ממן 14 אלגוריתמים

מאי חמדוהי

206232977

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.
* האלגוריתם

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

* 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***