# ממ"ן 11

**אלגוריתמים – סתיו 2020**

**מגיש: ישגב סרנגה**

# שאלה 1 – בעיית השידוך היציב

נפתור את שאלה 1.6 בספר הקורס באמצעות וריאציה על אלגוריתם הזיווג היציב שהוצג בפרק 1 של הספר.

רעיון האלגוריתם:

את תפקיד הגברים באלגוריתם הזיווג היציב ימלאו הספינות, ואת תפקיד הנשים ימלאו הנמלים. רשימת ההעדפות של כל אחת מהספינות תהיה רשימת הנמלים לפי הסדר בו היא עוגנת בהם, ואילו רשימת ההעדפות של כל אחד מהנמלים תהיה רשימת הספינות בסדר הפוך של סדר העגינה באותו נמל (כלומר, החל מהספינה האחרונה שעגנה באותו נמל ועד לספינה הראשונה שעגנה באותו נמל).

באופן זה נפעיל את אלגוריתם הזיווג היציב על הספינות והנמלים, ונקבל זיווג יציב בין ספינות לנמלים. מזיווג זה נבנה את לוחות הזמנים המקוצצים של כל ספינה: הספינה תעגון בכל נמל לפי לוח הזמנים המקורי, עד שתגיע לנמל אליו זווגה ושם תישאר עד סוף החודש.

האלגוריתם:

1. לכל ספינה בנה את רשימת ההעדפות שלה , כאשר הוא הנמל הראשון בו היא עוגנת לפי לוח הזמנים המקורי שלה, הנמל השני בו היא עוגנת, וכך הלאה.
2. *לכל נמל בנה את רשימת ההעדפות שלו , כאשר היא הספינה האחרונה שתעגון בו, הספינה הלפני-אחרונה שתעגון בו, וכך הלאה.*
3. *הפעל את אלגוריתם הזיווג היציב על הספינות והנמלים, כאשר הספינות הן בתפקיד הגברים והנמלים הם בתפקיד הנשים. הפלט המתקבל מהרצת אלגוריתם זה הוא זיווג יציב בין הספינות לנמלים.*
4. לכל ספינה בנה את *לוח הזמנים המקוצץ שלה: נניח כי הנמל שזווג לה הוא . אזי לוח הזמנים המקוצץ שלה יהיה זהה ללוח הזמנים המקורי שלה עד היום בו היא עוגנת בנמל , ומיום זה ועד סוף החודש היא תישאר בנמל זה ולא תצא ממנו.*

נכונות האלגוריתם:

נניח בשלילה כי האלגוריתם מחזיר לוחות זמנים שאינם מקיימים את הנדרש, כלומר שיש ספינה שמגיעה לנמל שכבר עוגנת בו הספינה שזווגה לו לפי האלגוריתם ואמורה לעגון בו עד סוף החודש.

כיוון שהספינה הגיעה לנמל לפני הנמל שזווג לה לפי האלגוריתם (אחרת הייתה נשארת בו ולא מגיעה כלל ל-), הרי שהנמל מדורג גבוה יותר מ- ברשימת ההעדפות של הספינה . כמו כן, כיוון שהספינה הגיעה לנמל לפני הספינה , הרי שהספינה מדורגת גבוה יותר מ- ברשימת ההעדפות של הנמל . הזוג הוא אם כן אי-יציבות ביחס לזיווג שהאלגוריתם החזיר, וזו סתירה לכך שאלגוריתם הזיווג היציב מחזיר תמיד זיווג יציב.

סיבוכיות האלגוריתם:

מכיוון שבניית רשימת ההעדפות של ספינה מסוימת דורשת סריקה של לוח הזמנים המקורי שלה במלואו, ומכיוון שלוח הזמנים של כל ספינה מכיל m ימים, הרי שבניית רשימת ההעדפות של ספינה מסוימת מתבצעת בזמן , ומכאן שבניית רשימות ההעדפות של כל n הספינות מתבצעת בזמן . מנימוקים דומים נקבל שבניית רשימות ההעדפות של כל n הנמלים מתבצעת גם היא בזמן .

לפי טענה 2.10 בעמ' 52 בספר הלימוד, אלגוריתם הזיווג היציב מתבצע בזמן .

בניית לוח הזמנים המקוצץ של ספינה מסוימת מתבצעת בזמן , ולכן בניית לוחות הזמנים המקוצצים של כלל הספינות מתבצעת בזמן .

נקבל כי סיבוכיות האלגוריתם היא , ומכיוון שנתון כי , מתקיים כי . *לכן סיבוכיות הזמן של האלגוריתם הוא .*

# שאלה 2 – הכוונת צלעות

ראשית, אם G אינו קשיר אז ניתן לכוון כל רכיב קשירות בנפרד ועדיין לקבל הכוונה חוקית של כל הגרף. לכן נניח לשם הפשטות כי G קשיר. כמו כן, בגרף מכוון שבו דרגת הכניסה של כל קודקוד גדולה מאפס בהכרח מכיל מעגל (זה נובע מטענה 3.19 בספר הלימוד, עמ' 110).

נבחר קודקוד כלשהו ב-G, ונריץ ממנו את אלגוריתם DFS. אם בעץ ה-DFS המתקבל מסריקה זו אין קשת לאחור, אז ב-G אין מעגל (עמ' 43 במדריך הלמידה) ולכן הגרף אינו ניתן להכוונה. אם יש קשת לאחור, נאמר , אז נכוון אותה כך ש- יצביע על . כעת נריץ מ- את אלגוריתם BFS, וכל קשת לא-מכוונת שניתקל בה במהלך ריצת האלגוריתם נכוון אותה כך שתצא מקודקוד ברמה הקרובה יותר ל- לקודקוד ברמה הרחוקה יותר. לאחר סיום ריצת ה-BFS נכוון כרצוננו את יתר הקשתות שטרם כוונו, אם קיימות כאלה.

דרגת הכניסה של היא לפחות 1 (כי כיוונו קשת אחת אליו), ומכיוון שהגרף קשיר וכל הקודקודים ישיגים מ-, לכולם דרגת כניסה 1 לפחות בשל האופן בו כיוונו את הקשתות במהלך BFS. הכוונת יתר הקשתות הבלתי-מכוונות אינה יכולה להקטין את דרגת הכניסה של קודקוד כלשהו, ולכן בסיום התהליך אכן קיבלנו הכוונה חוקית שבה דרגת הכניסה של כל קודקוד גדולה מאפס.

מכיוון שהאלגוריתמים DFS ו-BFS שניהם רצים בזמן , זהו גם זמן הריצה של התהליך כולו.

# שאלה 3 – בעיית הספיקות 2SAT

רעיון האלגוריתם:

נבנה גרף מכוון שמותאם לנוסחה ϕ באופן הבא: קבוצת הצמתים תהיה קבוצת הליטרלים , ולכל פסוקית מהצורה נוסיף לקבוצת הקשתות את שתי הקשתות , . לאחר מכן נפעיל על גרף זה סריקות DFS כדי לגלות האם קיים ליטרל כלשהו שהוא ושלילתו נגישים הדדית. במקרה כזה הנוסחה ϕ אינה ספיקה. בכל מקרה אחר האלגוריתם יחזיר השמה מספקת בהתאם לתוצאות סריקות ה-DFS.

האלגוריתם:

1. בנה גרף מכוון שמותאם לנוסחה ϕ באופן הבא:
   1. לכל משתנה שנמצא בנוסחה הוסף אותו ואת שלילתו ל-.
   2. לכל פסוקית מהצורה שנמצאת בנוסחה הוסף את הקשתות , ל-.
2. לכל משתנה שטרם קיבל ערך אמת כלשהו (True או False) בצע:
   1. הפעל אלגוריתם DFS החל מהצומת השייך למשתנה זה. שמור במערך עזר את כל הצמתים/משתנים הישיגים ממשתנה זה.
   2. אם שלילתו של המשתנה **אינה** נמצאת במערך העזר שלו:
      1. השם את הערך True לכל משתנה שנמצא במערך העזר, ואת הערך False לכל שלילה של משתנים אלה.
   3. אם שלילתו של המשתנה **כן** נמצאת במערך העזר שלו:
      1. הפעל אלגוריתם DFS החל מהצומת השייך לשלילת המשתנה . שמור במערך עזר (נוסף) את כל הצמתים/משתנים הישיגים משלילתו של משתנה זה.
      2. אם המשתנה **אינו** נמצא במערך העזר (של שלילתו):
         1. השם את הערך True לכל משתנה שנמצא במערך העזר, ואת הערך False לכל שלילה של משתנים אלה.
      3. אם המשתנה **כן** נמצא במערך העזר (של שלילתו), החזר False (הנוסחה אינה ספיקה).
3. החזר את ערכי האמת של כל המשתנים ב-. זו ההשמה שמספקת את הנוסחה ϕ.

נכונות האלגוריתם:

נשים לב כי פסוקית מהצורה שקולה לוגית לגרירות ו- . גרירה היא יחס טרנזטיבי, ולכן אם נתרגם את כל הפסוקיות לגרירות ו"נצמצם" אותן לפי היחס הטרנזטיבי, ייתכן שנקבל בסופו של דבר גרירה מהצורה או עבור משתנה כלשהו. אם קיבלנו את שתי הגרירות האלה גם יחד עבור אותו משתנה, אז הנוסחה כולה אינה ספיקה (מכיוון שאין השמה של שתקיים את ואת בו זמנית).

כל קשת בגרף שבנינו עבור הנוסחה ϕ מייצגת גרירה בין שני משתנים, ולכן מסלול (מכוון) בגרף זה מייצג שרשרת גרירות, לפיכך המשתנה שמופיע בתחילת המסלול גורר את המשתנה שמופיע בסוף המסלול. הרצת DFS ממשתנה מסוים מוצאת את כל המשתנים הנגררים ממנו, ולכן השמת ערך אמת מסוים במשתנה ההתחלתי גוררת השמת אותו ערך אמת במשתנים הנגררים ממנו (וערך אמת הפוך בשלילות של משתנים אלו).

אם בין המשתנים הנגררים לא נמצאת שלילת המשתנה ההתחלתי, אז הכל טוב ויפה, והאלגוריתם יחזיר השמה חוקית לנוסחה. אחרת – יש צורך בהרצת DFS החל משלילת המשתנה ההתחלתי כדי לבדוק אם המשתנה ההתחלתי נגיש ממנו בכיוון ההפוך. אם כן – אז הנוסחה אינה ספיקה והאלגוריתם יחזיר False כנדרש. אחרת – ההשמה שהאלגוריתם קובע לכל משתנה היא חוקית ומספקת את הנוסחה, ולכן האלגוריתם נכון.

סיבוכיות האלגוריתם:

בניית הגרף נעשית על ידי מעבר יחיד על הנוסחה ו"מילוי" הקבוצות ו-, לכן סיבוכיות שלב זה היא .

בשלב השני מבצעים לכל היותר פעמיים את אלגוריתם DFS עבור כל צומת ב-V. זמן הריצה של DFS הוא .

בסך הכל נקבל שזמן הריצה של האלגוריתם בכללותו הוא .

# שאלה 4 – מסלולים מזעריים דרך קודקודים מעודפים

רעיון האלגוריתם:

נבנה מתוך הגרף הנתון G גרף מכוון חדש G' באופן כזה שכל מסלול מ-s ל-t ב-G' מייצג מסלול מ-s ל-t ב-G שעובר בשני קודקודים בדיוק של U. נפעיל על G' את אלגוריתם BFS כדי למצוא את המסלול הקצר ביותר בין s ל-t (ב-G'), ונתרגם מסלול זה למסלול מ-s ל-t ב-G, וזה יהיה המסלול המבוקש.

האלגוריתם:

1. בהינתן הגרף המכוון בנה גרף מכוון חדש באופן הבא:
   1. הוסף את הקודקודים ל-.
   2. לכל קודקוד מועדף הוסף את הקודקודים ל-.
   3. לכל קודקוד שאינו מועדף ואינו אחד הקודקודים הוסף את הקודקודים ל-.
   4. לכל קשת כך ש- הוסף את הקשת ל-.
   5. לכל קשת כך ש- הוסף את הקשת ל-.
   6. לכל קשת כך ש-ו- הוסף את הקשתות ל-.
   7. לכל קשת כך ש-ו- הוסף את הקשתות ל-.
   8. לכל קשת כך ש- הוסף את הקשת ל-.
   9. לכל קשת כך ש-הוסף את הקשת ל-.
   10. לכל קשת כך ש-הוסף את הקשת ל-.
2. הפעל את אלגוריתם BFS על החל מהקודקוד . ברגע ש-BFS מגלה לראשונה את הקודקוד , הפסק את ריצתו. אם BFS סיים את ריצתו מבלי שהגיע לקודקוד , החזר False (לא קיים ב- מסלול מ-s ל-t שמבקר ב- בדיוק פעמיים).
3. את המסלול מ-s ל-t ש-BFS מצא ב- תרגם חזרה למסלול מ-s ל-t ב- על ידי החלפת כל ל- והחלפת כל ל-. זהו המסלול המבוקש – החזר אותו.

נכונות האלגוריתם:

לכל קודקוד מועדף אנחנו מוסיפים ל- שני עותקים שלו, , אשר מייצגים "ביקור ראשון ב-" ו"ביקור שני ב-" בהתאמה. כמו כן, לכל קודקוד לא-מועדף אנחנו מוסיפים ל- שלושה עותקים שלו, , אשר מייצגים "לפני ביקור ראשון ב-*", "בין ביקור ראשון לביקור שני ב-", ו"אחרי ביקור שני ב-", בהתאמה.*

*לאחר מכן, לכל קשת ב- אנחנו מוסיפים ל- קשת אחת או שתיים בהתאם לסוג הקודקודים שהקשת המקורית מחברת. למשל, אם יש קשת ב- שיוצאת מקודקוד לקודקוד לא-מועדף כלשהו, אז ברור שזו תחילת המסלול ושעוד לא ביקרנו בקודקוד מועדף ב-, ולכן נוסיף את הקשת*  ל-. דוגמה נוספת: אם יש קשת שיוצאת מקודקוד לא-מועדף ונכנסת לקודקוד מועדף, אז יכול להיות שזהו ביקור ראשון או שני ב-, ולכן נוסיף שתי קשתות מתאימות – אחת עבור המקרה שזהו ביקור ראשון ב-, והשנייה עבור המקרה שזהו ביקור שני ב-. באותו אופן מוסיפים את שאר הקשתות ל-.

בניית הגרף בצורה זו מבטיחה שכל מסלול ב- מ- ל- עובר בדיוק פעם אחת ב- ואז בדיוק פעם אחת ב-, בעוד שאר הקודקודים במסלול אינם יכולים להיות קודקודים מועדפים. אם נחליף כל קודקוד ב- בקודקוד המקורי שלו ב-, נקבל מסלול ב- שמקיים את הדרישה .

אלגוריתם BFS מבטיח לנו שהמסלול מ- ל- ב- הוא הקצר ביותר האפשרי בין ל- (ב-). לכן אם נחליף כל קודקוד במסלול זה בקודקוד המקורי שלו ב-, נקבל את המסלול הקצר ביותר האפשרי בין ל- ב-, וזה עונה על הדרישה השנייה בשאלה. ולפיכך האלגוריתם אכן מחזיר את המסלול המבוקש בשאלה.

סיבוכיות האלגוריתם:

בניית הגרף מבוצעת על ידי מעבר יחיד על איברי הקבוצות ו-, לכן זמן הריצה של הבנייה הוא . לאחר מכן מפעילים את אלגוריתם BFS שזמן ריצתו הוא גם כן . אורך המסלול מ-s ל-t המתקבל מריצת BFS הוא , ולכן התרגום שלו למסלול המקביל ב- מתבצע בזמן . בסך הכל נקבל שזמן הריצה של האלגוריתם המוצע כאן הוא .