ממן 14

מגיש: 311452254

שאלה 1

תיאור האלגוריתם:

בהינתן שריג ריבועי nxn עם מחירים אי שליליים על הקודקודים נמצא את המסלול במחיר המזערי מהשכבה השמאלית ביותר לימנית ביותר(לפי ההגדרה בשאלה), כשמחיר מסלול מוגדר כסכום מחירי הנקודות במסלול.

נגדיר: OPT(n,j) עבור כל j מ1 עד n הוא הערך האופטימלי(מינימלי) של תת קבוצה של איברים(קודקודים) כך שמחיר המסלול מזערי והוא בין השכבה השמאלית ביותר לימנית ביותר ע"פ מעברים חוקיים שהוגדרו בשאלה.

טענה:

OPT(i,j) =

נוכיח את הטענה:

נוכיח את הטענה בעזרת חלוקה למקרים-

מקרה 1: במידה ואנחנו חורגים מגבולות השריג כך שi,j =0 או I,j=n+1 אז "המחיר" של מסלול זה יהיה כך שבבחירת min מסלול , מסלול זה לא יבחר.

מקרה 2: במידה וi=1. מסלול אופטימלי לפי החוקים מהשכבה השמאלית ביותר לשכבה השמאלית ביותר (כלומר, להישאר באותה שכבה i=1) ,יהיה בעלות של הקודקוד c[i,j] שכן הצומת הנ"ל הוא היחיד שלוקח חלק במסלול (שכניו למעלה/מטה לפי החוקים אינם רשאים להופיע באותו מסלול שלו, וגם אם כן , בוודאות יוסיפו או לא ישנו את עלות המסלול שכן המחירים לא שליליים).

מקרה 3: בכל מקרה אחר, נחשב(או נקבל) את עלות(או המסלול) המסלול האופטימלי על ידי קבלת המסלול האופטימלי לשכבה אחת לפני , כלומר המסלול המינימלי לשכבה הקודמת(שמאלית יותר i-1). בנוסף , כיוון שעברנו שכבה(ימינה יותר) נוסיף לעלות המינימלית את עלות הקודקוד החדש שאליו הגענו c[i,j].

נגדיר את OPT להיות מטריצה מסדר n+2 על n+2 (לטובת פשטות ההסבר- כדי להימנע מחריגה מגבולות הסריג ) תאים i,j=0,n+1 יאותחלו ב.

*מחיר המסלול המינימלי הוא כאמור .*

*השחזור של המסלול יתבצע מהסוף להתחלה כך:*

*נמצא את ערך j עבורו הOPT[n,j] מינימלי נסמנו ב.*

*כעת נחשב את בעזרת עבור k=n-1…1:*

* *מועמד להיות או או .*
* ערך ה- המתאים *יקיים . אם המטריצה ומחיר המסלול המינימלי תקינים, יהיה קיים ערך מתאים.*

*בסיום, נקבל סדרה של אשר מסמלות את ערכי הj של הקודקודים המתאימים במסלול המינימלי.*

האלגוריתם – איטרטיבי – bottom-up:

נגדיר מטריצה OPT בגודל n+2 \* n+2 ונמלא אותה כך:

1. *לכל , נגדיר .*

לכל , *, נגדיר .*

1. *לכל , נגדיר .*
2. *לכל נגדיר:*

נמצא את עלות המסלול המינימלי מהשכבה השמאלית ביותר לימנית ביותר נסמן כmin\_cost:

1. נעבור על ערכי המטריצה בשכבה האחרונה i=n ונמצא את ערך OPT[n,j] עבור j=1…n כך שOPT[n,j] מינימלי. נציב ערך זה במשתנה min\_cost.

נשחזר את המסלול בעזרת min\_cost:

1. *נמצא ערך ה- מ1 עד n עבורו שווה לערך מחיר המסלול המינימלי. נסמן אותו בתור .*
2. *לכל :*
   1. *נחשב את לפי אלגוריתם השחזור שהצגתי מעלה.*
3. *נחזיר את התאים .*

נכונות האלגוריתם

בתחילה, אנחנו "ממלאים" את המטריצה OPT . מילוי המטריצה מתבסס על הטענה של OPT שהגדרתי מעלה והחלוקה למקרים. כל תא מחושב על בסיס ערכים שכבר חושבו , ובכל שלב כזה אנחנו בוחרים את זה המינימלי (שמאלה,שמאלה-למטה,שמאלה-למעלה) ונוסיף לו את עלות הקודקוד ה"נוכחי"(הקודקוד שהתא מייצג), כך שבשיטה זו ברור כי מקבלים את המינימלי מבין שלושת האופציות האפשריות.

יש חשיבות לכך שנבצע את התהליך משמאל לימין שכן שימוש בערכים "שמאליים" יותר בכל שלב, מכיוון שאנחנו עושים כן, אנחנו יודעים כי בכל שלב כאשר משתמשים בערכים "שמאליים" יותר, אז הערכים הללו נכונים(המצב לא היה נכון אם הינו מחשבים מימין לשמאל).

בשלב 4 – אנחנו בודקים מהו מחיר המסלול ה"חוקי" המינימלי , מכיוון שהוא נמצא בשכבה הימנית ביותר i=n ומנכונות נוסחאת הנסיגה שהגדרנו, אנחנו מקבלים את מחיר המסלול המינימלי.

בשלבים 5-7 אנחנו משחזרים את המסלול באותו אופן שבו חישבנו את שלבים 1-3 רק בסדר הפוך וע"ב הטענה על OPT.

לגבי עצירת האלגוריתם : האלגוריתם עוצר שכן נוסחאת הנסיגה מורידה את k שמייצג את i ב1 כל שלב. (שלבים 4-7) . השימוש באלגוריתם השחזור שהצגתי בתחילה גם כן סופי שכן הוא בודק מס' סופי של אופציות עבור . על כן האלגוריתם עוצר.

מכאן נובעות נכונות האלגוריתם.

סיבוכיות זמן

* *שלבים 1-3 בעת מילוי המטריצה דורשים צעדים.* 
  + *חישוב הערכים אורך זמן קבוע: שלב 1 ו2 ברורים , שלב 3 ניגשים לתאים ספציפיים במטריצה ועל כן קבוע.*
* *שלב 4 בודק מינימום מתוך n תאים ועל כן O(n) – רצה בזמן לינארי.*
* *שלב 5 גם לינארי כי הוא בודק n איברים ועל כן O(n).*
* *שלבים 6-7 עוברים על n-1 עמודות ובכל שלב מבצעות מס' קבוע של פעולות לפיכך גם O(n)- לינארי בגודל הקלט.*

*על כן, סה"כ זמן הריצה של האלגוריתם הינו .*

*שאלה 2*

*תיאור האלגוריתם:*

*תחילה, נמיין את התיבות לפי רוחביהן(w) לפי סדר יורד, כך ש: w(i)>w(j) iff j>i.*

*נגדיר מערך OPT בגודל n ככמות התיבות.*

*נגדיר מערך previous גם בגודל n.*

*נגדיר: OPT(i) - גובה המגדל היציב המקסימלי כך שתיבה i מונחת אחרונה. אם תיבה i מונחת אחרונה במגדל זה נבחין כי עבור כל תיבה אחרת j במגדל מתקיים: w(i)<w(j) וגם l(i)<l(j) שכן כל שאר התיבות מתחת לתיבה i צריכות להיות גדולות יותר ברוחב ובאורך.*

*טענה:*

*נוכיח את הטענה:*

* *מקרה 1 : מגדל בו התיבה i נמצאת לבדה . זאת מכיוון כי במערך הממוין של התיבות לפי רוחביהן אין תיבה בעלת רוחב גדול יותר ואורך גדול יותר(שני התנאים חייבים להתקיים). ועל כן האפשרות היחידה שתיבה זו תהיה עליונה במגדל זה אם תהיה "לבדה".*
* *מקרה 2: מצב בו התיבה i מתווספת לראש מגדל לפי החוקים.במקרה זה נבדוק מגדלים רלוונטים עבורם , התיבה העליונה במגדל j מקיימת: w(j)>w(i) וגם l(j)>l(i) , מתוך מגדלים שמקיימים זאת נבחר את המגדל בעל הגובה המקסימלי-גובה המגדל החדש יהיה : h(i)+max{OPT(j)} עבור j המקיים את החוקים מעלה.*

*כך opt(i) הוא המקסימום מבין האפשרויות מקרה 1 ומקרה 2 עבור אותו i.*

*נעבור על המערך OPT ונמלא אותו לפי הנוסחא ועל בסיס מערך התיבות הממוין שהגדרנו.*

*בנוסף נגדיר את המערך previous בגודל n ככזה שעבור כל מגדל שהתיבה העליונה שלו היא i , במקום הprevious[i] יופיע האינדקס j של התיבה שעליה הונחה התיבה i ומכאן שמו של המערך previous.*

*אם אין שום תיבה ששמנו עליה את התיבה i (מקרה 1) נשים במערך באינדקס i את הערך None.*

***גובה המגדל היציב המקסימלי הוא .***

*את השחזור נבצע מהסוף להתחלה , בסריקה על המערך OPT , נחפש את i בו OPT(i) מקסימלי ונסמן i זה בתור שכן אפשר להתייחס אליה כאל התיבה הראשונה – בראש המגדל.*

*נשתמש במערך previous. נתחיל במקום ה שם יופיע האינדקס של התיבה שמופיעה לפני התיבה i וכך הלאה. עד שנתקל בNone.*

*בסיום נקבל סדרת אינדקסים שמסמלים את התיבות לפי הסדר שיופיע במגדל מלמעלה(התיבה העליונה) עד התיבה התחתונה. המגדל שהתקבל הוא מגדל יציב מקסימלי.*

***האלוגריתם****:*

1. *נמיין את התיבות לפי הרוחב שלהם בסדר יורד.*

*נגדיר מערך OPT ומערך בגודל previous בגודל n כל אחד. נמלא אותם כך:*

1. *לכל i=1…n :*
   1. *נחשב את OPT[i] לפי נוסחאת הנסיגה שהגדרנו מעלה עבור OPT[i] ונכניס את התשובה במקום הi במערך.*
   2. *אם בסעיף הקודם a הגובה נבחר לפי מקרה 1 אז נכניס לprevious[i] את הערך None*
   3. *אחרת – אם בסעיף a הגובה נבחר לפי מקרה 2 , נכניס לprevious[i] את הערך j אשר j היא התיבה העליונה במגדל לפי הוספת תיבה i.*

*לאחר סיום בניית המערכים OPT וprevious ניתן לבדוק את גובה המגדל היציב המקסימלי :*

1. *נסרוק את המערך OPT ונבדוק את הערך OPT[i] המקסימלי מבין האינדקסים 1….n.*

*את ערך הגובה המגדל היציב המקסימלי נשים במשתנה max\_height.*

*את האינדקס בו מצאנו את הגובה המקסימלי נשים במשתנה אשר ישמש אותנו לשחזור סדר התיבות במגדל.*

1. *נשחזר את התיבות במגדל בעזרת max\_height והמערך previous.*
2. *נשתמש בערך שחישבנו קודם.*
3. *נגדיר ו i=1:*
4. *כל עוד previous[k]=!None*
   1. *k=previous[k]*
   2. *i=i+1*

*נחזיר את האינדקסים המהווים את סדר הופעת התיבות במגדל.*

*נכונות האלגוריתם*

*הוכחנו מעלה את הטענה על OPT(i).*

*שלב 2 משתמש בטענה על OPT כדי לבנות את המערך OPT וprevious.*

*בכל איטרציה בשלב 2 , עבור תיבה כלשהי i אנחנו משתמשים ב"ידע" שקיים לנו עבור תיבות אחרות j המקיימות : j<i ולכן לא יכול להיות שנשתמש בתא בOPT שטרם איתחלנו.*

*בנוסף, הוכחנו למעלה את שני המקרים מקרה 1 ומקרה 2 האפשריים : מקרה 1 כאשר i תיבה שלא מונחת על אף תיבה קודמת. מקרה 2 – כאשר התיבה מונחת על תיבה כלשהיא. בהתאמה , נכניס למקום הi במערך previous את הערכים None וj. בכך אנחנו מאתחלים מערך previous שיכיל עבור כל תיבה את התיבה הקודמת לה. בעזרת הmax הפנימי בנוסחאת הנסיגה אנחנו מבטיחים שבחרנו את המגדל המקסימלי מבין שתי האפשרויות (מבין כל המגדלים החוקיים האפשריים).*

*בשלב 3 אנחנו את המערך OPT כדי למצוא את גובה המגדל המקסימלי, מהוכחת הטענה(נוסחת הנסיגה) OPT אז מובטח לנו כי הגובה שימצא הוא הגובה המקסימלי של המגדל היציב.*

*בשלבים הבאים אנחנו משחזרים את התיבה לפי המערך previous. בצורה איטרטיבית עד אשר מגיעים לprevious[j]=None, משחזרים מראש המגדל לבסיסו.*

*לפיכך האלגוריתם פותר את הבעיה.*

*סיבוכיות זמן*

*מיון התיבות לפי הרוחב אורך זמן O(nlogn).*

*בשלב 2 אנחנו בונים את המערכים OPT וprevious:*

* *יש n תאים בOPT ועבור כל תא המקסימום תאים שנבדקים הוא O(n) ולכן זמן בניית המערך הוא לכל היותר .*
* *בניית previous נבנית במקביל לOPT ולא מוסיפה לסיבוכיות הזמן.*

*בשלב 3 – סורקים את המערך OPT ועל כן O(n).*

*שחזור התיבה מתבצע בסיבוכיות זמן O(n) לינארית.*

*שכן סורקים את previous בצורה לינארית.*

*על כן בסה"כ , סיבוכיות הזמן של האלגוריתם הינה O(n^2).*

*שאלה 3:*

*בשאלה זו נעשה שימוש ב* [**Neville's algorithm .**](https://en.wikipedia.org/wiki/Neville%27s_algorithm)

1. *נבחר :*

*נראה כי הנוסחא מתקיימת עבור הפולינומים שבחרתי מעלה.*

*נקבל:*

*המטרה להראות שהפולינום הנ"ל מקבל את אותם ערכים בנקודות מi עד j.*

*אם נסתכל על בצד הימני של המונה , בנק' הוא מתאפס ולכן לאחר הצמצום עם המכנה, נקבל כי*

ולכן הפולינום עדיין עובר ב-.

אם נסתכל על בצד השמאלי של המונה, בנק' הוא מתאפס ולאחר הצמצום עם המכנה נקבל כי:

ולכן הפולינום עדיין עובר ב- .

עבור מתקיים ולכן:

*, כנדרש.*

1. *נציע אלגוריתם תכנון דינאמי לבעיית האינטרפולציה ע"ב נוסחאת הנסיגה בסעיף הקודם:*

*רעיון האלגוריתם*

*נגדיר מטריצה M בגודל n\*n.*

*האיבר M[i,j] יחושב ע"ב יהיה פולינום האיטרפולציה שיחושב ע"ב הנוסחא בסעיף א' ועל בסיס הנקודות :*

.

אלכסון המטריצה יהיה M[i,i]= לכל i. מכיוון שזהו פולינום המבוסס על נקודה אחת , הפולינום ממעלה 0 ולכן .

*שאר האיברים יתבססו על נוסחאת האינטרפולציה מהסעיף הקודם.*

*התשובה שתחזור, המייצגת את הפולינום הרצוי המבוסס על n נקודות יהיה אפוא בתא: M[1,n].*

***האלגוריתם:***

1. *נגדיר מטריצה M בגדול n\*n כך ש:*
   1. *עבור i=1….n: M[i,i]=(האלכסון כפי שהוסבר מעלה).*
   2. *עבור k=2…n:*
      1. *i=1*
      2. *עבור j=k…n:*
         1. *נחשב את M[i,j] ע"ב הנוסחא בסעיף א' (נוסחא רקורסיבית)*
         2. *i=i+1*
2. *נחזיר את הפולינום בתא M[1,n].*

***נכונות האלגוריתם:***

*האלגוריתם מבוסס על נוסחאת הנסיגה בסעיף א' (והדרך שבה הגענו אליה , הוכחנו את נכונתה בסעיף א').*

*המטריצה מחושבת תחילה באלכסון הראשי שלה, שם אין שימוש בעוד תאים אחרים.*

*בשאר החישוב, ניתן לראות כי אנחנו משתמשים רק בתאים קודמים שכבר חושבו. כל חישוב תלוי במס' קבוע של פולינומים ( ערכים בתאים במטריצה).*

*נשים לב שאנחנו מחשבים כך שאנחנו תלויים רק בערכים שכבר חושבו:*

*בתחילה k=2 מחשבים את האיברים בתאים : (1,2) עד (n-1,n) - כאשר שני האינדקסים עולים בכל שלב(כמו שניתן לראות בתת לולאה / לולאה פנימית המסומנת בצהוב) - בעצם מדובר באלכסון במטריצה.*

*כלומר, בכל שלב של הלולאה החיצונית אנחנו מחשבים אלכסון אחר, כך בשלב k נחשב את התאים באלכסון שמתחיל ב) ומסתיים ב .*

*בעצם, בעת חישוב M[i,j] כאשר j>i (לפי האלגוריתם שבנינו, זה תמיד המצב), אנחנו משתמשים רק ב:*

*ב- ו (זהו שני הפולינום p בנוסחא בסעיף א').*

*באלכסון הראשי אין שימוש בכלל בתאים אחרים.*

*הראנו כי חישבנו בסדר נכון , ובכך לא נתקל באף מצב בו נשתמש בתא שלא חושב בעבר.*

*על כן , בשלב האחרון אכן נחזיר את הפולינום הנדרש בשאלה במקום M[1,n].*

*סיבוכיות זמן*

*1.a – רץ בזמן לינארי על האלכסון הראשי .*

*1.b- יש לנו לולאה חיצונית שרצה O(n) פעמים , ולולאה פנימית שרצה O(n) פעמים, בתוך הלולאה הפנימית מתבצעת עבודה קבועה על בסיס נוסחאת הנסיגה שהראינו , אמנם מדובר בנוסחא רקורסיבית, אך בגלל שהראנו כי אנחנו מחשבים בסדר "נכון" , אנחנו תמיד משתמשים בשני תאים בלבד. (ניתן להניח כי פעולות אריתמטיות יתבצעו בזמן קבוע). לכן , סעיף זה רץ ב.*

*2- רץ בזמן קבוע.*

*סה"כ זמן ריצה .*

1. *עבור:*

. נציב את הערכים הנתונים ונקבל את הנק' הבאות:

הנקודות שהתקבלו :

נריץ את האלגוריתם שבנינו בסעיף הקודם ונוודא שאכן קיבלנו את הפולינום p(x) הנתון.

נציג את החישובים להלן:

נגדיר מטריצה בגודל 5\*5.

1.a- נחשב תחילה את הערכים באלכסון :

עבור i=1…5 נחשב:

M[1,1]=46

M[2,2]=2

M[3,3]=0

M[4,4]=10

M[5,5]=98

1.b: נחשב את שאר התאים במטריצה,כפי שהוסבר בסעיף ב':

|  |  |  |  |
| --- | --- | --- | --- |
| *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* |  |

*בכך הראנו כי האלגוריתם שבנינו אכן משחזר את הפולינום הנתון.*

*מ.ש.ל.*

*שאלה 4*

1. *האלגוריתם מחשב את המסלול בעל העלות המינימלית מהקודקוד הנתון אל כל קודקוד . את התשובות הוא שומר במערך החד-ממדי .*

*צומת שלא ניתן להגיע אליו ישאר עם ערך כפי שאותחל המערך בתחילה. הערך במערך עבור r יהיה אפוא 0.*

*נוכיח באינדוקציה על האיטרציה החיצונית החל מi=0.*

*נגדיר מסלול נגיש באורך i אם אורכו בין 0 לi.(כולל הקצוות).*

*נקרא לו מסלול i -נגיש.*

*טענת האינדוקציה:*

*בסוף האיטרציה הi של הלולאה החיצונית, לכל צומת v:*

1. *כאשר אז אין מסלול i-נגיש מr ל-v.*
2. *כאשר (כאשר k סופי ואי-שלילי), אז קיים מסלול i-נגיש מr ל-v שעלותו k, וk קטן או שווה מהעלות המינימלית של כל המסלולים ה i-נגישים מr ל-v.*

*בסיס:*

*בסוף האיטרציה ה0 של הלולאה – למעשה לפני האיטרציה הראשונה , אז הצומת היחיד בעל ערך סופי במערך A הוא עבור הצומת r. ואכן , כל שאר הצמתים לא 0-נגישים. כיוון שזה מתקיים רק עבור הצומת עצמו.*

*הנחת האינדוקציה:*

*נניח שהטענה נכונה עבור n ונוכיח אותה עבור n+1.*

*ננסה להוכיח בשלילה כי התנאים לא מתקיימים,אנחנו צפויים להיכשל ובכך להוכיח שהטענה נכונה גם עבור n+1 ובכך להוכיח את הטענה.*

* *נניח בשלילה שקיים צומת v כלשהו עבורו הערך A[v] סופי, אך אינו מייצג עלות מסלול n+1 נגיש מr לv. לפי הנחת האינדוקציה כל הערכים A[v]=k כאשר k סופי מייצגים מסלולים n-נגישים מr לצומת כלשהו(והם חוקיים ומייצגים מסלולים חוקיים). כלומר , אם יש ערך A[v]=k שגוי הוא נוצר באיטרציה הנוכחית. נסמן בe=(u,v) את הקשת שבגינה הערך שעודכן שגוי. לפי האלגוריתם אחנו מעדכנים את A[v] כאשר A[u]+c(e) קטן מהערך הנוכחי בA[v]. לפי הנחת האינדוקציה הA[u] הוא משקל n נגיש מr לu. אם נוסיף לו את הקשת e ניצור מסלול n+1-נגיש מr לv , בסתירה להנחת השלילה.*
* *נניח כי קיים צומת v כלשהו שעבורו קיים מסלול n+1 נגיש שעלותו נמוכה יותר מהערך בA[v].*

*הקשת האחרונה במסלול זה נסמנה בe’=(u,v). לפי הנחת האינדוקציה A[u] הוא העלות המסלול המינימלי מr לu. לפי האלגוריתם אנחנו עוברים את כל הקשתות ומעדכנים לפי הקשת שתביא לנו את העלול המינימלית לכן לא יתכן שנבחר בקשת שהיא לא מינימלית בין u לv. בסתירה להנחת השלילה.*

* *נניח שקיים v צומת כלשהו שעבורו A[v]= אך קיים עבורו מסלול n+1 נגיש. אם קיים מסלול כזה כמובן שניתן לחשב את העלות שלו. אם נמחק את הקשת האחרונה במסלול e =(u,v) נקבל מסלול n-נגיש לפי הנחת האינדוקציה כי בA[u] נמצא העלות של המסלול הn נגיש המינימלי מr לu והוא מוגדר בצורה נכונה והערך הוא סופי(כי הוא נגיש). לכן במהלך האיטרציה כאשר אנחנו סורקים ומוסיפים קשת כלשהי שמגיעה מu לv אנחנו מוסיפים ערך סופי לערך סופי ולכן גם הסכום חייב להיות סופי. הסכום יבחן מול הערך הנוכחי ובמידה והוא מינמלי יוחלף.*

*אם ,אז בוודאות שהערך החדש קטן יותר ועל כן יוחלף.*

*ועל כן לא יהיה ערך ב A[v]. בסתירה להנחה.*

*סך הכל סתרנו את כל האפשרויות ולכן הטענה נכונה עבור n+1 ולכן נכונה עבור כל n.*

1. *לאור ההגדרה של n-נגיש , מכיוון שיש n צמתים, לאחר n-1 איטרציות בלולאה החיצונית האלגוריתם יעצור, שכן כבר נמצאו כל הערכים של המרחקים בA. לכן באיטרציה הn לא יהיו יותר עדכונים והאלגוריתם יעצור.*

*על כן B(n)=n.*

*נגדיר גרף שרוך כך:*

*משקל הקשתות יהיה 1. הגרף יראה כך:*



*בריצה של האלגוריתם על הגרף הנ"ל , בכל אירטציה יעודכן ערך של צומת אחד בלבד במערך A שכן, רק עבור זוג צמתים שכנים שלאחד מהם(השמאלי) כבר חושב הערך במטריצה יתבצע החישוב באלגוריתם ויעודכן הערך עבור הצומת הימני החדש. לכל השאר, הערכים הם ועל כן לא יתבצע עדכון עבורם.*

*מכאן שמס' האיטרציות בסדרת גרפים זו הוא B(n)=n.*

1. *נציג סדרת גרפים עבורם נכנסים ללולאה החיצונית פעמיים בלבד וזאת למרות שמס' הצלעות שלהם זהה לסעיף הקודם:*

*נגדיר את הגרף הבא(גם כן שרוך) עבור n:*

*משקל כל הקשתות 1.*

*הסריקה של הקשתות היא בסדר מספרי כך שסורקים את ואחר כך את וכן הלאה.*

*ועל כן כבר באירטציה הראשונה יעודכנו לפי הסדר כאשר כל אחד מהם יקבל את משקל המסלול המינימלי אליו(משום שהמסלול המינימלי עובר דרך הקשת הקודמת בלולאה וגם אליה כבר חושב באיטרציה הפנימית הקודמת המסלול המינימלי).*

*בריצה השניה של הלולאה לא יהיו יותר עדכונים והאלגוריתם יסיים את ריצתו.*