פרטי המגיש:  
שם: יניב גולדשטיין  
תעודת זהות: 207542903  
כתובת: תמר 11, הרצליה

ממ"ן 14 – אלגוריתמים – (20417)

שאלה 1:

* הנחות בשאלה
  1. השריג הנתון ריבועי בגודל nXn.
  2. הצעדים האפשריים מהשכבה השמאלית לימנית הם: ימינה-למטה, ימינה וימינה-למעלה.
* אינטואיציה:
  + נשים לב שכדאי להגיע מהשכבה השמאלית לימנית נוכל את הצעדים גם בצורה הפוכה, כלומר להתחיל מנקודות הקצה הימניות ולהמשיך שמאלה ע"י צעדים הפוכים (שמאלה, שמאלה-למעלה, שמאלה-למטה).
  + אם נחשוב לבצע את הצעדים בצורה הגולמית בה הם נתונים – לא נהיה תלויים בהכרח בצעדים שכבר חושבו בשלב מוקדם יותר, ולכן לא נוכל להשתמש בתכנון דינמי בצורה האופטימלית.
  + ועל כן - נשנה את כיוון הזרימה, אחרי שנמצא את אורך המסלול הקצר ביותר – נחלץ אותו.
* רעיון האלגוריתם:
  + נגדיר ב את מחיר המסלול המנימלי שמתחיל בנקודה כלשהיא בשכבה השמאלית ומגיע ל.
  + נסמן את המט' הנתונה ב A , נוסיף לה 2 שורות ועמודה כך :   
    הוספת שורות \ עמודות אלו נחוצות לחישוב התהליך הרקורסיבי אך אינן מוסיפות לסיבוכיות האלגו' מכיוון שהם מאותחלים כבר לפני ריצה האלגו'.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |
|  | *(n,n)* | *…* | *..* | *…* | *(1,n)* |
|  | *…* |  |  |  | *…* |
|  | *...* |  |  |  | *…* |
|  | *…* |  |  |  | *…* |
|  | *(n,1)* | *…* | *…* | *…* | *(1,1)* |
|  |  |  |  |  |  |

* + ע"פ ההגדרות לעיל עלינו להחזיר מהאלגו' את (המסלול הקצר ביותר משמאל לימין)
  + לאחר שמצאנו את הערך המינימלי – נשחזר את המסלול כך:
    - נמצא את הנקודה הימנית אליה הגענו בסוף המסלול ~ . נבדוק איזה ערך של J מתוך מקיים .
    - עבורהנקודות הבאות – בהינתן הנקודה הרי שלערך יש מספר אפשרויות : .
      * נבדוק לאיזה ערך מתקיים:  
           
        כדי לדעת מי יהיה ערך ה הבא.
      * מכיוון שכלל הצעדים שלנו כוללים התקדמות בכיוון הימין, הרי שמתקיים: , לכל
    - בסיום הריצה נקבל את סדרת האנדקסים שתסמן את המסלול האופטימלי.
* תיאור האלגוריתם:

1. נגדיר מטריצה בגודל אותה נסמן ב .
2. *לכל ,*
3. *לכל ,*
4. *לכל*
5. נסמן ב את ערך האינדקס עבורו
6. נחשב לכל את סדרת האינדקסים כפי שתיארנו לעיל
7. נחזיר את הסדרה

* פסודו-קוד:

1. // declare new metric, items initial to
   * 1. //

* נכונות האלגוריתם
  + בשורות 1-3 מתבצע איתחול של 'מסגרת' המט' שבנוייה מהשורות והעמודה שהוספנו, כאשר בעמודה הראשונה – מכיוון שהמסלול מתחיל בנקודה זו – הוספנו לכל תא את ערך הנקודה אותה הוא מייצג .  
    השימוש ב נועד כדי להוות נציב לכך שכל שימוש בmin על ערכים אלו יתעלם מהם מתמטית, כאשר השימוש בmin מספק אותנו במסלול הקצר ביותר האפשרי.
  + בשורה 4 מתבצע השלב הרקורסיבי, כפי שניתן להרשם מאיור הטבלה לעיל – כל צעד מתבצע על סמך צעדים שכבר קרו – ועל כן התכנון הדינמי תקף.
  + בשורה 5 מתבצע חישוב על ערך המסלול המינימלי מבין נקודות הסיום (שנמצאות על השכבה הימנית)
  + בשורה 6 חילצנו את האינדקס המינימלי למסלול האופטימלי.
  + שורה 7 – שימוש רקורסיבי לזיהוי המסלול האופטימלי, הוכחה מלאה ניתנה בסעיף 'רעיון האלגוריתם'.
  + מש"ל
* חישוב סיבוכיות
  + שורות 1-4 ~ מילוי תאי המט' מגודל n\*(n+2) ~
    - בשורות 1-3 התאים מאותחלים ישירות
    - בשורה 4 – מתבצע חישוב בפרק זמן קבוע שאכן הגישה למט' ה OPT מתקיימת לערכים שכבר חושבו בשלב זה.
  + שורה 5 – מציאת מינימום מבין n ערכים ~
  + שורה 6 - סקירת של n אינדקסים והשוואה לערך המינימום ~
  + שורה 7 – ביצוע מספר קבוע של פעולות בn איטרציות ~
  + סה"כ 🡨

שאלה 2:

* רעיון האלגוריתם
  + נמיין את התיבות לפי הרוחב שלהן, כאשר לכל תיבות אם אז .
  + נגדיר את להיות גובה המגדל היציב המקסימלי בו התיבה i נמצאת בראשו.
  + ע"פ הגדרת מגדל יציב , הרי שכל תיבה כאשר מקיימת .
  + כאשר נרצה לחשב את הרי שישנן 2 אפשרויות:
    - התיבה i עומדת לבדה.
      * גובה:
    - התיבה i היא חלק ממגדל יציב.
      * ככזו גובהה המקסימלי יהיה
  + כדי לכסות את 2 המקרים – נבחר את המקסימלי מביניהם, כלומר:
  + נמלא את המערך OPT החל מ1 עד n.
  + במקביל נתחזק מערך נוסף – PREV, בכל פעם שנחשב את נשמור ב PREV את התיבה שמעליה שמנו את התיבה i. אם שמנו את i על הרצפה – נכניס PREV null. (במטרה לשחזר את רצף התיבות.)
  + הרי שגודל המגדל הדרוש בשאלה הוא .
  + כדי לשחזר את רצף התיבות – נמצא את האינדקס עבורו הוא מקסימלי, ניגש למערך PREV במקום הi , נדפיס את הערך PREV [i] וניגש למערך במקום PREV [i] וכן הלאה.
    - קיבלנו את סדר התיבות בצורה הפוכה.
* תיאור האלגוריתם

1. נמיין את התיבות לפי הרוחב שלהן.
2. נגדיר 2 מערכים בגודל n: OPT, PREV.
3. החל i=1 עד ל n:
   1. עבור החל מj=1 ל n:
      1. אם
4. נסמן ב את ערך האינדקס עבורו ונדפיס אותו.
5. כל עוד
   1. נדפיס את

* פסודו-קוד:

1. HEAP-SORT(B) // optimal sorting using the width field

   2. print

* נכונות האלגוריתם
  + שורה 3 – חישוב הOPT
    - הצעד הרקורסיבי לחישוב הגובה המקסימלי של המגדל בודק את האפשרות לכך שהתיבה עומדת בפני עצמה. ואם זה אינו המקרה – אנחנו מחפשים את הערך הגדול ביותר.
    - התיבות היחידות שאנחנו מאפשרים לשים מעל תיבה j כלשהיא הן תיבות שקטנות ממנה בlength וב width בהתאם לדרישות מגדל יציב.
    - בכל איטרציה אנחנו מוסיפים את גובה התיבה למיקום הגובה המתאים.
    - בכל ריצה אנחנו מתבססים רק על קריאות קודמות ל-OPT.
  + שורה 3 - שמירת הPREV
    - בכך שאנחנו שומרים בכל איטרציה מי היה התיבה עליה הנחנו את i – אנחנו שומרים תיעוד דינמי תוך כדי מקסום הגובה של המגדל.
  + שורה 4 – בחירת הגובה המקסימלי מתוך OPT ע"י מעבר איטרטיבי על ערכי המערך.
    - בשלב זה אנחנו מגלים מי היא התיבה בראש המגדל.
  + שורות 5-6 – סריקה של המערך PREV
    - המערך PREV בנוי כך שב PREV[i] ממוקם מזהה התיבה שמעליה הרכיבו את התיבה i.
    - ולכן סריקה בצורה המתוארת לעיל תחזיר את מגדל התיבות (בסדר הפוך).
* חישוב סיבוכיות
  + שורה 1 – מיון n איברים לפי width ~
  + שורה 2 – הקצאת מקום ל2 מערכים כל אחד בגודל n ~
  + שורה 3 – הכנסה לכל אחד מתאי הOPT (מערך בגודל n)
    - במקרה הגרוע אנחנו מזהים את הערך המקסימלי מבין n ערכים שונים לפי 2 קריטריונים, (width אמנם ממויין , אך עדיין מתבצעת איטרציה פרטנית עבור length).
    - ולכן ~
  + שורות 4-5 ~ איטרציה על n איברים וביצוע פעולות קבועות בכל איטרציה ~
  + שורה 6 – סריקה של PREV
    - תנאי העצירה במקרה הגרוע יהיה NULL על ערך האיבר האחרון.
    - ומכיוון שבPREV יש n איברים ~
  + סה"כ 🡨

שאלה 3:

***א****.* התבקשנו להביא 3 פולינומים מדרגה 0 או 1, ולכן בשלב ראשון נוכל להגדיר אותם עם משתנים כך:

*מתוקף הגדרת האיטרפולציה – על הפולינום לקבל את אותם הערכים כל עוד מציבים את אותם הנקודות, ולכן עבור מתקיים:*

*כדאי לקיים את התנאי (1) עבור הביטוי שנתון בשאלה, עלינו לדרוש:*

* + **לאחר הצבה ב נקבל**
  + הצבנו c=1 לפשטות.

*וכדאי לקיים את התנאי (2) עבור הביטוי שנתון בשאלה, עלינו לדרוש:*

* + לאחר הצבה ב נקבל:
  + הצבנו a=1 לפשטות.

מ ו- נקבל:

*כעת יש 2 פתרונות: או . הדרישה איננה הגיונית, שכן בחירת הנקודות שרירותית ונוכל למצוא 2 נקודות שלא מקיימות משוואה זו 🡸 . 🡨 .*

*נציב בדרישה ונקבל:*

נסכם –

*ועל כן מתקיים:*

***ב****.*

* רעיון האלגוריתם
  + בהינתן n נקודות שונות נוכל להשתמש בנוסחאת הנסיגה המתוארת בסעיף הקודם: (הכפלנו מונה ומכנה ב 1-)
  + בצורה זו – אנחנו משתמשים בכל איטרציה בשימושים קודמים כך שנוכל להשתמש בתכנון דינמי בצורה פשוטה עם מנייה עולה של .
  + ניצור אם כן מערך חדש OPT בגודל , כאשר הוא האיטרפולציה של הנקודות .
  + נאתחל כל להיות פולינום האינטרפולציה של הנקודה .
  + נמלא את ערכי המערך לכל , כאשר נחליף את ב .
    - נשים לב שאין משמעות לתאים שמקיימים , ואכן במהלך ריצת האלגו' לא ניגש לאותם ערכים.
  + התשובה הסופית כאמור נמצאת ב . אינטרפולציה של כל n הנקודות.
* תיאור האלגוריתם

1. נאתחל מט' OPT בגודל .
2. לכל
   1. (נקודה אחת – ולכן ערך הY שלה הוא הישר המתאים)
3. לכל :
   1. לכל :
4. נחזיר את

* פסודו-קוד:

1. // declare new metric, items initial to
2. :
3. :

* נכונות האלגוריתם
  1. הפיתרון מסתמך על נכונות סעיף א' ונכונות נוסחאת הנסיגה שנתונה בשאלה.
  2. שורה 2 – מילוי האלכסון הראשי במט' ה OPT
  3. שורה 3 – מילוי המט' באלכסונים
     1. חישוב הערכים מתבצע בצורה אלכסונית, כלומר נמלא את האלכסון שמתחיל ב(1,2) ומסתיים ב ובהמשך בשלב ה k את האלכסון מ (1,k) עד ל
     2. במהלך הריצה – אנחנו לא פונים לערכים שנמצאים "מתחת" לאלכסון הראשי של המט'.
     3. בכל איטרציה בה אנחנו פונים לתאים שנמצאים במקומות ה , ועל כן מדובר בערכים שחושבו בשלב מוקדם יותר של הרקורסיה.
* חישוב סיבוכיות
  1. שורה 1 - הקצאת מקום למערך OPT - זמן קבוע ~
  2. שורה 2 – ריצה על האלכסון הראשי – זמן לינארי ~
  3. שורה 3 – ריצה על n איטרציות
     1. לולאה מקוננת שיכולה לרוץ במקרה הגרוע n-1 פעמים.
        1. בכל איטרציה אנחנו מבצעים עבודה בזמן קבוע (ע"ס הנתון בשאלה פעולות אריתמטריות מתבצעות בזמן קבוע).
        2. ולכן לחלק זה ~
  4. שורה 4 – זמן קבוע ~
  5. סה"כ 🡨

***ג.*** לאחר הצבה של 5 הערכים בפולינום קיבלנו את הנקודות:

|  |  |
| --- | --- |
|  |  |
|  | -2 |
|  | -1 |
|  | 0 |
|  | 1 |
|  | 2 |

נשתמש בטבלת מעקב כדי לתעד את ריצת אלגו' האינטרפולציה מהסעיף הקודם על 5 הנקודות שמצאנו.  
, ולכן נגדיר מט' OPT בגודל .

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Step* | *k* | *i* | *j* |  |
| *1* |  | *1* |  |  |
| *1* |  | *2* |  |  |
| *1* |  | *3* |  |  |
| *1* |  | *4* |  |  |
| *1* |  | *5* |  |  |
| *2* | *2* | *1* | *2* |  |
| *2* | *2* | *2* | *3* |  |
| *2* | *2* | *3* | *4* |  |
| *2* | *2* | *4* | *5* |  |
| *2* | *3* | *1* | *3* |  |
| *2* | *3* | *2* | *4* |  |
| *2* | *3* | *3* | *5* |  |
| *2* | *4* | *1* | *4* |  |
| *2* | *4* | *2* | *5* |  |
| *2* | *5* | *1* | *5* |  |

*מתקיים , כנדרש.*

***מש"ל***

שאלה 4:

**א.** האלגוריתם מחשב את המרחק הקצר ביותר מבין הצומת v אל שאר הצמתים בגרף. כאשר במיקום הA[v] נמצא המרחק המינימלי בין הצומת v ל r.

נוכיח באידוקציה כי בסיום האיטרציה ה-i-ית של הלולאה החיצונית, לכלל הצמתים שנמצאים במרחק **i צמתים** (כלומר נגישים מהצומת r באמצעות i קפיצות בין צמתים) – ערכם המתאים במערך A יכיל את המרחק המינימלי בין r אליהן.

בסיס האינדוקציה: כאשר i=0 , אנחנו באיטרציה הראשונה, מתקיים מקרה הבסיס בו , שכן המרחק המינמלי בין הצומת r לעצמו הוא 0.  
  
הנחת האינדוקציה: באיטרציה ה n, כל הצמתים שנמצאים במרחק n צמתים מצומת הבסיס (r) מעודכנים במערך במיקום המתאים עם ערך זה.

צעד האינדוקציה: נניח עבור n-1 ונוכיח עבור n. נסמן בתור המרחק המינימלי בין הצומת u לv.

* יהי v צומת במרחק n צמתים מr.
* מכיוון שv במרחק n צמתים, הרי שקיים צומת u כלשהו במרחק n-1 צמתים מr דרכו הגענו ל v, באמצעות קשת אותה נסמן ב
  + כאשר תודות ללולאה השנייה – הבחירה ב u וe אינה אקראית – אלא הבחירה המינימלית האפשרית.
  + שכן אם הייתה דרך קצרה יותר – היינו מגיעים אליו באמצעות העידכון בשורה (1ii)
* מכיוון שבצעד קודם של האיטרציה - מובטח לנו מהנחת האינדוקציה שהערך הוא המרחק המינימלי בין הצומת u לצומת v, הרי שהערך הנוכחי .
* בזאת הושלם צעד האינדוקציה.

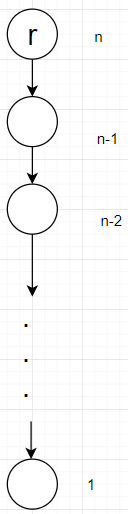
מש"ל

**ב.** לפי הטענה שהוכחה בסעיף הקודם – בכל איטרציה i אנחנו אנחנו מטפלים בכל הצמתים שנמצאים במרחק i צמתים מr, ומכיוון שבכל שלב עלינו לטפל לפחות בצומת אחד (אחרת לא יזוהו שינויים והלולאה הפנימית תעצור את האלגו'). לפיכך – כדי להגיע למספר המקסימלי של איטרציות בלולאה החיצונית – עלינו לטפל בדיוק בצומת אחד בכל איטרציה.

כלומר .

דרך קלה לעשות זאת, היא ע"י גרף של "עץ בו לכל צומת יש בדיוק בן אחד (פרט לעלה בסוף)".  
כאשר הם מסודרים בסדר לקסיקוגרפי הפוך – כלומר באיטרציה הi יסרקו כל הקשתות קודם ורק לאחר מכן תסרק הקשת , מה שיקדם אותנו לפיתרון התרגיל.

אילוסטרציה:

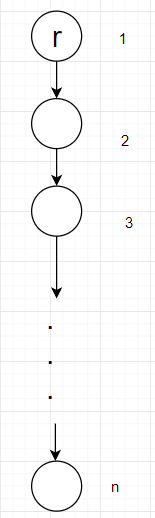


בצורה זו – אנחנו מתחילים מהצומת r ובכל איטרציה נוספת מגיעים לעוד צומת אחד. ובכך מבצעים n-1 איטרציות, נוסיף איטרציה אחרונה לסקירת הצמתים וזיהוי שאין שינויים ונקבל n איטרציות סה"כ.

**ג.** נבצע שינוי בסדר הלקסיקוגרפי על הגרף שבסעיף ב' כך שכל

* האיטרציה הראשונה – תעבור על כל הצמתים לפי כיווני הקשתות ומכיוון שאלו תואמים לסדר הלקסיקוגרפי – אנחנו למעשה סורקים את כל הצמתים כבר באיטרציה זו.
* האיטרציה השנייה של הלולאה החיצונית תעבור שוב על כל הצמתים – מכיוון שלמעשה אין שינויים נוספים שצריך לבצע – אנחנו נעצר בשלב זה.
  + והרי 2 איטרציות כנדרש.

אילוסטרציה:



נשים לב כי בגרף זה יש לנו גם n-1 קשתות, אותו מספר הקשתות שהיה בגרף המתואר בסעיף ב' ולכן מתקיימת הדרישה .

ומתקיימים כל התנאים כנדרש.

**מש"ל**