ממן 14 אלגוריתמים

1.

נגדיר opt(i,j) כמחיר המסלול המזערי מהשכבה השמאלית ביותר אל נקודה (i,j) .

האלגוריתם מחשב מסלול במחיר מזערי מהשכבה השמאלית ביותר לשכבה הימנית ביותר .

כלומר הוא מוצא את min(opt(n,j)) כאשר 1 ≤ j ≤ n.

מכיוון שהצעדים המותרים מ (i,j)

הם (i+1,j+1),(i+,j),(i+1,j-1)

אז כל מסלול מ השכבה השמאלית ל (i,j) כאשר i>1 חייב לעבור באחת הנקודות

(i-1,j+1), (i-1,j) ,(i-1,j-1)

ולכן המסלול המזערי ל (i,j) הוא המסלול המינימלי מבין המסלולים המזעריים לאחת מהנקודות הללו בתוספת הנקודה (i,j)כלומר הגענו לנוסחת הנסיגה

Opt(i,j)=min(opt(i-1,j+1), opt(i-1,j),opt(i-1,j-1)) + c(i,j)

האלגוריתם

Opt(i,j)

for x 🡨 1 to n

for y 🡨 1 to n

if x = 1 than M[y,x] 🡨c(x,y)

else if y = 1

than M[y,x] 🡨min (M[y,x-1], M[y+1, x-1]) + c(x,y)

else if y = n

than M[y,x] 🡨min (M[y,x-1], M[y-1, x-1]) + c(x,y)

else M[y,x] 🡨 min (M[y,x-1],M[y,x-1], M[y+1, x-1]) + c(x,y)

return (M[j,i])

כמו כן בכל פעם שהאלגוריתם יכניס ערך לאחד התאים במערך הוא יכניס מאיזה אינדקס הוא קיבל אותם

ולכן בשחזור פשוט נלך מאינדקס שאותו האלגוריתם מחזיר עד השכבה השמאלית ביותר כלומר עד ש i=1 ונדפיס כל ערך .

נכונות האלגוריתם:

מכיוון שהמסלול המזערי חייב לעבור באחד מ שלושת הנקודות (i+1,j+1),(i+,j),(i+1,j-1) נוסחת הנסיגה נכונה . ולכן אם כשנעבור על המערך לפי נוסחת הנסיגה ובכל שלב ניגש לאיברים שכבר מילאנו הרי שהאלגוריתם נכון .

אכן בn האיטרציות הראשונות y=1 כלומר אנחנו מאתחלים את העמודה הראשונה

לאחר n האירטציות הראשונות בכל איטרציה אנחנו פונים אך ורק לאיברים מהעמודה הקודמת אבל הרי לפי לולאות הfor אם אנחנו נמצאים בעמודה ה i כאשר 1 < i ≤n הרי עברנו כבר על העמודה i-1 . כלומר לפי נוסחת הנסיגה הערך שאנחנו ממלאים אכן נכון .

(בשורה הרביעית עד השורה 8 אנחנו בעצם מבצעים את נוסחת הנסיגה בהפרדה למקרי קצה בהם לא ניתן לגשת לאחד משלושת התאים בנוסחת הנסיגה)

בשורה האחרונה אנחנו מחזירים את המינימום בין המסלולים המזעריים מהעמודה השמאלית ביותר לעמודה הימנית ביותר ולכן קיבלנו את משקל המסלול המזערי מהעמודה הימנית ביותר לעמודה השמאלית ביותר .

בשחזור פשוט נלך מאינדקס שאותו האלגוריתם מחזיר עד השכבה השמאלית ביותר כלומר עד ש i=1 ונדפיס כל ערך . שהרי אם הכנסנו אינדקס של איבר קודם במסלול המזערי ביותר הרי שגם קודקוד זה נמצא במסלול המזערי ביותר.

ניתוח זמן הריצה

בבשלב הלולאות אנחנו בכל איטרציה ממלאים תא אחד במערך ומבצעים מספר קבוע של פעולות . מכיוון שיש במערך תאים הרי שזמן הריצה של שלב זה הוא

כמו כן לבסוף אנחנו עוברים על n תאים כדי למצוא את המינימום מבין המסלולים המזעריים כלומר זמן הריצה הוא

על מנת לשחזר את המסלול אנחנו עוברים בכל איטרציה על אינדקס אחד מכל עמודה שהרי לכל איבר יש אינדקס של האיבר הקודם לו במסלול המזערי כלומר על n תאים

ולכן זמן הריצה הוא O(n)

סה"כ זמן הריצה הוא

שאלה 2

ראשית נמיין את הקוביות לפי רוחבם (במערך M בגודל מספר הקוביות ).

נגדיר Opt(i) כמגדל יציב בגובה מקסימלי שבראשו עומדת הקובייה i .

נשים לב כי במגדל יציב הקובייה i יכולה להיות מונחת אך ורק על מגדלים יציבים שבהם הקובייה (נסמנה j )בראש המגדל מקיימת w(j) > w(i) וגם l(j) > l(i) . (במקרה ואין כאלה אז הקובייה i מגדל יציב בפני עצמו ).

המגדל היציב הגבוה ביותר בראשו הקובייה הוא המגדל הגבוה ביותר מבין המגדלים הללו.

כלומר קיבלנו את נוסחת הנסיגה opt(i) = h(i) + max(opt(j)) כאשר w(j) > w(i) וגם l(j) > l(i) .

האלגוריתם

HighestTower(A[1,…..n])

Sort A[1,…….n] by width (biggest is on index 1 smallest on index n)

Initialize M[1,……..n] 🡨0

Initialize Prev[1 ,…..n] 🡨 0

For i 🡨 1 to n

M[i] 🡨 h(A[i])

For j 🡨 1 to i -1

If w(A[j]) > w(A[i]) and l(A[j]) > l(A[i])

Than if h(A[i]) + M[j]

than M[i] 🡨 h(A[i]) + M[j]

prev[i] 🡨 j

return [M[k]] , k // also return the index

reconstruction(M)

max-index 🡨 max(M[1…..n]) /// maximum between array cell

i 🡨 max-index

while i > 0

print i

i 🡨 prev(i)

נכונות :

בכל שלב הצעד הרקורסיבי פונה רק לאיברים בהם w(A[j]) > w(A[i]) וגם l(A[j]) > l(A[i]) מכיוון שהמערך A ממוין לפי הרוחב מהגדול לקטן ו האיברים המקימים w(A[j]) > w(A[i]) וגם

l(A[j]) > l(A[i]) נמצאים משמאל לi במערך A הרי שלפי לולאת הfor כבר מילאנו את האיברים האלו באיטרציות הקודמות . ואין מצב בו אנחנו ניגשים לאיבר שעוד לא מולא .

נוסחת הנסיגה opt(i) = h(i) + max(opt(j)) נכונה מכיוון שבמגדל יציב מקסימלי שבו ראש המגדל הוא הקובייה i , הקובייה i מונחת על ראש מגדל יציב מקסימלי שבו ראש המגדל הוא קובייה המקיימת w(A[j]) > w(A[i]) וגם l(A[j]) > l(A[i]) או שהקובייה מונחת על הקרקע .

לבסוף כדי למצוא את המגדל היציב הגבוהה ביותר בהכרח הוא מגדל יציב מקסימלי שבראשו עומדת אחת הקוביות 1…….n ולכן בחירת המגדל הגבוהה ביותר מבין המגדלים היציבים המקסימליים שבראשם עומדת אחת הקוביות 1….n היא ההמגדל היציב הגבוהה ביותר.

כדי לשחזר את המגדל מכיוון שבכל שלב בו עדכנו את המגדלים היציבים האופטימליים בכך ששמרנו עבור כל קובייה i שהיא ראש של מגדל יציב גם את הקובייה שמתחתיה קל לשחזר את המגדל היציב ביותר ניקח את המגדל היציב הגבוהה ביותר ועבור כל קוביה בו אנחנו מדפיסים את הקוביה שמתחתיה כשנגיע לתחתית המגדל בוודאי הדפסנו את המגדל כולו . נגיע לתחתית המגדל מכיוון שלקובייה התחתונה מעולם לא עדכנו את prev שלה כלומר prev שלה הוא 0 שהרי אם היינו מעדכנים את ה prev שלה היה מגדל גבוה יותר שבו היא לא הייתה בקובייה התחתונה ביותר .

ניתוח זמן ריצה

מיון המערך A בזמן ריצה O(nlog(n)) .

אתחול שני המעריך בזמן O(n)

עדכון n תאים בערך M שכל תא בו אנחנו מחפשים את מקסימום מבין (במקרה הגרוע ביותר )כל התאים הקודמים לו כלומר עבודה של O(n) סה"כ

לשחזור המגדל אנחנו בהתחלה מחפשים את התא המקסימלי בזמן O(n) ולאחר מכן אנחנו יורדים למטה במורד המגדל קובייה בכל פעם מכיוון שלמגדל יש לכל היותר n קוביות המרכיבות אותו זמן הריצה של שלב זה O(n) .

כלומר סה"כ זמן הריצה של האלגוריתם הינו כנדרש .

שאלה 3

נציג באופן כללי את הפולינומים q(x),r(x),s(x) פולינומים מדרגה 1

P(x) = ax +α

R(x) = bx + β

S(x) = c(x) + γ

נדרוש כי האינטרפלציות של הנקודות i……….j והנקודות i…………….j+1 בנקודה i יהיו שוות

כמו כן נדרוש כי כי האינטרפלציות של הנקודות i………….j+1 והנקודות i+1……..j+1 יהיו שוות בנקודה j+1 .

אם זה לא קורא הרי שהאינטרפלציות אינן תקינות כי אלו אותן נקודות בדיוק

כלומר

וגם

כדי שהדרישה תתקיים :

לכן r(x) =

וגם

ולכן q(x) = x -

כלומר

*בוודאי שונה מה 0 כי מהנתון נובע כי*

*ולכן c=0 כלומר s(x) = γ*

לכן כלומר s(x) =

לסיכום r(x) = q(x) = x - s(x) =

ב.

רעיון כללי :

ניקח מטריצה מסדר nxn נאתחל את האלכסון הראשי כערך של הנקודות בx ואז לפי נוסחת הנסיגה נמלא כל אלכסון בטבלה לפי האלכסון הקודם לו (משמאל)

בסיום נחזיר את הפינה הימנית העליונה של המטריצה . שהיא הפולינום כולו .

נגדיר y(x) כערך y בנקודה x ו A מערך של נקודות

האלגוריתם

Polinom(A)

For t 🡨 1 to n

M[t,t] 🡨 y(A[t])

For a 🡨 2 to n

i 🡨 1

for j 🡨 a to n

use recursion to fill M[i,j]

I 🡨 i+1

Return M[1,n]

נכונות :

מכיוון שנוסחת הנסיגה היא נכונה כאמור על פי סעיף א.

כמו כן מכיוון שהמטריצה מחושבת לפי האלכסונים מהאלכסון הראשי והתקדמות אל האלכסון מימין בכל שלב מכיוון שיש צורך לחשב רק את המשולש הימני של המטריצה כי i≤j נתון .

מכיוון שאיתחלנו את האלכסון הראשי בערך הנכון כי

אז כל שלב שנחשב לפי הרקורסיה מילאנו כבר את האיברים במטריצה שאליהם אנחנו ניגשים . כי אלו איברים מהאלכסון השמאלי להם. נוכיח:

ברקורסיה שנמלא את התא ה M[I,j+1]אנחנו ניגשים אל M[i,j] ואת M[i+1, j+1] והרי מילאנו את איברים אלו באיטרציה הקודמת של הלולאה החיצונית . כי אם הלולאה הקודמת נמצאת באלכסון בו נמצא M[I,j] הרי שאיטרציה הבאה מכיוון שנקדם את a האלגוריתם יעבור על האלכסון המכיל את M[I,j+1] .

כמו כן M[1,n] הוא הפולינום כולו ולכן האלגוריתם מחזיר את ערכי הפולינום .

זמן ריצה

הלולאה הראשונה רצה בזמן O(n)

הלולאה השנייה רצה O(n) פעמים בכל איטרציה היא מחשבת את הרקורסיה O(n) פעמים

חישוב הרקורסיה נעשה בזמן O(n) מכיוון שהרקורסיה מחשבת פולינומים מסדר O(n) בכל שלב ולכן סה"כ האלגוריתם רץ ב .

ג.

נציב את הערכים -2, -1, 0, 1, 2

*קיבלנו את הנקודות*

*נריץ את האלגוריתם מסעיף ב על נקודות אלו*

נגדיר מטריצה M מסדר 5x5 .

נאתחל את האלכסון הראשי בהתאם לאלגוריתם

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | J  i |
|  |  |  |  | **46** |  |
|  |  |  | **2** |  |  |
|  |  | **0** |  |  |  |
|  | **10** |  |  |  |  |
| **98** |  |  |  |  |  |

נבצע את הלולאה השנייה בשלבים נוסיף כל איטרציה בטבלה נפרדת

כאשר K = 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | J  i |
|  |  |  | **-44x-42** | 46 |  |
|  |  | **-2x** | 2 |  |  |
|  | **10x** | 0 |  |  |  |
|  | 10 |  |  |  |  |
| 98 |  |  |  |  |  |

כאשר k = 3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | J  i |
|  |  |  | -44x-42 | 46 |  |
|  |  | -2x | 2 |  |  |
|  | 10x | 0 |  |  |  |
|  | 10 |  |  |  |  |
| 98 |  |  |  |  |  |

כאשר k=4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | J  i |
|  |  |  | -44x-42 | 46 |  |
|  |  | -2x | 2 |  |  |
|  | 10x | 0 |  |  |  |
|  | 10 |  |  |  |  |
| 98 |  |  |  |  |  |

כאשר k=5

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | J  i |
|  |  |  | -44x-42 | 46 |  |
|  |  | -2x | 2 |  |  |
|  | 10x | 0 |  |  |  |
|  | 10 |  |  |  |  |
| 98 |  |  |  |  |  |

כשנחזיר את M[1,n] = M[1,5] נקבל את הפולינום המבוקש.

שאלה 4

1. האלגוריתם מחשב את המסלול הקל ביותר (עלות מסלול מינימלית) מ צומת r לכל שאר הצמתים .

הוכחה:

נוכיח באינדוקציה את הטענה כי באיטרציה i של הלולאה החיצונית האלגוריתם יעדכן רק מסלולים שאורכם לפחות i ומסלולים אלה יהיו הקלים ביותר.

בסיס : עבור i=0 האלגוריתם מעדכן רק את r=0 המרחק מ rל r הוא לפחות 0 והוא הקל ביותר כמובן .

נניח שהטענה נכונה עבור איטרציות הקטנות מ i+1 נוכיח עבור i+1 .

ניקח את התא xהראשון המעודכן באיטצריה ה i+1 הרי שאם המסלול החדש נבחר קצר מ i+1 הרי שהמסלול ללא x בעלות שקטנה מ i ולכן היינו מוצאים אותו כבר לכל היותר באיטרציה ה i-1 ואז האלגוריתם היה בוחר אותו כמסלול הקל ביותר כבר באירטציה ה i . בסתירה לכך שהוא עלות המסלול באירציה ה i גדולה ממסלול זה . כלומר טענת האינדוקציה נכונה .

מכיוון שהמשקלים הם אי שליליים אורך המסלול הקל ביותר הוא לכל היותר n-1 מסלול ללא מעגלים . אז באיטרציה ה n בהחרך לא ישתנה כלום והאלגוריתם יעצור . ולכן בכל תא נקבל את את העלות המינימלית של מסלול מ r לצומת שהתא שייך אליה.

1. לפי א יש לכל היותר n איטרציות שמבצע האלגוריתם כלומר B(n) ≤ b=n נוכיח שאכן קיימת סדרת גרפים שהאלגוריתם פותר ב n איטרציות .

סדרת הגרפים שנציג הם גרפים שלכל צומת בגרף יש בדיוק בן 1 מלבד הצומת הרחוק ביותר מ r הצמתי מסודרים בסדר לקסיקוגרפי הפוך

n

;

1

n-2

n-1

r

מתחילים ב r בכל איטרציה אנחנו מגיעים לצומת חדש שלא היה מצאנו לו מסלול עד כה ל R לכן אנחנו עושים n-1 איטרציות ועוד איטרציה אחת בה אנחנו לא משנים כלום כלומר סה"כ n איטרציות ולכן B(n)= n

1. סדרת הגרפים דומה מאוד לסדרת הגרפים הקודמת רק שכעת הסר הלקסיקורפי הפוך לסדר מהסדרה הקודמת

n

r

2

3

n

כעת האלגוריתם יעדכן את כל המסלולים באיטרציה הראשונה כי יש בדיוק אורף מסלול אפשרי ובכל שלב כבר עודכן הצומת הקודם בגלל הצורה בו הגרף מסודר ,האלגוריתם יבצע איטרציה נוספת שלא משנה את המערך A ולכן האלגוריתם יסתיים .