תכנון אלגוריתמים - מטלה 3

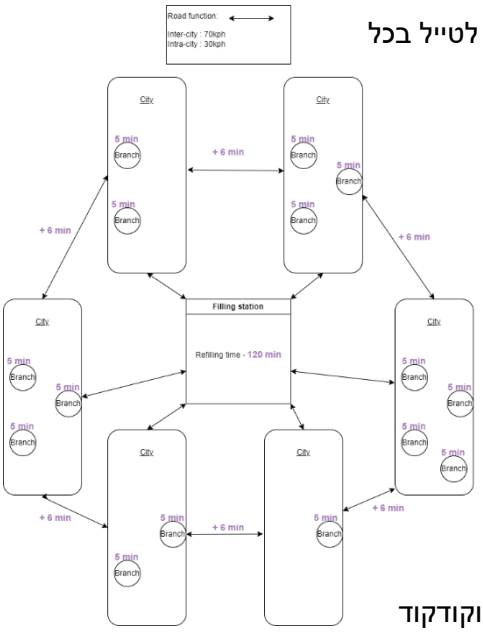
יוליה מושן : 319565610

גיל פסי : 206500936

שאלה 1

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 הניחו 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.

תחת זאת ניתן למצוא פתרון 'קרוב מספיק' בזמן ריצה פולינומיאלי.

כלומר כזה שלא יבטיח לנו את הדרך הקצרה ביותר אבל באופן סטטיסטי יבטיח לנו דרך שקצרה מרוב הדרכים האחרות.

לשם כך נשתמש בשיטת ה – Nearest Neighbor .

<u>האלגוריתם:</u>

1. אתחול כל הקודקודים כ- unvisited , הגדר משתנה זמן t .
2. בחר קודקוד שרירותי, הגדר אותו כקודקוד u הנוכחי .
3. סמן אותו כ - unvisited .
4. 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.
5. הוסף ל – t את משך כלל הפעולות : נסיעה + מילוי.

6. הגדר את v כקודקוד u הנוכחי . את v כ – visited. אם כל הקודקודים בגרף מסומנים כ - visited, החזר t. אחרת, חזור לשלב 3.

סיבוכיות:

ממבט על ניתן לראות שבסופו של דבר נעבור על כל הקודקודים - $\theta(n)$.

עבור כל קודקוד נבצע השוואה עם כל השכנים. כלומר במצטבר נשווה עם כל הקודקודים

האחרים - $\theta(n)$.

כלומר במצטבר מדובר ב – $\theta(n \cdot n) = \theta(n^2)$

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. אתחול כל הקודקודים כ- unvisited , הגדר משתנה זמן t .
2. בחר קודקוד שרירותי, הגדר אותו כקודקוד u הנוכחי .
3. סמן אותו כ - unvisited .
4. 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.
5. הוסף ל – t את משך כלל הפעולות : נסיעה + מילוי.
6. במידה עברנו 100 תחנות מאז המילוי האחרון , חזור לתחנת המילוי והפעל את האלגוריתם מחדש.
7. הגדר את v כקודקוד u הנוכחי . את v כ – visited. אם כל הקודקודים בגרף מסומנים כ - visited, החזר t. אחרת, חזור לשלב 3.

סיבוכיות:

השינוי שהוסף יתבצע עוד $\left\lfloor \frac{n}{100} \right\rfloor$ פעמים . כלומר לא תהיה השפעה אסימפטוטית על הסיבוכיות.

$$\theta(n^2)$$

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.

<u>האלגוריתם:</u>

גם כאן האלגוריתם לא יושפע בצורה דרמטית :

1. אתחול כל הקודקודים כ- unvisited , הגדר משתנה זמן t .
2. בחר קודקוד שרירותי, הגדר אותו כקודקוד u הנוכחי .
3. סמן אותו כ - unvisited .
4. גלה את הקשת הקצרה ביותר המחברת את הקודקוד הנוכחי u וקודקוד שעדיין לא ביקרנו בו – v וכן שיש בנק שמשלם על העדיפות שלה.
   אם לא קיימת כזאת גלה את הקשת ללא עדיפות. בחישוב קח בחשבון גם את מחיר המעבר מעיר לעיר (+6 דקות נוספות) וכן את המהירות בכביש עירוני לעומת כביש בינעירוני.
5. הוסף ל – t את משך כלל הפעולות : נסיעה + מילוי.
6. במידה עברנו 100 תחנות מאז המילוי האחרון , חזור לתחנת המילוי והפעל את האלגוריתם מחדש.
7. הגדר את v כקודקוד u הנוכחי . את v כ – visited .אם כל הקודקודים בגרף מסומנים כ - visited, החזר t .אחרת, חזור לשלב 3.

השינוי מוסיף השוואה אחת לכל איטרציה ולכן בסה"כ יוסיף $\theta(n^2)$ פעולות לכל היותר.

בסה"כ

$$\theta(n^2)$$

<u>מימוש ב – JAVA</u>

בקובץ מצורף

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

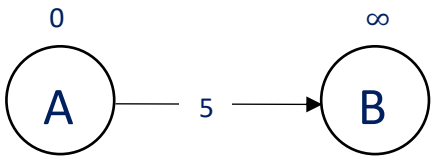
תכנן/י אלגוריתם המחשב את קוטר הגרף. הוכיחו בקצרה הנכונות וחשבו זמן הריצה.

א.

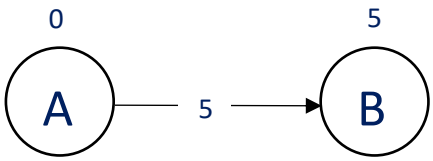
בהחלט ניתן לייצר עץ מיון בינארי מרשימה לא מממוינת. לשם כך ניעזר בשיטת ה – insertion של עץ חיפוש בינארי (זו ידועה ומוכחת). בנוסף שיטה זו משמרת את מבנה העץ. ניתן לחזור על הפעולה לכל חוליות ברשימה הלא -ממוינת.

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 ⇐ $V - 1 \le E \le \frac{V\cdot(V-1)}{2}$.

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.

<u>האלגוריתם</u>:

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.

<u>הוכחת נכונות:</u>

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 $A \subseteq \mathbb{R}$. 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.

<u>סיבוכיות:</u>

פלויד-ורשל : $\theta(V^3)$

סריקה ליניארית: $\theta(V)$

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

שאלה 3

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 w| 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.

??????????????????????

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:

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 הזוגיים. לדוגמה אם קיים מסלול זוגי מ-A אל C ואז B , נשמור בתא המתאים טבלת ערבול המכיל את המפתח "ACB" .
4. בכל איטרציה נבדוק אם מצאנו מסלול זוגי , ואם זה עדיין לא קיים , נוסיף אותו לטבלת הערבול.