# סיכום תרגולים

# קורס אלגוריתמים

# 67504 Algorithms

סמסטר ב' 2021

**מתרגלים:** דניאל רותם וגיא הכהן

**מסכמת:** קרן בן אריה

# תוכן עניינים


תרגול 1: חזרה 3-9

תרגול 2: אלגוריתמים חמדניים 10-16

תרגול 3: מטרואידים 17-24

תרגול 4: תכנון דינמי 25-29

תרגול 5: תכנון דינמי 30-37

תרגול 6: רשתות זרימה 38-45

תרגול 7: רשתות זרימה 46-53

תרגול 8: תכנון לינארי 54-62

תרגול 9: אלגוריתמי קירוב 63-68

תרגול 10: משקולות כפליים 69-74

תרגול 11: אלגוריתמים הסתברותיים 75-83

תרגול 12: אלגוריתמים הסתברותיים 84-90

תרגול 13: אלגוריתמים הסתברותיים של פולינומים 91-97

יום שלישי 16.3.21

מקרא:
- הוכחה
- דוגמא
- משפט
- טענה
- הגדרה

# אלגוריתמים
# שבוע 1 - תרגול 1

## מנהלות

**תרגילים:**
* בקורס יינתנו N תרגילים, בערך אחד כל שבוע.
* הציון של התרגילים יחושב לפי N-2 הטובים, ויהווה 10% מציון הקורס.
* ניתן להגיש בזוגות.
* תרגיל מוקלד יזכה ב-3 נק' בונוס.
* 3 נק' בונוס יינתנו על אי-איחור.

**בחנים:**
* בקורס יינתנו 4 בחנים במהלך הסמסטר, חובה להגיש 3.
* כל בוחן מהווה 7% מהציון הסופי.
* הגרלת שאלה באקראי מתרגילי הבית.

## מה נלמד בקורס?

* ארגז כלים של אלגוריתמים.
* שיטות וגישות שונות לפתרון בעיות (דוג' משפחת אלגוריתמים: חמדניים)
* איך יודעים שאלגוריתם מתאים לפתרון בעיה - הוכחת נכונות.

## נושאי התרגול:

* טענות לוגיות
* אינדוקציה
* חסמים אסימפטוטיים
* זמני ריצה

## טעויות נפוצות בטענות לוגיות

1. נאמר על טענה שהיא טריוויאלית אם היא נובעת ישירות מההגדרה.
* גרף $G$ עם 2 רכיבי קשירות אינו עץ. טענה טריוויאלית.
* בעץ עם $n$ קודקודים יש $n-1$ צלעות. טענה לא טריוויאלית.

2. נאמר שטענה מתקיימת באופן ריק אם היא מהצורה "לכל האיברים בקבוצה $S$..." וגם $S=\emptyset$.
* כל הסטודנטים שצופים בתרגול, יושבים על כדור פיצין. לא מתקיימת באופן ריק
* כל הפינגווינים שצופים בתרגול, הם בעלי מקור ירוק. מתקיימת באופן ריק.

## אינדוקציה

אינדוקציה היא שיטה להוכחת מספר גדול (בד"כ אינסופי בן-מניה) של טענות ב"מכה" אחת.

**בסיס:** נוכיח מספר קטן של טענות באופן פורמלי בודדים. בד"כ $a_1$.

**הנחה:** נניח נכונות של קבוצת טענות מתוך $A$. לרוב $a_n$

**צעד:** נשתמש בנכונות הטענה מההנחה בשביל להוכיח את $a_{n+1}$

**דוגמה:** לכל מספר טבעי $n\in\mathbb{N}$ מתקיים: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

**בסיס:** עבור $n=1$. $\sum_{i=1}^{1} i = 1 = \frac{1(1+1)}{2}$ ✓

**הנחה:** נניח שהטענה נכונה עבור $n$.

**צעד:** עלינו להראות כי נכונות עבור $n$ גוררת נכונות $n+1$.

$$\sum_{i=1}^{n+1} i = \sum_{i=1}^{n} i + (n+1) \underset{\text{הנ}}{=} \frac{n(n+1)}{2} + (n+1) = \frac{n(n+1)+2(n+1)}{2} = \frac{(n+1)(n+2)}{2}$$

כנדרש.

## טעויות נפוצות באינדוקציה:

טענה שקרית

1. **טענה:** הקבוצה $\mathbb{N}$ סופית.

**בסיס:** עבור $n=1$, הקבוצה $\{1\}$ אכן סופית.

**הנחה:** נניח נכונות עבור כל $1\le k\le n$.

**צעד:** נתבונן בקבוצה $\{1,\dots,n,n+1\}$ ניתן לכתוב אותה גם כך- $\{1,..,n\}\cup\{n+1\}$. מהנחת האינדוקציה $\{1,..,n\}$ סופית ומכיוון שגודלה של $\{1,\dots,n,n+1\}$ הוא 1 יותר מגודלה של $\{1,..,n\}$, כלומר סופית.

**בעיה:** בסה"כ הוכחנו שעבור כל $n\in\mathbb{N}$ קבוצת הטבעיים עד אליו סופית. הטענה שניסינו להוכיח לא נמצאת בסט הטענות שהוכחנו כמו בהגדרה ⊛.

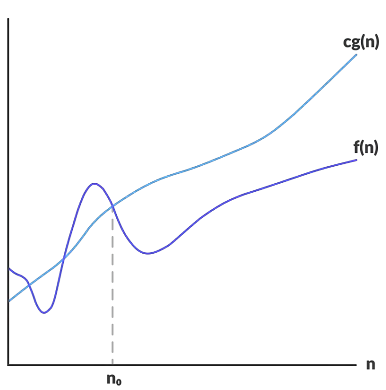
+ כשמבצעים אינדוקציה, חשוב לשים לב שהצעד מאפשר מעבר ממקרה הבסיס לטענה הבאה (דוגמת הסוסים באותו הצבע).

## חסמים אסימפטוטיים:

### חסם עליון - O גדול

**הגדרה:** יהיו $f(n)$ ו-$g(n)$ שתי פונקציות כך: $g,f:\mathbb{N}\to\mathbb{R}^+$.

$$O(g(n))=\{f(n)\mid \exists c>0 \wedge \exists n_0\geq 1 \text{ s.t } \forall n\geq n_0\ f(n)\leq c(g(n))\}$$



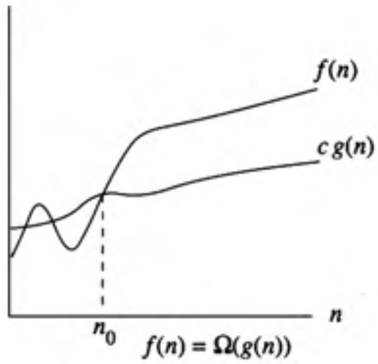
f(n) = O(g(n))

ניתן לראות שקיים $c>0$ ו-$n_0\geq1$ שעבור כל $n\geq n_0$ הפונקציה של $g(n)$ תמיד מעל הפונקציה $f(n)$. $g$ חוסמת את $f$ מלמעלה.

## חסם תחתון - $\Omega$ גדול

**הגדרה:** יהיו $f(n)$ ו-$g(n)$ שתי פונקציות כך: $g,f:\mathbb{N}\to\mathbb{R}^+$.

$$\Omega(g(n)) = \{f(n) \mid \exists c>0 \wedge \exists n_0>1 \text{ s.t } \forall n\ge n_0 \ f(n)\ge c(g(n))\}$$




←

ניתן לראות שקיים $c>0$ ו-$n_0>1$ שעבור כל $n\ge n_0$ הפונקציה של $f(n)$ תמיד מעל הפונקציה $c\cdot g(n)$. g חוסמת את f מלמטה.

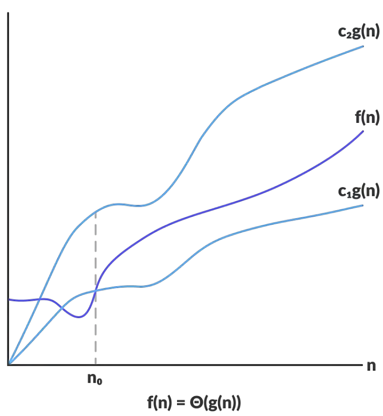
## חסם הדוק - $\Theta$ גדול

שילוב של עליון ותחתון

**הגדרה:** יהיו $f(n)$ ו-$g(n)$ שתי פונקציות כך: $g,f:\mathbb{N}\to\mathbb{R}^+$.

$$\Theta(g(n)) = \{f(n) \mid \exists c_1,c_2>0 \wedge \exists n_0>1 \text{ s.t } \forall n\ge n_0 \ c_1\cdot g(n)\le f(n)\le c_2(g(n))\}$$

חסם הדוק = חסם מלמעלה ומלמטה




←

ניתן לראות שקיימים $c_1,c_2>0$ ו-$n_0>1$ שעבור כל $n\ge n_0$ הפונקציה של $f(n)$ תמיד בין הפונקציות $c_2\cdot g(n), c_1 g(n)$.

* נניח שיש $n_1$ ו-$n_2$ מתאימים ל-$c_1,c_2$, נבחר $n_0=\max(n_1,n_2)$.

דוגמאות: * $500\sqrt{n} = O(n)$

עבור $C=500$ ו- $n_0=1$ ונקבל: $500\sqrt{n} \le 500\cdot n = c\cdot n \quad \forall n>1$

* $10\cdot n\cdot \log_2(\log_2(n)) = \Omega(n)$.

עבור $C=10$ ו- $n_0=4$ ונקבל:

$$\forall n>4 \quad 10\cdot n\log(\log(n)) \ge 10\cdot n\log(\log(4)) \ge 10\cdot n\cdot \log(2) = 10n = c\cdot n$$

## זמני ריצה ויעילות אלגוריתמים

הגדרה: נאמר על אלגוריתם שהוא רץ בזמן פולינומיאלי באורך הקלט (אלגוריתם פולינומי) אם קיים $m\in\mathbb{N}$, כך שמספר הפעולות המקסימלי שמבצע האלגוריתם על קלט בגודל $n$ הוא $O(n^m)$.

דוגמאות:

$n^2$ - פולינומי ✓ $\qquad$ $\log(n)$ - פולינומי כי $\log(n)=O(n)$ ✓

$n^3$ - פולינומי ✓ $\qquad$ $\sqrt{n}$ - פולינומי כי $\sqrt{n}=O(n)$ ✓

$n^n$ - לא פולינומי ✗ $\qquad$ $2^n$ - לא פולינומי ✗

* בקורס הזה, נתייחס לאלגוריתמים כיעילים אם זמן הריצה שלהם פולינומי, ולא יעילים אם הוא לא פולינומי.

## זמני ריצה פסודו-פולינומיאליים

קלט: מספר טבעי $n\in\mathbb{N}$.

נשים לב שגודל הקלט הוא $\log(n)$ כי מדובר בגודל "ייצוג בינארי" של מספר כלשהו.

פלט: 1 אם $n$ ראשוני, 0 אחרת

אלגוריתם: עבור $i=1,2,\ldots,\sqrt{n}$. נבדוק אם $n$ מתחלק ב-$i$. אם כן, נחזיר 0. אם ס"מ לעבור על כולם נחזיר 1.

**זמן ריצה:** עוברים על סדר גודל של $\sqrt{n}$ מספרים, לכל מספר מבצעים סס קבוע של פעולות, לכן זמן הריצה הוא:

$$O(\sqrt{n}) = O(n^{\frac{1}{2}}) = O(2^{\log(n^{\frac{1}{2}})}) = O(2^{\frac{1}{2}\log(n)}) \rightarrow$$

קיבלנו שסיבוכיות אקספוננציאלית בגודל הקלט!

* לכן במקרה כזה, לא הגענו לסדר גודל פולינומי, מכיוון שגודל הקלט הוא לא $n$ כמו במערך/רשימה.

**הגדרה:** אלגוריתמים שרצים בסיבוכיות פולינומית בגודל האבסולוטי של הקלט, ולא באורכו (הייצוג), נקראים **אלגוריתמים פסאודו פולינומיאליים** (פולינומיאליים במובן החלש).

## דוגמה - מציאת האיבר הכפול במערך

**קלט:** רשימה $A$ באורך $n+1$ המכילה את האיברים $1,\dots,n$ פעם אחת, מלבד חזרה אחת.

**פלט:** מספר $K \in [n]$ המופיע פעמיים ברשימה $A$.

**אלגוריתם:**
1. נגדיר מערך $B$ בגודל $n$.
2. לכל $i=0,\dots,n$:
3. אם $B[A[i]-1]==0$:
4. עדכן $B[A[i]-1]=1$
5. אחרת:
6. החזר את $A[i]$

**הוכחת נכונות:**

צריך להראות שהאלגוריתם מחזיר מספר $K$ אמ"מ $K$ מופיע פעמיים ב-$A$.

($\Leftarrow$): אם האלג' החזיר את $K$, קיימים 2 אינדקסים $1 \le i \neq j \le n$ כך

שמתקיים $A[i] = A[j] = K$.

($\Leftarrow$): נניח כי $K$ מופיע פעמיים במערך. מכאן שקיימים 2 אינדקסים $1 \le i \neq j \le n$ כך שמתקיים $A[i] = A[j] = K$.
מכיוון שהאלגוריתם עובר על כל זוג, לכן בהכרח יעבור גם על הזוג הזה ויחזיר את $K$ כנדרש.

**זמן הריצה:**

המקרה הגרוע ביותר עבור אלגוריתם זה הוא כאשר המספר הכפול מופיע ב-$A[n]$.
במקרה זה, האלג' מבצע $O(n)$ פעולות.

**סיבוכיות מקום:**

נשים לב שהגדרנו מערך $B$ בגודל $n$, בנוסף למערך $A$ של הקלט ולכן סיבוכיות המקום תהיה $O(n)$.

**תרגיל:** מצאו אלגוריתם לפתרון הבעיה שרץ בזמן $O(n)$ ובסיבוכיות מקום $O(1)$.

יום שלישי 6.4.21

מקרא:
דוגמה - ○
הוכחה - ○
טענה - ○
משפט - ○
הגדרה - ○

# אלגוריתמים
## שבוע 2 - תרגול 2

**הגדרה:** אלגוריתם ייקרא חמדן אם בכל שלב הוא בוחר באפשרות הנראית המשתלמת ביותר, מבלי להתחשב בהשלכות לטווח הרחוק.

**סכימה כללית להוכחת אופטימליות אלג' חמדן:**

1) מוכיחים חוקיות של הפתרון החמדן B.

2) טענה באינדוקציה, שלכל $1\le k\le m$ ישנו פתרון אופטימלי C שמסכים עם B על k הצעדים הראשונים.

**בסיס:** $k=0$, אין מה להוכיח.

**צעד:** מה"א עבור $k\le m$ יש פתרון $C=\{b_0,\dots,b_{k-1},b_k\}$.
נבנה בעזרת C פתרון $C'$ שמכיל גם את $b_k$. נראה חוקיות של $C'$ ונראה אופטימליות בהתבסס על דרך הפעולה החמדנית של B.

**מסקנה:** עבור $k=m$ קיים פתרון אופטימלי זהה ל-B.

**בעית תשו הדלק הקטן:**

**קלט:** $N\in\mathbb{N}$ מס' הק"מ שניתן לנסוע עם מיכל מלא.
$a_1,\dots,a_n\in\mathbb{N}$ - מיקום של תחנות הדלק במסלול.

**הנחה:** $a_{i-1}-a_i\le N$ לכל $1\le i\le n$

**פלט:** תת סדרה $b_1,\dots,b_n$ המקיימת $b_1=a_1$, $b_m=a_m$ ו-$b_{i+1}-b_i\le N$ לכל $1\le i\le m$.

## Small Gas Tanks $((a_1,\dots,a_n), N)$

פתרון:

$b_1=a_1$ $,prev=1$

for $i=2$ to $(n-1)$:

  if $a_{i+1}-b_{prev}>N$:

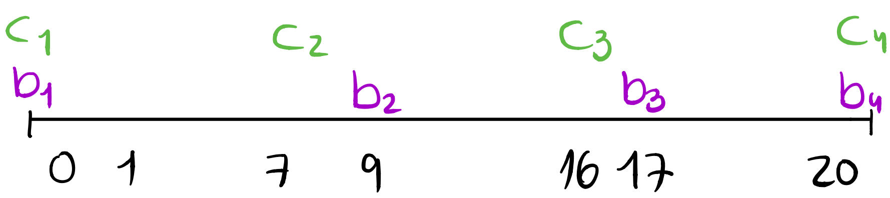
    $b_{prev+1}=a_i$

    $prev=prev+1$

$b_{prev+1}=a_n$

return $(b_1,\dots,b_{prev+1})$

קלט לדוגמה: $(0,1,7,9,16,17,20)$ $N=10$




### הוכחת נכונות:

חוקיות - מדרך פעולת האלג' $b_1=a_1, b_m=a_m$.

צ"ל שלכל $1\le i<m$ מתקיים: $b_{i+1}-b_i\le N$. נניח בשלילה שעבור $1\le j<m$ כלשהו מתקיים $b_{j+1}-b_j>N$. מההנחה על הקלט קיימת לפחות תחנה אחת בין $b_j$ ל-$b_{j+1}$. נסמן ב- $a_k$ את התחנה המקסימלית בקטע $(b_j,b_j+N)$. כאשר $i=k$ מתקיים $b_{prev}=b_j$, ומקסימליות $a_k$ בקטע, האלג' היה צריך לבחור $b_{j+1}=a_k$ בסתירה לכך ש- $b_{j+1}-b_j>N$.

אופטימליות - (למת ההחלפה)

טענה: לכל $1\le k\le m$ קיים פתרון אופטימלי מהצורה: $C=(b_1,\dots,b_k,c_{k+1},\dots,c_m)$

**הוכחה:** באינדוקציה על K.

**בסיס:** $K=1$. יהי C פתרון אופטימלי. מחוקיות C נובע $c_1=a_1=b_1$

**הנחה:** נניח כי הטענה נכונה עבור $1\le K<m$, כלומר קיים
$C=(b_1,\dots,b_K,c_{K+1},\dots,c_m)$

**צעד:** נראה עבור $K+1$. נתבונן בפתרון $C'=(b_1,\dots,b_{K+1},c_{K+2},\dots,c_m)$

נשים לב ש-$|C'|=|C|$ ולכן מספיק להראות חוקיות של $c'$.

מחוקיות C, מתקיים $c_{i+1}-c_i\le N$ לכל $K+1\le i<m$.

מחוקיות B, מתקיים $b_{i+1}-b_i\le N$ לכל $1\le i<K+1$.

נותר להראות $c_{K+2}-b_{K+1}\le N$.

מהנחת האינדוקציה, אנו יודעים ש- $c_{K+1}-b_K\le N$.

לכן מדרך פעולת האלגוריתם, $c_{K+1}\le b_{K+1}$.

ומכאן: $c_{K+2}-b_{K+1}\le c_{K+2}-c_{K+1}\le N$

חוקיות C (הנ"ל).

## עצים פורשים מינימליים

**הגדרה:** עץ הוא גרף קשיר חסר מעגלים.

- בין שני קודקודים בגרף יש בדיוק מסלול 1.
- הוספת צלע לעץ T, סוגרת מעגל 1.

**הגדרה:** יהי $G=(V,E)$ גרף קשיר לא מכוון אזי עץ פורש T של G, הוא תת גרף של $G'$ שהוא עץ, ומכיל את כל קודקודי G.

**הגדרה:** תהי $w: E \to \mathbb{R}$ פונקציית משקל של $G$, המשקל של עץ פורש $T=(V,E_T)$ של $G$ הוא: $w(T)=\sum_{e\in T} w(e)$

**הגדרה:** עץ פורש מינימלי (MST) הוא עץ פורש $T$ בעל משקל מינימלי, כלומר כזה שעבור כל עץ פורש $T'$ מתקיים: $w(T) \le w(T')$

**הגדרה:** יהי $G=(V,E)$ גרף לא מכוון ויהיו $A,B$ קבוצות קודקודים המקיימות:

$A\cup B=V$ ו- $A\cap B=\emptyset$.

נגדיר את החתך $(A,B)$ להיות קבוצת הצלעות המקיימת:

$$(A,B)=\{(u,v)\in E \mid u\in A,\ v\in B\}$$

כל הצלעות שמחברות בין הקבוצות A ו-B

## בעיית MST

**קלט:** גרף קשיר לא מכוון $G=(V,E)$ ופונקציית משקל $w:E\to\mathbb{R}$

**פלט:** MST עבור $G$.

## הכלל הכחול והכלל האדום

בהינתן גרף קשיר לא מכוון $G=(V,E)$ ופונקציית משקל $w:E\to\mathbb{R}$.

נגדיר את כלל הצביעה הבאה:

**הכלל הכחול:** בוחרים חתך $F\subseteq E$ שאין בו אף קשת כחולה וצובעים $e\in F$ בעלת משקל מינימלי בכחול (הקלה ביותר).

**הכלל האדום:** בוחרים מעגל פשוט $C$ שאין בו אף קשת אדומה וצובעים $e\in C$ בעלת משקל מקסימלי באדום (היקרה ביותר).

**משפט:** יהי גרף קשיר לא מכוון $G=(V,E)$ שחלק מצלעותיו צבועות אדום וחלק צבועות כחול באופן שרירותי. אם לא כל הצלעות צבועות, קיימת צלע לא צבועה שניתן לצבוע בעזרת הכלל הכחול או הכלל האדום.

**הוכחה:** נסתכל על הגרף $G'=(V,E')$ שנקבל מהורדת כל הצלעות האדומות מגרף $G$. אם יש בגרף $G'$ מעגל שמכיל צלע לא צבועה, הרי שאין בו אף צלע אדומה, והוא קיים ב-$G$. כלומר, ניתן להפעיל את הכלל האדום על המעגל הנ"ל, ולקבל את הנדרש. כעת, נניח שאין מעגלים עם צלע לא צבועה ב-$G'$. נסתכל על צלע לא צבועה כלשהי $e=(v,u)$.
אם $e$ לא חלק ממעגל, אז כל המסלולים בין $u$ ל-$v$ ב-$G'$ עוברים דרך $e$ (אחרת, היה מעגל). הסרת $e$ מ-$G'$ משאירה את $v,u$ בשני רכיבי קשירות שונים. נסמנם $U,V$. נגדיר חתך של $G$ באופן הבא: $A=V$, כל השאר $=B$. $e$ הצלע היחידה שאינה צבועה בחתך, אם קיימות צלעות נוספות הן לא היו ב-$G'$ ולכן הן אדומות. אם כך, ניתן להפעיל את הכלל הכחול של החתך וסיימנו. □

**מתכון לאלגוריתם חמדן לצביעת ה-MST**

1. נאתחל $E_B=\emptyset$ , $E_R=\emptyset$
2. כל עוד יש צלעות לא צבועות בגרף, נבחר בין:
   א. הפעלת הכלל הכחול ולהוסיף את הצלע הנבחרת ל-$E_B$
   ב. הפעלת הכלל האדום ולהוסיף את הצלע הנבחרת ל-$E_R$
3. נחזיר את $G=(V,E_B)$.

**טענה:** כל אלג' שמתקבל מהמתכון הנ"ל מחזיר MST של $G$.

## דוגמה 1 - האלגוריתם של פרים

1. נתחיל עם $E_T=\emptyset$ ו- $V_T=\emptyset$.
2. נבחר קודקוד אקראי v ונוסיף ל- $V_T$.
3. נפעיל את הכלל הכחול על החתך $(V_T, V/V_T)$ ונמצא את הקשת הנבחרת $e=(v,u)\in E$ בכחול.
4. נוסיף את v ו-u ל- $V_T$ ואת $e=(v,u)$ ל- $E_T$.
5. נחזור לשלב 3 עד ש- $V_T=V$.

**זמן ריצה:** פתרון נאיבי - בכל איטרציה, נעבור על כל הצלעות ונבחר את המינימלית שגם בחתך וזמן $O(|V|\cdot|E|)$.

**תזכורת:** תור קדימויות הוא מבנה נתונים אבסטרקטי המאפשר את פעולות:

* אתחול $O(n\log n)$
* שליפת מינימום $O(\log n)$
* הורדת ערך לאיבר $O(\log n)$

**פתרון טוב יותר:** נשמור בתור קדימויות עבור כל קודקוד v, מחיר $C(v)$ המייצג את המשקל של הוספת v ל- $V_T$. בהתחלה $C(v)=\infty$ לכל $v\in V$. בכל איטרציה, נוסיף את הקודקוד עם משקל החיבור המינימלי ונעדכן לכל שכניו, u, את המחיר להיות המינימלי מבין $C(u)$, $w(u,v)$.

**זמן ריצה חדש:** $O(\underbrace{v\log v}_{\text{מיון התחלתי}} + \underbrace{v\log v}_{\text{הרצת האלג'}}) = O(v\log v)$

**נכונות:** בכל איטרציה של האלגוריתם, הפעלנו תמיד את הכלל הכחול.
לכן מנכונות המתכון גם האלגוריתם הנ"ל מחזיר פתרון אופטימלי!

## דוגמה 2 - האלגוריתם של קרוסקל

1. נתחיל עם $E_T=\emptyset$ ו- $V_T=\emptyset$.
2. נמיין את הצלעות בסדר עולה $w(e_1)\le w(e_2)\le\ldots\le w(e_{|E|})$
3. עבור על הצלעות לפי הסדר:

א. אם הקשת $e_i=(u,v)$ סוגרת מעגל עם הצלעות ב-$E_T$.
נפעל על מעגל זה את הכלל האדום. ונצבע את $e_i$ באדום.
ב. אחרת, $e_i$ הצלע הקלה בחתך $(V_T\cup\{u\}, V/V_T\cup\{u\})$
לכן נצבע אותה בכחול ונוסיף את $e_i$ ל-$E_T$ ואת $v\in V_T$.

4. נחזיר את $T=(V_T,E_T)$.

**זמן ריצה:** מיון - $O(|E|\log|E|)$
מציאת מעגל - עם Union-Find ב- $O(|E|\alpha(V))$ כאשר $\alpha$
היא הופכית לפונקצית אקרמן.
לכן זמן הריצה הכולל: $O(|E|\log|E|)$

**Algorithm 3** *Prim*$(G=(V,E,w), root)$

**for** $v\in V$ **do**
  $key(v)\leftarrow\infty, \pi(v)\leftarrow$ null
$key(root)\leftarrow 0$
$Q = BuildMinHeap(V)$ according to $key$.
**while** $Q$ is not empty **do**
  $u\leftarrow ExtractMin(Q)$
  **for** every $v$ that is a neighbor of $u$ **do**
    **if** $v\in Q$ and $w(u,v)<key(v)$ **then**
      $\pi(v)=u$
      $DecreaseKey(v,w(u,v))$
**return** $(V,E')$

**Algorithm 2** *Kruskal*$(G=(V,E,w))$

**for** $v\in V$ **do**
  MakeSet$(v)$
sort $E$ into $e_1,e_1\ldots e_m$ according to their weight.
$E'\leftarrow\emptyset$
**for** $i=1,\ldots m$ consider $e_i=(u_i,v_i)$ **do**
  **if** FindSet$(u_i)\neq$ FindSet$(v_i)$ **then**
    $E'\leftarrow E'\cup\{e_i\}$
    Union$(u_i,v_i)$
**return** $(V,E')$

16.3.21 יום שלישי

מקרא:
הוכחה - ● דוגמה - ●
משפט - ● טענה - ●
הגדרה - ●

# אלגוריתמים
## שבוע 3 - תרגול 3

## אלגוריתמים חמדניים

### מטרואידים

הגדרה: מטרואיד הוא זוג $M=(E,I)$ כאשר $E$ קבוצה סופית כלשהי ו- $I\subseteq 2^{E}$ היא קבוצה לא ריקה של תתי קבוצות של $E$ המקיימת שתי תכונות:

1. ירושה: אם $A\in I$ ו- $B\subseteq A$ אז $B\in I$.
2. הרחבה: אם $A,B\in I$ וגם $|B|<|A|$ אז קיים $a\in A/B$ כך ש: $B\cup\{a\}\in I$.

הערה: $I$ קבוצה אינה ריקה, לכן $\emptyset\in I$ שכן לפי תכונת הירושה כל קבוצה חלקית שלאיבר ב-$I$ שייכת ל-$I$ ($\emptyset$ מוכלת בכל קבוצה).

### דוגמאות למטרואידים/לא מטרואידים

דוגמה 1: $M_1=(E_1,I_1)$ כאשר $E_1=\{1,2\}$ ו- $I_1=\{\{1\},\{2\},\{1,2\}\}$.

נשים לב ש- $M_1$ אינו מטרואיד מכיוון ש- $\{1\}\in I_1$ ו- $\emptyset\subseteq\{1\}$ אבל $\emptyset\notin I_1$. כלומר, תכונת הירושה אינה מתקיימת.

שמעתות כזאת $M_1=(E_1,I_1\cup\{\emptyset\})$ הוא מטרואיד.

דוגמה 2: $M_2=(E_2,I_2)$ כאשר $E_2=\{1,2,3,4\}$ ו- $I_2=\{\emptyset,\{1\},\{2\},\{3\},\{4\},\{1,2\},\{3,4\}\}$. $M_2$ אינו מטרואיד.

תכונת הירושה מתקיימת, אך תכונת ההרחבה אינה מתקיימת.

לראות מכיוון שעבור $B=\{3\}$ ו-$A=\{1,2\}$. מתקיים $|A|>|B|$, אבל
לכל $a\in A/B$ מתקיים $B\cup\{a\}\notin I_2$.
אם נגדיר $I_2'=I_2\cup\{1,4\}\cup\{2,3\}$ אז $M_2'=(E_2,I_2')$ הוא מטרואיד.

<u>דוגמה 3- המטרואיד הוקטורי</u> $M_V=(E_V,I_V)$ כאשר $E_V$ קבוצת וקטורים במרחב וקטורי כלשהו ו-$I_V$ קבוצת של וקטורים בת"ל מתוך $E_V$.

<u>דוגמה 4- המטרואיד הגרפי</u> יהי $G=(V,E)$ גרף ו-$M_G=(E_G,I_G)$ כאשר $E_G=E$ קבוצת הצלעות ו-$I_G=\{A\subseteq E \mid G_A=(V,A) \text{ יער}\}$.
נשים לב תחילה ש-$\emptyset\in I_G$ כי גרף ריק הוא חסר מעגלים ולכן לא ריקה.

1. <u>ירושה:</u> אם $A\in I_G$ הוא יער אז $G_A$ הוא יער. אם $B\subseteq A$ תת קבוצת צלעות מ-$A$ אז $B$ גם הוא יער, כי $G_B$ אינו מכיל מעגלים.
2. <u>החלפה:</u> יהיו $A,B\in I_G$ וגם $|A|>|B|$, נרצה להראות שיש $a\in A/B$ כך ש-$B\cup\{a\}\in I_G$. נגדיר $G_A=(V,A)$ וגם $G_B=(V,B)$ ונבחין כי שניהם חסרי מעגלים. נשתמש בטענת העזר הבאה:

> <u>טענת עזר:</u> הוספת צלע לגרף מקטינה את מס' רכיבי הקשירות ב-1 או סוגרת מעגל.

מכיוון שבגרף $G_A$ אין מעגלים, מספר רכיבי הקשירות בו קטן ממש מאשר שב-$G_B$, נטען שקיים רכיב קשירות ב-$G_A$ שנחתך עם שני רכיבי הקשירות של $G_B$. הדבר מתקיים, כי אחרת היינו מקבלים כי מס' רכיבי הקשירות ב-$G_A$ היה גדול/שווה לזה שב-$G_B$ (בסתירה ל-$|A|>|B|$ ולטענה).

לכן אפשר לבחור את הצלע שמחברת בין רכיבי הקשירות ב-$G_B$, נסמנה $a$.
אם כך, $G_B' = (V, B\cup\{a\})$ אינה סגורת מעגל ולכן $B\cup\{a\}\in I_G$.


## האלגוריתם החמדן של המטרואיד

<u>קלט:</u> מטרואיד $M=(E,I)$ ופונקציית משקל חיובית ממש $w: E\to\mathbb{R}^+$.

<u>פלט:</u> תת קבוצה $A\in I$ שגודלה מקסימלי, כלומר שלכל $B\in I$ מתקיים $|B|\le|A|$ וגם שמשקלה מקסימלי, כלומר, לכל $B\in I$ מתקיים:

$$w(A)=\sum_{a\in A} w(a) \ge \sum_{b\in B} w(b) = w(B)$$

קבוצה במשקל מקסימלי = קבוצה בגודל מקסימלי!

<u>הערה:</u> הפלט לבעיה שהגדרנו הינו תת קבוצה שמשקלה מקסימלי.
במונחים של חוקיות ואופטימליות, פתרון חוקי יהיה קבוצה כלשהי מ-$I$ ופתרון אופטימלי יהיה להחזיר מבין כל הקבוצות $A\in I$ את $A$ עם המשקל המקסימלי.
אם כך, ניתן להסתכל על $I$ כקבוצת הפתרונות החוקיים לבעיה.

## האלגוריתם הגנרי:

1. נמיין את איברי $E$ בסדר יורד כלומר $w(e_1) \geq w(e_2) \geq ... \geq w(e_n)$.
2. נתחיל $A = \emptyset$ ונעבור על איברי $E$ לפי הסדר ולכל $i \in [n]$:
   אם $A \cup \{e_i\} \in I$ נוסיף את $e_i$ ל-$A$.
3. לאחר מעבר על האיברים, נחזיר את $A$.

## זמן ריצה:

1. מיון לוקח $O(n\log n)$
2. לכל $i \in [n]$ צריך לבדוק האם $A \cup \{e_i\} \in I$.
   זמן ריצה כה תלוי בבעיה הספציפית לכן נסמן $O(f(n))$.
   לכן בסה"כ נקבל $O(n\log n + n \cdot f(n))$.

## הערות:

1. במידה והצלחנו להראות שבעיית אופטימיזציה מסוימת היא מטרואיד, ניתן להשתמש באלגוריתם כה ללא הוכחת נכונות נוספת.
   * יש לציין שפתרון כה אינו בהכרח אופטימלי!
2. נשים לב כי האלגוריתם החמדן ייתן תמיד פתרון מגודל מקסימלי.
3. נשים לב שמציאת מינימום היא בעיה שקולה. כל שעלינו לעשות הוא להפוך את המיון לסדר עולה או להחליף את הסימן של $w$.
   במצב כה, כבר לא נגדיר את הבעיה כבעיית משקל מינימלי, אלא מציאת קבוצה מגודל מקסימלי בעלת משקל מינימלי.

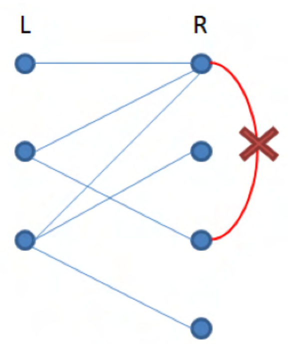
4. לא תמיד חייבים לעבור על כל $i \in [n]$, לפעמים ניתן יהיה לעצור לפני:

## דוגמה 5 - קרוסקל

כאשר נפעיל את האלגוריתם הגנרי (עם מיון בסדר לא יורד) על המטרואיד הגרפי, מקבלים יער מגודל מקסימלי (עץ פורש) ומשקל מינימלי, כלומר MST. וזהו בדיוק אלגוריתם קרוסקל.

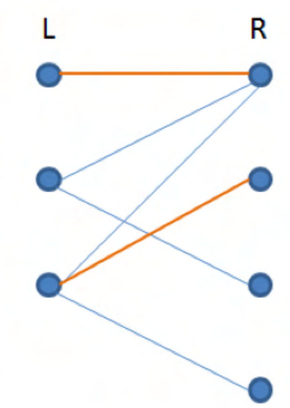
## דוגמה 6 - מטרואיד השידוכים

### תורת הגרפים - תזכורת:



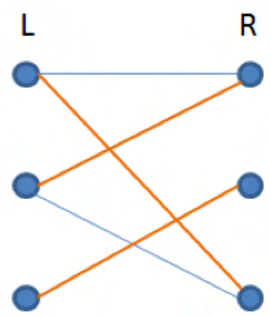
גרף דו-צדדי

1. **גרף דו-צדדי** $G=(L,R,E_G)$ הוא גרף המוגדר עם 2 קבוצות זרות של קודקודים, וכל הצלעות שלו מחברות בין קודקודים מקבוצות שונות.



התאמה

2. **התאמה** בגרף דו צדדי היא תת-קבוצה של $E_G$ שאין בה 2 צלעות שנוגעות באותו קודקוד.



התאמה מושלמת

3. **התאמה מושלמת** היא התאמה שנוגעת בכל הקודקודים.

**הערה:** בגרף בו $|L| \neq |R|$ לא תתכן התאמה מושלמת.

יהי $G=(L,R,E_G)$ גרף דו-צדדי. נגדיר $E=L$ וגם:
$I=\{L'\subseteq L \mid \exists R'\subseteq R, \exists \pi: L'\to R' \text{ s.t } \pi=\text{perfect match}\}$

**טענה:** $M=(E_M,I_M)$ הוא מטרואיד.

**הוכחה:**

נראה ש-$M$ האוכן מטרואיד.

1. $I_M$ <u>קבוצה לא ריקה</u> כיוון ש-$\emptyset$ היא התאמה מושלמת באופן ריק ולכן $\emptyset\in I_M$.

2. <u>ירושה:</u> תהי $A\in I_M$ ותהי $B\subseteq A$. (נרצה להוכיח ש-$B\in I_M$).
קיימת התאמה מושלמת $M_A: A\to R_A\subset R$ (כי $A\in I_M$).
נגדיר את $M_B: B\to R_B\subset R$ כך $M_B(b)=M_A(b)$ לכל $b\in B$.
מהגדרת $M_A$, זוהי התאמה מושלמת מ-$B$ ל-$R$ ולכן $B\in I_M$.

3. <u>הרחבה:</u> יהיו $A,B\in I_M$ כך ש-$|B|<|A|$. נרצה להוכיח שקיים $a\in A/B$ כך שקיימת התאמה מושלמת מ-$B\cup\{a\}$ ל-$R$.
מ-2. הראנו כי קיימות $M_A$ ו-$M_B$ התאמות מושלמות ל-$R_A$ ו-$R_B$.
נגדיר את הגרף הבא: $G'=(L,R,M_A\cup M_B)$ וננתח את רכיבי הקשירות:

| |
|---|
| <u>טענה:</u> מגרף $G'$ ישנם 2 סוגים של רכיבי קשירות:<br>1. מעגל פשוט ובו דרגת כל קודקוד היא 2.<br>2. מסלול פשוט, ובו דרגת כל קודקוד פנימי היא 2 וכל קודקוד חיצוני הוא מדרגה 1 (קצוות המסלול). |

דרגת כל קודקוד חסומה מלמעלה ע"י 2.

נבחין כי בכל רכיב קשירות של $G'$, הצלעות מופיעות מ-$M_A$ ו-$M_B$ לסירוגין (מדובר בהתאמות).

נזכר כי $|A|>|B|$ ולכן $|M_A|>|M_B|$.

נסיק מכך שקיים רכיב קשירות כלשהו שבו יש יותר צלעות מ-$M_A$ (נסמנם $C_A$) מאשר מ-$M_B$ (נסמנם $C_B$). בפרט, זהו מסלול פשוט מאורך אי-זוגי (חייב להיות כזה, כי $|M_A|>|M_B|$).

נסמן את הצלעות ברכיב זה ב-$C=C_A\cup C_B$ (מופיע בציור להמחשה)

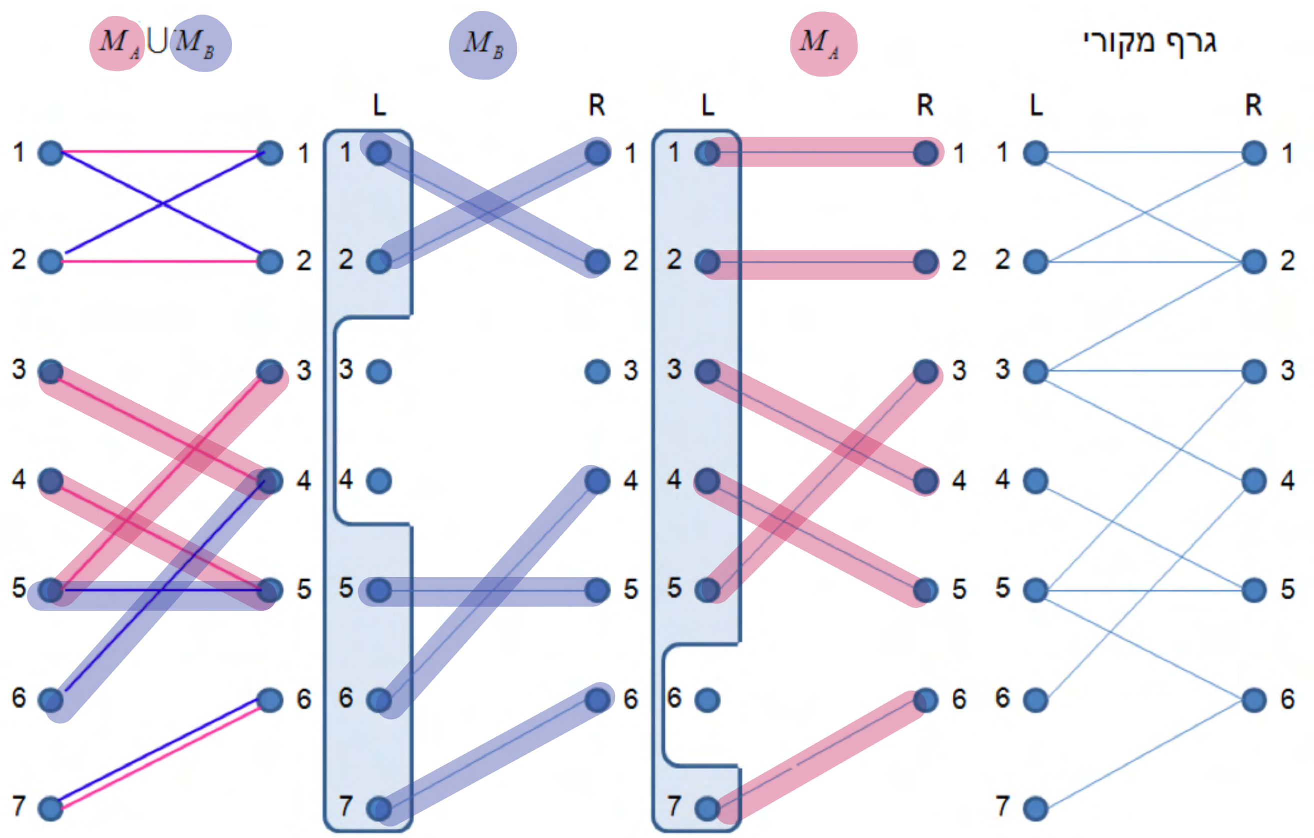
כיוון ש-$|C_A|=|C_B|+1$ (תרגיל) נובע שהמסלול מתחיל ומסתיים בצלעות מ-$C_A$, ומכיוון שהוא מאורך אי-זוגי, אחד מקצותיו ב-$L$.

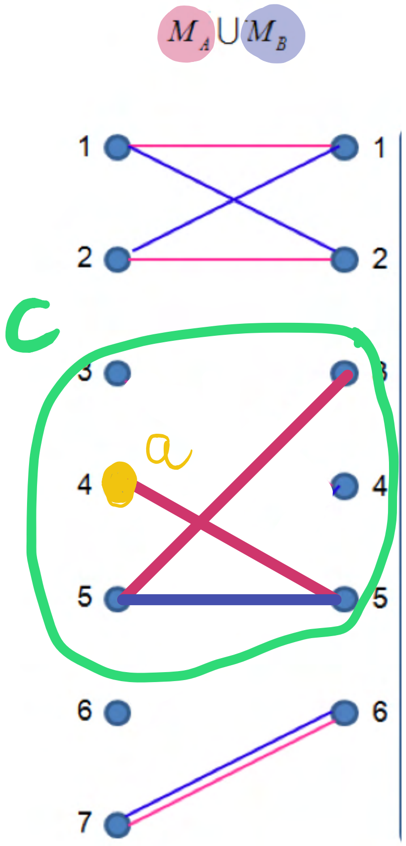
נסמן את הקצה הזה ב-$a$ ונשים לב כי $a\in A/B$.

נגדיר את השידוך עבור איברי $B\cup\{a\}$ באופן הבא: נשאיר את את כל הצלעות של $M_B/C_B$ כמו שהן, נוריד את הצלעות ב-$C_B$ ונחליף ב-$C_A$.

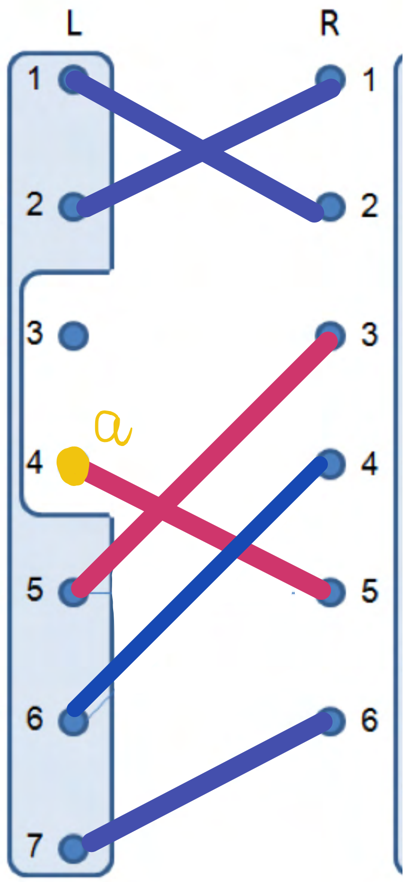
ובאופן פורמלי: $M_{B\cup\{a\}}=(M_B/C_B)\cup C_A$ וזוהי אכן התאמה משלמת

לכל קודקוד $b\notin C$ לא שינינו את ההתאמה. בנוסף, ההתאמה כוללת כל $b\in C$ יחד עם $a\notin B$ ולכן $B\cup\{a\}\in I_M$.






$M_{B\cup\{a\}} = (M_B / C_B) \cup C_A$




הערה: על מנת להשתמש באלגוריתם הגנרי למטרואיד כדי למצוא שידוך מקסימלי, והיה עלינו לבדוק בכל שלב האם $A\cup\{e_i\}\in I_M$ עבור קבוצת הקודקודים הנוכחית $A$ ו-$e_i$ הצלע ה-$i$ במיון. כלומר, האם קיימת התאמה משלמת עבור $A\cup\{e_i\}$.

באופן נאיבי, בדיקה כזו יקרה (מעבר על כל תתי הקבוצות $S\subseteq A\cup\{e_i\}$ ובדיקה האם $|S|<N_e(S)$ (משפט הול)).

מהמשפט הקודם נראה דרך יעילה למציאת שידוך מקסימלי.

20.3.21 יום שלישי

מקרא:
דוגמה - הוכחה -
טענה - משפט -
הגדרה -

# אלגוריתמים
## שבוע 4 - תרגול 4

### תכנון דינאמי

נשתמש בזיכרון כדי לחסוך זמן ריצה עבור פונקציות רקורסיביות.

**דוגמה** - מציאת האיבר ה-n בסדרת פיבונאצ'י

תזכורת: $fib(0)=0$ , $fib(1)=1$

$$fib(n)=fib(n-1)+fib(n-2)$$

**הפתרון הנאיבי:**

```
def fib(n):
  if n==0:
     return 0
  if n==1:
     return 1
  else fib(n-1)+fib(n-2)
```

גרוע משום שסיבוכיות זמן הריצה היא $O(2^n)$, אנחנו עושים עבודה כפולה הרבה פעמים.

הזיכרון יהיה $O(2^n)$.

**נראה כיצד ניתן לשפר את זה ע"י תכנון דינאמי:**

(ננסה להשתמש בזיכרון כדי לחסוך בזמן ריצה)

לא הדרך היעילה ביותר

```
def fib(n):
   history = [0,1]
   for i=2 to n:
      history[i] = history[i-1] + history[i-2]
   return history[n]
```

זמן ריצה: $O(n)$

זיכרון: $O(n)$

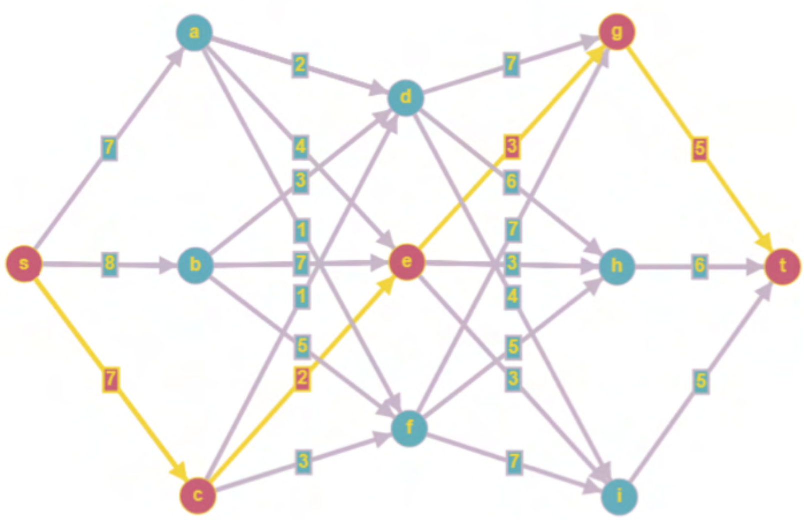
ניתן לחסוך - $O(1)$

## מתכון לפתרון בעיות של תכנות דינאמי

1. **הגדרת תתי הבעיות:** נגדיר מתמטית את תתי הבעיות שהאלגוריתם יחשב.
2. **ניסוח נוסחת רקורסיה:** נרשום מתמטית כיצד לחשב תת בעיה מתתי בעיות קודמות.
3. **הגדרת טבלה שתשמור את הפתרונות עד כה + סדר מילוי:** נגדיר את טבלת ההיסטוריה בה נשמור את תתי הבעיות ונסביר באיזה סדר אפשר למלא אותה.
   כאן נגדיר את לולאות ה-for
4. **חילוץ פתרון:** בהינתן טבלת ההיסטוריה המלאה, כיצד נחלץ פתרון.
5. **זמן ריצה:** בד"כ: $\boxed{\text{זמן מילוי תא} \times \text{גודל הטבלה}}$
6. **הוכחת נכונות:** מה נרצה להוכיח?
   1. נוכיח כי התאים בהיסטוריה מכילים את פתרון תתי הבעיות.
   2. חילוץ הפתרון היה נכון.

את ההוכחה נוכיח באינדוקציה על סדר מילוי הטבלה ונעשה זאת בעזרת נוסחת הרקורסיה.

### דוגמה - בלמן-שכבות




נתון גרף מכוון $G=(V,E)$, $w: E \to \mathbb{R}$

בגרף יש $K+2$ שכבות $V_0, \dots, V_{K+1}$

השכבות הן חלוקה זרה של קודקודי $V$

$$\bigcup_{k=0}^{K+1} V_k = V \qquad \forall i \neq j \quad V_i \cap V_j = \emptyset$$

$$\forall (u,v) \in E \quad \exists\, 0 \le k \le K \quad u \in V_k, v \in V_{k+1}$$

$$V_0 = \{s\}, \quad V_{K+1} = \{t\}$$

עבוד על פי המתכון:

1. **הגדרת תתי הבעיות:** לכל קודקוד $v \in V$ נמצא את המסלול הכי קצר מ-$s$ אל $v$, ויש לנו $|V|$ תתי בעיות.

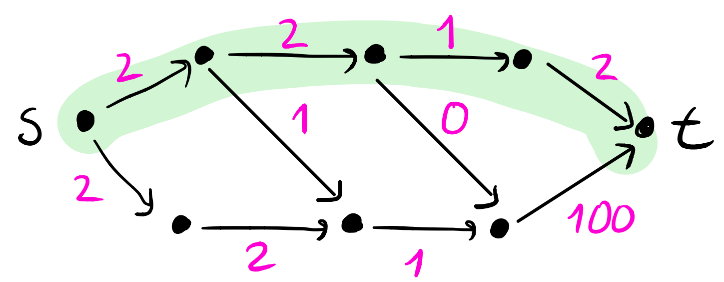
2. **נוסחת רקורסיה:**

$$\underset{\text{Distance}}{D[v]} = \begin{cases} 0 & v=s \\ \min\limits_{u \in \text{neighbors}(v)} \left(D[u]+w(u,v)\right) & \text{else} \end{cases}$$

3. **הגדרת טבלה + סדר מילוי:** נגדיר טבלה $m \times (K+2)$ ונמלא כל עמודה בנפרד (מלמעלה למטה) ולאחר מכן עמודות בסדר עולה (משמאל לימין).

+2 שכבה ל-$s$ ול-$t$

דוגמה:




| | s | 1 | 2 | 3 | t |
|---|---|---|---|---|---|
| 2 | X | 2 | 4 | 5 | X |
| 1 | 0 | 2 | 3 | 4 | 7 |

4. **חילוץ פתרון:** נשמור בטבלה גם pointer לאיפה קיבלנו מינימום ונלך בעקבות ה-pointer'ים מ-$t$.

5. **זמן ריצה:** גודל הטבלה $O(|V|)$, זמן למלא כל תא $O(m)$. $\cancel{\Leftarrow}$ $O(m|V|)$

וזמן הריצה הוא $O(|E|+|V|)$

עוברים על כל הקודקודים / עוברים על כל צלע בדיוק פעם אחת

6. **הוכחת הנכונות:** נוכיח באינדוקציה על סדר מילוי הטבלה כי הטבלה מולאה נכון:

**בסיס:** בתא התקבל לקודקוד $s$ 0 וזה באמת המרחק.

**הנחה:** כל התאים עד כה מולאו נכון

**צעד:** נוכיח כי קיים מסלול קצר. נסתכל על קודקוד $v_m^K$ שנמצא בשכבה $K$.

נסמן את הקודקודים בכל שכבה כך: $V^K=\{v_1^K,v_2^K,\ldots,v_m^K\}$, שכבה $K$

כתאו שאנחנו מסתכלים רשום: $\min_{u\in V^{k-1}}\{D[u]+w(u,V_m^k)\}$

האיבר $u'$ מקיים: $\min(D[u]+w(n,V_n^k)) = D[u']+w(u',V_m^k)$

אז מהנחת האינדוקציה קיים מסלול מ-$s$ ל-$u'$ באורך $D[u']$ ואם נמשיך

בצלע $(u',V_m^k)$ נקבל מסלול באורך הדרוש.

__אופטימליות:__ נרצה להוכיח שהמסלול הכה הכי קצר. נניח בשלילה שהוא לא.

בשכבה ה-$(k-1)$ המסלול עבר ב-$u''\neq u'$. האורך של המסלול: $D[u'']+w(u'',V_m^k)$

$\Leftarrow$ המינימום עבר על $u''$ ולא בחר בה בסתירה לפעולת המינימום.

$\square$

__הערות:__ יפעל על $G$ DAG ע"י מיון טופולוגי $O(|E|+|V|)$.

## __דוגמה - תת המחרוזת הכי ארוכה__

__קלט:__ 2 מחרוזות

__פלט:__ תת מחרוזת ארוכה, לא בהכרח רציפה

$X = ABCD$

$Y = BbC$

__פתרונות:__

__עיוני:__ נעבור על $G$ תתי המחרוזות ונשווה, __זמן:__ $O(2^n)$ :(

__דינמי:__ לפי המתכון:

1. __הגדרת תתי הבעיות:__ נסמן: $X^i = x_1,\dots,x_i$ רישא של $X$

$Y^j = y_1,\dots,y_j$ רישא של $Y$

לכל $1\le i\le n$ ולכל $1\le j\le m$ נמצא תת מחרוזת ארוכה של $X^i$ ו-$Y^j$.

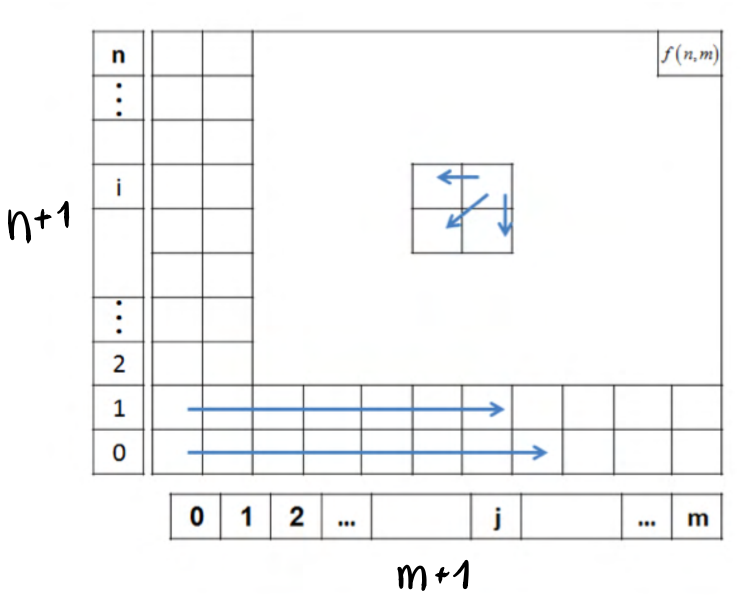
2. __ניסוח נוסחת רקורסיה:__

$$f(i,j)=\begin{cases} 0 & i=0 \vee j=0 \\ f(i-1,j-1)+1 & x_i=y_j \\ \max\{f(i-1,j),f(i,j-1)\} & x_i\neq y_j \end{cases}$$

## 3. הגדרת טבלה + סדר מילוי:

נמלא מקרה בסיס ואז משמאל
לימין ואז מלמטה למעלה.

$(n+1)\times(m+1)$




## 4. חילוץ פתרון: נשמור pointer'ים ונלך לפיהם.