



דר הנושאים והיקפם יכול להשתנות בהתאם לשיקול דעת המרצה

מפגש 1	חזרה על מונחים עיקריים בתורת הגרפים
מפגש 2	זרימה ברשותות. - flow ערך
מפגש 3	רשותות עם חסמים תחתונים, זרימה 1-0 ושידוכים - flow שער
מפגש 4	חזרה על תוכנות דינמי ואלגוריתמים מעוניינים - FFT
מפגש 5	התאמת מחוזות - KMP - $\text{O}(n^2)$ גוף-ה-ט
מפגש 6	אלגוריתמים ONLINE [ONLINE] כוונון וטוווטו.
מפגש 7	תורת הסיבוכיות, מחלקות P ו-NP, רדוקציה פולינומית
מפגש 8	CoNP
מפגש 9	משפט Cook, בעיות NP-שלמות
מפגש 10	בעיות NP-שלמות, PSpace
מפגש 11	אלגוריתמי קירוב $\left\{ \begin{array}{l} \text{הרמיג} \\ \text{היררכיה} \end{array} \right.$
מפגש 12	אלגוריתמי קירוב $\left\{ \begin{array}{l} \text{וירטואליות} \\ \text{טערם קוונר} \end{array} \right.$
מפגש 13	חזרה לҚරأت הבדיקה

נקודות: 80%, 40%
 נקודות מה: 20%. \rightarrow מבחן ליום חמישי, מבחן חמישי ויום שישי מוקדם יותר.
 מבחן סביר. \rightarrow מבחן (כנה 20%)

* הוכיח \rightarrow דיאגרמה נסורה, \rightarrow הוכיח שמי \rightarrow מוכיח, \rightarrow הוכיח נסורה \rightarrow מוכיח \rightarrow מוכיח
 * הוכיח \rightarrow מוכיח נסורה \rightarrow הוכיח נסורה \rightarrow מוכיח נסורה

* מבחן פירוי אובייקט, מבחן נסורה מבחן נסורה נסורה מבחן נסורה ...

every thing
gonna turn out
okay *



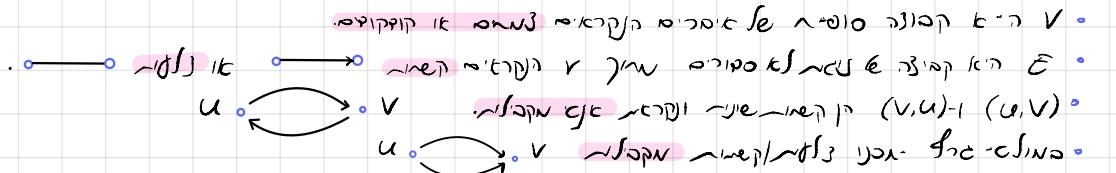
איך כריסטיאן הוכיח?

- נסמן G כgraft ו- H כgraph סטנדרטי (ולא יוניק). נסמן G' כgraft' ו- H' כgraph' סטנדרטי.
- הוכחה בדרכי שבר ונכון וריאנטה של הוכחה נורמלית (בנוסף לכך יש לנו הוכחה נוספת).
- כראוי מושג ווואר של G כפונקציית גראף ϕ_G המAPPING מ- G ל- H .
- מושג ϕ_G מושג כפונקציית גראף $\phi_{G'}$ מ- G' ל- H' .
- נסמן $\phi_G \circ \phi_{G'} = \phi_{G''}$.

לעומת

מבחן 20.0.0

$$G = (V, E) \quad \text{ולפ' } G' = (V', E')$$



(V, V) מושג מילויים נורמלים ככליפה ϕ מושג מילויים נורמלים ככליפה ϕ מושג מילויים נורמלים ככליפה ϕ

$\forall v \in V$ מושג מילויים נורמלים ככליפה $\phi(v)$

$\forall v \in V$ מושג מילויים נורמלים ככליפה $\phi(v)$

$m \leq n(n-1)$ מושג מילויים נורמלים ככליפה ϕ

$m \leq n(n-1)/2$ מושג מילויים נורמלים ככליפה ϕ

הזרם מושג מילויים נורמלים ככליפה ϕ

$\sum_{v \in V} d(v) = 2|E|$ מושג מילויים נורמלים ככליפה ϕ

$\sum_{v \in V} d_{in}(v) = \sum_{v \in V} d_{out}(v) = |E|$ מושג מילויים נורמלים ככליפה ϕ

$m = O(n^2)$ ו- $n \neq 1$, $m \leq n^2$ מושג מילויים נורמלים ככליפה ϕ

הזרם מושג מילויים נורמלים ככליפה ϕ

מושג מילויים נורמלים ככליפה ϕ - (directed acyclic graph) DAG

מושג מילויים נורמלים ככליפה ϕ

מושג מילויים נורמלים ככליפה ϕ מושג מילויים נורמלים ככליפה ϕ

מושג מילויים נורמלים ככליפה ϕ מושג מילויים נורמלים ככליפה ϕ

מושג מילויים נורמלים ככליפה ϕ

1. רוחש בדוקות גראף. אם מושג מילוי סדר. אזי ϕ מושג מילוי סדר, ורוחש גראף.

2. רוחש מושג מילוי (סיברגרף של הגרף G כפונקציית גראף).

3. מושג מילוי גראף, נזקירות זרעל דירקטוריון - (זיהוי כטבאי קובץ אפסים, דינמי מושג מילוי גראף).

BFS בערות תור

BFS(Graph G, Vertex s)**Queue Q = \emptyset**

// INIT d

for each vertex v do $d[v] \leftarrow \infty$

// Level 0

Q.Enqueue(v) $d[s] \leftarrow 0$

// Loop

while Q $\neq \emptyset$ do $u \leftarrow Q.Dequeue()$ **for each v $\in \text{Adj}[u]$ do**if $d[v] = \infty$ then $d[v] \leftarrow d[u] + 1$ **Q.Enqueue(v)**

• BFS ימוך מינימום?

• $V - 1$ עליות ניסיון?

• נסיעה קלה מרכז.

• נסיעה קשה מרכז.

הוכחה: סעיפים הרחוקים קדימה יאלאם דרכם הינה יפה.

• סעיף 1: אם v לא ישייך G .• סעיף 2: אם v שייך G .• סעיף 3: אם v לא שייך G .הוכחה: $\forall T \subseteq V$ $\omega(T) = \sum_{(u,v) \in T} \omega(u,v)$ • $\omega(T) = \sum_{(u,v) \in T} \omega(u,v)$ $\leq \sum_{(u,v) \in S} \omega(u,v)$ $\leq \sum_{(u,v) \in S} \omega(u,v) = \omega(S)$ הוכחה: $\forall T \subseteq V$ $\omega(T) = \sum_{(u,v) \in T} \omega(u,v)$

• נסיעה קשה מרכז.

הוכחה: $\forall T \subseteq V$ $\omega(T) = \sum_{(u,v) \in T} \omega(u,v)$ הוכחה: $\forall T \subseteq V$ $\omega(T) = \sum_{(u,v) \in T} \omega(u,v)$ הוכחה: $\forall T \subseteq V$ $\omega(T) = \sum_{(u,v) \in T} \omega(u,v)$ הוכחה: $\forall T \subseteq V$ $\omega(T) = \sum_{(u,v) \in T} \omega(u,v)$

• נסיעה קשה מרכז.

• נסיעה קשה מרכז.

 $F \leftarrow \emptyset$ while (V,F) is not a spanning treechoose "good edge" $e \notin F$ and let $F \leftarrow F \cup \{e\}$

צלע טובה = צלע שהוספה יוצרת קבוצה מבטיחה.

נמי הבחירה היחידה שקיים?

כל צבג ו/or נכון קני דס נסיעות, חילוק E

ולמי א ג קפלה קזקפיו הנכונה רכיב קזרה ב-H = (V, F).

תודה א עזב שבס-טנסיג כר S-V. V/F. קזרה נסיעות.

. $F \subseteq T$ - א ית פיר נסיעות, דס

. $F \cup \{e\} \subseteq T$ ית סעיף אט ו-

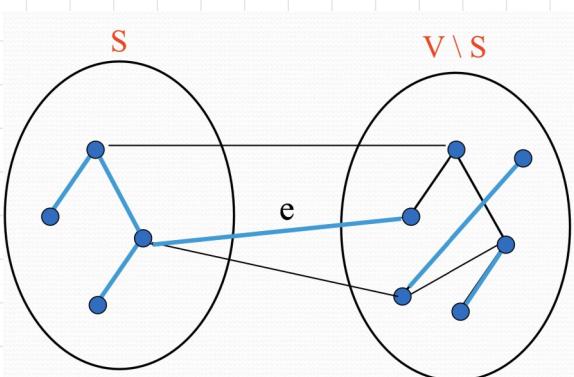
. $\text{sum } f_{ij} \in T \cup \{e\}$, מאריך

. $\text{sum } f_{ij} \in V \setminus S$ ית $e' \in T$ מאריך אט (הו, $e' \in T$ מאריך אט)

. $T' = T \cup \{e\} / \{e'\}$ מאריך אט

. $\omega(T') = \omega(T) + \omega(e) - \omega(e') \leq \omega(T)$ מאריך אט

. $V \setminus S$ ית e מאריך אט

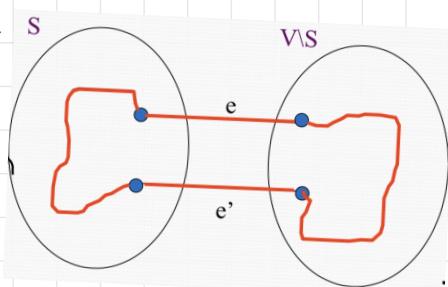


←

. $\omega(T) \leq \omega(T')$ מאריך אט T ית מאריך אט

. $\text{sum } f_{ij} \in T'$ מאריך אט, $\omega(T) = \omega(T')$ מאריך אט

. $F \cup \{e\}$ ית, $F \cup \{e\} \subseteq T'$



Union-Find

המכלול נלעט בפונקציית רקורסיבית, מאריך אט. אם קפזה זר לרשף איזה חומרה, מאריך אט.

ונכון רקורסיבי:

. כפ קפואה ווועץ זר, זר ית מאריך אט.

. (זר) קפואה ווועץ זר, זר ית מאריך אט.

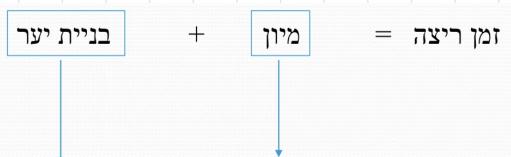
אפקט:

. נתקדש קפואה קפוא בפונקציית Find .

. כפ Find מאריך אט.

. $\text{Find} + \text{Union}$ מאריך אט, מאריך אט - הפלט כטבנ-Tarjan כטבנ. הילך אט(k, n) מאריך אט, $\Theta(kd(k, n))$ מאריך אט.

כפ הפלט כטבנ-Tarjan כטבנ כפ Find כפ Union כפ MakeSet כפ Find



n פעולות MakeSet
 $2m$ פעולות Find
 $n-1$ פעולות Union
 $O((m+n)\alpha(m+n,n)) = O(m\alpha(m,n))$

$\Theta(m \log m) = \Theta(m \log n)$

זמן הריצה נשלט ע"י שלב המיון, ומכך שהוא: $\Theta(m \log n)$

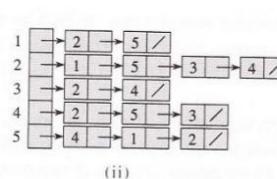
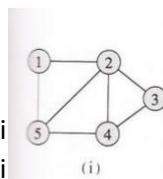
חזרה על גרפים

- (E,V,G) - תיאור של גרף בעל סט צמתים V וסט קשתות E. (לצמתים ניתן לקרוא קודקודים ולקשותות יש הקוראים צלעות).
 - |V| - מצין את מספר הצמתים
 - |E| - מצין את מספר הקשתות
- רשימת סמיכות – עדיף לציג גרפים דילילים (כלומר E קטן בהרבה מ-V²).
- מטריצת סמיכות – עדיף בגרף צפוף (כאשר E קרוב ל-V²) או כshedreshat יכולת לגלוות במהירות אם קיימת קשר המחברת שני קודקודים נתוניים.

*אותה סיבוכיות גם בזמן וגם למקומות

יצוגים של גרף לא מכון

- i. גרף לא מכון
- ii. רשימת סמיכות – סיבוכיות ($|V| + |E|$)
- iii. מטריצת סמיכות (סימטרית) סיבוכיות ($|V|^2$)

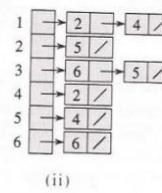
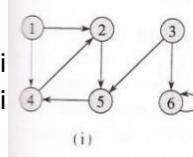


1	2	3	4	5	
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

איך 2.1 שני ייצוגים של גרף בלתי-wiecoן. (i) נור' בלתי-wiecoן G המכיל חמישה קודקודים ושבע קשותות. (ii) ייצוג של G על ידי רשימות סמיכות. (iii) הייצוג של G על ידי מטריצת סמיכות.

יצוגים של גרף מכון

- i. גרף מכון
- ii. רשימת סמיכות
- iii. מטריצת סמיכות



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

איך 2.2 שני ייצוגים של גרף מכון. (i) גרף מכון G המכיל שישה קודקודים ושמונה קשותות. (ii) ייצוג של G על ידי רשימות סמיכות. (iii) הייצוג של G על ידי מטריצת סמיכות.

סракת גרף – מעבר שיטתי על קשותות הגרף לצורך ביקור בקודקודיואלגוריתם BFS סיבוכיות ($|E| + |V|$)

חיפוש לרוחב על גרפים לא מכונים ומכונים. בהינתן גרף וקודקוד s המשמש כמקור,

חיפוש לרוחב בוחן בשיטתיות את הקשותות ב-G כדי לגלוות כל קודקוד שנitin להגעה אליו מ-s. האלגוריתם מחשב את המרחק (מספר הקשותות המינימלי) מ-s,

ובכן בונה "עץ רוחב" ששורשו הוא s, ומכליל את כל הקודקודים הללו.

עבור כל קודקוד v שנitin להגעה אליו מ-s, המסלול מ-s ל-v, בעץ הרוחב מתקבל מסלול הקצר ביותר מ-s לו ב-G.

האלגוריתם מתקדם כך שהוא מוצא את כל הקודקודים הנמצאים במרחב K מ-s לפני שהוא מגלה את הקודקודים שנמצאים במרחב K+1 מ-s.

```
BFS(G, s)
1   for each vertex u ∈ V[G] - {s}
2     do color[u] ← WHITE
3       d[u] ← ∞
4       π[u] ← NIL
5   color[s] ← GRAY
6   d[s] ← 0
7   π[s] ← NIL
8   Q ← {s}
9   while Q ≠ ∅
10    do u ← head[Q]
11      for each v ∈ Adj[u]
12        do if color[v] = WHITE
13          then color[v] ← GRAY
14            d[v] ← d[u] + 1
15            π[v] ← u
16            ENQUEUE(Q, v)
17            DEQUEUE(Q)
18            color[u] ← BLACK
```

- מדיידת מרחקים
- האלגוריתם צובע כל קודקוד לבן/אפור/שחור בהתאם לכל הקודקודים לבנים
- כל הקודקודים הסמוכים לקודקוד שחור יהיו אפורים/שחורים

הנחות

- הגרף מיוצג ע"י רשימת סמיכות.
- צבעו של כל קודקוד מיוצג ע"י color[u].color
- הקודקוד הקדום ל-u מאוחסן במשתנה [u].π.
- המרחק מהמקור s לקודקוד u מאוחסן ב[u].d.
- האלגוריתם משתמש בתור - Q לצורך ניהול קבוצת הקודקודים האפורים.

אלגוריתם DFS

חיפוש לעומק. בחיפוש זה נסרקות הקשתות היוצאות מין הקודקוד האחרון הקטgalו שעדין יש לו קשתות היוצאות ממנו וטרם נבדקו. לאחר שנבדקו כל הקשתות היוצאות מ- v , החיפוש "נסוג" וממשיר בבדיקה הקשתות היוצאות מהקודקוד שמן התגלה v . תהליך זה נמשך עד שמתגלים כל הקודקודים שניתן להגעה אליהם מקודקוד המקור המוקורי. אם נותרו קודקודים שטרם התגלו, אחד מהם נבחר כמקור חדש, והחיפוש נמשך ממקור זה. התהליך חוזר על עצמו עד שמתגלים כל הקודקודים.

- בכל פעם שמתגלה קודקוד v במהלך הסריקה על רישימת הסמוכות של קודקוד v ש- $f[v]$ שעתה מציין אירוע זה ע"י הצבת v ב- $\pi[v]$.

הקודקודים נצבעים במהלך החיפוש.

- בתחליה, כל קודקוד הוא לבן. הוא נצבע באפור כאשר הוא מתגלה, ובשחור כאשר הטיפול בו מסתיים.

DFS(G)

- ```

1 for each vertex $u \in V[G]$
2 do $color[u] \leftarrow \text{WHITE}$
3 $\pi[u] \leftarrow \text{NIL}$
4 $time \leftarrow 0$
5 for each vertex $u \in V[G]$
6 do if $color[u] = \text{WHITE}$
7 then DFS-VISIT(u)

```
- חיפוש לעומק שומר בכל קודקוד חותמת זמן.
  - ראשונה:  $[v, f]$  מצבת מתי סימן החיפוש לבחור את רישימת הסמוכות של  $v$  וצבע אותו שחור.
  - השנייה  $[v, f]$  מצבת מתי סימן החיפוש לבחור בין  $[u, f]$  לבין  $[v, f]$ . אפור בין  $[u, f]$  ושחור לאחר  $[u, f]$ .
  - קודקוד  $v$  הוא לבן לפני זמן  $[u, f]$ . אפור בין  $[u, f]$  לשחור לאחר  $[u, f]$ .
  - האלגוריתם תקף לגרפים מכונים ולא מכונים.

DFS-VISIT( $u$ )

- ```

1  $color[u] \leftarrow \text{GRAY}$            ▷ White vertex  $u$  has just been discovered.
2  $d[u] \leftarrow time \leftarrow time + 1$ 
3 for each  $v \in Adj[u]$            ▷ Explore edge  $(u, v)$ .
4   do if  $color[v] = \text{WHITE}$ 
5     then  $\pi[v] \leftarrow u$ 
6     DFS-VISIT( $v$ )
7  $color[u] \leftarrow \text{BLACK}$           ▷ Blacken  $u$ ; it is finished.
8  $f[u] \leftarrow time \leftarrow time + 1$ 

```
- חיפוש לעומק מביב מידע רב על מבנהו של הגרף.
 - מועד הגליי והסיום יוצרים מבנים סגורים.
- סוגי קשתות**
- קשת עז
 - קשת אחורית
 - קשת קדמית
 - קשת חוצה

רכיבים קשורים - Union-Find

גרף שהוא קיימ סולול מכל צומת לכל צומת. ההגדרה זהה גם בגרף מכון וגם בגרף לא מכון.
(ללא דואק קשת ישירה, אך חייב להיות מסולול).

רקי (רכיב קשור הטוב) של גראף מכון הוא קבוצה מקסימלית של קודקודים, בהם ע"ש שיכים ל- v , כך שעבור כל זוג קודקודים u ו- v -ב- u , מתקיים ש- v קשור ל- u ו- v קשור ל- v . יש רקי קשירות אחד

אם יש לנו גראף לא מכון, צריך לסרוק עם DFS או BFS. אם אחרי שהרצינו את האלגוריתם אנחנו רואים שיש צמתים או קשתות שלא הגענו אליהם, בין שהgraף לא קשור, ובכך מוצאים רכיבי קשירות. לאחר מכן נתחל מקודקוד אחר את החיפוש שנבחר, וממנו נבדוק את רכיב הקשירות הנוסף שיש לגראף הלא מכון.

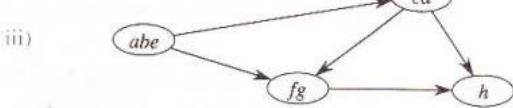
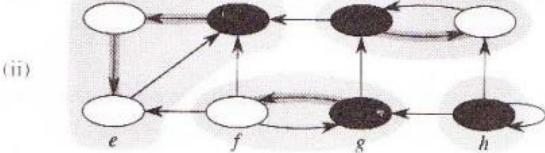
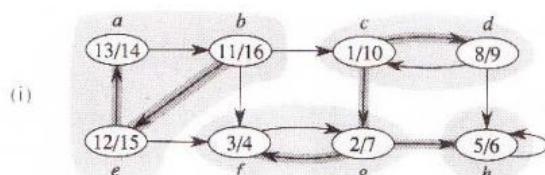
רוצים לבדוק כמה מסלולים יש מצומת לצומת.

סיבוכיות ($|E| + |V| \cdot O$)

האלגוריתם הבא, שזקן ריצתו ליניארי (זהינו, $(V+E)\Theta$), מוחשב את הרכיבים הקשורים היבט של נסיך טכון $G = (V, E)$ = באמצעות שני חיפושים לעומק, אחד על G ואחד על G^T .

STRONGLY-CONNECTED-COMPONENTS (G)

- 1 call DFS(G) to compute finishing times $f[u]$ for each vertex u
- 2 compute G^T
- 3 call DFS(G^T), but in the main loop of DFS, consider the vertices in order of decreasing $f[u]$ (as computed in line 1)
- 4 output the vertices of each tree in the depth-first forest of step 3 as a separate strongly connected component



מסלול אוילר

דרגה של צומת היא מספר הקשתות המחוורות אליו. בגרף מכון נבחין בין קשתות יוצאות לקשוטות נכנסות. גраф לא מכון סופי וקיים הוא גרפ אוילר אם ורק אם מתקיימים אחד התנאים הבאים:

1. יש בדיק 2 צמתים שדרוגם אי זוגית.
2. כל הצמתים דרגתם היא זוגית.
 - מסלול אוילר מסלול אוילר אם עוביים על כל קשת פעמי אחת.
 - מסלול יקר מסלול המיליטן אם עוביים בכל צומת פעמי אחת.

סיבוכיות ($|E| + |V|O$)

אלגוריתם Dijkstra

אלגוריתם למציאת המסלולים הקצרים ביותר ממוקור יחיד (צומת התחלתי), מתייחס למשקלים על הקשתות. מיישם רעיון דומה ל-BFS.

אלגוריתם זה הוא הרחבה של BFS – אך תקף גם לגרפים שארכיו צלעותיהם שונים מ-1. מוגבלו: האלגוריתם תקף גם ל蹶ה של קשתות עם משקלים אי שליליים.

מנהל קבוצה S של קודקודים אשר עברם כבר נקבעו סופית משקלם הקצרים ביותר מין המוקור s . לעומת עבור כל קודקוד $v \in S$ מתקיים $(v, s) = [v]$.

האלגוריתם בוחר בכל איטרציה את הקודקוד $s - v \in S$, בעל אומדן המינימלי של המסלול הקצר ביותר, מכניס את v ל- S וմבצע הקלה על כל הקשתות היוצאות מ- v . השימוש מניח שהגרף G ממושע"י רישימת סמיכות. משתמשים לרוב באלגוריתם זהה לשדרת פיבונאצ'י.

➢ בلمן פורד פוטר את הבעיה של הסיבוכיות, ($|E|^*|V|O$)

סיבוכיות ($|E| + |V| \log |V|O$)

עצים

גרף ללא מעגלים.

עץ פורש: קבוצת הקשתות בתוך גרפ ההופכת את הגרפ לעץ מבלי לפגוע בקשריו.

הגדרה חלופית:

עץ פורש לגרפ: הוא תת-גרף בצורת עץ המכיל את כל הצמתים ותת-קבוצה של הקשתות.

עץ פורש מינימום **בנייה עץ כתת-גרף של הגרפ הנטוון**, שמכיל את כל הצמתים של

קובוצת הקשתות המינימלית ההופכת את הגרפ לעץ ומשאייה אותו קשור.

משפט: בהינתן גרף קשיר G , האלגוריתם של קروسקל מוצא עץ פורש מינימלי.

הוכחה: נוכיה באינדוקציה על מספר הצלעות ב- F , שבכל שלב F מבטיחה.

משמעות זו נובעת נוכנות האלגוריתם (תרגיל!).
בסיס: $F = \emptyset$ ברור שזו קבוצה מבטיחה.

הנחה: נניח ש- F מבטיחה עד כה

מעבר: תהי $(v, u) = e$ הקשת אותה בוחר האלגוריתם להוספה.
כלומר, u, v שייכים לרכיבי קשרות שונות של העיר.

נסמן $-S$ את הרכיב שמכיל את u .
מתיחסיב ש- e היא קשת ממשקל מינימלי שייצאת מ- $-S$, כי אילו הייתה צלע נוספת קלה יותר, האלגוריתם היה נתקל בה קודם ומובחר בה)

אלגוריתם Kruskal

אלגוריתם חמדן למציאת עץ פורש מינימלי בגרף לא מכון.

האלגוריתם:

- מין הקשתות ע"פ משקליהן
- מתחילה ליטול קשתות ע"פ המשקל שלתן מהנמור לגבוי
- מוסיף כל קשת צו לחתת הגרפ כל עוד לא נוצר מעגל

הסיבוכיות של האלגוריתם נובעת מה הצורך לMIN את הקשתות
הपחמי ($|E| \log |E|O$)



אלגוריתם Prim

MST-PRIM(G, w, r)

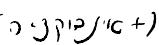
- 1 $Q \leftarrow V[G]$
- 2 **for** each $u \in Q$
- 3 **do** $key[u] \leftarrow \infty$
- 4 $key[r] \leftarrow 0$
- 5 $\pi[r] \leftarrow \text{NIL}$
- 6 **while** $Q \neq \emptyset$
- 7 **do** $u \leftarrow \text{EXTRACT-MIN}(Q)$
- 8 **for** each $v \in Adj[u]$
- 9 **do if** $v \in Q$ and $w(u, v) < key[v]$
- 10 **then** $\pi[v] \leftarrow u$
- 11 $key[v] \leftarrow w(u, v)$

אלגוריתם למציאת עץ פורש מינימלי. מיישם רעיון דומה ל-BFS.

האלגוריתם:

- מיוון הקודקודים ע"פ משקליהם
- מתחילה ליטול קודקודים ע"פ המשקל שלהם מהנמוך לגבוה ← ס. ה' נ' נ' נ'.
- מוסיף כל קודקוד צוזו לתת הגרף כל עוד לא נוצר מעגל

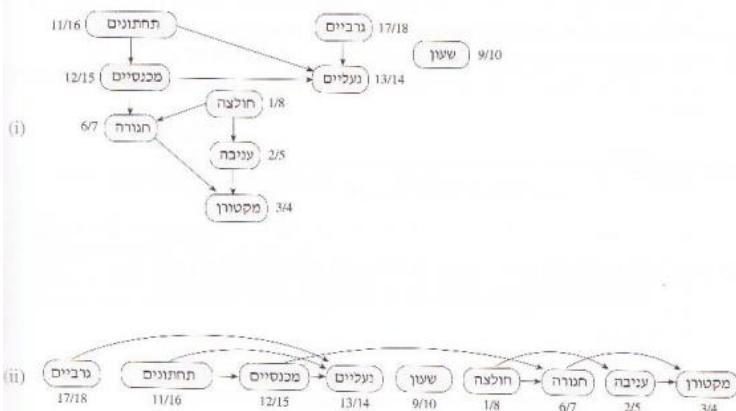
הסיבוכיות של האלגוריתם נובעת מהצורך למין את הקשתות בהתחלה והיא $(|E| + |V| \log |V|)$.

הוכחה (continued): ס. ג' המירה מהאיינט

הוכחה: ס. ג' קוקוא צאצ'ר מפוזע, ראנאי ס' הינה הינה כירא אוניברסיטת מפוזע גראן. כו' כו', רחן זיר קפ'נו'ס אוניברסיטת מפוזע נסיג'ה ע' אוניברסיטת כ'ן.

מיוון טופולוגי

שימוש בחיפוש לעומק על גרף גמ"ל – גרף מכון ללא מעגלים.
מיוון טופולוגי של גמ"ל הוא סידור לנארי של כל קודקוד G כך שם G מכיל קשת (v, u) אז מופיע v לפני u בסידור זה.
משתמש לצוין קדימות בטור קבוצת מאורעות.



האלגוריתם הפשטוט הבא ממיון גמ"ל מיוון טופולוגי.

TOPOLOGICAL-SORT(G)

- 1 call DFS(G) to compute finishing times $f[v]$ for each vertex v
- 2 as each vertex is finished, insert it onto the front of a linked list
- 3 return the linked list of vertices

KRUSKAL(WeightedGraph G)

```

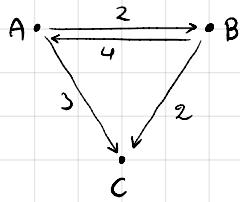
EdgeSet F  $\leftarrow \emptyset$  // Empty forest
DisjointSets UF;
List L  $\leftarrow \text{SORT}(E)$  // Sort edges by weight

for each  $v \in V$  do
    S.MakeSet( $v$ ) // Enter  $v$  into Union-Find structure

for each  $(u, v) \in L$  do // in ascending order
     $u' \leftarrow \text{UF.Find}(u)$ 
     $v' \leftarrow \text{UF.Find}(v)$  } Union-Find data structure
    if  $u' \neq v'$  then
         $F \leftarrow F \cup \{(u, v)\}$ 
        UF.Union( $u', v'$ )

```

return F

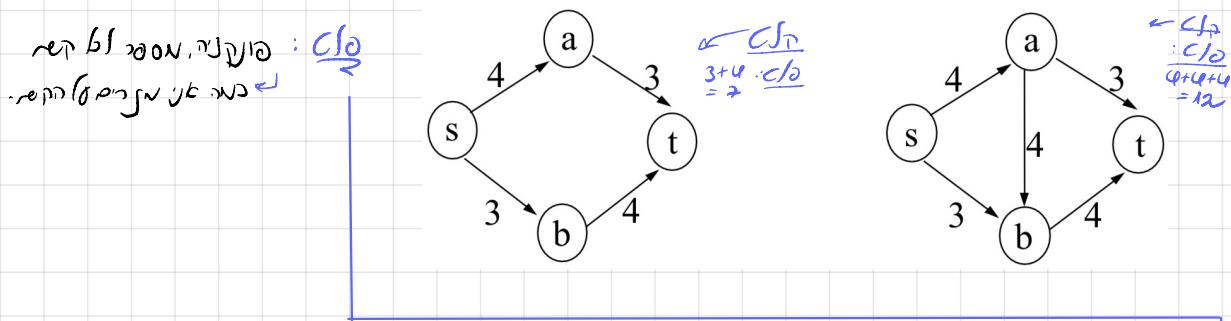


- גורם כפוי כנשכיה וסיבת מוקטינה בזרם.
- אם $C \in V$ לא מוגדר שפהם / אם נתקה A גוראה B אונשותי זהו מודם קחורה.
- גוראה גוראה נתקה זהה ומפיקת כהה מה שפהה הוא זרחה.
- מטה זרף זרף פלזר דה? וזה רוחש? וזה גוראה?

הצורה: נסמן \mathcal{G} הנקודות (nodes) V , קוווקט (edges) E , וזרם (flow) f .
 $f(u, v)$ מינימום (minimum capacity) $C(u, v)$ של קוווקט (u, v) .

תפקיד: נסמן \mathcal{G} הנקודות (nodes) V , קוווקט (edges) E , וזרם (flow) f .

השאלה: נסמן \mathcal{G} הנקודות (nodes) V , קוווקט (edges) E , וזרם (flow) f , ופואן $f(s, t) = f$?



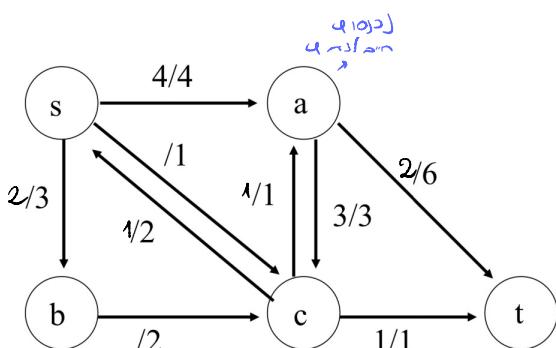
כל פערני, $f(s, t) = f$ $\forall (u, v) \in E$ ו- $f(u, v) \leq C(u, v)$ (I) פערני וקיטי הנקודות הינה פערני $f: V \times V \rightarrow \mathbb{R}$ (II) פערני וקיטי הנקודות $f(u, v) \leq C(u, v)$ (capacity constraint) (III) פערני וקיטי הנקודות $f(u, v) = C(u, v)$ (flow equality).

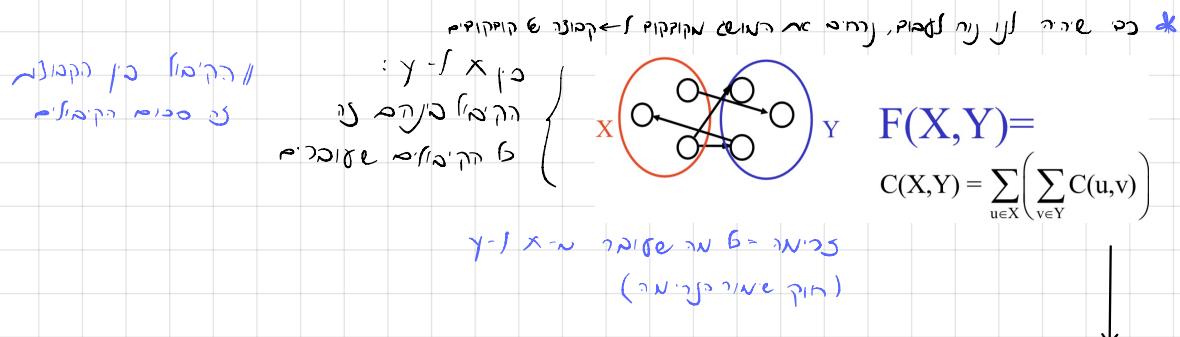
$\sum_{e \in \text{out}(u)} f(e) - \sum_{e \in \text{in}(u)} f(e) = 0 \quad \forall u \neq s, t$ (flow conservation): נסמן $\text{out}(u)$ ו- $\text{in}(u)$ כ- $e \in E$ ש- $e \in \text{out}(u)$ ו- $e \in \text{in}(u)$. $\sum_{v \in V} f(v, t) = |f|$ (flow to sink): הנקודות v ש- $f(v, t) > 0$ ו- $t \in V$. (flow from source): $\sum_{v \in V} f(s, v) = |f|$ (flow from source).

תרגיל

שנויות הזרימה הבאה רשותם כל הקיבולים ופרק חלקי מערכיו הזרימה.
 $f(s, a) = 4/4, f(s, b) = 2/3, f(a, c) = 1/1, f(b, c) = 1/2, f(a, t) = 2/6, f(b, t) = 1/1, f(c, t) = 3/3$

על גבי רשת הזרימה הבאה רשותם כל הקיבולים ופרק חלקי מערכיו הזרימה.
השלימו את ערכי הזרימה.

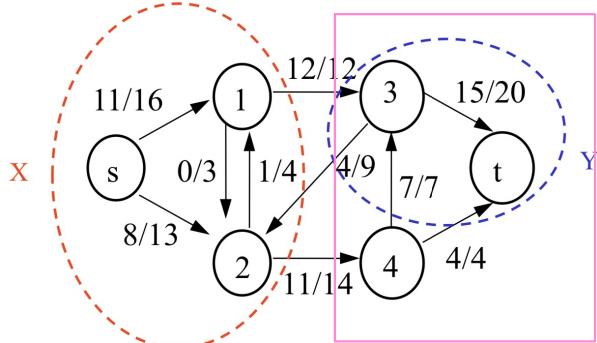




$C(X,Y) = C(1,3) + C(2,4) = 12 + 14 = 26$

$f(X,Y) = f(1,3) + f(2,4) + f(2,3) = 12 + 11 - 4 = 19$

* ברכישת קוקיז ב- $\{1,2\}$ במקורה גאנ.



$C(X,Y) = C(1,3) = 12$

$f(X,Y) = f(1,3) + f(2,3) = 12 + (-4) = 8$

טרכ נאקסום 8
זרכו 8

תכונות של פונקציית זרימה

משפט: תהינה V קבוצות כלשהן ותהי f פונקציית זרימה, אז מתקיים:

$$f(X,Y) \leq C(X,Y) \quad (1)$$

$$f(X,Y) = -f(Y,X) \quad (2)$$

$$f(X,X) = 0 \quad (3)$$

(חוק הפילוג) אם $X \cap Y = \emptyset$ אז:

$$f(Z, X \cup Y) = f(Z,X) + f(Z,Y) \quad \bullet$$

$$f(X \cup Y, Z) = f(X,Z) + f(Y,Z) \quad \bullet$$

טענה:

$$|f| = \sum_{v \in V} f(s,v) = \sum_{v \in V} f(v,t)$$

ניסוח שקול: $|f| = f(s, V) = f(V, t)$

. $t \in T \cup S$ $s \in S$ $t \in T$ $T = V \setminus S$ $S \subseteq T$ \rightarrow ~~פערוציון קומקזין גאנ~~ ~~פערוציון קומקזין גאנ~~

~~פערוציון קומקזין גאנ~~ f ~~פערוציון קומקזין גאנ~~ \rightarrow f ~~פערוציון קומקזין גאנ~~ \rightarrow ~~פערוציון קומקזין גאנ~~

פערוציון
פערוציון

$$C(S, T) = \sum_{u \in S} \sum_{v \in T} C(u, v)$$

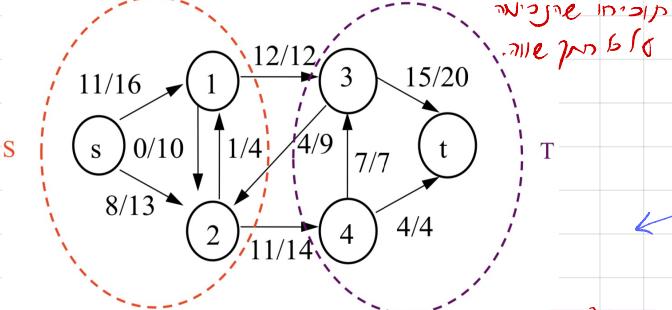
$$f(S, T) \leq C(S, T)$$

זרימה בחתך = סכום הקיבולים החוצים את החתך

$$f(S, T) = \sum_{e \in (S, T)} f(e) - \sum_{e \in (T, S)} f(e)$$

אבחן: לכל חתך (S, T) מתקיים

למה החתך: תהי f זרימה כלשהי ויהי (S, T) חתך כללי. אזי $|f| = f(S, T)$



$$\text{הוכחה: } \sum_{v \in T} \left(\sum_{e \in \text{in}(v)} f(e) - \sum_{e \in \text{out}(v)} f(e) \right) =$$

שי מניה בשני דרכים

$$\begin{aligned} \text{החותם} &= (2, 4) - (4, 2) = 0 \\ \text{החותם} &= (1, 3) + (4, 3) - (3, 1) - (3, 2) = 0 \\ \text{החותם} &= (4, t) + (3, t) \end{aligned}$$

1. לכל קדקוד מה שנכנס = מה שיצא מעת t

2. כל קשת נספרת פעמיים פעם בכניסה ופעם

ביציאה מעת החתך

אם ניקח גורם כל קשת מעת החתך נזקן?

וזו נגיעה למסקנה שזו הטענה נכונה.

צ'כאי בחתך

$$f(2, 4) - f(4, t) - f(4, 3) + f(1, 3) + f(4, 3) - f(3, t) - f(3, 2) + f(4, t) + f(3, t) = 0$$

ולכן סכום

למה הדזרמה בחתך (S, \bar{S}) מתקיים שהזרמה בו שווה לעוצמת הזרמה בראשת, כלומר $f(S, \bar{S}) = F$

הוכחה:

נתבונן בסכום הבא: $\sum_{v \in \bar{S}} \left(\sum_{e \in \text{in}(v)} f(e) - \sum_{e \in \text{out}(v)} f(e) \right)$. מצד אחד, לכל $v \in \bar{S} \setminus \{t\}$ מתקיים חוק הצומת: $\sum_{e \in \text{in}(v)} f(e) - \sum_{e \in \text{out}(v)} f(e) = 0$.

לכן הסכום שווה לעוצמת הזרמה:

$$\sum_{v \in \bar{S}} \left(\sum_{e \in \text{in}(v)} f(e) - \sum_{e \in \text{out}(v)} f(e) \right) = \sum_{e \in \text{in}(t)} f(e) - \sum_{e \in \text{out}(t)} f(e) = F$$

מצד שני, לכל קשת $e \notin (S, \bar{S})$ התרומה שלה לסכום היא 0, כי אם הזרמה שלה מופיעה בבייטוי עם $+$, היא מופיעה גם עם $-$. לכן הביטוי מסכם רק את הזרמות שבחתך:

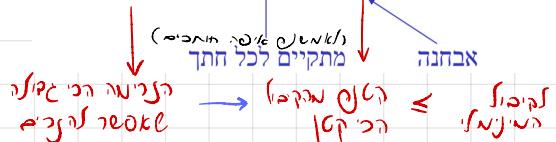
$$\sum_{v \in \bar{S}} \left(\sum_{e \in \text{in}(v)} f(e) - \sum_{e \in \text{out}(v)} f(e) \right) = \sum_{e \in (S, \bar{S})} f(e) - \sum_{e \in (S, \bar{S})} f(e) = f(S, \bar{S})$$

מסקנה: $\text{MaxFlow} \leq \text{MinCut}$ ← הטענה וכי גורה מושך חותם סימני הנקראן (כלומר הקיבול המינימלי בין כל קיבולי החתכים חוסם את הזרמה המקטולית)

הוכחה:

יהי (S, T) החתך בו מתקבל הקיבול המינימלי, ותהי f פונקציה זרימה בעלת $|f|$ מקסימלי. אזי מתקיים:

$$\text{Max Flow} = |f| = f(S, T) \leq C(S, T) = \text{Min Cut}$$



- למה החתך: תהי f זרימה כלשהי ויהי (S, T) חתך כללי. אזי $|f| = f(S, T)$ (זכורו $|f|$ מוגדר כסכום כל הזרימות שיוצאות מ- S)
- כול הזרימה שיוצאות מ- S
- $\text{MaxFlow} \leq \text{MinCut}$
- קוואלוות הם פסגת האבולוציה

• **פורך-פלקנסון**: $f(v) \geq f(u) + c(u,v)$ ו- $c(u,v) \geq 0$ ו- $f(v) \geq f(u)$

• **השלמה:** $f(v) = f(u) + c(u,v)$

• **מקרה ספציאלי שפה נטורה:** קורתן שפה נטורה L היא סט כל "טוקס" שוכחה על נטול מילים מה- L .
 (טוקס הוא כל כויה של אוסף מילים הכתובים ב- L)
 נסיגת שפה נטורה L שוכחה על שפה נטורה L' אם $L \subseteq L'$.

השיטה הבסיסית: הפלקנסון

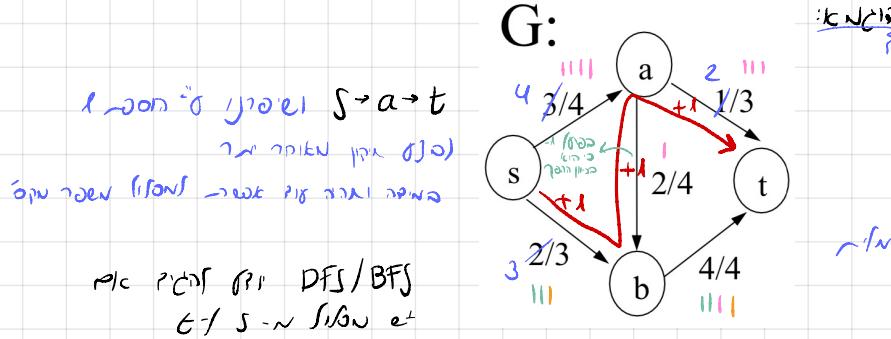
קלט: רשת זרימה G , פונקציית קיבול C ,
מקור ובור: s, t

פלט: זרימה f שערכה $|f|$ מקסימלי.

השיטה:

• **אתחול:** $f = 0$

• **כל עוד יש מסלול משפר P :** גודיל את הזרימה לאורך P



• **קשת מעוללה קדימה**, אם אינה רויה. $f(e) < c(e)$ // אמת כי אם קיימת קשת e מוקטן מוקטן

• **קשת מעוללה אחורה**, אם אינה ריקה כלומר. $f(e) > 0$ // אם קיימת קשת e מוקטן מוקטן

אנו שואים?
 קיימת קשת e מוקטן מוקטן
 ומייצרת קיבול חדש
 אך לא מוגדרת קשת שיוורית
 → DFS/BFS

- לכל קשת שמעוללה קדימה (v,u) מוגדרת קשת שיוורית $c(u,v) - f(u,v)$ // הקטנו טהיה פחס הרצאה
- לכל קשת שמעוללה אחורה (x,v) מוגדרת קשת שיוורית $c(v,x) = f(x,v)$

$$c(v,x) = f(x,v)$$

האלגוריתם של פורך-פלקנסון:

• **אתחול:**

זרימה 0 בכל הקשתות: $f(u,v) = f(v,u) = 0$

הגרף השיוורי G_f שווה לגרף המקורי G : $C_f(u,v) = C(u,v)$

DFS/BFS ↳

• **כל עוד יש מסלול משפר** (מסלול מ- s ל- t בגרף G_f)

נתק: א. נמצא מסלול כזה P ונחשב את קיבולו השיוורי $C_f(P)$ // גודלו של המסלול P מוגדר ב- $O(m+n)$

ב. גודיל את הזרימה ב- G לאורך P ב- $C_f(P)$ באופן:

לכל קשת (u,v) במסלול P נעדכן: $f(u,v) = f(u,v) + C_f(P)$

מכאן (ההערכות נמשכו)

ג. נעדכן את הגרף השיוורי לאורך המסלול P :

לכל קשת (v,u) במסלול P נעדכן את הגרף השיוורי כדי שהוגדר לפני מספר שקפים

מסלול משפר

הגדרה: מסלול משפר הוא מסלול מ- s ל- t ב- G_f

• נחזר את f

קיבול של מסלול P : הקיבול המינימלי לאורך P

$$C(P) = \min_{(u,v) \in P} C(u,v)$$

קיבול שיוורי של מסלול P : במקרה, ברשת השיוורית

$$C_f(P) = \min_{(u,v) \in P} C_f(u,v)$$

מסלול P הוא מסלול משפר אם ורק אם הקיבול השיוורי של P חיובי. \Leftarrow

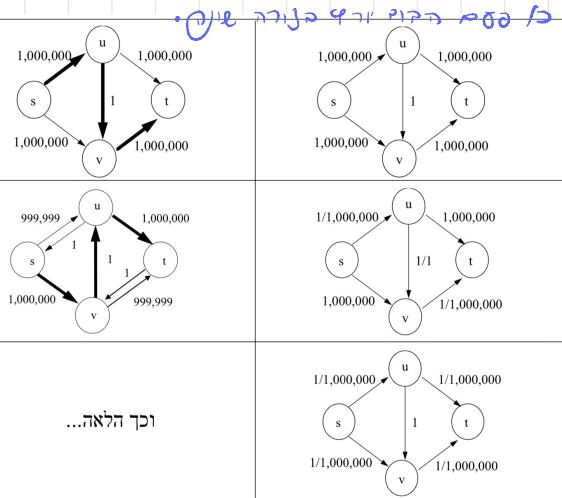
• **אם** **ולא** **אפשר** **הטענה**

• **אם** **ה** **אפשר** **הטענה**

• **אם** **ה** **אפשר** **הטענה**?

$O(m+n)$ ↳ **ולא** **אפשר** **הטענה**

סימון: MF = (ערך הזורימה המקסימלית).



טענה: בראשת שבת כל הקיבולים הם מספרים שלמים,MF האיטרציות שיבצעו האלגוריתם במקרה הגרוע הוא ||MF||.

הוכחה:

חסם עליון: בכל איטרציה, ערך הזורימה גדול לפחות ב- 1.

חסם תחתון: בשקף הבא ישנה דוגמא לgraf G וסדרה של בחירות למסלולים משפרים ←

מי יוציא מהטפסה? יוציא מהטפסה?

ברשת שיש בה קיבולים אי-רצינגולים, יתכן ששיטת פורד-פלקרסון לא תסתיים.

(טפסה יתעכבה)

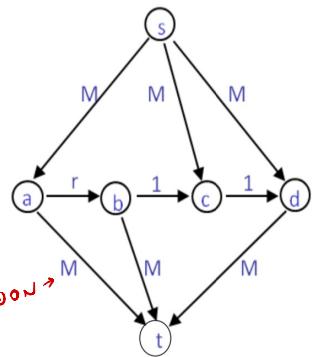
* קיימת קבוצה K כזו שהיא מוגדרת כטפסה!

זה לא אונליין כי ברגע אחד נוציא מהטפסה וזה לא יתאפשר!

במקרה אחד, גורר מטפסה לאין-סוף

מיירון עזובות ותולעת-ה-גאנט גאנט

$$r = \frac{-1 + \sqrt{5}}{2}$$



משפט Edmonds, Karp $O(nm^2)$

אם בוחרים מסלול משפר בעזרת BFS, מספר האיטרציות אינה עולה על nm .

$$O(m) \cdot O(n \cdot m) = O(nm^2) < O(n^5)$$

הערה: טענה זו נכונה עבור גרפים עם קיבולים ממשיים כלשהם.

Dintiz

שיפור בפאה של כל המסלולים המשפרים בעלי אותו אורך (n איטרציות)

מסקנה: מתקיים $m \leq n$

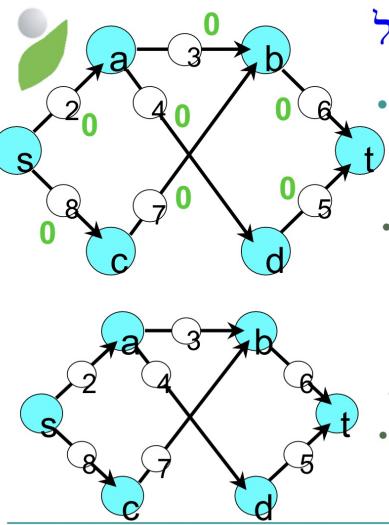
אם משתמשים באלגוריתם פורד-פלקרסון כאשר המסלולים המשפרים נבחרים בעזרת BFS, אזי זמן הריצה יהיה: $O(nm^2)$.

הລוקה והטפסה קיינס-טומאס-ריציאן
פ.ר. - גראונד אוניברסיטה - 2023

הגן-הוקה וגרה-ק.ה קאנז

*

דוגמת הרצה – איתחול



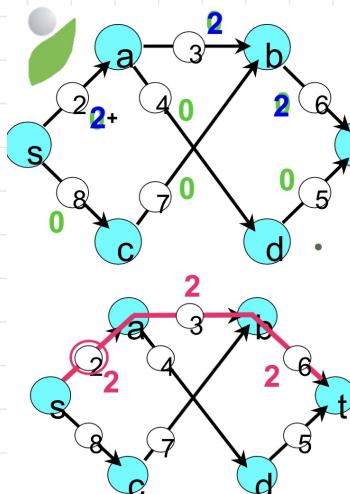
דוגמת הרצה – איתחול

נקבע זרימה התחלתית

- נהוג לבחור זרימה התחלתית כ-0 בכל קשת
- אולם ניתן להתחילה גם עם זרימה **תוקנית** אחרת

ע"ס הרשות והזרימה

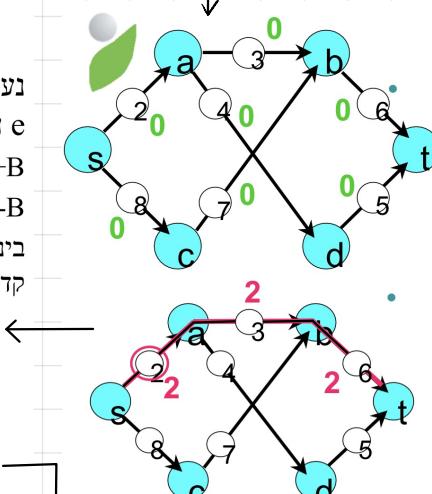
- בה, נבנה רשות **שיוירית**
- שם אחר – רשות תיקונים ש-FF, FF-הרשות אינה חיונית לאר מקרה על הרצתו



דוגמת הרצה – פaza 1

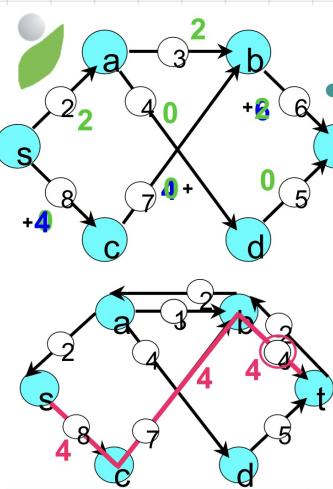
- נעתק את הזרימה לרשות
- המקורית: בכל קשת e המוקורי: $f(e) + B$
- מעילה אחריה e אם $f(e) - B$: מועילה אחרת e אם
- בнтיטים כל הקשות מהמעילות רק קדימה

$f(e) + B$ – זרימה
 $f(e) - B$ – זרימה
 מועילה אחרת
 לא מועילה



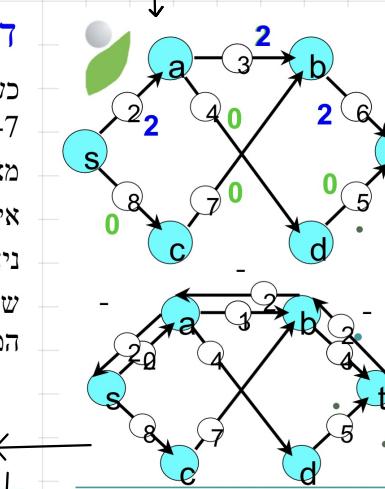
דוגמת הרצה – פaza 1

- נמצא מסלול שיפור בראשת השיוירית
- נאטר קיבול מינימלי במסלול (צואר הבקבוק).
- במסלול (צואר הבקבוק) B=2 (במקרה זה B=2) נסמננו זרמים במסלול הנ"ל זרימה בגודל



דוגמת הרצה – פaza 2

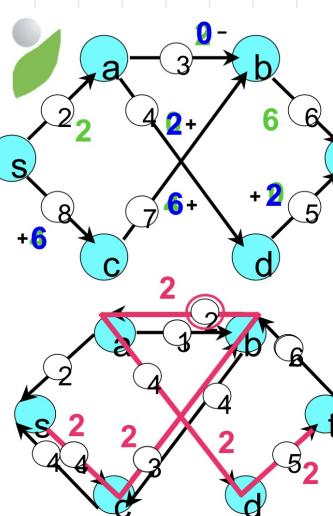
- כעת נחזור על כל השלבים 3-7 עד עוד ניתן לבצע
- מציאת מסלול שיפור
- איתור צואר הבקבוק
- ניצול המסלול
- שיפור הזרימה בראשת המקורית



דוגמת הרצה – פaza 1

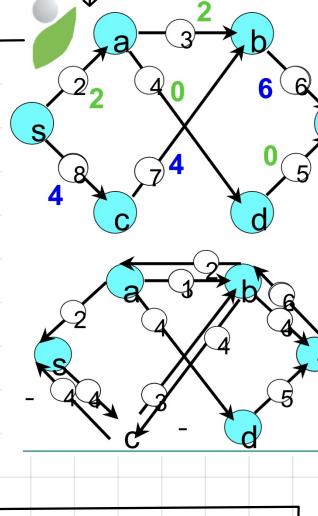
- נעתק את הזרימה לראשת
- המקורית: בכל קשת e המוקורי: $f(e) + B$
- מעילה אחריה e אם $f(e) - B$: מועילה אחרת e אם
- בнтיטים כל הקשות מהמעילות רק קדימה

נעדכן את הרשות השיוירית: בכל קשת (במהותה הזרימה) e קדימה $f(e) + B$ – זרימה
 $f(e) - B$ – זרימה
 מועילה אחרת e אם: $D(e) = f(e) - B$
 מועילה אחרת e אם: $D(e) = f(e)$



דוגמת הרצה – פaza 3

- מציאת מסלול שיפור
- איתור צואר הבקבוק
- מילוי המסלול
- שיפור הזרימה בראשת המקורית
- קשות במסלול מועילות 4 קדימה, והטיפול בהן כמו קודם
- הקשת האמצעית מועילה אחרת, לכן עדכון הזרימה בה נעשה ע"י חיסכון
- עדכון הרשות השיוירית



דוגמת הרצה – פaza 2

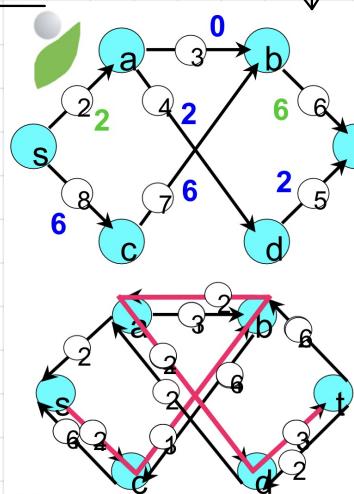
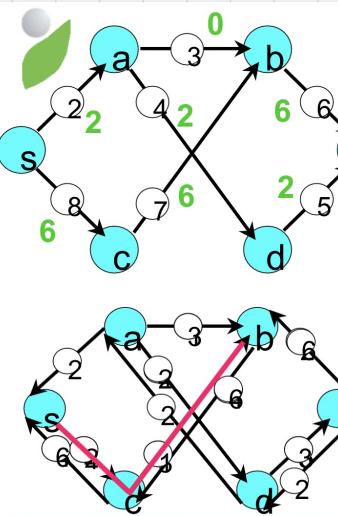
- כעת נחזור על כל השלבים 3-7 עד עוד ניתן לבצע
- מציאת מסלול שיפור
- איתור צואר הבקבוק
- מילוי המסלול
- שיפור הזרימה בראשת המקורית
- עדכון הרשות השיוירית

דוגמת הרצאה – פaza 3

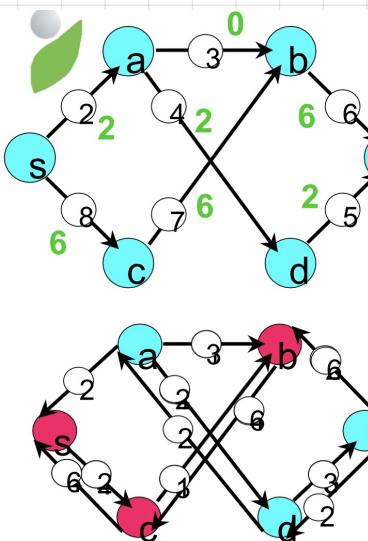
- העדרון נעשה כמו קודם
- מוציא לה קדימה e אם $D(e) = c(e) - f(e)$
- מוציא לה אחוריה e אם $D(e) = f(e)$

דוגמת הרצאה – פaza 4

- מציאת מסלול שיפור
- b – אבל אין המשך מצומת
- חייבים לסתיג חיבורים דרכיהם אחרות מ
- ג-חישוב – חזרים ל-
- נכשל – אין אפשרות c -מהחר וגם מ
- אין אפשרות t -מהחר וצתרים (את)
- (ריצת האלגוריתם
- מסקנה: הזרימה הנוכחית
- ברשות היא המקסימום



דוגמת הרצאה – תוצאות

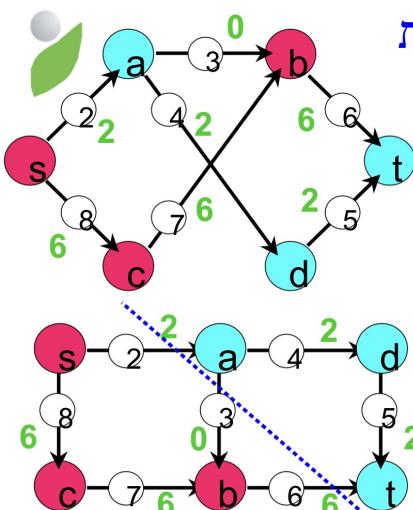


- מציאנו פונקציית זרימה (לכל קשת) הממקסמת את עוצמת הזרימה ברשות
- עוצמת הזרימה היא **8**
- מציאנו גם חתך מינימלי
- $S = \{s, b, c\}$ (הצטחים האדומים)
- $\bar{S} = \{t, a, d\}$

February 26, 2008

Advanced Algorithms © Maria Artishchev-Zapolotsky, 2008

13



דוגמת הרצאה – תוצאות

- בציר המקורי של הרשות, קשה לראות את החתך S שכן נשרטת את הרשות לאחר:
- ואז רואים כי קיובל החתך באמת שווה לגודל הזרימה

אנו יזכיר נושא אחד – איטטט
הנוסף לאלגוריתם – ארכיזם

אם האלגוריתם תמיד מסתים? או הוכח שם נספסים לא רצוקנים, והוכיחם יתגלו?

אם כן, האם הוא מוצא זרימה מקסימלית? כן. זה נכון הוכחה. אם קיימת הטענה שקיימת זרימה מקסימלית, אז קיימת זרימה מקסימלית.

כמה איטרציות ידרשו?

כיצד ובאיזה עליות נוכל לישם איטרציה?

הוכחה ונקודות ת העתקה מהרץ.

אם הטענה נכונה, אז f מוגדרת כזרימה מקסימלית. ובהכרח G_f מוגדרת כזרימת מקסימלית.

$\frac{f}{G_f}$ (3) (2) (1) (3) (2) (1)

משפט Max Flow, Min Cut

משפט: תהי f זרימה ברשת G . התנאים הבאים שקולים:

1. f זרימה מקסימלית ב- G .

2. בגרף השינוי G_f אין מסלול מ- s ל- t (אין מסלול משפר).

3. קיים חתך (S, T) שעבורו $|f| = C(S, T)$ (חתך רווי).

!

הוכחה: (1) \Rightarrow (2) (2) \Rightarrow (3) (3) \Rightarrow (1)

או אפשר להגדיל את הזרימה f בעורתו, ולכן f אינה מקסימלית.

$$|f| \leq C(S, T)$$

(3) לכל חתך (S, T) מתקיים:

לכן, אם קיים חתך (S, T) שעבורו $|f| = C(S, T)$ אז f זרימה מקסימלית.

(2) \Rightarrow (3) נניח שב- G_f אין מסלול מ- s ל- t , ונגידיר:

{יש מסלול מ- s ל- v ב- G_f ו- $v \in S$

• (3) חתך כי $v \in S$ וגם $S \notin t$ (כי אין מסלול מ- s ל- t ב- G_f).

$$f(u, v) = C(u, v) \quad u \in S, v \in T$$

אחרת הקשת (u, v) שיכת ל- G_f וואז גם $v \in S$.

$$|f| = f(S, T) = C(S, T)$$

פורד-פלקרסון – נסונות חלקית

משפט: אם אלגוריתם פורד-פלקרסון יוצר, אז זרימה f היא מקסימלית

נסון (1). נסונות חלקית סוגר
max f נסון (2).
3. נסון (3). נסונות חלקית

הוכחה:

אם האלגוריתם יוצר או אין בגרף השינוי G_f מסלולים משפרים. לפי משפט Max-Flow, Min-Cut זרימה f מקסימלית.

הערה: האלגוריתם מאפשר גם למצוא חתך (עם קיבול) מינימלי.

0. אונסן (4) $O(mn)$

1. אונסן (5) ועדי גודר כוכב.

2. אונסן (6) ועדי גודר כוכב.

פתרונות ידועות נוספת נוספות

$Mf \leq \min cut$ // גורילה האקסימר הולן
שא נאומן גאנז

לגרפים עם קיבולים שלמים:

$$C_{\max} = \max_{(u,v) \in V \times V} (C(u,v))$$

נמקן: 1. היה ש- $MF \leq nC_{\max}$, אז מספר האיטרציות שיבצע האלגוריתם הוא לכל היותר $\frac{1}{nC_{\max}}$. זכרו כי $\frac{1}{nC_{\max}} \leq k$.

2. אם בוחרים תמיד מסלול משפר בעל קיבול שיורי מרבבי, מספר האיטרציות אינו עולה על $(2m \log(C_{\max}))$ (מהו ייעילות האלגוריתם?).

לגרפים עם קיבולים ממשיים כלשהם:

$$\text{ראינו אלגוריתם שרצ' בזמן: } O(nm^2)$$

קיים אלגוריתם נוסף, למשל דינץ' $O(mn^2)$ גודלברג-טרוז' $O(n^3)$ ועוד.

* $C(s, \tau) \leq nC_{\max}$

$\min cut$

Mf

אלגוריתם רוז' - גודלברג

ליניציאן

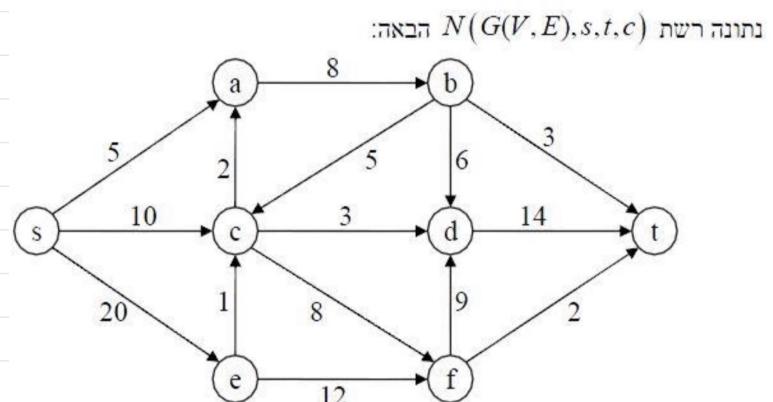
הוכחה של אופטימיזציה אוניברסלית, באמצעות ארכיטקטורה

באותה מטריצה פונקציית אופטימיזציה אוניברסלית,

ירגע פונקציית מינימיזציה ופונקציית מקסימיזציה כאותן.

- באיטרציה ה k נמצא את כל המסלולים באורך $k+1$ מינימום ומשפר על colum MIN
- מכאן k איטרציות אולם כל איטרציה תהיה מה

תרגיל

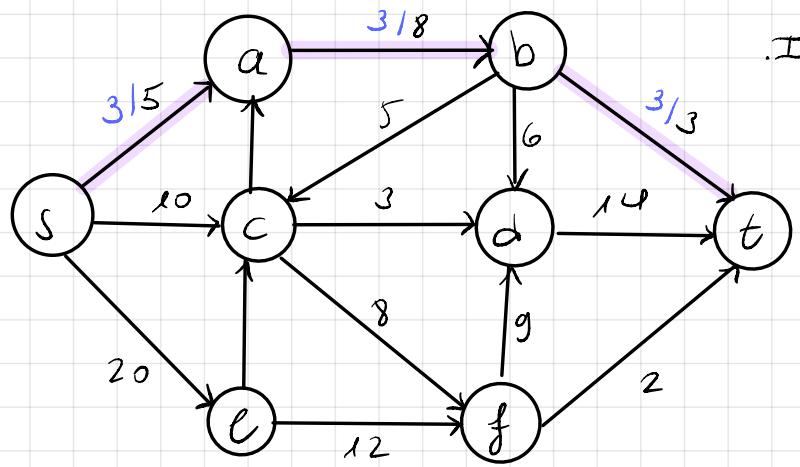


1. הריצו על הרשת את האלגוריתם של Edmonds-Karp

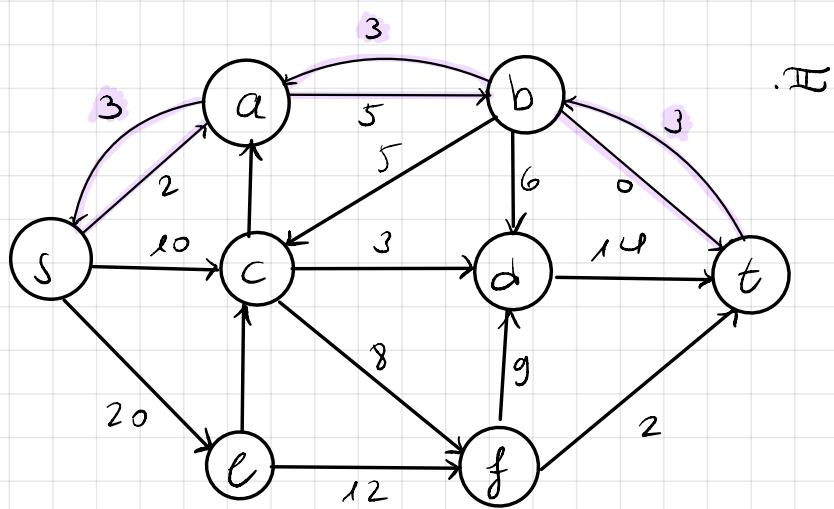
2. מה הערך האקסימר של הזרקה?

לעומת BFS,DFS מנסה למצוא את המסלול

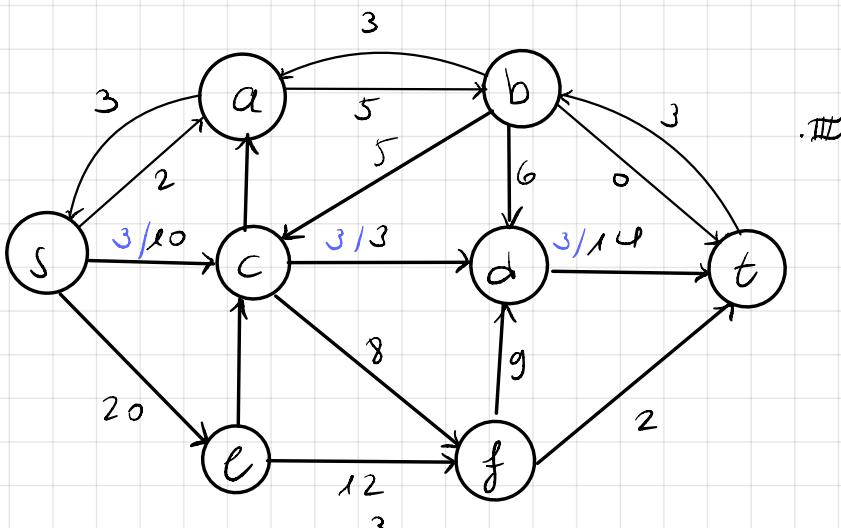
- BFS מנסה למצוא את המסלול
- DFS מנסה למצוא את המסלול
- BFS מנסה למצוא את המסלול



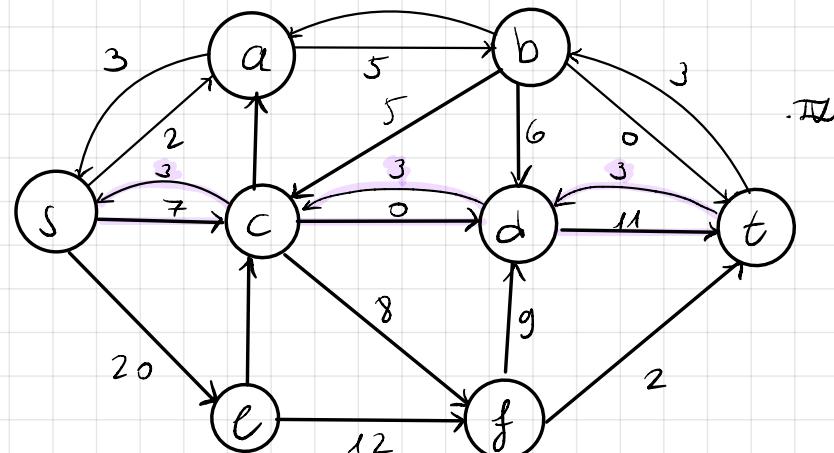
הדרישה היא למצוא מסלול מ-s ל-t.



הדרישה היא למצוא מסלול מ-s ל-t.



הדרישה היא למצוא מסלול מ-s ל-t.



* פונקציית הזרימה כפולה מוגדרת כפולה בפונקציית הזרימה
הפעלה של הזרם בפונקציית הזרם הינה רציפה

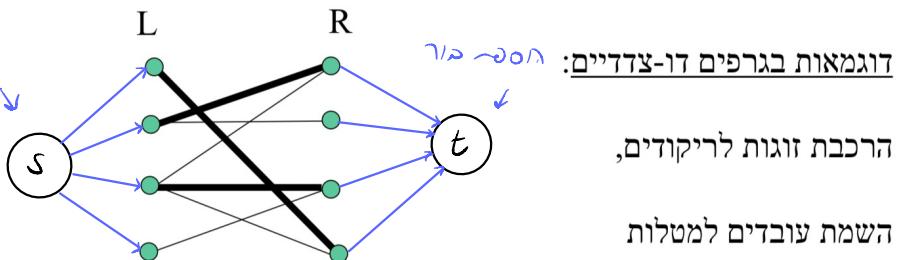
זיווגים - חיבור גראף זעיר קדילס

זיווג (Matching) בגרף לא מכוון G הוא תת-קובוצה M של צלעות, כך שכל קדקוד v שיך לכל היותר לצלע אחת מ- M .

זיווג מרבי (Maximum matching) הוא הזיווג הגדול ביותר שאפשרי ב- G .

לכט: קובוקים
לכט: מומן קבוק
לכט: נספֶּה אקור

זיווג מושלם (Perfect matching) הוא זיווג שבו משתתפים כל הקודקודים



הערה: זיווג מושלם בגרף דו-צדדי הוא זיווג שבו $|M| = |L| = |R|$.

אלגוריתם למציאת זיווג מרבי בגרף דו-צדדי

הרעיוון: רדוקציה לביעית הזרימה ברשת 1-0.

הקלט: גוף דו-צדדי לא מכוון $G = (L \cup R, E)$ (בגה"כ ללא קודקודים מבודדים).

האלגוריתם:

1. בניית רשת זרימה N באופן הבא:

- נוסיף שני קודקודים חדשים s, t .

- נוסיף קשתות מכוננות מ- s לכל הקודקודים בקובוצה L .

- נוסיף קשתות מכוננות מכל קדקוד בקובוצה R ל- t .

- נכוון כל צלע מ- L ל- R .

- הקיבולים של כל הקשתות יהיו 1.

2. נמצא זרימה מקסימלית f ברשת החדשיה בעזרת אלגוריתם פורד-פלקרסון בהינתן זרימה f , נגידיר זיווג M באופן הבא:

$$M = \{(u, v) \mid u \in L, v \in R, f(u, v) > 0\}$$

(I) **נכיהה - M זיווג:**

לכל קדקוד $L \in u$ ננכנת רקשת אחת: (u, s) (שיקולו 1).

לכן הזרימה המקסימלית שתיתכן לתוכה s היא 1.

בזרימה עם ערכי שלמים תיתכן לכל הזרום קשת אחת עם זרימה חיובית שווייצאת מ-1.

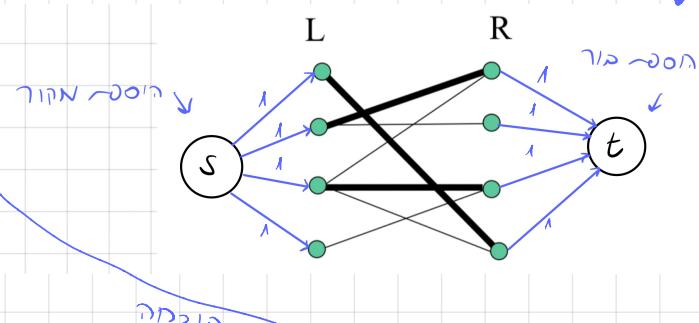
מכאן שכל קדקוד $L \in u$ מופיע לכל היותר בקשת אחת של M .

בדומה ניתן להראות שכל קדקוד $R \in v$ מופיע לכל היותר בקשת אחת של M .

זהו זיווג M . ↙

(II) **נכיהה - $|f| = |M|$:** הזרימה בחתק $(s \cup L, t \cup R)$ שערכה הוא $|f|$, וזה מספר הצלעות שימושת בזיווג M .

3. הזיווג M יהיה: $M \subseteq (u, v) \in L, v \in R$ ו- $f(u, v) > 0$.



CJS קראק וקואם

CJS קראק

הוכחה: געפין זרימה על כל צלע.

אם $(u, v) \in M$ אז: $f(s, u) = f(u, v) = f(v, t) = 1$

לכל יתר הקשתות נגידיר $f(u, v) = 0$

M הוא זיווג ולכן כל המסלולים $t \rightarrow v \rightarrow u \rightarrow s$ זרים בקדקודים (פרט לקצחות (s, t)).

מתקבלת זרימה חוקית, שגודלה הוא $|f| = |M|$ (הוכחה!). ↙

יעילות האלגוריתם

טענה: בהינתן גרפּ דו-צדדי $G = (L \cup R, E)$, ללא קודקודים מבודדים, האלגוריתם מוצא זיוג מרבי ביעילות $O(mn)$, כאשר $|L| = n$ ו- $|E| = m$.

הוכחה: בניית הרשת N לוקחת זמן ליניארי.

מס' הקשיות ב- N הוא: $n = m + n$

ולכן כל איטרציה של פורד-פלקרסון תיקח $O(m+n)$.

היות שלא תהיינה יותר מ- n איטרציות (מדוע?), או זמן מציאת הזרימה יהיה $O((m+n)n)$.

היות שבגרף אין קודקודים מבודדים, מתקיים $m \leq n$. ולכן $O(mn)$.

תרגיל: שידוכים ביגמיים

of 54

בעיר מסוימת יש n תושבים, בהם $\frac{n}{2}$ נשים ו- $\frac{n}{2}$ גברים. מותר לכל גבר לשאת 2 נשים; לאישה מותר להתחתן עם כמה גברים שהוא רוצה. לכל איש יש רשות גברים שעמם היא מוכנה להתחתן (הגברים אינם מסרבים).

כתבו אלגוריתם יעיל שモצא האם אפשר להתחתן את כל הנשים (כלומר שכל איש תהיה נשואה לפחות לגבר אחד).



ניתוח הייעולות: נסמן ב- m את מספר האיברים הכלול ברשימות השידוכים הפוטנציאליים של הנשים. הבינו את זמן הריצה כפונקציה של n ו- m .

תרגיל: רשימות נוסעים

N משפחות יוצאות לטויל ב- M מכוניות. במשפחה ה- i יש f_i נפשות ובמכונית ה- j יכולים לשבת v_j אנשים.

כתבו אלגוריתם הבודק האם אפשר לשבץ את האנשים למכוניות כך ששבשם מכונית לא יהיה יותר מאדם אחד מאותה המשפחה.



מה ייעולות?

לפניהם וארחין אקמנס כפ"ג ישי.

בג' גראן קואגמה. (ז) קואגמה אקלטוגר A נרכשת כל ב- קואגמה אקלטוגר B (קואגמה אקלטוגר A אף כי נרכשת לא מפ' קואגמה אקלטוגר B , אך אם גם הוא מפ' אקלטוגר A אף כי נרכשת לא מפ' קואגמה B .

טכ' ערך שטח הולר: $\Delta V = \frac{1}{2} \pi R^2 \Delta h$. נרכשת לא מפ' קואגמה B (קואגמה אקלטוגר A אף כי נרכשת לא מפ' קואגמה B).

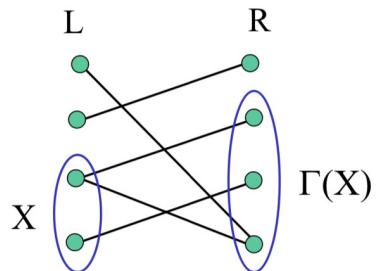


Hall משפט

יהי $G = (V, E)$ גרף לא-מנוון

$$\Gamma(x) = \{y \in V \mid (x, y) \in E\}$$

קובוצת השכנים של קדקוד v היא: $\Gamma(v)$



משפט Hall: יהי $G = (L \cup R, E)$ גרף דו-צדדי לא-מנוון.

(ב) יש זיווג שלם ל- L אם ורק אם $(\forall L \subseteq X \text{ מתקיים } |X| \leq |\Gamma(X)|)$

כל זוג זוגות

הוכחת מספיקות

נניח שלכל $X \subseteq L$ מתקיים $|X| \leq |\Gamma(X)|$. יהי $|L| = n$

נראה שהאלגוריתם למציאת זיווג מרבי מ חוזר ח' ובכך נוכחה שיש זיווג שלם ל- L .

נתבונן ברשף זרימה שמייצר האלגוריתם לזווג מרבי

ונוכחה שקיבול החתק המינימלי ברשף הוא ח'

הזרימה המקסימלית שווה בערכה ל- n
הזווג המרבי הוא בגודל n .

היות שלא תיתכן זרימה גדולה מ- n , די להראות שקיבולו של כל חתק $\leq n$

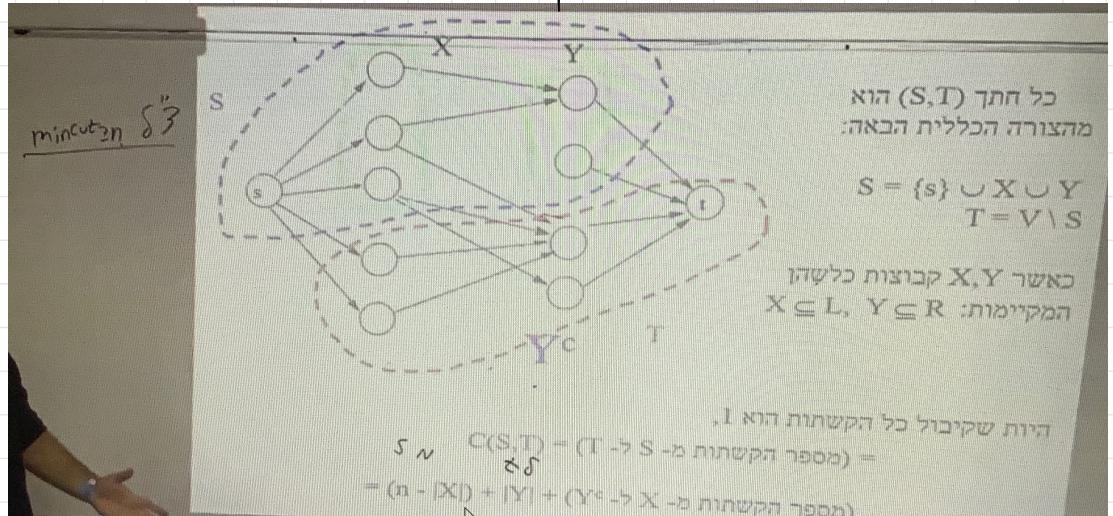
וכבר - הוכיחו

קיים זיווג \leftarrow זוג \leftarrow זוג

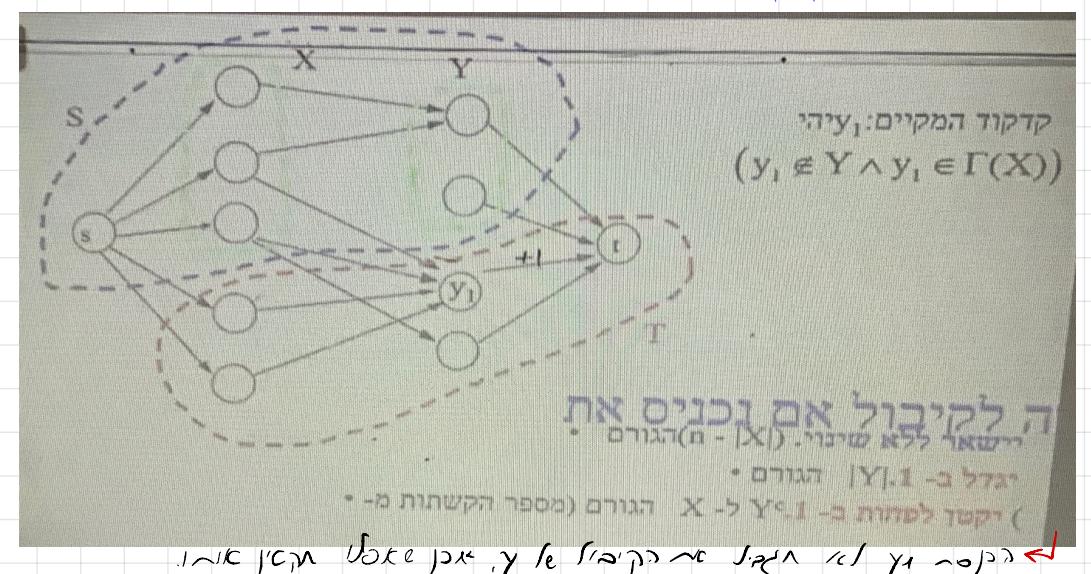
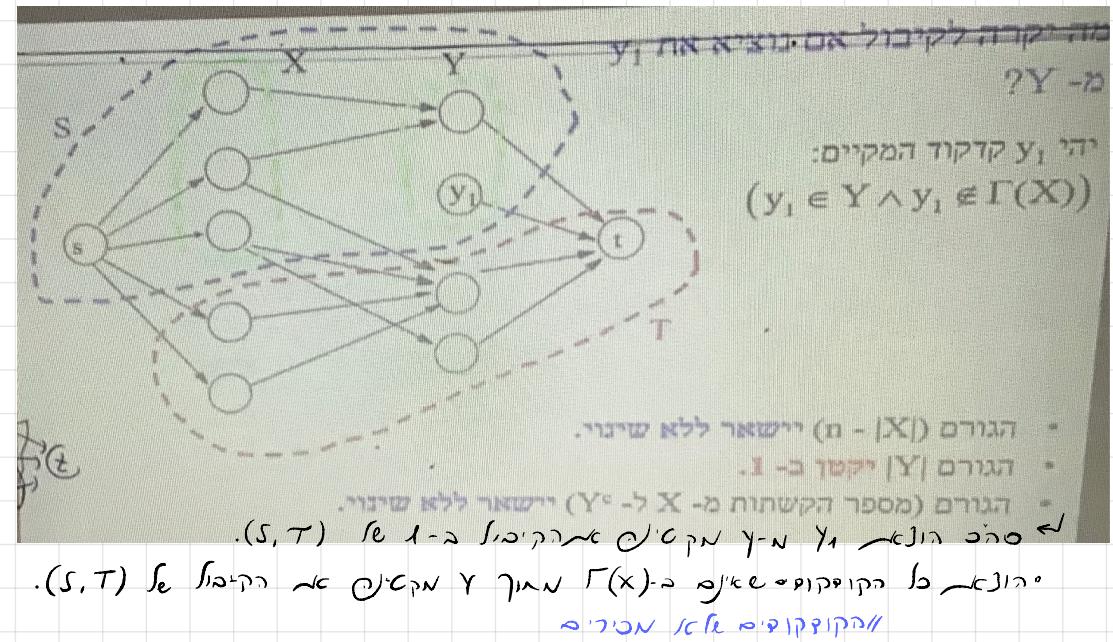
$f_{max} \geq n \leftarrow C(\min cut) \geq n$

קיים זיווג \leftarrow זוג \leftarrow זוג

ה證明 מתקיים בדקה. נוכיח שקיים זוג (S, T) המקיימים את הדרישה.



$$\begin{aligned} |X| &\leftarrow |x| \\ |Y| &\leftarrow |y| \\ |T| &\leftarrow |x| + |y| \\ |Y_c| &\leftarrow |x| + |y| \end{aligned}$$



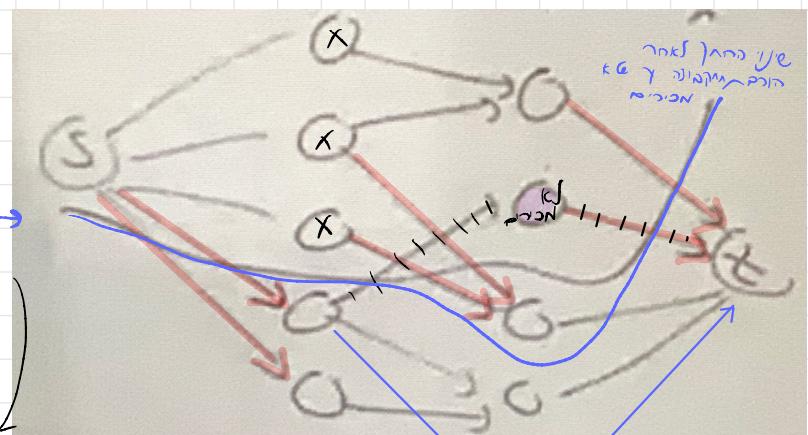
גקלס גולן 6 סעיף ב מילוי יפה נא שאלת שאלת

גקלס גולן 6 סעיף ב מילוי יפה נא שאלת שאלת

שאלה גקלס גולן 6 סעיף ב מילוי יפה נא שאלת שאלת

שאלה גולן 6 סעיף ב מילוי יפה נא שאלת שאלת

שאלה גולן 6 סעיף ב מילוי יפה נא שאלת שאלת



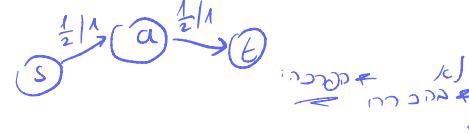
$$n < ((x) + |x| - 1)$$

$$Y = \pi(x) \text{ נס. כשל}$$

$$\begin{aligned} n &\geq |x| + \pi(x) \\ &\Updownarrow \\ |x| &\leq n - \pi(x) \end{aligned}$$

תרגיל מגניב

במזה שודרגן כמספרים אלמנטריים. ומקודם לו $\max(a, b)$. חישוב גודל הערך החדש נועז, מושג באמצעות גלגול הערך החדש. גלגול הערך החדש, מושג באמצעות צורה סימטרית, על ידי אובייקט אחד, וזהו אובייקט אחד. אם גולגולתיהם קיימת מוקסימום טריוויאלית, זרימה וריאנטית מוצפנת עירא $\min_{\text{cut}} = \max_{\text{fbv}}$



• **הוכחה או הפרדה:**

• **אם כל קיובלי הקשות ברשთ הם מספרים שלמים, אז כל זרימת מקסימום אפשרית ברשת מגדירה זרימה שלמה בכל קשת**

• **אם כל קיובלי הקשות ברשת הם מספרים שלמים, אז עצמת זרימת מקסימום ברשת היא בהכרח שלמה.**

לכן: אם קיימת זרימת מקסימום ברשת המגדירה זרימה שלמה, אז עצמת זרימת מקסימום ברשת היא בהכרח שלמה. לכן: אם קיימת זרימת מקסימום ברשת המגדירה זרימה שלמה, אז עצמת זרימת מקסימום ברשת היא בהכרח שלמה.

לכן: אם קיימת זרימת מקסימום ברשת המגדירה זרימה שלמה, אז עצמת זרימת מקסימום ברשת היא בהכרח שלמה.

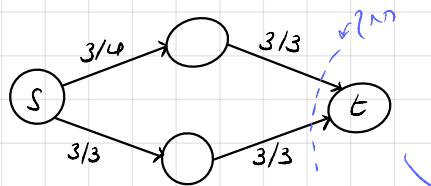
לעתה נראה כיצד ניתן לבנות אוסף מקסימום בעקבות גולגולת שלם.

תרגיל אש

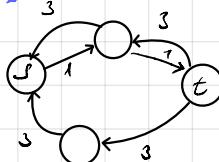
- 1: חיפוך ב- \mathbb{Z}_2 גזרה רק אם ה- \mathbb{Z}_2 גזרת
שניהם, אך אם הגדרה מוגבהת
כ- \mathbb{Z}_2 גזרה.

בבינתן זרימה מקסימלית f מגדילים קיבול של צלע
אחד ב-1 הצביעו אלגוריתמים יעיל לתיקון f

- בבנתן זרימה מקסימלית f הורדנו 1 מקיבול של צלע
הצביעו אלגוריתמים יועל לתיקון f



זמן אמצעי בזאת $O(m+n)$



זמן אמצעי: $O(m+n)$

2. אם ה- \mathbb{Z}_2 גזרה הינה מינימלית, מילא את ה- \mathbb{Z}_2 גזרה הינה. מילא את ה- \mathbb{Z}_2 גזרה הינה.

$O(m+n)$

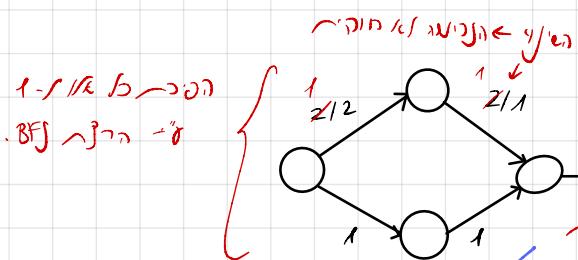
3. מילא את ה- \mathbb{Z}_2 גזרה הינה.

4. מילא את ה- \mathbb{Z}_2 גזרה הינה.

5. מילא את ה- \mathbb{Z}_2 גזרה הינה.

6. מילא את ה- \mathbb{Z}_2 גזרה הינה.

7. מילא את ה- \mathbb{Z}_2 גזרה הינה.



ב-2 BFS נסמן

8. מילא את ה- \mathbb{Z}_2 גזרה הינה.

9. מילא את ה- \mathbb{Z}_2 גזרה הינה.

(ב-2 BFS)

שאלות ממבחןים

נתונה רשת חקשותת ובה מחשבים שמחוברים ביניהם בקווים חד כיווניים.

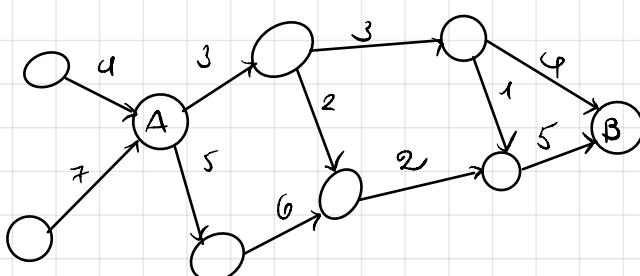
לכל קו יש עלות כספית נתונה. מי שרווכש קו שולט על התעבורה בקו זה.

מחשב A מעוניין לשלווח הودעה למחשב B.

חברה מסוימת מעוניינת למנוע מההודעה להגיע ליעדה, ולכן רוצה לרכוש קווים
כך שלא תאפשר תעבורה בקוים שלו, ולכן $\frac{1}{2}$ לא יוכל להעביר הודעה ל- $\frac{1}{2}$

אנו שואלים האם ה Hoduda מניין $A \rightarrow B$ מוגדרת מינימלית.

=
רעיון ש. נס. נס.



וליכטן רצויו...

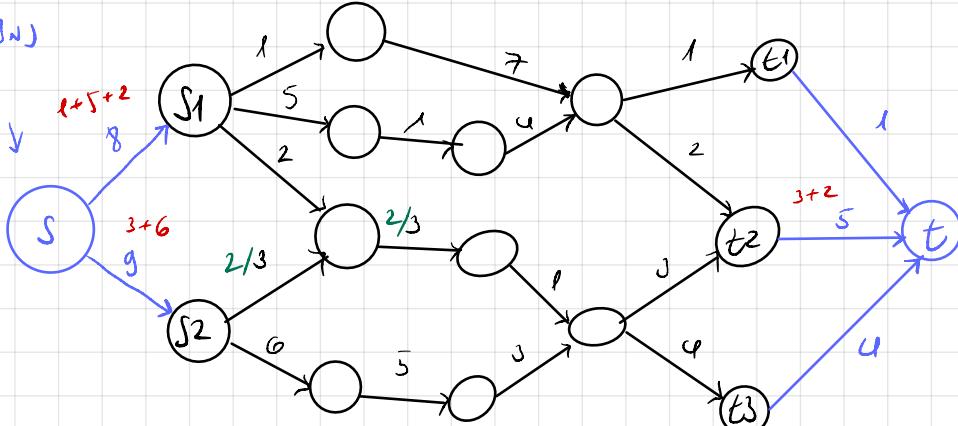
ולקחחים:

1. כרכבים \rightarrow סיבוב גורר פונקציית זרינטו.
2. או גורם זרינטו יוקטן \rightarrow קומונט זרינטו, רצינט זרינטו חילוקי גורם פונקציית זרינטו.
3. הטענה גנומית פגיעה פגיעה $\rightarrow A \rightarrow B$ הינה מוגדרת מינימלית.

לעתה: $O(nm^2) \rightarrow$ ה-EK: $O(nm^2)$
D.N.: $O(n^2m)$

נתונה רשת זרימה שבה יש קבוצת מקורות s_1, s_2, s_3 וקבוצת
בורות t_1, t_2, \dots, t_n .
מצאו זרימה חוקית שערכה מקסימלי כשערן הזרימה הוא סכום
זרימות מכל המקורות.
הסבריו כיצד ניתן לפתור בעיה זו ע"י רדוקציה לביעית הזרימה
המקסימלית.

(א) קזקז



האנו ב-A כה
הכ. הרכז צורה
הגביה-A-B-C
ל-A-B-C
(N-G-A-B)
ג-A-B-C

אנו יקח גאות גודלה $\frac{1}{2} t - t$?
יהי s_1, s_2, s_3 גורמי $\frac{1}{2}$ ובודהן נס. גורמי מה...?

Backtracking to knapsack problem

שיהור תת-הקבוצה

for $i = n, \dots, 1$

if $\text{Knap}[i, b] \neq \text{Knap}[i-1, b]$

print i

$b \leftarrow b - w_i$

סיבוכיות: $\Theta(n)$

אנו מודים, כי גודל סעפין נייר און-ליין

Sub set sum

knapsack → subset sum

- פירון רעכתי:
- רגנון ערך עלי, הקבוצה
- פירון קבוצה צדדי. וקיים הערך?

נקוט:

- $\text{Sum}(i, s)$ יגיד אם ניתן ל千古 סכום s עם תח קבוצה של i האברים הראשונים
- $s = a_1$ תנאי עצירה $i=1$ יחזיר כן אם s
- $\text{Sum}(i, s) = \text{sum}(i-1, s) \text{ or } \text{sum}(i-1, s-a_i)$
- המטרה לחשב $\text{sum}(n, s)$



נדיר מערך דו-מימדי $\text{Sum}[0..n][0..S]$

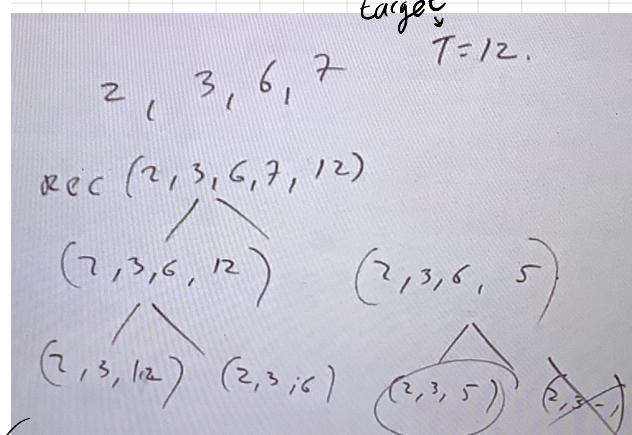
// Initialize

```
if  $t < 0$        $\text{Sum}[i, t] \leftarrow \text{FALSE}$ 
for  $t = 1, \dots, S$        $\text{Sum}[0, t] \leftarrow \text{FALSE}$ 
for  $i = 0, \dots, n$        $\text{Sum}[i, 0] \leftarrow \text{TRUE}$ 
```

// Main

```
for  $i = 1, \dots, n$ 
    for  $t = 1, \dots, S$ 
         $\text{Sum}[i, t] \leftarrow \text{Sum}[i-1, t] \text{ or } \text{Sum}[i-1, t-a_i]$ 
            // where  $\text{Sum}(i, \text{NegativeValue}) = \text{FALSE}$ 
```

שיטות
בבבובון
knapsack
algorithm



2 3 6 7 12												
2	+	-	v	-	-	-	-	-	-	-	-	-
2(3)	+ -	v -	v -	v -	v -	v -	v -	v -	v -	v -	v -	v -
2, 3, 6	+ - v	v - v	v - v	v - v	v - v	v - v	v - v	v - v	v - v	v - v	v - v	v - v
2, 3, 6, 7	+ - v v	v - v v	v - v v	v - v v	v - v v	v - v v	v - v v	v - v v	v - v v	v - v v	v - v v	v - v v

complexity: $O(ns)$ // מביא צורך זמן S ו n רק מוסף גודל סעפין n ו S .

שיהור תת-הקבוצה

complexity: $O(nS)$

```
for  $i = n, \dots, 1$ 
    if not  $\text{Sum}[i-1, S]$ 
        print  $a_i$ 
         $S \leftarrow S - a_i$ 
```

סיבוכיות: $\Theta(n)$

$T[1, \dots, n]$ כוכב. $C \setminus P$
 $P[1, \dots, m]$ גען.

$\Sigma = \{a, b, c, \dots, z\}$ ו- $\Sigma = \{0, 1\}$ lens, P-T ו- T-P סימן

$T = ababaca$, $P = aba$

אלגוריתם נאיבי

```
Search(char P[], char T[], int m, int n)
for i ← 1, ..., n - m + 1 do
    if P[1..m] = T[i..i+m - 1] then
        print "Pattern occurs with shift" i
```

פעולה ההשוואה $P[1..m] = T[i..i+m - 1]$ מתחבצעת כך:

```
for j ← 1, ..., m do
    if P[j] ≠ T[i+j - 1] then
        return false
return true
```

learning-made-fun ד"ר דוד שטנר

יעילות: $O((n - m)m) = O(nm)$

סימונים מקובלים בקשר למילים/מחרוזות

Σ – אלף-בית: קבוצה סופית של אותיות.

x.w – שרשור של המילה x אחרי המילה w.

a
n – קיצור למילה

ε – המילה הריקה.

| w | = האורך של המילה w.

Σ^i – קבוצת כל המילים באורך i שבנויות מאותיות הא"ב.

$$\bigcup_{i=1}^{\infty} \Sigma^i = \Sigma^+$$

$$\bigcup_{i=0}^{\infty} \Sigma^i = \Sigma^*$$

ד"ר דוד שטנר

אוטומט סופי - תיאור לא פורמלי

- אוטומט הוא מודל לסוג מסוים של אלגוריתמים על מהרווזת שנוח ליציג כדיagram מצבים.
- מתייחסים לאוטומט גם כ"מכונה מופשטת".

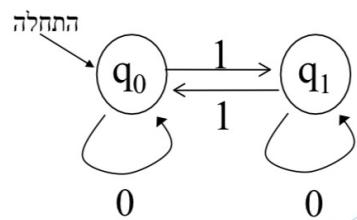
אוטומט סופי דטרמיניסטי

(DFA) Deterministic Finite Automaton

הוא מכונה המתוארת ע"י החמשייה הסדרה: $(Q, q_0, F, \Sigma, \delta)$ כאשר:

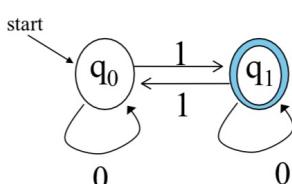
- Σ – א"ב
- Q – קבוצה סופית של מצבים
- q_0 – מצב תחيلي ($q_0 \in Q$)

(accepting states $F \subseteq Q$)
 קבוצת מצבים סופיים F – קבוצת מצבים סופיים $F \subseteq Q$ final (נקראים גם "מוזהים")
 פונקציית מעברים $\delta: Q \times \Sigma \rightarrow Q$ – פונקציית מעברים (המצב הבא = (תו, מצב הנוכחי) δ)



דוגמה

ייצוג גרפי:



דוגמה 1

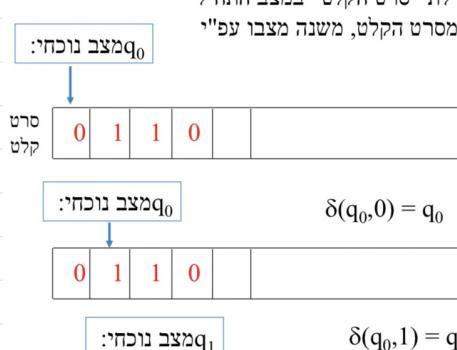
יצוג טקסטואלי:

- $\Sigma = \{0, 1\}$
- $Q = \{q_0, q_1\}$
- $F = \{q_1\}$
- פונקציית המעבר δ :

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

מצב התחלתי: האוטומט נמצא בתחילת "סרט הקלט" במצב התחלתי
 גענד חישוב: האוטומט קורא אות מסרט הקלט, משנה מצבו עפ"י
 ומתקדם לאות הבאה

הчисוב של אוטומט



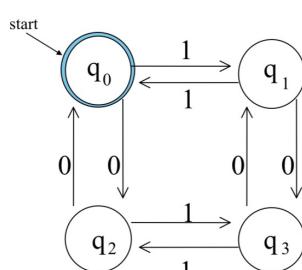
הגדרת עזר - הפונקציה $\hat{\delta}$

$$\hat{\delta}(q_0, w) = \begin{cases} q_0 & w \text{ contains even no. of 1's} \\ q_1 & w \text{ contains odd no. of 1's} \end{cases}$$

גדר פונקציה: $\hat{\delta}: Q \times \Sigma^* \rightarrow Q$
 כך שלכל $w \in \Sigma^*$, $w \in \Sigma^*$ ו- $\hat{\delta}(q, w)$ מבטא את המצב אליו יגיע האוטומט אם הוא מתחילה וקורא את המילה w במאזן

דוגמה 2

לנפדי כ' ליל' ס' ו' פה ז' ז' ז' ז'



הפונקציה תוגדר באופן אינדוקטיבי:
 $\hat{\delta}(q, \varepsilon) = q$ – בסיס: המילה הריקה

צעד: תהי $w \in \Sigma^*$ ותהי $a \in \Sigma$

$$q' = \hat{\delta}(q, w)$$

$$\hat{\delta}(q, wa) = \delta(q', a)$$

ויהי

או נגיד:

איך מוכחים שאוטומט מצהה שפה?

1. מגדירים את תפקידי המ מצבים:
לכל מצב מופיעים את קבוצת המילים אשר מגיעה אליו.
2. מוכיחים שמלים מסוימות במצב הנכון (באינדוקציה על אורך המילים):
 - בסיס – הוכחת נכונות למילה הריקה (היות ש- $q_0 = q_0(\epsilon) = \hat{\delta}(q_0)$, עליינו להראות ש- q_0 הוא המצב הנכון למילה ϵ).
 - צעדי – לכל $a \in \Sigma$, בהנחה ש- w מביאה את האוטומט במצב המתאים עבורה, יש להוכיח שהגדרת פונקציית המ מצבים גורמת למילה wa לעבורו במצב הרצוי עבורה.
3. מוכיחים את נכונות המ מצבים המוזהום:
יש להראות שהמלים השיכות לשפה L הן בבדיקה המילימית למצבים F .

הנשאלה היא $k \geq 0$ כך ש- $a^k \in L$

היא $k > 0$ קבוע כלשהו, והוא $\{a\}^k$.

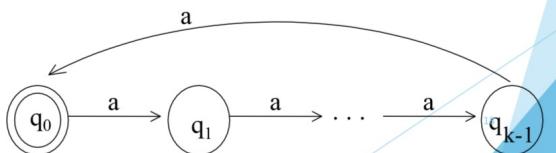
ובנה אוטומט לזיהוי קבוצת המחרוזות a^k עבורן $0 \leq k$.

בנייה: לאוטומט יהיו k מצבים, $Q = \{q_0, \dots, q_{k-1}\}$

. $|w| \mod k = i$ יהיה q_i אם ורק אם $i = \hat{\delta}(q_0, w)$

מצב מזהה: q_0 .

פונקציית המעברים: $\delta(q_i, a) = q_{(i+1) \mod k}$



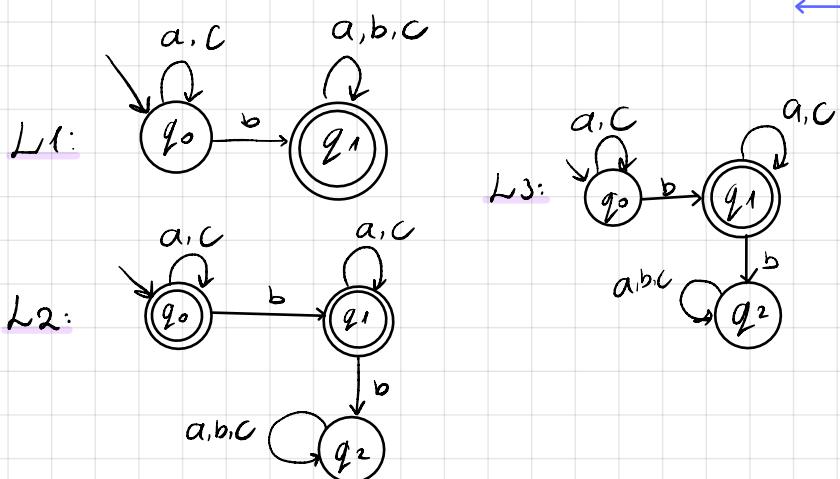
תרגיל

בנו אוטומטים לזיהוי השפות הבאות, מעל האלפבית:

L_1 : כל המילים עם b אחד לפחות

L_2 : כל המילים עם לכל היוטר b אחד

L_3 : כל המילים עם b אחד בדיקות



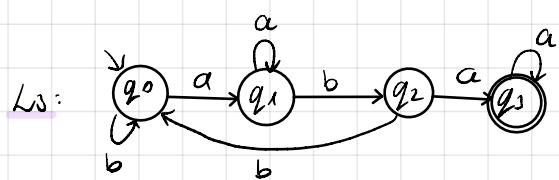
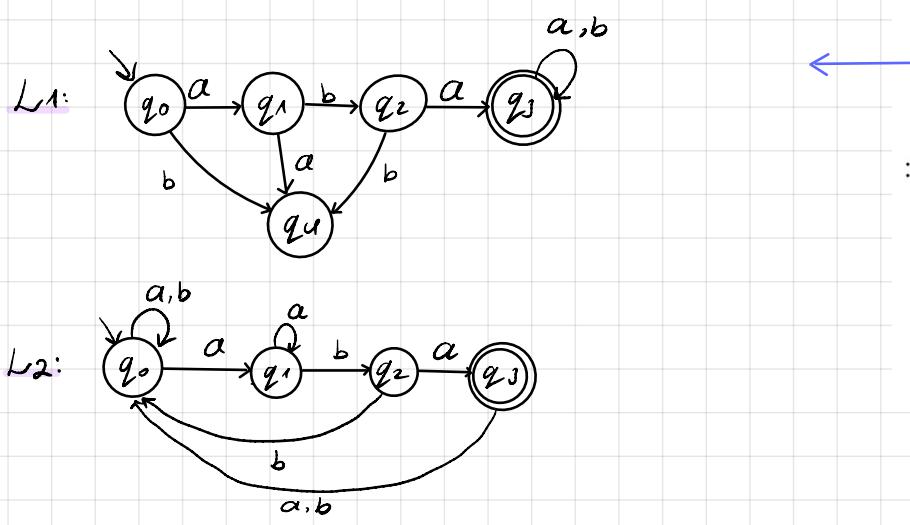
תרגיל

בנו אוטומטים לזיהוי השפות הבאות, מעל האלפבית $\{a, b\}$:

L_1 : המחרוזות המתחילות ברצף aba .

L_2 : המחרוזות המסיימות ברצף aba .

L_3 : המחרוזות המכילות את הרצף aba .



שלב ב - הרצת האוטומט על הטקסט

```
Search(char T[1..n], Transition Function δ)
s ← 0 // initial state
for i ← 1,...,n do
    s ← δ(s, T[i])
    if s ∈ F then
        print Match in position i - m + 1
```

פתרונות לבניית התאמת המחרוזות על ידי אוטומט

אלגוריתם:

שלב א: בניית אוטומט שמצווה מחרוזות שמשתיימות בתבנית P.

שלב ב: נרץ את האוטומט עם קלט T.

בכל פעם שהאוטומט מגיע לUMB מצב זהה, נזכיר על התאמת.

יעילות:

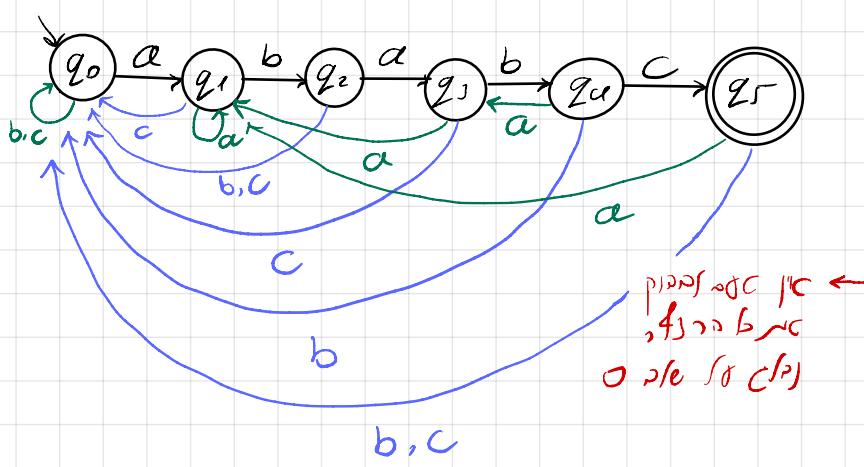
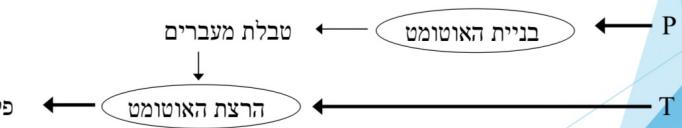
נניח שפונקציית המעברים δ מיוצגת כטבלה (מערך דו-ממדי)

\Leftarrow כל צעד לוקח זמן קבוע

\Leftarrow זמן הריצה הכלול: $\Theta(n)$

השערה: אין צורך לשמר את הטקסט [...] T בזיכרון!

לעתה נזכיר שקיים מושג $i \leq n$ כך ש $T[i:n] = CNICIC$ ו- $CNICIC$ מופיע ב T .



האוטומט יתבצע על תבנית Θ כ'
הינתן הרצף האটומטי α שאותו יתבצע.
זה עוזר לנו לארח את ה- α ולבזבז לא יותר
נתקי-הנוי כדוגמת $aabb$ או $abab$ (ולא יותר)

α
 $a b a b a c$
 α
($CNICIC$)

	a	b	c
0	1	0	0
1	1	2	0
2	3	0	0
3	1	4	0
4	3	0	5
5	1	0	0

בנין אוטומט מלבני
mismatch
נקודות נס-טראנסיזיה
הנוי
נקודות נס-טראנסיזיה
נקודות נס-טראנסיזיה

מismatch
נקודות נס-טראנסיזיה
נקודות נס-טראנסיזיה
(נקודות נס-טראנסיזיה)

אנו עושים מה?

בנייה האוטומט

הצהה:

נקודות נס-טראנסיזיה
נקודות נס-טראנסיזיה
נקודות נס-טראנסיזיה
נקודות נס-טראנסיזיה

1. הגדר $F = \{q_m\}$; $Q = \{q_0, \dots, q_m\}$

2. הגדר טבלה δ ובו שורה לכל מצב, ועמודה לכל אות בא"ב

3. מלא את שורה 0 (תרגיל)

4. לכל $m < i < n$

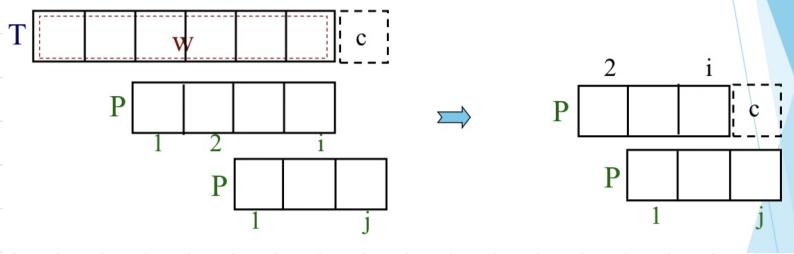
5. מלא את מעבר ה- **match** (האות הבאה תואמת לתבנית): $\delta[i, P[i+1]] \leftarrow i+1$

6. מלא את מעבר ה- **mismatch** (כל $c \in \Sigma, c \neq P[i+1]$): $\delta[i, c] \leftarrow 0$

7. מלא את $\delta[m, c]$ לכל $c \in \Sigma$

чисוב מעברים [c] סוג δ[i,c]

אבחנה:
הריישא המקסימלית $[j..i]$ P[1..i] שהוא סיפא של המחרוזת c·w, היא גם הרישא המקסימלית שהיא סיפא של המחרוזת c·P[2..i].



- $P[i+1] \neq c$ ו- $c \in \Sigma$ היפוא כאות נסיעה, לא נסיעה - *nys matell*

יעילות האלגוריתם

COMPUTE-δ(String p[1..m])

```

 $\delta[0,P[1]] \leftarrow 1$ 
for each  $c \in \Sigma$ ,  $c \neq P[1]$  do
     $\delta[0,c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i,P[i+1]] \leftarrow i+1$ 
        for each  $c \in \Sigma$ ,  $c \neq P[i+1]$  do
             $\delta[i,c] \leftarrow \text{SuffixMatch}(P, P[2..i] \cdot c)$ 
    for each  $c \in \Sigma$  do
         $\delta[m,c] \leftarrow \text{SuffixMatch}(P, P[2..m] \cdot c)$ 

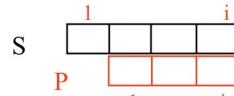
```

$$\sum_{i=1}^m |\Sigma| \cdot i^2 = |\Sigma| \cdot \sum_{i=1}^m i^2 = \Theta(|\Sigma| \cdot m^3)$$

דרוד שטח
אנו מגדירים $i = |\Sigma|$
בנוסף $m = n$

בעיית עזר

קלט: תבנית P ומחרוזת S (באורך $i \geq m$)
פלט: אורך הרישא המקסימלית של P המופיעה כסיפא של S.



פתרון נאייבי:

SuffixMatch (P,S)

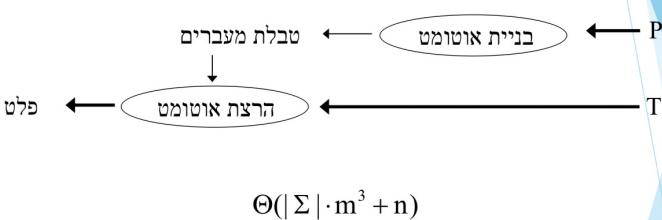
```

for  $j \leftarrow i$  downto 1 do
    if  $P[1..j]$  is a suffix of S then
        return j
    return 0

```

מה ייעילות?

יעילות הפתרון לבעיית התאמת המחרוזות



COMPUTE-δ(String p[1..m])

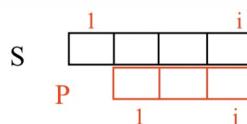
```

 $\delta[0,P[1]] \leftarrow 1$ 
for each  $c \in \Sigma$ ,  $c \neq P[1]$  do
     $\delta[0,c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i,P[i+1]] \leftarrow i+1$  match
        for each  $c \in \Sigma$ ,  $c \neq P[i+1]$  do
             $\delta[i,c] \leftarrow \text{SuffixMatch}(P, P[2..i] \cdot c)$ 
    for each  $c \in \Sigma$  do
         $\delta[m,c] \leftarrow \text{SuffixMatch}(P, P[2..m] \cdot c)$ 

```

פתרון יעיל יותר לבעיית העזר

($i \geq m$) (באורך S ומחרוזת P **קלט:** תבנית המופיעה כסיפא של P **פלט:** אורך הרישא המקסימלית של S.)



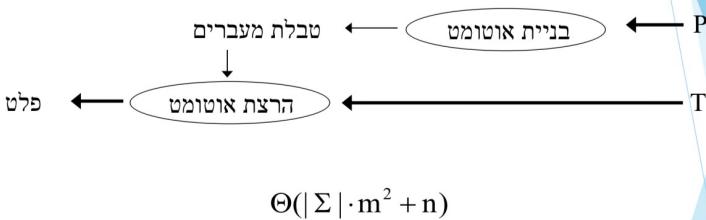
תזכורת:

באוטומט **שלנו**, עבור $m \leq j \leq n$, $\hat{\delta}(q_0, w) = q_j$ אם ורק אם w הוא אורך הרישא המקסימלית של P המופיעה כסיפא של w .

מסקנה: אם נריץ את האוטומט שבנו על המילה S, אזי המצב אליו הגיע האוטומט $\hat{\delta}(q_0, S)$ יהיה q_j אם ורק אם

יהי בדיקת הערך j המבוקש. -
המבחן נזק בפער זה, ואנו צריכים לו גמיש.

כגון נציגו בסעיפים,



```
COMPUTE-δ(String p[1...m])
 $\delta[0,P[1]] \leftarrow 1$ 
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0,c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i,P[i+1]] \leftarrow i+1$ 
    for each  $c \in \Sigma, c \neq P[i+1]$  do
         $\delta[i,c] \leftarrow \text{RUN}(\delta, P[2\dots i]\cdot c)$ 
for each  $c \in \Sigma$  do
     $\delta[m,c] \leftarrow \text{RUN}(\delta, P[2\dots m]\cdot c)$ 
```

```
RUN(Transition Function δ, char S[])
 $j \leftarrow 0$  // initial state
for  $i \leftarrow 1, \dots, \text{len}(S)$  do
     $j \leftarrow \delta[j, S[i]]$ 
return  $j$ 
```

יעילות האלגוריתם

```
COMPUTE-δ(String p[1...m])
 $\delta[0,P[1]] \leftarrow 1$ 
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0,c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i,P[i+1]] \leftarrow i+1$ 
    for each  $c \in \Sigma, c \neq P[i+1]$  do
         $\delta[i,c] \leftarrow \text{RUN}(\delta, P[2\dots i]\cdot c)$ 
for each  $c \in \Sigma$  do
     $\delta[m,c] \leftarrow \text{RUN}(\delta, P[2\dots m]\cdot c)$ 
```

$$\sum_{i=1}^m |\Sigma| \cdot i = |\Sigma| \cdot \sum_{i=1}^m i = \Theta(|\Sigma| \cdot m^2)$$

שיפור היעילות (2)

```
COMPUTE-δ(String p[1...m])
 $\delta[0,P[1]] \leftarrow 1$ 
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0,c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i,P[i+1]] \leftarrow i+1$ 
    state  $\leftarrow \text{RUN}(\delta, P[2\dots i])$ 
    for each  $c \in \Sigma, c \neq P[i+1]$  do
         $\delta[i,c] \leftarrow \delta[\text{state}, c]$ 
    state  $\leftarrow \text{RUN}(\delta, P[2\dots m])$ 
for each  $c \in \Sigma$  do
     $\delta[m,c] \leftarrow \delta[\text{state}, c]$ 
```

צעד שני:
העברה מיידית מאיטרציה
לאיטרציה הבאיה

שיפור היעילות (1)

```
COMPUTE-δ(String p[1...m])
 $\delta[0,P[1]] \leftarrow 1$ 
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0,c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i,P[i+1]] \leftarrow i+1$ 
    for each  $c \in \Sigma, c \neq P[i+1]$  do
         $\delta[i,c] \leftarrow \text{RUN}(\delta, P[2\dots i]\cdot c)$ 
for each  $c \in \Sigma$  do
     $\delta[m,c] \leftarrow \text{RUN}(\delta, P[2\dots m]\cdot c)$ 
```

צעד ראשון:
הוצאת חישובים חוזרים
(איננו ריאנטים)
אל מחוץ ללולאה הפנימית.

שיפור היעילות (2)

```
COMPUTE-δ(String p[1...m])  $\leftarrow \text{נוסף}$ 
 $\delta[0,P[1]] \leftarrow 1$  ✓
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0,c] \leftarrow 0$  ✓
    state  $\leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i,P[i+1]] \leftarrow i+1$  ✓
    for each  $c \in \Sigma, c \neq P[i+1]$  do
         $\delta[i,c] \leftarrow \delta[\text{state}, c]$  ✓
        state  $\leftarrow \delta[\text{state}, P[i+1]]
for each  $c \in \Sigma$  do
     $\delta[m,c] \leftarrow \delta[\text{state}, c]$$ 
```

זמן הריצעה:
 $\sum_{i=1}^m |\Sigma| = \Theta(|\Sigma| \cdot m)$

שיפור היעילות (1)

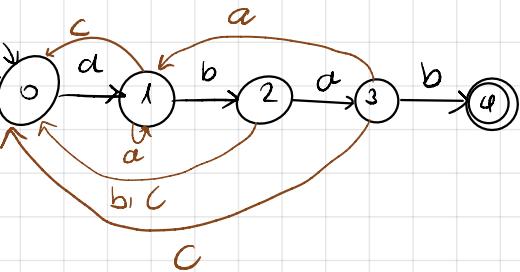
```
COMPUTE-δ(String p[1...m])
 $\delta[0,P[1]] \leftarrow 1$ 
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0,c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i,P[i+1]] \leftarrow i+1$ 
    state  $\leftarrow \text{RUN}(\delta, P[2\dots i])$ 
    for each  $c \in \Sigma, c \neq P[i+1]$  do
         $\delta[i,c] \leftarrow \delta[\text{state}, c]$ 
    state  $\leftarrow \text{RUN}(\delta, P[2\dots m])$ 
for each  $c \in \Sigma$  do
     $\delta[m,c] \leftarrow \delta[\text{state}, c]$ 
```

$$\begin{aligned} \sum_{i=1}^m (i + |\Sigma|) &= \\ &= \sum_{i=1}^m i + \sum_{i=1}^m |\Sigma| = \\ &= \Theta(m^2 + m |\Sigma|) \end{aligned}$$

	a	b	c	
0	1	0	0	
1	1	2	0	
2	3	0	0	
3	1	4	0	

1 2 3 4
abab

state = 0
 $i=1$
State = 0
 $i=2$
state = 1
 $i=3$
state = 2



!חישוב!

state i
 $f(P_2 \dots P_i)$ = ?

לפניכם:
אנו מודים לך
הרה.

כתבו אלגוריתםיעיל שמקבל:

- קובוצת תווים Σ
- פרמטר m
- מטריצה בגודל $|\Sigma| \times (m+1)$

ונחזר כפלט האם המטריצה היא אוטומט סופי ליזיהו מחרחת,
ואם כן מהי המחרחת.

$P \sim e^i P$

ריבוקן סוד עוזה ופזיק צפוי שתקבצנו. +. סוף רוחה.

ולכן, $e^i P$ סופי $\Rightarrow P$.

הוכן סוד, הגדיר קפנאר בפזק ג-ו וו.

2. ריבוקן אוטומט סופי $\sim P$
ונריבוקן, או חן את Σ .

כך נובע סוד ישר. בפזק ג-ו נזקן מוקדם מההסגרה.

$\checkmark \in \Sigma$

ולזה לאו ג-ו P הריבוקן, וריבוקן סדרה סדרה וריבוקן סדרה זוג זוג.

ולא יותר שמה הוא, כי גיאוגרפיה וקרטוגרפיה

KMP[†]

אלגוריתם Knuth, Morris, Pratt

הREFERENCE: (נאום קנתו בכנס ICACM 1980)

לעשות עבודה חיליקת בשלב בנית האוטומט ולהשלים את החישוב בזמן השימוש בו.

זמן הבניה:

נשמר בטבלה state[1,..., m] את ערכי המשנה state (מהגרסא הקודמת)

$\delta(q_0, P[2..i]) = \text{ערך של המשנה state בתחילת האיטרציה ה- } i = \text{state}[i]$

כשציריך את δ :

נקרא לפונקציה שמחשבת את $\delta(i, c)$ במקומות לשולף ערך זה מטבלה. ← נאום שפה נטול המבוקש מהתוצאות ה- Compute.

מבחן KMP נערך בהצלחה והוא תקין. COMPUTE / Compute ↢ תקין.

שימוש הפונקציה δ // נאום קנתו דיסרטציית שחקן קומס כ-Compute.

回忆: הכללים לחישוב פונקציית המעברים באיטרציה ה- i (למעט $i=0$)

$$\delta[i, p[i+1]] \leftarrow i+1$$

$$\delta[i, c] \leftarrow \delta[\text{state}, c] \quad c \neq p[i+1]$$

שלב הרצת האוטומט

```
Search(char T[1..n], Transition Function  $\delta$ )
s  $\leftarrow$  0      // initial state
for i  $\leftarrow$  1,...,n do
    s  $\leftarrow$   $\delta(s, T[i])$ 
    if s = m then
        print Match in position i - m + 1
```

$\delta(\text{State } i, \text{ Character } c)$

if $c = p[i+1]$ **then**
return $i + 1$

else if $i = 0$ **then**
return 0

else
return $\delta(\text{state}[i], c)$

// Transition Function

ההגשה בדעת הגנטיקה מושג יפה. Failure: ראיון ש-state נחוץ רק במקרה של אי-התאמה (failure).

(2)

$\sum_{i=1}^m |\Sigma|$

יעילות אלגוריתם Search

נשתמש בכלל חשבונו הבנק:

אין עליה על סכום ההפניות "אם ידוע שהחישבו תמיד א-שלילי, או סכום המשיכיות"

נסוג הפעולות של δ לשני סוגים:

1. הפעולות שעוצרות את הרקורסיה (אין מבצעות הפעלה רקורסיבית נוספת):

- כל הפעלה מקורית של δ מגיעה בדיקות פעם אחת להפעלה שעוצרת, ומכאן

שזה"כ ישנן הפעולות כנ"ל

- כל הפעלה שעוצרת יכולה להגדיל את i (המצב של האוטומט), לפחות ב- 1

state[i] ↣ state[i+1] ↣ ... ↣ state[m]

2. הפעולות שימושיות את הרקורסיה:

- כל הפעלה רקורסיבית מקטינה את המצב.

ה מצב תמיד א-שלילי וסכום ה"הפניות" (הגדלות המצבים) אינו עולה על n

↔ מספר המניות (הקטנות המצבים = מספר הקריאות הרקורסיביות)

אינו עולה על n. לסיום: זמן הריצה הוא $\Theta(n)$.

SetState(String p[1..m]) // Fill state array.

```
state[1]  $\leftarrow$  0
for i  $\leftarrow$  1,...,m-1 do
    state[i+1]  $\leftarrow$   $\delta(\text{state}[i], p[i+1])$ 
```

יעילות של δ

- מספר הקריאות לפונקציה δ מותך SetState הוא m (אורך התבנית P).
- מספר הקריאות הרקורסיביות חסום ע"י מספר הקריאות הלא-רקורסיביות

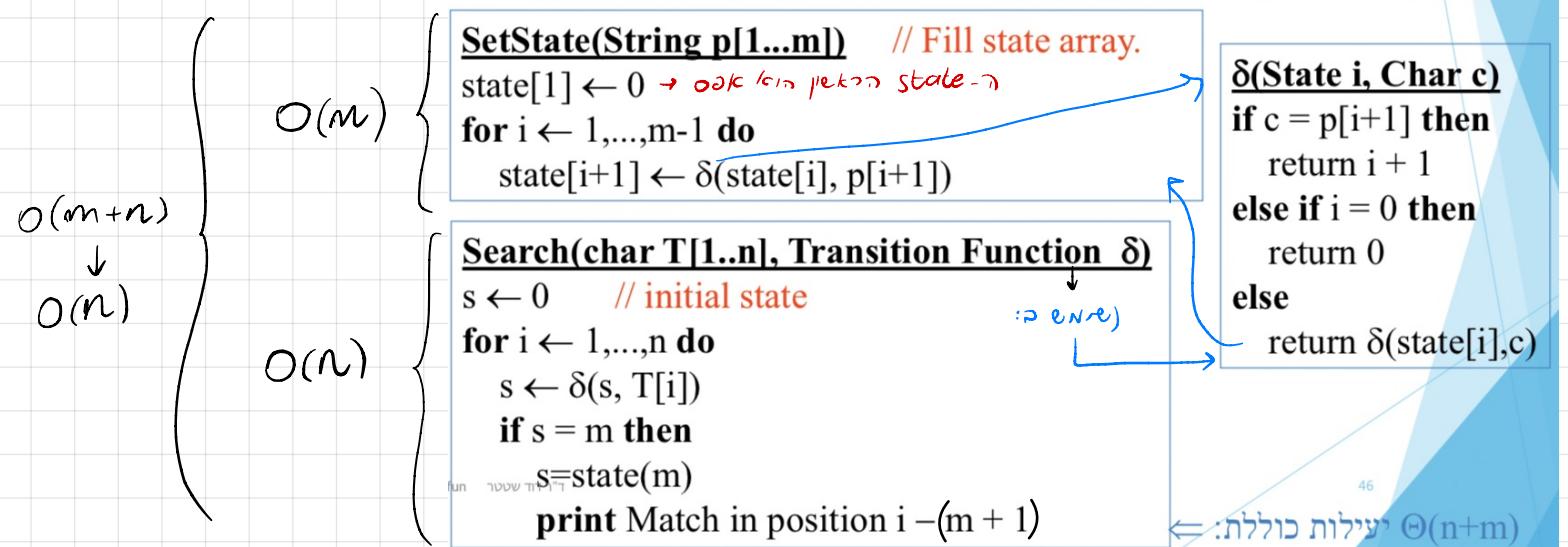
(אותו טיענון).

זמן הריצה הוא $\Theta(m)$.

אלגוריתם :KMP

קלט: מבנה P וטקסט T

1. נבנה את המערך state על ידי הפונקציה `SetState`.
2. נרץ את האוטומט עם הtekסט T כקלט בעזרת הפונקציה `Search` ונשתמש בפונקציית המעברים δ שכתבנו.



תרגיל

בנו את מערך state עבור המחרוזת xyxyxyz מעל $\Sigma = \{x, y, z\}$

הristol על הטקסט:

xxxxxyxyxyzxxzxy

Set State:

P: $x \ y \ x \ \underline{y} \ x \ z$

T: $x \ x \ x \ y \ x \ y \ x \ y \ x \ z \ x \ z \ x \ y$

state[0 0 1 2 3 0]

כ x x תוגע לא נזון שם
 בזיניכ, קמיהו סוף השיטות
 ז נסבון ————— אונטז אונטז / 0

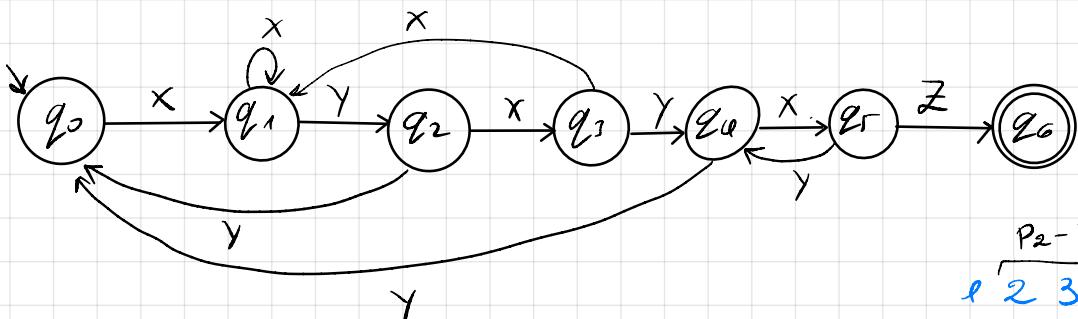
\downarrow
 $x \ y \ x \ y \ z$ setstate(0,y): $i=1$ state = 0 \rightarrow 0 0
 ?
 מחרוזת יתגלה
 יתגלה.

setstate(0,x): $i=2$ state = 0 \rightarrow 0 0 1
 ?
 מחרוזת יתגלה
 יתגלה.

setstate(0,y): $i=3$ state = 0 \rightarrow 0 0 1 2
 ?
 מחרוזת יתגלה
 יתגלה.

Setstate(0, x): $i=4$ state = 0 \rightarrow 0 0 1 2 3
 ? אם נקבע x מוקדם
 $2+1=3$ יופיע!

Setstate(0, z): $i=5$ state = 0 \rightarrow 0 0 1 2 3 0
 ? אם נקבע z מוקדם
 0 יופיע!



$P_2 - P_4$
 1 2 3 4 5 6
 0 0 1 (2) 3 0

$$\int_i = \int(0, P_2 \dots P_i) \rightarrow \int_4 = \int(0, P_2 - P_4) = 2$$

$\int_5 = 3$
 $\int_6 = 0$

P: x y x y x z
 1 1 1 2 3 4 5 6 1 0 1 2
 T: x x x y x y x y x z x z x y

Search:

$i=1, s=1$
 $i=2, s=11$
 $i=3, s=11$
 $i=4, s=12$
 $i=5, s=23$
 $i=6, s=34$
 $i=7, s=45$
 $i=8, s=54$
 $i=9, s=45$
 $i=10, s=56$

pattern \rightarrow מ' י' כ' ג' נ' כ' פ' נ'
 מ' ק' א' ג'
 $O(5 + 15)$? י' כ' ג' נ' כ' פ' נ'
 מ' ק' א' ג'
 ה' מ' ה' נ' כ' פ' נ'

שאלות מבחינות

state

הנפקת state. מתי state מוחזק ומתי state מוחזק?

אתם שוארים מוחזק מוחזק?

למה מוחזק מוחזק?

שאלה. איך האמינו שמיין גורין יפסיד?

א. נתונה מחרוזת P שאורכה m . נתון כי טבלת KMP נראית כך (משמאלי לימין):

0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	$m-1$	m	$m+1$	$m+2$	$m+3$	$m+4$

כלומר 0 בלבד. איזה תנאי מספיק והכרחי מקיימת המחרוזת P ?

$\nwarrow abcbabcbcb \swarrow$ מזוהה.

ב. נתונה מחרוזת P שאורכה m . נתון כי טבלת KMP נראית כך (משמאלי לימין):

0	0	1	2	3	4	5	$m-1$	m	$m+1$	$m+2$	$m+3$
0	0	1	2	0	0	1	0	0	1	2	0	0

איזה תנאי מספיק והכרחי מקיימת המחרוזת P ?

נקודות נסכימה ואנחנו יכולים לומר קיימת נסכמה?

שאלה שאלת
הישר קאנון

כתבו אלגוריתםיעיל שמקבל:

קובוצת תווים S

פרמטר m

מטריצה בגודל $|S| * (m+1)$

ומחזיר כפלט האם המטריצה היא אוטומט סופי לזיהוי מחרוזת,
ואם כן מהי המחרוזת.

* לא ניתן P מ S נסכימה.
הרכב יsb קי S נסכימה
הקיים והפוך
זיהוי נסכימה

? מה נסכימה כיו + נסכמה?
 $P \rightarrow M$
מטריצה

a	b	c
0	1	0
1	1	2
2	3	0
3	1	2
4	0	1
5	0	1
6	0	1

אקוונת
זרף

אקוונת קראון

אקוונת

Arcaffe

הוֹלָנְדִית

גְּזַעַן

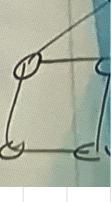
רצע אז מה העניין זהה עם להגדר

שפות

$\text{Con} = \{<G, s, t> \mid G \text{ is a graph and there is a path from } s \text{ to } t\}$

$\text{Clique} = \{<G, k> \mid G \text{ is a graph which contain a clique of size at least } k\}$

$\text{Chess} = \{s, n \mid \text{on a chess board of size } n \times n \text{ with state } s \text{ (the way all the chess pieces are arranged) white will always win the game}\}$



ה问题是 מה הכרזה - תרשים (אנו מודים לך, וזה פשוט יפה) סביר ואנכי שמי לא מודים לך ומי שמי לא מודים לך נבנה עליון נבנה עליון והוא מודם לך. R הינו היל האמירה של דס אלגוריתם הרכיבה. נסב, מתקגה R הוא גזרתנו רצף הרכיבה.

$$A(x) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases}$$

האחלה P הגדירה (יען זינאנר)

בהתאם לdefinition מוגדרות כפונקציית אינדיקטורה $M(x)$

השאלה היא האם ניתן למצוא אינדיקטורה M ומספר k כך $A = \{x \mid M(x) = 1\}$ קיימת דוגמא x כך $M(x) = 0$ ומספר k כך $M(x) = 1$ בז'רנשטיין.

שכחנו מכך ש- $M(x) = 1$ מוגדרת כפונקציית אינדיקטורה, ו- $M(x) = 0$ מוגדרת כפונקציית אינדיקטורה.

$$P = \{L \mid \text{קיימת אינדיקטורה } M \text{ כך } A = \{x \mid M(x) = 1\}\}$$

אינדיקטום קייניות

רצע אז מה זה אומר איזה שפה

משמעות יש ב P? // פירושו אינדיקטום

הפוכה - יפה

בצורה חסרה

לעתן פונקציית אינדיקטורה!

מסלול בגרף

כל המסלולים הקצרים

שידור

מין

KMP, CFS, DFS, BFS, Dijkstra, Ford-Fulkerson, זרימה ועוד

ובעצם כל אלג כמעט שאי פעם למדתם (למעט בעיית התרミילן) כ. הולן איזופרנילן.

במלים אחרות P מגדירה את כל השפות שלהם יש אלג יעיל או בערך כל אלג שלמדתם

אנו מודים לך נריה פון נירן וברוך לך נריה פון נירן!

P = כל אוסף של אינדיקטום יפה, כלומר אם אינדיקטום יפה

האחסקה NP הגדירה

לכידון

Non

האחסקה NP, סט גזיר של אוסף נקודות.

- הטענה $L \text{ נאותם } \text{כל } x \in L \text{ כפואים אזי } A(x) = 1$

כ"י אמת כפואים כ- L , סט גזיר קיימת $y \in L$ כך $A(y) = 1$.אוסף גזירים סט גזיר $x \in L$ כך $A(x) = 1$.אוסף גזירים סט גזיר $y \in L$ כך $A(y) = 0$.אוסף גזירים סט גזיר $x \notin L$ כך $A(x) = 0$.ק"י אמת כפואים כ- L אם ורק אם $\exists y \in L \text{ כך } A(y) = 1$.

$NP = \{L | \forall y \in L \text{ כך } A(y) = 1\}$
 נאותם גזירים סט גזיר
 גזיר נאותם גזירים סט גזיר.
 גזיר נאותם גזירים סט גזיר.

נה זנ� NP? (השאלה)

$$\begin{aligned} A(x, y) = 1 &\iff \exists z \in L \text{ כך } x \in z \text{ ו } y \in z \\ A(x, y) = 0 &\iff \nexists z \in L \text{ כך } x \in z \text{ ו } y \in z \end{aligned}$$

* $A \vdash \neg A$
 ו- $\perp \vdash \perp$

SAT

- משתנה בוליאני:

- משתנה שערךו אמת (T) או שקר (F).

ליטרל:

- משתנה (למשל x) או שלילתו (\bar{x}).

פסוקית (הסגר) בצורת CNF:

- אוסף ליטרלים עם כמות "או" (vee) ביניהם. למשל $(\bar{x}_2 \vee x_1 \vee \bar{x}_3)$

נוסחה בצורת CNF:

- פסוקיות CNF שביניהם כמות "וגם" (and). למשל $(x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee \bar{x}_3)$

נוסחה בצורת CNF:

- נוסחה CNF שבה יש לכל היותר k ליטרלים.

בדידה מכה שנייה

- השם: גזיר טרוי.

- התאמה של ערך T או F לכל משתנה. $(x_1, x_2, x_3) = (T, F, T)$.

- השם מספקת: גזיר טרוי כריזום עטוף T.

- השמה שהצבתה למשתני הנוסחה נותנת ערך T לנוסחה.

- למשל $(T, F, T) = (\bar{x}_1, x_2, x_3) \wedge (\bar{x}_1 \vee x_2 \wedge x_3)$ בנוסחה $(\bar{x}_1 \vee x_2 \wedge x_3) \wedge (\bar{x}_1 \vee x_2 \wedge \bar{x}_3)$ היא מספקת את T.

- יש ערך T ליטרל אחד לפחות בכל פסוקית.

- נוסחה ספיקת: גזיר טרוי עטוף ערך T.

- נוסחה שקיים לה השמה מספקת.

- השפה SAT:

- השפה kSAT.

$SAT = \{\phi | \phi \text{ is a satisfiable formula in CNF form}\}$

ההשורה גזיר טרוי כריזום עטוף (1, 2, 3...)

$kSAT = \{\phi | \phi \text{ is a satisfiable formula in kCNF form}\}$

פסוק CNF גזיר כפואים כ- Φ נאותם גזירים סט גזיר

נקודות גזיר כפואים כ- Φ נאותם גזירים סט גזיר.

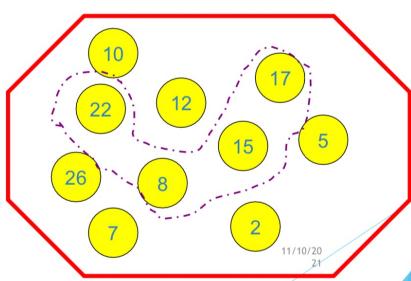
הטענה גזיר כפואים כ- Φ נאותם גזירים סט גזיר.

בעיות גণיטיות NP

בעיית החלוקה ((Partition problem)

$$\text{PAR} = \{S \mid \sum_{a_i \in S_1} a_i = \sum_{b_j \in S_2} b_j\}$$

קיים חקיקה seiomati קפואות נורא



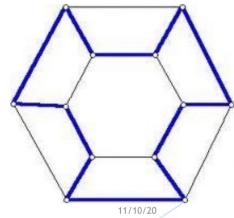
David Statter, 2019 ©

בעיית המעל ההAMILTONIAN (problem)

HAM-CYCLE = {G| קיימת מסלול סגור תחביב את כל הנקודות}



David Statter, 2019 ©



בעית הרוכב הנוסע (problem)

* מילוי תחנות

קיים מסלול סגור בין נקודות מינימלי, אולם

סבוך לשבור.



תCURON פיריאטי GEFENIM (Integer Linear Programming)

קיים מתרון גפניאם פוליאומט הינו יעיל לתCURON

דוגמיה: מפעלי צעצועים מייצר משאיות ואופניים.

את הצעצועים מייצרים שני פעולמים – אחד יזק את החלקים והשני מרכיב אותם.

כל פועל עובד לכל יותר 900 דקות ביום.

יציקת חילקי משאית דרושת 4 דקות, ואילו יציקת חלקן אופני דרושת 2 דקות.

הרכבת משאית דרושת 2 דקות, ואילו הרכבת אופני דרושת 3 דקות.

הרווח על משאית הוא 2 ורווח על אופני הוא 1 בלבד.



11/10/20
21

$$x_1 + x_2 = 1 \quad (1) \quad x_1, x_2 \in \{0,1\}$$

$$\max (2x_1 + x_2) : \quad (2)$$

$$4x_1 + x_2 \leq 900 \rightarrow x_1 = 103 \quad (3)$$

$$2x_1 + 3x_2 \leq 900 \rightarrow x_2 \leq 41.7 \quad (4)$$

הרווח הולך וגדל.
הרווח הולך וגדל.

הרווח הולך וגדל.

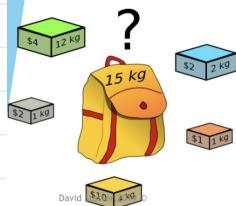
ה3מת גיאות NP כמכון פיריאטי גפניאם

בעית תרמילי הגב ((Knapsack problem)

$$\max Z = 4x_1 + 2x_2 + 10x_3 + x_4 + 2x_5$$

$$\text{s.t. } 12x_1 + x_2 + 4x_3 + x_4 + 2x_5 \leq 15$$

$$x_i \in \{0,1\} \quad 1 \leq i \leq 5$$



11/10/20
21

הגדרה co-NP

לפיה: $\exists x \in L \rightarrow \forall y : A(x,y) = 1$
 $\exists x \notin L \rightarrow \forall y : A(x,y) = 0$

$$\text{co-NP} = \{L \mid \bar{L} \in NP\}$$

כפיה:

$$P \subseteq NP \iff P = NP \iff NP \subseteq \text{co-NP} \iff P \subseteq \text{co-NP}$$

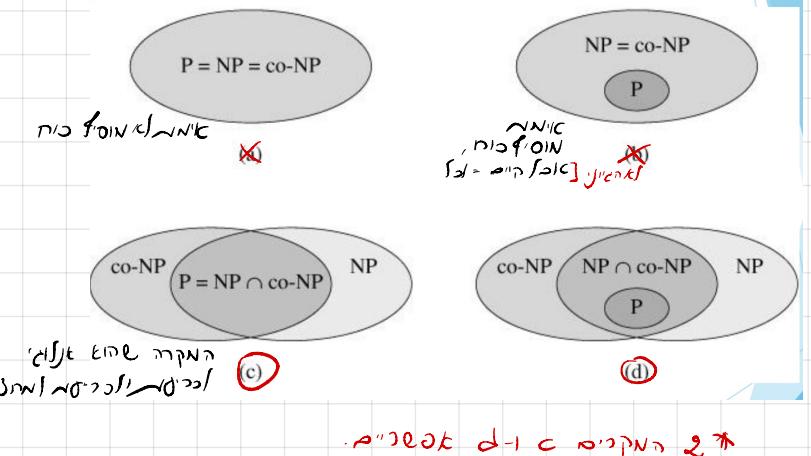
ראיתם שRE שונה coRE כלומר אין סגירות למשלים

באופן דומה NP לא בהכרח סגורה למשלים ומכאן coNP

גאומטרית הגדירנו P כsubset של NP כלומר P כולל כל איבר של NP אשר מקיים P .

ולוון ניקח A ובודק האם $A \subseteq P$ או $A \subseteq \bar{P}$ בואז מהו A ?

יחסים דפכליים בין הגדרות



יחסים זה מוסבר (זה השורה שלי בפיו)

$$P \subseteq NP$$

$$P \subseteq \text{coNP}$$

$$P \subseteq NP \cap \text{coNP}$$

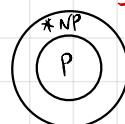
לא ברור אם יש שוויון

לפיה אין אלג הכרעה פולינומי אלא רק אלג איניות

לcoNP אין גם את זה (ניתן רק לאמתות שמיליה היא לא בשפה)

$$E(L \in NP) \rightarrow L \in P - 1 \quad P \subseteq NP \iff P \neq NP \quad \text{לפיה}$$

ולא זו ערך נסוך (פונקציית הסתברות)



במקרה (b) נוכיח ש $P \subseteq NP$

במקרה (a) $P = NP \subseteq \text{co-NP}$

מכחה $P \subseteq NP$

* $\exists A : L \in P \wedge \forall y : A(x,y) = 1 \wedge \forall y : A(x,y) = 0 \wedge \forall y : A(x,y) = 1 \wedge \forall y : A(x,y) = 0$

* * $\exists A : \forall y : A(x,y) = 1 \wedge \forall y : A(x,y) = 0$

** $\exists A : \forall y : A(x,y) = 1 \wedge \forall y : A(x,y) = 0$

$$\left\{ \begin{array}{l} \exists A : \forall y : A(x,y) = 1 \\ \exists A : \forall y : A(x,y) = 0 \end{array} \right. \rightarrow L \in P$$

$\forall y : A'(x,y) = A(x) = 0 \quad \text{לפיה } x \notin L \quad \text{ט}$

מכחה $P \subseteq \text{coNP}$

$\exists A : \forall y : A(x,y) = 1 \quad \text{לפיה } x \in L$

$\exists A : \forall y : A(x,y) = 0 \quad \text{לפיה } x \notin L$

$$\left\{ \begin{array}{l} \exists A : \forall y : A(x,y) = 1 \\ \exists A : \forall y : A(x,y) = 0 \end{array} \right. \rightarrow P \subseteq \text{coNP}$$

$L \in P \rightarrow L \in NP \rightarrow L \in \text{co-NP}$

$L \in P \rightarrow L \in \text{co-NP} \rightarrow L \in NP \rightarrow L \in P$

הוכחה:

הנושאים הקיימים בפער נספחים ל- Σ נספחים $\Sigma_1 \cup \Sigma_2$ ו- $\Sigma_1 \cap \Sigma_2$

$L_1 \in NP$, $L_2 \in NP \rightarrow L_1 \cup L_2 \in NP$

$A_1(x) = 1$, $x \in L_1$ אם A_1 מוגדר כמו שבסעיף P - S אז L_1
 $A_1(x) = 0$, $x \notin L_1$

: L_2 מוגדר A_2 מ- P

$A_2(x) = 1$, $x \in L_2$

$A_2(x) = 0$, $x \notin L_2$

אם $x \in L_1 \cup L_2$ אז $A(x) = A_1(x)$ או $A_2(x)$
 $\lambda = \text{מקרה}$

: $L_1 \cup L_2$ מוגדר A כ

אם $x \in A$ פירושו $x \in L_1$ או $x \in L_2$ או $x \notin L_1 \cup L_2$ מוגדר A כ

$\underbrace{\text{מקרה}}_{\text{מקרה}} P$, $\text{מקרה } 1$ מוגדר כ

אנו נוכיח $L_1 \cup L_2 \in NP$

$L_1 \cup L_2 \in NP \leftarrow L_1 \in NP$
 $L_2 \in NP$

$\exists y A_i(x, y) = 1 \leftarrow x \in L_i$ מוגדר y כמו, מוגדר כמו i מ- $L_i \in NP$
 $\forall y A_i(x, y) = 0 \leftarrow x \notin L_i$

A מוגדר כ

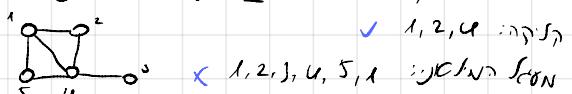
מקרה 1: $y = y_1, y_2$ מוגדר y כמו y_1, y_2 מוגדר y כמו

$\int A_1(x, y_1)$ מוגדר y_1 כמו y_2 מוגדר y_2 כמו

$A_2(x, y_2)$ מוגדר y_2 כמו

y_1, y_2 מוגדר y כמו

מקרה 2: $y = y_1, y_2, \dots, y_n$ מוגדר y כמו



✓ 1, 2, 4 מוגדר

✗ 1, 2, 3, 4, 5, 1 מוגדר

✗ 1, 2, 3, 4, 5, 1 מוגדר

מקרה 3: $y = y_1, y_2, \dots, y_n$ מוגדר y כמו

P מוגדר y כמו y_1, y_2, \dots, y_n מוגדר y כמו

P מוגדר y כמו y_1, y_2, \dots, y_n מוגדר y כמו

$A_1(x) = 1$, $x \in L_1$ אם A_1 מוגדר כמו שבסעיף P - S אז L_1
 $A_1(x) = 0$, $x \notin L_1$

: L_2 מוגדר A_2 מ- P

$A_2(x) = 1$, $x \in L_2$

$A_2(x) = 0$, $x \notin L_2$

$L_1 \cup L_2$ מוגדר A כ

$A(x_0, \dots, x_i) \wedge A(x_{i+1}, \dots, x_n)$ מוגדר $i = 1, \dots, n - 1$ מ-
 (זה יתאפשר רק אם $i < n$)

יפסונס סון וטפל בעקבות נושא זה, מושג אחד הוא שפה NP

$$L_1 \cup L_2 \in NP \leftarrow \begin{array}{l} L_1 \in NP \\ L_2 \in NP \end{array}$$

$\exists y A_i(x, y) = 1 \leftarrow x \in L_i$ נגיד יש y שקיים, נגיד יש x שקיים $\forall y A_i(x, y) = 0 \leftarrow x \notin L_i$

A סביר ל \exists
 y_1, y_2, i גורם ל y
לפניהם פועל אוסף של $A_1(x_0, \dots, x_i, y_1)$
לפניהם פועל אוסף של $A_2(x_{i+1}, \dots, x_n, y_2)$
לפניהם פועל אוסף של $A_3(x_0, \dots, x_n, y_3)$

...ensesto P

נגיד A' סביר ל \forall
 $A(x) = 1 \leftarrow x \in L$

$A(x) = 0 \leftarrow x \notin L$

L סביר ל \forall A' כפוף ל $A'(x) = \bar{A}(x)$

$\bar{A}(x) = 0 \leftarrow A(x) = 1 \leftarrow x \in L \leftarrow x \notin \bar{L}$

$\bar{A}(x) = 1 \leftarrow A(x) = 0 \leftarrow x \notin L \leftarrow x \in \bar{L}$

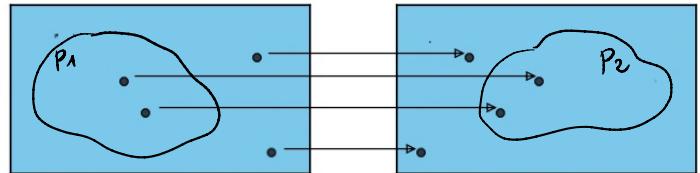
$NP = C_0 NP \leftarrow \text{ensesto } NP \text{ ו } P \neq NP$

P: סיבוב של פונקציית פולינומיאלית, כלומר $f(x) = P$
NP: סיבוב של פונקציית פולינומיאלית הולכת וגדלה.

$L \not\models P$, $L \models NP$: $f(x)$, $P \neq NP$: $f(x)$

רואים כ נושא P1 רשות / רוכוק נא הפוגה P2.

אנו קוראים לפונקציה $f(x)$ בז' חן ו- x הוא גנטיקון (P_1) או גן זם (P_2) ו- $f(x)$ מתקבל על-



ו-הו, היפוך: נושא נושא A שפוגה B.

בגדי צורה,

מה הוכחת הנכונות?

שים לב אין צורך להוכיח שום דבר על הטענה והפתרון של הטענה שאליה עושים רוכוקציה

דוגמה

הטענה

- שידוע A בהינתן מערך מספרים, ברצוננו למצוא את המינימלי מבין המספרים. נתון אלג' לממציא מספר מקסימלי במערך מספרים בוצרה עיליה

פתרון בעזרת רוכוקציה

- זההו הטענה אליה מbezיעים רוכוקציה: ידוע לנו SCNINT לפתרון בעיה אחרת, והוא בעית מציאת מקסימום בין מספרים במערך. הרוכוקציה היא בעיה "מציאת המינימלי" ("הרצוי") לבעיה ("מציאת המקסימלי" ("המצוי"))

- תרגום הקלט: נכפיל את כל המספרים במערך ב: 1

- נתמוך את הטענה של מציאת המקסימלי (לא ידוע מה האלגוריתם שיפטור אותה וייתכן (שקיים כמה אלגוריתמים שפותרים את הטענה

- תרגום הפלט: את הערך שקיבלו נכפיל ב: 1

- הוכחת נכונות: כאן יש להוכיח ש

```
def change_list(listy):
    return [num*-1 for num in listy]
def find_max(listy):
    return max(listy)
def find_min(listy):
    return find_max(change_list(listy))*-1
```

C נושא נושא סוף

ニアנו, נושא

סוף נושא נושא

A → B

בנוסף

ולאם חזקה

A ← max

B ← min

C ← max

D ← min

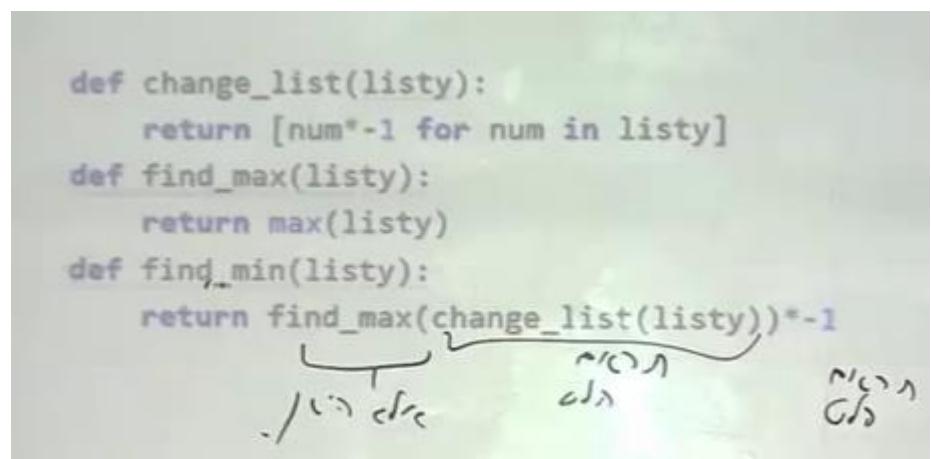
הגדכנו את C בתור כל הביעות שאנו יכולים לפתור, כלומר אלגוריתמים שיש להם פתרון פולינומי (A בחזקת 6 נגד N) ← לפי הגדירה
 ***קליקה** : חלק בגרף שיש בו חיבור בין כל הקשתות (דרך מלא/שלם)

הגדירה – רדוקציה:

פתרון רדוקטיבי נכון מרכיב מ: //עובר בין 2 בעיות ← לדוגמה מביעת המקסימום לביעת המינימום, וקיים 3 שלבים: תרגום קלט, תרגום פלט, והוכחת נכונות

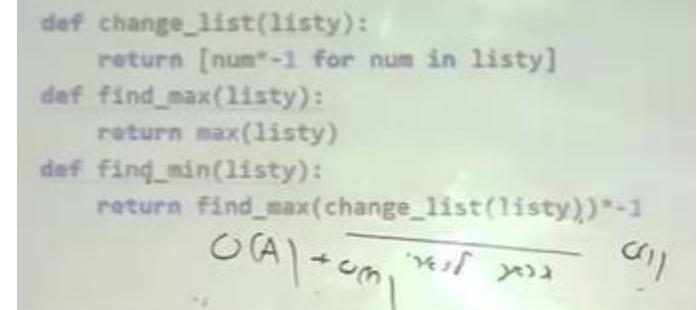
- זיהוי הבעיה אליה מוצאים רדוקציה.
- תרגום של הקלט הנוכחי לקלט הבעיה שאליה מוצאים רדוקציה.
- תרגום הפלט של הבעיה שאליה אנו מוצאים רדוקציה לפלט לביעת הנთונה.
- הוכחת נכונות תוך הסתמכות על נכונותו של פתרון בעיה שאליה ביצעונו רדוקציה.
- הוכחת נכונות צריכה להתמקד בנסיבות התרגומים.
- ניתוח סיבוכיות תוך הסתמכות על סיבוכיותה הידועה של פתרון בעיה אליה ביצעונו רדוקציה.

כל רדוקציה תכיל את שלושת האלמנטים האלה, ורק אותם.

חשוב מאוד:

הפתרון הרדוקטיבי משתמש בפתרון הרגיל אליו בוצעה הרדוקציה בלי להתייחס לפרטים ("קופסה שחורה")

קופסה שחורה:
 אין לנו מושג איך פונקציות מסוימות עובדות, אבל הן עובדות. לדוגמה ← `findmax` של פיתון

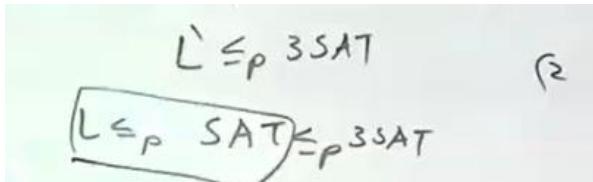
זמן ריצה:

הערה: פולינום + פולינום = פולינום

从 3CNF 转为 CNF 和 CNF 转为 3CNF

$$(x_1 \vee y_1) \wedge (\bar{y}_1 \vee \bar{x}_2 \vee y_2) \wedge (\bar{y}_2 \vee x_3)$$

\leq_{SAT} CNF3 \leftarrow בכל פסוקית ישן 3 ליטרלים.
היא שולמה:



1. SAT שיכת ל-NP.
2. היא קשה. (מכל שפה אפשר לתרגם אותה ל-3SAT)

לדוגמא, בהינתן פסוק F בצורה CNF, מופע של SAT נבנה (פונקציה הרדוקציה) פסוק 'F' בצורה CNF3 ומופע של SAT ב- $F = (\exists x_1 \wedge \exists x_2 \wedge \exists x_3) \exists y_1 \exists y_2 \exists y_3 [y_1 \wedge y_2 \wedge y_3 \rightarrow F]$

3SAT ∈ NPC

הגדירה:

הגדרה: פסוק בצורה 3-CNF הוא פסוק בצורה CNF שכל פסוקית בו מכילה בדיקת שלושה ליטרלים.

{לפסוק Φ בצורה 3-CNF קיימת השמה מספקת $\{\Phi = 3SAT\}$

נכונה להראות שאפשר לתרגם CNF בצורה פולינומית $CNF \xrightarrow{P} 3CNF$ $\rightarrow 3SAT$.

* נקבע סוג של קוד שמקובל לפסוקית ארכוּה מיד'
ופסוקית קצרה מיד', //פסוקית באורך כלשהו
ששמור על $T = F \vee T$
�יחזיר ביטוי שבו בכל פסוקית יש שלשה,
שהוא ספיק רק אם

טענה: $3SAT \in NPC$

הוכחה:

1. $3SAT \in NP$

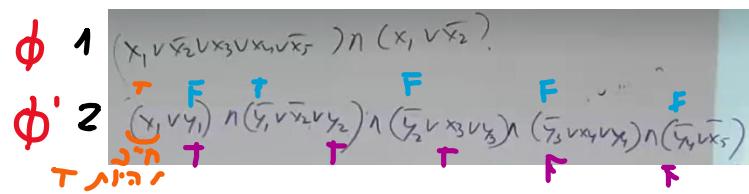
2. $SAT \leq_P 3SAT$

SAT $\leq_P 3SAT$ הוכחה?

לוקחים פסוקית ארכוּה (לדוגמא באורך 5) ומוסיפים לה 1-a משתנים חדשים.
מה האנותואיציה? במקור היה אחד ספיק (באורך 0),
از אם נוציא 1-a פסוקיות זה יסתדר.

איך נבצע?

1. ניקח לדוגמא פסוק שיש בו 4 ע"י.
כל אחד מהם יופיע פעם אחת לחוב ופעם אחת לשילוח.



2. מה証據 להראות?
אם השורה השנייה ספיקה אז גם השורה הראשונה, והההפר.
CTION 1:

נשים \heartsuit כי כל ע' יכול לעזור לי רק פעם אחת. כך שכל ע' חדש (שזהו True) יכול לטפל בפסוקית אחת – מה"כ 4.

CTION 2:
אם לפחות אחד מה א'ים הוא True, אז נספיק פסוקית אחת
ואת השאר מסדר עם הע'ים.

בהינתן פסוק Φ בצורה CNF מופע של SAT נבנה (פונקציה

הרדוקציה) פסוק ' Φ ' בצורה 3CNF מופע של 3SAT:

עבור כל פסוקית ב- Φ $m > 3$ $C = (l_1 \vee l_2 \vee \dots \vee l_m) \wedge$

$y_1, y_2, \dots, y_{m-1}, y_m$ לבניה ת פסוקיות ב- Φ ונוסיף פסוקים אוטומיים
 $C' = (l_1 \vee y_1) \wedge$
 $(\sim y_1 \vee l_2 \vee y_2) \wedge$
 $(\sim y_2 \vee l_3 \vee y_3) \wedge$
 \dots
 $(\sim y_{k-1} \vee l_k \vee y_k) \wedge$
 \dots
 $(\sim y_{m-2} \vee l_{m-1} \vee y_{m-1}) \wedge$
 $(\sim y_{m-1} \vee l_m)$

CTION 1:

בב' לא סביר כי כל ע' גדול מ-4 לא בלאו כי לא בלאו כי לא בלאו

CTION 2:
לא אבד פתרון

$$(x_1 \vee x_2 \vee \bar{x}_2) \quad x \vee x = x$$

בשביל הפסוקית הקצרה: $(\bar{x}_2 \vee x_1) \wedge (\bar{x}_1 \vee x_2)$

סיבוכיות:

- פסוקית קצרה
- **תרגומם :** $|Q| \geq 3 \Rightarrow Q \text{ להגיא מנוסחה כללית ל } 3CNF$

↳ מסקנה של מעבר SAT ל-3SAT: 3SAT שיר NPC לאינטואיטיבית, אם יודעים לפתור 3SAT אז יודעים לפתור על בעיה ב-NP * הערה: 2SAT היה בעיה קלה, קל להציג אותה בצורה לינארית.

קבוצה בלתי תלויה: קבוצה קודקודים שאין בינם קשר שמחברת ביניהם.

IS הגדרה

זו שיר ל-NP. יש תת קבוצה של קודקודים בגודל K, ניתן לבדוק בזמן K^2 שאין אף צלע בקבוצה בלתי תלויה. = **קליקה ב-PN**

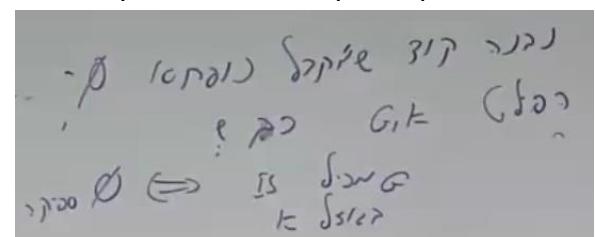
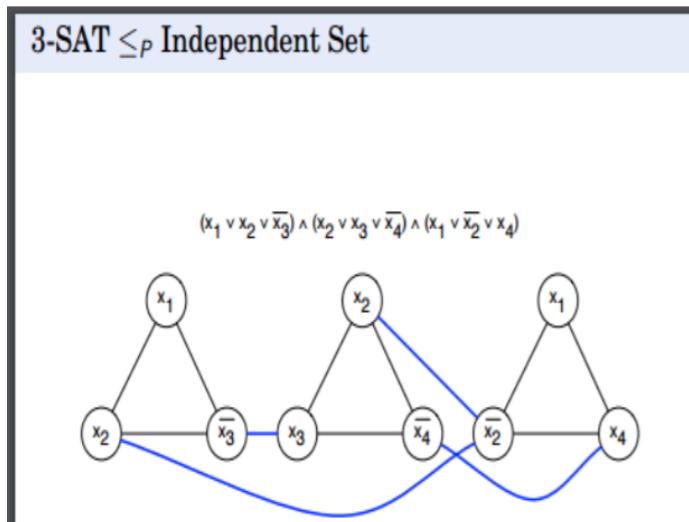
IS – קבוצה של קודקודים שנוגעת בכל קשר.

- $= IS \text{ קבוצה ב}"\tau \text{ של קודקודים, כלומר אוסף של קודקודים ללא קשרות ביניהם}$

- $IS = \{(G, k) : G \text{ is a graph with } IS \text{ of size } k\}$

זו שיר ל-PN, צ"ל זו קשה

↳ יצור קוד שמקבל נוסחה ובונה גרף



- הוכחה:
1. כל פסוקית תהיה משולש ועושים OR בינם
 2. עושים AND בין כל פסוק
- *שלילה = F, אחרת T.
ונוטן השמה מספקת לפי לוגיקה.

נוסחה היא ספיקה אם בכל פסוקית יש לפחות ליטרל אחד מסופק לפחות ליטרל אחד נבחר בה

הבנייה לא מאפשרת לבחור יותר מלייטרל אחד בכל פסוקית או משתנה ושלילתו כעת, אם הנוסחה ספיקה – ניתן לבחור ליטרל אחד מכל פסוקית ומכאן זו בגודל K

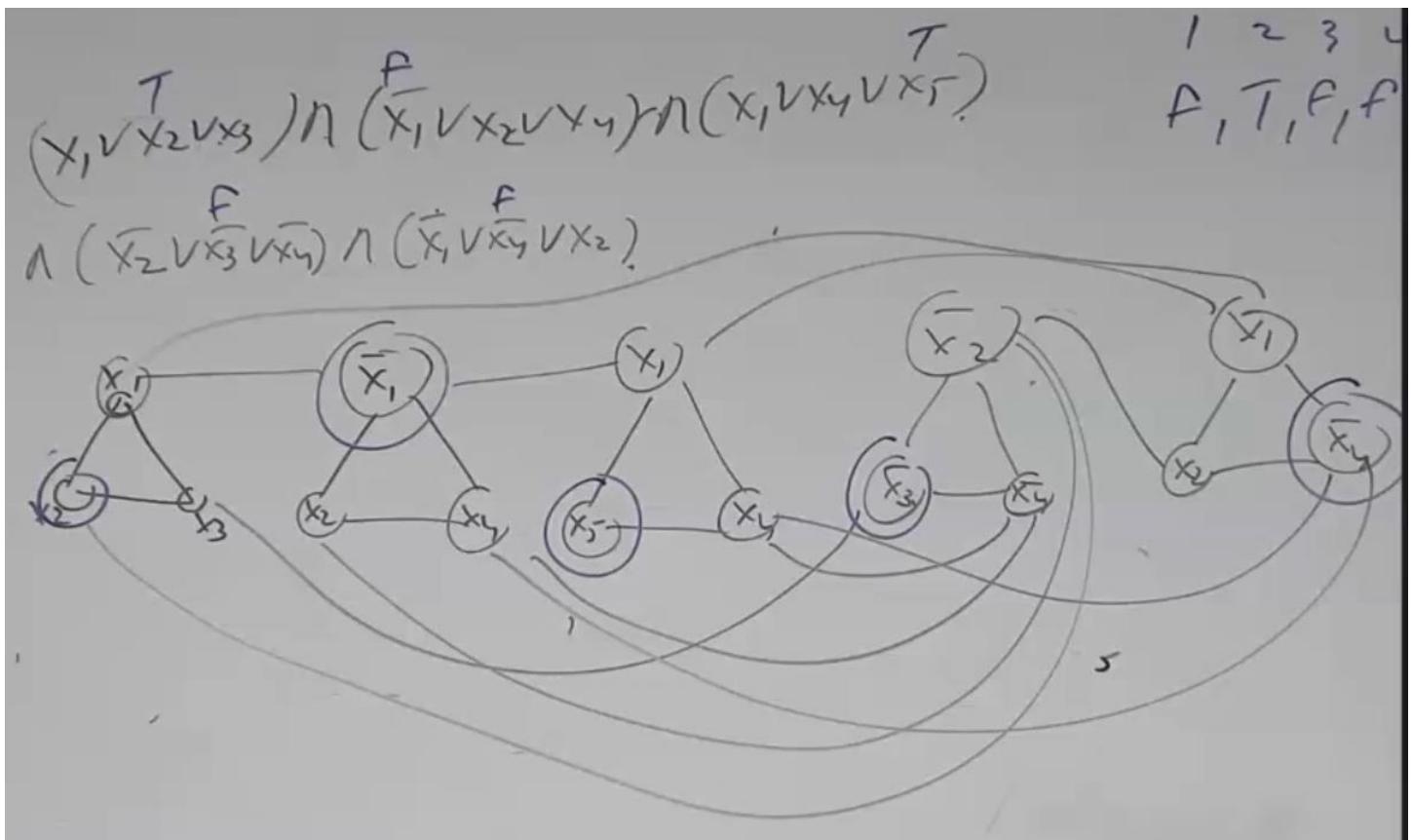
תרגום:

קילט Q כל פסוקית תהה משולש עם שמות קודקודים לפי ליטרלים בנוסף נבחר כל x עם NOT x

הבנייה פולינומית, ונוטן K = מספר הפסוקים (מספר המשולשים).

↳ אם Q ספיקה, לכל פסוקית x יש jx שמספק אותה. נבחר אותו במשולש ה-x.

ובפרט זה לא ליטרל ושלילתו, ולכן יש K קודקודיים בלתי תלויים.
 ↵ אם יש K קודקודיים בלתי תלויים, אז יש אחד בכל משולש והם תמיד לא ליטרל ושלילתו.
 הקודקו שבסהר במשולש ה- i מספק פסוקית \neg
 תרגום SAT $\neg \leftrightarrow$ IS



הערה: IS זה לא בהכרח קליקה

*יש דרך לתרגם IS ל-SAT.

גם קליקה היא NP שלמה: // NP שלמה = NPC

1. ציר להראות שהוא NP – ההוכחה הקודמת ^ שרצה בזמן K^2
2. ציר להראות שהוא קשה – ציר רדוקציה של שפה אחרת קשה אליה

נתרגם את השאלה: האם הגרף ב- K ? שואלים האם יש IS בגודל K ? ונתרגם אותו לגרף אחר שם נשאל האם יש קליקה בגודל K . אם כן, הקליקה היא NPC.
 נרצה להראות רדוקציה \leftarrow IS, קליקה.

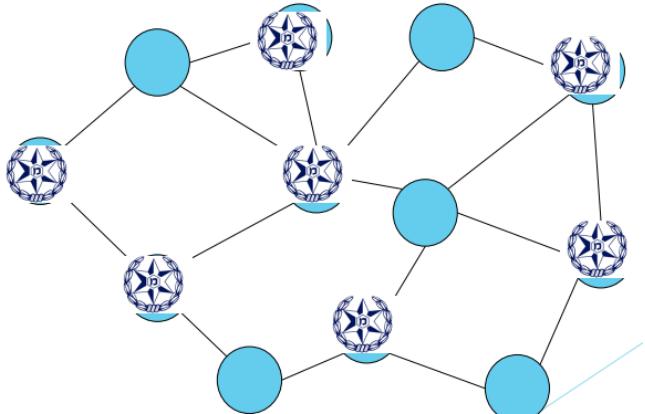
הקלט: גרף (G, K) ושאלה האם G הוא IS בגודל K ?
 ↵ בניית G transpose colum: הקשת (y, x) שייכת ל- G אם ורק אם הקשת (y, x) לא שייכת G NOT.
 * הבנייה פולינומית.

↳ נשאל האם יש קליקה (G, K) .
 ↵ אם הם A איז IS ב- G , אז A היא קליקה ב- G NOT ולהיפך.
 אם (y, x) שייך ל- A NOT (y, x) שייך ל- E NOT ב- G NOT.

אבחן: הרדוקציה היא זו כיווניות/הופכית. כאמור – אותו טיעון יראה קליקה \leftarrow IS.

cliques \leq_p IS.

בעיית CISIO קודקודים , VC – Vertex Cover



נתונה רשת כבישים.
יש לפזר תחנות מטריה בצלמי הרשת, כך שכל כביש "יכוסה" ע"י תחנה אחת לפחות.
מהו מספר התחנות המינימלי הנחוץ?

הגדרה VC:
CISIO בקודקודים V הוא CISIO אם בכל קשת ב G יש לפחות קצה אחד ב-V
 $VC = \{(G, k) : \text{there is a } VC \text{ in } G \text{ of size } k\}$

* אימוט: $\forall e \in E \exists x, y \in A \text{ such that } e = x \cup y \text{ // מעבר פעם אחת על כל הקשתות } \leftarrow \text{לינארית}$

нерצה להראות: $VC \in NPC$

$\leftarrow \text{נדגים רדוקציה מס-} IS \text{ ל-} VC // IS \leq_p VC$

קלט: (G, K) , האם יש IS בגודל K

פלט: (G', K') , האם יש VC בגודל K

אבחן: אם קבוצה A היא VC אז כל הקודקודים בקבוצה A הם IS.
 $\leftarrow \text{למה? אם A היא קבוצה בלתי תלוייה A = } y \cup x, \text{ אין קשת } y \cup x \text{ לא-VC.}$

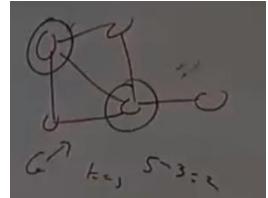
כל קשת בגרף, קצה אחד שלה הוא לא A.

$\leftarrow \text{נניח ש A קבוצה בלתי תלוייה, לכל } x \cup y \text{ אין ביןיהם קשת (לפי ההגדרה של קבוצה בלתי תלוייה)}$

כל קשת בגרף A,B או ש, או ש, או שניהם – לא-IS.

במקרה שלנו, אותו אחד יהיה ב-VC. // לפחות אחת הקצוות בIS לא אדומה, כי אם אין IS הן בלתי תלויות, כלומר לא בAO.
אם ניקח את כל הקודקודים האדומים שלו אז בטוח ניגע בכל קשת – לא לפי הגדרה

רדוקצייה: על קלט $(G, n-K) \leftarrow (G, n)$
 אם יש VC בגודל $n-k$, יש IS בגודל C.



רדווקציות

$$\text{LENP} \leq_p SAT \leq_p 3SAT \leq_p IS \leq_p \text{clique} \leq_p IS \leq_p VC$$

ירא זו

$VC \rightarrow \text{clique}$

אם למשל נרצה להוכיח CISIO קליקה,
 אז נעשו $VC \leftarrow IS \leftarrow VC$

1. למה: אם $X \setminus A$ הוא VC אז X הוא SI (או קלילקה בG)
2. למה: X מקסימלי כלומר לא קיים קודקוד י' שמחובר ל- $X \in Z$.

רדוֹקצִיה עצמִית (או שפּות שׂמְסֶפְּקוּת אֶת עַצְמֵם)

3 סוג בעיות:

1. בעיתת הכרעה \leftarrow האם יש קלילקה בגודל K (קלילקה זה באמצעות גמ"ל, אבל זאת שאלת CI/IA)
 2. בעיתת מקסימום \leftarrow מה גודל הקלילקה הכי גדול?
 3. בעיתת optimum \leftarrow מצא את הקלילקה הכי גדולה
- *SAT: קפיצה מ(1) ל(3), וההפר.

\leftarrow נניח שהקיים אלגוריתם A פולינומי שיעד להכריע את קלילקה //מקבל גרפ' הגדל K וידע להגיד אם יש שם קלילקה בגודל K
 בנו אלגוריתם B שМОציא את הקלילקה המקסימלית //בדיקה כמו שעשינו ב-SAT, רק עכשו בклילקה

\leftarrow אפשר לדעת מה גודל הקלילקה הכלולות.

רדוֹקצִיה עצמִית ל-Clique

שלב 1: לדעת מה גודל הקלילקה

שלב 2: למציאת הקלילקה

1. נלקח את הגרף , נעשה איטרציה על כל הקודקודים
 ומוחק קודקוד

2. אז שואלים את האלגוריתם האם יש עדין קלילקה בגודל הנכון
 אם כן, הקודקוד הזה יורד \leftarrow לא צריך אותו.

\leftarrow היחידים שיכולים להיות בклילקה הם לא השכנים שלו

לכן אפשר להרוג גם את השכנים של אותו קודקוד
 אם לא, מחזירים את הקודקוד

► a. מציאת גודל ה-clique

► b. מציאת ה-clique

► 1. עבור כל $V \subseteq V$ בצע:

► G' \leftarrow G ללא הצומת v \leftarrow G'

► 1.2 אם $A(G',k)=1$ אז $G \leftarrow G'$

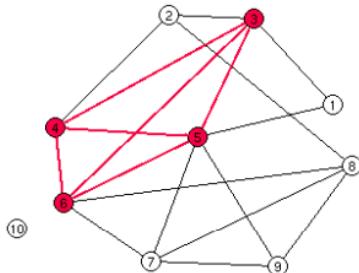
דוגמה: מעגל המילטון, VC, SI, קלילקה subsetsum, תרミיל ,

3col ?

► אבל לעיתים לא

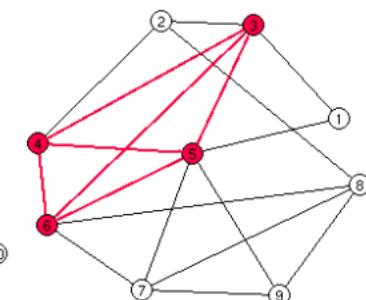
► TSP , 3SAT , צביע,

כזקזיה אצנית סגנון קפ' ג"ת קפ'יקת



נתון אלגוריתם A המכריע את בעיית ה-קליקה.
מציג אלגוריתם פולינומי B העושה שימוש ב-A ומוצא את הקליקה הגדולה ביותר

כזקזיה אצנית סגנון קפ'יקת



שלב א' – מציאת גודלו של הקבוצה:

$$k \leftarrow 1, a \leftarrow 1, b \leftarrow |V|.1$$

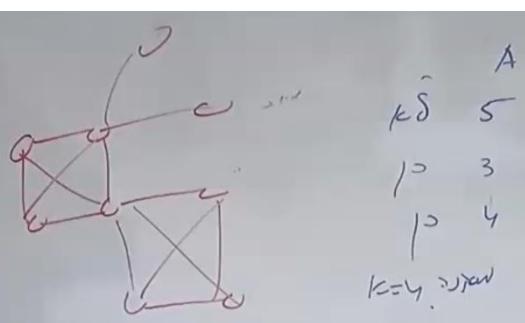
2. כל עוד $b > a$, בצע:

$$k \leftarrow \lfloor (a+b)/2 \rfloor$$

2.2 אם $A(G,k)=1$ אז $a \leftarrow k+1$

2.3 אחרת $b \leftarrow k-1$

3. החזר את k



מופיעים רק כמספרים קודקודים.

← שונה מהרדוקציה שראינו לפני שעה
שימוש אחר במנגנון הרדוקציה:
שמור באותה פונקציה הרבה פעמים

$$\log(n) * T(A) \leftarrow 1$$

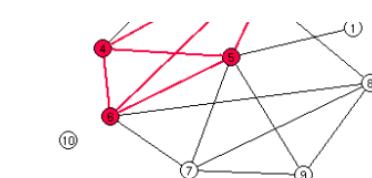
שימו לב שאם
 $A(G',k)=0$ אז v
בhcrcח בקליקה ואפשר
להמשיך רק עם
השכנים שלו

שלב ב' – מציאת תת-הgraf האופטימלי:

1. עבור כל $V \subseteq V$ בצע:

1.1 $G' \leftarrow G$ ללא הצלמת v

1.2 $G \leftarrow G - A(G',k)=1$



אם שואלים: הראו רדוקציה עצמית ל-SAT:

- ← בהינתן אלגוריתם K(G) A שאומר כן אם יש VC בגודל K:
 1. מצא K ע"י חיפוש בינארי $\min_{A(G)}$
 2. נמחק קודקוד V ונרץ (K-1, G)
 אם חזר כן אז VC, אחרת נחזיר את V וنمשייר.

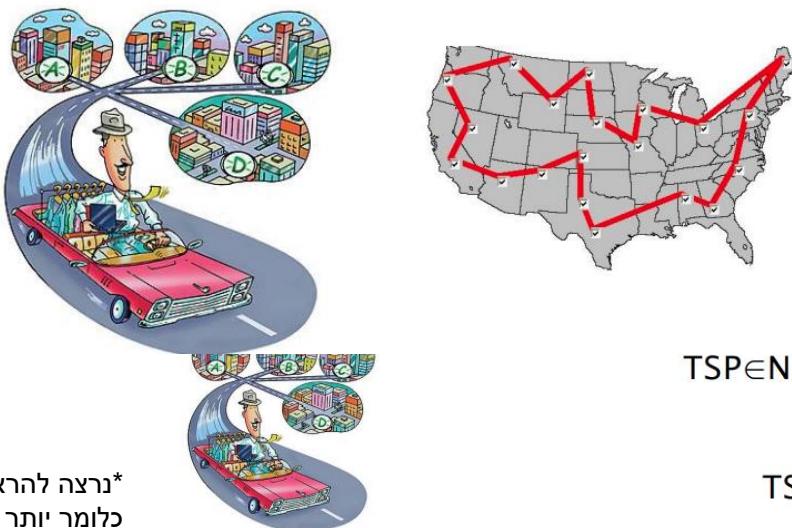
***רדוקציה עצמית: מעבר מבעיית ההכרעה לבעיית האופטימום**

← מעגל המילוטני זה מסלול שעובר בכל הקודקודים ולא עבר באותו קודקוד פעמיים
 \leftarrow קודם כל המשפט אומר שהוא-BP, "הגד" צריך לתת מסלול ← כולם, אפשר לאמת מסלול המילוטני.
קיים קוד A שאומר כן אם יש מעגל המילוטני. בנו קוד שאומר כן אם יש מסלול המילוטני.
 1. $\text{Hamilton Path} \in \text{NP}$
 ← רדוקציה עצמית: למה מעגל המילוטני

בעיית הסוכן הנוסע (Traveling salesman problem)

$$\Gamma_{\text{TSP}} = \{(G, c, k) \mid \text{קיים מסלולHamilton path ב-} G \text{ עם עלות} \leq k\}$$

קלט: גרפ' שלם, עם פונקציית משקל על הצלעות ומספר K
פלט: האם יש מעגל המילוטני בעלות קטינה/שווה K



טענה: $\text{TSP} \in \text{NPC}$

הוכחה:

$\text{TSP} \in \text{NP}$. 1

$\text{HAM} \leq_p \text{TSP}$. 2

בhinintן גרפ' $G = (V, E)$ מופיע של HAM נבנה גרפ' משוקל ' G'
 ועריך מופיע של TSP:

$$C(u, v) = \begin{cases} 0 & (u, v) \in E \\ 1 & (u, v) \notin E \end{cases} \quad \text{עם פונקציית משקל } G' = (V, E'), E' = V \times V$$

כמו כן נקבע $k=0$.

← איך נתרגם את הבעיה? קודם כל צורך להוסיף את כל הצלעות לקילקה (של המילוטן) כדי שהיא גרפ' שלם.

המשקל על הצלעות החדשות יהיה 1, ועל הישנות 0.
 נשאל: האם יש TSP במשקל 0.
 אם כן – יש מעגל המילוטן בגרף המקורי. ← גורר שיש מסלול עם משקל 0 בגרף החדש.

טענה: $\text{G} \in \text{HAM} \leftrightarrow (\text{G}', c, k) \in \text{TSP}$
 הוכחה: (בשיעור)

הראו ש $TSP \leq_p NPC$

רדוקציה: $Ham \leq_p TSP$

קלט גראף G עם שאלת האם ה- G יש מעגל במיליטוני.

פלט קליקה עם פונקציית משקל $0 = (y, x)$ אם ורק אם ב- G קיימן (y, x) . אחרת – $w(x, y) = 1 - K=0$.

↳ אם יש ב- G מעגל המיליטוני אז ב- G' יש מעגל במיליטוני.
 ↳ אם יש ב- G' מעגל במיליטוני אז יש ב- G מעגל המיליטוני.

הראו TSP מספקת את עצמה

נ裏ץ אלגוריתם (A, W, K). לכל צלע e נגדיר $w(e) = w(e) + 1$ – תוסיף 1 לאחת הצלעות

אם התשובה עדין כן, לא צריך אותה וنمשייר.

אחרת – נחזר את W ונמשיר.

הוכחה: $G, k: \{x_1, x_2, x_3\} \rightarrow \{y_1, y_2, y_3\}$

$$C \stackrel{p}{\leq} C_2$$

$$(x_1 \vee x_2 \vee x_3) \wedge (y_1 \wedge y_2)$$

$$G, k$$

$$\neg x_1 \wedge (y_1 \wedge y_2)$$

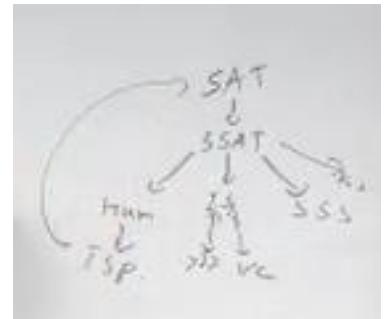
$x_1 = T$	$x_2 = F$	$x_3 = F$
$y_1 = T$	$y_2 = F$	$y_3 = F$

קליקה: גרפ שלם בגודל K

כך: קבוצת קודקודים בלתי תלויה (שאין ביניהם קשר)

C: ה כיסוי בקודקודים V הוא כיסוי אם בכל קשר ב G יש לפחות קצה אחד ב- V

עם קוק לין יש מסלול מ- TSP ל- SAT
בעיית הכרעה מספיקה כדי למצוא בעית אופטימום



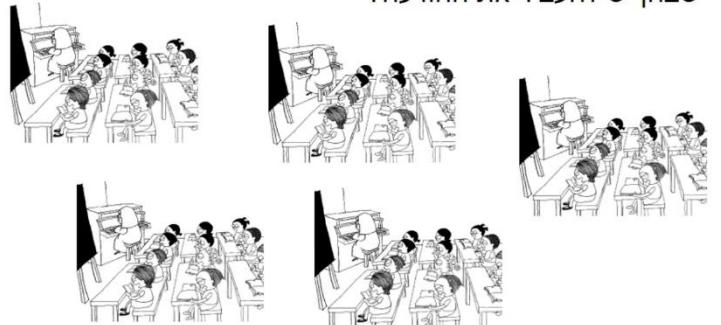
יש אוסף קבוצות: S_1 מכיל את d, e
 S_2 מכיל את h, i, e וכו'

S.. כיתות, אוטיות.. סטודנטים
השאלה היא: האם אפשר להיכנס ל-3 כיתות לפגוש את כל הסטודנטים? (נגיד ויש 7 כיתות)
← אקספוננציאלי, NP

כאיית כיסוי קבוצות

SC – Set Cover

במכללה ח סטודנטים ו- m כיתות (לאו דואק אזרות).
יש להעביר הודעה לכל הסטודנטים. מה מספר הכיתות המינימלי
שbane יש להעביר את הודעה?



כאיית כיסוי קבוצות

SC – Set Cover

הגדרה: נתונה קבוצה S ותתי קבוצות שלה S_i ($i \leq m$)

האוסף S_{jr} ($j \leq m$) יקרא כיסוי של S אם מתקיים

{ S_i ל- S כיסוי בקבוצות (S_i) בגודל k לכל היותר |(S_i, S_j , k)|}

תכלify:

$SC \in NP$

קיימת עדות ל- SC ב- NP , ניתן לבדוק שכל נתוני העולם מוכאים.

← VC הכי מזכיר את SC. אין כל נס清澈 בין כיסוי קודקודים לכיסוי קבוצות.

לכן אם נרצה להראות $SC \in NPC$, נוכל להשתמש בהוכחה של VC.

← נרצה להראות $SC \in NPC$ יותר קשה.

קלט: גרפ K עם השאלה: האם יש VC בגודל K

הרדוקציה: צריכה לתרגם את הגרף (G, K) לאוסף קבוצות. //בgraf כיסוי, קשנות עם קודקודים, בקבוצות נסחה איברים על קבוצות כל קודקוד יהיה קבוצה, וקשת יהיה איבר. //לדוגמא: הקבוצה 1 c, b, a הקבוצה 2 תהיה e a , הקבוצה 3 d, c , הקבוצה 4 d, e, f , הקבוצה 5 f .

//עכשווי למצא SC יהיה כמו VC . אם ניקח את הקבוצות 4 ו-1, אז כולם יוכסו (קבוצה 5 –).

← אנחנו יודעים VC קשה, אבל עכשווי הראנו SC פותרת אותה – כלומר היא יותר קשה. $*VC \leftarrow SAT \leftarrow \text{כל } N$

מסקנה: אי אפשר לתרגם קבוצות לגרף, אפשר לתרגם גרפ לקבוצות //גרף זה מקרה פרטי של קבוצות

1. נראה רדוקציה מ- VC ל-

* שאלה שבודדות תהיה בבחינה: הראו שהשפה X היא NPC .

Hitting set

נרצה לבחור עכשווי אנשים כדי שיהיה נציג בכל וועדה. כך שהבחירה של AB תיתן לנו את כל הוועדה.

חיפשנו אלגוריתם פולינומי ולא מצאנו.
למה הבעיה היא NP?

בහינתן קבוצת נציגים ניתן לבדוק בזמן פולינומי שהיא מכסה את כל הבעיות

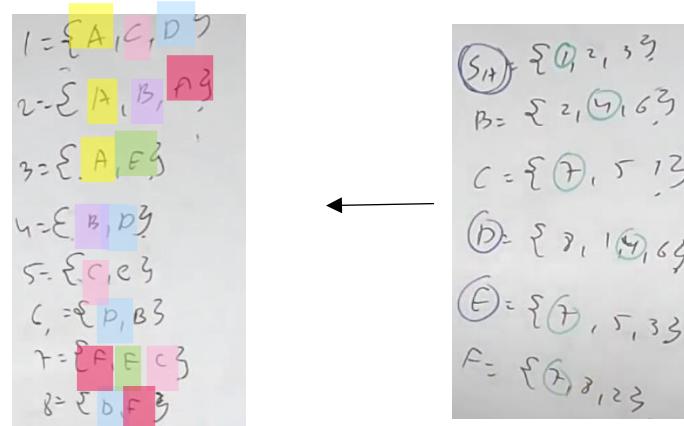
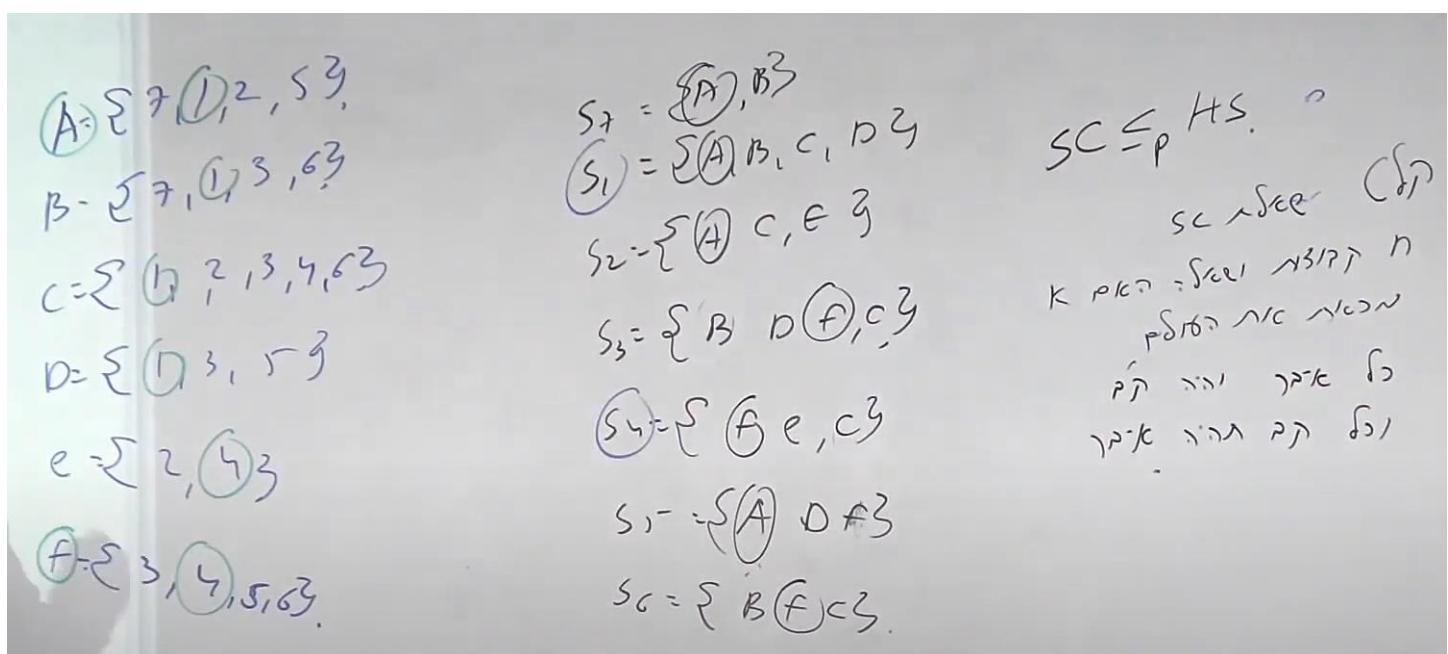
- ה בעיה ההפוכה יש לבחור אנשים ככה שיהיה נציג אחד בכל קבוצה
- הראו $\text{hitting set} \in \text{NPC}$
- נראה $\text{Hitting set} \leq_p \text{Set Cover}$

נרצה להראות שה hitting set קשה.

גלו: בשאלת ST, קלומר – N קבוצות.

שאלה: האם K מהם מכסות את העולם?

דоказתה: כל איבר יהיה קבוצה, וכל קבוצה תהיה איבר.



מצאו SC: בגודל 3, A,D,E : כחול
מצאו HS : בגודל 1: ירוק
הראו דоказת LH המתאים

*אחד ממש הפוך לשני

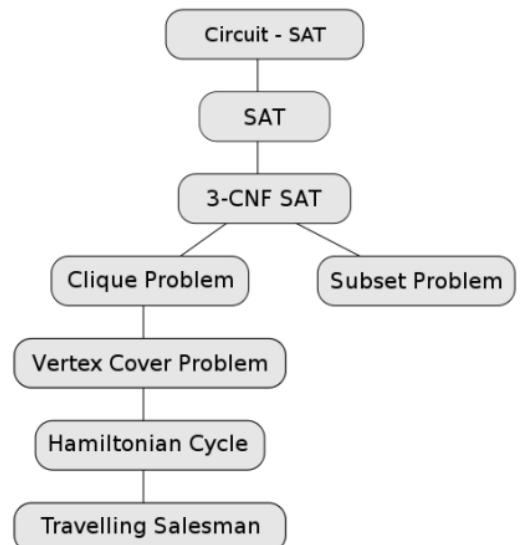
← אומרים שככל הבעיות הללו הן שוות קושי
כלומר, אם אנחנו יכולים לפתור אחת אז נוכל לפתור גם את השניה

רדוֹקצִיה עצמִית (או שפּוֹת שׁמְסֻפּוֹקָות אֶת עצַמֵּם) זה לא נשמע טוב

- **סוגי בעיות :**
 - בעית הכרעה : האם גרף G מכיל קלייקה בגודל K
 - בעית מקס : מה גודל הקלייקה הגדולה ביותר ב- G
 - בעית אופטימיזציה : מצא את הקלייקה הגדולה ביותר ב- G

אם אפשר לפתור להכריע אפשר למצוא אופטימום הראנו על SAT
- נגיד רדוֹקצִיה אשר משתמשת באלג אחר כקופסה שחורה מספר פולינומי של פעמים
- שימוש לבן כל הרדוֹקצִיות של השלמות שראיתם עשו רק צעד אחד

כזוקז'ום NPC



בעית הצביעה

- כל מפה ניתן לצבוע ב-4 צבעים
- שאלת האם גרף הוא 3 צבע היא NPC
- ככל ככל מספר גדול מ-3
- השאלה האם גרף הוא 2 צבע שקרה לשאלת ...
- אם הוא דו-צ.

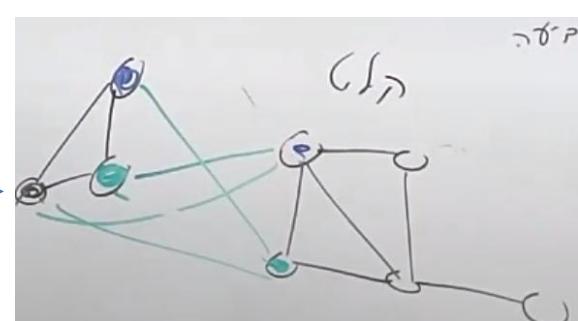
^{**}אם יש קלייקה בגודל 4 צריך 4 צבעים. //
*קלייקה היא חסם תחתון אבל לא חסם הדוק.

← ניתן להגדיר את השפה: { G graph that is able to color in K colors} ← האם השפה NP? בהינתן עדות של הצביעה, מה הצביע של כל קודקוד – ניתן לבדוק שהיא חוקית . //ניתן להראות שנייתן לצבוע קלייקה K עם K צבעים, לא ניתן להראות שאין אפשרות לצבוע עם $K-1$ צבעים.

לשאול אם גרף הוא 2 צבע, זה כמו לשאול אם גרף הוא דו-צדדי.
3 צבע הוא קצת יותר קשה.

הראו ש-3 צבע (3-col) מספק את עצמו:

אם קיימים אלגוריתם שמכרע 3 צבע, אז ניתן למלא את הצביעה.
קלט: מקבלים גרף לא צבוע והאלגוריתם אומר איך לצבוע אותו ב-3 צבעים.
רדוֹקצִיה: נוסיף קלייקה ליד הקלט, בגודל 3. זה בודדות יבצע ב-3 צבעים.
 לחבר את הקלייקה לגרף הקלט ואז בודדות תוכל לצבוע את הגרף.
← עוברים בכל קודקוד פעמיים, כלומר $2k$, כלומר פולינומי.



תרגיל

השאלה היא: האם 4 צבע יכול לעזור עם 3 צבע? וההpf.

המטרה: לעשות רדוקציה מ-3 צבע ל-4 צבע

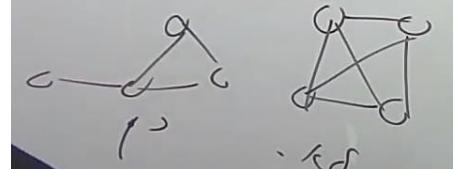
- נתון אלג המכريع אם גраф הוא 4 צבע האם ניתן להשתמש בו על מנת להכريع עם גраф הוא 3 צבע?

- מה לגבי הכוון ההפוך? אם יש אלג ל-3 צבע האם יש אלג ל-4 צבע?

קְלַט: גֶּרֶף

פְּלָט: בעזרת אלגוריתם שאומר האם הגראף 4 צבע

*סוג טעות: עלולים להגיד כן יותר מדי פעמים – לקבל יותר מקרים ממה שצרכית



$$3 \leq 4$$

רדוקציה: מ-3 צבע ל-4 צבע

- נוסיף קודקוד וחיבורו אותו לכל הקודקודיים בגראף הקְלַט, אם הוא מחובר הוא חייב להיות עם צבע ייחודי مثل עצמו
- אם היה אפשר לצבוע את הגראף הישן ב-3 צבעים \leftarrow אז את הגראף החדש אפשר לצבוע ב-4 צבעים

← רדוקציה לפי דoid

- על הקְלַט G נוסיף קודקוד X לא G וnochבר אותו לכל 7 שיר ל-7
- נסמן 'G'. ברור של-X צבע ייחודי.
- אם 'G' 4 צבע \leftarrow G-3-צבע (X+G)'
- אם G 3 צבע \leftarrow G'-4-צבע

Partition

נתנים N מספרים, ואז שואלים אם ניתן לחלק את המספרים הללו ל-2 קבוצות, כך שהסכום של 2 קבוצות שווה.

לדוגמא: הקבוצה 1, 3, 5, 7 שווה ל-10
 $10 = 7 + 3 \leftarrow 7, 5, 4, 3$

- $\{i_j\}_{j \in T} = \sum_{j \in T} i_j : \exists T : \sum_{j \in T} i_j = \sum_{j \notin T} i_j$
- כלומר קיימת חלוקה של המספרים לשתי קבוצות שווות גודל

- הוכח חישוב היא NP שלמה

← קיימת רדוקציה קלה, עם subset sum של subset sum partition. *השאלה היא האם יש רדוקציה הפוכה, מsubset sum לpartition.

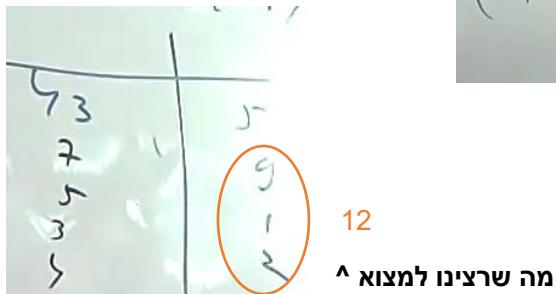
קְלַט: שאלת subset sum $\{a_1, \dots, a_m\}$ subset sum partition : $\{a_1, \dots, a_m\}$ שפוגר partition : $\{a_1, \dots, a_m\}$

רדיונציה: נסמן סכום $a_i =$

1. נסuum את כל המספרים: $K = 12$ (המספר הרצוי), $n=31$ (הסכום של כל המספרים)
2. נסיף 2 מספרים: $a_k + a_{n-k}$ למה? כי אם יש לנו משווה בגודל K אז נחלק את זה ל-2 קבוצות של k ו- $n-k$

היאן שלמה
 $(3, 4, 7, 1, 2, 9, 5)$, $n-k$, $n-k$
 $n+k$, $n-k$

סכום כל המספרים כאן הוא 124,
זה ש策יר לחלק ל-2 קבוצות בגודל 62.



היאן שלמה
 $(3, 4, 7, 1, 2, 9, 5) \text{ for } k=12, n=31$
 $(3, 4, 7, 1, 2, 9, 5, 43, 58)$.

יש לנו 2 מושגים – זמן ומקום
כעת נדבר על סיבוכיות מקום

הערה: מקום פולינומי נמצא בזמן אקספוננציאלי

אבחנות

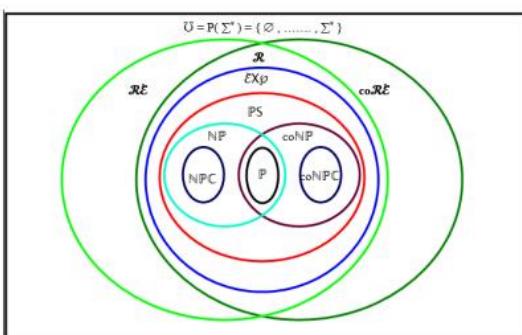
- $PSpace \subseteq NPspace$ ►
- ברור כי חישוב דטרמיניסטי הוא מקרה פרטי של חישוב לא דטרמיניסטי (אפשר להתעלם מהגמר)
- $P \subseteq PSpace$ ►
- כי אם זמן החישוב הוא פולינומי אז ברור שלא ניתן לצורך יותר מקום פולינומי
- באופו שהול $Nn \in NSpace$ ►

از כמה קונפגרציות יש

- נסמן את האב ב Σ
- הזכירון של הריצה הוא לכל היותר $(\alpha)^k + \text{מצב סופי}$ (שורה בקד) ונוק בקלט ($N \log(n)$)
- סהכ $O(2^N) = \log(n) + A + \Sigma^{P(n)}$
- כלומר מספר המצבים לפני שנגיע לולאה אין סופית הוא לכל היותר מעריצי
- מסקנה $Nspace \subseteq EXP$
- ופרט הם ב R

SAVITCH משפט

- $NPSpace \subseteq PSpace$ ►
- המשפט מראה קשר בין חסר דטרמיניזם לעלות מקום
- אם אלג אימות דרוש מקום $(\alpha)^P$ אז אלג הכרעה דרוש מקום $(\alpha)^{P^2}$
- הובודה שזה 2 ולא יותר חשובה להמשר (כל להראות פולינום במשוואת הטימר)
- לאלג יש נוק התחלת אחת ומספר נוק סיום של קבלה נניח אתת
- נסמן m חסם על מספר הקונפגרציות $M \leq |Q|^{|P(n)|} \Sigma^{P(n)}$
- נריין $\text{Reach}(C_{start}, C_{end}, m)$



- ▶ לאLONG יש נק התחלה אחת ומספר נק סיום של קבלת כניסה אחת
- ▶ נסמן m חסם על מספר הקונפוגרציות $M \leq |Q|^P(n) \Sigma^{P(n)}$
- ▶ נרץ $\text{Reach}(C_{\text{start}}, C_{\text{end}}, m)$

לכל $W < m$

Reach(C_1, C_2, m)

1. if $m = 0$
2. return $C_1 = C_2$
3. if $m = 1$
4. return $C_1 \models C_2 // M$ ניתן לעבור מההצורה C_1 להצורה C_2 על ידי מעבר אחד של M
5. for each configuration D לכל $|D|^{P(n)} \cdot |\Sigma|^{P(n)} \cdot |Q|^{P(n)}$ הצורות
6. if $\text{Reach}(C_1, D, \left\lfloor \frac{m}{2} \right\rfloor)$ and $\text{Reach}(D, C_2, \left\lceil \frac{m}{2} \right\rceil)$
7. return "yes"
8. return "no"

רקורסיה

- ▶ בעצם יוצרים גרף של קונפוגרציות של האLONG
- ▶ יש קשר בין שני קודקודים אם ניתן לעבור ביניהם בצעד אחד
- ▶ בודק Reach בזאת שיש מסלול בין start ל end
- ▶ למה לא ? BFS ?
- ▶ כי הגרף מעריצי ב נ

אז כמה יצא ?

- ▶ צריך לשמר את הקונפוגרציה שאנוחנו בדקים + W וכן נפח כאן הוא $O(P(n))$
- ▶ הקריאה הרקורסיבית יתבצעו לא באופן מקביל אלא בירידה בענף כלומר $m, \frac{m}{2}, \frac{m}{4}, \dots, \frac{1}{0}$
- ▶ וכן עומק המחסנית הוא $LogM$. כל קראיה ממחשבת 2 קונפוגרציות מספר m וערך סחכ $O(p(n) \log M) = O(p(n) \times \log(|\Sigma|^{P(n)})) = O(P(n) \log(|\Sigma|^{P(n)}))$
- ▶ וכן בכלל שלב 2 $O(p(n) \log M) = O(p(n) \times \log(|\Sigma|^{P(n)})) = O(P(n) \log(|\Sigma|^{P(n)}))$

מסקנות

- Npspace=Pspace ▶
- $P \subseteq NP \subseteq PSpace \subseteq EXP$ ▶
- כול הטענות הם חלשות אבל סבוריים שהם הכלות ממש
- $P \neq EXP$ הדבר היחיד שידוע ממש הוא ▶

מה עם משלימים

- CoPSpace=PSpace ▶
 - ברור כי נרוץ במקום פולינומי ונהזיר את התשובה ההפוכה
 - מסקנה ▶
- $$CoNP \subseteq CoNPSPACE = PSPACE$$

בעיות שלמות ב PSpace

הגדנה: L שייך ל-PSpace והוא קשה: לכל L' ב-PSpace יש רדוקציה פולינומית ב- L .

$L \in PSpace$ ▶

$\in L' \text{ יש דוקציה } \Leftarrow L \in PSpace$ ▶

1. ניתן לבדוק מסלול בגראף ע"י זיכרון לוגריטמי בגודל הגראף.

ראינו גראף שהוא $(2n)^0 \leftarrow$ פולינומי.

2. אם יש עץ מעירכי, ניתן לבדוק ענף בזמן/מקום פולינומי.

בעיות שלמות ב PSPACE

QBF הוא פסוק לוגי שכול משטנה שלו קשור לכמת לדוגמא ▶

$\forall x_1 \exists x_2 \forall x_3 (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \bar{x}_2)$ ▶

$| \quad \{ \langle Q \rangle = \text{TQBF} \text{ שערכו 1 QBF ביטוי } \}$ ▶

טענה TQBF שלמה ב-PSpace ▶

TQBF אלו כל הנוסחאות מהצורה זו שהן True. והשפה הזאת היא PSpace Complete

TQBF \in PSPACE

אלגוריתם: קלט $\langle Q \rangle$:

1. אם Q לא כולל כמתים, אז הוא קבוע 1 או 0. נחזיר אותו.

2. אם $\exists x P = Q$, נקרא רקורסיבית ל- P עם $x=0$, אח"כ נקרא רקורסיבית ל- P עם $x=1$, ונחזיר 1 אם שניהם 0 ו- 0 אם לא.

3. אם $\forall x P = Q$, נקרא רקורסיבית ל- P עם $x=0$, אח"כ נקרא רקורסיבית ל- P עם $x=1$, ונחזיר 1 אם אחד מהם 1 ו- 0 אם לא.

▶ נסמן \vdash משפטנים ונוסחה באורך m אזי

סבוכיות נפח: מכיוון שהקריאות הרקורסיביות מתבצעות בזו אחר זו, אז בכל שלב יש במחסנית לכל היותר m ביטויים לוגיים, כאשר m מספר המשתנים, שאורך כל אחד מהם $O(m)$, כאשר m נפח הקלט, ולכן בכל שלב הנפח הוא $O(mn)$ פולינומי.

□

חלוקת הקשה

שלכל $L \in PSpace$ מתקיים $L \leq_p TQBF$

7 אלו בעיות שניתן להכريع במקומות \log // פולינומי במקומות לוגריטמי

*אי אפשר לקרוא לקלט בזמן לוגריטמי

- אם אין אלגוריתמים פולינומי מה זה אומר ?
- יש אין סופ קלטים "קשים" אבל עדין יתכן והרוב קל

← איך מתמודדים עם בעיות בPN? דוגמאות לכמה גישות שיש בתעשייה:

בעיות בNP מענייניות מה עושים

פתרונות ארוכים בזמן Brute Force

פתרונות של תת בעיה (לא בכלל הגרפים קשה למצוא SI)

מנסים ומקווים לטוב
שינוי של מיד בעיה
קירוב

		פולינומי
זמן	מקום	
חישוב זמן הכרעה	PLACE	
חישוב רגיל הכרעה	PS	P
לא דטרמיניסטי	NPS = PS	NP // קיימ
משלים לא דטרמיניסטי	CoNPSpace = PS	CoNP // לכל

$$\begin{array}{c} \mathbb{R}^{(n)} \\ \hookrightarrow \\ \mathbb{R}^{(n)} = \mathbb{R}^n \\ \text{סידור} \end{array}$$

$$\text{NSpace}(T) = \text{Space}(T^2)$$

משפט: ניתן לבדוק מסלול מסוים בגרף כדי זיכרון לוגריתמי בגודל הגרף.

הזיכרון הוא פולינומי, ולכן מספר הקונפיגורציות הוא $2^{|V|}$. כך שהזיכרון הוא:

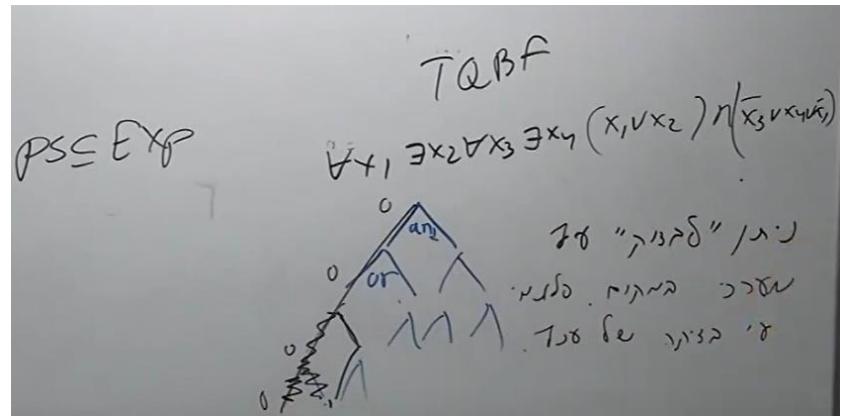
DFS: זמן לינארי ומקום לינארי. ואם הגרפים ענקיים אז ניתן לשלב זמן פולינומי עם מקום לוגריתמי.

//off trade הפוך. בד"כ הינו רוצה לחסוך זמן ולבזבז זיכרון. אבל בגלל שהגרף כל כך גדול אז נרצה לבצע זמן כדי לחסוך זיכרון.

ניתן לפתור זאת ע"י רקורסיה (גוזלת זמן וחוסכת זיכרון).

מחלקה 1

- 1 – כל הביעות שניתנו להכריע במקום PSPACE
- $\text{NL} \cup \text{log} // \text{כל הביעות שיש פחות מקום (זיכרון) מהקלט}$
- $\text{NL} - \text{כל הביעות שניתנו לאמת במקום}$
- פולינומי, ע"י חישוב לא דטרמיניסטי //



- $\text{SC} - \text{כל הביעות שניתנו לאמת במקום פולינומי ב-} \text{a} // \text{כל הביעות שיש בהם יותר מקום (זיכרון) מהקלט}$

קירובים לבעיות NP שלמות

מה זה אומר קשה?

- אם אין אלגוריתם פולינומי, מה זה אומר? // מה זה אומר שביעיה היא קשה?
- יש אין סוף קלטים "קשיים", אבל עדין יתכן והרחב כל.

מה עושים?

- פתרונות ארוכים בזמן
- פתרונות של תתי בעיה (לא בכל הגרפים קשה למצוא צו)
- שינוי של מידת הבעיה
- קירוב

קירוב

← מה זה קירוב? אנחנו מביניםuai שאי אפשר לפתור את הבעיה, אבל נרצה למצוא תשובה קרובה מספק לדוגמה: אומרים למצוא את הקliquה המקסימלית, ובודגמא הנтуונה היא בגודל 6. אבל אי אפשר למצוא קliquה בגודל 6 אז ננסה למצוא את הקliquה הכילגולה שהכי קרובוה ל-6, 5 – וזה מספיק טוב. נרצה לעצמנו שגיאה קטנה.

← כמובן, האלגוריתם יטעה אבל לא יותר מדי.

← אלגוריתם יעיל פולינומי.

אלגוריתם קירוב

נניח אלגוריתם קירוב לבעיית קliquה

$$\begin{aligned} \text{אם } \text{ידעו } \text{שהפתרון } 100 = C^*, \\ \text{אז } \text{nich } \text{ש } 2 <= C^* / C \\ 2 = 100/50 = 2 \text{ עבר בעיית מקסימום} \\ \text{*כל שהמספר מתקרב ל-1,} \\ \text{האלגוריתם טוב יותר} \end{aligned}$$

אנו אומרים כי אלגוריתם קירוב A הוא בעליחס קירוב ρ ($\rho \geq 1$)

אם עבר כל קלט היחס בין הערות C של הפתרון שמאפיין A

עלות הפתרון האופטימלי * C מקיים:

$$\frac{C^*}{C} \leq \rho \quad \text{מעבר בעיית מקסימום או}$$

$$\frac{C}{C^*} \leq \rho \quad \text{מעבר בעיתות מינימום.}$$

← נניח שיש אלגוריתם A קירוב 3 לקלliquה, והחזיר תשובה 150

מה גודל הקלliquה המקסימלית בגרף? $G = (A)$

התשובה: $C^* = 150 \Rightarrow 450 // 450 \text{ בغالל קירוב 3 לקלliquה}$

← נניח שיש אלגוריתם 2 קירוב LC על גרף G והחזיר 200.

מה גודל הVC הנכון? // בעיתות מינימום

התשובה: $C^* = 200 \Rightarrow 100 // 100 \text{ כי זה הפוך מבחינת מינימום}$

הפרוכסינט קיורוב פגאיית כיסוי קומפקטיים

טענה:

האלגוריתם APPROX-VERTEX-COVER הוא בעל יחס קירוב 2

הוכחה:

נסמן ב- A - את קבוצת הקשתות (u, v) שנבחרה בשורה 4. קבוצה זו אינה מכילה קשתות שלן צמתים משותפים, לפיכך $|C| = 2|A|$.

עתה, כמות הפתרון האופטימלי $* C$ מכסם את קשתות A ולכך מカリים לפחות צומת אחד לכל קשת ומכאן $|A| \leq |C^*|$. ובסך הכל נקבל $|C| \leq 2|C^*|$.

APROX-VERTEX COVER(G)

1. $C \leftarrow \emptyset$
2. $E' \leftarrow E$
3. **while** $E' \neq \emptyset$
4. **do let** (u, v) **be an arbitrary edge of** E'
5. $C \leftarrow C \cup \{u, v\}$
6. remove from E' every edge incident on either u or v
7. **return** C

↙ הוכיח הנ"ל הוא לינארי לגודל הגרף, $O(n+h)$

תרגילים כיתה ז' פונק אנטון

.1. (20 נקודות) שאליה זו דנה באלגוריתם הקירוב לביעית CISI הקודקודים (VC).

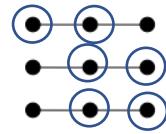
א. (10 נקודות) הצג גרף הכלול 9 קודקודים ו-6 קשתות, כך שבכל הרצה של אלגוריתם הקירוב יתקבל יחס קירוב 2.

ב. (10 נקודות) הצג גרף הכלול 6 קודקודים ו-5 קשתות, כך שבכל הרצה של אלגוריתם הקירוב יתקבל יחס קירוב 2.

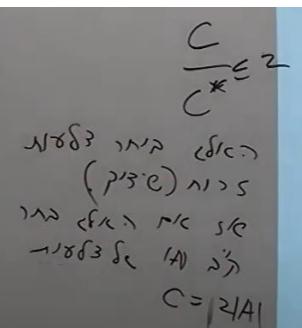
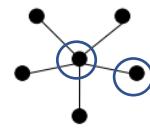
פתרונות:



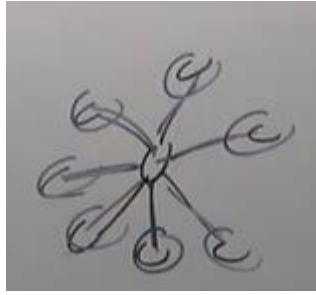
א.



ב.



תכלית



במקרה זה ^ ל淮南 7 קודקודים במקום 1

לפניכם אלגוריתם הקירוב לבנייה VC שעבר שני, כך שבכל איטרציה נבחר באופן אחד מצמת הקשר. הראה כי לכל ערך קבוע k . קיים גראף שבו האלגוריתם עשו שלא לספק k -קירוב.

APROX-VERTEX COVER(G)

1. $C \leftarrow \emptyset$
2. $E' \leftarrow E$
3. **while** $E' \neq \emptyset$
4. **do** let (u,v) be an arbitrary edge of E'
5. $C \leftarrow C \cup \{u\}$
6. remove from E' every edge incident on u
7. **return** C

אפקטיבית קירוג פגאיית CISI קבוצות



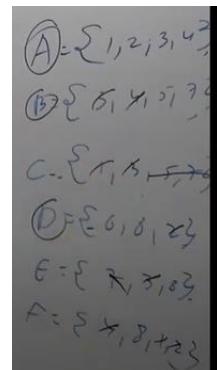
APROX-SET-COVER(S, (S_i))

1. $U \leftarrow S$
2. $C \leftarrow \emptyset$
3. **while** $U \neq \emptyset$
4. **do** select an S_i that maximize $|S_i \cap U|$
5. $U \leftarrow U - S_i$
6. $C \leftarrow C \cup \{S_i\}$
7. **return** C

מה הבעיה? יש הרבה תת-קבוצות וריצים לבחור קבוצות כך שנכסה את כל העולם.

← זהה בעיה שאין לנו קירוב ממש טוב שלו.

מחפשים את הקבוצה היכי גדולה.

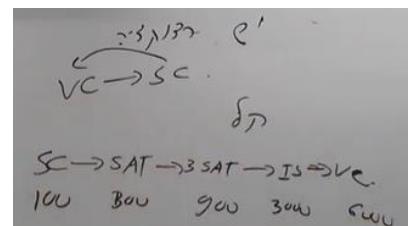


*האלגוריתם נותן יחס קירוב שטלי בגודל $1 + \log(s) + 1$

טענה האלגוריתם APPROX-VERTEX-COVER הוא בעלי יחס קירוב $(|S| + 1) \log(|S|)$

↙ הראנו שיש רדוקציה מ-VC ל-SC.
AIR? כל קודקוד יש לו קודקוד שנוגע בו, זו הקבוצה ואז פותחים SC.
מה הכוון הפוך? (כי VC שלמה) כל בעיה נוכל לתרגם ל-SAT, את SAT מתרגמים ל-3SAT, את 3SAT מתרגמים ל-2SAT, ו-2SAT מתרגמים ל-VC.

- לכל הבעיה האלו יש קושי אם לפתור או לא לפתור.
- מבחינת קירוב זה עולם אחר לחילוטין \
- כולל NP קשות (NPC), אבל לא כולן שווות קירוב



נגיד יש גראף וUMBRAHSים לייצר חתך (כלומר, לחלק את הגראף ל-2 קבוצות של קודקודים), כדי שיהיו הכי הרבה קשיות בחתך.

↳ נושא הפור: אם נרצה הכי מעט קשיות בחתך, נעשה זאת ע"י זרימה.

נשים 1 על כל הקשיות, ונמצא חתך מינימלי. //כלומר, מציאת חתך מינימלי היא בעיה BC: יש אלגוריתם פולינומי שפותר אותה.

↳ מציאת חתך מקסימלי היא בעיה NP שלמה. *זאת אחת הדוגמאות היחידות שבה מינימום != מקסימום

MAX cut

1. חתך מקסימלי בגרף הגדירה

2. בניית קירוב (חלוקת של הגראף לשתי קבוצות)

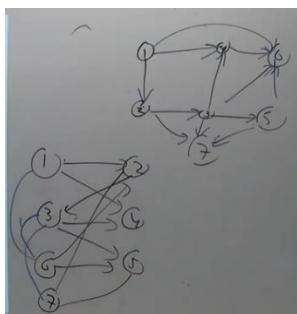
1. ניתן לכל קודקוד שם מ-1 עד ח

2. נכוון את הקשיות מקטן לגודל

3. נורץ מ-1 עד ח

1. שים את הקודקוד ה-1 בצד שממকסם את מספר הקשיות שנכנסות אליו

↳ בכל שלב נסתכל ק על הקשיות שנכנסות לקודקוד (בשלב 1 נסתכל על כל הקשיות שנכנסות לקודקוד 1) מה קורה בתכלס? חילקנו את הקשיות ל-ח קבוצות כך שכל קשת שייכת לדיאק לקבוצה אחת. (קשת 1-5 שייכת לקבוצה 5. הקשת 2-4 שייכת לקבוצה 4 ↳ ככלומר, הקצה הגדול שלה) //כל קשת שייכת לקבוצה אחת בדיק.

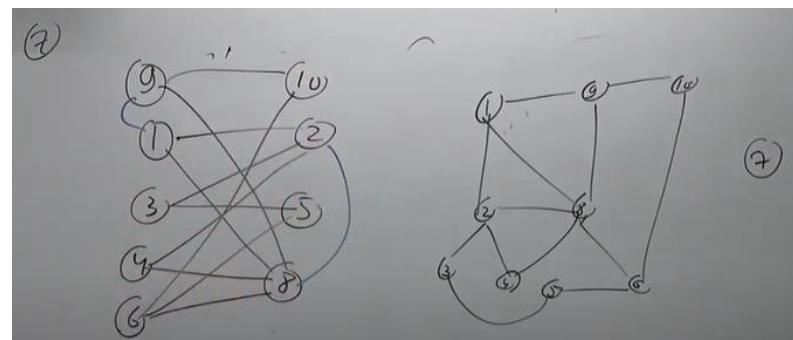


מה האלגוריתם שנעשה? //זה שמתואר לעלה

1. בכל צעד, בצד ה-1, שים את קודקוד 1 בצד שיתפeo הכי הרבה קשיות מהקבוצה שלו.

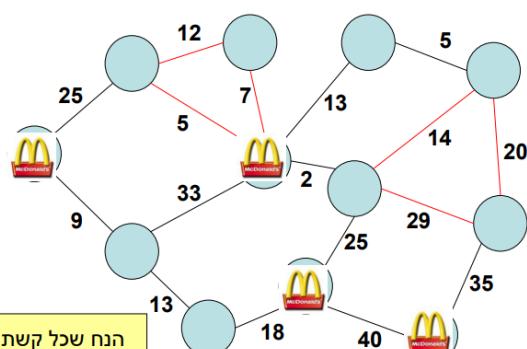
//בכל שלב לוקחים לפחות חצי מהקשיות

← כך שהפתרון גדול שווה מחצי מהקשיות בגרף (מלל הקשיות)



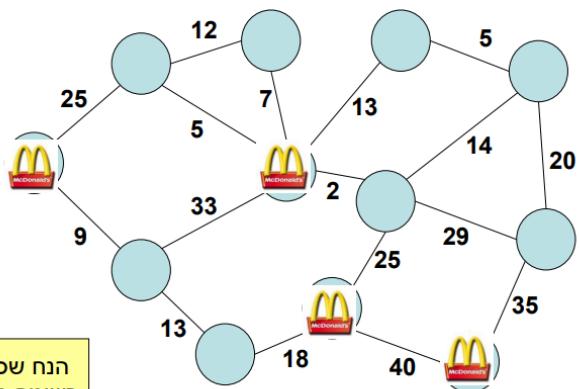
愧因 k-ארכטימט הארכטימט Metric k-Center

נתון גרף שלם, משקלל ולא מקוון ($G = (V, E)$) אשר משקלותיו מקיימים את א-שוויון המשולש וכן קבוע טבוי k . יש למצאו קבוצה $V \subseteq S$, $|S|=k$ עבורו $\max_{v \in V} \text{dist}(v, S) \leq k$



ה愧יה $\leq k$ מ-ק-ארכטימט

נתונה רשת כבישים. יש להקימ 4 סניפים, כך שהמרקם הגדול ביותר מעריך לסניף הקרוב ביותר יהיה מינימלי.



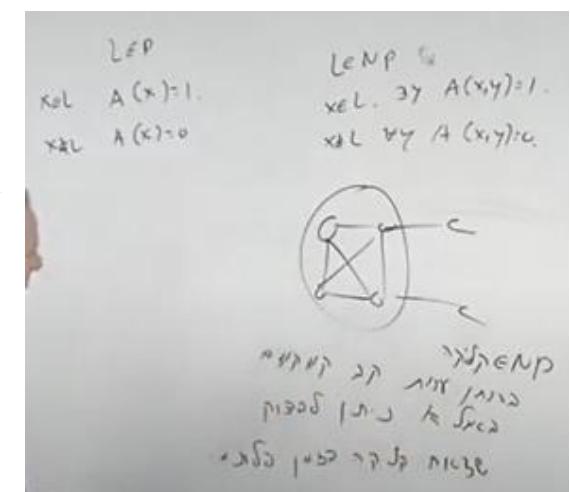
שאלה: מבקשים להוכיח שהבעיה שיכת ל-PN, מה הכוונה?

תשובה: שפה שיכת ל-PN אם ניתן לאמת אותה. אם בהינתן עדות ניתן לבדוק אותה בזמן פולינומי.

↳ קליקה שיכת ל-PN אם אומרים לי שקודקודים מסוימים הם קליקה ונitin לבדוק שזה נכון – בזמן פולינומי.
(זה גם ההבדל בין מציאת תשובה – מציאת מסלול, לבין אימומות תשובה – אם אומרים לי שהיא קיימת, ניתן למצוא אותה ולדעת אם היא)

*שפה שיכת ל-PN אם קיימ עד שקרה אותה.

↳ בודאות היה בבחן שאלה כזו: הראו ש L שיר NPC לדוגמה.



הגדרה של רדוקציה: רדוקציה מראה קשר בין 2 בעיות, בעזרת בעיה אחת ניתן לפתור את השניה. אך בעצם נראה קשר ביניהם. לכן אם עושים רדוקציה משפה A ל-B, כלומר נתרגם בעיה A לבעיה B

$$A \xrightarrow{\beta} B$$

$$A \leq_p B$$

בעולם האלגוריתמי זה שימושי כדי להגיד שמצאנו פתרון.

בתחילת הקורס עשינו רדוקציה משידוך לזרימה. //השתמשנו ברדוקציה בתור כל אלגוריתמי.

תרגמוני בעית שידוך לבעית זרימה. פתרוני זרימה ובעזרת זה פתרתי שידוך. = כל אלגוריתמי.

לפי היגיון זהה, אם נפתרו רדוקציה יותר קשה אז נפתרו את הרדוקציה הקודמת לה.

ובמילים שלנו: אם A היא NPC אז גם B NPC. $NP\ Complete = NPC//$

אלגוריתמים מקוונים (online)

אלגוריתם מסוון אינו מקבל את כל הקלט שלו מראש, אלא תוך כדי "ריצה", ועל האלגוריתם זהה לבצע החלטות בתנאי חוסר וודאות ניתוח של אלגוריתם מסווג זה נקרא ניתוח תחרותי והוא נעשה בהשוויה לאלגוריתם offline אופטימלי אשר מכיר את כל הקלט. במדד להצלחת האלגוריתם נקרא "יחס תחרותיות".

//אלגוריתם מסווג זה צריך להתחיל להחליט לפני פני שראינו את כל המידע הדרוש

גא'ית הסקי'



עלות השכרת ציוד סקי – 100 דולר ליום
עלות רכישה של ציוד סקי – B מאות דולרים

איןנו יודעים כמה ימים תמשך חופשת הסקי'
אם כדי לשכור או לקנות את הציוד?

ໄຟັ້ນເກົວ



- דפודוף (paging) – מערכת הפעלה צריכה להחליט אילו דפים היא תיקח מהזיכרון האיטי (דיסק) לזמן המהיר (מטמון, cache) מבלי לדעת אילו דפים ידרשו בעתיד.



- ניוטו רובוט – יצא מהbbox על סמך תמונה חלקית שיש ברשותו.



- בעית החתונה – בחור קבע האם להמשיך/לסיים קשר מרבי לדעת מי היי הבחירה שיופיע בהמשך בעית החתונה – בחור קבע האם להמשיך/לסיים

გა'ית הסקי'



- ח – מספר ימי הסקי (לא ידוע מראש)
- ס – סדרת אירועים

$\text{OPT}(\sigma) = \min(B, n)$ offline

$$ON_T(\sigma) = \begin{cases} n & T > n \\ T + B - 1 & T \leq n \end{cases}$$

פתרונות מקווני ON_T

- שוכר ציוד עד היום T - T
- ביום $-T$ רכוש את הציוד

- אם החופשה נמשכת מ-10 ימים והלאה שווה לקנות, אחרת (פחות מ-10 ימים) אך כבר שווה לשכור את הציוד.

- נתונים: קנייה – 1000
שכרה – 100

- תשובה: offline: אם $10 \geq T \leftarrow$ תקנה

- אלגוריתם online כל פעם חושב מחדש אם לקוחות או לא, אך ברור שיש סיכוי שהוא

- טעה – הטעות היא שהחופשה נגמרה יומם אחריה הקניה (לדוגמה).

- האסטרטגיה שאומרת איך לקנות את זה אם שנקנה כשהזכנו מספיק.

- ככלומר, אלגוריתם online אומר שתקנה ביום העשוי.

- ואז הוא שכר 9 ימים וקנה ביום העשוי, ולכן שילם 1900.

- לעומת זאת offline שילם רק 1000.

- אלגוריתם online מפסיד על סדרות קטנות, כך ש online יקנה בוודאות offline יקנה.

- אלגוריתם offline קונה בוודאות אחרי 10 ימים.

השועל בכרם

גא'ית הערף והכלם



חץ צרי:
שועל מתקרב לכרם, יושוף מספר לו כי
קיימת פרצה בגדר, אך הוא אינו זוכר
באיזה כיוון ובאיזה מրחק. מה יעשה השועל?

חץ כהן:
שועל מתקרב לכרם, ינשוף מספר לו כי
בمرחק חקיימת פרצה בגדר, אך הוא
אינו זוכר באיזה כיוון. מה יעשה השועל?

אלגוריתם NO
הו 3 - תחרותי

פתרון אופטימלי offline $OPT(\sigma) = n$
פתרון מקוון NO
• פניה ח 2 צעדים ימינה.
• אם הפרצה לא נמצאה פניה ח 2 צעדים שמאלה

- כיוון 1:** נלך בצעדים קטנים.
אם האלגוריתם מפשל? אם הפרצה מאד רוחקה אז מספר הצעדים יהיה פי N מהמרחק.
 - כיוון 2:** נלך בצעדים גדולים.
אם האלגוריתם מפשל? אם הפרצה נורא קרובה אז נפספס אותה, אך נשלם המון במקום על צעד קטן.
- הפתרון: צעד בגודל משתנה.
הצעד הראשון, הצעד השני יהיה 2, הצעד השלישי יהיה 4, הרביעי 8 וכו' (סדרה הנדסית עולה)
← נגדיל את הסיכון בין הצעדים,

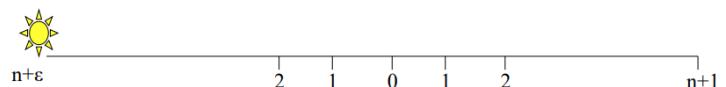
פתרון מקוון שני ALG_2

ALG ₂	
t ← 0	.1
כל עוד לא נמצאה הפרצה, בצע:	.2
אלגוריתם ALG ₂	
הו 13 - תחרותי	
.3 חזר למרכז	
.4 פנה ח 2 צעדים ימינה.	
.5 חזר למרכז	
.6 אם הפרצה לא נמצאה פנה ח 2 צעדים שמאלה.	
t ← t+1	.7

פתרון מקוון ראשון ALG_1

ALG ₁	
t ← 1	.1
כל עוד לא נמצאה הפרצה, בצע:	.2
חזר למרכז	.3
פנה ח 2 צעדים ימינה.	.4
חזר למרכז	.5
אם הפרצה לא נמצאה פנה ח 2 צעדים שמאלה.	.6
t ← t+1	.7

מהו יחס התחרותיות של ALG_1 ?



יחס התחרותיות מהלকת

פתרון אופטימלי offline $OPT(\sigma) = n + \epsilon$

בහינתן אלגוריתם מקוון NO (יסומן גם ALG) ייחס התחרותיות
החזקת הוא – α אם מתקיים: $(\sigma) \leq \alpha \cdot OPT(\sigma)$

פתרון מקוון $\epsilon + \alpha \cdot OPT(\sigma) = n + \epsilon$

$$\text{יחס התחרותיות} = 2n + 5 + \frac{2}{n}$$

Paging

פאיינט גזפינג (Paging)

יש אינטראס לשימוש כמה שיותר מידע ממאזות מקומות

נגיד יש מקום ל-10 דפים בזיכרון
ומגיעות בקשות – offline יכול לבצע
אופימיזציה להה.
האלגוריתם אומר שכרגע יש לו בזיכרון את
דפים 1,2,4 אבל כרגע ביקשו את 3.

SLRU – אם רואים דף מסמננו,
אם סימנו את-column אז מוחקים.
בנוקודה הזו מפסיקים להיות SLRU.
(כי פתאום Column נהיים שוויים)

מערכת הפעלה צריכה לנקח דפים מהזיכרון האיטי (DISK) לזכרון המהיר (MATTELON, cache) המכיל K דפים. לצורך כך עליה להוציא דפים מהזיכרון המהיר מבלי לדעת אילו דפים ידרשו בעתיד.



אפשרויות:

- first in first out – FIFO
- הדף הוותיק ביותר בזיכרון המתמן יצא מן הזיכרון המהיר LRU – least recently used
- הדף שלא נדרש זמן ארוך ביותר יצא מן הזיכרון המהיר LFU – least frequently used
- הדף הפחות נדרש (בשכיחותו) יצא מן הזיכרון המהיר

אלגוריתם הסימון למימוש בפועל של שיטת LRU:

האלגוריתם SLRU הוא הכי טוב שנייתן לקבל

- דף נדרש מסומן (בין אם היה בזיכרון המהיר או לא)
- אם כל הדפים מסומנים הסימון נמחק מכלם
- מן הזיכרון המהיר נזרק בעת הצורך דף לא מסומן

התהילר עם bit dirty הוא K
תחרותי

טענה: אלגוריתם LRU הוא k-תחרותי.

K = גודל ה pages. (כמה מקום יש בזיכרון המהיר)
לפי האלגוריתם זהה, אם SLRU הוא K
תחרותי, אז SLRU לעומת offline טועה
יותר פי K. (לכל היוטר בפקטור K)
* פיא היא קבוצה שמכילה K דפים
שונים ככה שהצעד הבא יהיה K+1.

הוכחה:
נסמן את הדרישות לדפים – $\sigma_1 \sigma_2 \dots \sigma_n = \sigma$
נחלק את הדרישות לSUBSETS: $\Pi_1 \dots \Pi_m = \sigma$
כאשר $\sigma_1 \sigma_2 \dots \sigma_n = \Pi_1$ מכיל K דפים שונים
- $\sigma_{t_1+1} \sigma_{t_1+2} \dots \sigma_{t_1+2} = \Pi_2$ מכיל 1+K דפים שונים.
באופן דומה $\sigma_{t_2+1} \sigma_{t_2+2} \dots \sigma_{t_2+2} = \Pi_3$ מכיל K דפים שונים
- $\sigma_{t_2+1} \sigma_{t_2+2} \dots \sigma_{t_2+2} = \sigma$ מכיל 1+K דפים שונים.

עתה בכל סבב האלגוריתם האופטימלי offline דורש לפחות דף אחד מהזיכרון האיטי והאלגוריתם הנוכחי SLRU דורש לכל היוטר K דפים.

- אך עכשו לכל אלגוריתם מקוון ניתן לבנות סדרת אירועים רעה שבה, ברגע שהוא מותר על דף – זה יהיה הדף שצורך.
از לאחר הגדרת האלגוריתם, ניתן לשים K+1 ועוד לפחות במכoon את הדף.
לא משנה איך נגדיר, תמיד נבקש את הדף הזרוק.
↳ יש מצב זהה בזיכרון (אלגוריתם).
אבל אם יש K+1 בקשות, ניתן לבקש מערכת הפעלה לעבוד בדיק בפקטור K.

טענה: כל אלגוריתם מקוון הוא לפחות k-תחרותי.

הוכחה:

- נניח כי בסך הכל ישנו $1+k$ דפים וمتבצעים k קריאות
- עבור כל אלגוריתם מקוון נתיחס לסדרת אירועים רעה – סובה הדף שנזרק הוא הדף שנדרש בקריאה הבאה.
- האלגוריתם המquoון דורש k דפים מן הזמן הנוכחי בזמן שהאלגוריתם האופטימלי מסתפק בדף אחד.

כאיית האלגוריתם



לפי ההסבר של דויד:

נקבל סדרה של מועדים, וכל אחד מהם קיבל ציון: 6, 5, 7, 12, 4, ... ועברנו על המועד עם הציון 12, אבל אי אפשר לחזור אליו

פה השאלה היא לא מה הסיכוי לטעות, היא שахץ אופטימום: או שלקחנו את המועד הכי טוב או שלא.

↳ למשה, נרצה למקסם את האפשרות למצאו את המועד הכי טוב.

↳ יחס התחרות הוא מה הסיכוי להיות עליו להחליט אם לבחור במועדת הנוכחית או לדוחתה (ambil צויך).

מנהל מראין ח מועדות לתפקיד מזכירה, לאחר כל ראיון עלוי להחליט אם לבחור במועדת הנוכחית או לדוחתה (ambil אפרשות לפנות אליה בהמשך)

השאלה העיקרית היא מה
ה**k**?

כמו כל הבעיות שראינו, יש 2 סוג טויות: אם נקטין יותר מידי לא נגיע לטוב, ואם נגדיל יותר מידי נפספס את הטוב.

נססה למקסם את ההסתברות
לצוא את המועד הכי טוב.

↳ צריך להבין מה
ההסתברות הכי טובה

אלגוריתם מקוון ALG:

- דחה את k המועדות הראשונות

- בחר במועדת הראשונה המוצלת מ-k המועדות הראשונות
(אם אין צו זיה בחר במועדת האחרונה)

טענה: e/k הוא הערך המביא למקסימום את ההסתברות
לבחירה המועדת המוצלת ביותר (מרטין גרדנר)

הוכחה:
 מבלי לחזור אחורה (אם פספסת אז
 נסמן: A – האלגוריתם המקורי בוחר במוועמדת המוצלחת ביותר.
 פספסות)
 B_i – המועמדת ה-*i* היא המוצלחת ביותר

לפי נוסחת ההסתברות השלמה:

$$P(A) = \sum_{i=1}^k P(A | B_i) \cdot P(B_i) = \frac{1}{n} \sum_{i=1}^n P(A | B_i)$$

את כל ה*A* הראשונים פושלים = 0

עבור $k \leq i \leq n$ מתקיים: $P(A | B_i) = 0$

$$\text{ולכן: } P(A) = \frac{1}{n} \sum_{i=k+1}^n P(A | B_i)$$

עבור $k > n$ הסיכוי שתבחר המועמדת המוצלחת ביותר הוא הסיכוי
 שהמועמדת המוצלחת מבין 1-*n* המועמדות הראשונות נמצאת כבר
 בין *k* המועמדות שנדחו.

$$\text{ולכן: } P(A | B_i) = \frac{k}{i-1}$$

$$\text{ומכאן: } P(A) = \frac{1}{n} \sum_{i=k+1}^n \frac{k}{i-1} = \frac{k}{n} [H_{n-1} - H_{k-1}] \approx \frac{k}{n} \ln\left(\frac{n}{k}\right)$$

עתה נסמן $x = \frac{k}{n}$ ונחפש ערך עבורו $P(A)$ מקסימלי.

$$P'(A) = -\ln x - 1 = 0$$

$$\text{ולכן: } x = \frac{1}{e}$$

מש"ל

$$\text{ומכאן: } k = \frac{n}{e}$$

שאלה: יש ביןין בן 100 קומות, 2 כドורי בדולח החיים – מאייזו קומה הcadori בדולח נשברים?

תשובה:

שאלה: לוקחים קלף בין 1 ל-1000 ושים אותו הפוך על השולחן, צריך לנחש מספר גדול מהמספר שיש בקלף. ברגע שהצלחנו אנחנו מקבלים פ' 5 מגודל הקלף, אבל משלמים על גודל הניחוש שלנו. ניחשו 100 ומשלמים 500.

מותר לתת כמה ניחושים.

תשובה: אלגוריתם: אסטרטגיה אחת: תתקדם בקפיצות של 1, זה לא טוב אם X גדול. אסטרטגיה שנייה: תתקדם בקפיצות גדולות, זה לא טוב אם X קטן.

↙ איך AMAZON? משנים את גודל הקפיצה (online)

ונבדוק מה קורה עם קפיצות ריבועית: נתחל עם 1, 2, 4, 8, 16, ... : W.C $X = 2^n + 1$

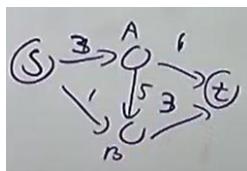
: offline

יחס התחרות: יחס התחרות 4. (אם שואלים על למצוא אלגוריתם שבו לפחות יחס התחרות 4, אז זה נכון)

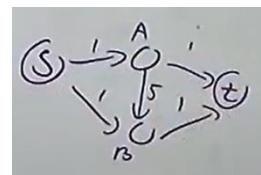
זרה לבחן

1. * לפני פתרת השאלה להבין אותה ולצייר דוגמא

נתונה רשת זרימה $G = (V, E) \rightarrow \mathbb{R}^+$ עם קבועים $c: E \rightarrow \mathbb{R}^+$, מקור s ובור t , וקשת $(v, u) = e$ בראשת. תארו אלגוריתםיעיל, הקובע האם בל זרימה מקסימלית בראשת, קובעת על e זרימה חיובית. הוכחו את נכונות האלגוריתם ותשבו את סבוכותו.



בדוגמא זו ←
זרימת מקסימום
תשتمש בקשת AB



בדוגמא זו לא ←
כל זרימת מקסימום ←
תשتمש בקשת AB

• נרצה להבדיל בין 2 המקרים לעיל

↙ קבועים הם תמיד חיוביים (פונקציית הזרימה היא תמיד מ- \mathbb{R}^+ ל- \mathbb{R}^+)

↙ ניתוח השאלה: האם כל זרימה מקסימלית קובעת זרימה חיובית?

מה זה אומר על e? שחייבים אותה בשביל להגעה למקסימום // זה קשת אחת ספציפית, לדוגמא AB
כך שהשאלה שקיים להאם ניתן להגעה למקסימום ללא e.

אלגוריתם:

ופציה 1, 2 הרצות:

1. נרץ EK על G ונקבל $|F| \leftarrow$ זרימת מקסימום
2. נרץ על גרפ G לאחר ההורדה של e את EK, אם $|F'| = |F|$ אז יש זרימת מקסימום בלי e. אחרת – אין.

ופציה 2, הרצת 1:

1. נרץ EK על גרפ G ללא e ונקבל $|F| \leftarrow$ זרימת מקסימום
2. נוסיף את e ל-G ונרץ איטרציה EK
3. אם נקבל $|F|$ בצד (2), אז יש זרימת מקסימום בלי e. אחרת – אין.

נכונות: אם $|F'| > |F|$ אז יש זרימה חיובית על F ולא יהיה ניתן להגעה ל-F ללא e.

סיבוכיות:

- **ופציה 1:** 2 הרצות של EK, כלומר ($\Theta(m^2)$) 20
- **ופציה 2:** $O(\Theta(m^2)) + O(m)$ ← יותר עיל בקצב

• קבעו לאיזה מחלוקת שיצת השפה הבא והוכחו

ונדר שפה $\{(G, k, r) : G \text{ contains a Clique of size } r \text{ and a VC of size } k\}$ גורפים שיש בהם גם קliquה בגודל r וגם CISCI בגדלים k . כלומר, השפה L מכילה

צריך לחשב על השאלה בתור שפה בפני עצמה ולא 2 דברים מוכרים, אם בודקים בשפה קliquה בגודל R וקliquה בגודל K . מה עושים?

1. הטענה היא ש $CNP \subseteq L$, צריך להראות 2 דברים:

a. $L \subseteq NP$ למה?

i. הראנו ש NP סגורה ליחסון

ii. או בהינתן עדות קliquה בגודל R ו VC בגודל K ניתן לאמת את העדות בזמן פולינומי. $E + K + R^2$

b. $L = \text{הראות של } L \text{ יש רדוקציה פולינומית ל } A\text{. נרצה לקבל בעיה אחרת, לתרגם לשפה הצעת ואם הרשונה אומרת כן אז השפה הרצiosa גם אומרת כן, וההפר.}}$

2. ננסה להראות רדוקציה מ VC ל- L : $VC \leq_p L$

a. היקלט ב VC הוא גרף G, K וنم רץ את האלגוריתם $A(G, K)$, אז הכל בסדר. אחרת לא נבין למה האלגוריתם החזר.

b. ננסה להימנע מהבעיה שמקבלים לא.

נمز VC את האלגוריתם $A(G, 1, K)$, כלומר על קliquה בגודל 1 ← לכל גרף יש קliquה בגודל 1.

i. אם האלגוריתם אמר כן, אז בטוח יש VC בגודל 1.

ii. אם האלגוריתם אומר לא, אז הוא אומר לא בגודל K ולא בכלל גודל הקliquה הנתון (1).

תשובה נוספת:



נិיח את הגרף

על מנת לקבל קliquה בגודל R נדרש להוסיף $R-1$ קודקודים. אז נمز $A(G', R, K+R-1)$.

האלגוריתם:

1. נניח שיש אלגוריתם A ל- L : $A(G, R, K)$

2. נראה את הרדוקציה מ VC ל- L : $VC \leq_p L$

3. קלט: K וنم G, R ונمز $A(G, 1, K)$ //בעצם מה נשאל בשפה? האם G יש גם קliquה בגודל 1 וגם VC בגודל K . והאלגוריתם

שאנו מrzicim הוא תמיד T ולמעשה עונה על השאלה האם יש תמיד VC בגודל K.

← זה נכון בכלל שזאת שאלת יותר קשה מ- VC . ונותנת עדות על מקרה קטן.

• לcoli בעיה בידקו האם היא ב-P או ב-NPC

1. בהינתן קבוצה של מספרים שלמים חיוביים ושיוגים $\{a_1, a_2, \dots, a_n\}$, האם קיימת תת-קבוצה של A , שסכום איבריה שווה ל- $1,000$?

2. בהינתן עץ $(V, E) = T$, האם יש לו מסלול המילטון?

. הסעיף הנ"ל מגדיר את subset sum .

סוכם S \leftarrow NPC

מסלול המילتون \leftarrow NPC.

מספר התתי קבוצות הוא 2^{1000} . זה מספר עצוםא בל הוא קבוע. لكن אפשר תמיד לדעת כמה איטרציות יש להריץ על הבעיה, ולכן היא ב-P.

**כל שפה סופית ניתנת להכרעה באוטומט, ולכן כל שפה כזאת היא ב-P.

**כל שאלה על משאנו קבוע היא ב-P. //

2. במקרים אחרים שואלים אם העץ הוא שרשות. בכל מקרה אחר זה לא יעבוד.

אם הדרגה של כל קודקוד היא 2 לפחות שני העלים .

*מסלול המילتون היא בעיה שהיא NP קשה, על גוף כללי היא NP קשה. **כשմבטיחסים שזה עץ, המילتون היא P.**

↳ בעיה ב-P זו בעיה שאפשר לבנות לה אלגוריתם שרצה בזמן קבוע (כמו $O(1)$).

VC בגודל 7 זה קל, VC בגודל N זה קשה.

שאלה במספר כללי של הרוצת זו שאלה קשה.

• מה אם נסיר את ההגבלה שמהספרים ב-1 הם חיוביים? כלומר נתונה סידרה של N מספרים האם יש תת-קבוצה שסכוםה 1000?

NPC, נראה שהיא NP ואז נעשה רדוקציה משפה אליה.

1. בהינתן עדות הקבוצה שסכוםה הוא 1000 קל לבדוק.

2. נמצא שפה L' שהוא NPC ונראה רדוקציה אליה.

3. יש רדוקציה דרך SAT (לא רלוונטי לנו).



הופיכו או הפריכו את הטענה הבאה:

$(\text{co-NP})\text{-complete} = \text{co-(NP-complete)}$

מה כתוב כאן?
שפה שלמה בCONP היא בעצם המשלים של שפות NP שלמות.

במילים אחרות NOT SAT היא CONP שלמה.

$$\overline{L} \in \text{NP-C} \iff L \in \text{NP-C}.$$