לירון שלי

313576209

אלגוריתמים 20417

07/01/20

**ממ"ן 14**

**שאלה 1**

*נגדיר OPT:*

*נסמן ב את המחיר המזערי של המסלול המתחיל בשכבה השמאלית ביותר i=1 ועד לקודקוד .*

*נטען כי מתקיים לכל : (הוכחה בהמשך)*

*נשתמש ב memozation כדי לשמור לכל קודקוד בנוסף למחירו*

*גם את ערך המסלול בעל המשקל המינימלי מהשכבה השמאלית ביותר אליו - וגם את הדרך של אותו מסלול (באילו נקודות עבר) .*

*את הבדיקה נבצע לפי עמודות וכך זמן הריצה של החישוב של כל OPT יהיה O(1). (פירוט בהמשך)*

***האלגוריתם:***

*מקבל מטריצה בגודל ע"י הוספת שורות j=0,n+1 . נטפל בחריגה מגבולות השריג:*

***הוכחת נכונות:***

*נתחיל בהוכחת הטענה כי*

*מגדיר את המחיר המינימלי של המסלול מהשכבה השמאלית ביותר בשריג i=1 עבור כל .*

*נבצע הוכחה באינדוקציה על i .*

*עבור i=1 ברור – המחיר המינימלי של איבר בשכבה השמאלית ממנו לעצמו הוא מחירו שלו.*

*נניח כי הטענה מתקיימת עבור כל i עד k-1, נראה כי מתקיימת לכל k:  
הראנו עבור ונראה מה קורה כאשר . OPT מוגדר כך:*

המחיר המינימלי של מסלול (נסמן: m) מהשכבה השמאלית ביותר לקודקוד (k,j) מקיים אם כך:

מאחר והתנועות המותרות בשריג הן ימינה, ימינה-מטה או ימינה-מעלה, המינימלי מבין שלושת המסלולים שממשיכים אל קודקוד , יהיה מסלול בעל מחיר מינימלי אל מקודקוד בשכבה השמאלית ביותר. (אין מסלול אחר שמוביל אליו מהשכבה השמאלית ביותר).

אם כך, משקלו של מסלול בעל מחיר מינימלי זהו מחיר המסלול בעל המחיר המינימלי כולו מהשכבה השמאלית ביותר אל:

בתוספת מחירו של הקודקוד .

כלומר:

מהנחת האינדוקציה ידוע כי הם מסלולים בעלי מחיר מינימלי, כלומר:

וקיבלו כי

כלומר, OPT הוא המחיר המינימלי של מסלול המתחיל בשכבה השמאלית ועד לקודקוד ובכך השלמנו את הוכחת האינדוקציה.

האלגוריתם מוצא ושומר עבור כל קודקוד באריג את ה- OPT שלו ואת הדרך שהובילה אליו, מהעמודה הראשונה ועד האחרונה. בסיום שמירת ערכי הOPT וה-PATH, נבדוק מי הOPT המינימלי בעמודה הימנית ביותר, והוא למעשה המחיר הנמוך ביותר של מסלול מהשכבה השמאלית ביותר לימנית ביותר. הPATH של אותו קודקוד לו שייך הOPT הוא המסלול שנדרשנו למצוא שלו המחיר המזערי המקיים את הנדרש באריג.

**זמן ריצה:**

הוספת 2n שורות לטיפול בחריגות המטריצה .

חישוב הOPT בעמודה הראשונה ושמירת ערכיו וערכי הPATH מתבצע בעלות קבועה לכל קודקוד, ישנם n קודקודים בעמודה זו ולכן .

העמודה השניה משתמשת בערכי העמודה הראשונה ולכן לא נדרשת לחשב מחדש אף ערך, כלומר גם עבור עמודה זו בדיקת המינימלי ושמירת ערכיו וערכי הPATH מתבצעים בעלות קבועה לכל קודקוד.

וכך הלאה, עד לעמודה האחרונה. ישנן n עמודות ולכן

לאחר מכן חיפוש המינימלי מתוך n ערכים ושמירת הPATH שלו גם תתבצע ב ולכן.

סה"כ זמן ריצה , כנדרש.

שאלה 2

*נמיין את התיבות לפי אורכן (l) מהארוכה ביותר (תהיה תיבה עם אינדקס 1) לקצרה ביותר (תיבה עם אינדקס n).*

*נסמן ב את הגובה המקסימלי של מגדל בו התיבה i היא העליונה.*

*תהי j התיבה בעלת הגובה המקסימלי עבורה .*

*אם קיימת תיבה כזו, מהגדרה, לכל :*

*אם לא קיימת תיבה כזו, אז המגדל בו i היא התיבה העליונה הוא מגדל המכיל את התיבה i בלבד (אין תיבות תחתיה, ומאחר והיא התיבה העליונה גם אין תיבות מעליה),*

*ואז .*

*נשתמש ב memozation כדי לשמור לכל תיבה את גובה המגדל המקסימלי - בו היא עליונה וגם את רשימת התיבות שנכללות במגדל .*

***האלגוריתם:***

*מקבל מערך של n תיבות, ממויינות לפי אורכן 1 התיבה הארוכה ביותר ו-n הקצרה ביותר.*

*OPT(1)=h(1)*

*For (i=2 to n)*

*(i)*

**הוכחת נכונות:**

מאחר ואורך התיבות שונה, נוכל לדעת בוודאות כי לאחר מיון התיבות, כשהראשונה היא התיבה הארוכה ביותר, תחת אותה תיבה ראשונה לא יכולות להיות תיבות, ולכן גובהה המקסימלי כאשר היא התיבה העליונה הוא הגובה של התיבה עצמה: OPT(1)=h(1).

תחת התיבה השניה, תוכל להיות רק התיבה הראשונה – היא היחידה שארוכה ממנה, וגם זאת, במידה והתיבה הראשונה גם רחבה יותר ממנה. וכך נתקדם כשתחת כל תיבה יכולה להיות רק תיבה שהופיעה לפניה, כלומר, תיבה ארוכה יותר.

יבדקו כל התיבות שמופיעות לפני התיבה הנוכחית, ונמצא את התיבה עם הגובה המקסימלי שהתיבה הנוכחית תוכל להיות מעליה. מאחר וכבר חושב גובהן המקסימלי של כל התיבות שמופיעות לפני התיבה הנוכחית קודם לכן, ניתן למצוא את המקסימלי בהן שעונה על קריטריון הרוחב, נצרף את גובה התיבה הנוכחית ונקבל את הגובה המקסימלי של התיבה כשהתיבה הנוכחית היא התיבה העליונה.

לבסוף נבדוק עבור כל i בתחום מהו הגובה המקסימלי, גובה זה הוא גובה מגדל התיבות היציב הגבוה ביותר שניתן לבנות מהתיבות הנתונות.

מאחר ושמרנו גם את שרשור התיבות המופיעות במגדל המקסימלי ניתן בפשטות להחזיר אותו ובכך להראות מאילו תיבות נבנה המגדל המקסימלי.

**זמן ריצה:**

מיון התיבות לפי אורך .

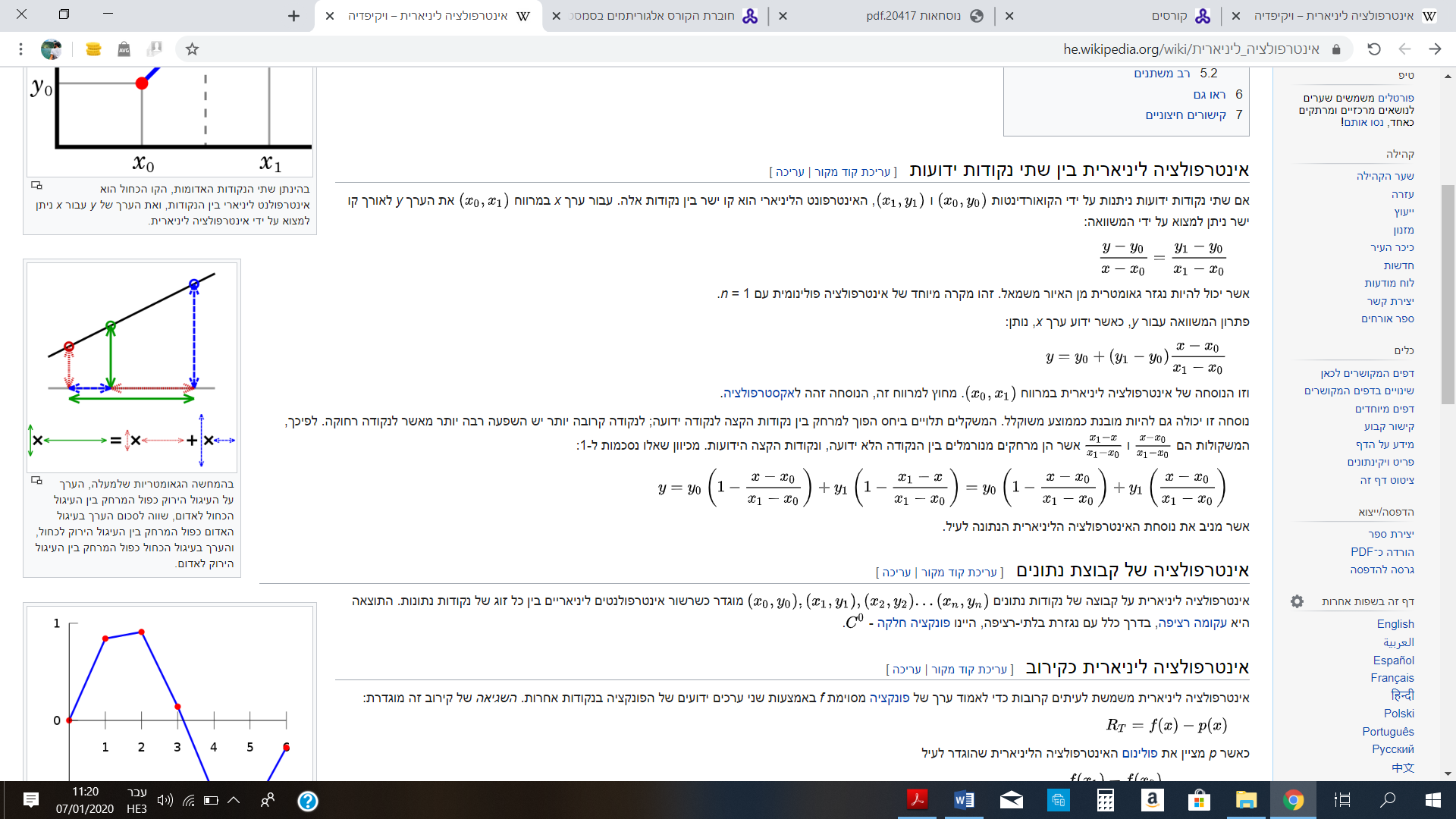
לכל i מתבצעות מספר קבוע של פעולות מול n-i תיבות, כלומר

מציאת המקסימלי מבין כל הOPT - .

סה"כ זמן ריצה , כנדרש.

שאלה 3

1. נוסחה למציאת אינטרפולציה ליניארית בין שתי נקודות ידועות:

{\displaystyle y=y\_{0}\left(1-{\frac {x-x\_{0}}{x\_{1}-x\_{0}}}\right)+y\_{1}\left(1-{\frac {x\_{1}-x}{x\_{1}-x\_{0}}}\right)=y\_{0}\left(1-{\frac {x-x\_{0}}{x\_{1}-x\_{0}}}\right)+y\_{1}\left({\frac {x-x\_{0}}{x\_{1}-x\_{0}}}\right)}

כלומר: *.*

כל נקודה x שנציב בפולינום תקיים אם כך:

נציב בנוסחה:

כלומר: .

אם כך הפולינומים שנדרשנו למצוא הם:

1. נוסחת הנסיגה מוגדרת ע"י:

כאשר  *מגדיר את הפולינום דרכו עוברות הנקודות .*

*לצורך חסכון בזיכרון ישמרו התוצאות במערך M (ניתן יהיה לשמור את מקדמי הפולינום).*

**האלגוריתם:**

\*בלולאה השנייה ה-i מסמן את המרחק בין i ל-j.

**נכונות:**

האלגוריתם מקבל סדרה של n נקודות ומחזיר את הפולינום דרכו עוברות כל הנקודות הללו. אין צורך להוכיח את נכונות נוסחת הנסיגה (נתונה והוכחה בסעיף קודם).

תחילה מחושבים הפולינומים בעלי נקודה אחת (ערכי y של אותן נקודות), O(1).

לאחר מכן מחושבים פולינומים בעלי שתי נקודות, ע"פ הנוסחה שדרושת לדעת את ערכי הפולינומים בעלי נקודה אחת (שכבר חושבו).

וכך הלאה, חישוב של פולינום בעל k נקודות דורש ע"פ הנוסחה לדעת את ערכי הפולינומים בk-1 נקודות. מאחר ואלה כבר חושבו, ניתן לעשות זאת גם כן בזמן O(1), אולם חישוב מכפלת פולינום בגודל n-1 בפולינום מגודל 1 ניתנת לביצוע בזמן O(n).

**זמן ריצה:**

ישנן n נקודות ועלינו לבדוק סדר גודל של  *פולינומים, כשכל בדיקה מסתמכת על בדיקה קודמת ולכן מתבצעת ב* O(1), אך ישנו צורך בחישוב מכפלת פולינום ממעלה 1 בפולינום ממעלה n-1 במקרה הגרוע ( האחרון), ומשך הזמן הדרוש לכך יהיה .

לכן זמן הריצה במקרה הגרוע יהיה  עבור כל  *סה"כ זמן ריצה .*

1. הקלט לאלגוריתם:

*המערך M לאחר האיטרציה הראשונה למציאת הפולינומים בעלי נקודה אחת:*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 | M |
|  |  |  |  | 46 | 1 |
|  |  |  | 2 |  | 2 |
|  |  | 0 |  |  | 3 |
|  | 10 |  |  |  | 4 |
| 98 |  |  |  |  | 5 |

באיטרציה הבאה i=1 ו-j רץ מ1 עד n-1 (כלומר עד 4):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 | M |
|  |  |  |  | 46 | 1 |
|  |  |  | 2 |  | 2 |
|  |  | 0 |  |  | 3 |
|  | 10 |  |  |  | 4 |
| 98 |  |  |  |  | 5 |

באיטרציה הבאה i=2 ו-j רץ מ1 עד n-2 (כלומר עד 3):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 | M |
|  |  |  |  | 46 | 1 |
|  |  |  | 2 |  | 2 |
|  |  | 0 |  |  | 3 |
|  | 10 |  |  |  | 4 |
| 98 |  |  |  |  | 5 |

באיטרציה הבאה i=3 ו-j רץ מ1 עד 2:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 | M |
|  |  |  |  | 46 | 1 |
|  |  |  | 2 |  | 2 |
|  |  | 0 |  |  | 3 |
|  | 10 |  |  |  | 4 |
| 98 |  |  |  |  | 5 |

באיטרציה האחרונה i=4 ו-j רץ מ1 עד 1:

והאלגוריתם מחזיר את הפולינום המקורי , כנדרש.

שאלה 4

1. האלגוריתם מחשב את המסלול הקל ביותר מקודקוד r לכל שאר הקודקודים בגרף.

נוכיח באינדוקציה על מספר הקשתות בגרף:

עבור n=1, ישנה קשת אחת .

בה"כ u=r, הקודקודים u,v מאותחלים A[u]=0, A[v]=∞, בסריקות הלולאה יתבצע עדכון A[v]=A[u]+c(e) ולאחר מכן האלגוריתם יסתיים. כלומר נקבל את משקל המסלול הקל ביותר מv ל-u, הוא משקל הקשת היחידה בינהם c(e).

נניח כי הטענה מתקיימת עבור k-1 קשתות, נראה כי היא מתקיימת גם עבור k קשתות.

עבור n=k, בה"כ u1=r, כלל שאר הקודקודים u בגרף מאותחלים A[u]=∞.

כל קודקוד u שאין דרך כלשהי להגיע אליו מ- u1 ישאר עם ערך ∞.

אם קיים לפחות קודקוד אחד כזה, הרי שבשאר הגרף (מלבדו) יש k-1 קודקודים, ולפי הנחת האינדוקציה אותם קודקודים מקיימים את הנדרש. מאחר וגם הוא מקיים את הנדרש (הרי שאין מסלול) קיבלנו כי הטענה נכונה עבור k.

אם לא קיים אף קודקוד כזה, הרי שיש דרך להגיע לכל קודקוד בגרף מ- u1.

לכן, קיימת צלע עבור i כלשהו, ומתיאור האלגוריתם נקבל כי יתבצע באיטרציה הנוכחית העדכון .

כל עוד ישנם עדכונים האלגוריתם ימשיך, וכשיתבצע לאחר מעבר על כל הצלעות אם לא ידרשו עדכונים, הרי שהקודקודים כולם מעודכנים לערכם המינימלי, מש"ל.

1. מספר האיטרציות המירבי על גרף בעל n קודקודים הוא n .

האלגוריתם סורק את הצלעות בסדר לקסיקוגרפי, כלומר, במידה ומדובר במסלול שהצלע הראשונה שתקיים את התנאי A[v]>A[u]+c(e) בו היא הצלע בעלת הערך הלקסיקוגרפי הגבוה ביותר, תתבצע למעשה אחרונה. מאחר ובוצע עדכון נבצע את האיטרציה החיצונית פעם נוספת.

אם באיטרציה הבאה הצלע הראשונה שתקיים את התנאי A[v]>A[u]+c(e) תהיה בעלת הערך הלקסיקוגרפי השני הגבוה ביותר, שוב יתבצע עדכון של קודקוד יחיד אחד בלבד.

מסקנה, אם נתקדם בקצב כזה של עדכון אחד בכל אינטרציה, לעדכון כלל הקודקודים נדרש למספר אינטרציות כמספר הקודקודים -n .

נציע סדרת גרפים:

1. נציע את סדרת הגרפים:

לפי הסדר הלקסיקוגרפי הבדיקה תתחיל מהצלע ויתבצע עדכון של כל הצלעות למשקלן המינימלי. בסיום הלולאה הפנימית מאחר ובוצעו עדכונים נדרש לבצע שוב את הלולאה החיצונית.

באיטרציה השניה לא יתבצעו עדכונים כלל כי הקודקודים מעודכנים למשקל המסלול המינימלי מr. ולכן האלגוריתם יסתיים כעבור שתי איטרציות בלבד, כנדרש.