**ממן 11 – אלון גולדמן**

**שאלה 1**

**תיאור האלגוריתם**:

נשתמש באלגוריתם הזיווג היציב G-S , על בסיס רשימת העדפות של ספינות ונמלים. האלגוריתם G-S ירוץ כרגיל, ונראה שהוא מקיים את התנאי הנוסף בשאלה (תנאי ◊), שאסור ל2 ספינות להיות יחד באותו הנמל באותו היום.

**רשימת ההעדפות:**

נסמן את קבוצת הספינות בS, והם יהיו ה**גברים** באלגוריתם לזיווג יציב. רשימת ההעדפות של כל ספינה תהיה לפי הלו"ז הנתון שלה – היא מעדיפה את הנמל שהיא מגיעה אליו ראשון, לאחר מכן את הנמל השני וכו'. (נתעלם מהימים שהספינה נמצאת בים).

נסמן בP את קבוצת הנמלים. הם ייצגו את **הנשים** באלגוריתם לזיווג יציב. רשימת ההעדפות של הנמלים תהיה בסדר הפוך לסדר שבו הספינות מגיעות אליהם.

דוגמה ממחישה:

אם הלו"ז של ספינה X הוא:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ביקור בP3 | ביקור בP1 | שייט בים | ביקור בP2 | שיים בים |

אז רשימת ההעדפות של Sx היא:

1. P3
2. P1
3. P2

אם לנמל מסויים Y, סדר ההגעה של הספינות הוא:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ספינת S2 | ספינת S3 | יום חופש | יום חופש | ספינת S1 |

אז רשימת ההעדפות של Py היא:

1. S1
2. S3
3. S2

בניית רשימת ההעדפות היא פשוטה – עוברים על הלו"ז של כל ספינה וספינה, ומעדכנים את רשימת ההעדפות של הספינות. לאחר מכן עוברים לפי ימים בחודש, ובונים (מהסוף להתחלה) את רשימת ההעדפות של הנמלים. מתעלמים מימים בהם הספינות בים הפתוח.

**האלגוריתם**:

נריץ את האלגוריתם **גייל-שייפלי** ללא שינוי. (מלבד כמובן שינוי אנשים לספינות, ונשים לנמלים).

הזיווג שיתקבל משדך לכל ספינה את הנמל שכשהיא מגיעה אליו בלו"ז שלה, היא צריכה לעצור בו לטיפולים.

**הוכחת נכונות:**

מלבד תנאי ◊, אין הבדל בין האלגוריתם הזה לאלגוריתם הרגיל. ולכן נקבל מיון יציב ומושלם. נוכיח שגם תנאי ◊ מתקיים:

נניח בשלילה שיהיה יום אחד שבו יש 2 ספינות באותו הנמל. בחודש רגיל (ללא עצירה לתחזוקה), אין התנגשות בין הספינות בנמלים. הלו"ז הראשוני הוא נתון, ונתון שהוא מקיים את תנאי ◊. לכן, הפרה של ◊ יכולה להקיים רק במקרה ש:

* ספינה א' עצרה בנמל A, לתחזוקה. (ז"א זהו השידוך הסופי שלה).
* ספינה ב' מגיעה לאחר מכן לנמל A.

אם נמל A מעדיף את ספינה ב' – אז השידוך הסופי של ספינה א' לא היה נמל A, וזה לא יכול להיות כי הפעלנו את אלגוריתם הזיווג היציב.

אם נמל A מעדיף את ספינה א', זה אומר שספינה א' מגיעה אחרי ספינה ב', וזה בסתירה לכך שספינה ב' מגיעה אחרי.

קיבלנו סתירה, ולכן תנאי ◊ מתקיים.

הוכחה שתמיד אפשר לבנות לוח זמנים כזה נובעת מהנכונות של אלגוריתם גייל-שייפלי.

**סיבוכיות:**

אין שינוי באלגוריתם גייל-שייפלי, ולכן הסיבוכיות היא O(n^2). אם נצטרך לעבור על הלו"ז של כל אחת מהספינות (באורך m, מספר הימים בחודש), כדי לבנות ממנו רשימת העדפות – אז נצטרך לעבור על העדפות של n ספינות, באורך m, ונקבל שהסיבוכיות היא O(mn)>=O(n^2).

**שאלה 2**

**רעיון האלגוריתם:**

נוכל לקבוע אם ניתן לכוון את הצלעות כנדרש לפי מעגל בגרף. לאחר מכן, נכוון את הקשתות לפי מעגל זה.

**הסבר**:

מעתה והלאה נניח שהגרף קשיר. במידה והוא לא קשיר, ניתן לחלק אותו לרכיבי הקשירות שלו ולהריץ את האלגוריתם על כל רכיב בנפרד.

האלגוריתם מתבסס על טענה 3.19 בספר, שלפיה לכל גמ"ל יש צומת כלשהי שדרגת הכניסה אליה היא 0.

1. נחפש מעגל כלשהו בגרף, ע"י סריקת DFS של הגרף מצומת כלשהו.
2. אם אף אחת מהקשתות שנתקלנו בהן אינה קשת חוזרת, אז הגרף המכוון שיווצר יהיה גמ"ל, ותהיה בו צומת שלא נכנסת אליה אף קשת. נחזיר FALSE.
3. אם נתקלנו בקשת חוזרת – סימן שסגרנו מעגל כלשהו. נכוון את הקשת החוזרת אל הצומת (V) שאליה הגענו.
4. נבצע סריקת BFS מצומת V לשאר הצמתים, ונכוון כל קשת עם כיוון הסריקה. (ז"א אם הגעתי מצומת א' לצומת ב' אז כיוון הקשת יהיה מצומת א' לב').
5. נחזיר את הסידור שהתקבל.

**נכונות**:

מציאת מעגל נובעת מהגדרת קשת חוזרת. טענה 3.19 מוכחת בספר. נוכיח באינדוקציה שבמקרה שמצאנו מעגל, לכל צומת תהיה קשת שמכוונת אליה:

**טענה**: לכל צומת ברמה כלשהי, יש קשת מצומת ברמה שמצביעה אליה.

**בסיס האינדוקציה**: עבור L0 , הקשת החוזרת מצביעה אליו. לאחר מכן צומת זו תצביע לשאר השכנים שלה, שיהוו את רמה .

**צעד האינדוקציה**: נוכיח את נכונות הטענה עבור כלשהו, ונראה עבור .

**הוכחה**: בסריקה BFS, כל האיברים ברמה Li יהיו במחסנית. כאשר נעבור עליהם ונכניס למחסנית את רמה , נצביע על איברי . אחרי שסיימנו לעבור על כל איברי Li, בהכרח יש קשת שמכוונת לכל איברי .

מכאן שהטענה נכונה והאלגוריתם מחזיר את התשובה הנכונה.

**סיבוכיות (לפי המספור הנ"ל):**

1. O(m+n)
2. O(1)
3. O(1)
4. O(m+n)
5. O(1)

בסך הכל קיבלנו O(m+n)

**שאלה 3**

**הרעיון**:

נבנה גרף מכוון שמותאם לנוסחה ϕ, ע"י גרירות לוגיות, ואז ע"י חיפוש בגרף נוכל למצוא את ההשמה הנכונה.

**האלגוריתם**:

בניית הגרף:

כל משתנה והיפוכו יהיו צמתים בגרף. למשל, אם ב ϕ יש את הפסוקית , אז יהיו 4 צמתים: .

הקשרים הלוגים יהיו קשתות מכוונות בגרף. הביטוי שקול לביטויים , ולכן נוסיף לגרף את הקשתות .

אלגוריתם בניית הגרף:

1. עבור כל פסוקית:
   1. צור צמתים בהתאם למשתנים
   2. צור קשתות בהתאם לפסוקית

השמות:

מכיוון שכל קשת מייצגת גרירה לוגית, אם הקשת קיימת זה אומר שאם x=T אז בהכרח y=T. לכן, אם (לצומת כלשהי X) קיימת הקשת , אז x!=T, כי אחרת הדבר יגרור x=T=~x וזאת סתירה. לכן, בקשת כזו x=F.

אם, באותו הגרף, קיימת גם הקשת , אז יש סתירה, כי הראנו שx=F ואז !x = T, וזה גורר x=T, וזה לא יכול להיות. לכן, אם ניתקל במשתנה שיש מסלול ממנו אל ההופכי שלו **ובחזרה** – סימן שאין השמה שתספק את הגרף.

הערה: באלגוריתםDFS נבצע שינוי קטן, כך שאם צומת עברה השמה בעבר, הסריקה לא תיכנס לענף זה.

לכן, האלגוריתם הוא כדלהלן (לאחר שבנינו את הגרף):

1. כל עוד יש משתנים ללא השמה:
   1. בחר צומת כלשהי X, שלא עברה השמה.
   2. בצע סריקה DFS מX. שמור את הצמתים שמתקבלים בקבוצה S.
   3. אם :
      1. בצע סריקת DFS שניה, ושמור את הצמתים המתקבלים בקבוצה S`.
      2. אם , החזר FALSE – אין השמה שמספקת את ϕ.
      3. אחרת, בצע השמה
   4. סמן את כל הצמתים בS כT, ואת כל ההופכיים שלהם כF.
2. החזר הTRUE ואת רשימת ההשמות

**נכונות**:

נגדיר "**לולאה**", כמשתנה שברכיב הקשירות שלו יש מסלול ממנו אל ההופכי שלו, ומההופכי שלו בחזרה אליו.

נוכיח סדרה של טענות:

1. אם יש לולאה ממשתנה כלשהו בגרף אל ההופכי שלו ובחזרה, ϕ לא מסופק.
2. אם לאף משתנה אין קשת ממנו אל ההופכי ובחזרה, אז ϕ מסופק. (לא נובע ישירות מטענה מס' 1!)
3. האלגוריתם שלנו יחזיר תשובה נכונה גם אם לא התחיל את הסריקה מהמשתנה עם הלולאה (אלא איפשהו בסריקת הDFS יש לולאה של משתנה אחר).
4. אין משתנה שכבר הושם, שבסריקה נוספת תתבצע לו השמה שניה
5. למרות שבסריקת DFS לא מבקרים במשתנים שעברו השמה, לא יכול להיות (בשלב 1.3 של האלגוריתם) שבענף שכבר עבר השמה, יהיה ההופכי של המשתנה ממנו התחלנו את הסריקה (לכן אין צורך לבדוק צמתים שכבר עברו השמה).

ההוכחות:

1. הראנו כבר שאם יש קשת ממשתנה להופכי שלו וחזרה, האלגוריתם יחזיר תשובה שלילית.
2. נניח בשלילה שאין אף משתנה שיש לו קשת ממנו להופכי ובחזרה, אבל ϕ **לא מסופק.** זה יכול להירגם בגלל 2 מקרים:
3. נגרמה אי ספיקות בגלל המשתנה שממנו התחלנו את סריקת הDFS (נסמן בX את המשתנה הזה). זה לא יכול לקרות, מכיוון שאי ספיקות תיגרם אם X יגרור את !X, וגם !X יגרור את X. זה סותר את ההנחה שאין לולאה כזו לאף משתנה, ולכן זו **סתירה**.
4. נגרמה אי ספיקות בגלל משתנה אחר, שנמצא בתוך רכיב הקשירות של X. למשל, אם יש מסלול כלשהו מX לY, וגם מX ל!Y. אם יש מסלול מX לY, זה שקול לביטוי , ששקול לפסוקית ששקול ל. קיים , וגם , **בסתירה** להנחה שלנו שאין משתנים עם לולאות.

מכאן, שאם לכל המשתנים אין לולאות, הביטוי מסופק.

1. נשאר רק להראות שהאלגוריתם שלנו **ימצא** לולאה, במידה ויש כזו, ויחזיר תשובה שלילית.
2. אם נתחיל את בדיקת DFS ממשתנה כלשהו X, והוא המשתנה עם הלולאה – נמצא אותו בשלב 3 של האלגוריתם, ונגלה שהוא לולאה בשלב 3.2. נחזיר תשובה שלילית.
3. המקרה השני הוא שיש לולאה, אבל היא לא של X אלא היא קיימת בתוך רכיב קשירות X. למשל, אם בתוך רכיב הקשירות של X יש מסלול כלשהו מY ל!Y, ובחזרה מ!Y לY.

נניח שאנחנו בשלב 1.3 של האלגוריתם – ז"א שמצאנו את !X ברכיב הקשירות של X, ועכשיו אנחנו סורקים DFS מ!X. (בלי הנחה זו, אפשר פשוט להראות פעמיים את ההוכחה, וזה שקול). אם Y וגם !Y בתוך רכיב הקשירות של !X, זה אומר שמתקיים וגם . אם קיים המסלול אז מתקיים גם . קיבלנו , ונחזיר תשובה שלילית.

ז"א שבכל מקרה האלגוריתם יחזיר תשובה שלילית במקרה בו יש לולאה.

1. נניח משתנה כלשהו X שבוצעו השמות בו ובמשתנים שנגישים ממנו. נניח משתנה Y שבוצעה לו השמה. (למשל T). נניח שיש משתנה אחר שנבחר רנדומלית, לאחר מכן, Z, שממנו יש מסלול ל~Y. עלול לקרות השמה שניה, ואז Z=T יגרור ש~Y=T, וזאת סתירה. נוכיח שזה לא יכול לקרות:

אם , אז גם . וזה אומר שבסריקה הראשונה Z כבר עבר השמה, ולכן לא יכול להיות שנבחר אותו להשמה השניה.

1. נניח בשלילה שיתכן מצב כזה. למשל, אם צומת כלשהי X וכל הצמתים הנגישים ממנה עברו השמה. לאחר מכן, עבור צומת אחרת Y, נבצע סריקת DFS. נניח ש~Y נמצא בין הצמתים הנגישים מX, ואנחנו עלולים לפספס אותו אם לא נבדוק גם צמתים שכבר עברו השמה. אבל אם ~Y הוא בין הצמתים שעברו השמה תחת X, אז Y עצמו כבר עבר השמה ולא יתכן שנבחר בו להתחיל את סריקת הDFS. מכאן סתירה להנחת השלילה.

הוכחנו את הנכונות.

**סיבוכיות**:

נסמן:

N – מספר המשתנים ב ϕ

M – מספר הפסוקיות ב ϕ

הגרף שנבנה יכיל 2n צמתים ו2m קשתות. בניית הגרף אורכת זמן קבוע עבור כל צומת/קשת, ולכן לוקחת O(m+n).

בכל סריקה כפולה של הגרף, שמתחילה ממשתנה כלשהו X - כל הצמתים הנגישים מX מקבלים השמה. שינינו את DFS כך שאם משתנה כבר עבר השמה, הסריקה לא תיכנס לתוכו. מכאן שמשתנה יכול לקבל השמה לכל היותר פעם אחת. למרות זאת, יתכן שבסריקה נעבור על משתנה כמה פעמים. למשל, אם ביצענו סריקה בשלב 1.2 באלגוריתם, ומצאנו את ההופכי של המשתנה. (ז"א שלא ביצענו השמה למשתנים שסרקנו בהתחלה). לאחר מכן יתכן שנעבור על צמתים אלו שוב בסריקה אחרת וכד').

מכאן שעבור כל משתנה נבצע סריקת DFS, זמן הריצה הוא . זהו גם זמן הריצה הכולל.

**שאלה 4**

**הרעיון**:

נבנה גרף G' מהגרף המקורי, ולאחר מכן נמצא את המסלול הקצר ביותר ע"י BFS.

**האלגוריתם**:

סימונים:

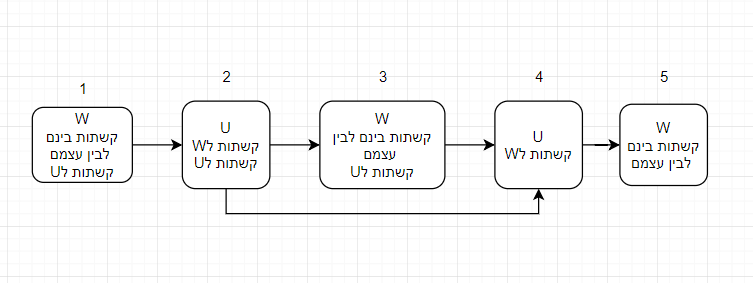
s – נקודת ההתחלה של המסלול

t – נקודת היעד של המסלול

U – רשימת הקודקודים המועדפים, שדרכם חייב לעבור פעמיים בדיוק

W – רשימת כל הקודקודים האחרים, שאינם בקבוצה U.

נבנה את הערף G' בצורה הבאה (המספרים מעל הריבועים הם לשם הנוחות של ההסבר):



נסביר:

נשכפל את הגרף המקורי G 5 פעמים, כך ש:

* בתת גרף 1:
  + יהיו כל הצמתים שבW.
  + יהיו קשתות בין איברי W לאיברי W אחרים
  + יהיו קשתות בין איברי W לבין איברי U שבתת גרף 2.
* בתת גרף 2 יהיו:
  + כל הצמתים שבU
  + קשתות בין איברי U לאיברי W בתת גרף 3
  + קשתות בין איברי U לאיברי U בתת גרף 4
* בתת גרף 3 יהיו:
  + יהיו כל הצמתים שבW.
  + יהיו קשתות בין איברי W לאיברי W אחרים
  + יהיו קשתות בין איברי W לבין איברי U שבתת גרף 4.
* בתת גרף 4 יהיו:
  + כל הצמתים שבU
  + קשתות בין איברי U לאיברי W בתת גרף 5
* בתת גרף 5:
  + יהיו כל הצמתים שבW.
  + יהיו קשתות בין איברי W לאיברי W אחרים

האלגוריתם לבניית הערף G' :

1. עבור על כל הצמתים בגרף:
   1. אם צומת שייכת לU, שכפל אותה לתתי גרפים 2,4
   2. אם צומת שייכת לW, שכפל אותה לתתי גרפים 1,3,5
2. עבור על כל הקשתות בגרף:
   1. אם קשת יוצאת מW לW היא תשופכל לתתי גרפים 1,3,5.
   2. אם קשת מW לU, היא תחבר בין תתי גרפים 1 ל2, ובין 3 ל 4.
   3. אם קשת מU לW, היא תחבר בין תתי גרפים 2 ל3, ובין 4 ל5
   4. אם קשת מU לU, היא תחבר בין תתי גרפים 2 ל4

לאחר מכן נמצא את המסלול הקצר ביותר ע"י סריקת BFS, מצומת s בתת גרף 1 לצומת t בתת גרף 5. נחזיר את המסלול שהתקבל. במידה ולא נמצא מסלול כזה, האלגוריתם יחזיר שלא קיים מסלול כנדרש.

**נכונות**:

נוכיח:

1. כל מסלול כנדרש בG קיים גם בG'
2. כל מסלול כנדרש בG' הוא מסלול גם בG
3. כל מסלול שנחזיר מG' הוא תקני
4. המסלול שמצאנו הוא הקצר ביותר

הוכחות:

1+2: הקשתות שנמצאות בגרף G' הם "תמונת מראה" של הקשתות בG, ולכן עבור כל קשת בגרף G יש קשת שמתאימה לה בG'. מסלול תקני בG נראה ככה:

וככה בדיוק גם נראה מסלול תקני בG'. כל הצמתים בG קיימים גם בG' וכל הקשתות קיימות גם בG'.

1. מסלול תקני מקיים את התנאים הבאים:
   1. יוצא מs
   2. מסיים בt
   3. עובר ב2 צמתים של U בדיוק.

א וב' וודאי מתקיימים, ע"י סריקת BFS. המסלול עובר בדיוק ב2 צמתים של U, מכיוון שאין קשתות פנימיות בתתי הגרפים 2,4. ז"א שברגע שהמסלול "ביקר" בצומת כלשהי בתת גרף 2, אז הוא לא יכול להישאר בתת גרף 2. כך גם לגבי תת גרף 4, ומכאן שהמסלול מבקר לכל היותר 2 פעמים באיברי U.

המסלול לא יכול לבקר **פחות** מ2 פעמים באיברי U, מכיוון שאין קשתות ש,"מדלגות" על צמתי U – אין קשתות בין תתי גרפים 1, 3, 5, ולכן המסלול חייב לעבור בדרך בתתי גרפים 2,4 לפחות פעם אחת בכל תת גרף.

1. זה נובע מנכונות BFS, המוצא את תת הגרף הקשר ביותר.

**סיבוכיות**:

בניית הגרף G' עוברת על כל צומת פעם אחת בלבד, ועל כל קשת פעם אחת בלבד, ומבצעת זמן קבוע של פעולות בכל פעם. לכן זמן הבניה הוא O(n+m).

מציאת המסלול הקצר היא BFS שלוקח זמן לינארי בגודל הגרף.

מספר הצמתים בגרף G' הוא W+u+w+u+w, ומכיוון שW+U=n, נקבל שמספר הצמתים הוא O(3n).

מספר הקשתות גם הוא חסום ע"י O(3m), ז"א בסך הכל זמן הריצה של BFS הוא לינארי.

סך כל זמן הריצה: O(n+m).