ממ"ן 11

אלגוריתמים

# שאלה 1

**בעיית השידוך היציב**

## א. תיאור דרך כללית:

נשתמש באלגוריתם הזיווג היציב שמתואר בפרק 1 בספר בהרחבה. כאשר נבצע "התאמה" של הספינות לגברים ואת הנמלים נתאים לנשים. נרצה להתאים כל ספינה לנמל כאשר כל התאמה מייצגת ספינה שהגיע לנמל ונשארת שם עד סוף החודש לעבודות תחזוקה בנמל.

מספר הנמלים והספינות הינו זהה (ושווה לn) כנדרש באלגוריתם הזיווג היציב (הרי לא יתכן 2 ספינות בנמל אחד – בדומה לפוליגמיה בזיווג היציב). אנחנו נסמן את ספינות כ S ואת הנמלים בP. כאשר לכל נמל יש רשימת העדפות ולכל ספינה יש רשימה העדפות שנפרט כאן.

**העדפות:**

ספינה: לכל ספינה נגדיר את העדפה של ספינה לנמל p1 על פני נמל p2 רק ורק אם נמל p1 אכן נמצא בלוח הזמנים של s (ספינה) לפני נמל p2 (כאשר p2 וp1 הינם נמלים), כך שהנמל הראשון אליו מגיע הספינה נמצאת בעדיפות גבוה יותר, הנמל השני בעדיפות שנייה וכו'

נמל: לכל נמל אנחנו נגדיר את העדפה של נמל לספינה s1 על פני ספינה s2 רק ורק אם ספינה s1 אכן נמצא בלוח הזמנים של p(נמל) לפני ספינה s2 (כאשר s2 וs1 הינם ספינות), כך שהספינה שצריכה להגיע אחרונה לנמל p – תהיה בעדיפות הראשונה, הספינה השנייה בעדיפות שניה וכו'

### שלבי בניית האלגוריתם:

*-ראשית נבנה רשימת העדפות של הספינות מתוך הלוח זמנים המקורי*

*לכל ספינה si (כאשר i הינו בין n ל1) נבצע סריקה של הלוח זמנים המקורים מ1 עד m (ימים) כאשר כל נמל שמופיע בלוח זמנים יכנס לסוף הרשימה (כמו שהצגנו ברשימת העדפות מעלה).*

*-נבנה רשימת העדפות של הנמלים מתוך לוח זמנים המקורי*

*נרצה לעבור בלולאה על m הימים של החודש אבל בסדר יורד (בגלל העדפות הפכיות כמו שציינו מעלה) כלומר נרוץ מ m ונסיים ב1 (נסמנו כמצביע day) לאחר מכן, נעבר על כל הספינות בסדר יורד גם כן מn עד 1 (כאן נסמנו כs כספינה) ובשלב האחרון, נבדוק האם ברשימת ההעדפות המקורית הספינה s הינה עוגנת בנמל p (p כלשהו) ביום day (המצביע שלנו) – במידה וכן אז נכניס לסוף רשימת העדפות של p נמל את s ספינה. ואם לא, נמשיך לאיטרציה הבאה של הלולאה.*

*-נריץ את האלגוריתם הזיווג היציב*

*בהתחלה הספינות והנמלים אינם משוייכים זה לזה (לפי האלגורתם של הזיווג היציב)*

*נרוץ כל עוד קיים ספינה s שהיא םפנויה (עדין לא הציעה נישואין לנמל p כלשהו)*

*הספינה s זו תציע נישואין לנמל הראשון ברשימת העעדפות (לפי העדפות שציינו מעלה) והכי חשוב שלא הציע לו בעבר נצביע עליו ונסמן אותו בp (נמל)*

*במידה ואותו p הינו כבר מארוס לספינה אחרת s’ (ספינה) אבל s גבוה יותר (בעדפות) אזי האירוסין עם s’ מבוטלים (כנדרש בזיווג היציב) וישר לאחר מכן s וp יתארסו (בשעה טובה ומוצלחת) ובמידה וp אינו מאורס אז הוא כמובן יסכים להצעה וp וs יתארסו גם כן (מזל טוב), במידה ו2 התנאים הנ"ל אינם מתקיימים אז לא יבוצע שום חגיגת אירוסין ואנחנו נצטרך לעבור לנמל הבאה ברשימה (העדפות).*

*בסוף הלולאה כל האירוסין יהפכו לנישואין והשידוך/זיווג המוצלח יסתיים.*

## ב. הוכחת נכונות

כדי להתאים את הנוכחות של האלגורתם שציינו כמו לספר אנחנו חייב לוודא שאכן מתקיימים התנאים הדרושים (בתיאום מלא לספר).

1. רשימת ההעדפות ל כל נמל מופיע כלל הספינות רק פעם אחת בלבד ומתקיים שכל ספינה עוגנת בכל נמל בדיוק פעם אחת (לא רוצים מקרים מצערים חס וחלילה)
2. אותו דבר לגבי ספינה (ברשימת של כל ספינה מופיע כל הנמלים)
3. בסוף (בשלב האחרון שלנו – שורה אחרונה) קשר הנישואין (שהפך מאירוסין לנישואין\*) הוא קשר מחייב והספינה תישאר בנמל עד לסוף התקופה (סוף החודש) לעבודות התחזוקה כנדרש

לאחר מכן נצטרך להוכיח כי מתקיים האילוץ שבו 2 ספינות לא יהיו ביחד באותו נמל p (במילים אחרות שאם s ספינה ו p נמל אכן נשואים לא תגיע שום ספינה בין הימים m לday.

1. נניח בשלילה ש s וp הינם נשואים ש s מגיע לp ביום day וספינה s’ (אחרת) מגיע ביום שבין m לday.
2. כלומר נמל p שהיה ברשימה של s’ במקום גבוה מאשר p ברשימה של s ועל כן שp יהיה צריך לבחור ספינה מסוימת הוא כמובן יעדיף את s’ על פני s וזו סתירה מוחלטת לכך ש p וs אכן נשואים - משל

## ג. ניתוח זמן ריצה

ננתח את זמן הריצה בהתאם לפעולות שנדרש לעשות באלגוריתם זה:

1. בניית/חישוב רשימת העדפות (-
2. ישנן n ספינות ולכן בניית כל הרשימות היא פעולה שלוקחת Θ(m\*n)
3. באופן דומה גם בניית רשימת העדפות הנמלים היא פעולה בסיבוכיות Θ(m\*n)

לאחר כל לולאה במהלך האלגוריתם נבחן זוג אחד של נמל וספינה שלא נבחן קודם לכן (בין שזה זוג חדש כאשר נמל פונה לספינה פנויה ובין שזה זוג חדש שנבחן בהשוואה לזוג קיים כבר). יש סה"כ n^2 זוגות אפשריים, ולכן מספר הלולאות חסום מלמעלה עם O(n^2)

כמובן שהחסם העליון שמצאנו בסעיף האחרון הוא קטן אסימפטומטית מאשר החסם הצמוד שמצאנו בסעיפים הקודמים (כיוון שמתקיים m>n וכתוצאה מכך כמובן m\*n>n^2), ולכן הוא מוכל בתוך הסיבוכיות:

כלומר בשורה התחתונה סיבוכיות האלגוריתם הינה

# שאלה 2

**הכוונת צלעות**

בגרף G (גרף מכוון) שלכל צומת יש דרגת כניסה שהיא גדולה מאפס מכיל בתוכו מעגל כאשר אנחנו מתחילים בצומת מסוים לצורך הדוגמה צומת v ואנחנו עוברים במסלול (u,v) לצומת u נוכל לעשות זאת עבור כל צומת בגרף G בגלל שלכל צומת יש דרכת כניסה ולכן שנעשה את התהליך n+1 פעמים (|V|=n) .לפי עיקרון שובך היונים (מבדידה) ישנה צומת שעברנו בו פעמיים לפחות כלומר, המסלול שבו נענו מכיל מעגל. (2 משפטים)

לאחר שהוכחנו שהגרף המכוון G מכיל מעגלים אז אנחנו גם יכול להגיד שגם הגרף הלא מכוון המקורי חייב להכיל מעגלים כי אם לא היה מעגלים בגרף המקורי אז גם לא יהיו מעגלים בגרף המכוון בשונה ממה שהוכחנו. נוכל להגיד גם שכל רכיב קשירות בגרף המקורי יכיל מעגל בגלל שאם יש רכיב קשירות שלא מכיל מעגל אז נוכל להתייחס אליו כתת גרף ולהגיע לאותה סתירה בדיוק. (2 משפטים)

נסרוק את הגרף ובכל מעגל שאנחנו ניתקל לבחור כיוון יחיד למעגל )במילים אחרות כל צומת במעגל מצביע על הצומת הבא( מה שבעצם מבטיח שלכל צומת שהיא חלק מהמעגל יהיה חייב להיות דרגת כניסה מעל אפס. מה שנשאר עכשיו זה לבדוק שהצמתים שלא נמצאים במעגל (מסלול שקצה אחת הוא במעגל כל קשת במסלול כזה תכוון החוצה מהמעגל כלומר הצומת מהמעגל יכנס לצומת מהמסלול. שיכנס לצומת הבא במסלול כך גם הבטחנו שהצמתים במסלול יהיו עם דרכת כניסה גדולה מאפס כמבוקש. (3 משפטים)

= סה"כ 7 משפטים (עד 8 משפטים) כנדרש

# שאלה 3

**בעיית הספיקות**

## א. תיאור דרך כללית:

הרעיון המרכזי הוא להמיר את הפסוקית הלוגית לקשרי גרירה ולהכניס אותה לגרף מכוון. נבצע השמות על רכיבי קשירות נפרדים תוך כדי שויידיאנו שלא קיבלנו סתירה כלל.

### שלבי בניית האלגוריתם:

בשלב הראשון נבנה גרף מכוון כך שלכל ליטרל x ניצור (V קודקודים) ולכל פסוקית מהצורה נירצה ליצור את הקשת לפי השיקלות הלוגית עבור הפסוקים p,q :

כעת בשלב השני, כל עוד קיים (קודקוד) לא עבר השמה נבחר קודקוד ונריץ DFS (כנדרש במצגת בשיעור) נסמנו בK כעת אם לא נמצא כלל ב אזי עלינו לבצעה את ההשמה T לכל קודקוד בקבוצה שאכן נמצאה ו F לכל הקודקודים הנגדים שלהם . במידה ולא, נסרוק DFS על ונסמן אותה הפעם בR. אם לא נמצא בסריקה נבצע את ההשמה (כנ"ל) אחרת return false.

בשלב האחרון, בסיום ההשמה מוצלחת (בא אנחנו לא קיבלנו כלל סתירות) return true (מצב אמת).

## ב. הוכחת נכונות

בכל איטרציה על צומת v שאין השמה אנחנו מבצעים סריקה DFS כדי למצוא את הצמתים ברכיב הקישורת שלו, במידה ו אכן נמצא כבן של אז נקבל סתירה במסלול הפסוקי. כעת ובדגש המסלול ההפוך עשוי ויכול להתקיים במידה וקיים מסלול מ שאינו מגיע כלל ל ההשמה שלו לא בהכרח מבוילה כאן לסתירה. בגלל זה אם החלק הראשון אכן הופרך עלינו להריץ DFS על ונבדוק האם הוא אכן בן של

במידה והוא כן הבן שלו אז הצמתים נגישים ובכל השמה נקבל סתירה return false

במידה והוא לא הבן שלו אז נבצע השמנה לכל הבנים של ונמשיך בלולאה(איטרציה) כאשר בסופו אם הגענו למצב שלכל צומת יש השמה (ולא קיבלנו שום סתירה) אז return true.

## ג. ניתוח זמן ריצה

בניית גרף G מתבצעת ע"י סריקה של n המשתנים -

בכל ריצה בלולאה על כל הצמתים (שעוד לא בוצע בהם השמה) במקרה הקיצוני/גרוע ש אכן הבן אנחנו נבצע סריקה DFS פעמיים על כל איטרציה בלולאה. סימון בk את מס' הקודקודים אותם נרוץ בלולאה כאשר n לא קטן מk מבצע 2k השמה לליטרלים.

מספר הקוקדוים שאנחנו מבצעים להם השמה הינו קבוע כלומר לא ייתכן שנבצע השמה פעמים לכן אנחנו על

כלומר בשורה התחתונה סיבוכיות האלגוריתם הינה

# שאלה 4

**מסלולים מזערים דרך קודקודים מועדפים:**

## א. תיאור דרך כללית

נחלק את המסלול ל5 שלבים (בדומה לבניית אוטומ-סופי-דטרמינסטי כנלמד)

כאשר x קבוצת קודקודים מתוך V שאינם קודקודים מועדפים (ללא s,t) - X = V \ (U∪{s,t})

נרצה להוכיח בצורה קונסט' בניה של גרף מכוון G’ שמורכב מ5 השלבים(הנ"ל) כאשר לבסוף נמצא את המסלול המינימלי באמצעות סריקת BFS (המוצגת במצגת בשיעור).

### שלבי בניית האלגוריתם:

בשלב הראשון נגדיר X = V \ (U∪{s,t}) ולאחר מכן, נבנה G(V,E) מכוון חדש G’(V’,E’) בדרך הבאה:

1. לכל קודקוד מועדף
2. את s.t נעתיק אותם לגרף החדש G’
3. לכל קודקוד ניצור בגרף G’ את (i=1,3,5) לפי
   * + 1. *לכל קשת ,*
       2. *לכל קשת ,*
       3. *לכל קשת ,*
       4. *לכל קשת ,*
       5. *לכל קשת ,*
       6. *לכל קשת ,*
       7. *לכל קשת ,*
4. נריץ סריקה BFS על הגרף החדש החל מs כאשר בהופעה הראשונה של t בסריקה המסלול מs עד לאותו t הוא המסלול הקצר ביותר שמקיים את התנאים (אם לא נמצא t בסריקה אז לא קיים מסלול כזה כלל)

## ב. הוכחת נכונות

### 1.

*בזכות תיוג הקודקודים מU והגדרת הקשתות – אנחנו בעצם אוכפים את המעברים בין השלבים השונים – דילוג על שלבים 2 ו4 לא (בהתאמה להגדרה הפורמלית של אוטו'-דטרמינסטי)*

*ג.בהתאם למעברים (מעלה- לפי סדר מספרי)*

* + - 1. *לכל קשת ,* -> *מעבר מקודקוד התחלתי s אל איבר אחר בשלב 1*
      2. *לכל קשת ,* -> *מעבר מקודקוד התחלתי s ישירות לשלב 2*
      3. *לכל קשת , -* -> *מעברים משלב 1 ל2 ומשלב 3 ל4*
      4. *לכל קשת ,* -> *מעבר משלב 2 ל3 ומשלב 4 ל5*
      5. *לכל קשת ,* -> *מעבר משלב 2 ל4*
      6. *לכל קשת ,* -> *מעבר משלב 4 לקודקוד הסופי t*
      7. *לכל קשת ,* -> *מעבר משלב 5 לקודקוד הסופי t*

1. נריץ סריקה BFS על הגרף החדש החל מs כאשר בהופעה הראשונה של t בסריקה המסלול מs עד לאותו t הוא המסלול הקצר ביותר שמקיים את התנאים (אם לא נמצא t בסריקה אז לא קיים מסלול כזה כלל) -> כל מסלול בגרף החדש אשר מסתיים בשלב 5 מקיים את התנאים (מתקבל עבור אוטו')

### 2. המסלול הקצר ביותר

השימוש בסריקת BFS מבטיח לנו קבלה של המסלול הקצר ביותר (הגדרה) בגלל שהסריקה מתבצעת בשלבים כאשר הקודקוד הקודם לא יופיע בשלב קדימה ממנו.

### 3. המסלול שמצאנו קיים בגרף G (המקורי)

על ידי זה שאנחנו מסירים את התיוגים של הקודקודים ואת הסימונים מהקשתות (בהתאמה להומורפיזם-אוטו') אנחנו נקבל מסלול תקין בגרף G (המקורי)

## ג. ניתוח זמן ריצה

ראשית בניית הגרף החדש G’ מתבצע על ידי שאנחנו עוברים על הקודקודים והקשתות של G פעם אחת (לפי הגדרה מעלה) כלומר . שנית, סריקה BFS מעל G’ כלומר

כלומר בשורה התחתונה סיבוכיות האלגוריתם הינה