ממן 14 אלגוריתמים

# שאלה 1

## תיאור האלגוריתם

האלגוריתם יתבסס על תכנות דינמי. בהתחלה, האלגוריתם יעבור על כל הקודקודים מימין לשמאל (כלומר מהסוף להתחלה) ויחליט לכל קודקוד מהו הכיוון האופטימלי לתנועה ומה המחיר שיתקבל מכך. את זה הוא יכול לחשב מתוך המחירים האופטימליים של שלושת הקודקודים שהו יכול לעבור אליהם- כיוון למעלה למטה וללא שינוי.

לאחר חישוב המשקלים והכיוונים האופטימליים, האלגוריתם יבחר את הקודקוד בעמודה השמאלית שעבורו המחיר שחושב כבר קודם הכי קטן. זהו הקודקוד שממנו מתחיל המסלול האופטימלי הכולל. לבסוף, האלגוריתם ייצור את המסלול עצמו החל מהקודקוד הנבחר לפי מטריצת הכיוונים.

//find the shortest path from vertex (i,j) to the rightmost column in the matrix A

//Put the direction in D[i,j]. Also put the cost and put it in C[i,j]

shortest\_path\_from(A, D, C, i, j):

//find the actual path from the directions matrix from some starting point [i,j] on the leftmost column.

//adding arrays is defined as appending all the items in the second array to the first.

//The actual algorithm- find the shortest path from the left column to the right column

הערה- אין כאן התייחסויות למקרי קצה (למשל כאשר j=1 לא ניתן לגשת לj-1 כי הוא מחוץ לגבולות. לשם פשטות, האלגוריתם מוגדר לא לבצע אף חישוב מחוץ לגבולות ולהתעלם מהם.

## הוכחת נכונות

נוכיח באינדוקציה על i(כאשר הבסיס הוא n ומגיעים ל1) שלכל עמודה i האלגוריתם יוצר כיוון שמצביע על המסלול האופטימלי לכל j

בסיס האינדוקציה: i=n, האלגוריתם נכון- המסלול הכי קצר לכל קודקוד שם כולל רק את המחיר של עצמו.

נניח שהאלגוריתם תחזק את C,D כמו שצריך עבור i ונראה שהאלגוריתם יעבוד עבור i-1:

יהיה j כלשהו. אז המסלול הקצר ביותר מ לעמודה הימנית חייב לעבור באחד מהקודקודים האלה:

בהנחה שהמסלול שנמצא עבור כל אחד מהקודקודים האלו אופטימלי ושהמשקל שלהם מתוחזק כמו שצריך, אז המסלול הקצר ביותר מ יהיה זה שעובר דרך הקודקוד מתוך השלושה שהמסלול האופטימלי שלו הכי קצר. האלגוריתם בוחר בדיוק את הכיוון שיוצר את זה, ומתחזק את המשקל שלו כראוי (פשוט לחבר את המשקל של הקודקוד למשקל של המסלול החל מהקודקוד הבא).

כלומר, הוכחנו שלכל i ולכל j הכיוון שהאלגוריתם שם עבורו בD מכוון לקודקוד שהוא חלק מהמסלול האופטימלי ממנו לעמודה הימנית, ושהמשקל שהוא שם עבורו בC הוא המשקל של אותו מסלול.

התוצאה מכך היא שכאשר בוחרים את הj עבורו הכי קטן מקבלים את המסלול הקודקוד שממנו מתחיל המסלול האופטימלי מצד שמאל לצד ימין- כי זהו המסלול הכי קצר שמתחיל ב(1,j) ולכל j אחר המסלול הקצר ביותר ארוך יותר.

לבסוף, האלגוריתם find\_path מקבל את הקודקוד שהמסלול האופטימלי מתחיל ממנו ומחזיר את המסלול ממש (מערך של קודקודים). הוכחה באינדוקציה: בסיס: כאשר המסלול הוא אכן רק הקודקוד הנוכחי. צעד: המסלול מקודקוד כלשהו הוא הוספה של הקודקוד הנוכחי למסלול מהקודקוד שאליו המסלול עובר כרגע (לפי D), וזה אכן מה שאלגוריתם מבצע.

לסיכום, האלגוריתם אכן מוצא לכל קודקוד את הכיוון האופטימלי למסלול מזערי לקודקוד כלשהו בעמודה הימנית, ואז בוחר את הקודקוד שממנו יוצא המסלול האופטימלי מהעמודה השמאלית לעמודה הימנית, ואז מוצא ומחזיר את המסלול הזה. המסלול שהוא מחזיר הוא אכן המסלול האופטימלי.

## זמן ריצה

האלגוריתם של מציאת הכיוון האופטימלי shortest\_path\_from עובד בזמן ריצה קבוע כי כל הפעולות בו הן אלמנטריות בודדות.

לכן, לולאת הfor הכפולה באלגוריתם רצה ב כלומר ב פעולות אלמנטריות.

יצירת המערכים כמובן לוקחת פעולות אלמנטריות.

לולאת הfor השנייה באלגוריתם רצה n פעמים ומבצעת כמות קבועה של פעולות כלומר רצה ב פעולות אלמנטריות.

לבסוף, זמן הריצה של find\_path(D,C,i,j) הוא מכיוון שבכל איטרציה קוראים לפעולה עם i גדול ב1 ועוצרים בi=n כלומר לכל היותר n איטרציות, כלומר לכל היותר פעולות אלמנטריות.

סה"כ זמן הריצה כנדרש.

# שאלה 2

נתכנן אלגוריתם שמקבל את הממדים של התיבות בתוך שלושה מערכים, מערכך אורכים, רוחבים וגבהים:



4.1.1

5.

6.

6.1.

6.1.1

6.1.2

7.

8.

8.2

8.2.1

8.1

8.2



3.1.1.

3.2.

3.2.1.

3.3.

3.3.1.

3.4

3.4.1

3.4.2

4.

5.

## תיאור האלגוריתם

אלגוריתם העזר tallest מחשב לתיבה כלשהי את המגדל הכי גבוה שניתן לבנות **עליה** באופן רקורסיבי. לכל תיבה שאפשר לשים על התיבה המקורית מחשבים את המגדל הכי גבוה שניתן לבנות עליה, ואז בוחרים את התיבה שמאפשרת את המגדל הכי גבוה וכן שומרים את גובה המגדל הזה (ועוד הגובה שלה) לצורך חישובים עתידיים. כך יוצא שבודקים את הגובה המקסימלי מכל תיבה בדיוק פעם אחת. לבסוף, מחפשים את התיבה שעבורה הגובה המקסימלי מקסימלי, ואז בונים את המגדל הזה לפי הערכים שכבר מצאנו.

## הוכחת נכונות

נוכיח באינדוקציה שtallest(i,…) מציב בP[i] את הגובה של המשקל הכי גבוה שניתן לבנות כך שלבנה i היא הבסיס ובB[i] את הלבנה הבאה במגדל הזה. האינדוקציה היא על כל הלבנים שניתן להניח על הלבנה הi. זוהי אינדוקציה תקפה מכיוון שהיא לא מעגלית (אין שתי לבנים שיכולות להיות גם אחת מעל השנייה וגם השנייה מעל הראשונה), ומכיוון שהיא סופית (היא עוצרת כאשר מגיעים ללבנים שאי אפשר להניח עליהן אף לבנה).

בסיס: לבנה שלא ניתן להציב עליה אף לבנה אחרת- אז בלולאת הfor לכל לבנה יתקבל שהיא רחבה יותר או ארוכה יותר (אחרת היה אפשר להציב אותה עליה), ולכן לולאת הfor לעולם לא תעשה כלום. בסופו של דבר, m יהיה 0. לכן P[i] יהיה h[i], וזהו אכן גובה המגדל הגבוה ביותר האפשרי עם i בבסיס (מגדל שמורכב רק מi).B[i] נשאר -1 מכיוון שאין לבנה במגדל הזה.

צעד: נניח שעבור כל לבנה j שניתן להניח על i מתקיים שtallest(j,…) מציב בP[j] את גובה המגדל הגבוה ביותר שניתן לבנות עם j בבסיס ונראה שtallest(i,…) מציב בP[i] את גובה המגדל הגבוה ביותר שניתן לבנות עם i בבסיס. האלגוריתם tallest(i,…) יתחיל בלקרוא לtallest(j,…) לכל j שניתן להציב על i, או ישירות או בדרך עקיפה דרך קריאות רקורסיביות נוספות (כי הדרך היחידה שיהיה בP[j] ערך זה אם tallest(j,…) סיימה את ריצתה, והאלגוריתם מריץ tallest(j,…) לכל j שעבורו אין ערך בP[j] עדיין). כלומר, כשמגיעים לשורה 3.4 עם j כלשהו הערך בP[j] כבר נכון לפי הנחת האינדוקציה. לכן, בסוף הלולאה של שורה 3 m הוא גובה המגדל הגבוה ביותר שבסיסו ניתן להצבה על i. לכן, הוא אכן גובה המגדל הגבוה ביותר עם בסיס i וזה הערך שמוצב בP[i]. כמו כן, m יהיה מספר הלבנה הבאה במגדל עם הגובה המקסימלי (זו שמוצבת על i במגדל הזה).

הוכחנו שבסוף הרצת tallest(i,…) עבור i כלשהו, הערך הוא גובה המגדל הגבוה ביותר עם בסיס i ולכל לבנה j בו B[j] הוא הלבנה הבאה במגדל (או -1 אם זו הלבנה האחרונה).

*באלגוריתם הראשי, בסוף הלולאה בשורה 4, tallest(i,…) רץ לכל i בדיוק פעם אחת. יש הוכחה מפורטת לכך בחלק של ניתוח זמן הריצה. לכן לכל לבנה P[i] מכיל את גובה המגדל הכי גבוה שניתן לבנות עם בסיס i וB[i]מכיל את הלבנה שצריך לשים על i כדי לקבל את המגדל הגבוה ביותר האפשרי.*

*לכן, בסוף ריצת הלולאה בשורה 6, מכיל את הלבנה שהמגדל הכי גבוה איתה כבסיס הוא הכי גבוה, כלומר זהו המגדל הכי גבוה שניתן לבנות באופן כללי בסט הלבנים שניתן כקלט. בלולאה בשורה 8, כל לבנה כזו מוכנסת לרשימה עד שמגיעים ללבנה -1 כלומר לסוף המגדל. לכן הרשימה שמוחזרת היא רשימת הלבנים שיש להניח אחת על השנייה כדי לקבל את המגדל הכי גבוה שניתן.*

## זמן ריצה

נבחין כי לכל לבנה i, הפונקציה tallest(i,…) נקראת **בדיוק** פעם אחת. זה נובע משלוש סיבות:

1. tallest(i,…) לא גורמת לקריאה נוספת של עצמה עד לסיום ריצתה- tallest עבור i כלשהו קוראת לtallest נוספים רק עבור קוביות שניתן להציב על i. כל קריאה רקורסיבית כזו קוראת שוב רק ללבנים שניתן להניח על הלבנים שכבר נקראו. בכל מקרה, לא ניתן להציב את i על אף לבנה שאפשר להציב על i. לכן ההרצה עצמה של tallest(i,…) לא תגרום בעצמה לעוד הרצה של tallest(i,…).
2. לאחר שtallest(i,…) סיימה את ריצתה, לא יקראו לה ממקום אחר יותר אף פעם- כשtallest(i,..) מסיימת את ריצתה בפעם הראשונה, היא מציבה בP[i] ערך חיובי (ובפרט שונה מ-1). אף קריאה אחרת לא יכולה לשנות את P[i]. כמו כן, בכל מקרום שבו קוראים לtallest עם ערך כלשהו, בודקים קודם שיש בP במקום הזה ערך -1. לכן ברגע שtallest הסתיימה, לא יקראו לה יותר אף פעם.
3. tallest(i,...) תיקרא לפחות פעם אחת- כאמור, המקום היחיד שיכול לשנות את הערך P[i] הוא הפונקציה tallest(i,…) עצמה. לכן, לולאת הfor באלגוריתם הראשי בשורה 4 תקרא לtallest לכל i לפחות פעם אחת, כי לכל i אם P[i] שונה מ-1 (כלומר אם הפונקציה לא נקראה), היא קוראת לtallest עם ה-i הזה.

סה"כ, מ1+2 נובע שלכל i הפונקציה נקראת לכל היותר פעם אחת, ומ3 נובע שלכל i היא נקראת פעם אחת, לכן לכל i הפונקציה tallest(i,…) נקראת בדיוק פעם אחת.

כמות הפעולות האלמנטריות בtallest היא (לא כולל הקריאות הרקורסיביות). כלומר, אם היא נקראת בדיוק n פעמים,, אז זמן הריצה הכולל של לולאה 4 הוא .

הלולאה בשורה 6 והלולאה בשורה 8 רצות שתיהן n פעמים כלומר .

כל שאר הפעולות הן אלמנטריות ולכן סה"כ כמות הפעולות האלמנטריות .

# שאלה 3

## סעיף א

הוכחה:

בהנחה ש הוא האינטרפולציה של ו הוא האינטרפולציה של , אז צריך להראות ש לכל .

1. *אם :*
2. *אם :*
3. *אחרת (x הוא בטווח שנכון גם ל וגם ל):*

*כלומר זהו אכן פולינום האינטרפולציה.*

## סעיף ב

## תיאור האלגוריתם

האלגוריתם מקבל רשימה של קואורדינאטות x ורשימה של קואורדינאטות y ומחזיר רשימה של מקדמים כאשר המקדם הראשון הוא החופשי () והאחרון הוא המקדם שם . הפולינום שנוצר הוא האינטרפולציה של הנקודות שהוכנסו.

האלגוריתם קורא לשיטת עזר שמחשבת פולינום מתוך תת רשימה של נקודות (מתוך הנקודות המקוריות) על פי הנוסקחה בסעיף א'. האלגוריתם משתמש בתכנון דינמי ושומר את התוצאה של כל פולינום של כל תת סדרה שהוא חישב כבר, כלומר הוא מחשב לכל תת סדרה את הפולינום שלה רק פעם אחת.

## הוכחת נכונות

נוכיח באינדוקציה על גודל תת הסדרה שבודקים בכל איטרציה, כלומר על , שinterp(xs,ys,i,j,M) מחזיר את מקדמי הפולינום שעובר בתת הרשימה מאינדקס i עד j של xs,ys.

בסיס: נקודה אחת: i=j: אז האלגוריתם יחזיר בשורה 2.1. כנדרש.

צעד: נניח שעבור כל תת קבוצה בגודל k האלגוריתם עובד ונראה שהוא עובד על קבוצה בגודל k+1. בשורה 3 האלגוריתם יחשב אינטרפולציה של פולינום באורך k (כי j הקוטן ב1), או ישלוף את התוצאה של הפולינום הזה שכבר חושבה. לפי ההנחה, זהו אכן פולינום שעובר בכל הנקודות מi עד j-1. באותו אופן, בשורה 5 מקבלים פולינום שעובר בכל הנקודות מi+1 עד j. בשורה 8 ו-9 מחשבים פולינום, שלפי סעיף א, עובר בכל הנקודות מi עד j, לפי כפל וחיבור פולינומים ב"brute force" (פתרון אינטואיטיבי). כלומר מוחזרים מקדמים של פולינום שעובר בכל k+1 הנקודות מi עד j.

לסיכום, האלגוריתם מחזיר לכל תת רשימה של נקודות מהרשימה ההתחלתית מקדמים של פולינום שעובר בכל הנקודות האלו. בפרט, עבור הקבוצה עצמה שהיא תת קבוצה של עצמה. לכן האלגוריתם יעבוד.

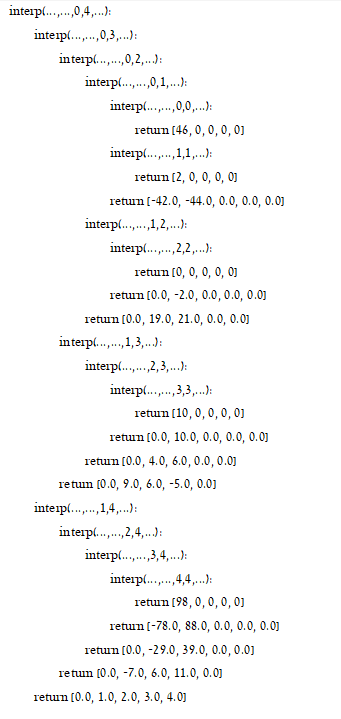
## זמן ריצה

זמן הריצה של האלגוריתם הוא . לכל זוג של אינדקסים i,j האלגוריתם מחשב פעם אחת בדיוק את הפולינום שהוא האינטרפולציה של תת סדרת הנקודות , יש סדרות כאלה. כל הרצה של סדרה מריצה לולאת for של n איטרציות, כלומר סה"כ .

## סעיף ג

חמשת הנקודות:

ריצת האלגוריתם:



האלגוריתם מחזיר כלומר כנדרש.

# שאלה 4

## סעיף א

האלגוריתם מחשב את המרחק הקצר ביותר בגרף מקודקוד r נתון כלשהו לכל קודקוד v אחר.

הוכחה:

נוכיח באינדוקציה שאחרי האיטרציה הi של הלולאה החיצונית ערכו של אינו עולה על אורך המסלול הקצר ביותר מr לv הכולל לכל היותר i קשתות.

בסיס: בסיום האיטרציה הראשונה (i=1), לכל , . כלומר, בסיומה, אם לא קיימת קשת כך ש כלומר כך ש כלומר כך שu שכנה של v, אז עדיין (כי אין מסלול באורך קשת אחת מr לv). אם קיימת קשת כזו, אז אם e היא הקשת **המינימלית** כזו, אז

שהוא אכן קצר יותר או שווה (שווה) מהמסלול הקצר ביותר מr לv הכולל לכל היותר קשת 1.

צעד: נניח שבתחילת האיטרציה הi+1 לכל צומת v, אינו עולה על המסלול הקצר ביותר מr לv הכולל לכל היותר i קשתות, ונראה שבסוף האיטרציה הi+1 לכל צומת v אינו עולה על אורך המסלול הקצר ביותר מr לv הכולל לכל היותר i+1 קשתות.

יהי צומת v. נניח שיש מסלול עם i+1 צמתים שהוא הכי קצר מבין המסלולים באורך קטן או שווה i+1 צמתים, מr לv, אחרת טירוויאלי שהאלגוריתם יעבוד עבור v כי הוא כבר נכון בתחילת האיטרציה.

נניח שמסלול זה נגמר ב ומשקלו . אז חלק המסלול עד לu משקלו . זהו המסלול הקצר ביותר באורך i צמתים לכל היותר מr לu לפי הנחת האינדוקציה. לכן בתחילת האיטרציה. כמו כן, מכיוון שזהו המסלול הקצר ביותר עם i+1 צמתים לכל היותר, אז בפרט הוא קטן מ הקודם שהו מבוסס על מסלול באורך i. כלומר, בסוף האיטרציה, כנדרש, ולכן טענה נכונה לכל i.

*נובע מכך שעבור i=n (כאשר ) בסוף ההרצה ה-n מתקיים ש קטן או שווה מאורך המסלול הקצר ביותר באורך n צמתים, שהוא המסלול הקצר ביותר בכלל מכיוון שאין מעגלים שליליים ולכן כל מעגל אפשר לקצר.*

*נבחין כי האלגוריתם לא ייעצר לפני שיגיע למסלול הכי קצר כלשהו מכיוון שאם מסלול הוא הכי קצר מr לv אז הוא גם הכי קצר לכל קודקוד עליו. כלומר בכל איטרציה תת-המסלול באורך i שלו יהיה הכי קצר ולכן ייבחר ויבצע שינוי.*

## סעיף ב

המספר המירבי של איטרציות שמתבצעות בלולאה החיצונית על גרפים בעלי קודקודים הוא . גרף יהיה מוגדר כשרשת של קודקודים, כאשר כל אחד מחובר רק לזה שלפניו וזה שאחריו. הקודקוד r יבחר בתור אחד הקודקודים הקיצוניים. כך, בכל איטרציה יתווסף קודקוד אחד שהערך של A עבורו לא אינסוף (רק זה שאחרי הקודקוד שהתווסף קודם). בסוף האיטרציה הn כל הקודקודים יהיה בעלי ערך חיובי ואז באיטרציה הn+1 לא יהיה שינוי ולכן סה"כ האלגוריתם יעשה n+1 איטרציות.

לא יכול להיות שהאלגוריתם ירוץ יותר מn+1 איטרציות כי בשלב הזה הוא כבר עבר על כל המסלולים שלא כוללים מעגלים, ולכן על כל המסלולים שיכולים להיות יותר קצרים מכל המסלולים האחרים לקודקוד כלשהו.

## סעיף ג

מספר הצלעות בסעיף הקודם הוא n-1. סדרת הגרפים היא כזו שבה r הוא מרכז שמש, וכל שאר הקודקודים מחוברים רק לr. באיטרציה הראשונה כל הקודקודים יקבלו ערך שהוא הערך של הקשת ביניהם לבין r. באיטרציה הבאה האלגוריתם עבר כבר על כל מסלול ולכן יסתיים, כלומר 2 איטרציות. כמות הצלעות היא ככמות הקודקודים שאינם המרכז כלומר n-1 צלעות.