**ממן 14: אלגוריתמים**

**שאלה 1:**

נציג אלגוריתם למציאת מסלול במחיר מזערי מהשכבה השמאלית ביותר לימנית ביותר.

נחפש מסלול מהשכבה הימנית ביותר בשריג (i=1) לשכבה השמאלית היותר בשריג (i=n). המסלול יכול להתחיל מכל קודקוד - נסמנו 1,k. (), ויכול להסתיים בכל קודקוד – נסמנו n,j (). בסיום מציאת המסלול במחיר המזערי נסיים בשורה ה-j.

הצעדים המותרים הם : (i+1,j), (i+1,j-1), (i+1,j+1) , לכן כדי לשחזר את המסלול בעל המחיר המזערי מהקודקוד n,j נבחר במסלול המזערי מבין שלושת הצעדים המותרים: . המסלול המינימלי מביניהם הוא המסלול האופטימלי מכיוון שאלו הצעדים היחידים שאותם ניתן לבצע – וכלומר המסלול המינימלי חייב לעבור דרכם – ואותו בחרנו. למסלול זה נוסיף את משקל הקודקוד הנוכחי.

לכן נשתמש בנוסחה על כל i מ-1 עד n:

*מכיוון שאלו הצעדים המותרים – אם הקודקוד הנוכחי הוא בקצה – נחרוג מהגודל הנתון של השריג. לכן במקרה זה נחזיר . בסוף נקבל כי הערך בOPT[I,j] הוא המסלול בעל המחיר המזערי מהשכבה הראשונה אל הקודקוד בעל הקוארדינטות I,j.*

***האלגוריתם:***

*יהי c מערך בעל המשקלים של הקודקודים, M מטריצה בגודל nXn.*

1. *לכל :*
   1. *לכל :*
      1. *אם i=1 אז*
      2. *אחרת נשתמש בנוסחה כדי למצוא את המסלול בעל המחיר המזערי. במצב שבו אחד מהצעדים חורגים מהמטריצה נחזיר ∞ כך שעדיין נקבל המינימלי.*
2. *נסמן את n,k כ-*
3. *כל עוד i>0 בצע:*
   1. *נשתמש בצעדים המותרים כדי למצוא את המסלול בעל המחיר המזערי. במצב שבו אחד מהצעדים חורגים מהמטריצה נחזיר ∞ כך שעדיין נקבל המינימלי. נשים ב-k:*
   2. *נסמן את i,k כ-*
4. *נחזיר את סדרת זוגות האינדקסים שהם המסלול בעל המחיר המזערי.*

**נכונות:**

נוסחת הנסיגה: תמיד תעצור מכיוון שאנחנו מקטינים את i ב-1 בכל קריאה ולכן בשלב כלשהו הנוסחה תגיע למצב בו i=1. בכל שלב OPT[I,j] יכיל את המסלול בעל המחיר המזערי (על פי התיאור מעלה) – מכיוון שמותרים רק 3 הצעדים ומתוכם נבחר המסלול המינימלי ביותר. לא ידועה לנו השורה בה המסלול מסתיים ולכן אנחנו מחשבים את כל ב-OPT[j,n] ומתוכם אנחנו מוצאים את המינימום.

נכונות האלגוריתם נובעת ישירות מנכונותה של נוסחת הנסיגה.לכל I,j אנחנו מחפשים את המינימום מבין שלושת הצעדים המותרים – ומכיוון שהלולאה החיצונית מקדמת את i מ-1 עד N -כאשר נגיע לשורה כלשהי תמיד כבר יהיו ערכים בשורה הקודמת לה (בתאים התואמים לצעדים המותרים). לאחר שחישבנו את כל הערכים נשאר למצוא את המסלול המינימלי מבין הקודקודים בשכבה השמאלית ביותר – ולהחזיר את המסלול- שהוא האפשרי המזערי ביותר.

**זמן ריצה:**

שורה 1 – ריצה על מערך דו מימדי, עליו אנחנו מבצעים פעולות קבועות. לכן זמן הריצה הוא . שורה 2 עוברת על השכבה השמאלית ביותר – בגודל n כדי למצוא את המינימום – לכן זמן ריצה הוא . בשאר האלגוריתם אנחנו עוברים על המסלול על מנת לשחזר אותו- מסלול בגודל n – ומבצעים פעולות קבועות לכן זמן הריצה הוא . לכן זמן הריצה הוא .

***שאלה 2:***

*יהי מערך OPT בגודל n (כגודל מספר התיבות).*

*תחילה נמיין את n התיבות בסדר יורד על פי הרוחב שלהן. לאחר מכן נגדיר את OPT(i) כגובה המגדל היציב המקסימלי שבו התיבה ה-i היא בראש המגדל. במצב זה , כל תיבה אחרתj מקיימת – מכיוון שבמגדל יציב אם I בראש המגדל אזי כל התיבות מתחתיה גדולות ממנה ברוחב ובאורך.*

*כדי למצוא את OPT(i) יתכנו 2 מקרים:*

1. *\* מגדל בו התיבה i לבד - וגובה המגדל יהיה h(i)*
2. *\* מצב בו התיבה i תתווסף לראש מגדל אחר קיים אם היא מתאימה. במקרה זה נבדוק רק המגדלים שמעניינים אותנו – המגדלים שבהם התיבה בראשם היא באורך ורוחב גדול יותר מתיבה i – כלומר כאשר j<i (סידרנו את המערך על פי הרוחב בסדר יורד) ו- l(j)>l(i) – ומתוך המגדלים הללו נבחר במגדל בעל הגובה המקסימלי – גובה המגדל יהיה*

*אנחנו מעוניינים בגובה המגדל (היציב) המקסימלי ולכן נבחר המגדל המקסימלי מבין שתי האפשרויות.*

*לכן נגדיר את OPT(i) על פי הנוסחה:*

*כך נעבור על n התיבות כדי למלא את OPT על פי הנוסחה. כמו כן נגדיר מערך נוסף prev\_box שיכיל במקום ה-i א התיבה j שנמצאת בראש המגדל שעליו שמנו את תיבה i – במידה ומקרה 1 נבחר נשים את הערך NIL.*

*לאחר הפעולות הללו גובה המגדל היציב המקסימלי יהיה הערך המקסימלי ב-OPT. מכאן נשחזר את סדר התיבות ונסיים את האלגוריתם. השחזור יתבצע מהסוף להתחלה: תהי i התיבה שנמצאת בראש המגדל היציב המקסימלי (מקום הערך המקסימלי ב-OPT). נסמנה כ- .ניגש למערך prev\_box במקום על מנת לקבל את התיבה מתחתיה, ונסמנה – וכך נמשיך הלאה עד שנגיע לערך NIL במערך prev\_box . בסיום נקבל סדרת אינדקסים שמייצגת את האינדקסים של התיבות של המגד על פי הסדר שבו הן מסודרות במגדל היציב המקסימלי.*

***האלגוריתם:***

*נגדיר מערך OPT בגודל n ומערך prev\_box בגודל n.*

1. *נמיין את התיבות לפי הרוחב שלהן בסדר יורד.*
2. לכל :
   1. נחשב את גובה המגדל המקסימלי על פי הנוסחה שהצגנו, נכניס את התשובה לOPT(i)
   2. *אם הגובה המקסימלי נבחר על פי מקרה 1 – נשים*
   3. *אחרת אם נבחר על פי מקרה 2 – ו-j האינדקס שנבחר – נשים*
3. *נמצא את ערך i מ-1 עד n עבורו מתקיים ונסמנו*
4. *נסמן*
5. *כל עוד :*
6. *נחזיר את הרשימה .*

***נכונות האלגוריתם:***

*חישוב ערכי OPT - הצעד הרקורסיבי של חישוב תיבה משתמש בתוצאות החישוב של תיבות אחרות המקיימות בלבד. לכן, כאשר אנו מחשבים את ערכי OPT מ ועד , לא נתקל באף שלב בניסיון קריאה לערך בOPT שעוד לא איתחלנו אותו. בכל צעד רקורסיבי אנחנו מחפשים את גובה המגדל היציב המקסימלי: התיבה i לבדה במגדל (1) או התיבה i מעל למגדל קיים אחר. המגדלים היחידים שאנחנו יכולים לשים עליהם את התיבה i הם מגדלים שבראשם יש תיבה j המקיימת ו- , - משום שהתיבות ממוינות, שאורך ורוחב התיבה בראש המגדל גדולים מאורך ורוחב התיבה i, ורק אז נוכל להניח את תיבה i על j ועדיין לקבל מגדל יציב. נדרוש שהמגדלים יהיו מקסימליים - ואז נבחר מביניהם את המגדל שייתן לנו גובה מגדל מקסימלי.*

*בשורה 3 אנו בוחרים את הגובה המקסימלי של מגדל יציב מבין כל n האפשרויות שחישבנו עד כה, שבהם התיבה i נמצאת בראש המגדל.*

*בשורות 4-6 אנו משתמשים במערך prev\_box שבנינו, ומצביע עבור כל תיבה i על התיבה שנמצאת לפניה במגדל המקסימלי שמצאנו. לכן, ריצה אחורנית על כל התיבות בצורה זו מאפשרת לנו לסרוק את כל התיבות במגדל היציב המקסימלי (בסדר הפוך). לפיכך, האלגוריתם פותר את הבעיה.*

***זמן ריצה:***

*שורה 1 רצה בזמן של . בשורה 2 אנחנו עוברים על המערך OPT בגודל n – ועל כל אחד מהתאים אנחנו מבצעים במקרה הגרוע קריאות למערך OPT, ולאחר מכן אנחנו מחפשים את המקסימום מבין אלו – כלומר עבור כל תא זמן ריצה של – וסכ"ה שורה 2 רצה בזמן של .*

*שורה 3 רצה בסיבוכיות לינארית. שורות 4-5 ו-7 רצות בסיבוכיות קבועה.*

*שורה 6 רצה על כל התיבות במגדל המקסימלי, ומבצעת פעולות בזמן קבוע. במקרה הגרוע מדובר ב - n תיבות, ולכן רצה בזמן לינארי.*

*סה"כ זמן ריצה של , כנדרש.*

***שאלה 3:***

1. *לכל נסמן ב- את פולינום האינטרפולציה של הנקודות . נגדיר את שלושת הפולינומים q,r,s בצורה הבאה:*

נראה שעבור פולינומים אלה מתקיים:

*נראה שיוויון בין הפולינומים ע"י כך שנראה שיוויון בין כל הנקודות – אם כל ערכי שווים בהתאמה אזי הפולינומים שווים – והשיוויון לעיל מתקיים. תחילה נראה עבור נקודות הקצוות :*

*נציב באגף הימני של הפולינום ונקבל:*

*ידוע כי (על פי ההגדרה) ולכן קיבלנו שיוויון כנדרש.*

*נציב באגף הימני של הפולינום ונקבל:*

ולכן גם פה קיבלנו שיוויון כנדרש. כעת נותר להוכיח את השיוויון עבור כל הנקודות בטווח :

לכל k כך שמתקיים נראה שמתקיים השיוויון לעיל: נציב באגף הימני של הפולינום:

על פי ההגדרה מתקיים: , ולכן מתקיים:

כנדרש.

הוכחנו שעבור שלושת הפולינומים q s r מתקיים השיוויון לעיל. מש"ל.

1. נציג אלגוריתם תכנון דינאמי לבעיית האינטרפולציה, המבוסס על נוסחת הנסיגה מסעיף א.

נגדיר את להיות פולינום האינטרפולציה של הנקודות . M תהיה מטריצה מגודל .

נאתחל את M כך שלכל i, (מתקיים מכיוון שזהו פולינום ממעלה 0 שעובר בנקודה - פולינום האינטרפולציה של הנקודה ). נאתחל את שאר המטריצה M עבור כל ערכי i,j המקיימים לפי נוסחת הנסיגה מסעיף א'. *התשובה שנצטרך היא הערך ב- .*

**האלגוריתם**:

*נגדיר מטריצה בגודל*  כך ש:

1. לכל , .
2. לכל :
   1. לכל :
      1. חשב את לפי הנוסחה הרקורסיבית שמצאנו בסעיף א'
3. נחזיר את .

**ננמק את נכונות האלגוריתם לפתרון הבעיה:**

*נכונותו של האלגוריתם נובעת מנכונותה של נוסחת הנסיגה שהוכחנו בסעיף א.*

*המטריצה M מחושבת בסדר לפי אלכסונים: בשלב 1 אנחנו מחשבים את הערכים עבור האלכסון הראשי, ובשלב 2 אנחנו מחשבים לפי הסדר את האלכסונים בתור מימין (כלומר בתחילת שלב 2 - , נחשב את האלכסון שמתחיל ב- ומסתיים ב. לאחר מכן בשלב בלולאה נחשב את האלכסון שמתחיל בתא ) ומסתיים ב ).* בעת חישוב כש- , אנחנו משתמשים *רק ב- ו , וכש- אנחנו לא משתמשים כלל בערכי M.*

*משום שהערכים שמעניינים אותנו הם רק ערכי M המקיימים , ומשום שחישבנו את ערכי M בסדר נכון, לא נתקל באף שלב בניסיון קריאה לערך בM שטרם איתחלנו.*

**ננתח את סיבוכיות זמן הריצה:**

*שורה 1 רצה בזמן לינארי.*

*בשורה 2 יש לנו לולאה שרצה פעמים, ובתוכה לולאה מקוננת שרצה במקרה הגרוע פעמים. הלולאה המקוננת מבצעת בכל פעם עבודה קבועה (ניתן להניח שפעולות אריתמטיות יחשבו כפעולות אלמנטריות שמחשבים בסיבוכיות קבועה).*

*לכן, שורה 2 רצה ב.*

*שורה 3 רצה בזמן קבוע.*

*סה"כ זמן ריצה של , כנדרש.*

1. יהי . נציב ב-p(x) את הערכים הנתונים:

*כלומר קיבלנו את הנקודות הבאות:*

*נריץ את האלגוריתם מסעיף ב ע"י הצגת טבלת מעקב על המטריצה M ונראה כיצד הפולינום p(x) מתקבל ב-M[1,n]:*

*במקרה זה . נגדיר את M בגודל .*

*שלב 1 של האלגוריתם:*

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

*שלב 2 של האלגוריתם:*

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

*ואכן מתקבל בסוף האלגוריתם: , כנדרש.*

***שאלה 4:***

1. *האלגוריתם מחשב את המסלול בעל המשקל המינימלי מהקודקוד הנתון אל כל קודקוד . את התשובות הוא מאחסן במערך A[v]. במידה ולא קיים מסלול המערך יכיל במקום הקודקוד. נוכיח זאת בשיטה האינדוקציה על מספר האיטרציות של הלולאה החיצונית. נתחיל מהi=0. בתחילה המערך מאותחל ב- .*

*טענת האינדוקציה: בסוף כל איטרציה i של הלולאה החיצונית, המערך A יכיל את המשקל המינימלי מבין כל המסלולים מ-r ל-v שעברנו עליהם עד כה בכל האיטרציות של i.*

*נבדוק את הטענה עבור i=0:*

*עבור i=0, כל המקומות במערך יכילו (כך הגדרנו), מלבד A[r] – ערך זה שווה ל-0, מכיוון שהמסלול מ-r לעצמו בעל המשקל המינימלי הוא 0. עברנו רק על r ולכן הטענה מתקיימת.*

נניח שהטענה נכונה עבור i<n, ונראה שהיא נכונה עבור i=n:

יהי צומת ב-G. נניח כי קיים מסלול מ-r אל v שלא נסרק ב-n-1 איטרציות הקודמות (מכיוון ש-k=n) – ונוכיח שהוא המסלול בעל המשקל המינימלי מבין כל המסלולים שסרקנו עד כה. (כלומר המסלול ה-n הוא המסלול בעל המשקל המינימלי בסוף האיטרציה n הנוכחית.)

נסמן P בתור המסלול מ-r ל-v שעדיין לא נסרק בתום n-1 האיטרציות הראשונות של הלולאה החיצונית, כך ש: . w הוא קודקוד בגרף G כך שמתקיים . מכיוון שהצומת w מוכלת במסלול P – עברנו עליה באיטרציות הקודמות, ולכן הערך במקום A[w] הואהמשקל המינימלי מבין כל המסלולים שכבר נסרקו.

עד כה ידוע לנו על פי הנחת האינדוקציה שהערך במקום-A[w] הוא הערך של המסלול בעל המשקל המינימלי שנסרק עד כה (בין r עד w), ושהמשקל של המסלול P (המסלול שנסרק אחריו) הוא (הקשת המחברת בין w ל-v). במידה ומתקיים : אזי המסלול P הוא בעל משקל קטן יותר מהמסלול בעל המשקל המינימלי שנמצא עד כה – ולכן נעדכן את הערך ב-A[v] במשקל של P. אחרת זהו לא מסלול בעל משקל נמוך יותר ולכן לא נעדכן את הערך ב-A[v].

לכן בסיום האיטרציה הזו – קיבלנו שהערך ב-A[v] מכיל את המסלול בעל המשקל המינימלי, מש"ל.

נראה שגם כאשר לא קיים מסלול הטענה נכונה (מקודם הנחנו כי קיים מסלול):

כאשר לא קיים מסלול . לכן נניח בשלילה שמתקיים ונראה שמתקבלת סתירה:

אם ת אזי קיים צומת ב-V, נסמנו w, כך שקיימת קשת e בין w ל-, v כך שמתקיים: . אי שיוויון זה לא יתקיים אם A[w] = ∞ - כלומר קיים מסלול בעל משקל A[w] מ-r אל w. קיים מסלול מ-r ל-w וקיימת הקשת e מ-w ל-v – ועל כן קיים מסלול בין r ל- v – כלומר קיבלנו סתירה להנחת השלילה! לכן כאשר לא קיים מסלול בין r ל-v הערך במקום A[v] יהיה ∞.

1. המספר המירבי של איטרציות בלולאה החיצונית הוא . סדרת הגרפים המקיימת זאת היא סדרת הגרפים בעלי n קודקודים, בהם אל כל צומת (חוץ מ-r) מגיעה קשת אחת בדיוק ומכל צומת (חוץ מהצומת האחרונה) יוצאת קשת אחת בדיוק, בצורה הבאה:

………..

n

r

רק צומת אחת תתעדכן בכל איטרציה, ולכן לאחר n איטרציות המערך A כבר יכיל בתוכו את כל הערכים הנכונים – כך שבאיטרציה ה-n+1 לא יתבצעו שינויים והאלגוריתם יסתיים.

1. נגדיר את סדרת הגרפים הבאה: כל הגרפים להם n צמתים , כך שאל כל צומת (חוץ מהצומת r) מגיעה קשת אחת בדיוק – כך שכל הקשתות יוצאות מצומת r, בצורה הבאה: סכ"ה n צמתים:

r

באיטרציה הראשונה כל הקשתות יעמדו בתנאי – מכיוון שלכל

…

הצמתים קיים מסלול (קשת) מ-r – ולכן A יכיל בסוף האיטרציה

…

הראשונה עבור כל צומת את משקל הקשת מ-r אליו. באיטרציה השניה

הלולאה לא תבצע כלום – לא יהיה שינוי והאלגוריתם יסתיים.