היותר 4 מצבים חדשים ולכן מקדם הסעיוף הינו 4. 

הערה: כל מצב חוזר לעצמו אם לא ניתן לבצע עליו את אחת הפעולות (לא כתוב בגרף כי לא רציתי שיהיה מבולגן).

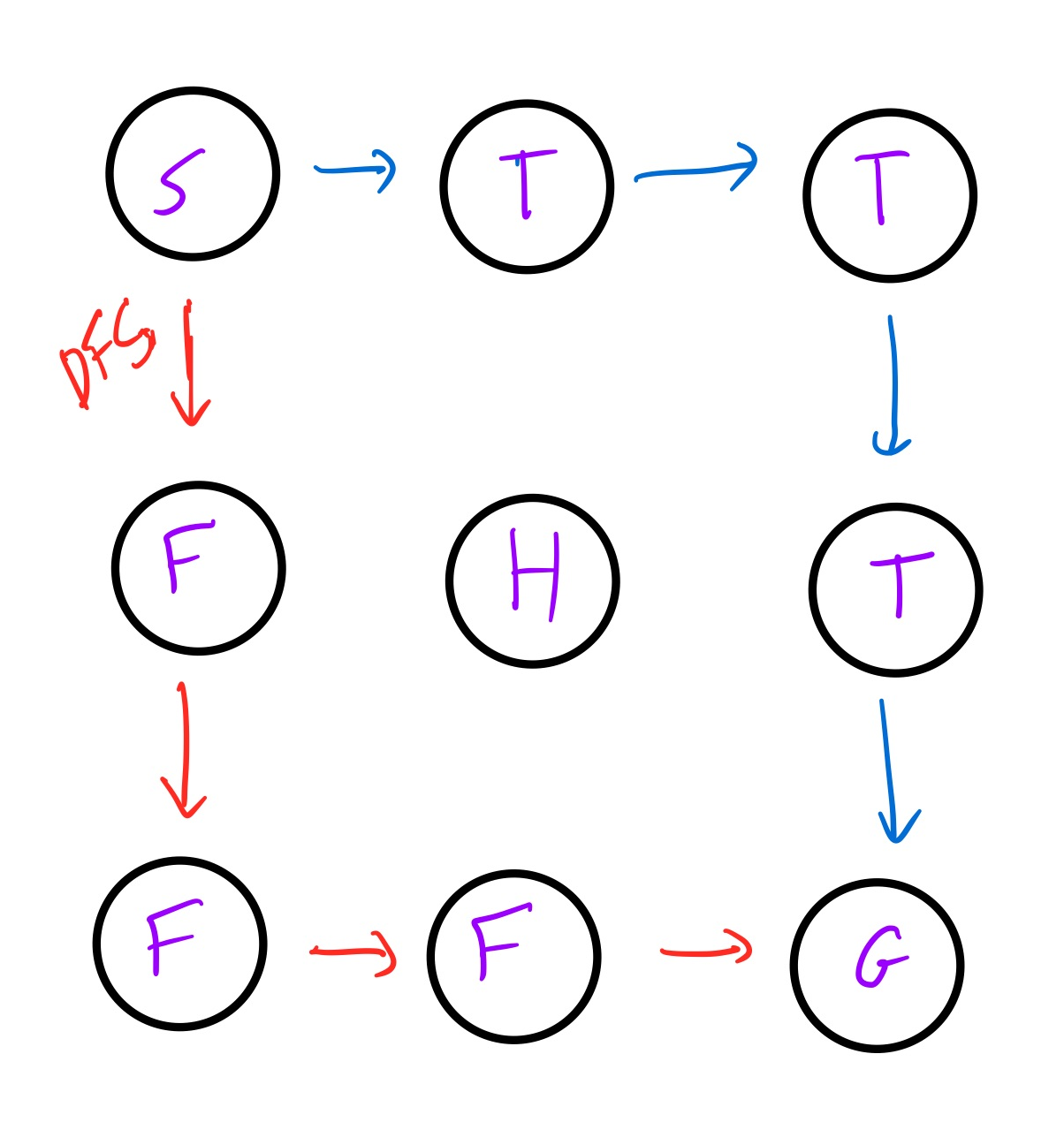
1. במקרה הגרוע ביותר הוא לא יגיע ליעד לדוגמא אם ילך ימינה ואז שמאלה ויחזור על כך (בהנחה שאין portals במשבצות האלו). במקרה הטוב ביותר הוא יגיע תוך 2 צעדים, נניח עבור דוגמא בו יש portal מימין למצב ההתחלה והportal השני נמצא ליד הgoal.
2. 

ניתן לראות בדוגמא הנגדית כי המסלול האדום הוא הקצר ביותר למטרה במונחים של מרחק מנהטן והמחיר שלו זה 122 ולעומת זאת המחיר של המסלול הירוק הוא 102 למרות שהוא יותר רחוק לפי מרחק מנהטן.

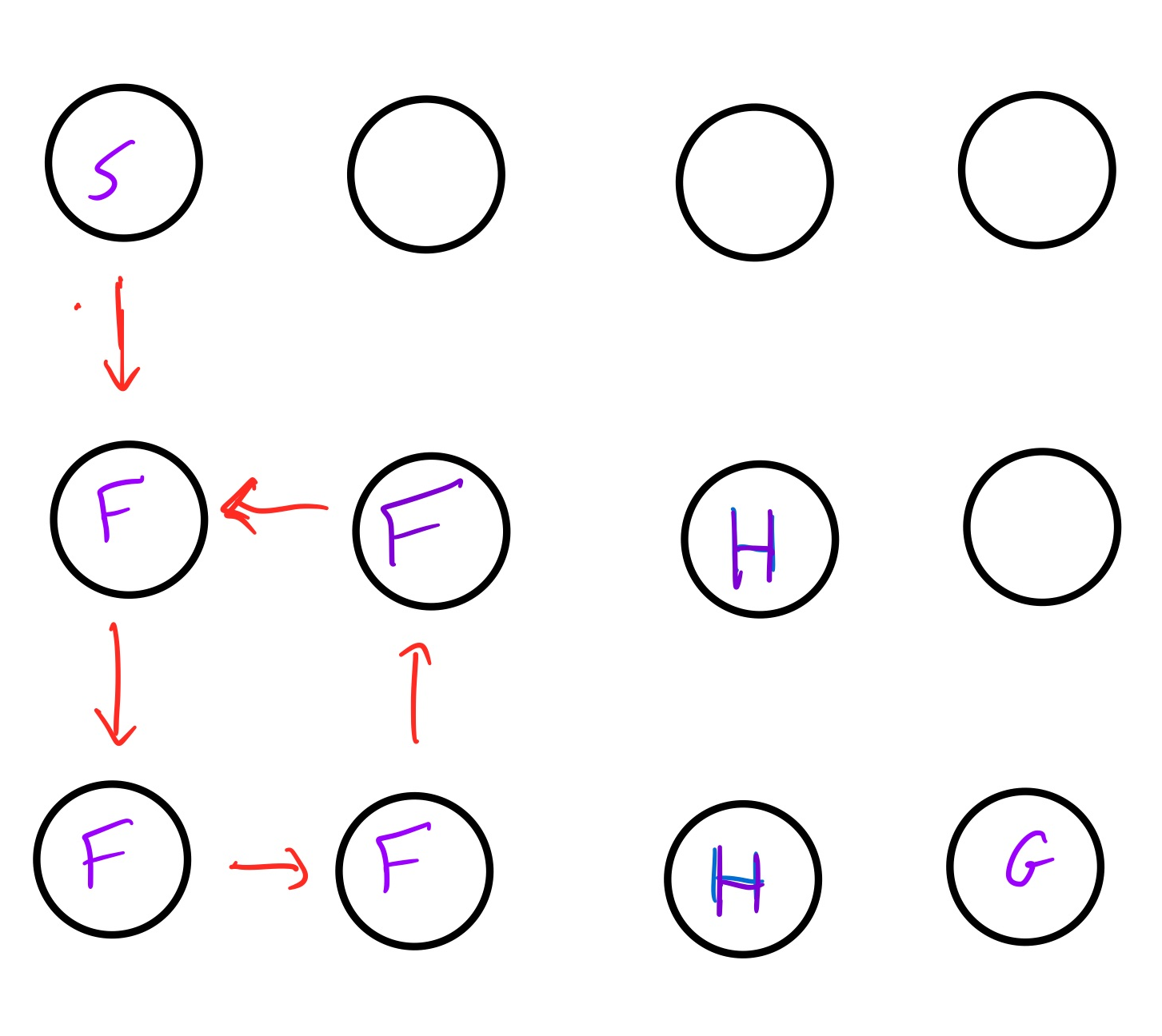
# שאלה 2

1. בהנחה שקיים פתרון (כלומר אנחנו לא חוסמים את המעבר על ידי חורים) האלגוריתם BFS יהיה שלם כי בסופו של דבר נעבור על כל האופציות עד נגיע לפתרון כלשהו. הפתרון לא קביל לדוגמא אם הצומת מטרה שלנו נמצאת 5 משבצות מתחת להתחלה בBFS נלך לכל השכנים הקרובים במקום ללכת ישר למטרה ולכן זה לא פתרון אופטימלי ולכן לא קביל.
2. ההבדל הוא שבBFS על גרף אנחנו שומרים close וopen ולכן לא נעבור באותו צומת פעמיים לכן על ידי הוספת close וopen לעץ שהוא יהיה זהה לBFS על גרף.
3. על מנת לקבל פתרון אופטימלי עם אלגוריתם BFS נגדיר T פונקציה שלוקחת את הגרף G ובין כל שתי צמתים מכניסה דרכם צמתים כאשר e זאת צלע כלשהי וw זאת הפונקציה המחזירה את המחיר של הצלע. בין כל הצמתים החדשים שהכנסנו משקל של כל 2 צמתים עוקבים יהיה 1 בדיוק ולכן סה"כ כשנסכום את הצלעות שהוספנו נקבל בדיוק . וכעת מכיוון שBFS עושה חיפוש לרוחב אנחנו בוודאות נגיע לG עם המסלול הקצר ביותר.
4. מכיוון שאנחנו עובדים עם BFS וזהו חיפוש לרוחב והצומת מטרה נמצאת הכי רחוק מהצומת התחלה נקבל כי יווצרו כל הצמתים

# שאלה 3

1. בוצע
2. האלגוריתם שלם, מכיוון שאנחנו בסופו של דבר נעבור על כל המצבים האפשרים ונגיע לgoal אם זאת האלגוריתם אינו קביל. דוגמא נגדית: 

ניתן לראות כי קיים מסלול שהוא יותר זול (המסלול הכחול).

1. לא כי בDFS (על עץ) אנחנו לא בודקים בclose אם היינו כבר בצומת כלשהי ולכן עלולים להיכנס למעגל, לדוגמא: 
2. 1) נלך למטה וכאשר לא יהיה אפשר לרדת יותר נלך ימינה עד שנגיע לצומת מטרה כלומר סה"כ 2N-2 יפותחו כי לא נספור שוב את צומת ההתחלה והמטרה. ובכל צומת בו נעבור יווצרו כל הצמתים מסביב לאותו צומת (אם ניתן) ולכן סה"כ יווצרו 4N-5 צמתים.

2) עם backtracking האלגוריתם יצור את הצמתים באופן עצל ולכן סה"כ יווצרו 2N-1 צמתים ויפותחו 2N-2 כי לא נפתח את צומת המטרה.

1. 1) נשנה את מרחב החיפוש על ידי בחירת פעולות חדשות במקום נגדיר את כלומר כעת יהיו לנו 16 פעולות אפשריות. דבר זה יאפשר לנו ללכת שתי משבצות בפעולה אחת ובכך נצליח להגיע ב צעדים.

2) מקדם הסיעוף החדש הינו 4b. כי ומתקיים כי .

3) סיבוכיות של DFS-L היא ומכיוון שאצלנו העומק הוא וגם אזי כלומר סה"כ נקבל:

4) דוגמא בה מרחב החיפוש החדש יותר טוב היא אם אנחנו בלוח N על N בלי חורים עם אותם ערכים בכל המשבצות וgoal אחד בפינה הימנית למטה. מכיוון שנלך למטה עד תחתית הלוח ואז ימינה עד המטרה. במרחב החיפוש החדש אנחנו נבצע בכל פעולה 2 צעדים קדימה ואילו במרחב הקודם היינו מבצעים צעד אחד בכל פעולה. ולכן יפותחו חצי מהצמתים במרחב החיפוש החדש. דוגמא בה המרחב המקורי עדיף היא אם נפתח יותר מדי צמתים וזה יכול לקרות כי כעת נפתח פי 4 צמתים (מקדם הסעיוף שלנו הוא 16 במקום 4).

# שאלה 4

# שאלה 5

תהיינה שתי יוריסטיקות קבילות h1,h2:

1. 1) לא נכון, הדוגמא הנגדית היא היוריסטיקה האוקלידית עבור מסלול ריבועי כאשר כל צלע בעלות 1 ואנחנו רוצים להגיע מקודקוד 1 לקודקוד ההופכי לו לכן ו ולכן הינם קבילות אבל הסכום שלהם לא קביל כי: ולכן זה סתירה להגדרת הקבילות.

2) נכון, כי ולכן הסכום שלהם יהיה ואז נחלק ב2 ונקבל ולכן h קבילה.

2) 1) לא נכון, כי אנחנו יודעים שעקביות גורר קבילות ולכן לא קבילה גורר לא עקבית והראנו בסעיף הקודם כי לא קבילה ולכן לא עקבית.

2) נכון, כי אם וגם אזי אם נחבר נקבל:

ולכן h עקבית לפי הגדרה.

4) כן, נסתכל על המקרה הראשון בו לא עוברים בportals לכן העלות הכי נמוכה שיש בלוח היא 1 וכדי להגיע מS לG צריך לפחות ולכן נשים לב שזו יוריסטיקת מנהטן כלומר מתקיים (עבור g הכי קרוב לs) ולכן:

אם כן עוברים בportal במהלך המסלול אז במידה ונבחר את המינימום בין מרחק מנהטן למחיר הportal ולכן נבחר את המחיר של הportal במידה והportal תגרום לנו לעבור מעל ל100 צעדים ונקבל כי כי הראנו שבבעית החיפוש שלנו מתקיים: .

5) כן, מכיוון ש: ולכן לפי הגדרה נחלק ל2 מקרים:

מקרה 1: צעד רגיל:

ולפי הגדרת המפה אנחנו יודעים כי ולכן עקבית.

מקרה 2: כאשר הצעד הוא בתוך portal נקבל:

כאשר ולכן אי השוויון מתקיים וזה גורר כי סה"כ נקבל ש עקבית.

# שאלה 6

# שאלה 7

2) מה שעשינו הוא להכפיל את f המקורית בקבוע שהוא חצי כלומר לכל צומת הערך f שלה הוכפל בחצי ולכן מכיוון שאנחנו משווין ביניהם כדי לקבל מינימום כל פעם אנחנו לא נפגע בהשוואה. במילים אחרת, אזי מתקיים גם ולכן ולכן נקבל אותו מסלול וזה גורר כי גם העלות תהיה שווה.

4) יתרון וחסרון של האלג' A-ID \*ביחס ל-\*A:

**יתרון:** הקטנת צריכת זיכרון, כלומר במידה ואנחנו יודעים שיש הרבה מצבים בהם נצטרך לעבור.

**חסרון:** אנחנו עלולים לפתח מצבים שכבר היינו בהם בעבר. לכן נרצה להשתמש בA\* כאשר אין לנו הגבלה על הזיכרון ואילו בA\*-ID כאשר כן יש לנו הגבלה על הזיכרון.

5) יתרון וחיסרון של האלג' epsilon-\*A ביחס ל-\*A:

**יתרון:** נותן לנו יותר גמישות בלפתוח את הצומת הבאה כי יש אפשרות שתהיה לנו יוריסטיקה לא קבילה שקרובה ליוריסטיקה האופטימלית כלומר h\*, כלומר יהיה לנו מבחר יותר גדול.

**חסרון:** אנחנו מוותרים על האופטימליות של A\* לדוגמא: אם המחיר האופטימלי הוא 200 אז עבור אפסילון כלשהו אנחנו יכולים לקבל מחיר שהוא בטווח של 150-250, ולכן אנחנו עלולים לקבל פתרון שהוא 250 וזה פחות טוב.

# שאלה 8

# שאלה 9

1. S={כל הקבצים עם n מילים כך שכל מילה מופיעה בדיוק פעם אחת} זה שקול לכך שיש פונקציה חח"ע ועל שממפה את המילה הi למקום הj. כאשר i היא אחת המילים מתוך n וj הוא המיקום בו נמצאת המילה i.
2. כי S שקולה לקבוצת הפונקציות שממפות כל מילה שונה מקום אחד. וראינו בקורס אחר כי יש בדיוק n! פונקציות כאלה.
3. כן, תהא w1 מילה שאינה במקום הנכון, אזי בהכרח גם המילה שנמצאת במקומה של המילה w1 נמצאת במקום הלא נכון נסמנה w2. ולכן אם נחליף בין w1 וw2 אזי w1 תהי במקום הנכון וw2 אולי כן ואולי לא תהיה במקום הנכון אבל בכל מקרה שיפרנו את המצב ולכן תמיד יהיה מצב שאליו יהיה עדיף לעבור אליו.
4. 1) כן, לפי סעיף 3 הוכחנו כי לא נגיע למצב שבו אין לנו אפשרויות להתקדם. ולכן מכיוון שזהו אותו אלגוריתם מלבד לsideways steps (שגם ככה לא נשתמש בהם) ניתן להסיק כי נקבל אותו תשובה ובהכרח נמצא פתרון.

2) זה יהיה אותו אלגוריתם כי אנחנו לא "נתקע". בנוסף, כל פעם ניתן לתקן מילה אחת או 2 מילים לכל היותר ולכן בשתי האלגוריתמים הם יבחרו תמיד לתקן 2 מילים אם ניתן ואם לא, אז מילה אחת (תמיד יש) עד שנגיע למטרה. ההבדל היחיד שיכול להיות זה שאולי האלגוריתם with sideways steps יעשה פעולה שונה מהאלגוריתם השני מה שיגרום לכך שהקובץ ישתנה "לטובתו" כך שיהיה ניתן בשלב הבא לתקן 2 מילים ולאלגוריתם השני יהיה ניתן לשפר רק במילה אחת.

1. כן, מכיוון שאנחנו תמיד נשפר בכל צעד את הפתרון אז לכל היותר תוך n צעדים נגיע למטרה (הוכחנו בסעיף 3 שתמיד יש מצב שמשפר).