תכנון אלגוריתמים - מטלה 3

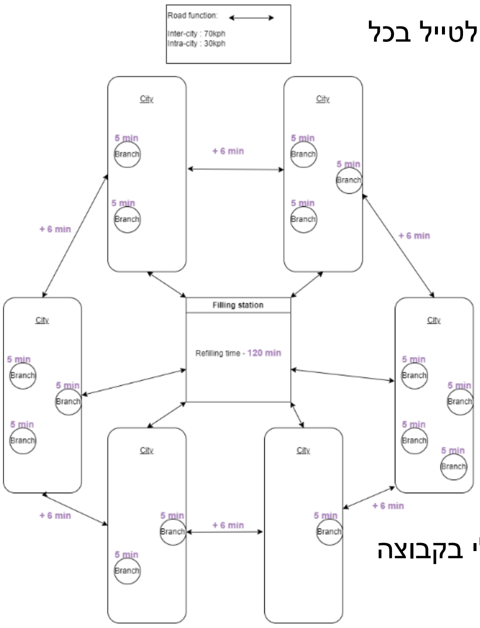
יוליה מושן : 319565610

גיל פסי : 206500936

<u>שאלה 1</u>

חברת מילוי הכספומטים "כספיר בע"מ" רוצה לתכנן את מסלול מילוי הכספות של כל הכספומטים הנמצאים בשירות הבנקים היעיל ביותר בזמן. לשם כך היא הורידה את רשימת מיקומי הכספומטים מהמאגר הרשמי בקישורית
https://data.gov.il/he/dataset/automated-devices/resource/b9d690de-0a9c-45ef-9ced-3e5957776b26
או ע"י הורדה מאתר הקורס. יש לוודא כי במאגר שלכם יש קרוב ל4000 ערכים.
ידוע כי מילוי כספומט עורך 5 דקות. מהירות הנסיעה הממוצעת של רכב המילוי היא 30 קמ"ש בעיר ו70 קמ"ש בין ערים. מעבר בין עיר לעיר תמיד יוסיף 6 דקות לנסיעה (גם אם הן צמודות יחסית). רכב המילוי יכול למלא 100 כספומטים עד שהוא צריך לחזור למחסן (הנמצא בדיוק במרכז הגיאוגרפי של מדינת ישראל) כדי להצטייד במזומן נוסף שם לוקח לו שעתיים למלא את הרכב.
בשאלה זו אנחנו רוצים למצוא את מסלול המילוי הקצר ביותר בזמן.
א. תכנן/י אלגוריתם המחשב כמה זמן יקח למלא את כל הכספומטים אם ידוע כי כולם ריקים אבל הרכב אינו צריך לחזור למחסן כדי להצטייד מחדש. לצורך הסעיף הניח כי קיימת שיטה בשם Road המקבלת כפרמטרים מיקומים של שני כספומטים ומחזירה את המרחק בינהם. נתח/י את סיבוכיות זמן הריצה של האלגוריתם.

א.



בעיה זו שקולה לבעיית איש המכירות (TSP) - מהי הדרך הקצרה ביותר לטייל בכל הנקודות בגרף.

בעיה זו שייכת לסוג Hard Problems ולא קיים פתרון ידוע בזמן ריצה פולינומיאלי.

לכן אציג פתרון בזמן ריצה מעריכי .

<u>האלגוריתם:</u>

בקצרה:

1. ניצור קבוצה של כל תתי הקבוצות מ – E שגודלן הוא בדיוק |V – 1| כלומר שיש להן פוטנציאל ליצור עץ פורס.
2. נסיר מהקבוצה את כל תתי הקבוצות שאינן מייצרות עץ פורס מנוון (= מסלול) .
3. נבחר את תת הקבוצה (שהיא גם עץ פורס) עם המשקל המינמלי בקבוצה .

פסאודו קוד:

```
Algorithm Best-Route (V , E){ //Where V is the vertices' set , E is the edges set
   let potentialRoutes be the getPoetntiaRoutes(E); //Potential spanning trees.
                                                    //All of them has |V - 1| edges
  let routes be ϕ

//Filter none-tiable subsets
  for subset in potentialRoutes
         if(isDegenerated(subset))
         routes += subset
   return weight(min(routes.weights))  //Calculates the weight of the whole
route
}

Function getPoetntialRoutes(E){
   let pr be ϕ //Potential routes
   let perTable be the getAllnPr(|set|) //All permutations' table

   for i to len(perTable)
      let currentSet be ϕ
      for j to len(perTable[0])
         if(perTable [i][j]) // 1 = true , 0 = False
            currentSet += set[i] //Append this edge

      pr += currentSet;
   return pr
}
```

```
Function getAllnPr(n , r , current , permutations){
   if(len(current) = n AND onesCount(current) = r)
      permutations += current

   if(len(current) < n)
      getAllnPr(n , r , current += "0" , permutations )
      getAllnPr(n , r , current += "1" , permutations )

}

Function isDegenerated (subGraph , V){
   do{
      let e ← subGraph.removeStart()
         // If this vertex does not exists in the group , that means
         //It is already taken by another edge.
         if( NOT V.contains(e.src))return false;
         V.remove(subGraph.src)
   }while(NOT subGraph.isEmpty())
  return true;
}
```

סיבוכיות:

א. מציאת כל העצים המנוונים: $\theta(2^{|V|^2}) = \theta(2^{|E|})$ שכן הרקורסיה תגיע לכל האפשרויות (קבוצת החזקה) .

מאחר ומדובר בקליקה $|E| = |V|^2$

ב. סינון גרפים לא מנוונים :סריקה ליניארית של כלל תתי הגרפים בקבוצה.

נזכיר כי הקבוצה המדוברת היא אוסף הפרמוטציות של $|V| - 1$

הקשתות (מדובר בעץ) מעל $|V|(|V| - 1)$ האפשרויות לבחור קשת.

לכן מתוך הבינום של ניוטון ניתן לראות כי גודל הקבוצה המדוברת לא יעלה על מחצית מגודל האפשרויות , כמו בכל פעם שמפעילים (n ,r) nPr .

מכאן שאם גודל קבוצת הגרפים הפוטנציאליים הוא $2^{\frac{|V|\cdot(|V|-1)}{2}} \sim \sqrt{2}^{|V|^2}$

פעולת isDegenerated סורקת את הקשתות בקבוצה ולכן לא תעבור את $\theta(|v| - 1)$

לכן השילוב של השתיים יניב

$$\theta\left(\sqrt{2}^{|V|^2} \cdot |V|\right)$$

ג. מציאת מינימום בקבוצה: סריקה ליניארית על כל המשקלים. מציאת משקל - $\theta(|V| - 1)$.

ניתחנו כבר את גודל הקבוצה ומאותם הנימוקים , זה לא יעלה על $\sqrt{2}^{|V|^2}$ ולכן

מדובר בסה"כ ב - $\theta\left(|V|\sqrt{2}^{|V|^2}\right)$

נסכום את כל השלבים :

$$\theta(2^{|V|^2}) + \theta\left(\sqrt{2}^{|V|^2} \cdot |V|\right) + \theta\left(|V|\sqrt{2}^{|V|^2}\right) = \theta\left(|V|\sqrt{2}^{|V|^2}\right)$$

ב. תכנן/י אלגוריתם המחשב כמה זמן יקח למלא את כל הכספומטים אם ידוע כי כולם ריקים וכעת הרכב כן צריך לחזור למחסן כדי להצטייד מחדש כאשר הוא מתרוקן. לצורך הסעיף הניחו כי קיימת שיטה בשם Road כמו בסעיף הקודם. נתח/י את סיבוכיות זמן הריצה של האלגוריתם.

ב.

סעיף זה דומה לסעיף א', עם שינוי קטן : יש להתייחס למצב שבו נגמר הכסף ברכב.

האלגוריתם:

1. ניצור קבוצה של כל תתי הקבוצות מ – E שגודלן הוא בדיוק |V – 1 | כלומר שיש להן פוטנציאל ליצור עץ פורש מנוון.
2. נסיר מהקבוצה את כל תתי הקבוצות שאינן מייצרות עץ פורש מנוון.

3. לכל עץ פורס מנוון בקבוצה נגדיר מצביע לשורש העץ , נעקוב עם המצביע אחרי כל הקודקודים עד אשר נגיע לקודקוד ה – 100 .נסמן קודקוד זה כ- v , הבן היחיד של קודקוד זה הוא u (מאחר והעץ מנוון). כעת אם נגדיר את קודקוד תחנת המילוי כ- t אז נבצע את הפעולה הבאה :
נסיר את הקשת (v,u)
נוסיף את הקשת (v,t)
נוסיף את הקשת (u,t)

כעת נחזור על הפעולה לכל 100 קודקודים.
בדרך זו יצרנו דימוי של חזרה לתחנת המילוי.

4. נבחר את תת הקבוצה (שהיא גם עץ פורס) עם המשקל המינימלי בקבוצה גלה את הקשת הקצרה ביותר המחברת את הקודקוד הנוכחי u וקודקוד שעדיין לא ביקרנו בו – v . בחישוב קח בחשבון גם את מחיר המעבר מעיר לעיר (+6 דקות נוספות) וכן את המהירות בכביש עירוני לעומת כביש בינעירוני.

סיבוכיות:

השינוי שהוסף יתבצע עוד $\left\lfloor\frac{|V-1|}{100}\right\rfloor$ פעמים לכל מסלול בקבוצת המסלולים . ישנם לכל היותר $2^{\frac{|V|\cdot(|V|-1)}{2}}$.

ניתן לראות שהמכפלה בין השניים קטנה מזמן הריצה המקורי ולכן לא יהיה שינוי אסימפטוטי בזמן הריצה :

$$\theta\left(|V|\sqrt{2}^{|V|^2}\right)$$

ג. כעת אנחנו מוסיפים את האפשרות שבנק כלשהו יכול לשלם כדי לקבל תיעדוף במילוי הכספומטים שלו. תכנן/י אלגוריתם המחשב כמה זמן יקח למלא את כל הכספומטים אם ידוע כי כולם ריקים וכעת הרכב כן צריך לחזור למחסן כדי להצטייד מחדש. לצורך הסעיף הניחו כי קיימת פונקציה בשם Road כמו בסעיף הקודם. נתח/י את סיבוכיות זמן הריצה של האלגוריתם.

כתבו פונקציה בJAVA שמממשת את האלגוריתם עבור סעיף ג ותוכנית ראשית שמשתמשת בה ומפעילה את האלגוריתם עבור כל בנק בנפרד ושומרת את רשימת הזמנים בסדר לא יורד.

ג.

האלגוריתם:

1. רדוקציה: נמצא את משקל הצלע המקסימלית , נסמן אותו ב – x . נוסיף את x לכל צלע שאינה מובילה לסניף מתועדף. בדרך זו הבטחנו את עליונות של הבנקים המנויים.
2.
3. ניצור קבוצה של כל תתי הקבוצות מ – E שגודלן הוא בדיוק |V – 1| כלומר שיש להן פוטנציאל ליצור עץ פורס מנוון.
4. נסיר מהקבוצה את כל תתי הקבוצות שאינן מייצרות עץ פורס מנוון.
5. לכל עץ פורס מנוון בקבוצה נגדיר מצביע לשורש העץ , נעקוב עם המצביע אחרי כל הקודקודים עד אשר נגיע לקודקוד ה – 100 .נסמן קודקוד זה כ- v , הבן היחיד של קודקוד זה הוא u (מאחר והעץ מנוון). כעת אם נגדיר את קודקוד תחנת המילוי כ- t אז נבצע את הפעולה הבאה :

נסיר את הקשת (v,u)
נוסיף את הקשת (v,t)
נוסיף את הקשת (u,t)

כעת נחזור על הפעולה לכל 100 קודקודים.
בדרך זו יצרנו דימוי של חזרה לתחנת המילוי.

6. נבחר את תת הקבוצה (שהיא גם עץ פורס) עם המשקל המינימלי בקבוצה גלה את הקשת הקצרה ביותר המחברת את הקודקוד הנוכחי u וקודקוד שעדיין לא ביקרנו בו – v . בחישוב קח בחשבון גם את מחיר המעבר מעיר לעיר (6+ דקות נוספות) וכן את המהירות בכביש עירוני לעומת כביש בינעירוני.

מציאת צלע מקסימלית: $\theta(|E|) = \theta(|V|^2)$

הוספת X: $\theta(|E|) = \theta(|V|^2)$

גם כאן אין שינוי בקנה מידה אסימפטוטי:

$$\theta\left(|V|\sqrt{2}^{|V|^2}\right)$$

מימוש ב – JAVA

בקובץ מצורף

ד.

שאלה 2

א. האם קיים אלגוריתם המייצר עץ חיפוש בינארי מרשימה לא ממויינת? אם כן תארו אותו, אחרת הסבירו מדוע אינו קיים.

ב. טענה - כדי לפתור את בעיית המרחק הקצר ביותר בגרף ללא צלעות שליליות יש להרפות relax לפחות חלק מהצלעות לפחות פעמיים. הסבירו או הפריכו טענה זו.

ג. טענה - בגרף מכוון וקשיר עם צלעות בעלות משקל חיובי בלבד, אלגוריתם בלמן-פורד יפעל בסיבוכיות זמן ריצה כמו דייקסטרה באופן אסימפטוטי. הסבירו או הפריכו טענה זו.

ד. נתון גרף מכוון G עם פונקצית משקל $w: E \rightarrow \mathbb{R}$ ללא מעגלים שליליים.
קוטר גרף של G מוגדר להיות המרחק הגדול ביותר בין שני קודקוד כלשהם בגרף. כלומר
$$dim(G) = max_{u,v \in V}\{dist(u, v)\}$$

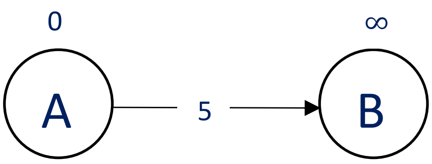
תכנן/י אלגוריתם המחשב את קוטר הגרף. הוכיחו בקצרה הנכונות וחשבו זמן הריצה.

א.

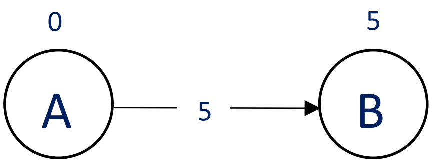
בהחלט ניתן לייצר עץ מיון בינארי מרשימה לא מממוינת. לשם כך ניעזר בשיטת ה – insertion של עץ חיפוש בינארי (זו ידועה ומוכחת). בנוסף שיטה זו משמרת את מבנה העץ. ניתן לחזור על הפעולה לכל חוליות ברשימה הלא -ממוינת.

ב.

לא בהכרח שיש צורך להרפות את הצלעות יותר מפעמיים , נציג דוגמה פוסלת :



Relax (A , B)



ניתן לראות שהפעלנו הרפיה על כל אחת מהקשתות (במקרה שלנו יש רק אחת כזאת) רק פעם אחת.

ג.

נתבונן בחסמים ההדוקים של האלגוריתמים דייקסטרה ובלמן-פורד:

דייקסטרה : $\theta(E + V\,log\,V)$
בלמן פורד : $\theta(EV)$

נתון שהגרף קשיר , לכן קיימת קשת מכל קודקוד לכל קודקוד ⇐ קיימת לפחות קשת אחת מכל קודקוד לקודקוד אחר ⇐ $V - 1 \leq E \leq \frac{V \cdot (V-1)}{2}$.

כעת נציב את ערכי המינימום והמקסימום של הטווח בתוך זמני הריצה של כל אחת מהחסמים.

דייקסטרה : $\theta((V-1)+V\,log\,V) \le T \le \theta\left((\frac{V\cdot(V-1)}{2})+V\,log\,V\right)$

$$\Rightarrow \theta(V\,log\,V) \le T \le \theta(V^2)$$

בלמן-פורד : $\theta((V-1)V) \le T \le \theta\left((\frac{V\cdot(V-1)}{2})V\right)$

$$\Rightarrow \theta(V^2) \le T \le \theta(V^3)$$

ניתן לראות שאין חפיפה בשום שלב בין התחומים. לכן ניתן לטעון בביטחון שתרחיש זה לא יגרום להתקרבות אסימפטוטית בזמני הריצה.

ד.

אתייחס לשאלה כאילו הכוונה למסלול הארוך ביותר בחישוב המשקל. אם הכוונה לחישוב הקוטר ללא התחשבות בפונקציית המשקל w , אז שאלה זו נשאלה כבר במטלה 2 שאלה 6 . תשובתי לא תשתנה.

האלגוריתם:

1. שימוש באלגוריתם פלויד-ורשל על מנת לקבל מטריצה עם כל המסלולים הקצרים ביותר מכל קודקוד לכל קודקוד.
2. נסרוק את כל המטריצה ונחפש את הערך המקסימלי. זה יישקף את קוטר הגרף.

הוכחת נכונות:

1. אלגוריתם פלויד-ורשל יניב את קבוצת המרחקים (בהתחשבות במשקל) המינימליים בין כל שני קודקודים – נסמן אותה A . את זאת הוכחנו בכיתה.
2. מאחר ו – A היא קבוצת מספרים מתקיים ש $\mathbb{R} \subseteq A$. וכן A קבוצה סופית שכן בגרף חייב להיות מספר קודקודים סופי.
3. $\mathbb{R}$ היא קבוצה המהווה יחס סדר מלא ולכן כל קבוצה הקטנה ממנה (בפרט A) גם מהווה יחס סדר מלא. מכאן שאם A סופית חייב להיות בה איבר מקסימלי.
4. לכן בסריקה ליניארית בוודאות נמצא את האיבר המקסימלי, זה שקול לקוטר הגרף.

סיבוכיות:

פלויד-ורשל : $\theta(V^3)$

סריקה ליניארית: $\theta(V)$

סה"כ : $\theta(V)+\theta(V^3)=\theta(V^3)$

<u>שאלה 3</u>

נתון גרף $G = (V,E)$ קשיר, לא מכוון וממושקל כאשר כל הקשתות בעלות ערך חיובי.

נאמר שקשת e היא **מיצר** בין קודקודים u,v אם היא קשת בעלת הערך הכי נמוך wl הנמצאת בכל המסלולים ביותר בין שני הקודקודים. כלומר, מסלול כלשהו בין u,v עובר בקשת e או בקשת אחרת בעלת אותו משקל ואין קשת אחרת במסלול בעלת ערך נמוך יותר.

תכנן/י אלגוריתם המוצא את המשקלים של כל המיצרים בין כל זוג קודקודים בגרף באופן היעיל ביותר ונתחו את סיבוכיות זמן הריצה שלו.

הדרכה: אתחל/י מערך בגודל |V| על |V| שישמור על המשקלים של כל המיצרים. הערכים בהתחלה צריכים להיות נכונים עבור גרף חסר קשתות. לאחר מכן הוסיפו את הקשתות אחת אחרי השנייה.

<u>האלגוריתם</u>:

1. נשתמש באלגוריתם פלויד-ורשל ועליו "נלביש" הגדרה נוספת.
2. נגדיר מטריצה שיכולה לייצג את כל ההצטלבויות בין שני קודקודים.
3. כעת במקום למצוא את הערכים המינימליים כמו בפלויד-ורשל , נעדכן בתא של המטריצה טבלת ערבול המכילה את ייצוג של כל המסלולים המינימליים. לדוגמה אם קיים מסלול מינימלי מ-A אל C ואז B , נשמור בתא המתאים טבלת ערבול המכיל את המפתח "ACB" .
4. בכל איטרציה נבדוק אם מצאנו מסלול הנמוך מהמסלולים הקיימים בטבלת הערבול.
   - במידה והמסלול ארוך מהמסלולים האחרים , נתעלם ממנו כמו בפלויד ורשל.
   - במידה והמסלול זהה, נצרף אותו למסלולים הקיימים.
   - במידה והמסלול קטן , מצאנו מסלול מינימלי חדש. נשחרר את הזיכרון של טבלת הערבול ונגדיר טבלה חדשה המכילה את המסלול שזה עתה מצאנו.
5. בתום הריצה תתקבל מטריצה שבכל תא שלה קיימת קבוצת מסלולים מינימליים. נסרוק את הקבוצה ונחפש את הקשת המינימלית בכל אחת מקבוצות. כעת נסרוק פעם נוספת את הקבוצה ונחפש מופע נוסף של הקשת. אם התקבלו למעלה מ – 2 מופעים , מדובר במיצר , נדפיס אותו.

<u>סיבוכיות</u>:

שלבים 1 – 4 : לא היה שינוי בקנה מידה פולינומיאלי בתוך כל איטרציה ,

זמן הריצה זהה לזה של פלויד ורשל: $\theta(V^3)$

שלב 5 : אם נחזיק מראש לכל טבלת ערבול עוד רשימה עם המצרים , קל יותר לראות שהוספה לרשימה היא פעולה בודדת ולכן לא תשנה את סדר הגודל .

זמן הריצה זהה לזה של פלויד ורשל: $\theta(V^3)$

סה"כ : $\theta(V^3)$

שאלה 4

נתון נתון גרף $G = (V, E)$ קשיר, מכוון, חסר מעגלים וממושקל כאשר כל הקשתות בעלות ערך חיובי שלם.

תכנן/י אלגוריתם יעיל ככל הניתן המוצא את כל המסלולים בעלי משקל זוגי בין כל זוג קודקודים בגרף.

תזכורת - משקל מסלול הוא סכום כל משקלי הקשתות במסלול.

האלגוריתם:

6. נשתמש באלגוריתם פלויד-ורשל ועליו "נלביש" הגדרה נוספת.
7. נגדיר מטריצה שיכולה לייצג את כל ההצטלבויות בין שני קודקודים.
8. כעת במקום למצוא את הערכים המינימליים כמו בפלויד-ורשל , נעדכן בתא של המטריצה טבלת ערבול המכילה את ייצוג של כל המסלולים הזוגיים. לדוגמה אם קיים מסלול זוגי מ-A אל C ואז B , נשמור בתא המתאים טבלת ערבול המכיל את המפתח "ACB" .
9. בכל איטרציה נבדוק אם מצאנו מסלול זוגי , ואם זה עדיין לא קיים , נוסיף אותו לטבלת הערבול.