**ממן 14:**

**שאלה 1:**

תיאור האלגוריתם:

נבנה מערך בגודל NXN כאשר כל תא במערך מיצג קודקוד בשריג. נרוץ על התאים מהשכבה השמאלית לימנית ובכל תא נכניס את מחיר המסלול המזערי מהשכבה השמאלית ועד התא הנוכחי. לאחר מכן נרוץ על כל התאים בשכבה הימנית ונחזיר את מחיר המסלול המזערי של התא עם המחיר הקטן ביותר כמחיר המסלול המזערי מהשכבה השמאלית לימנית.

For j1 to n

A[1,j]c(1,j)

For i2 to n

For j1 to n

A[I,j]min(A[i-1, j-1], A[i-1, j], A[i-1, j+1] +c(I, j)

מחשב את המינימום של האינדקסים המוגדרים בלבד. לכן אם לדוגמא J=N אז לא יכליל בחישוב של המינימום את A[i-1, j+1] וכן הלאה במקרה ש J=1 או במקרה שj=1 וגם J=N. //

for j

min

return min

זמן ריצה:

האלגוריתם רץ על מערך בגודל NXN ומבצע על כל תא מספר סופי של פעולות. לכן זמן הריצה יהיה

הוכחת נכונות:

הלולאה הפנימית מכניסה בA[I,j] את המינימלי האפשרי (במידה ומוגדר) מבין A[i-1, j-1], A[i-1, j], A[i-1, j+1] בתוספת המחיר c(I, j) לכל תא j בשכבה i.

שמורת לולאה חיצונית:

בסיום כל איטרציה, השכבה הi תכיל בכל תא את מחיר המסלול המזערי מהשכבה השמאלית עד לתא הנוכחי.

עבור i=1 הדבר מתקיים כיון שכבר נמצאים בשכבה השמאלית ביותר ולכן רק הכנסנו את מחיר הקודקוד הנוכחי.

נניח שהטענה מתקיימת עבור השכבה הi=k.

עבור השכבה הi=k+1 :

בכל תא בשכבה הk יש את מחיר המסלול המזערי מהשכבה השמאלית ועד לאותו התא לפי הנחת האינדוקציה. כדי להגיע לתא כלשהו j בשכבה הk+1 יש בדיוק 3 דרכים (לכל היותר, במידה והן מוגדרות) להגיע אליו לפי הצעדים שהוגדרו בשאלה. דרך התאים: (k, j), (k, j-1), (k, j+1).

לכן מחיר המסלול מזערי מהשכבה השמאלית לתא הנוכחי יהיה המינימלי מבין המחירים בתאים שלעיל ועוד המחיר של הקודקוד הנוכחי. כלומר, קבלנו שבכל תא בשכבה הk+1 יש את אורך המסלול המזערי מהשכבה השמאלית ועד לאותו התא.

כאשר I=N הכנסנו בשכבה הN לכל תא את מחיר המסלול המזערי המינימלי מהשכבה השמאלית ועד לתא הנוכחי.

כעת, כאשר נרוץ על התאים בשכבה הN ונחפש את המינימלי מבניהם, נמצא את מחיר המסלול המזערי מהשכבה השמאלית ועד השכבה הימנית.

**שאלה 2:**

נמיין את התיבות לפי האורך שלהן. נשתמש בi לסמן את המיקום היחסי של התיבה ביחס לאורך, כלומר עבור התיבה הארוכה ביותר i=n ועבור התיבה הקצרה ביותר i=1.

יהי w מסמן רוחב כלשהו של תיבה.

כמו כן נמיין את התיבות לפי הרוחב שלהן. נשתמש בj כדי לסמן את המיקום היחסי של התיבה ביחס לרוחב. כלומר עבור התיבה הרחבה ביותר j=n ועבור התיבה הצרה ביותר j=1.

נגדיר את נוסחת הנסיגה שלהלן:

אם i=0:

אם :

אחרת:

כאשר Opt מחשב את הגובה המקסימלי של מגדל המורכב מתיבות מ1 ועד i כאשר הרוחב של כל תיבה שמוסיפים למגדל יהיה לכל היותר ברוחב w.

נגדיר max\_w להיות הרוחב המקסימלי מבין רחבי התיבות.

הקריאה הראשונה תבצע:

נוכיח את נכונות נוסחת הנסיגה.

כאשר i=0 אין שום תיבה ולכן הגובה של המגדל יהיה 0.

כאשר :

הרוחב המקסימלי של תיבות שאפשר להוסיף למגדל הוא W ולכן אי אפשר להוסיף את התיבה i למגדל וממילא המגדל האופטימלי הוא באותו גובה של המגדל האופטימלי של התיבות מ1 עד i-1.

אחרת אפשר להוסיף את התיבה הi למגדל או לא להוסיף אותה.

אם נוסיף אותה, גובה המגדל האופטימלי יהיה גובה המגדל של כאשר הגבלנו את רוחב התיבות המרכיבות המגדל המורכב מתיבות מ1 ועד i-1 להיות לכל היותר רוחב התיבה שהוספנו. כיון שאורך התיבה הi גדולה יותר מארכי התיבות מ1 ועד i-1 ולכן אם נוסיף תיבה מהתיבות 1 עד i-1 זוהי תהיה הוספה לא חוקית כיון שתהיה תיבה שהאורך שלה גדול מהתיבה השניה, אבל הרוחב שלה קטן מהתיבה השניה. W(i)<=w כיון שאחרת לא היינו מגיעים למשפט הנ"ל, ולכן לא סתרתי את הגבלת הרוחב הראשונית.

אם נבחר שלא להוסיף אותה, הגובה האופטימלי של המגדל יהיה הגובה האופטימלי של המגדל המורכב מתיבות 1 עד i-1 עם הגבלת רוחב הנתונה מכבר.

כלומר, המקסימלי מבין 2 המגדלים שתארנו הוא הגובה של המגדל המקסימלי העונה על הנדרש.

תיאור האלגוריתם:

נבנה מטריצה A בגודל NXN. i=0 יסמל מצב שאין תיבות וj=0 יסמל מצב שהרוחב המקסימלי הוא 0.

נאתחל את העמודה הi=0 והשורה הj=0 להיות 0.

לאחר מכן נרוץ על העמודות מi=1 ועד n ונמלא אותן בהתאם לנוסחת הנסיגה.

לבסוף גובה המגדל המקסימלי יהיה A[n, n].

Sort boxes by length

Sort boxes by width

return A[n, n]

indexOf(w) מחזיר את האינדקס הj שמציין את הרוחב W. בהכרח כיון שנתון שואינדקס J מייצג רוחבים בסדר עולה.

נוכיח באינדוקציה על i.

כאשר i=0 אתחלנו את המערך ב0.

נניח שהעמודה הk מלאה בערכים תקינים,

עבור העמודה הk+1

העמודה הk+1 משתמשת בעמודה הk כדי למלא את ערכיה לפי נוסחת נסיגה שהוכחנו לעיל. כיון שהערכים בעמודה הK נכונים, הרי שגם הערכים בעמודה הk+1 נכונים.

אם כן, כאשר k=n העמודה הn תתמלא בערכים נכונים.

הערה: המונח עמודה מתייחס לתאים מהשורה ה1 עד השורה ה0 לא כולל השורה ה0 (שהוגדרה מראש.)

במקום הA[n,n] נקבל את הגובה האופטימלי של מגדל שיכול להשתמש בכל התיבות ללא הגבלה של רוחב התיבות. (הערך של J במקרה זה הוא n שמייצג את הרוחב הגדול ביותר של תיבה. )

זמן ריצה:

מיון תיבות ייקח זמן של . הריצה על המערך ומילויו ייקח זמן של . את indexOf() ניתן לממש בזמן קבוע (ע"י מערך עזר) ולכן זמן הריצה יהיה .

**שאלה 3:**

א.

לכן:

לכל :

כאשר

ב.

נגדיר מערך בגודל nXn.

בכל תא במערך A[I,j] נשמור את .

תיאור האלגוריתם:

האלגוריתם רץ על מעלת הפולינומים (מ0 ועד n-1) ובכל תא שומר את .

Return A[1,n]

נכונות:

נוכיח באינדוקציה על

עבור פולינומים ממעלה 0 האלגוריתם מבצע את הנדרש כיון שהגדרנו את הפולינומים ממעלה 0 להיות .

נניח האלגוריתם מוצא את הפולינומים הנכונים ממעלה k.

עבור מעלה k+1 נשתמש בפולינומים ממעלה k לחישוב כל פולינום ממעלה k+1. כיון שהפולינומים ממעלה k חושבו כנדרש הרי שגם הפולינומים ממעלה k+1 יחושבו כנדרש.

עבור פולינומים ממעלה n-1 נחשב את הפולינום שמתבסס על פולינומים ממעלה n-2 המחושבים כהלכה ונקבל את הפולינום הרצוי.

זמן ריצה:

האלגוריתם ממלא חצי מערך בגודל nXn בזמן קבוע ולכן זמן הריצה שלו הוא) .

ג.

נסמן את הפולינומים בוקטורי מקדמים.

עבור פולינומים ממעלה 0: (לפי ההגדרה בשאלה.)

כעת נשתמש נציב נוסחה שקבלנו בסעיף א' ונקבל:

עבור פולינומים ממעלה 1:

עבור פולינומים ממעלה 2:

עבור פולינומים ממעלה 3:

עבור פולינומים ממעלה 4:

הרי שקבלנו בעזרת האלגוריתם את המקדמים של הפולינום המקורי כנדרש.

**שאלה 4:**

1. האלגוריתם מחשב את המרחק המסלול הקצר ביותר מr לנקודה כלשהי. כאשר אין מסלול מr לצומת כלשהי המרחק יהיה איןסוף.

לכל צומת v כלשהי, ערכה יהיה מחיר המסלול הקצר ביותר מr לv שעובר דרך צמתים שהיו מאותחלים בתחילת האיטרציה הקודמת. אם קיים מסלול כלשהו שעובר דרך קשת כלשהי והצומת u עודכנה במהלך האיטרציות הפנימיות שקדמו לאיטרציה הe מחיר המסלול הקצר ביותר יחושב לפי

הוכחה:

נוכיח כי כל עידכון שיבוצע בלולאה הפנימית לעולם לא יקטן ממחיר המסלול המינימלי האפשרי.

עבור ההרצה הראשונה של שורת הפקודה בלולאה הפנימית ברור כי הדבר מתקיים כי אם יש עדכון נכניס את המחיר של המסלול מr לצומת המעודכנת.

נניח הטענה מתקיימת עבור הרצה k כלשהי של שורת הפקודה (ללא קשר האם ההרצה היא באיטרציה הראשונה של הלולאה החיצונית או באיטרציות עוקבות)

עבור הרצה k+1 כלשהי ערכי המסלולים שבצמתים לא קטנים מערך המסלול הקצר ביותר, ולכן גם אם נעדכן את מחיר הצומת הנוכחי, נעדכן אותו למחיר שלא יהיה קטן מערך המסלול הקצר ביותר.

בהרצה האחרונה של שורת פקודה זו גם כן העדכון לא יקטן ממחיר המסלול הקצר ביותר כיון שאין צומת שמחירו קטן ממחיר המסלול הקצר ביותר.

נגדיר שמורת לולאה ללולאה החיצונית.

בתחילת כל איטרציה, נאמר כי הצמתים המאותחלים (שערכם בתא המייצג אותם לא איןסוף) יהיו קטנים או שווים למרחק בין הצמת הנוכחית וצומת r אם נתייחס רק למסלולים שעוברים דרך צמתים שהיו מאותחלים בתחילת האיטרציה הקודמת.

כמו כן נאמר כי מחיר מסלול מr לצמת כלשהו לעולם לא יקטן ממחיר המסלול המינימלי.

עבור האיטרציה הראשונה, הצמת היחיד שמאותחל הוא r והוא שווה 0 כי המרחק מr לr הוא 0.

נניח שמורת הלולאה מתקיימת באיטרציה הk.

עבור האיטרציה הk+1:

בכל ריצה בלולאה הפנימית, מעדכנים את הערך של צומת מסוים כאשר קיים קשת כלשהו שהמסלול דרכו מr קצר יותר מאורך המסלול הנוכחי. העדכון הוא לאורך של המסלול שעובר דרך הקשת הנ"ל. העדכון מתבסס על כך שהעדכונים שנעשו באיטרציה החיצונית הk היו חוקיים. ייתכן והיו עדכונים נוספים בגלל שינויים במחירי המסלולים של צמתים נוספים במהלך האיטרציה הנוכחית, אולם עדכונים אלו לא יעלו את המחיר של המסלול. כמו כן, כפי שהוכנו לעיל, הם לעולם לא יהיו קטנים מאורך המסלול המינימלי. לכן שמורת הלולאה מתקיימת.

כאשר ישנה איטרציה ללא כל שינוי, לפי שמורת הלולאה מחירי הצמתים המעודכנים הם מחיר המסלול המינימלי(או קטן ממנו.) שיכול לעבור דרך כל הצמתים שהיו מאותחלים באיטרציה הקודמת, שזה בדיוק הצמתים שמאותחלים באיטרציה הנוכחית. כיון שרצנו על כל הקשתות ולא מצאנו עדכונים נוספים לבצע, הרי שאין יותר צמתים קשירים שלא אותחלו וכיון שלעולם מחיר המסלול של צומת כלשהו יהיה גדול או שווה למחיר המסלול המינימלי לכן הצמתים מכילים את מחיר המסלול המינימלי האפשרי.

1. בכל איטרציה כל הצמתים הקשורים לצמתים מעודכנים מהאיטרציה הקודמת מתעדכנים לערך המסלול המינמלי מבין הצמתים שעודכנו באיטרציה הקודמת. (או ערך מסלול קטן יותר שעובר דרך צמתים שעודכנו באיטרציה הנוכחית...)

לכן בכל איטרציה מעודכן לכל הפחות צומת אחד שהיה שווה אינסוף קודם לכן לבד מהאיטרציה האחרונה בה לא נעדכן דבר.

תתכן איטרציה אחת לפני האחרונה שרצה ומעדכנת כאשר אין צמתים ששווים לאיןסוף רק אם היה לפחות איטרציה אחת בה עדכנו יותר מצומת אחד.

יוצא שבגרף בעל n צמתים יהיו לכל היותר n איטרציות.

דוגמא לגרף המקיים זאת:

יהי גרף כלשהו בעל n צמתים. נגדיר את הגרף כחוט כך שלכל צומת בדיוק קשת אחת היוצאת ממנה וקשת אחת הנכנסת אליה לבד מr ממנה רק יוצאת קשת ומv הקשת האחרונה בחוט אליה רק נכנס קשת. נגדיר את סדר הצמתים כך שv תהיה הצומת הראשונה, הצומת שלפני v תהיה השניה וכך הלאה כשהצומת האחרונה היא r.

באיטרציה הראשונה נשנה רק את הצומת לפני r. באיטרציה השניה רק את הצומת לפני הצומת ששננו בפעם שעברה וכך הלאה. לבסוף מתברר שיהיו n איטרציות.

1. הגרף החדש שנגדיר יהיה זהה לגרף שהגדרנו בסעיף קודם לבד מהעובדה שסדר הקשתות יהיה הפוך ולכן באיטרציה הראשונה יעודכנו כל הקשתות לערך הנכון ובאיטרציה השניה והאחרונה לא יעודכן כלום.