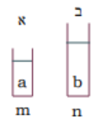
תרגול 1-

בעיית הבקבוקים:

- נתונים 2 בקבוקים בשני גדלים שונים: בקבוק א' בגודל 5 ליטרים, ובקבוק ב' בגודל 3 ליטרים.
- נרצה למלא את בקבוק א' ב-4 ליטרים.
- הפעולות המותרות:
  - למלא בקבוק עד הסוף
  - לרוקן בקבוק עד הסוף
  - למזוג מבקבוק אחד לבקבוק שני

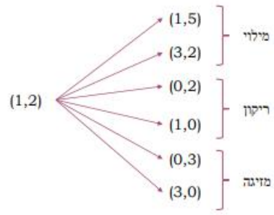
- איך נעשה זאת?

- המצב ההתחלתי יכול להיות כל מצב אפשרי: או שנתחיל ממצב (0,0) שאומר ששני הבקבוקים ריקים, או לדוגמא אפשר להתחיל ממצב (0,1) שאומר שבקבוק א' ריק, אבל בקבוק ב' מלא ב-1 ליטר.
- המטרה הסופית: לבדוק האם מכל מצב התחלתי אפשר להגיע למצב יעד כלשהו, עבור כל מצב יעד שנבחר.
- נגדיר רגע תיאור בעיה כללי: יש לנו 2 בקבוקים בגדלים n,m המלאים בתכולה כלשהי (a,b):



- כעת, נגדיר את הפעולות המותרות:
  - מילוי בקבוק א'- (m,b)
  - מילוי בקבוק ב'- (a,n)
  - ריקון בקבוק א'-(0,b)
  - ריקון בקבוק ב'- (a,0)
  - מזיגה מבקבוק א' ל-ב' – (a+b-min(n,a+b), min(n,a+b))
  - מזיגה מבקבוק ב' לבקבוק א'-(min(a+b,m), a+b-min(a+b,m))

- המשימה שלנו היום היא לייצר את גרף הבקבוקים. הגרף מכיל קודקודים וצלעות, כך שבעצם מכל קודקוד נסמן לאיזה קודקוד אחר ניתן להגיע.
- <u>לדוגמא:</u>
- מהקודקוד (1,2) לאן ניתן להגיע?



- כל צלע מקשרת לנו בין קודקוד לקודקוד

- בגרף הבקבוקים נרצה לעשות זאת עבור כל מצב אפשרי כמו: מ-(0,0) לאן ניתן לעבור, מ-(0,1) לאן ניתן לעבור וכו'.
- בשלב הראשוני נרצה לדעת רק לאן ניתן להגיע ממצב מסויים, ולא את כל המסלול (שאותו נראה בשבוע הבא בעז"ה)
- איך נייצג גרף במחשב?
- ע"י מטריצת שכנויות
- <u>לדוגמא:</u>
- עבור  נקבל את המצבים הבאים:

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)

- נשים לב שמכל מצב ניתן להגיע לכל אחד מהמצבים הללו ולכן כדי ליצור טבלת קשרים נצטרך מטריצה במימד רביעי- קורדינאטות i,j הן המצב שממנו באים, קורדינאטות k,l הן המצב שאליו מגיעים, ובכל תא במטריצה i,j,k,l נרשום 0 אם אין קשר בין המצבים ו-1 אם יש קשר.
- זה מאוד מסובך.. אז ננסה לצמצם את זה..
- איך?

- נצמצם את מטריצת השכנויות

(0,0)	(0,1)	(0,2)
(1,0)	(1,1)	(1,2)

למערך

(0,0)	(0,1)	(0,2)	(1,0)	(1,1)	(1,2)

- וכעת נוכל לרשום מטריצת שכנויות דו מימדית, ולמלא אותה באחדות בכל תא שיש קשר בין המצבים שלו:

	(0,0)	(0,1)	(0,2)	(1,0)	(1,1)	(1,2)
(0,0)	1	0	1	1	0	0
(0,1)	1	1	1	1	1	0
(0,2)	1	0	1	0	1	1
(1,0)	1	1	0	1	0	1
(1,1)	0	1	1	1	1	1
(1,2)	0	0	1	1	0	1

- בעצם, באלכסון קיבלנו שיש קשר בין מצב לעצמו. השאלה האם אפשר להגיע ממצב לעצמו היא שאלה פילוסופית. לכן, זה לא מאוד משנה לנו האם יהיה שם 0 או 1.
- מתי זה ישנה לנו?
  - תלוי בנו. אם בשימוש נוסף במטריצה יעזור לנו שיש באלכסון 1, אז נשים 1, אבל אם נצטרך 0 נשים 0.
- דבר נוסף, מה נשים באינדקסים של המטריצה? הם לא יכולים לכלול 2 קורדינאטות.. אז איך נבצע המרה?

- ננסה להבין את עיקרון ההמרה על ידי דוגמא:

(0,0) 0	(0,1) 1	(0,2) 2	(0,3) 3
(1,0) 4	(1,1) 5	(1,2) 6	(1,3) 7
(2,0) 8	(2,1) 9	(2,2) 10	(2,3) 11

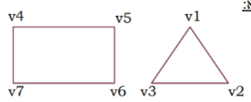
- כל תא במטריצה בנוי כך: k (i,j) ואנו נרצה להבין איך לבצע המרה בין (i,j) לבין ה-k המתאים.
- במעבר משורה לשורה אנו קופצים בקפיצות של 4 - לפי מספר העמודות. וכדי להגיע לעמודה ספציפית, נוסיף את ה-j שבכל שורה.
- <u>לדוגמא:</u> כדי להגיע לתא k=6, נבצע 6 = 2+(4*1) = (4*i)+j
- אבל איך נדע כמה עמודות יש לנו?
- המטריצה למעלה מתאימה לבעיה  . כמות המצבים שאנו מקבלים מבעיה זו היא (3+1)*(2+1)=12.

- סה"כ ניתן לומר שמספר העמודות הוא (3+1) או עבור בעיה כללית יותר  (n+1) ולכן באופן כללי נרשום: k = (n+1)*i + j

- מכאן נסיק ש:
  - i = k/(n+1) כי זה נותן לנו בדיוק את השלם
  - j = k%(n+1) כי זה נותן לנו את שארית החלוקה

תרגול 2-

- בתרגול שעבר יצרנו מטריצת שכנויות. הגדרנו אותה כך- בכל תא שמופיע בו 1 או true נדע שיש קשר (או קשת) בין המצבים, ובתאים שמופיע בהם 0 או false אין קשר.
- בהינתן מטריצת שכנות, איך אפשר לדעת האם קיים מסלול, ואם כן, אז מהו?
- לדוגמא:

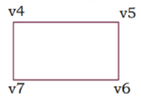


	v1	v2	v3	v4	v5	v6	v7
v1		1	1				
v2	1		1				
v3	1	1					
v4					1		1
v5				1		1	
v6					1		1
v7				1		1	

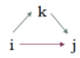
- האם יש צלע בין v1 ל-v2?
  - נלך לתא המתאים בטבלה- אם יש שם 1 אז יש צלע ואם יש שם 0 אז אין צלע
- האם יש מסלול בין v4 ל-v6?
  - בשביל מסלול הטבלה לא עוזרת לנו ישירות, כי אולי אין צלע, אבל יכול להיות שיש מישהו שמחבר ביניהם ויוצר מסלול

1. אלגוריתם Floyd-Warshall

- הרעיון שעומד מאחורי האלגוריתם הזה הוא לבדוק האם קיים קודקוד כלשהו בגרף שיכול לחבר לנו בין 2 קודקודים אחרים וליצור מסלול.
- לדוגמא: בשביל לענות על השאלה מקודם 'האם יש מסלול בין v4 ל-v6?' נוכל לראות שקודקוד v7 יכול לחבר ביניהם ולגרום לכך שיהיה מסלול.



- כלומר, האלגוריתם עובר על כל הקודקודים ובודק:



אם קיים מסלול בין i ל-k ובין k ל-j אז קיים מסלול בין i ל-j

- האלגוריתם:

```
for k=1 to n:
    for i=1 to n:
        for j=1 to n:
            T[i,j] = T[i,k]^T[k,j]
```

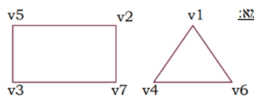
- מה הבעיה באלגוריתם?
  - שאם יש לי כבר מסלול בין i ל-j, אז k יכול להפריע לי- כי אולי אין צלע בין i ל-k או בין k ל-j.
- מה הפתרון?
  - להוסיף תנאי 'או' שלא יפגע במה שקיים בתא, כלומר קודקוד לא יכול לפגוע אלא רק לעזור

```
for k=1 to n:
    for i=1 to n:
        for j=1 to n:
            T[i,j] = (T[i,k] ^ T[k,j]) v T[i,j]
```

- ברגע שנריץ את האלגוריתם על מטריצת השכנויות שלנו, נוכל לדעת האם קיים מסלול בין כל 2 קודקודים.
- האם יש מסלול בין קודקוד לעצמו?
  - תשובה: כן, למשל המסלול של קודקוד v1 לעצמו הוא- v1-v2-v3-v1.
  - או, שניתן אפילו לומר שיש מסלול באורך 0.
  - **חידוד**: v1 לא שכן של עצמו, אבל קיים מסלול ממנו אל עצמו.
- בואו נבצע הרצה של FW ונקבל את מטריצת המסלולים:

	v1	v2	v3	v4	v5	v6	v7
v1	1	1	1				
v2	1	1	1				
v3	1	1	1				
v4				1	1	1	1
v5				1	1	1	1
v6				1	1	1	1
v7				1	1	1	1

- ניקח בחשבון שלא תמיד המטריצה תהיה כזו יפה.
- לדוגמא:



	v1	v2	v3	v4	v5	v6	v7
v1	1			1		1	
v2		1	1		1		1
v3		1	1		1		1
v4	1			1		1	
v5		1	1		1		1
v6	1			1		1	
v7		1	1		1		1

- המטריצות נותנות לנו את אותו המידע, רק בסדר קודקודים שונה..
- האם ניתן להגיע ממטריצה מבולגנת כזו למטריצה מסודרת כמו מקודם?
  - כן, ע"י החלפת סדר הקודקודים כמו שהיה מקודם- נחליף עמודות ושורות.
  - איך עושים את זה בצורה יעילה? נסו לחשוב לבד..

2. האם יש מסלול בין vi ל-vj?

- נפעיל FW על הגרף ואז ב-o(1) נוכל לגשת לתא המתאים במטריצה ולראות האם יש שם 1 או 0

3. האם הגרף קשיר? (אנו מדברים רק על גרפים לא מכוונים)

- ברגע שכל המטריצה מלאה באחדות נדע שהגרף קשיר כי ניתן להגיע מכל קודקוד לכל קודקוד אחר.

ב-o(n^2) נוכל לעבור על כל המטריצה ולבדוק

- האם יש פתרון יעיל יותר?
  - ב-o(n) נעבור על השורה הראשונה ונבדוק שהכל שם אחדות.
  - למה זה נותן פתרון?

	v1	v2	v3	v4	v5	v6	v7
v1	1	1	1				
v2	1	1	1				
v3	1	1	1				
v4				1	1	1	1
v5				1	1	1	1
v6				1	1	1	1
v7				1	1	1	1

4. כמה רכיבי קשירות יש? (אנו מדברים רק על גרפים לא מכוונים)

- נעבור על המטריצה ונבדוק.
- מה הבעיה?
  - שלאו דווקא נקבל מטריצה כזו יפה
- אז מה נעשה?

4. כמה רכיבי קשירות יש? (אנו מדברים רק על גרפים לא מכוונים)

- נייצר מערך עזר

v1	v2	v3	v4	v5	v6	v7
0	0	0	0	0	0	0

- נעבור על המערך וכל פעם שנראה 0 נדע שעוד לא הגענו לקודקוד הזה. נבדוק במטריצת השכנויות מי השכנים שלו ונסמן במערך. נוסיף counter שבודק לנו מה מס' רכיבי הקשירות ונעלה אותו כל פעם. כך נעבור על כל המערך, ורק על השורות הרלוונטיות במטריצה.
- דוגמת הרצה:

	v1	v2	v3	v4	v5	v6	v7
v1	1			1		1	
v2		1	1		1		1
v3		1	1		1		1
v4	1			1		1	
v5		1	1		1		1
v6	1			1		1	
v7		1	1		1		1

v1	v2	v3	v4	v5	v6	v7

counter =

- כך קיבלנו מערך המכיל לנו מידע עבור מספר רכיבי הקשירות

5. מיהם הקודקודים ברכיבי הקשירות?

- נוכל לעבור על המערך ולראות מי נמצא באיזה רכיב

- שאלת מחשבה: איך אפשר למצוא איבר שלא נמצא בתת קבוצה מסוימת? כלומר, אם יש לי 7 קודקודים וקבוצה {2,3,5,7} איך אוכל בצורה יעילה לדעת מי לא נמצא שם? במקום לעבור על המערך ולראות איפה נשארו אפסים..

6. מציאת מטריצת מסלולים

- מקודם מצאנו מטריצת שכנויות שמראה לנו בין אלו קודקודים יש מסלולים, אך איך נדע מהם המסלולים?
- נייצר מטריצת עזר של מחרוזות ובכל תא נרשום איך מגיעים מקודקוד לקודקוד. זה לאו דווקא המסלול הקצר ביותר, אבל לפחות נותן לנו פתרון.
- ניקח את FW ונמיר את התנאי האחרון בו לתנאי שיעזור לנו לשרשר מסלולים.

7. בקבוקים- האם קיים מסלול בין (i,j) ל-(k,l)?

- נחזור לבעיית הבקבוקים. עכשיו, אם נרצה לדעת האם קיים מסלול בין מצב מסוים למצב אחר נוכל להמיר מ-(i,j) ל-k, להפעיל FW להמיר חזרה מ-k ל-i ו-j ולראות האם קיים מסלול או לא. אם ממש נרצה את המסלול נפעיל את האלגוריתם שמחשב לנו את המסלולים עצמם ונחזיר את הפתרון.

8. אם כן, תנו דוגמא

- לאחר שתריצו את 7 החזירו דוגמא ספציפית יותר.

9. סידור המטריצה

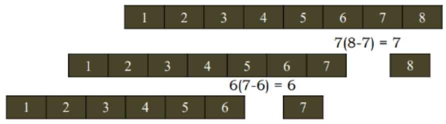
- דיברנו על זה מקודם. נחליף עמודות ושורות ונהפוך את המטריצה למטריצה יפה יותר

בעיית השוקולד

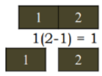
- יש פס שוקולד המכיל n קוביות. על כל חלוקה באינדקס k משלמים k(n-k)
- המטרה: לחלק את השוקולד לקוביות בודדות ולשלם את המינימום האפשרי.

- <u>לדוגמא:</u> ניקח פס שוקולד המכיל 8 קוביות: 

- ניסיון 1: שבירה לאחדות



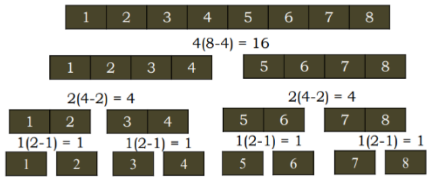
...



סה"כ נקבל:
1+2+3+4+5+6+7=
8(8-1)/2 = 28

- <u>לדוגמא:</u> ניקח פס שוקולד המכיל 8 קוביות: 

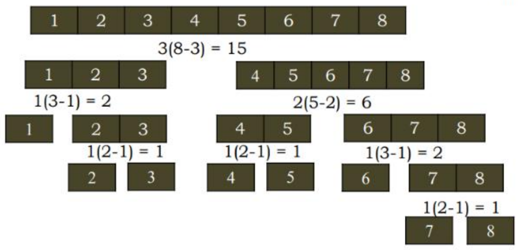
- ניסיון 2: שבירה באמצע



סה"כ נקבל:
16+8+4=28

- <u>לדוגמא:</u> ניקח פס שוקולד המכיל 8 קוביות: 

- ניסיון 3: פיבונאצ'י



סה"כ נקבל:
15+2+6+1+1+2+1=28

- <u>לדוגמא:</u> ניקח פס שוקולד המכיל 8 קוביות: 

- מסקנה: על כל חלוקה אפשרית נצטרך לשלם n(n-1)/2
- **הוכחה:** נוכיח באינדוקציה על מספר הקוביות
- <u>בסיס:</u> n=2. אם יש לנו רק 2 קוביות יש רק פיצול אחד שנשלם עליו 1(2-1)=2(2-1)/2 = 1
- <u>הנחה:</u> הטענה נכונה עבור k ≤ n-1
- <u>שלב האינדוקציה:</u> נוכיח עבור k=n
- כאשר נפצל ב-k כלשהו, כל תת מקטע יהיה בגודל קטן או שווה ל-n-1. החלוקה עצמה תעלה לנו k(n-k).
- מקטע שמאל [1..k] יעלה לנו k(k-1)/2 לפי הנחת האינדוקציה
- מקטע ימין [k+1…n] יעלה לנו (n-k)(n-k-1)/2 לפי הנחת האינדוקציה
- כלומר העלות הכוללת תהיה:

$$k(n-k)+\frac{k(k-1)}{2}+\frac{(n-k)(n-k-1)}{2}$$

- <u>לדוגמא:</u> ניקח פס שוקולד המכיל 8 קוביות: 

- נמשיך לפתור:

$$\frac{2kn-2k^2+k^2-k+n^2-nk-n-kn+k^2+k}{2}$$

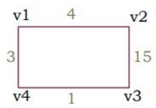
$$\frac{\cancel{2kn}-\cancel{2k^2}+\cancel{k^2}-\cancel{k}+n^2-\cancel{nk}-n-\cancel{kn}+\cancel{k^2}+\cancel{k}}{2}$$

$$\frac{n^2-n}{2} \Rightarrow \frac{n(n-1)}{2}$$

- ונגיע למסקנה ממקודם- לא משנה איזה חלוקה נבצע תמיד נשלם n(n-1)/2
- מש"ל ☺

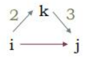
תרגול 3-

- בתרגול שעבר יצרנו מטריצת שכנויות ובעזרת FW הגדרנו גם מטריצת מסלולים שעוזרת לנו לדעת האם קיים מסלול בין זוג קודקודים או לא.
- מה יקרה אם נוסיף משקלים לצלעות?
- לדוגמא:



	v1	v2	v3	v4
v1	0	4	∞	3
v2	4	0	15	∞
v3	∞	15	0	1
v4	3	∞	1	0

- קיבלנו גרף של כל המרחקים והמשקלים שיש בין 2 צלעות
- בעזרת FW נוכל לשאול האם קיים מסלול בין הקודקודים, אך גם נוכל לשאול מה המסלול הקצר ביותר בין כל שני קודקודים, ולעדכן אותו:
  - אם כבר יש לנו מסלול בין i ל- j, נבדוק אם קודקוד k יכול לעזור

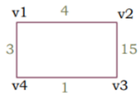


1. מציאת מטריצת המרחקים- משקלים על הצלעות

- האלגוריתם עכשיו יראה כך:

```
for k=1 to n:
    for i=1 to n:
        for j=1 to n:
            D[i,j] = min(D[i,k]+D[k,j], D[i,j])
```

- דוגמת הרצה:



	v1	v2	v3	v4
v1	0	4	∞	3
v2	4	0	15	∞
v3	∞	15	0	1
v4	3	∞	1	0

	v1	v2	v3	v4
v1	0	4	19	3
v2	4	0	8	7
v3	19	8	0	1
v4	3	7	1	0

2. מציאת מטריצת המסלולים- משקלים על הצלעות

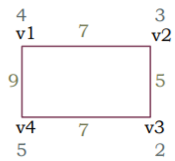
- בדיוק כמו בתרגול שעבר: נייצר מטריצת עזר של מחרוזות ובכל תא נרשום איך מגיעים מקודקוד לקודקוד. ניקח את FW ונמיר את התנאי האחרון בו לתנאי שיעזור לנו לשרשר מסלולים.

3. חזרה וסיכום של בעיית הבקבוקים

- נוכל לשפר עוד קצת את הבעיה בכך שנמצא את המסלול הקצר ביותר ממצב למצב.
- אם נרצה לדעת האם קיים מסלול בין מצב מסוים למצב אחר נוכל להמיר מ-(i,j) ל-k, להפעיל FW החדש, להמיר חזרה מ-k ל-i ו-j ולראות האם קיים מסלול או לא. אם ממש נרצה את המסלול נפעיל את האלגוריתם שמחשב לנו את המסלולים עצמם ונחזיר את הפתרון.

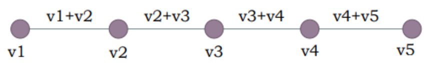
- כעת, נעבור לבעיה אחרת- משקלים על הקודקודים
- איך ניתן כעת למצוא את המסלול הקצר ביותר מכל קודקוד לכל קודקוד אחר?
  - תשובה א': לא נתקלנו בבעיה כזו מעולם ולכן בואו נחפש אלגוריתם חדש
  - תשובה ב': בואו ננסה לתרגם את הבעיה לבעיה שאנו כבר מכירים ולפתור לפיה

- אז בואו ננסה להמיר את הבעיה הזו לבעיית משקלים על הצלעות
- לדוגמא:



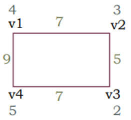
- נוצרה כאן בעיה. מהי?
  - כשאנו רוצים למצוא את המסלול בין v1 ל-v3 אנו מקבלים 7+5=12, אך זה לא נכון כי המשקלים על הקודקודים מראים: 4+3+2=9.

- או בצורה כללית יותר:



- במקום לומר שהמסלול בין v1 ל-v5 הוא: v1+v2+v3+v4+v5 נקבל: v1+v2+v2+v3+v3+v4+v4+v5
- פתרון?
  - ניקח את כל הכפולים ונוריד אותם
  - מיהם הכפולים?
    - כל מי שבאמצע..
- פתרון נוסף:
  - אם כל האמצעיים מופיעים פעמיים, אז נוסיף עוד פעם אחת את הקודקודים בקצוות ומחלק ב-2.

- לדוגמא:
- אנו מקבלים כקלט מטריצת שכנויות שנראית כך:



	v1	v2	v3	v4
v1	0	1	∞	1
v2	1	0	1	∞
v3	∞	1	0	1
v4	1	∞	1	0

- ומערך:

v1	v2	v3	v4
4	3	2	5

- הפלט שלנו יהיה מטריצת המרחקים.
- איך נעשה זאת?
  - נבצע המרה- נמיר את מטריצת השכנויות במטריצת שכנויות של צלעות- ניקח כל i ו-j שיש ביניהם 1 במטריצה ונכניס לתא את החיבור של העלויות שלהם מהמערך.
  - נבצע FW שיתן לנו את המרחקים הקצרים בין כל קודקוד לכל קודקוד אחר
  - אמרנו שהפתרון יוצא לא נכון כי יש קודקודים שנכללים פעמיים, אז בשלב זה נתקן.
- תשובה סופית:

	v1	v2	v3	v4
v1	0	7	9	9
v2	7	0	5	10
v3	9	5	0	7
v4	9	10	7	0

2. מציאת מטריצת המסלולים- משקלים על הקודקודים

- בדיוק כמו מקודם: נייצר מטריצת עזר של מחרוזות ובכל תא נרשום איך מגיעים מקודקוד לקודקוד. ניקח את FW ונמיר את התנאי האחרון בו לתנאי שיעזור לנו לשרשר מסלולים.

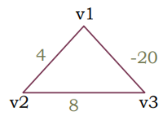
- איך נסמן אינסוף?
  - פתרון זמני: אם אנחנו מסתמכים על זה שקלט הערכים שלנו נע בגבול נמוך יחסית, נוכל להגדיר את אינסוף כמיליון.
  - אם נוסיף תנאי לפני הנוסחא ב-FW שאומר: אם אף אחד מהם לא 1- או אף אחד לא MAXVALUE אז תבצע את התנאי.

3. החזרת דוגמא למסלול הקצר ביותר עבור כל i ו-j

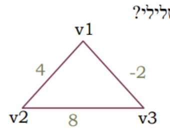
- להדפיס את מטריצת המסלולים שחישבנו מקודם..

תרגול 4-

- בהינתן גרף **לא מכוון**- האם קיים מעגל שלילי בגרף?
- <u>לדוגמא:</u>

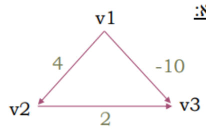


- מה הכוונה מעגל שלילי?
  - מ-v1 לעצמו ניתן להגיע בפחות מ-0: 4+8-20=-8
- אז האם פה יש מעגל שלילי?



  - ודאי שכן, כי מ-v1 לעצמו ניתן ללכת ל-v3 ולחזור ל-v1 וסה"כ לשלם 4- או 8- וכו'..
- בגרף לא מכוון, מספיק שיש צלע אחת שלילית כדי שיהיה מעגל שלילי בגרף.
- לכן, מרחק קצר ביותר בין קודקודים כבר לא רלוונטי כי ניתן להגיע עד $-\infty$
- ומכאן השאלה: האם יש מעגל שלילי בגרף, שקולה לשאלה: האם יש משקל שלילי בגרף.
- מה הסיבוכיות במציאת המשקל השלילי?
  - o(m) כאשר m זה מספר הצלעות בגרף.

- בהינתן גרף **מכוון**- האם קיים מעגל שלילי בגרף?
- <u>לדוגמא:</u>



- באמת אין פה מעגל, אז גם אין מעגל שלילי. אבל אם נהפוך את החץ כן נקבל מעגל ואפילו מעגל שלילי.
- אז איך נזהה?
  - נפעיל FW שנותן לנו את כל המסלולים שיש בגרף
- רק מה הבעיה?
  - כשיש מעגל שלילי בגרף FW לא נותן את התשובה הנכונה, אבל הוא יכול להצביע על זה שיש מעגל שלילי. איך?

- האלכסון מראה לנו את המרחק מקודקוד לעצמו- שבתרגול הקודם אמרנו שהוא 0, ולכן אם נראה שם עכשיו מרחק שלילי נדע שיש לנו מעגל שלילי בגרף.
- מכאן השאלה: האם בגרף מכוון יש מעגל שלילי, שקולה לשאלה: האם באלכסון הראשי יש מספר שלילי.

- תנו דוגמא למעגל שלילי
- יש לנו פונקציה שמוצאת מטריצת מסלולים. אם נזהה שיש באלכסון מספר שלילי, נחזיר את המסלול מהקודקוד לעצמו.
- <u>לדוגמא</u>: אם נזהה שמ-v3 לעצמו יש מספר שלילי באלכסון, נפעיל את מטריצת המסלולים ונחזיר את המסלול שימצא בתא מ-v3 לעצמו.

מציאת תת מערך עם סכום מקסימלי

- בהינתן מערך בעל n איברים, מהו תת המערך שסכום איבריו הוא הגדול ביותר?
- ומהו תת המערך הזה הקצר ביותר?

- <u>הערה</u>: תת המערך חייב להיות רצף של תאים
- נחלק למקרים:
  - אם כל התאים חיוביים: ניקח את כולם
  - אם יש תאים שליליים: האם הם עוזרים לנו וכדאי לקחת אותם?

- <u>לדוגמא:</u>

2	-1	4

- משתלם לקחת את 1- כי הוא מגדיל לנו את 4 בהמשך.
- אבל כאן:

7	-11	14

- כבר לא ממש משתלם לנו לקחת את 11- כי הוא מקטין לנו את 14.
- מסקנה: לא תמיד כדאי לנו לקחת את השליליים.
- אז מתי כן כדאי?
  - אם המספר השלילי משאיר אותנו בסכום חיובי, כדאי לקחת אותו כי הוא מוסיף לנו להמשך. אבל אם הוא מוריד אותנו לסכום שלילי זה כבר לא משתלם כי אולי בהמשך יהיה משהו טוב יותר שעלול להיפגע.

- **פתרון א':**
- נייצג את המערך במטריצה, כאשר המערך נמצא באלכסון וכל תת מערך מיוצג על ידי תא מסוים במטריצה
- <u>לדוגמא:</u>

-10	2	4	3	-3

-10	-8	-4	-1	-4
	2	6	9	6
		4	7	4
			3	0
				-3

- M[i,j] = M[i+1,j]+M[i,i]
- או
- M[i,j] = M[i,j-1]+M[j,j]
- מה יותר טוב?
  - זה אותו הדבר בדיוק

- **פתרון א':**
- מה יותר מהיר? האם יותר כדאי לעבור על המטריצה בשורות או בעמודות?
- כלומר, האם כדאי for i ואז for j או הפוך?
  - השאלה נראית מטופשת אבל היא חשובה!
  - זה ממש תלוי בקומפיילר שלנו. האם קל לו יותר לגשת ל-i או ל-j
  - אז בגלל שכל כך התגעגעתם לסימולציות ☺ הנה עוד אחת: תעשו סימולציה שפעם עוברת על השורות ופעם על העמודות ותראו לפחות ב-java מה יותר מהיר.
- עוד הצעת ייעול: מה יקרה אם נבצע את המשוואה כך: M[i,j] = M[i+1,j]+M[i,i] → M[i,j] = M[i+1,j]+arr[i]
- האם יותר מהיר לגשת לתא במטריצה או לתא במערך?
  - תעשו סימולציה גם כאן, כדי שתראו את הפערים ותדעו להיות מתכנתים יותר טובים ☺

- לסיכום פתרון א':
- נמלא את המטריצה, נעבור על כולה ב-o(n^2) ונחפש את המספר המקסימלי ביותר. כשנמצא אותו נדע באיזה תא הוא נמצא ולכן נוכל לדעת באיזה תאים במערך נמצא הסכום הזה.
- בהמשך, אם נרצה את תת המערך הקצר ביותר, פשוט נבדוק מה האורך שלו וניקח את האורך הקצר.

תרגול 5-

מציאת תת מערך עם סכום מקסימלי- Best

- בהינתן מערך בעל n איברים, מהו תת המערך שסכום איבריו הוא הגדול ביותר?
- ומהו תת המערך הזה הקצר ביותר?

- <u>הערה</u>: תת המערך חייב להיות רצף של תאים

- בשבוע שעבר הגענו למסקנה: לא תמיד כדאי לנו לקחת את השליליים.
- אז מתי כן כדאי?
  - אם המספר השלילי משאיר אותנו בסכום חיובי, כדאי לקחת אותו כי הוא מוסיף לנו להמשך. אבל אם הוא מוריד אותנו לסכום שלילי זה כבר לא משתלם כי אולי בהמשך יהיה משהו טוב יותר שעלול להיפגע.

- פתרנו את הבעיה בעזרת מטריצה ב-o(n^2) והיום נפתור בעזרת המסקנה ב-o(n).
- איך?
- במערך שקיבלנו יש לנו את כל הנתונים שאנו צריכים:
  - מה הסכום המקסימלי
  - מה תת הקטע עם הסכום המקסימלי- אינדקס התחלה ואינדקס סוף

- <u>דוגמא נוספת</u>:

7	-7	3	9	-15	2

- נבנה מערך עזר:

7	0	3	12	-	2

- מה תת הקטע עם הסכום המקסימלי במערך זה?
- יש 2 אפשרויות:
  - [0,3]
  - [2,3]
- אם ביקשו את תת המערך הקצר ביותר אז נבחר בתשובה השנייה ואם ביקשו את הארוכה ביותר נבחר בתשובה הראשונה.

- מה קורה אם המערך שלנו מעגלי?
- אם נפעיל Best רגיל לא בהכרח נקבל את התשובה הנכונה.
- <u>לדוגמא</u>:

7	-9	2	1

- לאחר הפעלת Best נקבל

7	-	2	3

- ולכן 7 תהיה התשובה, אך התשובה הנכונה היא 7,2,1 שנותן לנו 10.
- אז איך נעשה זאת?
- נסתכל על זה בכמה דרכים:
  - מטריצה
  - מערך כפול
  - אפשרות יעילה יותר

- בואו נראה..

- **פתרון 1: מטריצה**
- נמלא את המטריצה בדיוק כמו בתרגול שעבר: M[i,j] = M[i+1,j]+M[i,i]

7	-2	0	1
	-9	-7	-6
		2	3
			1

- נקבל שהסכום הגדול ביותר הוא 7 במיקום [1,1].
- אך מה אומר לנו התא הצבוע בסגול?
- אם אנחנו מסתכלים על כל תא במשמעות של החל מ-i מסוים ועד j מסוים במערך, אז התא הסגול הוא [3,1] כלומר מתחילים במיקום 3 במערך ומסיימים במיקום 1 במערך.
- איך זה אפשרי?
  - המערך שלנו מעגלי.
- לכן, כל תא במשולש התחתון כאשר i>j מראה לנו תתי קטעים רציפים במערך מעגלי וכל תא במשולש העליון כאשר i<j מראה לנו תתי קטעים רציפים במערך רגיל.

- איך נמלא את המשולש התחתון?
  - בצורה ישירה- אם המשולש העליון מלא כבר, נוכל פשוט להכניס את המשלים שלו בכל הסדרה. כלומר, אם נחשב את הסכום של כל הסדרה (שמופיע לנו בתא הכי ימני למעלה) ונפחית את אחד התאים נקבל את הסכום של המשלים.
  - <u>לדוגמא:</u> [3,1] = S-[2,2] ולכן 10 = 1-(-9) וזה אכן הסכום של תת המערך בקטע [3,1].

7	-2	0	1
	-9	-7	-6
		2	3
			1

- האם זה יתן לנו את הסכום המקסימלי?
  - לאחר שנמלא את כל המטריצה נצטרך לעבור עליה ב-o(n^2) ולחפש את המספר המקסימלי.

- **פתרון 2: מערך כפול**
- ניקח את המערך, נשכפל אותו ונפעיל Best. כך נוכל למצוא את תת הקטע המקסימלי במערך מעגלי.
- <u>לדוגמא:</u>

7	-2	4	5

- אם נפעיל Best נקבל:

7	5	9	14

אך הסכום הגדול ביותר הוא 16 בתת הקטע [3,1].

- אז נשכפל את המערך ונקבל:

7	-2	4	5	7	-2	4	5

- נפעיל עליו Best ונקבל:

7	5	9	14	21	19	23	28

- איך זה קרה?
  - חישבנו פעמיים את אותו המספר וכך חישבנו תתי מערכים הגדולים יותר מאורך המערך עצמו.
- מה נעשה?
  - נפעיל Best כל פעם רק על 4 תאים מתוך המערך הכפול וכך נוודא שאנו לא חורגים.
- כמובן שבתכנות לא נצטרך לשכפל את המערך אלא להשתמש במודולו.
- מה הסיבוכיות?
  - o(n^2)
  - למה?
  - על כל אחד מהאיברים אנו עוברים פעם נוספת על כל איברי המערך.
- לא הצלחנו לרדת בסיבוכיות..
- מה צריך בשביל לרדת?
  - לוותר על מטריצה
  - לא לעבור בכל איבר שוב על כל האיברים

- איך נעשה את זה?

- **פתרון 3:**
- נבין את התובנה הבאה: כדי לקבל את הסכום הגדול ביותר נוכל לקחת את הסכום הכולל ולהפחית ממנו את הסכום המינימלי.
- אבל זה יהיה הגדול ביותר בצורה מעגלית. יכול להיות שבצורה סטנדרטית יהיה לנו סכום גדול יותר.
- לכן, נצטרך לבצע מקסימום בין התוצאה המעגלית לתוצאה הסטנדרטית.
- אבל איך נמצא את התוצאה המעגלית?
  - או שנשנה את אלגוריתם Best שעכשיו ימצא לנו סכום מינימלי
  - או שנשנה את הנתונים- נכפיל ב- 1- את כל האיברים במערך ונקבל את המערך ההופכי שלו. כשנפעיל עליו Best נמצא את הסכום המקסימלי במערך ההופכי שמייצג לנו בעצם את הסכום המינימלי במערך המקורי.
- <u>לדוגמא:</u>

A=

7	-9	2	1

- אז

-A=

-7	9	-2	-1

- כשנפעיל Best על A- נקבל את 9 ולכן נדע שהסכום המינימלי ביותר ב-A הוא 9-, וכמו שאמרנו מקודם הסכום של תת הקטע במעגלי המקסימלי הוא הסכום הכולל פחות הסכום המינימלי- 10 = (9-)-1.
- או בצורה כללית: S - (-Best(-A))
- וכך נמצא את הסכום המקסימלי ביותר במערך מעגלי.

- אבל אמרנו שאולי בצורה הסטנדרטית התוצאה תהיה טובה יותר ולכן סה"כ נקבל:
- Max(Best(A), S-(-Best(-A)))

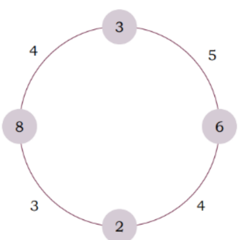
- מה הסיבוכיות עכשיו?
- o(n)- ביצענו פעמיים את Best.

כעת נגיש הכל...

בעיית תחנות הדלק

- נתון מסלול עם תחנות דלק. בכל תחנה ניתן למלא כמות ליטרים מסוימת. כל מקטע דרך בין 2 תחנות צורך כמות ליטרים מסוימת.
- האם ניתן להתחיל בתחנה מסוימת, לעבור בכל התחנות בדרך ולבצע סיבוב שלם בלי שייגמר לנו הדלק?

- **בעיית תחנות הדלק**
- <u>לדוגמא:</u>



- בתור התחלה נצטרך לוודא שכמות הדלק הכוללת שאפשר למלא יותר גדולה מהכמות שאנו צורכים, כי אחרת לא יהיה פתרון לבעיה.
- איך נקבל את הבעיה כקלט?
  - באמצעות 2 מערכים: מערך של תחנות ומערך של עלויות.
  - בדוגמא שלנו נקבל:

A=

3	6	2	8

B=

5	4	3	4

- **בעיית תחנות הדלק**
- כלומר נצטרך לוודא ש: $\sum a_i \geq \sum b_i$
- ניצור מערך נוסף C שהמשמעות שלו תהיה A-B - האם בכלל אפשר להגיע מתחנה אחת לתחנה שאחריה. מערך C יראה כך:

-2	2	-1	4

- כדי להצליח נרצה לצבור כמה שיותר דלק בכל רגע נתון
- או במילים אחרות: נחפש את תת המערך עם סכום מקסימלי כדי שנוכל להמשיך בדרך
- בעצם, נפעיל Best בצורה מעגלית וכך נדע מה הסכום המקסימלי וכן מאיזה אינדקס להתחיל.

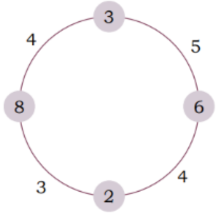
- <u>דוגמת הרצה:</u> ניקח את מערך C

-2	2	-1	4

- ונפעיל עליו Best:

-	2	1	5

- נקבל שהסכום הגדול ביותר בצורה סטנדרטית הוא 5 ואנו צריכים להתחיל מתחנה 2.



A=

3	6	2	8

B=

5	4	3	4

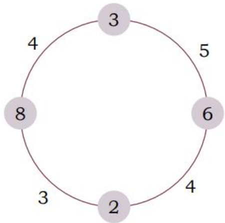
- **בעיית תחנות הדלק**
- כעת נחשב את התוצאה המעגלית: מערך C- נראה כך:

2	-2	1	-4

- נפעיל עליו Best:

2	-	1	-

- נקבל שהסכום הגדול ביותר הוא 2 ולכן זהו הסכום הקטן ביותר המערך C המעגלי.
- נחשב את הסכום הכולל פחות הסכום המינימלי ונקבל: 3-(-2) = 5
- לכן סה"כ נקבל: Max(Best(A), S-(-Best(-A))) = Max(5,5) = 5
- קיבלנו מקטע סטנדרטי – [2,4] ולכן מומלץ להתחיל מתחנה 2 ולהשלים מעגל ממנה.



A=

3	6	2	8

B=

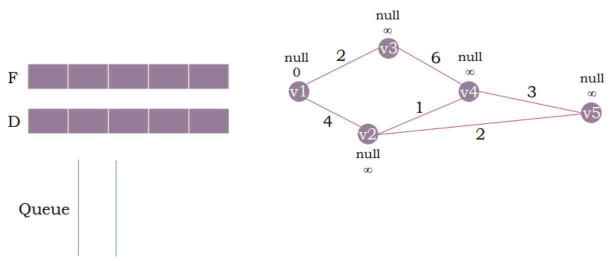
5	4	3	4

תרגול 6-

דייקסטרה

- דייקסטרה הוא אלגוריתם חמדן. המטרה שלו היא לחשב את המסלול הקצר ביותר מקודקוד אחד לכל שאר הקודקודים. (בניגוד לFW שמחשב לנו את המסלול הקצר ביותר מכל הקודקודים לכל הקודקודים).
- איך הוא עובד?
- לכל צלע בין 2 קודקודים יש משקל
- נאתחל את הערך של קודקוד ההתחלה ל-0 ואת שאר הקודקודים ל-∞
- נתחיל לחשב את המסלול הזול ביותר באופן הבא:
- נכניס את קודקוד ההתחלה והשכנים שלו לתור. נעדכן את המרחק מקודקוד ההתחלה לכל שכן ונוציא את קודקוד ההתחלה.
- נבחר את הקודקוד המינימלי בתור, נעדכן את המרחק ממנו אל השכנים שלו, נכניס אותם לתור ונוציא אותו.
- נמשיך כך עד שנוציא את כל הקודקודים מהתור
- כעת, הערך שיש בכל קודקוד הוא המסלול הקצר ביותר מקודקוד ההתחלה אליו.

- ואם נרצה לשחזר את המסלול עצמו? מה נעשה?
- נוכל לשמור עבור כל קודקוד מי האבא שלו. ניצור מערך של אבות ובכל פעם שנאתחל קודקוד מסוים בערך חדש, נרשום במערך מאיפה הגענו אליו.
- לדוגמא:



- פסאודו קוד:

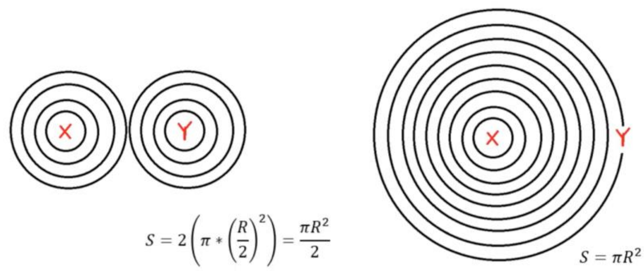
```
Dijkstra(G, v0):
for each v in V:
    d[v] = ∞
    f[v] = null
d[0] = 0
Queue q ← V
while q not empty:
    u ← q.dequeue() //שליפה של הקודקוד המינימלי
    for each w in N(u):
        if d[w] > d[u]+Mat[u,w]
            d[w] = d[u]+Mat[u,w]
            f[w] = u
return f,d
```

- אם נרצה למצוא מסלול מקודקוד ההתחלה לקודקוד מסוים מה נעשה?
- נשנה את תנאי עצירת האלגוריתם שיעצור ברגע שאנו מסיימים לחשב את המרחק של קודקוד זה והוא יוצא מהתור.

- נוכל לשפר את דייקסטרה בצורה הבאה:

דייקסטרה דו כיוונית

- דייקסטרה דו כיוונית היא דייקסטרה שמתקדמים בה גם מקודקוד ההתחלה וגם מקודקוד הסיום בו זמנית.
- הרעיון שעומד מאחורי זה הוא שבדייקסטרה בכל איטרציה אנו מתקדמים בשכבות של הבנים וכך נוצרים מעין מעגלים מסביב לקודקוד.
- אם נסמן את המרחק הקצר ביותר בין X ל-Y ב-R, השטח שנצטרך לבדוק הוא:



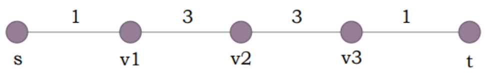
$$S = 2\left(\pi * \left(\frac{R}{2}\right)^2\right) = \frac{\pi R^2}{2}$$

$$S = \pi R^2$$

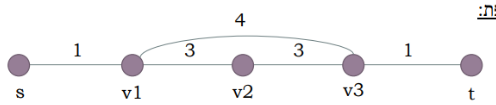
- נגדיר רגע גרף רוורס:
- <u>הגדרה:</u> גרף רוורס G^R עבור גרף G, הוא גרף עם קבוצת קודקודים V וקבוצה של הרוורס של הצלעות E^R, כך שכל צלע (u,v)ϵ E קיימת צלע $(v,u)\ \epsilon\ E^R$ ולהפך.
- <u>לדוגמא:</u>

- כעת, לאחר שהבנו את ההגדרה, ננסה לחשוב איך זה עוזר לנו בפתרון דייקסטרה דו כיוונית.
- אכן, נוכל להשתמש בזה כך:
- נבנה G^R
- נריץ את האלגוריתם של דייקסטרה מ-s ב-G ומ-t ב-G^R
- נחליף לסירוגין בין שלבי דייקסטרה ב-G וב-G^R
- נעצור בשלב שקיים קודקוד v כלשהו שטופל (יצא מהתור) גם ב-G וגם ב-G^R
- לבסוף, נחשב את המסלול הקצר ביותר בין s ל-t.

- אבל האם בהכרח המסלול הקצר ביותר עובר דרך קודקוד v זה?
- <u>לדוגמא:</u>

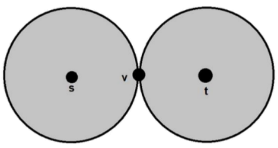


- כאן אנו עוצרים בקודקוד v2 ובהכרח עוברים במסלול הקצר ביותר
- <u>דוגמא נוספת:</u>

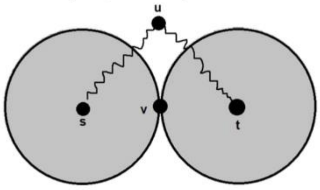


- כאן אנו גם עוצרים בקודקוד v2 אך לא מקבלים את המסלול הקצר ביותר..
- כדי לענות על השאלה נראה למה:

- נסמן ב-d[u] את המרחק באלגוריתם דייקסטרה החל מקודקוד s ב-G ונסמן ב-$d^R[u]$ את המרחק באלגוריתם דייקסטרה החל מקודקוד t ב-G^R
- <u>למה:</u> לאחר שקודקוד מסוים v טופל גם ב-G וגם ב-G^R, אזי קיים מסלול קצר ביותר מ-s ל-t שעובר דרך קודקוד כלשהו u שטופל ב-G או ב-G^R או בשניהם והמשוואה הבאה מתקיימת: $d(s,t) = d[u] + d^R[u]$
- כלומר, אחרי שמצאנו קודקוד v שטופל ב-2 האלגוריתמים נוכל למצוא קודקוד u שגם טופל כבר בשני האלגוריתמים וגם נמצא על המסלול הקצר ביותר. עבור אותו קודקוד u המשוואה מתקיימת.
- נוכיח את הלמה:
- נניח שקיימים לנו 2 מעגלים הנוצרים מריצת האלגוריתם וקודקוד v הוא נקודת המפגש ביניהם.

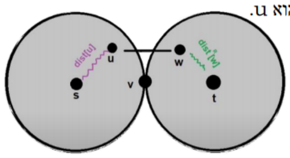


- ראשית נוכיח שאם קיים קודקוד u שלא טופל ע"י אף אחד מהאלגוריתמים ודרכו עובר המסלול הקצר ביותר, אז קיים לפחות קודקוד נוסף שהמסלול הקצר ביותר עובר דרכו- קודקוד v.



- אם באמת יש קודקוד u שלא טופל בכלל, אז קיים קודקוד v נוסף שטופל ע"י האלגוריתמים והמסלול הקצר ביותר עובר דרכו. בהכרח המרחק שיש ב-v נכון כי אחרי שהוא טופל הוא התקבע למסלול הזול ביותר בשני הכיוונים.

- מקרה נוסף שנרצה להוכיח הוא כאשר אין קודקודים שלא טופלו בשני האלגוריתמים חוץ מנקודת המפגש בין שני המעגלים- v.
- נניח שיש מסלול זול בין s ל-t ונניח שהקודקוד האחרון שטופל באלגוריתם הישיר הוא u.

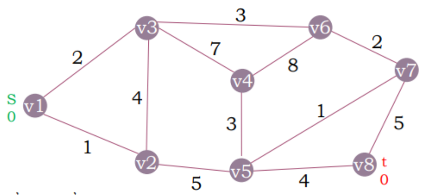


- הקודקוד הבא במסלול הקצר ביותר יהיה w כי הוא לא טופל באלגוריתם הישיר (כי u האחרון שטופל בו) ומצד שני לא יכול להיות שהוא לא טופל כלל כי הוא על המסלול הקצר ביותר. ולכן w חייב להיות בתוך המעגל של t והוא בהכרח טופל ע"י האלגוריתם ההפוך.
- לכן, המרחק כרגע בין s ל-t הוא: $d(s,t) = d[u] + l(u,w) + d^R[w]$. נרצה להוכיח שזה שווה למשוואה מהלמה:
- $d(s,t) = d[u] + d^R[u]$
- בעצם אנו מוכיחים שנקודת המפגש v לא בהכרח נמצאת על המסלול הזול ביותר אלא שיהיה קודקוד אחר u שהמרחק מ-s אליו והמרחק ממנו ל-t יהיה הקצר ביותר.

- $d(s,t) = d[u] + l(u,w) + d^R[w]$
- $d(s,t) = d[u] + d^R[u]$
- בשתי המשוואות יש d[u] ולכן כדי להשוות אותן נצטרך להשוות בין $l(u,w) + d^R[w]$ לבין $d^R[u]$
- נשים לב ש- $d(u,t) \le d^R[u]$ כי בכל רגע של הפעלת האלגוריתם המרחקים רק משתפרים.
- מצד שני, w טופל ע"י האלגוריתם ההפוך ולכן הוא 'הקל' על הצלע (u,w) שיוצאת ממנו ולכן $l(u,w) + d^R[w] \ge d^R[u]$
- בעצם, $l(u,w) + d^R[w]$ הוא המרחק הקצר ביותר בין u ל-t שעובר דרך w, ולכן לפי כלל הסנדוויץ' $d^R[u]$ שווה למרחק הקצר ביותר בין u ל-t.
- מש"ל ☺

- כעת נממש:
- נבצע החלפות בין האלגוריתמים בכל איטרציה עד שנפגוש קודקוד שטופל בשניהם.
- ניקח את כל הקודקודים שטופלו ונחפש מינימום בסכום של האלגוריתם הישיר וההפוך. אותו מספר מינימלי יהיה המרחק הזול ביותר.
- כדי לייצר את המסלול עצמו נשרשר את המסלול מ-s לאותו קודקוד מינימלי והרוורס של המסלול מ-t לאותו קודקוד.

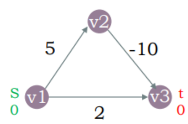
- <u>דוגמת הרצה:</u>



כלומר, העלות הזולה ביותר היא 10
והמסלול הקצר ביותר הוא v1-v2-v5-v8

- לסיכום, הסיבוכיות של דייקסטרה דו כיוונית היא בדיוק כמו הסיבוכיות של דייקסטרה רגילה.
- מבחינת זמן ריצה- זה יכול להיות פי 2 יותר מהיר
- מבחינת זיכרון- תופסים פי 2 אחסון כי צריך לייצר גם את הגרף ההפוך.

- למדנו על אלגוריתם FW שמוצא מסלולים קצרים ביותר מכל קודקוד לכל קודקוד. הסברנו שהוא יכול להתמודד עם משקלים שליליים אבל לא עם מעגלים שליליים. (ברגע שיש מעגל שלילי ברף השאלה של מה המסלול הקצר ביותר כבר לא רלוונטית)
- האם דייקסטרה יכול להתמודד עם משקלים שליליים?
  - לא
- <u>לדוגמא:</u>



- אנו רואים שהוא לא מצא את המסלול הקצר ביותר: 5-, לעומת FW שכן היה מוצא..

תרגול 7-

מציאת תת מטריצה עם סכום מקסימלי

- בהינתן מטריצה בגודל m*n, מהי תת המטריצה שסכום איבריה הוא הגדול ביותר?
- <u>לדוגמא:</u> נתונה מטריצה:

2	1	-3	-4	5
0	6	3	4	1
2	-2	-1	4	-5
-3	3	1	0	3

- מהי תת המטריצה בעלת הסכום המקסימלי עבורה?
- איך נחשב זאת?
- בכמה דרכים:

- **פתרון א'- חיפוש שלם**
- כמו שאנו יודעים, חיפוש שלם עובר על כל תתי המטריצות האפשריות, מחשב את הסכום שלהן ומחזיר לנו בסוף את התשובה הנכונה.
- מה החיסרון שלו?
  - בדר"כ זמן הריצה שלו גדול..
- איך נגדיר מלבן?

- כלומר, נצטרך לקחת את כל האפשרויות של i,j – הפינה השמאלית העליונה של המלבן, וכל האפשרויות של k,l- הפינה הימנית התחתונה של המלבן, לסכום ולקחת את המקסימום.
- כמה לולאות for נצטרך?
  - 6
  - 2 בשביל i,j, 2 בשביל k,l, ועוד 2 בשביל לסכום

- פסאודו קוד:

```
for i=1 to n
  for j=1 to m
    for k=i to n
      for l=j to m
        for x=i to k
          for y=j to l
            נחשב סכום
            נעדכן מקסימום ואינדקסים
```

- סיבוכיות:
  - o(n^6)

- **פתרון ב'- מערך עזר**
- בפתרון זה נוכל להשתמש ב-Best כדי שיעזור לנו.
- נייצר את כל תתי השורות ואת כל תתי העמודות, נפעיל Best ונדע את התשובה.
- <u>לדוגמא:</u>

2	1	-3	-4	5
0	6	3	4	1
2	-2	-1	4	-5
-3	3	1	0	3

- אם נרצה לחבר את שורות 1-2 נעשה זאת כך:

2	4	2	8	-4

- אם נפעיל עליו את Best מה הוא יחזיר?
  - את תאים 1-4
  - אבל מה זה בעצם אומר?
    - שהוא מחזיר לנו את תת המטריצה מתא (0,1) לתא (2,3)
- כלומר, נרצה לקחת את כל תתי השורות האפשריות, להכניס למערך עזר, להפעיל את Best על המערך ולקבל את התשובות עבור תת מטריצה.

2	1	-3	-4	5
0	6	3	4	1
2	-2	-1	4	-5
-3	3	1	0	3

- **פתרון ב'- מערך עזר**
- סיבוכיות:
  - o(n^2) לייצר את כל תתי השורות
  - o(n^2) לסכום את כל תת המטריצה שקיבלנו
  - o(n) להפעיל Best
- סה"כ נקבל: o(n^4)

- פסאודו קוד:

```
for i=1 to n
  for j=i to n
  איפוס של מערך העזר
    for k=i to j
      for l=1 to m
        arr[l] += Mat[k,l]
    Best(arr)
    נעדכן מקסימום ואינדקסים
```

- **פתרון ג'- מטריצת עזר**
- בפתרון זה נוכל להשתמש בתכנות דינאמי.
- הרעיון שעומד מאחורי הפתרון הוא שאם נתונים לנו סכומים חלקיים נוכל בעזרתם לחשב את הסכום הכולל.
- מה הכוונה?
- נגיד שאנו רוצים לחשב את תת המטריצה הזו:

2	1
0	6

  ויש לנו כבר את הסכום של השורה הראשונה = 3, העמודה הראשונה = 2, ו-2-, אז נוכל לקחת את הערך של 6 להוסיף לו את הסכום של השורה הראשונה, להוסיף את הסכום של העמודה הראשונה ולהוריד את הערך של 2 (כי ספרנו פעמיים) וכך לקבל סה"כ = 9
- נוכל לחשב את זה כך: נייצר מטריצת עזר שהערך בכל תא יהיה הסכום של המטריצה מנקודת ההתחלה ועד התא הזה
- <u>לדוגמא:</u>

1	-3
6	3

- מטריצת העזר תראה כך:

1	-2
7	7

- איך זה עוזר לנו?

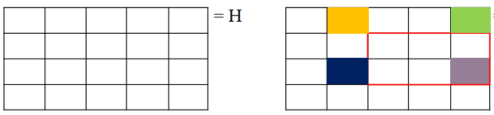
- **פתרון ג'- מטריצת עזר**
- בעזרת מטריצת העזר נוכל לחשב סכומים אחרים.
- <u>דוגמא קטנה:</u> נתונה המטריצה

1	-3
6	3

  ומטריצת העזר שלה

1	-2
7	7

- אם נרצה לחשב את הסכום בתא (2,2) נוכל לקחת את הסכום הכולל של (1,1) עד (2,2) במטריצת העזר, ולהוריד ממנו את הסכומים האחרים וכך להישאר עם הסכום שרצינו.
- ועכשיו בגדול: A = H =



- אם נרצה לחשב את תת המטריצה האדומה המתחילה בתא (1,2) ומסתיימת בתא (2,4) נוכל לחשב זאת כך:
- בתא הסגול יש לנו את הסכום הכולל מ-(0,0) ועד (2,4), בתא הירוק יש לנו את הסכום הכולל מ-(0,0) ועד (0,4), בתא הכחול יש לנו את הסכום הכולל מ-(0,0) ועד (2,1) ובתא הכתום יש לנו את הסכום הכולל מ-(0,0) ועד (0,1).
- כעת, נחשב: התא הסגול – התא הירוק – התא הכחול + התא הכתום = הסכום של תת המטריצה האדומה שזה בדיוק מה שאנחנו צריכים ☺

- **פתרון ג'- מטריצת עזר**
- אבל איך בונים את מטריצת העזר H?
- נתחיל מהתאים הקלים: נמלא תחילה את השורה הראשונה והעמודה הראשונה:
- H[0,j] = H[0,j-1] + A[0,j] וכנ"ל את העמודה (רק H[i,0])
- בשביל למלא את שאר המטריצה נוכל להיעזר במה שמילאנו כבר:
- H[i,j] = A[i,j] + H[i,j-1] + H[i-1,j] – H[i-1,j-1]

A =

2	1	-3	-4	5
0	6	3	4	1
2	-2	-1	4	-5
-3	3	1	0	3

H =

- אחרי שבנינו את מטריצת העזר H, אנו צריכים לעבור עליה ולחשב את הסכומים של כל תתי המטריצות האפשריות:

- for i=1 to n
  - for j=1 to m
    - for k=i to n
      - for t=j to m
        - sum = H[k,t] – H[k,j-1] – H[i-1,t] + H[i-1,j-1]
        - נעדכן מקסימום ואינדקסים

- סיבוכיות:
  - o(n^4)

- **פתרון ד'- super Best**
- איך ניתן עכשיו לשפר את הסיבוכיות ל-o(n^3)?
- מה עזר לנו לרדת בסיבוכיות מ-o(n^6) ל-o(n^4)?
  - להוסיף עזר כדי לשמור נתוני ביניים
  - תכנות דינאמי/Best
- בואו נשתמש בעזרים אלו כדי לשפר עוד קצת את הסיבוכיות:
- במקום לאפס את מערך העזר בכל פעם, נשתמש בקיים ורק נוסיף את השורה החדשה, נפעיל Best ונמצא את כל תתי המטריצות שקשורות לתאים אלו.
- <u>דוגמת הרצה:</u>

2	1	-3	-4	5
0	6	3	4	1
2	-2	-1	4	-5
-3	3	1	0	3

start R =
start C =
end R =
end C =
sum =

- פסאודו קוד:

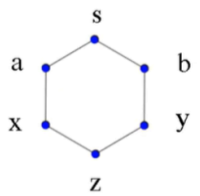
- for i=1 to m
  - איפוס של מערך העזר
  - for j=i to m
    - for k=1 to n
      - arr[k] += Mat[k,j]
    - Best(arr)
    - נעדכן מקסימום ואינדקסים

- סיבוכיות:
  - o(n^3)
  - אבל זה רק סדר גודל.. הסיבוכיות האמיתית היא o(n*m^2)
- האם כדאי לרוץ על השורות ולסכום עמודות או לרוץ על העמודות ולסכום את השורות?
  - תלוי בקלט- אם המטריצה לאורך נרוץ על שורות ואם היא לרוחב אז נרוץ על עמודות.

תרגול 8-

- לפני שנתחיל לדבר על האלגוריתם, ניזכר איך אפשר לממש גרף בתכנות:
  1. להשתמש באובייקטים ולייצר מחלקות: Edge, Vertex, Graph וכו'
  2. להשתמש במטריצת שכנויות
  3. להשתמש ברשימת שכנויות
- הרעיון שעומד מאחורי האלגוריתם הוא סריקה לפי שכבות:
- בהינתן גרף G=(V,E) וקודקוד מסוים s המשמש כמקור, חיפוש לרוחב בוחן בשיטתיות את הקשתות ב-G כדי "לגלות" כל קודקוד שניתן להגיע אליו מ-s.
- הוא מחשב את המרחק (מספר הקשתות המינימלי) מ-s לכל הקודקודים שניתן להגיע אליהם מ-s.
- האלגוריתם פועל גם על גרפים מכוונים וגם על גרפים לא מכוונים.

- <u>לדוגמא:</u>



- בהינתן הגרף הזה ניתן לסרוק אותו ב-2 דרכים:
  - s, a, b, x, y, z
  - s, b, a, y, x, z

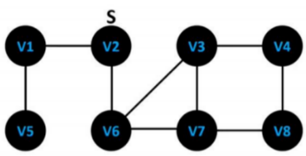
- האלגוריתם מחלק את הקודקודים ל-3 צבעים:
  - לבן- הקודקוד טרם התגלה
  - אפור- הקודקוד התגלה ולא טופל
  - שחור- הקודקוד טופל

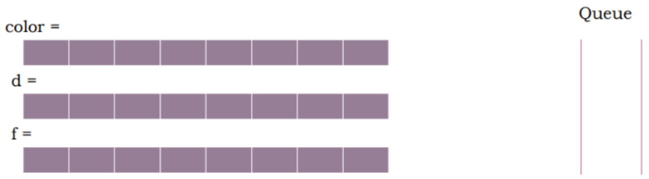
- הוא בונה עץ רוחב ששורשו הוא קודקוד המקור s ולפי השכבות והמרחקים ממנו הוא בונה את שאר הגרף. לכן, קודקוד במרחק k יתגלה לפני קודקוד במרחק k+1.
- כל קודקוד שמגלה את שכניו הופך להיות האבא שלהם. בגלל שכל קודקוד מתגלה פעם אחת אז יש לו רק אבא אחד.
- המסלול שיוצא מ-s לקודקוד v כלשהו בעץ רוחב, מקביל למסלול הקצר ביותר בין s ל-v בגרף.

- <u>פסאודו קוד: (נניח שהגרף מיוצג לפי רשימת שכנויות)</u>

```
BFS(G,s):
for each vertex in V(G)\{s}
    color[u] = WHITE
    d[u] = ∞
    f[u] = null
color[s] = GRAY
d[s] = 0
f[s] = null
Queue q
Enqueue(q, s)
while q not empty:
    u = dequeue(q)
    for each v in N(u):
        if color[v] = WHITE:
            color[v] = GRAY
            d[v] = d[u]+1
            f[v] = u
            Enqueue(q, v)
    color[u] = BLACK
```

- <u>דוגמת הרצה:</u>





- <u>פסאודו קוד: (נניח שהגרף מיוצג לפי רשימת שכנויות)</u>

- <u>BFS(G,s):</u>
- for each vertex in V(G)\{s}
  - color[u] = WHITE
  - d[u] = ∞
  - f[u] = null
- color[s] = GRAY
- d[s] = 0
- f[s] = null
- Queue q
- Enqueue(q, s)
- while q not empty:
  - u = dequeue(q)
  - for each v in N(u):
    - if color[v] = WHITE:
      - d[v] = d[u]+1
      - f[v] = u
      - Enqueue(q, v)
  - color[u] = BLACK

o(|v|)
o(|v|)
o(deg(v))
o(|E|)

סיבוכיות:
- אנו צובעים כל קודקוד רק פעם אחת בלבן ולכן מובטח לנו שכל קודקוד יכנס פעם אחת לתור ויצא רק פעם אחת מהתור- הוצאה והכנסה = o(1), לכן נקבל o(|v|)
- כל איטרציה, אנו עוברים על השכנים של אותו הקודקוד ולכן נעבור על o(deg(v))
- או במילים אחרות נקבל: o(|E|)
- סה"כ נקבל: o(|V+E|)

- מסקנות:
- לאחר הרצת האלגוריתם BFS(G,s) בגרף G=(V,E) נאמר ש:
- כל רכיב הקשירות של s התגלה
- מערך d מכיל את המסלולים הקצרים ביותר מ-s לכל קודקוד
- דרך מערך f ניתן לשחזר את המסלול הקצר ביותר

- כעת, איך ניצור או נדפיס מסלול?
- בעזרת אלגוריתם רקורסיבי:

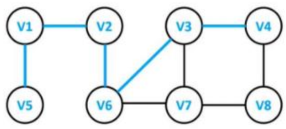
- <u>Print-Path(G,s,v):</u>
- if v=s
  - print s
- else if f[v] = null:
  - print “no path from” s “to” v “exists”
- else
  - Print-Path(G,s,f[v])
- print v

סיבוכיות:
o(|V|)

- שאלות:

1. איך אפשר לדעת האם הגרף קשיר?
   - אם לאחר הרצת האלגוריתם כל הקודקודים צבועים בשחור
   - או אם יש רק null אחד במערך f
   - או אם כל המספרים במערך d סופיים
2. איך אפשר לדעת כמה רכיבי קשירות יש?
   - אם לאחר הרצת האלגוריתם יש עוד קודקודים לבנים, נפעיל את האלגוריתם מחדש על אחד הלבנים ונספור שזו איטרציה שניה. נמשיך כך עד שכל הקודקודים נצבעים בשחור
3. איך אפשר לדעת מיהם הקודקודים בכל רכיב קשירות?
   - כמו בשאלה הקודמת, רק שבכל איטרציה נסמן או נשמור את הקודקודים שבכל רכיב

- קוטר של גרף
- קוטר בגרף הינו המרחק המקסימלי שקיים בין שני קודקודים בגרף.
- <u>לדוגמא:</u> בהינתן הגרף



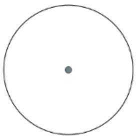
- המרחק המקסימלי בו הוא 5

- אלגוריתם למציאת קוטר (1):

- diam = 0
- for each s in V:
  - call BFS(G,s)
  - max = find max value in d
  - if max > diam:
    - diam = max
- return diam

סיבוכיות:
o(|V| * |V+E|)

- קוטר של גרף
- אלגוריתם נוסף נובע מגיאומטריה במישור:



- אלגוריתם למציאת קוטר (2):

- select s from V
- call BFS(G,s)
- u = find the vertex with max value in d
- call BFS(G,u)
- return max value in d

סיבוכיות:
o(|V+E|)

תרגול 9-

- מהו עץ?
1. גרף קשיר ללא מעגלים
2. גרף קשיר עם n-1 צלעות
3. גרף עם n-1 צלעות ללא מעגלים

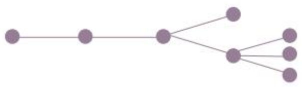
- ברגע שלפחות 2 מההגדרות מתקיימות, אז הגרף בהכרח עץ

- מושגים:
1. מרכז- הנקודה ממנה יוצא הרדיוס
2. רדיוס- קטע המחבר מרכז לשפת המעגל
3. קוטר- הקטע הכי ארוך בין 2 קצוות מעגל. בהכרח עובר במרכז המעגל

- איך מושגים אלו קשורים לעצים?

- המושגים האלה קיימים גם בעצים:
1. קוטר- המרחק המקסימלי בין 2 קודקודים
   - איך מוצאים את הקוטר?
     - מפעילים פעמיים BFS
2. רדיוס- ½ מהקוטר
3. מרכז- אמצע הקוטר

- אותנו יעניין הערך המספרי של המושגים האלה.
- <u>לדוגמא:</u>

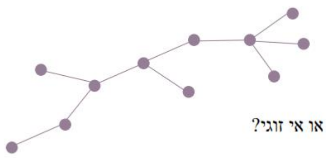


- מה הקוטר?
  - 4
- מה הרדיוס?
  - 2
- כמה מרכזים יש?
  - 1

- <u>דוגמא נוספת:</u>

- מה הקוטר?
  - 3
- מה הרדיוס?
- הגדרה חדשה לרדיוס: המרחק הגדול ביותר מהמרכז לשפת המעגל.
  - 2
- כמה מרכזים יש?
  - 2
- מסקנה: אם הקוטר זוגי אז הרדיוס הוא ½ מהקוטר ויש מרכז אחד. אם הקוטר אי זוגי אז הרדיוס הוא ½ מהקוטר בעיגול כלפי מעלה ויש 2 מרכזים.
- סיבוכיות:
  - $O(|V|+|E|)$

- כעת, נראה אלגוריתם נוסף למציאת קוטר, רדיוס ומרכז:
- **אלגוריתם שריפה**
- ברגע שמורידים עלים אנו עדיין נשארים עם עץ
- למה?
  - לא סגרנו מעגלים ועדיין נשארנו עם גרף קשיר ונשארו עדיין n-1 צלעות
- הרעיון של האלגוריתם הוא 'לשרוף' את העלים בכל פעם עד שנשאר רק עם המרכז/ים.
- <u>לדוגמא:</u>



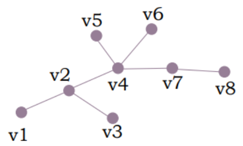
- נשאל:
1. הקוטר זוגי או אי זוגי?
   - זוגי
2. כמה מרכזים יש?
   - 1
3. מה הרדיוס?
   - כמות השריפות
4. מה הקוטר?
   - פעמיים כמות השריפות

- **אלגוריתם שריפה**
- <u>דוגמא נוספת:</u>

- נשאל:
1. הקוטר זוגי או אי זוגי?
   - אי זוגי
2. כמה מרכזים יש?
   - 2
3. מה הרדיוס?
   - כמות השריפות + 1
4. מה הקוטר?
   - פעמיים הרדיוס פחות 1

- קיבלנו נוסחאות שונות עבור קוטר זוגי ואי זוגי

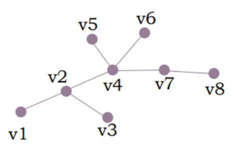
- **אלגוריתם שריפה**
- איך אנו מקבלים את הקלט?
- מערך שכנויות
- <u>לדוגמא:</u>



V1	v2			
V2	v1	v3	v4	
V3	v2			
V4	v2	v5	v6	v7
V5	v4			
V6	v4			
V7	v4	v8		
V8	v7			

- איך נזהה מי העלים?
  - כל מי שיש לו רק שכן אחד ברשימת שכנויות
- נשמור את כל העלים בתור (שמירה כזו היא $O(|V|)$) ואז נשרוף אותם.
- כמו שאנחנו רואים, זה קצת מורכב ולא בהכרח יעיל..
- בואו ננסה לשפר ולייעל

- באמצעות מערך עזר:
- כל תא במערך ייצג לנו את הדרגה של אותו קודקוד. כך המעבר יהיה מהיר יותר וקל יותר.
- כאשר הערך בתא הוא 1 נדע שזה עלה ונכניס אותו לתור. כאשר שרפנו עלה, נוריד דרגה אחת מהשכן שלו.
- <u>לדוגמא:</u>



1	3	1	4	1	1	2	1
v1	v2	v3	v4	v5	v6	v7	v8

- שיפור נוסף, להוסיף משתנה counter שיספור לנו כמה עלים עוד לא שרפנו, וכשהמשתנה יהיה שווה ל-1 או 2 נדע לעצור את השריפה כי הגענו למרכז.
- סיבוכיות:
  - $O(|V|)$

תרגול 10-

- **בניית עץ מרשימת דרגות**
- בהינתן רשימת דרגות הקודקודים בגרף מסוים, האם רשימה זו באמת יכולה להיות רשימת דרגות? האם היא יכולה להיות רשימת דרגות של עץ?
- לדוגמא:

$$1,1,1,2,2,3,4$$

- לפני שנגיע לתשובה, נראה כמה הגדרות:

1. בכל עץ $|E| = |V|-1$ כלומר, כמות הצלעות שווה לכמות הקודקודים פחות 1
2. $\sum_{v \in V} \deg(v) = 2|E|$

- מסקנה: $\sum_{v \in V} \deg(v) = 2(|V|-1)$
- אם כך, צריך לוודא שסכום הדרגות יהיה פעמיים מספר הצלעות. אך האם זה מספיק?

- **בניית עץ מרשימת דרגות**
- לדוגמא:

$$1,1,1,1,2,2,3,4$$

- נסכום את האיברים: $1+1+1+1+2+2+3+4 = 15$
- נסיק מסקנה לגבי כמות הקודקודים:

$$2(|V|-1) = 15$$
$$|V|-1 = 7.5$$
$$|V| = 8.5$$

- דוגמא נוספת:

$$1,1,1,2,2,3,4$$

- נסכום את האיברים: $1+1+1+2+2+3+4 = 14$
- נסיק מסקנה לגבי כמות הקודקודים:

$$2(|V|-1) = 14$$
$$|V|-1 = 7$$
$$|V| = 8$$

- קיבלנו סתירה. למה?
  - כי כמות הקודקודים ברשימה היא 7

- **בניית עץ מרשימת דרגות**
- אם כך, מתי רשימת דרגות יכולה להיות עץ?

1. כאשר $\sum_{v \in V} \deg(v) = 2|E|$
2. כאשר יש לנו מספיק נתונים לכמות הקודקודים

- בואו נראה דוגמא נוספת:

$$1,1,1,1,2,2,3,3$$

- נסכום את האיברים: $1+1+1+1+2+2+3+3 = 14$
- נסיק מסקנה לגבי כמות הקודקודים:

$$2(|V|-1) = 14$$
$$|V|-1 = 7$$
$$|V| = 8$$

- וזה אכן מתקיים!

- **בניית עץ מרשימת דרגות**
- רעיון האלגוריתם:
- משפט: בכל עץ יש לפחות 2 עלים
- לכן, ננסה לחבר (במידת האפשר) עלה לקודקוד שהדרגה שלו גדולה מ-1 (כדי שיישארו עלים)

- לדוגמא:

v1	v2	v3	v4	v5	v6	v7	v8
1	1	1	1	2	2	3	3

- ניקח את הקודקוד הראשון שמייצג עלה ונחבר אותו עם הקודקוד הראשון במערך שהוא לא עלה. לאחר החיבור נוריד את דרגות הקודקודים האלו.
- נמשיך כך עד שכל הערכים במערך יהיו 0.

- **בניית עץ מרשימת דרגות**
- פסאודו קוד:

```
BuildTreeFromDegreesArray(deg[]):
N = deg.size()
sum = 0
tree[N] = null
for i=1 to N:
      sum += deg[i]
if sum/2 + 1 != N:
      print(Not a tree degrees array)
      return
deg[] = sort(deg[])
j = first index that deg[j]>1
for i=1 to N-2:
      tree[i] = j
      deg[j] = deg[j]-1
      if deg[j] == 1
            j = j+1
tree[N-1] = N
return tree
```

- כאשר מערך tree מייצג את האבות

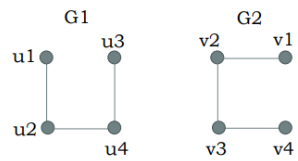
- **סכום המטריצה הגדול ביותר**
- בהינתן מטריצה המלאה ב-1 ו- -1 בצורה רנדומלית, איך נהפוך את המטריצה לבעלת הסכום הגדול ביותר?
- ניתן לקחת כל שורה וכל עמודה ולהכפיל ב- -1

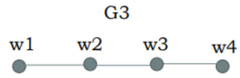
- תהליך זה הוא סופי. למה?
- כי יש לנו חסם עליון: לא יכול להיות שהסכום יהיה גדול יותר ממספר התאים במטריצה

- עליכם לבצע סימולציה בה תגרילו מטריצת אחדות ומינוס אחדות ותמצאו את המספר המינימלי של הכפלות ב- -1 כדי להגיע לסכום הגדול ביותר במטריצה זו
- שאלת מחשבה: האם סדר ההכפלות של השורות והעמודות משנה? האם זה יוריד את מספר ההכפלות?

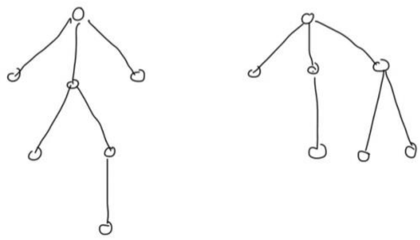
תרגול 11-

- **עצים איזומורפיים**
- נתחיל מהגדרה של גרפים איזומורפיים:
- גרפים G ו-H הם איזומורפיים אם קיימת פונקציה f: V(G)->V(H) חח"ע ועל, כך שעבור כל u,v є V(G) מספר הקשתות המקשרות בין u ו-v זהה למספר הקשתות המקשרות בין f(u) ו-f(v).
- במילים אחרות, הם מייצגים את אותו הגרף רק נראים בצורה שונה.
- <u>לדוגמא:</u>



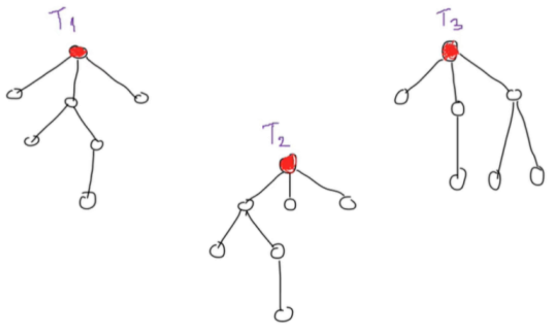


- בהינתן 2 עצים, האם הם איזומורפיים?
- <u>לדוגמא:</u>

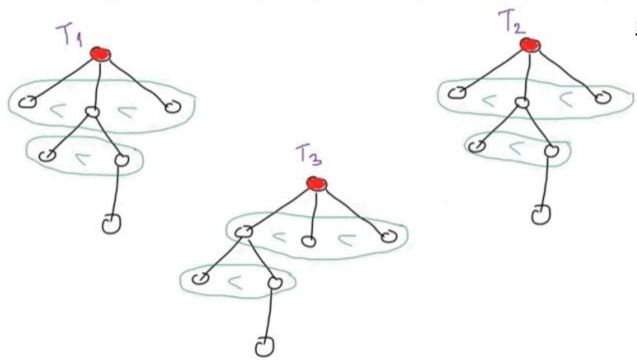


- כן, אנו רואים שזה אותו העץ בשינוי צורה..

- **עצים איזומורפיים- עצים עם שורש**
- בעצים אלו, אנו מגדירים מיהו אותו קודקוד עליון שממנו אנו מתחילים את כל החישוב.
- איך נדע עכשיו האם הם איזומורפיים?
- <u>לדוגמא:</u>

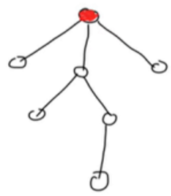


- נראה שאכן T1 ו-T2 איזומורפיים, אבל T3 לא איזומורפי אליהם כי יוצא ממנו רק בן אחד שהוא עלה בניגוד אליהם
- הרעיון שעומד מאחורי זה הוא שצריך להחליט על סדר בין הבנים של כל קודקוד וכך נוכל להשוות ביניהם.
- כלומר, ננסה שהבנים יהיו ממויינים בסדר כלשהו וכך נוכל להכריע בבעיית איזומורפיזם.
- <u>לדוגמא:</u>



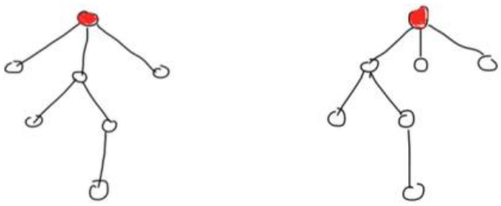
- אכן ניתן לראות ש-T1 ו-T2 איזומורפיים, אך T3 לא כי לאחר המיון הסדר שלו שונה..

- לפני שנחליט על מיון, נצטרך לראות איך בכלל עוברים על העץ..
- נסרוק את הגרף לעומק, ונשמור את הפעולות שלנו: בכל פעם שנרד נרשום 0 ובכל פעם שנעלה נרשום 1.
- <u>לדוגמא:</u>



- לאחר הסריקה נקבל מחרוזת התואמת את העץ. אם נצליח למצוא עץ נוסף בעל אותה מחרוזת נוכל לומר שהם איזומורפיים.

- יופי, אז אחרי שיש לנו פתרון בוא נראה אותו מתקיים. ניקח 2 עצים איזומורפיים ונראה האם המחרוזות שלהם אכן שוות:



- אנו רואים שלא קיבלנו מחרוזות שוות. אז האם העצים איזומורפיים או לא?
- נכון, עוד לא מיינו את הבנים ולכן זה לא עבד.. אז נצטרך למיין את הבנים לפני שנעתיק את המחרוזות שלהם למחרוזת של האבא

- אלגוריתם ליצירת מחרוזת המתארת עץ מושרש:
  1. נסרוק את הגרף לעומק
  2. כשנגיע לעלה, נגדיר את המחרוזת שלו כ-01
  3. כאשר נגדיר את המחרוזת של האבא (נגדיר אותה רק אחרי שנסיים עם כל המחרוזות של הבנים שלו) נתחיל אותה ב-0, נסיים אותה ב-1 והאמצע יהיה שרשור ממוין של מחרוזות הבנים.

- כאשר נרצה להשוות בין עצים ולראות האם הם איזומורפיים, נפעיל את האלגוריתם על כל אחד מהעצים ונראה האם נקבל מחרוזות שוות.

- **עצים איזומורפיים- עצים בלי שורש**
- מה לגבי עצים לא מושרשים?

- פתרון א':

נבחר בכל פעם קודקוד אחר שיהיה השורש, נפעיל עליו את האלגוריתם, עד שנמצא מחרוזות שוות, או עד שנעבור על כל הזוגות האפשריים ולא נמצא כלום..

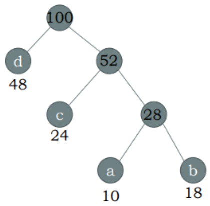
- פתרון ב':

אם נבחר קודקוד שיכול להיות שורש אופציונאלי, נוכל להפעיל את האלגוריתם רק פעם אחת ממנו ולקבל את התשובה.

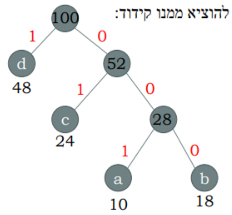
- מי הוא יכול להיות?
  - המרכז! כל העץ יכול לצאת ממנו, ואם נמצא אותו (יש לנו אלגוריתם יעיל למצוא אותו) אז נוכל להפעיל את האלגוריתם רק פעם אחת ולקבל תשובה.
- שאלת מחשבה: מה קורה אם יש יותר ממרכז אחד? האם עץ עם מרכז אחד יכול להיות איזומורפי לעץ עם שני מרכזים? והאם 2 עצים עם שני מרכזים יכולים להיות איזומורפיים?

תרגול 12-

- **קידוד הופמן**
- מטרה: לקודד טקסט בשימוש במינימום ביטים ולהצליח לפענח אותו בחזרה בצורה מדויקת
- <u>לדוגמא</u>: a=10%, b=18%, c=24%, d=48%
- נרצה שככל שהתו מופיע יותר פעמים הוא יהיה קרוב לשורש העץ, וככל שהוא מופיע פחות הוא יהיה קרוב לעלים.
- בוא נבנה את העץ:



- קיבלנו עץ שניתן להוציא ממנו קידוד:



a = 001
b = 000
c = 01
d = 1

- כעת ניתן לחשב כמה זיכרון זה יתפוס:
- 3*10 + 3*18 + 2*24 + 1*48 = 180
- מה אומר ה-180 הזה?
  - אם יש לנו מחרוזת באורך x שאלו האחוזים של הקידוד אז כל תו יתפוס 1.8 עבור כל תו בממוצע

- איך ניתן לממש את העץ?
- בכמה דרכים:
- **<u>דרך א'- סטטית:</u>**
- ברגע שיש 4 תווים אנו יודעים מה יהיה גודל העץ. איך?
  - n-1 קודקודים פנימיים + n קודקודים של התווים = 2n-1
  - כך ניתן פשוט ליצור מטריצה בגודל הרלוונטי
- נחזור לדוגמא מקודם: a=10%, b=18%, c=24%, d=48%
- כדי לא לקבל קידודים שונים, נגדיר שהבן עם הערך הקטן יותר יהיה הבן השמאלי.
- כמובן שלא נשכח לסמן בכל סוף איטרציה במי השתמשנו כבר

בן שמאלי 1	בן ימני 0	אבא			
			10	a	**1**
			18	b	**2**
			24	c	**3**
			48	d	**4**
					5
					6
					7

- למה לא כל המטריצה מלאה?
  - נכון מאוד, לכל העלים אין בנים, ולשורש אין אבא
- האם המטריצה היא עץ?
  - כן, במקום לייצר עם Node וקשרים אחרים, פישטנו את זה. כל אחד יודע מי האבא שלו, מי בן שמאלי וכו'
- מה הסיבוכיות?
  - כמה איטרציות היו?
    - n-1
  - מה עושים בכל איטרציה?
    - מחפשים 2 תווים מינימלים- n+logn-2
    - מקצים להם אבא- o(1)
    - מציבים לאבא את הבנים- o(1)
    - מציבים לבנים את האבא- o(1)
    - סוכמים אחוזים לאבא- o(1)
    - מסמנים בנים כ'משומשים'- o(1)
  - לכן סה"כ נקבל- o(n^2)
- איך ניתן לשחזר את הקידוד?
  - נשאל כל תו מי אבא שלו, נשאל את האבא האם התו בן ימני או שמאלי, ונמשיך כך עד שנגיע לשורש.

בן שמאלי 1	בן ימני 0	אבא			
		5	10	a	**1**
		5	18	b	**2**
		6	24	c	**3**
		7	48	d	**4**
1	2	6	28		**5**
3	5	7	52		**6**
4	6		100		**7**

- <u>פסאודו קוד של **דרך ב'**:</u>
- לפני שנתחיל, מה הפעולה שחוזרת על עצמה הכי הרבה פעמים?
  - חיפוש מינימום
- אם נחפש עכשיו כל כך הרבה פעמים זה יעלה לנו ביוקר, אז במה נוכל להעזר?
  - ערמת מינימום- הממומשת ע"י תור עדיפויות

```
Huffman(C):
o(1)      n = |C|
o(n)      Q = C
          for i=1 to n-1:
o(1)          z = allocate Node()
o(logn)       x = z.left = Extract_Min(Q)
o(logn)       y = z.right = Extract_Min(Q)
o(1)          f(z) = f(x) + f(y)
o(logn)       insert(Q,z)
          return Extract_Min(Q)
```

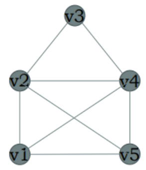
סיבוכיות?
o(nlogn)
הצלחנו להוריד את הסיבוכיות כאשר שינינו את מבנה הנתונים

- **<u>דרך ג':</u>**
- נניח ונותנים לנו את התווים ממויינים. נוכל לשפר עוד את הסיבוכיות אם נשתמש במיון וכך השליפה מהתור תהיה ב-o(1):
- נשתמש ב-2 תורים כך:

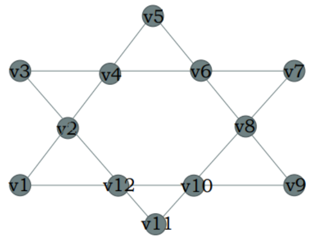
- כמה איטרציות עשינו?
  - n-1
- בכל איטרציה איזה פעולות עשינו?
  - השוואה, שליפה והכנסה לתור
- זאת אומרת שהסיבוכיות בסה"כ היא o(n)

תרגול 13-

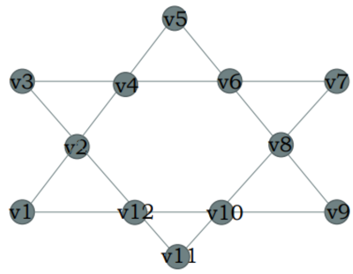
- **אוילר**
- <u>הגדרה</u>: יהי G(V,E) גרף לא מכוון.
מסלול $P_{x,y}$ נקרא **מסלול אוילר** ב-G, אם הוא עובר בכל הצלעות של G כך שכל צלע מופיעה בו פעם אחת בלבד, ובנוסף $x \neq y$. (כלומר, מתחילים בקודקוד אחד ומסיימים בקודקוד אחר)
- <u>לדוגמא</u>:



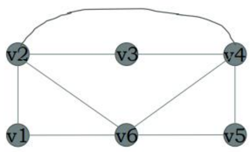
- <u>הגדרה</u>: יהי G(V,E) גרף לא מכוון.
מעגל אוילר ב-G הוא מסלול אוילר סגור, כלומר מסלול העובר בכל צלעות הגרף פעם אחת בלבד והקודקוד ההתחלתי הוא גם קודקוד הסיום.
- <u>לדוגמא</u>:



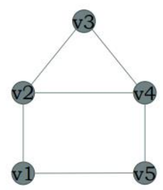
- <u>הגדרה</u>: יהי G(V,E) גרף לא מכוון.
גרף אוילריאני הוא גרף המכיל מעגל אוילר
- <u>לדוגמא</u>:



- <u>משפטי זיהוי אוילר בגרפים</u>:
1. יש בגרף G מעגל אוילר (גרף אוילריאני) אם ורק אם G קשיר וכל דרגות הגרף זוגיות.
2. יש בגרף G מסלול אוילר אם ורק אם G קשיר ובדיוק 2 קודקודים בעלי דרגות אי זוגיות.
- <u>לדוגמא</u>:



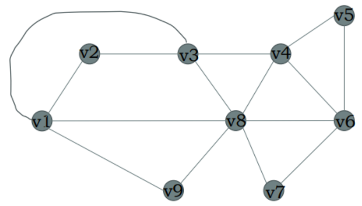
מעגל



מסלול

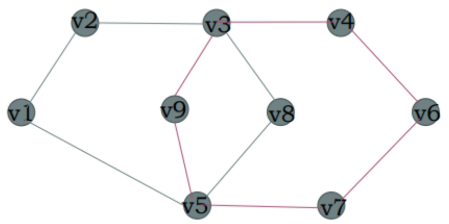
- <u>אלגוריתם למציאת מעגל אוילר- רעיון</u>:
1. נתחיל מקודקוד מסוים ונתחיל ממנו מסלול (לא חוזרים על צלע פעמיים) עד שהוא סוגר מעגל
2. אם עברנו על כל הקודקודים נעבור לסעיף 3. אחרת, נוריד את המעגל מהגרף ונחזור לסעיף 1
3. אם יש רק מעגל אחד- סיימנו. אחרת, נאחד בין המעגלים: נתחיל מסלול ממעגל מסוים, וכאשר נגיע למעגל חדש נסגור קודם את המעגל החדש ואז נחזור למעגל הקודם

- <u>לדוגמא</u>:

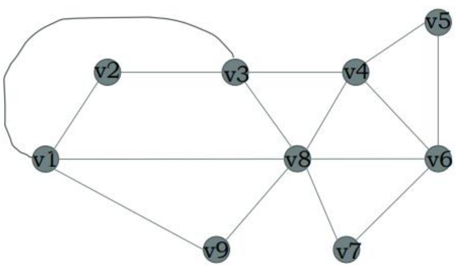


- **איך נאחד בין המעגלים?**

- <u>לדוגמא</u>:

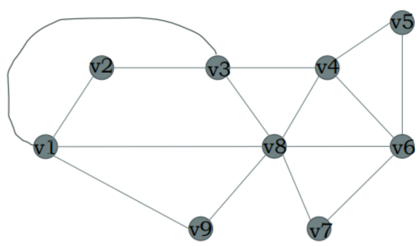


- <u>נחזור לדוגמא שלנו</u>:



- <u>אלגוריתם נוסף למציאת מעגל אוילר- רעיון</u>:
1. בחר קודקוד
2. התחל מסלול ממנו
3. אם סגרך מעגל, תדפיס את הקודקודים בחזרה עד שיהיה אפשר להמשיך בדרך אחרת

- <u>לדוגמא</u>:

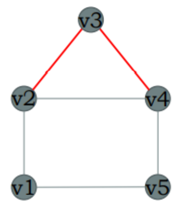


פסודו קוד:

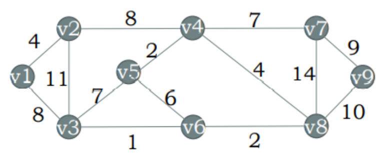
```
EulerCycle(G):
stack S = null, stack C = null
S.push(v0)
while S is not empty:
    u = s.top()
    if deg(u)=0:
        S.pop()
        C.push(u)
    else
        v = v є N(u)
        S.push(v)
        E = E – {u,v}
return C
```

```
Prim(G, root):
o(n)      for each v in V(G):
              visited[v] = false
              key[v] = inf
              father(v) = NULL
          key[root] = 0
o(n)      Q = V(G) // Q = min heap
          while (Q is not empty) // or for i=1 to N-1
o(logn)       u = Extract_Min(Q)
              for each v in N(u)
                  if(visited[v] == false and key[v] > weigh
                      key[v] = weight(u,v)
                      father[v] = u
o(logn)               decreaseKey(Q,v, weight(u,v))
              visited[u] = true
```

- **עץ פורש מינימלי- פרים**
- האלגוריתם של פרים הוא אלגוריתם חמדני המשמש למציאת עץ פורש מינימלי בגרף ממושקל לא מכוון.
- האלגוריתם מתחיל את בניית העץ מקודקוד פתיחה שנבחר באקראי.
- בכל צעד האלגוריתם נוסיף לעץ את הצלע בעלת המשקל המינימלי מבין אלה היוצאות מקודקודי העץ ולא סוגרות מעגל.
- איך נוודא שלא סגרנו מעגל?
  - כאשר נרצה להוסיף צלע לעץ, נוודא שרק אחד מהקודקודים קיים בעץ.
- לדוגמא:



- איך נשמור את העץ?
  - בעזרת מערך אבות
- בנוסף, נרצה לדעת את המשקל המינימלי של העץ הפורש שמצאנו ולכן נשתמש גם במערך משקלים שבכל את נשמור את המשקל המינימלי שיצא מאותו קודקוד בעץ שנבחר
- לדוגמא:



	father	key	visited
v1			
v2			
v3			
v4			
v5			
v6			
v7			
v8			
v9			