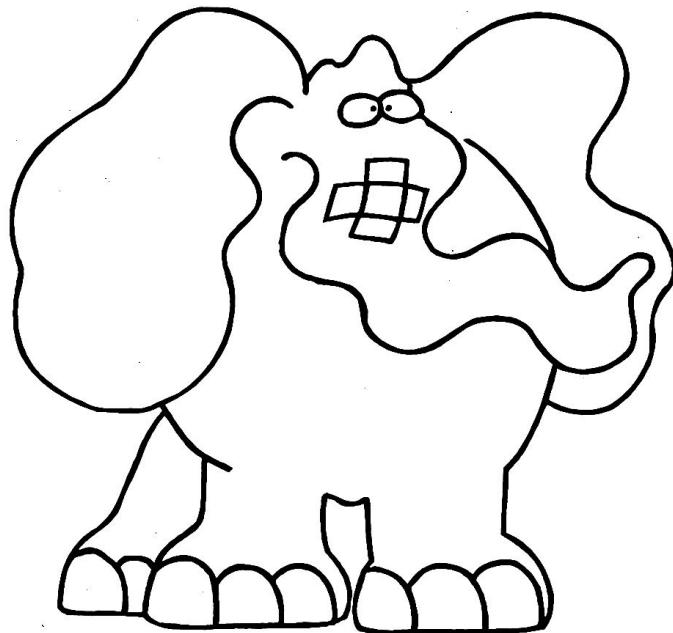


אלגוריתמים

תשפ"א



מאת: מיכל פרנס ואmir בן עמרם
עדכנים ותוספות: איתן לשם, עדי עקבה, איריס גבר, שלומית אריאן

לשימוש פנימי בלבד

מהדורה: 2021, סטודיו ב' תשפ"א



©

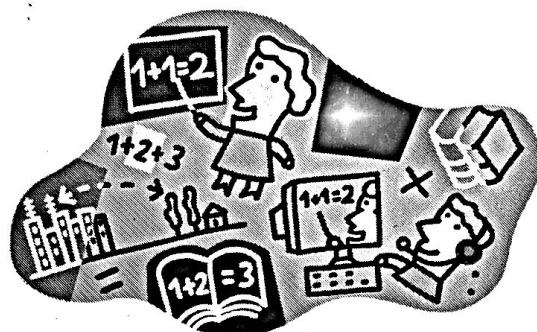
מיכל פרנס, אמיר בן-עמרם
כל הזכויות שמורות למחברים

אין להעתיק, לצלם, לאחסן במאגר מידע, להפיצו או לפרסם מחדש
שום חלק מהחברת זו ללא אישור בכתב מהמחברים.

תוכן עניינים

עמוד	נושא
1	מושג האלגוריתם וסוגי אלגוריתמים
12	מושגים מתורת הגרפים
20	מישון טופולוגי
25	מעגל אוילר בגרף
39	חיפוש عمוק בגרף
66	חיפוש רוחב בגרף
77	מסלולים בגרפים עם משקלים
114	כפל מהיר של מספרים ומטריצות
120	תכנונות דינמי
134	עץ פורש מינימלי
147	זרימה ברשתות ובעיית היזוגים
179	התאמת מחרוזות, אוטומטים סופיים ושפות רגולריות

מושג האלגוריתם וסוגי אלגוריתמים



הבעיה האלגוריתמית

בעיה אלגוריתמית:

תאור של הקלט המצופה (מופע הבעיה) והגדרת הפלט הרצוי.

פתרון לבעיה: אלגוריתם - שיטה איך לספק לכל קלט חוקי את הפלט הרצוי עבورو.

אלגוריתם נכון: פותר בצורה נכוןה את הבעיה הנתונה עבור כל קלט אפשרי.

אלגוריתם לא נכון: לא עוצר על קלטים מסוימים, או שהפלט שגוי.

סוגי אלגוריתמים

אלגוריתם מקבילי או מבוזר. אלגוריתם סדרתי.

אלגוריתם הסתברותי. אלגוריתם דטרמיניסטי.

הצעדים בפתרון בעיה

- קביעת הקלט והפלט הנדרשים.
- הגדרת מודל מתמטי מופשט לייצוג הבעיה.
- מציאת אלגוריתם לפתרון הבעיה.
- תיאור עליי (לרוב מילולי)
- תיאור מפורט (פסאudo-קוד + שימוש במבני נתונים מתאימים)
- בדיקות (דוגמאות) והוכחת נכונות.
- ניתוח סיבוכיות זמן וזיכרון.
- ניסוי מעשי.

תיאור אלגוריתמים

שימוש בפסאודו-קוד :PseudoCode

דומה לשפת תכנות, אולם משתמשים בשפה רגילה לתיאור שלבים מסוימים.
מתעלמים מטיפול בשגיאות קלט ובუות תכונות אחרות.

סימונים מקובלים:
 $a = b$ השוואה
 $a \leftarrow b$ השמה

שימוש בשיטופי נתוניים מופשטים .ADT, Abstract Data Type

ניתוח יעילות אלגוריתם

- זמן ריצה וזכרון כפונקציה של גודל הקלט.
- ספירה של מספר הפעולות הבסיסיות.
- אסימפטוטיקה של פונקציות - O, Θ, Ω .
- זמן ריצה:

המקרה הגורע ביותר ביותר

Best Case

המקרה הטוב ביותר (מהו המודל הסטטיסטי של הקלט?)

אלגוריתם פולינומייאלי ⇔ אלגוריתם לא פולינומייאלי
(אקספוננציאלי - מעריכי)
(לינארי, ריבועי)

דוגמה: מיון מספרים

קלט: סדרה של n מספרים (a_1, a_2, \dots, a_n) .

פלט: חמורה $(a'_1, a'_2, \dots, a'_n)$ של המספרים שבה $a'_1 \leq a'_2 \leq \dots \leq a'_n$.

אלגוריתם: מיון بواسות.

מבנה נתונים: מערך של מספרים.

פסאודו-קוד של מיון بواسות

Input: $A[1, \dots, n]$

Output: A sorted in increasing order.

BUBBLE-SORT(A)

```
for i ← 1, ..., n do
    for j ← 1, ..., i-1 do
        if A[j] > A[j+1] then
            swap A[j], A[j+1]
        if (no swaps done) then
            stop;
```

יעילות: במקרה הכללי: $O(1+2+3+\dots+n) = O(n^2)$
במקרה הגרוע: $\Theta(n^2)$ דוגמה: סדרה ממונת יורדת.
במקרה הטוב: $\Theta(n)$ דוגמה: סדרה ממונת עולה.

ניתוח יעילות של אלגוריתמים

מכאן ואילך: יעילות האלגוריתם = זמן ריצה במקרהorst case.

למשל: יעילות מילון בוועות היא $\Theta(n^2)$



שיטות לתוכנון אלגוריתמים

- רדוקציה
- אלגוריתם חמדני (Greedy Algorithm)
- הפרד ומשול (רקורסיה, Divide and Conquer)
- תכנות (תוכנון) דינامي (Dynamic Programming)
- שיפור הדרgtiy
- Backtracking
- Branch and Bound •

רדוקציה

הרעיוון:

- "מתרגמים" את הבעיה לבעה אחרת שאורה יודעים לפתור.
- פותרים את הבעיה המתורגם.
- "מתרגמים" את הפתרון של הבעיה השנייה לפתרון לבעיה המקורית שלנו.

תהליך התרגום צריך להיות עיל.

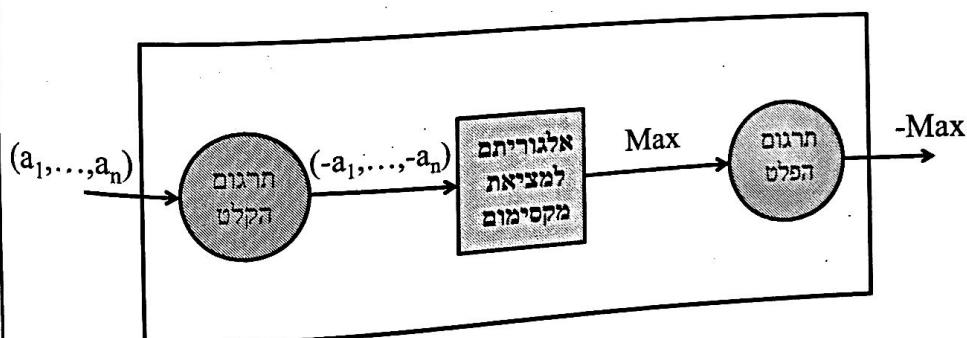
דוגמה: נתון אלגוריתם A למציאת מקסימום בין מספרים.
ברצוננו למצוא מינימום בין המספרים.

אלגוריתם:

1. נכפיל כל מספר ב- -1
2. נמצא מקסימום Max מבין כל המספרים החדשים בעזרת A
3. הפלט יהיה: -Max

רדוקציה

אלגוריתם למציאת מינימום

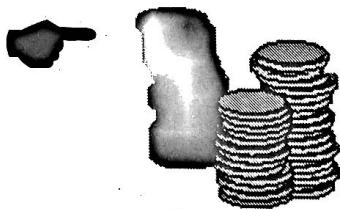


ב証明ה הנכונות מtabסים על נכונות האלגוריתם למציאת מקסימום (קופסה שחורה).

השיטה החמדנית

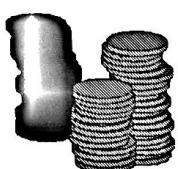
הרעיון: פתרון של בעית אופטימיזיה, כאשר בכל שלב נבצע פעולה שנותנת את השיפור (הרווח) המיידי הגדול ביותר.

הסנה: לא תמיד נגיע לאופטימום.



דוגמאות בקורס: האלגוריתם של דיקסטרה למסלולים מינימליים במשקל, האלגוריתמים של קרטוסקל ופרים למציאת עץ פורש מינימי.

דוגמה: סכום במינימום מטבעות



יש מטבעות של שקל, 50 אגורות, 10 אגורות, 5 אגורות, אגורה
(אין מחסור באף מטבע).

קלט: סכום N (מספר שלם של אגורות)
המטרה: לפרק את N לכמה שפותחות מטבעות.

אלגוריתם:
בכל שלב נבחר מטבע מסוימי שערך לא עבר את הסכום הנותר.

נכונות: האופטימליות תלויות בערכי המטבעות! (וגם עצם קיומו של פתרון...)

לא תמיד אופטימלי: סכום נדרש: 8. מטבעות: 1,4,6.

שיטת הפרד ומשול (ركورסיה)

הרעיון:

1. מחלקים את הבעה לכמה בעיות קטנות יותר מאותו סוג.
2. פותרים כל אחת מהבעיות הקטנות באותה גישה.
3. מצרפים את הפתרונות החלקיים לפתרון הבעיה המקורית.

"Top-down" ⇐

ניתוח זמן הריצה: בעזרת נוסחה רקורסיבית.

דוגמיה: אם הבעיה מתפרקת לשתי בעיות שגודלה חצי מהבעיה המקורית,
ומחריר הפירוק וצירוף הפתרונות הוא $f(n)$ או זמן הריצה הוא:

$$T(n) = 2T(n/2) + f(n)$$

דוגמה: MergeSort

אלגוריתם:

- נמיין חצי המספרים.
- נמיין את החצי השני של המספרים.
- נמזג את שתי הרשימות הממוינות.

זמן ריצה: $T(n) = 2T(n/2) + \Theta(n)$

פתרון הנוסחה הרקורסיבית:
 $T(n) = \Theta(n \log n)$

דוגמה: חישוב מקדים ביןומים

הבעיה: קלט: מספר חיובי n . פלט: לכל $0 \leq k \leq n$ אנו רוצים לדעת כמה קבוצות של k אברים אפשר לבחור מתוך n אברים שונים.

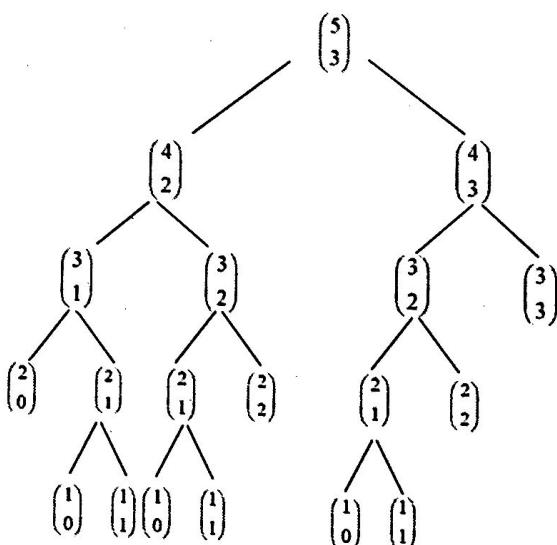
$$0 < k < n \quad \binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

זהות פסקל:

$$\binom{n}{n} = \binom{n}{0} = 1$$

נראה שסיבוכיותו של אלגוריתם רקורסיבי פשוטי היא מעריכית.

דוגמה לעץ הרקורסיבי



$$n = 5, k = 3$$

$$\text{סיבוכיות לפחות: } \binom{n}{k}$$

יש תת-עצים שלמים שמחושבים שוב ושוב!

$$\binom{n}{n/2} \approx \frac{2^n}{\sqrt{n}}$$

תכנות דינמי

בגישה הפרד ומשול לעתים פותרים את אותה בעיה קטנה כמה פעמים
בגל העבודה Top-Down.

הפתרונות:
מחילה מפתרון הבעיה הקטנה, שומרם את הפתרונות החלקיים
וממשתמשים בהם לקבלת פתרון לבעיות הולכות וגדלות.

"Bottom-Up" ↘

דוגמאות בקורס: האלגוריתם של פלויד למציאת מסלולים מינימליים במשקל,
סגור טרנזיטיבי של גרף.

חישוב ע"י תכנות דינמי

نبנה טבלה ונמלא אותה שורה שורה בעזרת זהות פסקל.

	0	1	2	...	$k-1$	k	n
0	1						
1	1	1					
2	1	2	1				
$n-1$					$\binom{n-1}{k-1}$	$\binom{n-1}{k}$	
n						$\binom{n}{k}$	

זמן ריצה: $\Theta(nk)$

זיכרון: גודל הטבלה $\Theta(nk)$. ניתן ב- $\Theta(k)$, נשמר שורה באורך k ונעדכן מימין.



שיעור הדרגתית

הרעיון: מתחילה מ对照检查 התחלתי כלשהו ובכל שלב משפרים את הפתרון.

דוגמאות: אלגוריתם למיון.

הגדרה: ג.ו זוג רע אם אינו בסדר הנכון.

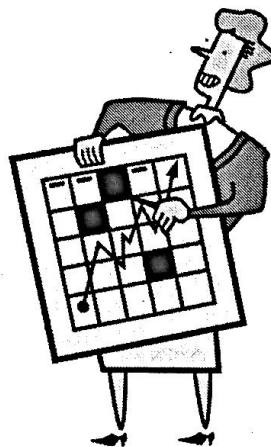
אלגוריתם: כל עוד יש זוגות רעים
נמצא זוג רע ונחליף אותו.

טענה 1 (נכונות חלקית): אם החישוב הסתיים, המערך ממון.

טענה 2 (עצירה): לאחר $\binom{n}{2}$ החלפות לכל היותר, החישוב מסתיים.

דוגמאות בקורס: פורד - מסלולים מינימליים במשקל. פורד-פלקרסון - זרימה ברשותות.

מושגים מתורת הגרפים



מושגים בסיסיים על גרפים

גרף $G = (V, E)$

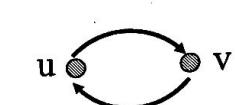
V קבוצת הנקודות (vertices) או הצלמים
E קבוצת הצלעות (edges) או הקשתות (arcs).



קשת

צלע

(u, v) ו- (v, u) הן קשתות שונות ונקראות אנטי-מקבילות.



במולטי-גרף יתכנו צלעות/קשתות מקבילות

בגרף פשוט אין צלעות/קשתות מקבילות ואין לו לאות עצמיות (v, v)
בגרף פשוט יתכנו קשתות אנטי-מקבילות!



u → v שcn של u אם יש קשת מ- u אל v.

דרגות של קדקודים

בגרף לא-מכוון:

דרגת קדקוד $d(v)$ = מספר הקצחות של קשתות הנוגעות ב- v .

בגרף מכוון:

דרגת כניסה $d_{in}(v)$ indegree = מספר הקשתות הנכנסות ל- v .

דרגת יציאה $d_{out}(v)$ outdegree = מספר הקשתות היוצאות מ- v .

מקור source הוא קודקוד שדרגת הכניסה שלו היא אפס.

בור sink הוא קודקוד שדרגת היציאה שלו היא אפס.

הקשר בין מספר קדקודים לצלעות

בגרף לא-מכוון: $\sum_{v \in V} d(v) = 2 |E|$

בגרף מכוון: $\sum_{v \in V} d_{in}(v) = \sum_{v \in V} d_{out}(v) = |E|$

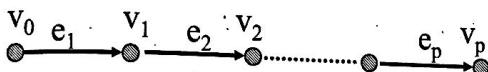
n = מס' הקודקודים, m = מס' הצלעות/קשתות

בגרף פשוט לא מכוון: $m \leq n(n-1)/2$ בגרף פשוט מכוון: $m \leq n(n-1)$

בשני המקרים מתקיים $m \leq n^2$, כלומר $O(n^2) = m$ (גם אם יש לו לאות עצמאיות)

מסלולים בגרפים

מסלול: סדרת קדקודים וקשתות $(v_0, e_1, v_1, e_2, \dots, e_p, v_p)$ כך שלכל $i \leq p$ היא קשת מ- v_{i-1} ל- v_i .



בגרף פשוט נרשם רק את הקדקודים (v_0, v_1, \dots, v_p)

סימונים: $v_0 \rightarrow v_1$ $v_0 \rightsquigarrow v_p$

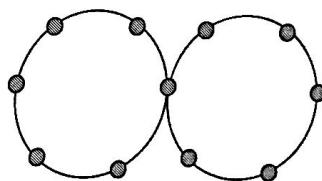
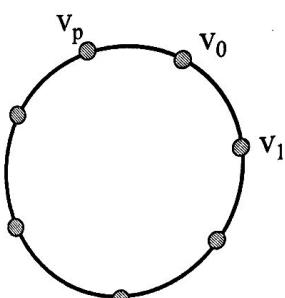
מסלול פשוט אם כל הקדקודים לאורכו שונים זה מזה.
אורך מסלול = מספר הקשתות בו.

גם בגרף לא מכוון, למסלול יש כיוון.

מעגלים בגרפים

מעגל cycle הוא מסלול שבו לפחות קשת אחת, והקדקוד האחרון זהה הראשון.
במעגל פשוט אין (פרט לכך) שום חזרות על קדקודים.
במעגל פשוט מספר הקדקודים = אורך המרجل.

למעגל לא-דווקא פשוט קוראים גם סיור tour או circuit.

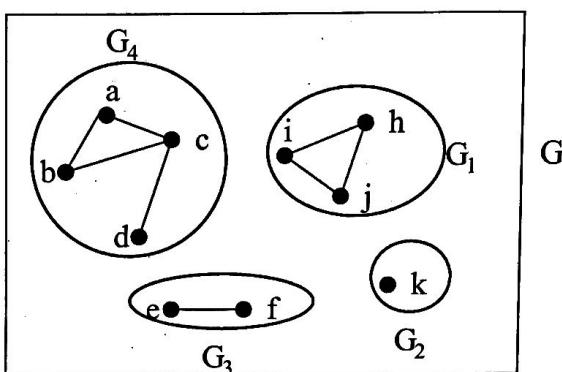


граф אציקלי - גרף חסר מעגלים.
graph acyclic (directed acyclic graph) DAG - גראף מכוון אציקלי.

קשירות בגרף לא-מכוון

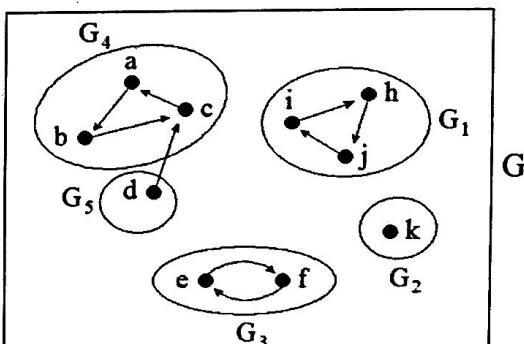
קשירות: גרף לא-מכוון נקרא קשור אם בין כל שני קודקודים יש מסלול.
(בגרף קשור מתקיים $1 \leq m \leq n$)

רכיבי קשירות – חלוקה של הגרף לפי מחלקות השיקילות
של היחס $v \sim u$



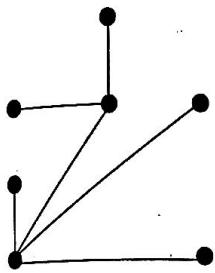
קשירות חזקה בגרף מכובן

קשירות חזקה - גרף מכובן נקרא קשור חזק אם בין כל שני קודקודים יש מסלול מכובן.
רכיבי קשירות חזקה – **strongly connected components** – חלוקה של
הגרף לפי היחס $(v \rightarrow u \text{ ו גם } u \rightarrow v)$

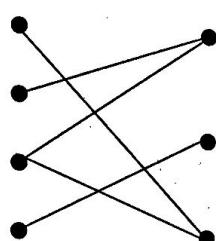


המושג "קשור" מתייחס לקשרות לא-מכובנת.
אם משתמשים בו בקשר ל그래ף מכובן הכוונה היא לקשרות של הגרף ללא הכוונים.

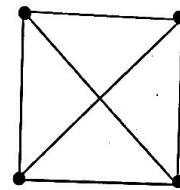
גרפים מיוחדים



עץ (לא מכוון)
(undirected) tree



граф דו-צדדי
bipartite graph



graf שלם (K_4)
complete graph

עץ = גרף לא מכוון קשור וחסר מעגלים.
יער = אוסף עצים = גרף לא מכוון חסר מעגלים (קשר או לא).

הגרף כ-ADT

פעולות בסיסיות:

MakeGraph(n)
NeighborList(u)
AreNeighbors(u, v)
AddEdge(u, v)
DeleteEdge(u, v)

- ליצור גרף ריק עם n קודקודים.
- מי כל השכנים של קודוד?
- האם צלע מסוימת קיימת בגרף?
- הוספה צלע
- הורדת צלע

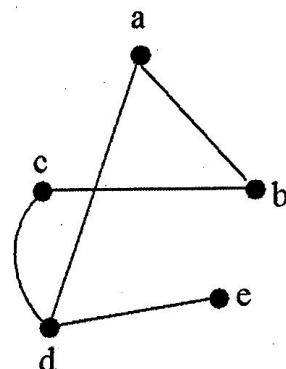
מבנה נתונים:

- מטריצה שכנות או סמיכויות .Adjacency Matrix
- רשימה שכנות או סמיכויות .Adjacency List

מטריצה שכנויות/סמיכויות

Adjacency Matrix

	a	b	c	d	e
a	0	1	0	1	0
b	1	0	1	0	0
c	0	1	0	1	0
d	1	0	1	0	1
e	0	0	0	1	0



זיכרון: $\Theta(n^2)$

לפעמים נשמר גם שדה משקל לכל צלע.

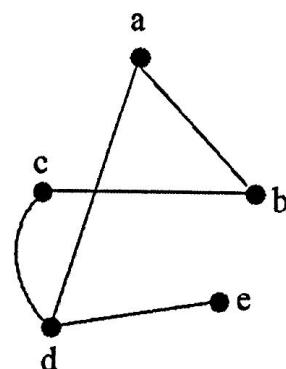
יעיל לגרף צפוף dense.

רשימות שכנויות/סמיכויות

Adjacency Lists

Adj

- | | | | | | | | | |
|---|---|---|---|------|---|------|---|------|
| a | → | b | → | d | → | NULL | | |
| b | → | a | → | c | → | NULL | | |
| c | → | d | → | b | → | NULL | | |
| d | → | a | → | e | → | c | → | NULL |
| e | → | d | → | NULL | | | | |



לפעמים נשמר גם:

- מצבים דו-כיווניים בין שני "העותקים" של אותה צלע.
- שדה משקל לכל צלע.

זיכרון: $\Theta(n+m)$

יעיל לגרף דליל sparse.

השוואת מבני הנתונים

	רשימות שכנות	מטריצה שכנות
MakeGraph(n)		
AreNeighbors(u, v)		
NeighborList(u)		
AddEdge(u, v)		
DeleteEdge(u, v)		
זיכרון		

דוגמה: הדפסת כל הצלעות של גרף

```

for each vertex  $v \in V$ 
  for each  $u \in \text{Adj}[v]$ 
    print  $(u, v)$ 
  
```

יעילות:

מטריצה שכנות: $\Theta(n^2)$
 רשימת שכנות: $\Theta(n+m)$

אלגוריתם שיעילותו $O(n+m)$ הוא אלגוריתם ליניארי.

תרגילים

- ידי (V, E) = גרף מכוון פשוט. הגרף ההפוך **Transpose** של G הוא הגרף $(V, E^T) = G^T$ כאשר $E^T = \{(v, u) \mid (u, v) \in E\}$

כתבו אלגוריתם המחשב את G^T מ- G באופן הייל ביותר.

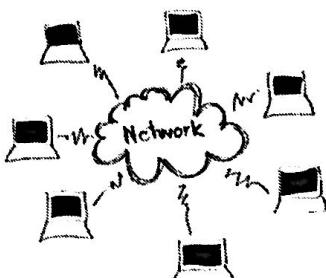
מהי סיבוכיות האלגוריתם שכתבתם כאשר:

1. G מיוצג ע"י מטריצת סמוכיות.
2. G מיוצג ע"י רשימה שכנוויות.

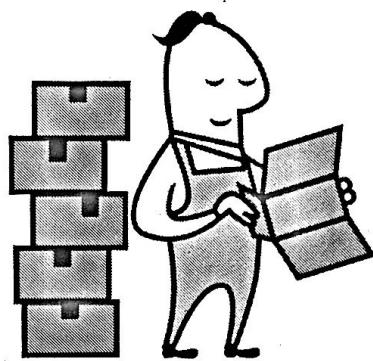
- הוכיחו שבכל גרף מכוון אציקלי יש לפחות קדקוד אחד שהוא מקור.

דוגמאות לשימושים בגרפים

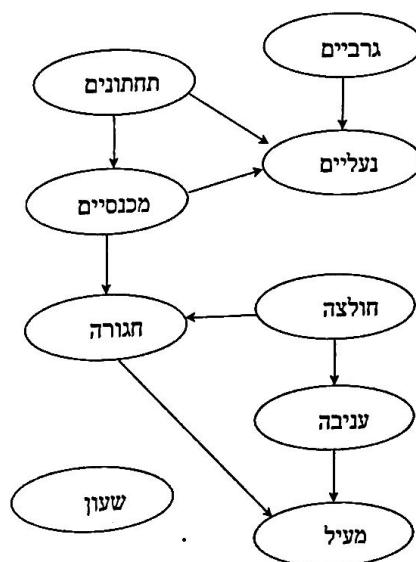
- רשת מחשבים.
- כיצד לשלוח הודעה באינטרנט בדרך הקצחה והמירה ביותר.
- העברת הודעה לכל המחשבים בראשת.
- מציאת מסלול חלופי להעברת הודעה כשייש תקלת בראשת.
- כיצד להרוויח כסף בבורסה.
- בעיית השיזוקים.



מיון טופולוגי



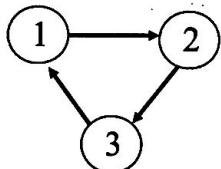
דוגמה



מיון טופולוגי

הגדרת הבסיסית קלה: גוף מכובן אציקלי פשוט ($G = (V, E)$) יש למצאו סידור ליניארי של כל קדקודיו G , כך שאם $v \rightarrow u$ אז u יופיע לפני v בסידור.

הערה: אם G אינו אציקלי או אין מיון טופולוגי של קדקודיו.
הוכחה: אם יש מעגל אז אף קדקוד אינו יכול להיות ראשון במיון.



שימושים:

- באיזה סדר למדוד את הקורסים במלילה כך שיתקיים כל דרישות הקדם.
- באיזה סדר יש להרכיב את חלקה של מכונית.
- כיצד תלבש הפרופס/or המפוזר בبوك.

אלגוריתם

רעיון בסיסי:

כל עוד יש קדקודים בגרף:
נמצא מקור, נdfs אותו, ונוריד אותו ואת קשתותיו מהגרף.

שאלות:

1. האם מובטח לנו שבכל שלב יימצא מקור בגרף?

2. אם אכן יש מקור, כיצד נמצא אותו בצורה יעילה?

3. מתי נעצור את התהליך?

4. וכמו כן, האם האלגוריתם נכון ומה ייעילתו...

אלגוריתם למון טופולוגי

מבנה נתונים:

- Q – תור למקורות
- $[n]$ – מערך דרגות הכניסה של הקדוקודים.

אלגוריתם:

- נמצא לכל קדוקוד את דרגת הכניסה שלו.
- נכנס לתור את כל הקדוקודים שדרגת כניסה שלהם 0 (מקורות).
- כל עוד התור לא ריק:
 - נויריד קדוקוד מראש התור וננדפיס אותו.
 - מעבר על כל אחד משכניו ונקטין, באחד את דרגת הכניסה שלהם.
 - אם דרגת הכניסה של קדוקוד כלשהו התאפסה - נכנס אותו לתור.
 - אם יש בסיום קדוקוד שדרגת כניסה > 0 אז נודיעו שיש מעגל.

```
TOPOLOGICAL-SORT(Graph G)
Queue Q           // Queue of sources.
int indegree[n]  // Array of indegrees of vertices

for each v  $\in$  V do          // INIT
    indegree[v]  $\leftarrow$  0
for each (u,v)  $\in$  E do
    indegree[v]  $\leftarrow$  indegree[v]+1
for each v  $\in$  V do
    if indegree[v]=0 then Q.Enqueue(v)

while Q  $\neq \emptyset$  do          // MAIN LOOP
    u  $\leftarrow$  Q.Dequeue()
    print u
    for each v  $\in$  Adj[u] do
        indegree[v]  $\leftarrow$  indegree[v]-1
        if indegree[v]=0 then Q.Enqueue(v)

for each v  $\in$  V do          // check if all nodes reached
    if indegree[v]  $\neq$  0 then
        print "Cycle!"; stop
```

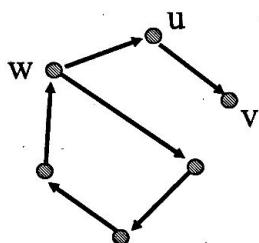
מה היעילות?

הוכחת נכונות

טענה 1: אם אין ב- G מעגל - האלגוריתם מוצא מין טופולוגי.

הוכחה: נניח שב- G אין מעגלים ונווכה:

1. כל קדוקי הגרף יודפסו.
2. כל קדוק יודפס לכל היותר פעם אחת.
3. סדר ההדפסה נכון.



כל קדוקי הגרף יודפסו:

נניח בsvilleה ש- v לא הודפס. לכן דרגתו לא התאפסה.

◀ קיימים $v \rightarrow u$ ו- $v \rightarrow w$ וגם v לא הודפס. נמשיך כך ונקבל מעגל (כי מספר הקדוקים סופי ולכן נחזור לקדוק שכבר היו בו).

זו סתירה להנחה – לכן לא קיימים קדוק שלא הודפס.

כל קדוק יודפס לכל היותר פעם אחת:

נראה שככל קדוק נכנס לثور לכל היותר פעם אחת.

(1) לכל קדוק v : ערכי $\text{indegree}[v]$ הם סדרה מונוטונית יורדת (להלן) בזמן.

(2) קדוק v נכנס לثور באחד מ-2 מצבים:

- בזמן האתחול, אם $\text{indegree}[v] = 0$.
- בזמן הולאה, כאשר $\text{indegree}[v] \neq 0$.

(3) יהיו v_0 קדוק כלשהו בגרף:

- אם $\text{indegree}(v_0) = 0$, אז הוא ייכנס לثور בזמן האתחול ולא ייתכן שייכנס לثور שוב.
- אם $\text{indegree}(v_0) > 0$, אז $\text{indegree}(v_0)$ יהיה מ-1 ל-0 לכל היותר פעם אחת במהלך הולאה ולכן ייכנס לثور לכל היותר פעם אחת.

סדר ההדפסה נכון:

תהי $v \rightarrow u$ צלע.

לפי האלגוריתם v לא יכול להיכנס לثور לפני שנעבור על רשימת השכנים של u .

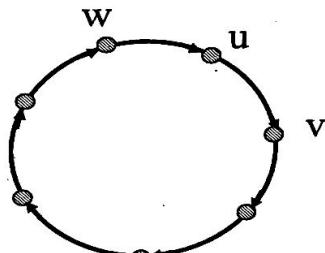
נעבור על רשימת השכנים של u רק אחרי v – v יוצא מהثور.

v נכנס לثور אחרי v – v יצא ממנו. ← v יודפס אחרי v . ←

המשך הוכחה

טענה 2: אם ב- G יש מעגל - האלגוריתם ידפיס Cycle.

הוכחה: נניח שיש מעגל. נוכיח שדרגת הכניסה של כל הקדקודים במעגל לא תתאפס
ולכן יודפס Cycle.



נניח בשלילה שדרגת הכניסה של קדקוד v
במעגל התאפסה.

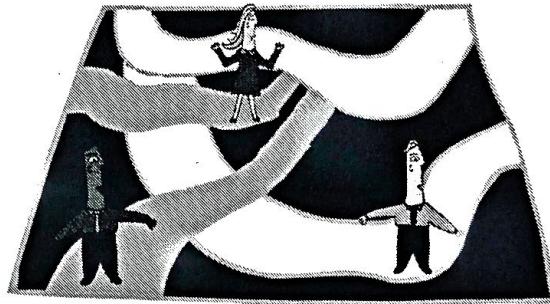
יהי u הקדקוד שנמצא לפניו על המעגל.

↖ u הוכנס לטור לפני v (בדומה להוכחת טענה 1).
נשיד כך לאורך המעגל ונגיע למצב ש- v הוכנס לטור לפני u , וזה סתירה.

תרגילים

- האם אפשר להשתמש במחסנית במקום בתור?
- מה המספר המקסימלי של קדקודים שיכולים להיות בתור בעת ובעונה אחת?
בailo גראפים זה יקרה?

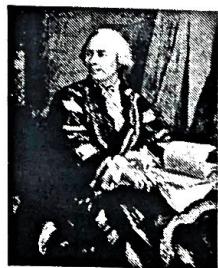
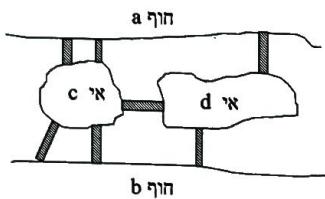
מעגל אוילר בגרף



מעגל אוילר

בעיית הגשרים של קניגסברג Königsberg

האם ניתן לבצע טויל מעגלי בעיר,
כך שלא כל גשר עוברם בדיק פעם אחת?



לאונרד אוילר (1707-1783):
מתמטיקאי ופיזיקאי שוודי, שנחשב
למוביל בתחוםו במאה ה-18.

אוילר הוכיח בשנת 1735 כי הטויל אינו אפשרי.
זה היה המאמר הראשון בתורת הגרפים.

משפט לمعال אואילר

معال אואילר Euler Tour:معال שմבקר בכל קשת בגרף בדיקות אחת (אולם יכול לבקר בקדקוד יותר מפעם אחת).

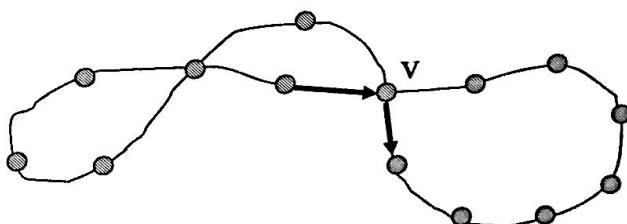
משפט (לגרף לא מכובן):יהי G גרף לא מכובן פשוט וקשיר. אז מתקיים:
יש ב- G معال אואילר \leftrightarrow דרגת כל קודקוד G זוגית.

משפט (לגרף מכובן):יהי G גרף מכובן פשוט וקשיר חזק. אז מתקיים:
יש ב- G معال אואילר \leftrightarrow לכל קדקוד v מתקיים: $d_{in}(v) = d_{out}(v)$.

גרף מכובן – הכרחיות של תנאי הדרגות

יהי G גרף מכובן ונניח כי קיים ב- G معال אואילר.

יהי v קדקוד כלשהו.
כל מעבר של המעלן דרך v מכסה כניסה אחת ל- v ויציאה אחת.



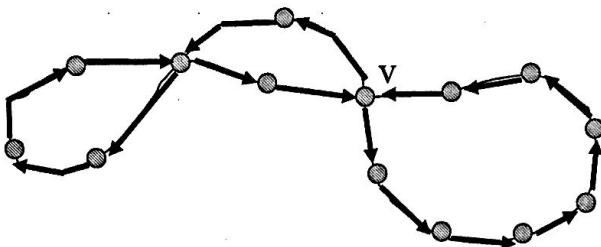
זהمعال אואילר והוא מבקר בכל הקשות, ולכן ייספרו כל הכניסות וכל היציאות.
אבל מספר הכניסות שנספרו שווה למספר היציאות שנספרו.

$$d_{in}(v) = d_{out}(v)$$

גרף לא-מכוון – הכרחיות של תנאי הדרגות

יהי G גרף לא-מכוון ונניח כי קיים ב- G מעגל אוילר.

אם נתן לכל צלע כיוון לפי המעגל, נקבל גרף מכוון שבו לכל קדקוד שמתקיים $d_{in}(v) = d_{out}(v)$.



ולכן v בעל דרגה זוגית. $d(v) = d_{in}(v) + d_{out}(v)$

הוכחת מספיקות (לגרף לא-מכוון)

נניח ש- G קשור ולכל קדקודיו דרגה זוגית ונראה אלגוריתם שモציא מעגל אוילר ב- G .

נתאר תחילה פונקציה $\text{Find-Circuit}(v_0)$ שמוציאה מעגל שמתחל ומסתיים בקדקוד נתון v_0 .

הרעיון: נצא מהקדקוד v_0 ונטיל בgraf תוך כדי שמירת הכלל שלא לשוב על אותה צלע פעמיים.

Find-Circuit

FIND-CIRCUIT(Vertex v0)

// Find a circuit starting at v0. Return list of vertices.

v \leftarrow v0
List L $\leftarrow \langle v0 \rangle$ // initialize list

repeat
 u \leftarrow a neighbour of v via an unused edge
 mark (v,u) used
 L.Append(u)
 v \leftarrow u
until v has no unused edge
return L

נכונות Find-Circuit

טענה: אם יוצאות מ- v_0 צלעות וכל הדרגות בגרף G זוגיות,
או הפונקציה מוצאת מעגל כנדרש.

הוכחה:

• התהיליך מסתיים:

מספר הצלעות בגרף סופי, ומבקרים בכל צלע לכל היתר פעם אחת.

• התהיליך מסתיים בבדיקה v_0 :

נניח בשילhouette שהטהיליך מסתיים בבדיקה $x \neq v_0$.

כל מעבר של המסלול ב- x (כניסה ויציאה) סימן 2 צלעות המכילות את x ,
פרט לצעד האחרון שסימן רק כניסה.

אבל כל הדרגות בגרף זוגיות ולכן יש $-x$ צלע לא מסומנת,
בסתירה לכך שהטהיליך מסתיים ב- x .

אלגוריתם למציאת מעגל אוילר

אלגוריתם:

- נתחיל בבדיקה כלשהו v_1 ונמצא מעגל L שמתחליל בו.
- כל עוד יש ב- L קדקוד עם צלע יוצאת לא-מוסמנת:
 - יהיו v קדקוד ראשון כזה ב- L .
 - נמצא מעגל L_1 שמתחליל ב- v .
- נבדיק את L_1 לתוך L במקום הבדיקה v .
- נחזיר את L .

אלגוריתם למציאת מעגל אוילר

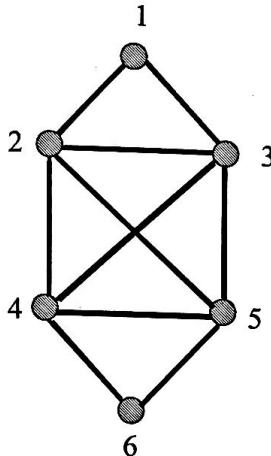
EULER(Graph G) // Find an Euler circuit.

L \leftarrow FIND-CIRCUIT(v_1)

while there is a vertex in L with unused edges
 $v \leftarrow$ first such vertex in L
 $L_1 \leftarrow$ FIND-CIRCUIT(v)
 “paste” L_1 into L instead of v

return L

דוגמת הרצאה



שלב 1: נמצא את $(1,2,3,1)$.

שלב 2: נמצא את $(2,4,5,2)$.
לאחר הדבקה נקבל $(1,2,4,5,2,3,1)$.

שלב 3: נמצא את $(4,6,5,3,4)$.
לאחר הדבקה נקבל $(1,2,4,6,5,3,4,5,2,3,1)$.

Euler נכונות

משפט: אם הגרף G קשור וכל הדרגות זוגיות, האלגוריתם מוצא מעגל אוילר.

הוכחה:

- 1) כל קריאה ל- Find-Circuit תצליח
 - הגרף שנותר לאחר הורדת המעלג הוא גרף שבו שכל דרגותיו זוגיות.
 - קדקוד ההתחלה נבחר כך שיש צלע לא מסומנת היוצאת ממנו.

2) האלגוריתם יסתהים כי מספר הצלעות סופי, ובכל שלב אנו מסמנים מעגל נוסף.

3) L תמיד מכילה מעגל

4) המעלג יכול את כל הצלעות
נניח בשלילה שיש צלע (v,u) שלא ביקרנו בה, אז קדקודים אלו אינם בראשימה T. מצד שני הגרף קשור ולכן יש מסלול מ- v ל- u . ומכאן שיש קדקוד ב- L עם צלע לא מסומנת.

מעגל אוילר בגרף מכוון

ברמת פסאודו-קוד — אין שום הבדל.

ההוכחות שונות במקצת.

ניתוח יעילות

EULER(Graph G) // Find an Euler circuit.

L \leftarrow FIND-CIRCUIT(**v1**)

while there is a vertex in L with unused edges

v \leftarrow first such vertex in L

L1 \leftarrow FIND-CIRCUIT(**v**)

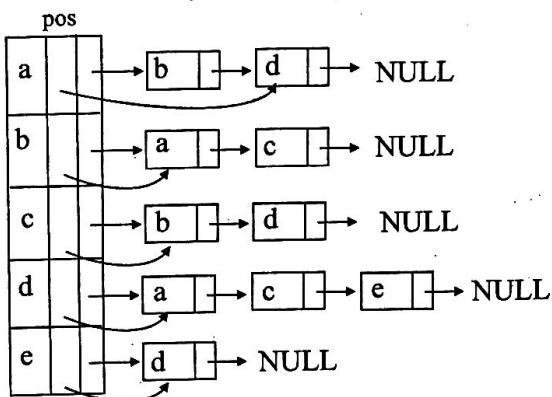
“paste” **L1** into **L** instead of **v**

return L

(1) זמן ריצה מוגבר של FIND-CIRCUIT

(2) זמן ריצה של Euler מוחז ל-

מבנה נתונים ויעילות: גרף מכוון

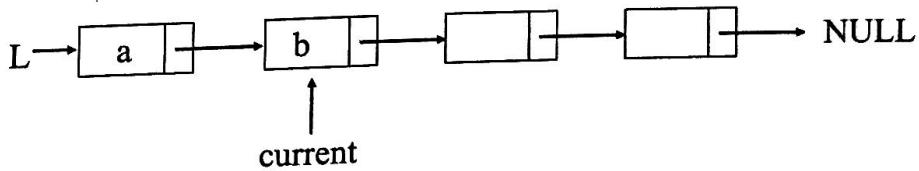


- לכל קדוקוד: מצביע pos לקשת הלא-מוסומנת הבאה.

מצביי pos זים רק קדימה בכל רשימת שכנים, ולכן סה"כ הם זים m פעמים.

\Leftarrow זמן ריצה כולל ב- FIND-CIRCUIT $\Theta(n + m)$

מבנה נתונים ויעילות – אחזקה L



נשמר מצביע current לקדוקוד הראשון ב- L שעדין יוצאה ממנו קשת לא מסומנת. המצביע יקודם בשורה האומרת "מצא קדוקוד ב- L עם קשת יוצאה לא-מוסומנת".

בסה"כ current יתקדם m פעמים.
 \Leftarrow זמן ריצה כולל מחוץ ל- FIND-CIRCUIT $\Theta(m)$

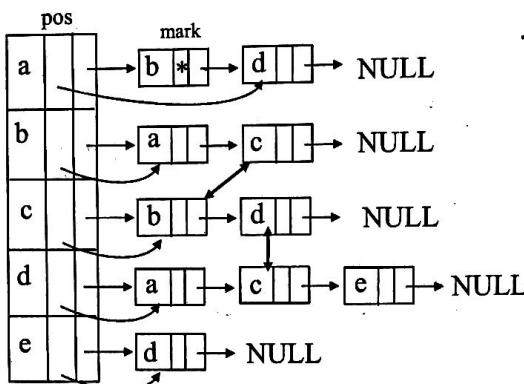
יעילות האלגוריתם: $\Theta(n + m)$

מבנה נתונים ויעילות: גרף לא מכוון

הבעיה:

כל צלע מיוצגת בשתי רשימות שכנים.

כיצד נפתרת אותה?



- לכל צלע: דגל mark (מוסמנת/לא מסומנת).
- לכל קדוקוד: מצביע pos לצלע הלא-מוסמנת הבאה.
- ונשמר מצביעים דו-כיווניים בין "עתיקים" של אותה צלע.

מסלול אוילר

מסלול אוילר Euler Path: מסלול בגרף שմבкар בכל קשת בדיקות פעם אחת (אולם יכול לבקר בקדוקוד יותר מפעם אחת).

טענה: יהיו G גרף לא מכוון קשור.

(ב-) G יש מסלול אוילר \leftrightarrow (קיימים 0 או 2 קדוקודים בעלי דרגה אי-זוגית).

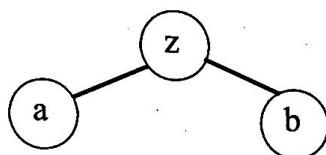
הוכחה:

כיוון ראשון: נניח שיש מסלול אוילר.

אם המסלול הוא גם מעגל אז יש 0 קדוקודים שדרוגתם אי-זוגית. אחרת, לקדוקוד הראשון והאחרון לאורכו המסלול יש דרגה אי-זוגית ולכל יתר הקדוקודים דרגה זוגית (הוכחה דומה למקרה של מעגל).

כיוון שני: אם יש 0 קדוקודים שדרוגם אי-זוגית אז יש מעגל אוילר ולכון גם מסלול.
לכן נניח שיש 2 קדוקודים עם דרגה אי-זוגית, a,b.

כדי למצוא מסלול אוילר נבצע רדוקציה אל הבעה של מציאת מעגל אוילר.
נוסיף קדוקוד חדש z וצלעות (a,z), (z,b).

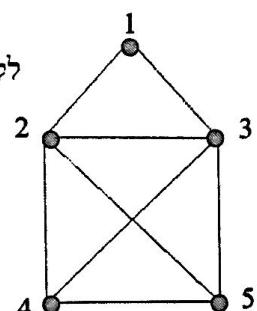
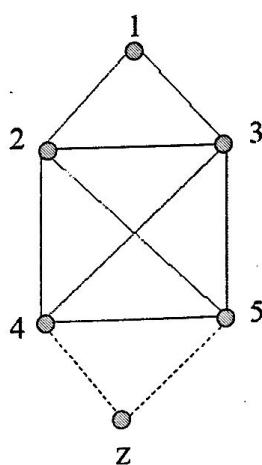


קיבלונו גרף חדש קשיר שכל דרגותיו זוגיות.
קיים בגרף מעגל אוילר, שמתחליל ומסתיים ב-z.

נשميיט מהמעגל את הצלעות שהוספנו ואת הקדוקוד z
ونקבל מסלול אוילר שמתחליל ב-a ומסתיים ב-b.

דוגמה למציאת מסלול אוילר

לקדוקודים 4,5 יש דרגות אי-זוגיות.



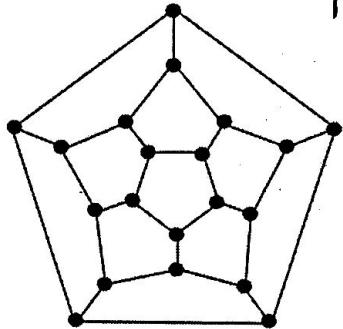
נוסיף קדוקוד z ונמצא מעגל אוילר:

(z,5,3,4,5,2,3, 1,2,4,z)

נוריד את z ונקבל מסלול אוילר בגרף המקורי.

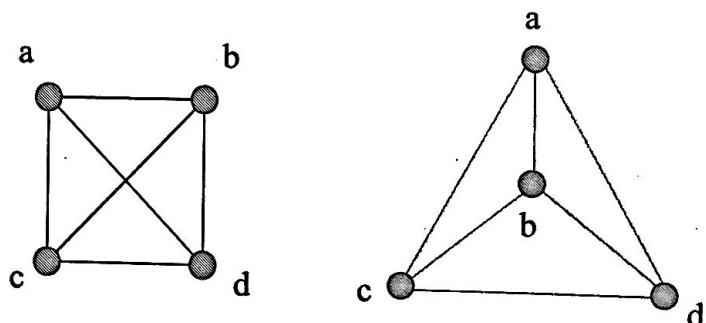
(5,3,4,5,2,3,1,2,4)

מעגל המילטון



מעגל (מסלול) המילטון: מעגל (מסלול) שմבker
בכל קדוק של הגרף בדוק פעם אחת.

סיר המילטון (1805-1865):
מתמטיקאי אירי, שעסוק גם בפיזיקה. המציא משחק שמטרתו
מציאת מעגל המילטון בלוח משחק שנראה כמו הגרף הזה.



K_4 : אין מעגל אוילר. יש מעגל המילטון.

אלגוריתם למציאת מעגל המילטון?

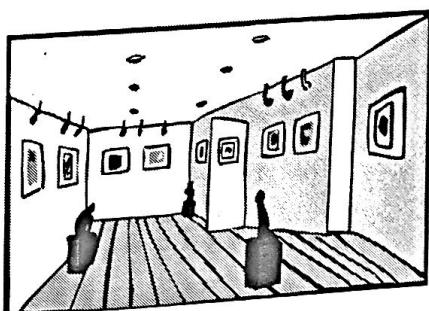
ביעית המעגל המילטוני היא שלמה ל-NP (NP-Complete):
כל לבדוק האם מעגל נתון הוא אכן המילטוני, אולם קשה למצוא את המעגל
(יש אלגוריתם מעריצי, מעריכים שאין פולינומילי).

◀. פרטים נוספים בקורסים
הבאים...

ביקורת במוזיאון

במוזיאון התמונות תלויות במסדרונות וברצוננו לראות את כל התמונות באופן הבא:

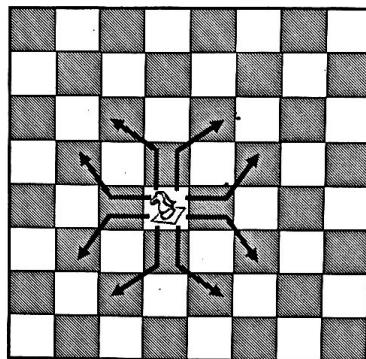
- לעבור בכל מסדרון בדיק פעמי, פעם לכל כיוון.
- לחזור בסיום לנקודת התחלה.



באילו תנאים זה אפשרי?
יכיזד למצוא את המעגל?

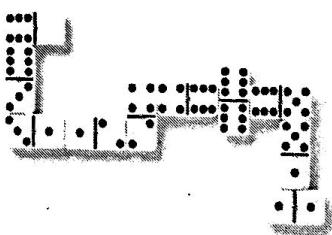
כיסוי לוח שח על ידי צעדי פרש

האם אפשר לכנות לוח בגודל מאתה על ידי צעדי פרש,
כאשר אסור לפרש לדרכו פעמים על אותה משבצת?



סידור אבני דומינו

נתונה עירימה של m אבני דומינו שונות, בה כל אבן מכילה שני מספרים שונים בין 1 ל- m (אין חшибות לסדר בין המספרים, וכל מספר מופיע לפחות על אבן אחת).



נגידו סידור חוקי של אבני הדומינו:

האבני יסודרו בשורה, כאשר ניתן להניח שתאי אבני זו לצד זו אם המספר הכתוב מצד הימני של האבן שਮונחת משמאלו שווה למספר שכחוב מצד השמאלי של האבן שמנחת מימין.

הציעו אלגוריתם המכריע האם קיים סידור חוקי של כל אבני הדומינו.

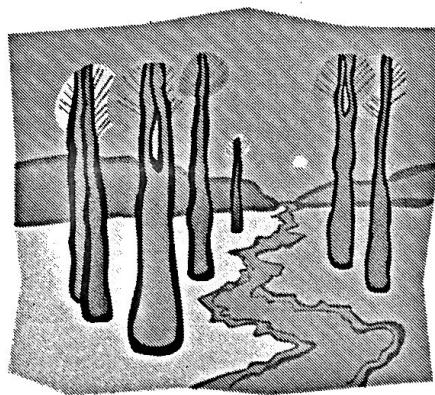
גרף תערוכה

יהי G גרף פשוט, קשיר ולא-מכoon. הגרף G נקרא **גרף תערוכה** מקדים ו \forall אם כל הפעלה של הפונקציה **Find-Circuit** (שמבצעת "טיול" על הגרף) החל מקדקוד v יוצרת מעגל אוילר (ללא תלות בסדר הביקור בקשתות).

הוכחו או הפריכו:

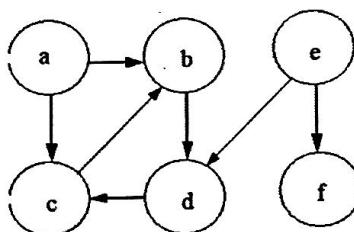
הגרף G הוא **גרף תערוכה** מקדים \forall אם ורק אם \exists ב- G יש מעגל אוילר וגם \forall נמצא על כל מעגל פשוט של G .

חיפוש عمוק בגרף



סרייהן גרפים

המטרה: למפות את הגרף – לחשוף תכונות מבניות של הגרף.
השיטה: סריקה של קדודי הגרף וקשתותיו לפי סדר מסויים.



שיטות לסרייהן של גרף:

חיפוש عمוק: DFS, Depth First Search
חיפוש רוחב: BFS, Breadth First Search

חיפוש עומק - DFS

רעיון בסיסי: סריקה רקורסיבית מכל קדקוד בגרף שעדין לא ביקרנו בו.

נסמן כל קדקוד כשמগলিম אותו (מבקרים בו פעם ראשונה) וכאשר הסריקה ממנה הסתיימה.

מבנה נתונים: מערך $\text{Color}[n]$ כאשר:

עדין לא ביקרנו ב- v . $\text{Color}[v] = \text{white}$
כבר ביקרנו ב- v אבל עדין לא ביקרנו בכל שכניו. $\text{Color}[v] = \text{gray}$
סיימנו לטפל ב- v (ביקרנו בכל שכניו). $\text{Color}[v] = \text{black}$

DFS(Graph G)

```
// INIT
for each vertex u do
    Color[u] ← white           // white = unprocessed

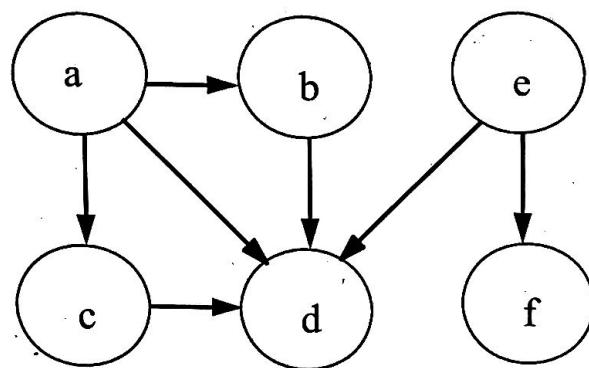
// MAIN LOOP
for each vertex u do
    if Color[u] = white then VISIT(u)
```

VISIT(Vertex u)

```
Color[u] ← gray           // begin processing of u
for each v ∈ Adj[u] do
    if Color[v] = white then
        mark edge (u, v)
        VISIT(v)
Color[u] ← black          // end processing of u
```

יעילות: עוברים פעם אחת על כל אחת מרשימות השכנים $\Theta(n+m)$

דוגמה

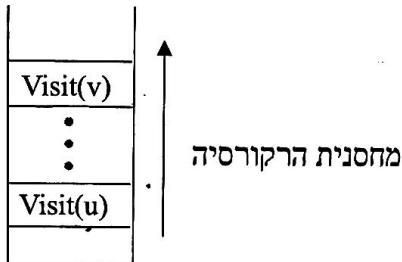


התהיליך מחלק את הגרף לעצי DFS.

תכונות ה-DFS

עקרון הקינון: v_i ו- v_j קדוקו. כל קדוקו שמשמעותו בין גילוי ו- לסיום ו- הוא צאצא של ו- בעץ ה-DFS. קדוקו ש- v_i גם יסתהם לפני ו-

הוכחה: נובע מתוכנת הרקורסיה.



טענה: לעולם במהלך הריצעה של ה-DFS אין קשת מקדוק שחור לקדוק לבן.

הוכחה: אם הקדוק שחור פירוש הדבר שניסינו לבקש בכל שכניו.

עקרון המסלול הלבן

עקרון המסלול הלבן: אם בגרף יש מסלול $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k$, כאשר v_1 הוא הקדוק הראשון שמשמעותו לאורך המסלול זה, או כל הקדוקים במסלול יהיו צאצאים של v_1 בעץ ה-DFS.

הוכחה: באינדוקציה על k .
בטיס: $1 = k$, טריוויאלי.

נניח נכונות ל- $(k-1)$ ונוכיח למסלול $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k$.

לפי הנחת האינדוקציה כל הקדוקים במסלול $v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_{k-1}$ יהיו צאצאיו של v_1 בעץ ה-DFS. נראה שגם v_k יהיה צאצא של v_1 .

לפי הנחת האינדוקציה כ- v_i מסתיימים או- v_{k-1} כבר שחור (כי הוא צאצא של v_1).
לא ניתן שבשלב זה v_k לבן. ←

כלומר v_k התגלה אחרי ש- v_i התגלה ולפניה סיום.
לפי עקרון הקינון, v_k הוא צאצא של v_i . ←

סיווג קשתות

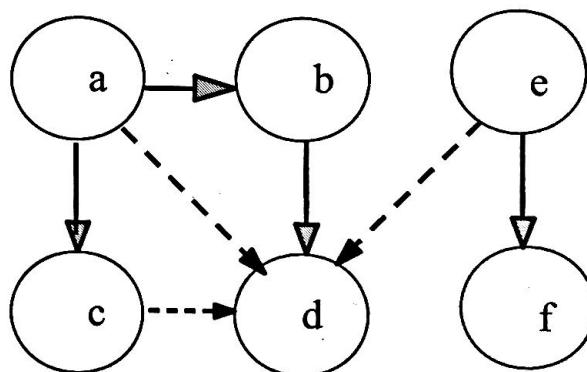
נסוג במהלך ביצוע ה- DFS את הקשתות שאנו סורקים.

קדוקוד v צאצא של u אם שניהם באותו עץ DFS ויש מסלול בעץ מ- v ל- u .
במקרה זה v הוא אב קדמון של u .

נניח שאנו כרגע בקדוקוד v ובוחנים את שכנו u . הקשת (v,u) היא:

- **קשת עץ :Tree Edge**: אם v לבן, במקרה זה u יהיה ההורה של v בעץ.
- **קשת חוזרת :Back Edge**: אם v אפור, כלומר v הוא אב קדמון של u בעץ.
- **קשת קדמית Forward Edge או קשת חוצה Cross Edge**: אם v שחור.
אם v הוא צאצא של u בעץ זו תהיה הקשת קדמית.
אחרת זו קשת חוצה (יכול להיות באותו עץ או בין עצים שונים).

דוגמה



```
DFS(Graph G) // DFS edge classification
```

```
for each vertex u do // INIT  
    Color[u] ← white
```

```
for each vertex u do // MAIN LOOP  
    if Color[u] = white then VISIT(u)
```

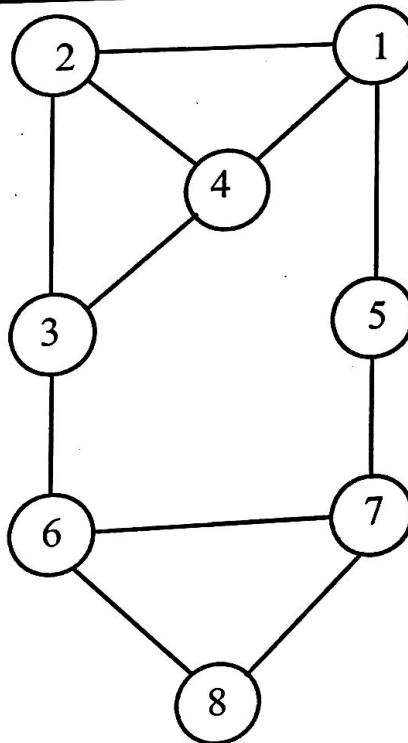
```
VISIT(Vertex u) // print tree and backward arcs.
```

```
Color[u] ← gray // begin processing of u
```

```
for each v ∈ Adj[u] do  
    if Color[v] = white then  
        print (u,v) "is a tree arc"  
        VISIT(v)
```

```
else if Color[v] = gray then  
    print (u,v) "is a back arc"
```

```
Color[u] ← black // end processing of u
```



בגרף לא מכוון, כל צלע מסווגת כשנתקלים בה לראשונה.

סוגי קשתות בגרף לא-מכוון

טענה: בגרף לא-מכוון יש רק קשתות עז וקשתות חוזרת.



הוכחה: יהיו $u, v \in V$ שכנים. נניח (בגאה"כ) ש- u הtagלה לפני v .
 v יסתים לפני u , כי v שכן של u .
לכן v יהיה צצא של u .

1. אם גילינו את v לראשונה דרך הקשת (v,u) :
או הקשת תסוג **קשת עז**.

2. אם גילינו את v לראשונה דרך קשת אחרת:
או נעבור על הקשת בין v ל- u לראשונה בכיוון $u \rightarrow v$
(כי כאשר מגלים את v מנסים לבקר את כל שכניו)
במקרה זה הקשת (u,v) תסוג **קשת חוזרת**.

שימושים ב-DFS

DFS לשימוש ל-

ה"מפה" שנוצרת במהלך ה-DFS משמשת (בין השאר) לפתרון הבעיות הבאות:

- האם גרף לא-מכוון הוא קשור?
- מציאת רכיבי קשרות בגרף לא מכוון
- מציאת מעגל בגרף
- מינון טופולוגי
- האם גרף מכוון הוא קשר חזק?
- מציאת רכיבי קשרות חזקה בגרף מכוון
- מציאת גרף העיל של גרף מכוון
- מציאת גשרים בגרף לא מכוון

בדיקות קשרות של גרף לא-מכוון

קלט: גרף לא-מכוון ($G = (V, E)$)

פלט: האם G קשור?

אלגוריתם:

- נריצ' (s) Visit(s) מקדקוד התחליה כלשהו s.
- נבדוק אם הגיענו לכל הקדקודים.

יעילות: $\Theta(n+m)$

טענה 1: אם הגרף קשור אז (s) Visit(s) יגיע לכל הקדקודים.
הוכחה: נובע מתכונת המסלול הלבן.

טענה 2: אם (s) Visit(s) יגיע לכל הקדקודים אז הגרף קשור.

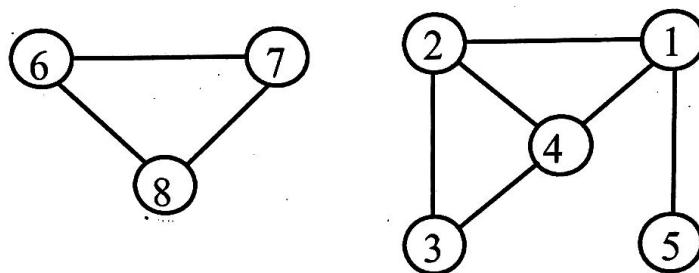
מציאת רכיבי קשירות בגרף לא-מכoon

קלט: גרף לא-מכoon $G = (V, E)$.

פלט: חלוקה של קדוקוד G לרכיבי קשירות.

אלגוריתם:

- נבצע DFS ולכל קדוקוד נזכיר את שורש העץ שלו.
- כל עץ הוא רכיב קשירות.



יעילות: $\Theta(n+m)$ כמו DFS רגיל.

מציאת רכיבי קשירות בגרף לא-מכoon

```
DFS(Graph G) // DFS with maintenance of tree roots

for each vertex u do // INIT
    Color[u]  $\leftarrow$  white

for each vertex u do // MAIN LOOP
    if Color[u] = white then
        CurrentRoot  $\leftarrow$  u // a global variable!
        VISIT(u)
```

VISIT(Vertex u) // Set Root[u] and visit u.

```
Root[u]  $\leftarrow$  CurrentRoot
Color[u]  $\leftarrow$  gray // begin processing of u
for each v  $\in$  Adj[u] do
    if Color[v] = white then
        VISIT(v)
    Color[u]  $\leftarrow$  black // end processing of u
```

נכונות האלגוריתם לרכיבי קשירות

אלגוריתם:

- נבצע DFS ולכל קדקוד נזוכר את שורש העץ שלו.
- כל עץ DFS הוא רכיב קשירות.

טענה: יהי C רכיב קשירות, או קיים עץ T כך ש- $T=C$ (כקבוצות קדקודים).
הוכחה: יהי C רכיב קשירות ויהי x הקדקוד שהתגלה ראשון מהרכיב.
לפי עקרון המסלול הלבן, כל קדקוד ב- C יהיה צאצא של x .
מайдע, ברור שככל צאצא של x הוא קדקוד ב- C .

כעת אפשר לענות על שאלות כגון:

- האם שני קדקודים הם באותו רכיב קשירות?
- הדרשת רכיבי הקשירות.

מציאת מעגל בגרף מכובן

קלט: גרף מכובן

פלט: האם הגרף אציקלי, או שיש בו מעגל (אחד לפחות).

אלגוריתם:

- נבצע DFS בגרף.
- אם מתגלה קשת חוזרת - נעצור ונזכיר שיש מעגל.
- אחרת, נזכיר שהגרף אציקלי.

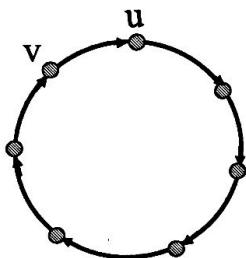
יעילות: $\Theta(n+m)$

הוכחת נכונות

טענה: אם מרכיבים DFS בגרף מכון G , אז מתקיים:
(ב- G יש מעגל) \leftrightarrow (קיימת קשת חוזרת במהלך ריצת ה-DFS)

הוכחה:

(\rightarrow) קשת חוזרת מחברת קדקוד לאב קדמון שלו.
לכן היא סוגרת מעגל. מעגל כזה נקרא מעגל בסיסי.



(\leftarrow) יהיו u והקדקוד הראשון שמתגלה לאורך המעגל.
תהיה (u, v) הקשת שסוגרת את המעגל.
כל הקדקודים לאורך המעגל נמצאים על מסלול $v - u$.
לפי עקרון המסלול הלבן, הם יהיו צאצאים שלו.
 (v, u) תסוג כקשת חוזרת.

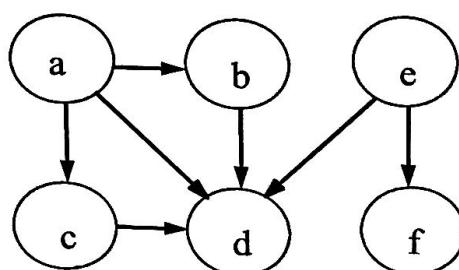
סדר הקדקודים ע"י DFS

- רשימת גילוי: רושמים קדקוד כאשר הוא מתגלה.
- רשימת סיום: רושמים קדקוד כאשר הטיפול בו מסתיים.

dfs

מספרי גילוי: ממספרים קדקוד כאשר הוא מתגלה. סימון: $d(v)$

מספרי סיום: ממספרים קדקוד כאשר הוא מסתיים. סימון: $f(v)$



מיכון טופולוגי על ידי DFS

קלט: גרף ממוקן פשוט $G = (V, E)$

פלט: סידור ליניארי של כל קודקי G , כך שם $v \rightarrow u$ אז u יופיע לפני v
(או הודעה שאין כזה סידור).

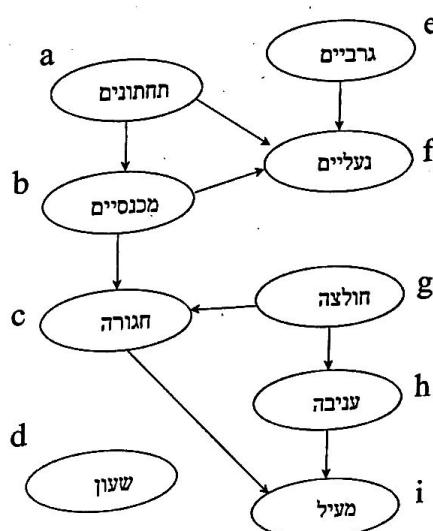
אלגוריתם:

בצע DFS, תוך בנית רשימת סיום.
הdeps את רשימת הסיום בסדר הפוך.

יעילות: $\Theta(n+m)$



דוגמה



Topological Sort(Graph G)

```

Stack F_Stack
for each vertex u do
    Color[u] ← white
for each vertex u do
    if Color[u] = white then
        Visit(u)
Print F_Stack

```

Visit(Vertex u)

```

Color[u] ← gray
for each v ∈ Adj[u] do
    if Color[v] = gray then
        print "Error: CYCLE"
        exit(1)
    if Color[v] = white then
        Visit(v)
Color[u] ← black
F_Stack.Push(u)

```

הוכחת נכונות

טענה 1: אם הגרף אציקלי, אז האלגוריתם מוצא מון טופולוגי.

הוכחה: נניח ש- $v \rightarrow u$ ונראה שהקדוק u יסתים אחרי v , כלומר: $f(v) < f(u)$.

כשנברך לראשונה בקדוק u הצבע של v יכול להיות:

- לבן: v יהיה צצא של u .
לכן הקריאה הרקורסיבית של v תסתיים לפני זו של u , כלומר $f(v) < f(u)$.

- אפור: גילינו מעגל - לא יתכן כי אין מעגל בגרף.

- שחור: v כבר הסתיים, ומכובן $f(v) < f(u)$.

טענה 2: האלגוריתם יdump "מעגל" אם ורק אם יש מעגל.
הוכחה: הוכחנו שיש קשת חוזרת אם ורק אם יש מעגל מכון והאלגוריתם בודק זאת.

בדיקות קשיירות חזקה של גרף מכוון

קלט: גרף מכוון $G = (V, E)$.

פלט: האם G קשר חזק?

גרף מכוון נקרא קשר חזק אם לכל שני קודקודים u, v יש מסלול $u \rightarrow v$ וגם $v \rightarrow u$.

אלגוריתם:

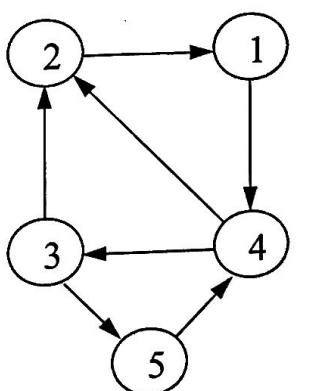
- נרץ $\text{Visit}(s)$ בגרף G (s - קודקוד התחלה שמיוחס).
- אם לא הגיעו לכל הקודקודים - התשובה לא.
- אחרת - נרץ $\text{Visit}(s)$ על הגרף G^T (ע"י הפוך הקשיות).
- אם הגיעו לכל הקודקודים - התשובה היא "הגרף קשר חזק", אחרת לא.

יעילות: $\Theta(n+m)$

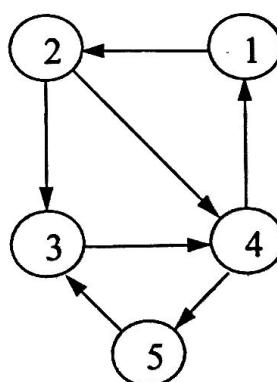
דוגמאות הרצה

$s = 1$

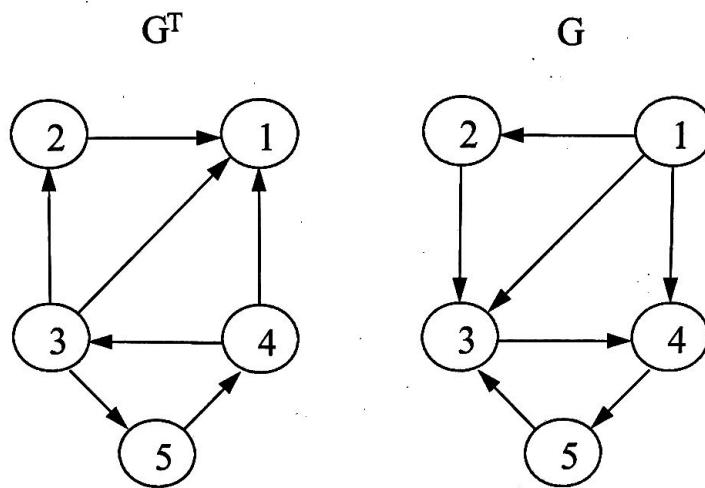
G^T



G



$s = 1$

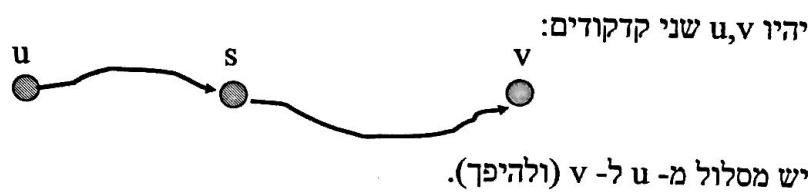


הוכחת נכונות

$$u \xrightarrow{G} v \iff u \xrightarrow{G^T} v$$

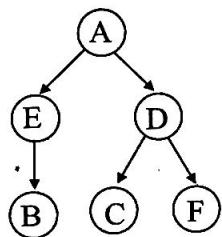
אבחנה:

- משפט: האלגוריתם עונה שהגרף קשור חזק אם ורק אם זה כך.
- הוכחה: (כיוון "רק אם", כלומר: אם האלגוריתם עונה "כן" אז הגרף קשור חזק)
- DFS על G מבטיח שיש ב- G מסלול מ- u לכל הקדוקדים.
 - DFS על G^T מבטיח שיש ב- G^T מסלול מ- v לכל הקדוקדים.
 - ב- G יש מסלול מכל הקדוקדים ל- s .



תרגילי חזרה – DFS

- נתון עץ DFS שהתקבל מעבר DFS על גרף מכון G . סדר הילדים של כל קדקוד הוא משמאלי ימינו לפי הסדר שבו ביקרו בהם.



הוכיחו או הפריכו את הטענות הבאות:

- בגרף G תיתכן הקשת (A,B) .
- בגרף G תיתכן הקשת (B,C) .

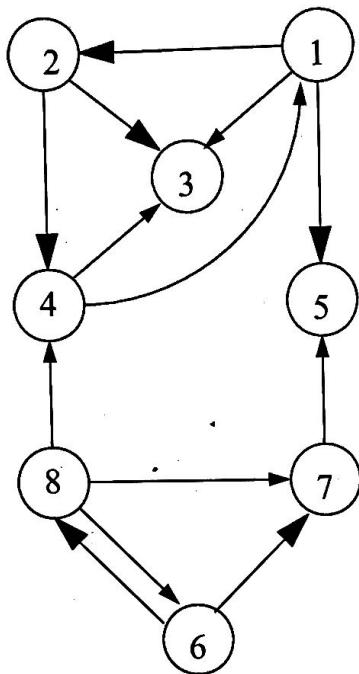
תרגילי חזרה – DFS

- יהי G גרף שרשרת מכון (מסלול פשוט) בן n קדקודים. מה ניתן לומר על מספר העצים שיוצריםDFS על G ? במה זה תלוי?

- אותה שאלה לגבי גרף לא מכון.

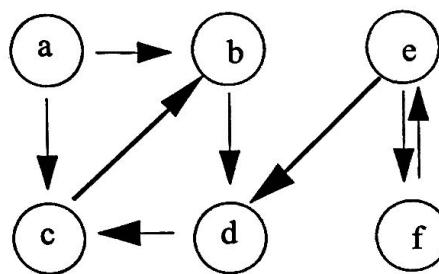
- מצאו חסם תחתון למספר המעלגים הפחותים בgraf השלים K_n . מה דעתכם על אלגוריתם שעובר על כל המעלגים בgraf כדי לבדוק משהו?

סובגו את הקשתות וציירו עצי DFS



מציאת רכיבי קשירות חזקה

הגדרה: רכיב הקשירות החזקה של v הוא קבוצת כל הקודודים שיש מסלול מכובן מ- v אליהם ולהיפך.
 \Leftrightarrow תחת-הגרף הקיים בחזקה הגדול ביותר המכיל את v .



הבעיה: קלט – גרף מכובן G
 פלט – פירוק של G לרכיבי קשירות חזקה.

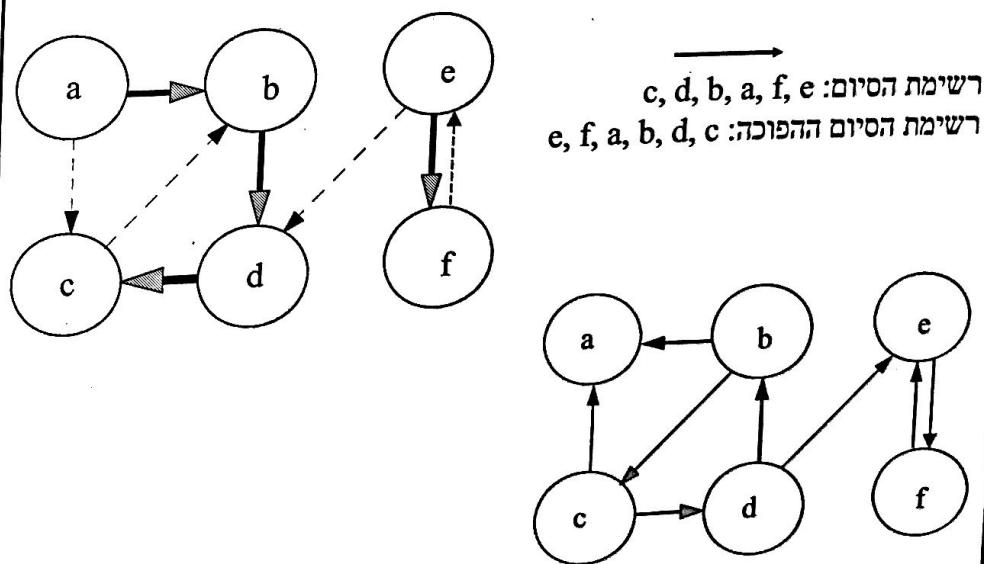
אלגוריתם שְׁרִיר-קֹסֶרוֹן

אלגוריתם:

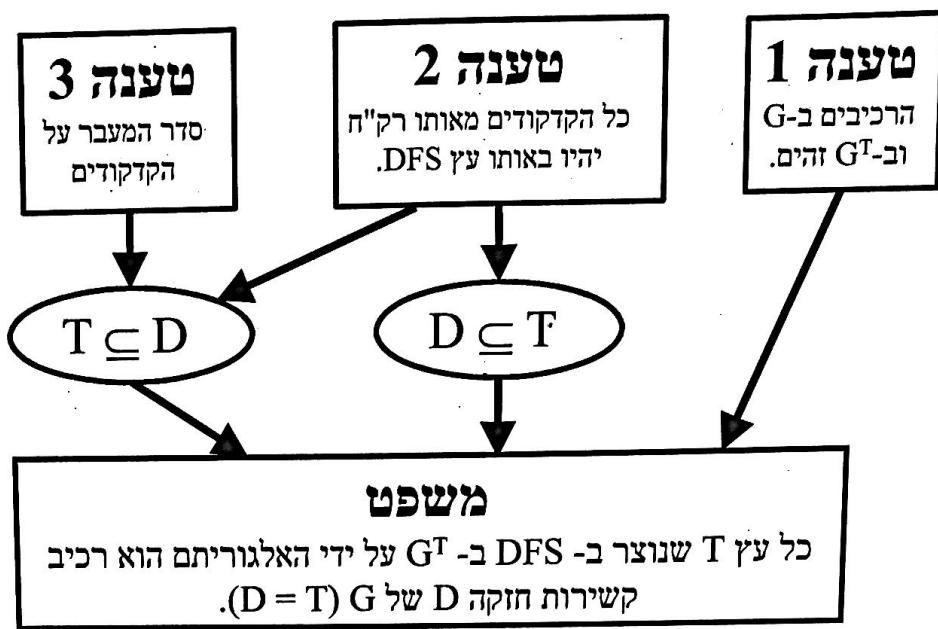
- נבצע DFS על הגרף G . במהלך הסריקה, נבנה רשימת-סיום של הקדקודים.
- נבנה את הגרף G^T .
- נבצע DFS על G^T , תוך שימוש ברשימה הסיום בסדר הפוך בטור רשימת הקדקודים בלולאה הראשית.
- נחלק את קדקודיו הגרף לרכיבי קשרות חזקה לפי עצי ה-DFS המתקבלים מסריקת G^T .

יעילות: $\Theta(n+m)$

דוגמת הרצה



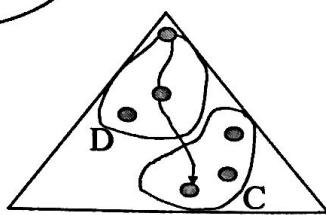
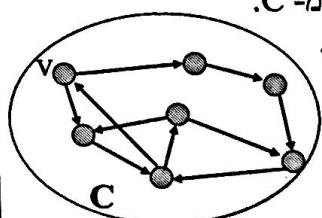
מציאת רכיבי קשירות חזקה בגרף מכוון מבנה הוכחת נכונות



הוכחת נכונות

טענה 1: רכיבי הקשירות החזקה ב- G וב- G^T זהים.
הוכחה: ברור מהסימטריה של הגדרת רכיבי הקשירות החזקה.

טענה 2: כל הקדקודים מאותו רכיב קשירות חזקה יהיו באותו עץ DFS של G (של G^T).
הוכחה: יהיו C רכיב כלשהו ויהי v הקדקווד הראשון שהתגלה מ- C .
יש מסלולים לבנים מ- v לכל אחד מהקדקודים ב- C .
לפי עקרון המסלול הלבן, כולם יהיו צאצאים של v בעץ.



הערה: ניתן שבאותו עץ DFS של G יהיה יותר מרכיב קשירות חזקה אחד.

סימון: $f_v = \text{זמן סיום הביקור בקדקוד } v \text{ בגרף } G$.

טענה 3: *יהי C רכיב קשרות ויהי v הקדקוד הראשון מתוך C שהתגלה בסריקה של G .
יהי u קדקוד (לא דוקא מותך C) כך שיש מסלול מ- v אל u ב- G . אז $f_v < f_u$.*

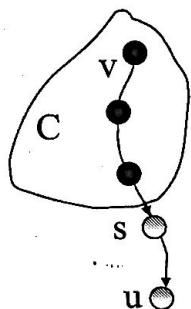
הוכחה: נניח בשלילה ש- $f_v > f_u$. נתבונן ברגע שהביקור ב- v מסתיים.

לפי הנחתנו, u טרם הסתיים (אינו שחור).

יהי s הקדקוד הראשון לאורך המסלול מ- v ל- u שאינו שחור.

לפיכך הוא לבן או אפור.

אולם לא ניתן קשת משחור לבן. לכן, s אפור.



\leftarrow *s אב קדמון של v בעין ה-DFS. כמו כן יש מסלול מ- v ל- s , לכן, s , v , u שייכים לאותו רכיב C , ואילו s התגלה לפני v , סתירה לכך ש- v הריאן מהרכיב שהתגלה.*

מסקנה: $f_u < f_v$.

משפט: כל עץ שנוצר ב-DFS ב- G^T על ידי האלגוריתם
הוא רכיב קשרות חזקה של G .

הוכחה: *יהי r השורש של עץ כלשהו T , שנוצר בסריקה של G^T ,
ויהי D רכיב הקשרות של r . נוכיה ש- $D = T$.*

הראינו כבר ש- $D \subseteq T$ (טענה 2). נותר להוכיח ש- $T \subseteq D$.

נניח בשלילה שיש קדקוד ב- T שאינו שייך ל- D ,

אללא לרכיב אחר C . גם $C \subseteq T$.

יהי v הקדקוד הראשון מתוך C שהתגלה לראשונה ב-DFS על G .

מצד אחד: החיפוש ב- G^T הzbוצע על פי רשימת סיום הפוכה
(=זמן סיום יורד), ו- r הוא שורש עץ ב- G^T , לכן $f_r > f_v$.

מצד שני: הקדקוד v יצא צאצא של r ב- T , לכן v נחkerת ב- G^T .
ולפי טענה 3: $f_v < f_r$

קיים $f_r > f_v$ וגם $f_v < f_r$ וזה סתירה.

שריר וקוזרזו'

קוסרزو' ושריר גלו כל אחד בנפרד את האלגוריתם למציאת רכיבי קשרות חזקה בסביבות 1980.



מיכה שריר (נולד 1950):
מתמטיקאי ואיש מדעי המחשב מאוניברסיטת תל-אביב
(במיוחד בתחום הgiואומטריה החישובית).



קוסרזו' (נולד 1943): נולד בהודו, פרופ' למדעי המחשב בארא"ב
בתחומי האלגוריתמים, גם אלגוריתמים מקבילים.

תרגיל חזרה – DFS ורכ"ח

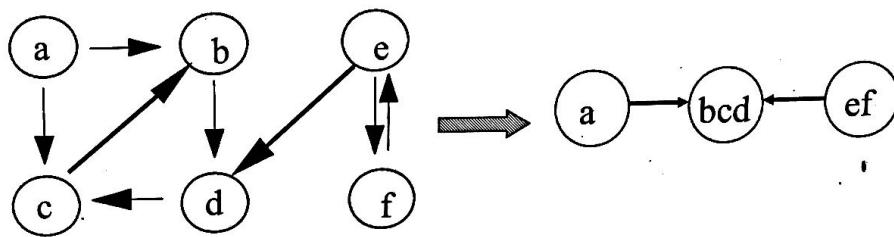
הוכיחו או הפריכו את הטענות הבאות.

1. יהיו G גרף מכון כך שיש בו G מסלול מ- v ל- u , ו- v מתגלה לפני u במהלך DFS. אז הביקור ב- v מסתיים לפני שהביקור ב- u מסתיים.
2. ב-DFS של גרף מכון, הקדוקדים של כל רק"ח מופיעים ברציפות בראשימת הסיום.

גרף העל של גרף מכוון

הגדרה: יהיו G גרף מכוון. **גרף העל של G** או **גרף רכיבי הקשרות חזקה של G** הוא גרף מכוון שקדקודיו הם רכיבי הקשרות החזקה של G וקשותתו מייצגות גישות בין הרכיבים.

פורמלית, יהיו A, B שני רכיבי קשרות חזקה של G .
או בגרף העל יהיו קדוקדים A, B ותהי קשת מ- A ל- B
אם יש קשת (אחת לפחות) ב- G מקדוקוד ב- A לקדוקוד ב- B .



חישוב גרף העל

כיצד נמצא את גרף העל?

קלט: גרף מכוון (V, E)

פלט: גרף העל של G

נדרוש שהאלגוריתם יירוץ בזמן ליניארי ויבנה גרף פשוט
(לא יוכל קשותות כפולות בין רכיבי הקשרות החזקה).

תכונות של גרף ה عل

אבחן: גרף ה

عل
 הוא גרף אציקלי.
(נובע מכך שמעגל שיך יכול לרכיב קשורות אחד).

טענה: האלגוריתם למציאת רכיבי קשורות חזקה מגלה את הרכיבים (=קדקודיו של גרף ה

عل
) בסדר של מון-טופולוגי.

כלומר: אם יש קשת $D \rightarrow C \rightarrow G$, אז C יתגלה לפני D .

הוכחה:

נראה שגם C ו- D שני רכיבי קשורות חזקה ב- G , וקיים קשת המחברת קדקוד ב- C לקדקוד ב- D , אז C יתגלה לפני D יתגלה.

אם יש קשת מ- C ל- D אז בגרף G^T יש קשת מקדקוד ב- D לקדקוד ב- C .
 D ו- C הם עצים שונים ב-DFS של G^T ולכן קשת החוצה בין-עצית.
קשותות החוצה תמיד מכוונת אחורה (մבנית סדר הגילוי) ולכן C התגלה קודם.

שימוש בגרף ה عل

לעתים קרובות קל לטפל בשני המקרים המוחדים:

- 1) גרף קשיר חזק.
- 2) גרף אציקלי.

גרף ה

عل
 מאפשר לאחד שני פתרונות אלה, כדי לטפל בכל גרף מכוון.

דוגמאות

קלט: גרף מכוון ($G = (V, E)$).

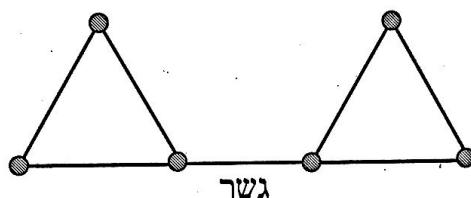
פלט: האם יש קדקוד שמננו ניתן להגיע לכל קדקוד הגרף.

קלט: גרף מכוון ($G = (V, E)$).

פלט: האם יש מסלול מכוון (או דוווקא פשוט) שכולל את כל קדקוד הגרף.

גשרים בגרף לא-מכוון

יהי G גרף לא-מכוון קשיר. גשר הוא צלע שהסרתה מנתקת את הגרף.

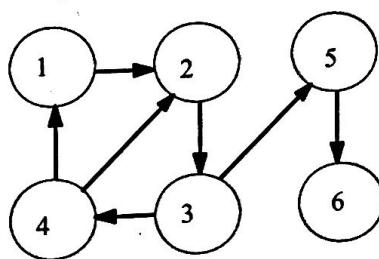
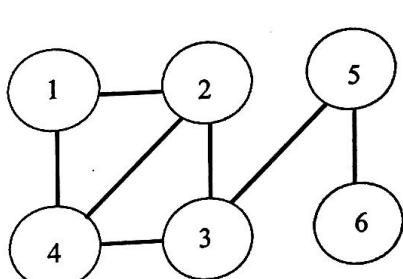


הבעיה: נתון גרף קשיר לא-מכוון G . צריך למצוא את כל הגשרים.

אלגוריתם:

- נסורך את הגרף ע"י DFS ונכוון את הצלעות לפי כיוון המעבר הראשון בהן.
- גשר יהיה צלע שמחברת בין רכיבי קשרות חזקה בגרף המכון שהתקבל.

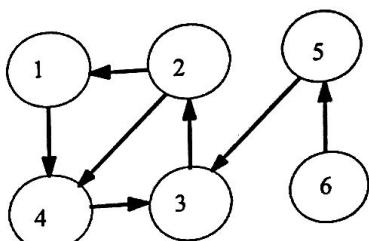
דוגמת הרצאה



רשימת הסיום: 4, 6, 5, 3, 2, 1

רשימת הסיום ההפוכה: 1, 2, 3, 5, 6, 4

הграф הפוך : G^T



הוכחת נכונות

טענה: צלע איננה גשר אם ורק אם היא נמצאת על מעגל.

הוכחה: תהי (v,u) צלע שאיננה גשר. אם נסיר את (v,u) מהגרף עדיין יהיה מסלול מ- v ל- u . לכן בגרף המקורי יש מעגל שכולל את הצלע (v,u) .

הכוון השני דומה.

מישפט: יהיו G^* הגרף המתקיים על ידי כיוון הצלעות של G בעזרת DFS.

צלע היא גשר אם ורק אם היא בין רכיבי הקשרות החזקה של G^* .

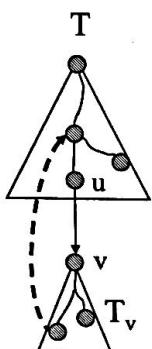
טענה 1: צלע השיכת לרכיב קשרות חזקה של G^* איננה גשר של G .

הוכחה: צלע בתחום רכיב קשרות חזקה נמצאת על מעגל מכוון של G^* , ולכן על מעגל לא-מכוון של G , ומכאן שאיננה גשר.

טענה 2: קשת המחברת בין רכיבי קשרות חזקה של G^* היא גשר של G .

הוכחה: תהי (v,u) קשת בין שני רכיבי קשרות חזקה. נסחכל על הסיווג שלה לפי ה-DFS שיצר את G^* .
(v,u) לא קשת חוזרת: אחרת היא שייכת למעגל מכוון ולכן נמצאת בתחום רק "ת".

◀ (v,u) קשת עז. נתבונן בעז ה-DFS שכולל את (v,u) , ויהי T_v תחת-העץ של v והשורש שלו.



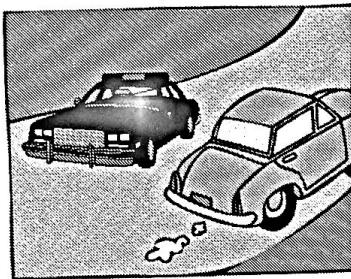
נניח שיש קשת נוספת שמחברת את T_v אל יתר הגרף. ב-DFS של גרף לא מכוון יש רק קשותות עז וחזרות, ולכן היציאה מ- T_v נעשית ע"י קשת חוזרת.

קשת כזו מחברת צאצא של v לאב קדמון של u , ולכן (v,u) נמצאת על מעגל מכוון ונקבל סתירה.

◀ המצב שהנחנו אינו אפשרי. כלומר, אין אף קשת המחברת את T_v ליתר הגרף בלבד (v,u), ולכן קשת זו היא גשר.

תרגיל: עיר חד סטרית

נתונה מפה (גרף לא-מכוון קשור) המייצגת עיר עם רחובות דו-סטריים. רוצים להפוך את כל הרחובות לחד-סטריים, אבל שתהיה דרך להגעה מכל נקודה לכל נקודה. יש לקבוע האם זה אפשרי, ואם כן כיצד עושים זאת.



הՃכה:

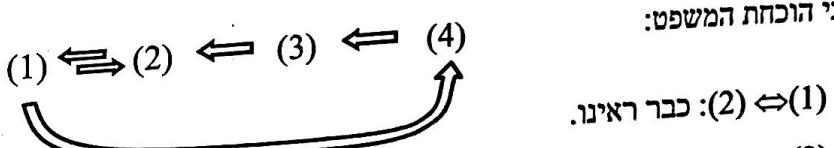
- מה אם יש גשרים במפת העיר?
- מה אם אין?

הוכחת נכונות (ועוד)

מ Shepard: יהיו G גרף לא-מכוון קשור. אז הטענות הבאות שקולות:

- (1) ב- G אין גשרים.
- (2) כל צלע ב- G נמצא על מעגל.
- (3) ניתן לכוון את צלעות G כך שיתקיים גרף קשור חזק.
- (4) כיוון G ע"י DFS יוצר גרף קשור חזק.

שלבי הוכחת המשפט:



(2) \Leftrightarrow (1): כבר ראינו.

(3) \Rightarrow (2): בגרף קשור חזק, כל קשת שייכת למעגל.

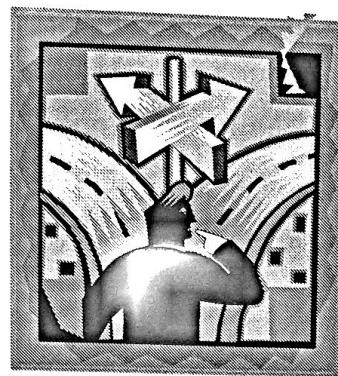
(4) \Rightarrow (3): טריויאלי.

(1) \Rightarrow (4): מהוכחת נכונות של האלגוריתם למציאת גשרים.

תרגילים נוספים

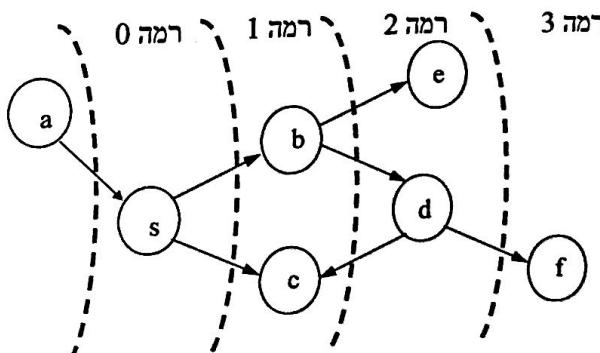
- האם כל מעגל בגרף לא-מכוון הופך למעגל מכוון
אחרי שמכוונים את הצלעות של הגרף הלא-מכוון בעזרת DFS?
- האם קיום מעגל אוילר בגרף לא-מכוון הוא תנאי הכרחי ומספיק
לכך שניתן לנו מפת רוחבות דו-סטרית למפה קשירה חזק וחד-סיטרית?

חישוב רוחב בגרף



BFS - חישוב רוחב

המטריבציה: מסלולים קצריים ביותר / מציאת מרחק.
המרחק של קדוקד v מקדוקד s = אורך המסלול הקצר ביותר מ- s ל- v .
היעיון: ביקור בגרף לפי רמות.



מבני נתונים ואלגוריתם

קלט: גרף G וקודקוד התחלה s .

פלט: מערך L של קבוצות (רשימות) של קודקודים.
 $L[i]$ תהיה קבוצת הקודקודים שמרחיקם מ- s הוא i .

אלגוריתם:

1. אתחליל כל $L[i]$ לקבוצה ריקה.
2. $i = 0$, $L[0] = \{s\}$
3. כל עוד $L[i]$ איננו ריק:
 - לכל קודקוד u ב- $L[i]$,
הכנס $L[i+1]$ את כל השכנים של u שעוד לא ביקרנו בהם.
 - קדם את i ב-1.

תוצר נסף: $[v]$ יהיה ארכו של מסלול קצר ביותר מ- s ל- v , ו- ∞ אם אין מסלול כזה.

BFS(Graph G, Vertex s)

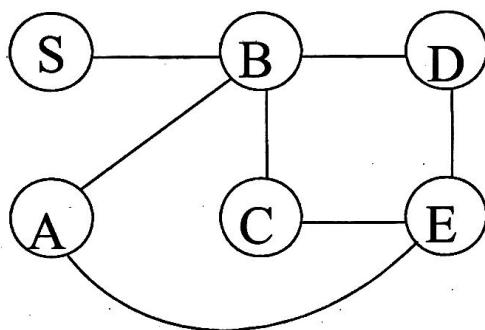
Set $L[]$

```
// INIT d
for each vertex v do
    d[v] ← ∞

// Level 0
L[0] ← {s}
d[s] ← 0

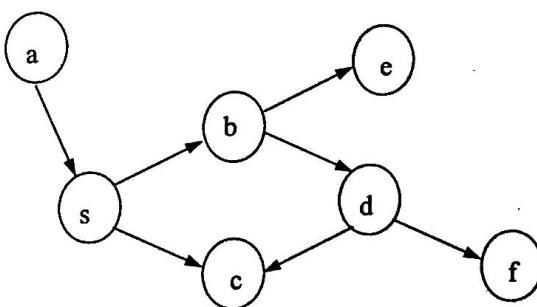
// Loop builds L[i+1] from L[i]
i ← 0
while L[i] ≠ ∅ do
    for each u ∈ L[i] do
        for each v ∈ Adj[u] do
            if d[v] = ∞ then
                d[v] ← i + 1
                add v to L[i+1]
    i ← i + 1
```

דוגמת הרצאה



פונקציית המרחק δ

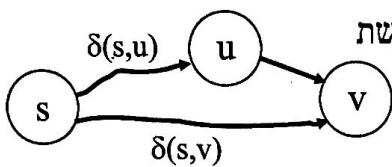
$\delta(s,v)$ = המרחק של v מ- s (= אורך של המסלול הקצר ביותר מ- s ל- v).



$$\delta(s,d) = 2, \quad \delta(s,c) = 1, \quad \delta(s,a) = \infty$$

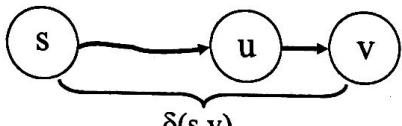
תכונות של הפונקציה δ

אי-שוויון המשולש: תהי $E(v,u)$. אז $\delta(s,v) \leq \delta(s,u) + 1$.



הוכחה: המסלול הקצר ביותר מ- s ל- v בתוספת הקשת (u,v) הוא מסלול מ- s ל- v אשר אורכו $\delta(s,v) \leq \delta(s,u) + 1$ ולכן: $\delta(s,u) + 1$

טענה 1: יהי $v \neq s$ ויהי v הקדקוד הפלני-אחרון על מסלול קצר ביותר מ- s ל- v .
 $\delta(s,u) = \delta(s,v) - 1$



הוכחה: המסלול הקצר ביותר מ- s ל- v ללא הקשת (u,v) הוא מסלול מ- s ל- u , שאורכו $\delta(s,v) - 1$.
 אם היה מסלול קצר יותר מ- s ל- u , אז היה גם מסלול קצר יותר מ- s ל- v .

הוכחת הנכונות של אלגוריתם BFS

טענה: האלגוריתם מכניס את הקדקוד v ל- $L[i]$ אם ורק אם $i = \delta(s,v)$.

הוכחה: באינדוקציה על i .

ביסיס: האלגוריתם מכניס v ל- $L[0]$ במהלך ריצתו רק את s , ואכן s הוא הקדקוד היחיד שمرחקו מ- s הוא 0.

הנחה האינדוקציה: נניח שלכל $(i-1), i, \dots, 0$, ב- $L[i]$ נמצאים בדיקת הקדקודים כך ש- $\delta(s,v) = i$.

שלב המעבר: יהי v קדקוד כך ש- $\delta(s,v) = i+1$, ויהי u הקדקוד שלפניו במסלול הקצר

ביותר מ- s ל- v . לפי טענה 1, $\delta(s,u) = i$.

לכן לפי ההנחה האינדוקציה, u נמצא ב- $L[i]$.

כמו כן v נמצא ב- $L[i+1]$, ולכן לפי ההנחה האינדוקציה v לא נמצא ב- $L[i], \dots, L[0]$.

לכן האלגוריתם ימצא את v במעבר על $L[i+1]$ ויוכניס אותו ל- $L[i]$ כנדרש.

יש להוכיח גם של- [i] לא יכנס אף קדקוד v אשר $i \neq v$.

נניח בשיילה שיש קדקוד v כך ש $i > v$ ו $\delta(s,v) > \delta(s,i)$.

$\delta(s,u) = i-1$ קיימים $v \in L[i-1]$ שכון של v לפי הנחת האינדוקציה \leftarrow

לפי אי-שוויון המשולש $i \leq \delta(s,u) + 1 = \delta(s,v) + 1$, בסתירה להנחה \leftarrow

$\delta(s,v) = i$ אם $v \in L[i]$ \leftarrow

מימוש BFS בעזרת תור

BFS(Graph G, Vertex s)

Queue Q

for each vertex v do

$d[v] \leftarrow \infty$

Q.Enqueue(s)

$d[s] \leftarrow 0$

while $Q \neq \emptyset$ do

$u \leftarrow Q.Dequeue()$

for each $v \in Adj[u]$ do

if $d[v] = \infty$ then

$d[v] \leftarrow d[u] + 1$

Q.Enqueue(v)

יעילות: $\Theta(n+m)$

שימושים פשוטים של BFS

• האם יש מסלול מ- u ל- v ?

• מה ארכו של המסלול הקצר ביותר.

• מציאת רכיבי קשירות בגרף לא-מכוון.

• אם מסמנים את הקשתות שבעורתן מגלים לראשונה קדוק
נווצר עז שנקרא עז.BFS

שאלה ממבחן

כתבו אלגוריתם שמקבל כקלט גרף פשוט ומכוון G וקדוקוד כלשהו u , ומחזיר
כפלט את המעגל המכונן הקצר ביותר ש- v משותף אליו, או מודיע על לא
נמצא על אף מעגל.

הדפסת מסלול קצר ביותר

- הוספה מערך d :

```
if  $d[v] = \infty$  then  
     $d[v] \leftarrow d[u] + 1$   
     $p[v] \leftarrow u$ 
```

(הנחה: המערך מאותחל ל-NULL).

- הדפסת מסלול:

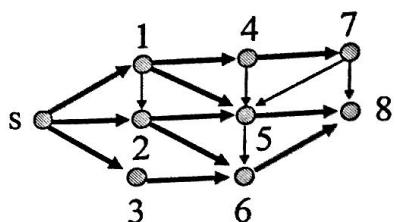
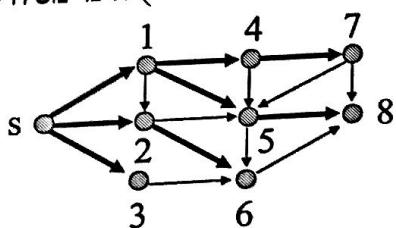
Path(v)

```
if ( $p[v] \neq \text{NULL}$ ) // print path up to v  
    Path( $p[v]$ )  
print v
```

עץ וגרף המסלולים הקצרים ביותר

עץ מסלולים קצרים ביותר מקדוקוד s :

עץ מכון ששורשו s וכל מסלול בו (המתחיל מהשורש) הינו מסלול קצר ביותר מ- s ב- G .



גרף המסלולים הקצרים ביותר מקדוקוד s :
כולל את כל המסלולים הקצרים ביותר מ- s ליתר קדוקוד הגרף.

בנייה עץ המסלולים הקצרים ביותר

הרעיון: ניצור את המערך $[p]$, אשר לכל צומת ישמר את ההוראה שלו בעץ.

BFS-ShorterPathTree(Graph G, Vertex s)

```

Queue Q
for each vertex v do
    d[v] ← ∞
    p[v] ← null
Q ← {s}
d[s] ← 0

```

```

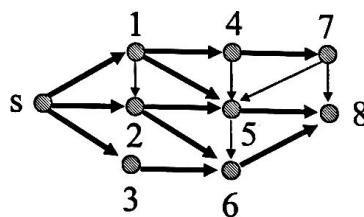
while Q ≠ ∅ do
    u ← Q.Dequeue()
    for each v ∈ Adj[u] do
        if d[v] = ∞ then
            d[v] ← d[u] + 1
            p[v] ← u
            Q.Enqueue(v)

```

עץ ה- BFS הוא

עץ מסלולים קצרים מ- s.

בנייה גרפ' המסלולים הקצרים מ- s



- נריצ' BFS על הגרף מ- s ומחשב לכל קדקוד v את $d[v]$
- נכניס את (v, u) לגרף המסלולים הקצרים מ- s אם $d[v] = d[u] + 1$

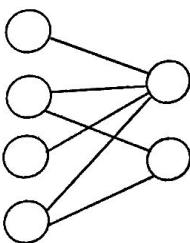
יעילות: $\Theta(n+m)$

הוכחת נכונות: תרגיל הביתה

תרגילים

- הוכחו שאחרי הפעלה BFS בגרף לא-מכוון קשור, כל צלע היא בין קזקודות מאותה רמה, או בין קזקודות מרמות עוקבות.
- האם הטענה נכונה לגרף מכוון?

אם גרף הוא דו-צדדי?



הבעיה:

קלט: גרף קשור לא-מכוון G

פלט: האם G הוא דו-צדדי

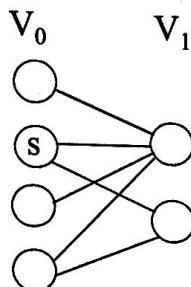
אלגוריתם:

- נריץ BFS מקזקודה s כלשהו ונמצא לכל קזקודה את מרחקו מ- s .
- נעבור על הצלעות ב- G ואם יש צלע (v, u) כך ש- $[v]_d = [u]_p$ אז נחזיר לא.
- אחרת, נחזיר כן.

הערה: אם הגרף לא קשור, אפשר להפעיל את האלגוריתם על כל רכיב קשירות.

הוכחת נכונות

משפט: האלגוריתם מוכיח "כן" אם הגרף דו-צדדי.



הוכחה: כיוון ראשון: נראה שאם האלגוריתם מוכיח כן, הגרף דו-צדדי.

נגיד:

- V_0 קבוצת הקדקודים שמרחיקם מ- s זוגי
- V_1 קבוצת הקדקודים שמרחיקם מ- s אי-זוגי

נראה שצלעות הגרף מחברות רק בין V_0 ל- V_1 ולכן הגרף דו-צדדי.

יהיו $v \in V_0, u \in V_1$. נניח בsvilleה ש- $E \in (v, u)$. קיימות שתי אפשרויות:

1. $d[u] = d[v]$: אבל זה לא נכון, כי הנהנו שהאלגוריתם מוכיח כן.

2. $d[u] \neq d[v]$: בהכרח ההפרש ביניהם גדול מ-1, אבל זה סותר את אי שווויון המשולש (תרגיל).

בדומה מוכיחים עבור $V_1 \in v, u$.

כיוון שני: נראה שאם הגרף דו-צדדי, האלגוריתם מוכיח כן.

נניח שהגרף דו-צדדי. אזי קיימת חלוקה של הקדקודים לשתי קבוצות A ו- B, כך שצלעות הגרף הן רק בין A ל- B.

בה"כ נניח ש- s שייך ל- A.

לכן כל קדקוד מרמה 1 בעץ ה- BFS של s חייב להיות ב- B;

כל קדקוד מרמה 2 בעץ ה- BFS של s חייב להיות ב- A;

כל קדקוד מרמה 3 בעץ ה- BFS של s חייב להיות ב- B;

וכך הלאה (ומכיון שהגרף קשור גנייע לכל הקדקודים).

לכן, $A = V_0$ ו- $B = V_1$.

לפי הנחתנו, אין צלעות בין קדקודים ב- A ובין קדקודים ב- B.

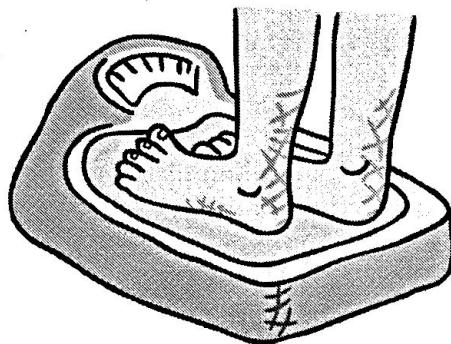
ולכן האלגוריתם יוכיח כן.

תרגילים נוספים – BFS

- כתבו אלגוריתם יעיל אשר בהינתן גרף קשור לא מכוון, בודק אם כל המוגלים בו הם באורך זוגי.

- הוכחו או הפריכו:
 כי G גרף לא מכוון פשוט קשור ודו-צדדי וכי s קדוקוד כלשהו בגרף.
 אז הגרף G כולו הוא גרף המסלולים הקצרים ביותר מ- s .

מסלולים קלים ביותר בגרפים עם משקלים



מסלולים בגרפים עם משקלים

גרף עם משקלים: גרף G ופונקציית משקל $w: E \rightarrow \mathbb{R}$ המתאימה לכל קשת משקל $w(u,v)$

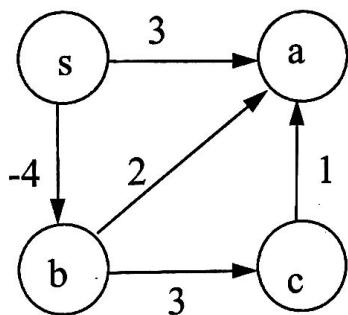
- שימוש נפוץ – המשקלים שמוררים במבנה הנתונים (היכן?)
- ייחן גם משקל לקדוקדים (פחות נפוץ)

משקל של מסלול P : $w(P) = \text{סכום המשקלים לאורכו.}$

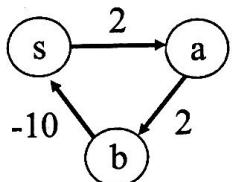
$\delta(s,t) = \text{משקלו של מסלול קל ביותר (מסלול במשקל מינימלי) מ- } s \text{ ל- } t.$

- נקרא גם "המרחק (במשקל) מ- s ל- t " – "weighted distance"
- המסלול נקרא גם "מסלול קצר ביותר (במשקל)" – "shortest path"

דוגמה



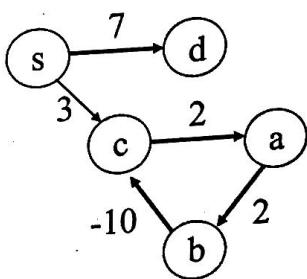
המסלול הקל ביותר מ- s ל- a
הוא: $s \rightarrow b \rightarrow a$



ובגרף זה??

מעגלים שליליים

הגדרה: מעגל שמשקלו הכולל שלילי יקרא מעגל שלילי.

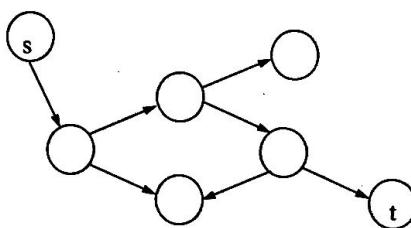


אבחן: מעגל שלילי אפשר להקטין את משקל המסלול עד אינסוף.
 ← בנסיבות מעגל שלילי ניתן שאון מסלול קל ביותר.

בעיות מסלולים קלים ביותר

- (1) **מוצא יחיד:** מהו משקלו של המסלול הקל ביותר (מק"ב) מ- s לכל קדקוד v.
הפלט: מערך $[p]$ כך ש- $[v]$ יהיה המרחק מ-s.
פלט נוספת: מערך $[k]$ שמספק את המסלול עצמו.
- (2) **יעד יחיד:** מהו משקלו של המסלול הקל ביותר (מק"ב) מכל קדקוד v אל t.
- (3) **זוג יחיד:** מהו משקלו של המסלול הקל ביותר (מק"ב) מ- s ל- t.
- (4) **כל הזוגות:** מהו משקלו של המק"ב מכל קדקוד u אל כל קדקוד v.
(הפלט: מטריצה ריבועית)

מצ洲ם מספר בעיות לדיוון



1. בעית היעד היחיד שcola לה בעית המוצא היחיד.

2. את בעית הזוג היחיד נפתר ע"י שימוש באלגוריתם
לבעית המוצא היחיד ושליפת התוצאה עבור t.

סוגי גרפים ומשקלים

משקלים: ההנחה כרגע - מספרים ממשיים כלשהם.
בالمושך נתיחס למקרים מוגבלים (מספרים אי-שליליים...)

גרפים: במושך הדיוון כל הגרפים מכונים ופשותיים.

כיצד נטפל בגרף לא מכון?

כיצד נטפל בקשותות מקבילות?

תכונות של מק"בים

טענה 1: כל תת-מסלול של מסלול מינימלי מ- s ל- v גם הוא מינימלי.

הוכחה: יהי P תת-מסלול של מסלול מינימלי מ- s ל- v .



אם P אינו מינימלי, או יש מסלול קל יותר מ- x -ל- y , בסתיויה למינימליות של המסלול המקורי.

טענה 2: יהי s הקדוק המפני-אחרון על מסלול מינימלי מ- s ל- v .
או $\delta(s,v) = \delta(s,u) + w(u,v)$



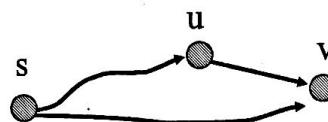
הוכחה: נובע מכך שקטוע המסלול מ- s עד s גם הוא מינימלי (טענה 1).

תכונות של מק"בים - המשך

טענה 3 (אי-שוויון המשולש):

$$\delta(s,v) \leq \delta(s,u) + w(u,v) \quad \text{לכל } u, v \in E$$

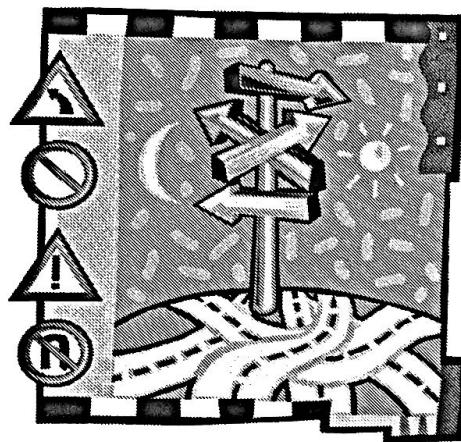
הוכחה: אגף ימין מתייחס למסלול אפשרי מ- s ל- v , העובר דרך u .



טענה 4: אם יש מסלול קל ביזהר מ- s ל- v , אז יש מסלול קל ביותר פשוט.

הוכחה: ניקח מסלול מינימלי. אם הוא מכיל מעגל, נוריד את המעגל ממנו ונקבל מסלול שמשקליו אינם גדולים יותר, כי משקל המעגל ≤ 0 .

מסלולים קלים ביותר בעיית המוצא היחיד



בעיית המוצא היחיד

קלט: $s \in V, w: E \rightarrow \mathbb{R}, G = (V, E)$.

פלט: אם אין מעגל שלילי נגיש מ- s , האלגוריתם יחזיר SUCCESS ובנוסף:

1. משקלי המסלולים הקלים ביותר מ- s .

2. עץ מסלולים קלים ביותר מ- s .

- אחרית – האלגוריתם יחזיר NEGATIVE-CYCLE.

שיטת פורד פתרון ע"י שיפור הדרגת

מבנה נתונים: לכל קדקוד $V \in V$ נשמר שני שדות -

$[v]_d$ - משקלו של המסלול המינימלי מ- s ל- v שגילינו עד כה.

$[v]_p$ - הקדקוד שנמצא לפני v על המסלול המינימלי שגילינו.

בכל שלב ננסה לשפר (להקטין) **לכל** קדקוד v את משקל המסלול המינימלי שגילינו עד כה מ- s ל- v .

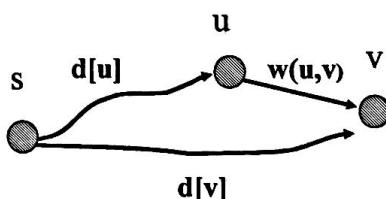
INIT(Vertex s)

$d[s] = 0, p[s] = \text{NULL}$
 $\forall v \neq s: d[v] = \infty, p[v] = \text{NULL}$

אתחל:

פועלת שיפור מקומית Relax

צעד שיפור מקומי עבור קשת (u, v) **Relax(u, v)**



אם המסלול אל v דרך u עדיף מהמסלול שיש לנו עד כה ל- v , נאמץ אותו.

RELAX(Vertex u, Vertex v)

```
if  $d[v] > d[u] + w(u, v)$  then
     $d[v] \leftarrow d[u] + w(u, v)$ 
     $p[v] \leftarrow u$ 
```

קשת (v, u) שmbיאה לעדכון $[v]_d$
 נקראת קשת משפרת.

שיטת פורד

INIT(Vertex s)

$d[s] = 0, p[s] = \text{NULL}$

$\forall v \neq s: d[v] = \infty, p[v] = \text{NULL}$

אלגוריתם:

אתחול:

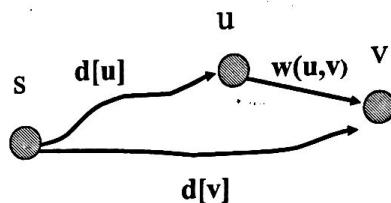
INIT(s)

לולאה:

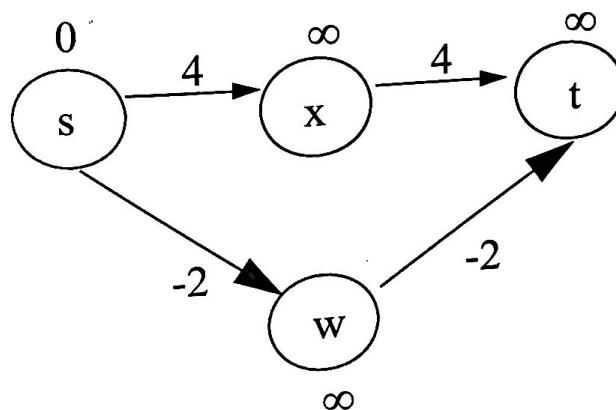
כל עוד קיימת קשת (v, u) משפרת
בצע Relax(u, v)

RELAX(Vertex u, Vertex v)

```
if  $d[v] > d[u] + w(u,v)$  then
     $d[v] \leftarrow d[u] + w(u,v)$ 
     $p[v] \leftarrow u$ 
```



שיטת פורד: דוגמת הרצה



המוצא: s

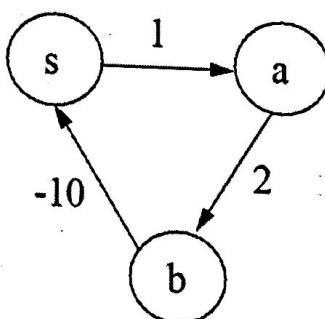
שיטת פורד – נקודות למחשה

אם ניתן להראות שאחרי מספר שיפורים מסוימים, נעצר?

אם האלגוריתם עוצר, האם בטוח שנמצאו מסלולים מינימליים?

כיצד לישם את החיפוש אחר הקשת המשפרת?

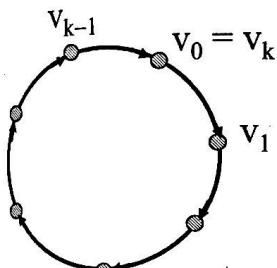
דוגמה



שיטת פורד ומעגליים שליליים

משפט 1: אם יש מעגל שלילי נגיש מ-s, לא ניתן להגיע ל对照检查 שבו אין מה לשפר.

הוכחה: יהי $C = (v_0, v_1, \dots, v_k)$ מעגל שלילי, כאשר $v_k = v_0$, ו- C נגיש מ-s.



נניח בשלילה שאין אף אפשרות לשפר. אזי:

$$\forall 1 \leq i \leq k, \quad d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i)$$

$$\sum_{i=1}^k d[v_i] \leq \sum_{i=1}^k d[v_{i-1}] + \sum_{i=1}^k w(v_{i-1}, v_i) \quad \leftarrow$$

אולם: $\sum_{i=1}^k d[v_i] = \sum_{i=1}^k d[v_{i-1}]$

וזו סתירה להנחה שהמעגל שלילי.

$$\sum_{i=1}^k w(v_{i-1}, v_i) \geq 0 \quad \leftarrow$$

בסיס להוכחות: תכונות של Relax

כל יישום של שיטת פורד מקיים:

טענה א: הערך של $[v]^p$ יורך מונוטונית במהלך האלגוריתם.

טענה ב: בכל שלב שבו $\infty \neq d[v], d[v]^p$ הוא משקלו של מסלול מ-s לו.

בסיס להוכחות: תוכנות של Relax

$$\text{טענה ג: בכל שלב } d[v] \geq \delta(s, v)$$

הוכחה: נניח בשלילה שהטענה אינה נכונה וכי v הקדוק הראשון שאחרי ביצוע $\text{Relax}(u, v)$ מתקיים $d[v] < \delta(s, v)$.

לכן מיד לאחר ביצוע $\text{Relax}(u, v)$ מתקיים:

$$\delta(s, v) > d[v] = d[u] + w(u, v) \geq \delta(s, u) + w(u, v) \geq \delta(s, v)$$

↑ ↑ ↑
 וזו סתירה. $\text{Relax}(u, v)$ v היה ראשון אי-שוויון המשולש

מסקנה: אם בשלב כלשהו מתקיים $d[v] = \delta(s, v)$, אז השוויון יתקיים גם בהמשך האלגוריתם.

תוכנות חלקית של שיטת פורד

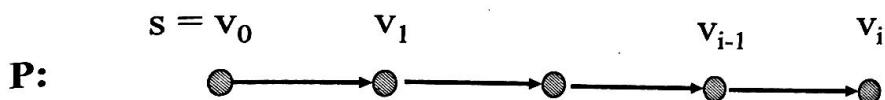
משפט 2: אם אין אפשרות לשפר, או לכל $V \in \mathcal{V}$ מתקיים $d[v] = \delta(s, v)$.
כלומר: אם ישום של שיטת פורד מסתים, מובטחת תוצאה נכונה.

הוכחה: יהיו $V \in \mathcal{V}$ קדוק כלשהו וכי $P = (v_0, v_1, \dots, v_k)$ מסלול מינימלי במשקל מ- s ל- v , כאשר $v = v_k$.



נכיה באינדוקציה על k : $d[v_k] = \delta(s, v_k)$

בעד אינדוקציה: נניח שהטענה נכונה למסלולים מינימליים באורך -1 -ו ו נכיה $-i$.
בסיס: $0 = k$, ואכן לאחר האחול מתקיים $0 = \delta(s, v_0) = [v_0]_d$.



נתון שאך קשת אינה משפרת, ולכן הקשת (v_{i-1}, v_i) אינה משפרת. ומכאן:

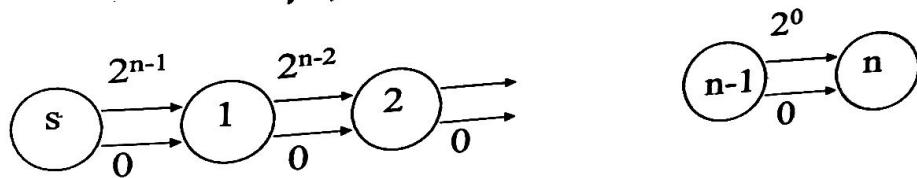
$$d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i) = \delta(s, v_{i-1}) + w(v_{i-1}, v_i) = \delta(s, v_i)$$

הנחה האינדוקציה ↑ ↑

"טענה 2"
(תת מסלול של
מסלול מינימלי)

מכיוון שתמיד $(s, v_i) \geq \delta(s, v_j)$, קיבלנו $\delta(s, v_i) = d$ כנדרש.

דוגמה למספר מעריצי של שיפורים באלגוריתם פורץ



סדר שיפור הקשתות:

1. נשרף את הקשת \rightarrow s שמשקלה 2^{n-1} .

2. נשרף רקורסיבית בגרף ללא s.

3. נשרף את הקשת \rightarrow s שמשקלה 0.

4. נשרף רקורסיבית בגרף ללא s.

$$T(n) = 2 + 2T(n-1) = \Theta(2^n)$$

מספר השיפורים:

אלגוריתמים המישרים את שיטת פורד

- **בלמן פורד** – אלגוריתם כללי למציאת מסלולים קלים ביותר בגרף, כאשר אין מידע על מבנה הגרף ועל המשקלים של הקשתות
- **מק"כム ב- DAG** – אלגוריתם למציאת מסלולים קלים ביותר בגרף אציקלי
- **דיקסטרה** – אלגוריתם למציאת מסלולים קלים ביותר בגרפים בהם כל המשקלים אי-שליליים

אלגוריתם בלמן-פורד

הרעיוון:

האלגוריתם מחולק לאיטרציות, שככל איטרציה עוברים על כל הקשתות ומבצעים Relax.

נראה ש- 1-m איטרציות מספיקות.

הסיבה: באיטרציה ה- k נמצא את כל המסלולים המינימליים במשקל שאורכם בצלעות הוא k .

יעילות: $\Theta(nm)$ 

BELLMAN-FORD(Graph G, Weight w, Vertex s)

INIT(s)

```

for i ← 1,..., n - 1 do      // Main loop
    for each (u,v) ∈ E do
        RELAX(u,v)

for each (u,v) ∈ E do          // Check Termination
    if d[v] > d[u] + w(u,v) then
        return "NEGATIVE-CYCLE"
return "SUCCESS"

```

RELAX(Vertex u, Vertex v)

```

if d[v] > d[u] + w(u,v) then
    d[v] ← d[u] + w(u,v)
    p[v] ← u

```

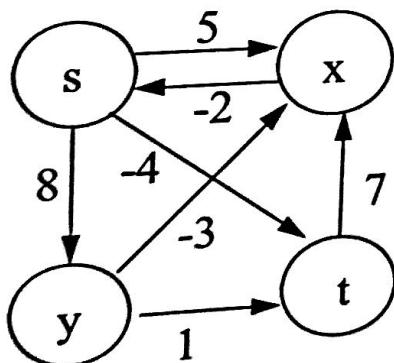
INIT(Vertex s)

```

d[s] = 0, p[s] = NULL
∀v ≠ s: d[v] = ∞, p[v] = NULL

```

דוגמת הרצה



המוצא: s

הקשנות

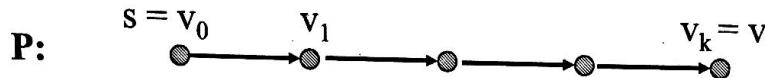
- | | |
|--------|---|
| sx(5) | • |
| xs(-2) | • |
| sy(8) | • |
| yt(1) | • |
| tx(7) | • |
| st(-4) | • |
| yx(-3) | • |

הוכחת נכונות

משפט 1: אם אין מעגלים שליליים אז בסיום האלגוריתם מזריר SUCCESS
ומתקיים: $\delta(s, v) = \min_{v \in V} d[s, v]$.

הוכחה:

nocivity באינדוקציה על k שאחרי האיטרציה ה- k , לכל קדוקוד v כך שיש מסלול כל
ביוור מ- s ל- v המכיל k קשתות, מתקיים $d[v_k] = \delta(s, v_k)$.



אפשר להניח כי P מסלול פשוט (כי אין מעגלים שליליים), ולכן $1-n \leq k$.

בסיס: $k=0$, ואכן לאחר האתחול מתקיים: $0 = \delta(s, v_0) = d[v_0]$.

צעד אינדוקציה: נניח נכונות למסלולים עם קלים ביוור עם $1-n$ קשתות ונוכיה ל- i .

באיטרציה ה- i מבצעים פעולה Relax(v_{i-1}, v_i) כי עוברים על כל הצלעות.

לאחר ביצוע הפעולה מתקיים:

$$d[v_i] \leq d[v_{i-1}] + w(v_{i-1}, v_i) = \delta(s, v_{i-1}) + w(v_{i-1}, v_i) = \delta(s, v_i)$$

↑ ↑ ↑
 Relax(v_{i-1}, v_i) הנחת האינדוקציה "טענה 2"
 ↓ ↓ ↓
 (תת מסלול של
מסלול מינימלי)

מכיוון שתמיד $\delta(s, v_i) \geq d[v_i]$, קיבלנו $\delta(s, v_i) = \min_{v \in V} d[s, v]$ כנדרש.

לכן אחרי האיטרציה ה- $(1-n)$ כל הערכות יעדכנו בצורה נכונה, כי לכל קדוקוד
נגיש מ- s , קיימ מסלול מינימלי מ- s אליו שלאורךו לכל היותר $1-n$ קשתות.

משפט 2: אם יש מעגל שלילי נגיש מ- s , אז יוחזר NEGATIVE-CYCLE.

הוכחה: ראיינו שכאשר יש מעגל שלילי, לא ניתן להגיע למצב שאין מה לשפר.

לכן אחת הבדיקות בלולאה האחורונה תצליח ויוחזר NEGATIVE-CYCLE.

בלמן ופורד

רייצ'רד בלמן (1920-1984): מתמטי אמריקני.
הציג את הרעיון של תכנות דינמי בسنة 1953.



לستر פורד (1927-2017): מתמטי אמריקני.
התמחה במיוחד בעיות של זרימה ברשתות.

פורד ובלמן פיתחו את אלגוריתם בלמן-פורד בנפרד
אבל האלגוריתם נקרא על שם שניהם.

תרגילים

תרגיל 1: נתון גרף ממוקן G עם משקלים על הקשתות.
דרוש אלגוריתם ייעיל שמכריע האם יש מעגל שלילי ב- G .

תרגיל 2: נתון גרף ממוקן אציקלי G עם משקלים על הקשתות.
האם מעבר על הקשתות על ידי BFS מ- s (חוך ביצוע פעילות Relax) ימצא את המסלולים המינימליים במשקל מ- s ?

רעיון לפתרון יותר מהיר

אבחנה: יהי $(v_0, v_1, \dots, v_k) = P$ מסלול קל ביותר מ- s ל- v ,
כאשר $s = v_0$, $v_k = v$.



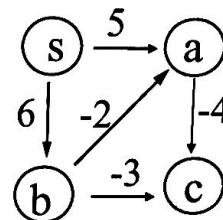
אם נבצע Relax על קשתות המסלול לפי סדר הופעתן במסלול (משמאל לימין)
נקבל $\delta(s, v) = d[v]$, וכל זאת בעבור שיפורים אחד בלבד!

הבעיה: איך יודעים מהו הסדר הנכון?

מק"בים בגרף אציקלי

אלגוריתם:

- נמיין טופולוגית את קדקודיו הגרף.
- אתחול המרחקים: $\text{INIT}(s)$
- לכל קדקוד $V \in V$ על פי הסדר הטופולוגי:
 $\text{Relax}(u, v)$ לכל קשת (v, u) נבצע

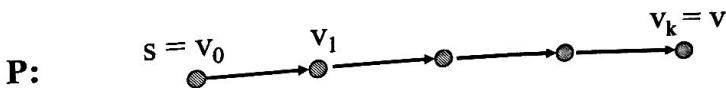


יעילותות: $\Theta(n+m)$

הוכחת נכונות

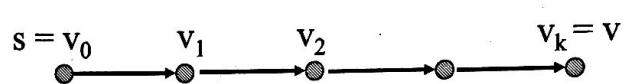
משפט: בסיום האלגוריתם, $d[v] = \delta(s, v) \quad \forall v \in V$.

הוכחה: יהיו $P = (v_0, v_1, \dots, v_k)$ מסלול קל ביותר.



מכיוון שהאלגוריתם עובר על הקדקודים בסדר טופולוגי, נשפר על קשותות המסלול P לפי סדר הופעתן במסלול.
 קל להראות שבתום המעבר על כל הקשותות, לכל v מתקיים: $d[v] = \delta(s, v)$.

משקלים אי-שליליים – האלגוריתם של דיקסטרה



אבחנה:

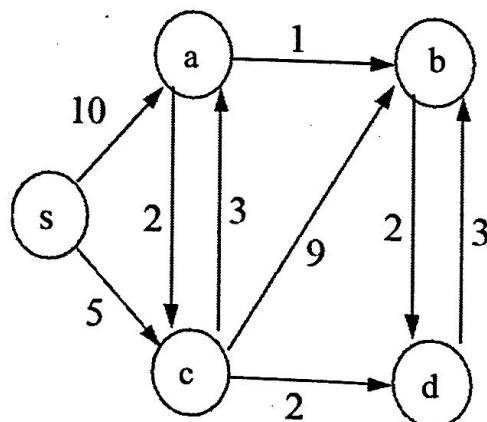
מכיוון שמשקל הקשות אי-שליליים, סדר הקדוקודים במסלול הוא סדר עולה של מרחקים מ-s.

אלגוריתם:

- נתחל את $[n]p$ כרגיל.
- כל עוד לא ביקרנו בכל הקדוקודים:
 - יהי u קדוקוד בעל $[n]p$ מינימלי, מלאה שטרם בוקרו.
 - נשפר על כל הקשות לשכניו של u .

איזה טיפוס נתונים מופשט יעזר לנו למצוא קדוקוד u כזה בכל איטרציה?

דוגמת הרצה



DIJKSTRA(Graph G, Weight w, Vertex s)

PriorityQueue Q

INIT(s)

// Build priority queue

Q.Build(V, d)

// SEARCH

while Q ≠ ∅ **do**

 u ← Q.Delete-Min()

for each v ∈ Adj[u] **do**

if d[v] > d[u] + w(u,v) **then**

 d[v] ← d[u] + w(u,v)

 p[v] ← u

 Q.Decrease-Key(v, d[v])

יעילות של מבני נתונים למימוש תור עדיפות

שיפור מפתח	הוצאת המינימום	בנייה תור עם n איברים	ישום
Q.DecreaseKey (Item,NewKey)	Q.DeleteMin()	Q.Build (Items,Keys)	רשימה ממוינת
O(n)	O(1)	O(n log n)	ץ חיטש מאוזן
O(log n)	O(log n)	O(n log n)	ערימה בירנית
O(log n)	O(log n)	O(n)	מערך לא ממויין
O(1)	O(n)	O(n)	

יעילות האלגוריתם של דיקסטרה

זמן הריצה נשלט ע"י פעולות התור!

<u>מימוש Q על ידי מערך:</u>	
$\Theta(n)$	בנייה התור:
$\Theta(n^2)$	n פעולות :DeleteMin
$\Theta(m)$	m פעולות :DecreaseKey

<u>מימוש Q על ידי ערימת מינימום:</u>	
$\Theta(n)$	בנייה התור:
$O(n \log n)$	n פעולות :DeleteMin
$O(m \log n)$	m פעולות :DecreaseKey

מימוש Q על ידי ערימת פיבונצ'י:

הוכחת נכונות

טענה עזר: בגרף שבו משקלים אי-שליליים, אם u נמצא על מק"ב מ- s ל- v , אז $\delta(s,u) \leq \delta(s,v)$



הוכחת נכונות

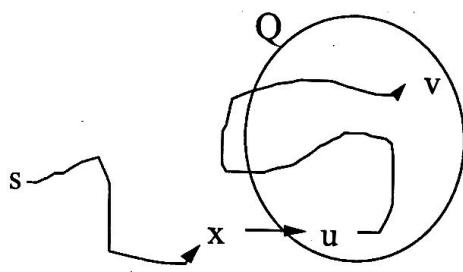
משפט: בסיום האלגוריתם של דיקסטרה מתקיים $\delta(s,v) = d[v]$ לכל $v \in V$.

הוכחה: נוכחה באינדוקציה על סדר הוצאת הקודמים מהתור Q שלכל קדוק v שיצא מהתור מתקיים $\delta(s,v) = d[v]$.

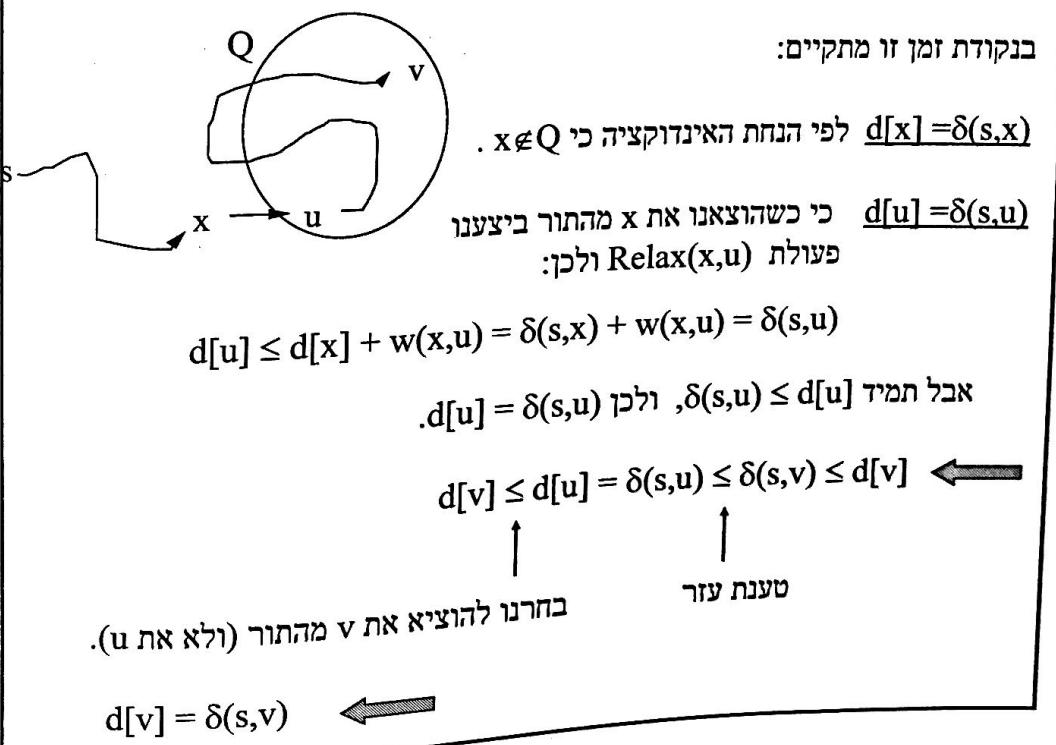
ביסיס: הקדוק s יוצא ראשון ואז גם $0 = \delta(s,s) = d[s]$.

שלב האינדוקציה: יהיו v הקדוק שיצא מהתור באיטרציה ה- t .

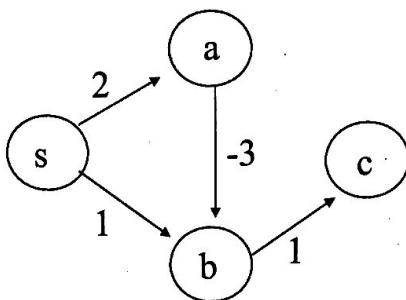
נסתכל על מסלול קל ביותר מ- s ל- v :



יהי u הקדוק הראשון על המסלול הזה שנמצא עדין בתור, ויהי x הקדוק מיד לפניו.
(יתכן כמובן ש- $x = s$ או $v = u$).



דוגמה לכך שהאלגוריתם של דijkסטרה טועה כשייש משקלים שליליים



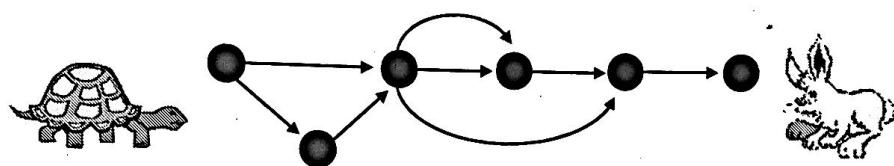
דijkסטרה



: (1930-2002) Edsger W. Dijkstra
מהנדס וחוקר הולנדי בתחום מדעי המחשב.
יש לו תרומה למדעי המחשב בתחוםים רבים כגון
מערכות הפעלה, קומפיילרים, היישוב מבוזר ומקבילי.
זכה בפרס טיוריינג בשנת 1972.

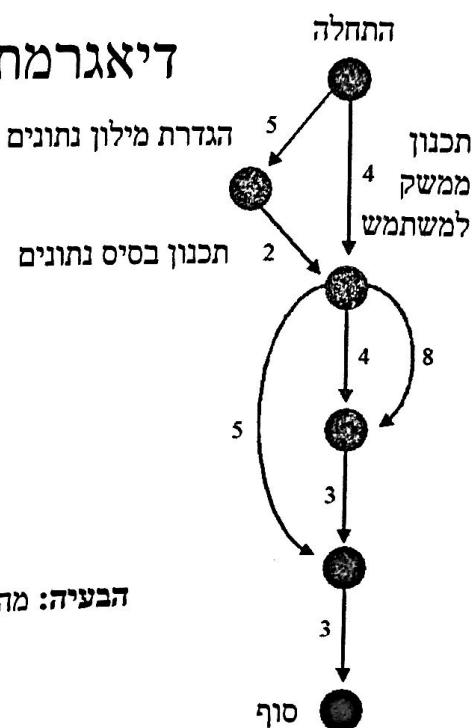
- 1959 – Shortest-Path and Minimum Spanning Tree algorithms
- 1965 – semaphores and the Dining Philosophers Problem
- 1968 – “*Go To Statement Considered Harmful*”

שימושים באלגוריתמים לבעיה המוצאה היחיד



דיאגרמת PERT

PERT - Program Evaluation
and Review Technique



הבעיה: מהו משך הזמן הדרוש לסיום הפרויקט?

מסלול במשקל מקסימלי ב-DAG

פתרון 1: רזוקציה לבועית המסלולים הקלים ביותר.

- באיזה אלגוריתם נפתרת הבועה המתבקשת? מה עילוות הפתרון?

פתרון 2: שינוי האלגוריתם כך שימצא מסלולים מקסימליים במשקלם.

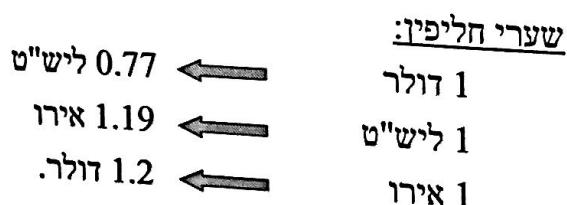
השינויים:

- אתחול: $d[v] = -\infty \quad \forall v \neq s$
- פעולה Relax: הופכים את כיוון האי-שוויון.

יש להוכיח את נכונות האלגוריתם החדש!

דוגמה – מציאת ארביטראז' Arbitrage

המטרה: משתמשים בפערים בשער החליפין של מטבעות כדי להחליף כספים ולהרוויח.



בדולר אחד אפשר לקנות $0.77 \times 1.19 \times 1.2 = 1.099$ דולרים.

הכללה:

- יש n מטבעות c_1, c_2, \dots, c_n בגודל $n \times n$ של שער חליפין: $R(i,j) =$ כמות המטבעות מסווג j שאפשר לקנות תמורה מטבע מסווג i .

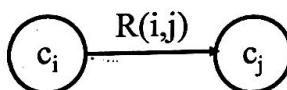
הבעיה: האם יש סדרה של מטבעות $c_{i1}, c_{i2}, \dots, c_{ik}$ כך ש-

$$R(i_1, i_2) \times R(i_2, i_3) \times \dots \times R(i_{k-1}, i_k) \times R(i_k, i_1) > 1$$

יצוג הבעיה ע"י גרף מכובן:

- לכל סוג מטבע יש קודקוד,

- יש קשת ממטבע למטבע משקללה הוא שער החליפין.



השאלה: האם יש בגרף מעגל שמכפלת משקלותיו גדולה מ-1?

דרישה סדרה i_1, i_2, \dots, i_k כך ש:

$$R(i_1, i_2) \times R(i_2, i_3) \times \dots \times R(i_{k-1}, i_k) \times R(i_k, i_1) > 1$$

לאחר הפעלת \log נקבל:

$$\log R(i_1, i_2) + \log R(i_2, i_3) + \dots + \log R(i_{k-1}, i_k) + \log R(i_k, i_1) > 0$$

נכפיל את כל המשקלים ב- (-1) ונקבל:

$$-\log R(i_1, i_2) - \log R(i_2, i_3) - \dots - \log R(i_{k-1}, i_k) - \log R(i_k, i_1) < 0$$

קיבלנו רדוקציה של הבעיה שלנו לבעיית מציאת מעגל שלילי בגרף. ←
המשקל של קשת (j,i) בגרף יהיה $-\log R(i,j)$

תרגיל - סולמות וחבלים

נתון לוח משחק הכלול משבצות וקובייה הממוספרת 1-6.



41	42	43	44	45	46	47	48 עד
33	34	35	36	37	38	39	40
25	26	27	28	29	30	31	32
17	18	19	20	21	22	23	24
9	10	11	12	13	14	15	16
1	2	3	4	5	6	7	8 מוצא

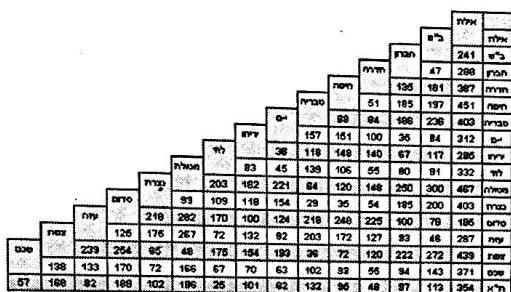
- מהו מספר הטלות המינימלי של הקובייה שמאפשרות להגעה מהמוצא ליעד?
- מהו סכום הנקודות המינימלי של הטלות קובייה שמאפשרות להגעה מהמוצא ליעד?

תרגילים נוספים

תרגיל 1: נתון גרף מכoon ($G = (V, E)$) עם משקלים אי-שליליים על הקשתות, כך שכל קשת צבעה באדום או בכחול. בנוסף נתון זוג קדקודים $V \in t, s$. כתבו אלגוריתם עיליל כל הנתון, המוצא את משקל המסלול הקל ביותר מס-ל-ז, המכיל מס' זוגי של קשתות אדומות.

תרגיל 2: נתון גרף מכoon ($G = (V, E)$) עם משקלים שליליים על הקשתות, יהיו $V \in s$. כתבו אלגוריתם עיליל כל הנתון, המוצא את משקלי המסלולים הקלים ביותר מס-ל-הכל הקדקודים. אם קיימים מעגל שלילי בגרף – האלגוריתם יוציא הודעה

מסלולים קלים ביותר בעיית כל הזוגות



מסלולים קלים ביותר בין כל זוגות הקדקודים

קלט: גרף ממוקן פשוט G עם משקלים כלשהם.

פלט: מסלול קל ביותר מ- s ל- v , לכל זוג קדקודים (v, u).

פתרון קל: הריצת בלמן-פורד מכל קדקוד התחלה. שמרית התוצאות
במערכות $[v][u][d], [v][u][p]$.

יעילות: $n \times$ זמן הריצה של בלמן-פורד, כולם $\Theta(mn^2)$

פתרון מותחכם: אלגוריתם פלייז-וורשל

זיכרון לתוכנות דינמי

מתאים לביעות שיכולות להיפטר גם ברקורסיה, אבל ניגשים לפתורן בצורה "bottom up" ומקבלים פתרון יעיל יותר.

דוגמה: חישוב מקדים בינומיים

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

זהות פסקל:

בנייה טבלה של הערכים בסדר עולה של n יعلاה יותר מביצוע רקורסיבי.

דוגמה: חישוב מקדים בינומיים

האפשרויות לבחור k איברים מתוך n

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

מחלקות לשני סוגים:

איבר n נבחר!

איבר n לא נבחר!

יישום לבעיית המסלולים הקלים

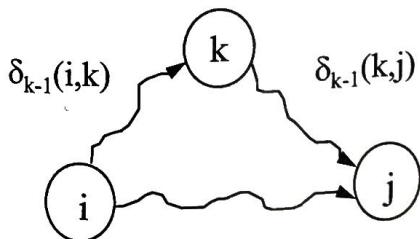
הנחה: $V = \{1, 2, \dots, n\}$

גדר $\delta_{i,j} = \text{משקל מסלול מינימלי מ- } i \text{ ל- } j \text{ שעובר רק דרך קדוקדים מהקובוצה } \{1, \dots, k\} \text{ (פרט לקדוקדי הקצה } j, i).$



$\delta_n(i,j) = \delta(i,j) \iff \text{משקל מסלול מינימלי מ- } i \text{ ל- } j \text{ שਮותר לו לעבור דרך קדוקדים מ- } \{1, \dots, n\}.$

חישוב רקורסיבי של δ_k



רעיון התכונות הדינמי:

$$\delta_k(i,j) = \min \{ \delta_{k-1}(i,j), \delta_{k-1}(i,k) + \delta_{k-1}(k,j) \}$$

לא בחרנו לעبور בקדוקוד k !

בחרנו לעبور בקדוקוד k !

זה נכון כשאין מעגלים שליליים כי אין סיבה לעبور ב- k יותר מפעם אחת.

תנאי עצירה: $\delta_0(i,j) = 0$, $\delta_0(i,i)$ ו- $\delta_0(i,j) = \infty$ ואחרת.

מבנה הנתונים

- מערך דו-מימדי d עבור המרחקים

$d[i,j] = \text{משקל המסלול המינימלי מ- } i \text{ ל- } j \text{ שמצאנו עד כה.}$

- מערך דו-מימדי p עבור המסלולים

$p[i,j] = \text{הקודוד האחרון לפני } j \text{ על המסלול המינימלי מ- } i \text{ ל- } j \text{ שמצאנו עד כה.}$

אתחל מערכים אלו יעשה לפי תנאי העצירה:

$$\delta_0(i,i) = 0$$

$$\text{אם } (i,j) \text{ קשת, } \delta_0(i,j) = w(i,j)$$

$$\text{אחרת. } \delta_0(i,j) = \infty$$

FLOYD(Graph G, Weight w)

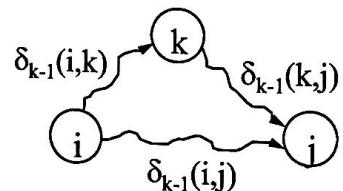
```
Weight d[1..n, 1..n]  $\leftarrow \{\infty, \dots, \infty\}$ 
int p[1..n, 1..n]  $\leftarrow \{\text{NULL}, \dots, \text{NULL}\}$ 
```

// INIT

```
for i  $\leftarrow 1, \dots, n$  do
    d[i,i]  $\leftarrow 0$ 
for each (i,j)  $\in E$  do
    d[i,j]  $\leftarrow w(i,j)$ ; p[i,j]  $\leftarrow i$ ;
```

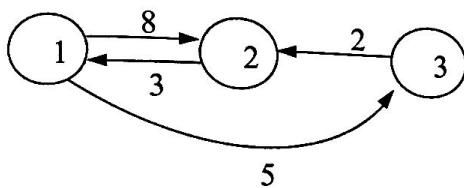
// MAIN LOOP

```
for k  $\leftarrow 1, \dots, n$  do
    if d[k,k]  $< 0$  then
        print "NEGATIVE CYCLE"; stop;
    for i  $\leftarrow 1, \dots, n$  do
        for j  $\leftarrow 1, \dots, n$  do
            if d[i,k] + d[k,j]  $< d[i,j]$  then
                d[i,j]  $\leftarrow d[i,k] + d[k,j]$ 
                p[i,j]  $\leftarrow p[k,j]$ 
```



יעילות: $\Theta(n^3)$
זיכרון: $\Theta(n^2)$

דוגמת הרצה



	1	2	3
1	0	8	5
2	3	0	∞
3	∞	2	0

δ_0

	1	2	3
1	0	7	5
2	3	0	8
3	5	2	0

δ_3

	1	2	3
1	0	8	5
2	3	0	8
3	5	2	0

δ_2

	1	2	3
1	0	8	5
2	3	0	8
3	∞	2	0

δ_1

הוכחת נכונות

משפט: אם אין מעגלים שליליים או בסיום האלגוריתם $d[i,j] = \delta_n(i,j)$.

(זיכרון: $\delta_n(i,j) = \delta(i,j)$)

הוכחה: נוכיח באינדוקציה שבסיום האיטרציה ה- k מתקיים $d[i,j] = \delta_k(i,j)$
בסיס: $d[i,j] = \delta_0(i,j) = 0$, ואכן לאחר האתחול $d[i,j] = \delta_0(i,j)$.
שלב האינדוקציה: נניח נכונות ל- $(k-1)$ ונווכיה ל- k .
באייטרציה k האלגוריתם מחשב כך:

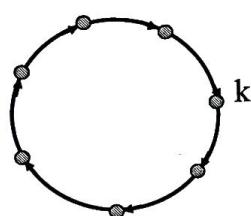
```
if  $d[i,k] + d[k,j] < d[i,j]$  then
     $d[i,j] \leftarrow d[i,k] + d[k,j]$ 
```

לפי הנחת האינדוקציה ב- d נמצאים בתחילת האיטרציה ה- k ערכי δ_{k-1} .
לכן הערך החדש של $[j,i]d$ יהיה (כי אין מעגלים שליליים):

$$\min \{ \delta_{k-1}(i,j), \delta_{k-1}(i,k) + \delta_{k-1}(k,j) \} = \delta_k(i,j)$$

משפט: אם יש מעגל שלילי האלגוריתם יודיע על כך.

הוכחה: יהיו C מעגל שלילי ויהי k הקדוקו ב- C שמספרו הסידורי מקסימלי.



בדומה למשפט הקודם מראים שבסיום האיטרציה ה- $k-1$ האלגוריתם כבר מוצא את משקלו של המסלול המינימלי מ- k לעצמו שעובר רק דרך קדוקדים $> k$ (זהו $(\delta_{k-1}(k,k))$).

לכן בתחילת האיטרציה ה- k יתקיים: $d[k,k] < 0$

◀ האלגוריתם יודיע שיש מעגל שלילי.

סיכום – אלגוריתמים למק'בים

כל הזוגות	מוצא יחיד	בכל המקרים: גרף מכונן פשוט
	בלמן-פורד $\Theta(nm)$	ללא מגבלות
	דייקסטרה $\Theta(m + n \log n)$	משקלים אי- שליליים
		גרף אציקלי

סגור טרנסיזיטיבי – אלגוריתם וורשל

$G^* = (V, E^*)$ הregon הטרנסיזיטיבי-רפלקסיבי של G הוא הגרף

$$E^* = \{(i,j) \mid i \xrightarrow{G} j\}.$$

רעיון אינדוקטיבי: בשלב k נזהה מסלול מ- i ל- j שעובד רק בקדושים $\geq k$.

יש שני מקרים:

(1)

(2)

// Assume $V = \{1, 2, \dots, n\}$.
// $T[i, j]$ will be 1 if there is a path from i to j , 0 otherwise.
Transitive Closure(Graph G)

```
// INIT
for i ← 1,..., n do
    for j ← 1,..., n do
        if (i = j or (i,j) ∈ E) then
            T[i,j] ← 1
        else T[i,j] ← 0

// Compute closure.
for k ← 1,..., n do
    for i ← 1,..., n do
        for j ← 1,..., n do
            T[i,j] ← (T[i,j] or (T[i,k] and T[k,j]))
```

סגור טרנסיטיבי ב-DAG

יהי G גרף מכוון. לקודוד v , נסמן ב- $R(v)$ את קבוצת הקדקודים הנגושים מ- v .
או אפשר לרשום את המשווה הרקורסיבית הבאה:

$$R(v) = \{v\} \cup \left(\bigcup_{u \in \text{Adj}[v]} R(u) \right)$$

יש מסלול מ- v לקדקוד כלשהו t

אם ורק אם

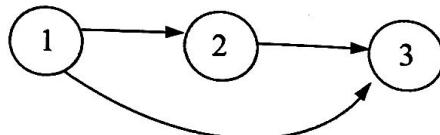
$$t = v$$

או

יש מסלול אחד השכנים של v ל- t .

במילים:

אם הגרף חסר מעגלים, אפשר לישם זאת כתוכנית רקורסיבית, והיא
תחשב את $R(v)$ לכל קדקוד שנבחר



... אבל בזמן מערכי.

הפתרון: תכנות דינמי!

Transitive Closure(Graph G)

int T[1..n, 1..n] = {0,...,0}

List L \leftarrow Topological-Sort(G)

for each v **in** reverse(L)

T[v,v] \leftarrow 1

for each u \in Adj[v]

for j \leftarrow 1,..., n **do**

T[v,j] \leftarrow T[v,j] **or** T[u,j]

$\Theta(n^2 + nm)$ יעילות:

$\Theta(n^2)$ זיכרון:

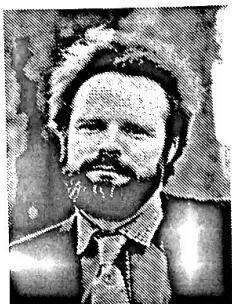
תרגיל

גרף ייחסין הוא גרף שקודודיו מייצגים בני אדם, וקשת (v,u) פירושה ש-u הינה הורה של-v.

בהתנען גרף כזה, רוצים לבנות גרף אחר שבו תהיה קשת (v,u) אם ורק אם v הוא צאצא של u.

באיזה אלגוריתם השתמש ולמה?

רוברט פלויד



רוברט פלויד (1936-2001):

חוקר אמריקני בתחום מדעי המחשב.

סיים בית ספר תיכון בגיל 14 ותואר ראשון בגיל 17.

זכה בפרס טיוריינג בשנת 1978.

- 1962 – Transitive-Closure and Shortest-Path algorithms
- 1964 – Building a heap in linear time.
- 1967 – a paper that started formal program verification
- 1973 – selection in linear time (w/ Blum, Pratt, Rivest, Tarjan)
- 1975 randomized selection (with Rivest)

סטפן וורשל



סטפן וורשל (1935-2006):

חוקר אמריקני בתחום מדעי המחשב.

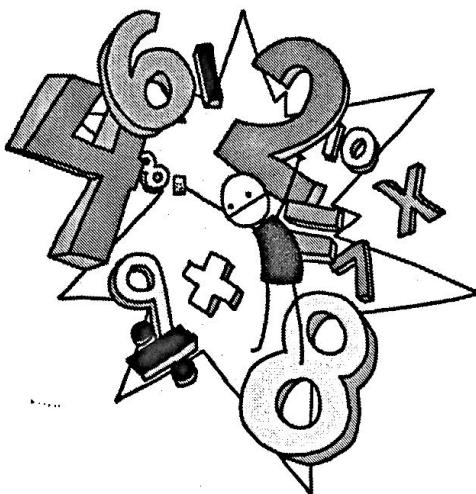
עסק בתחום מערכות הפעלה, קומפיילרים, שפות חכמת.

וורשל הוכיח את נכונות האלגוריתם לסגור טרנסיטיבי

לאחר התurbות בין חבר מי יוכיח קודם אם האלגוריתם נכון.

וורשל הוכיח ראשוני שהאלגוריתם נכון וזכה בבקבוק רום.

כפל מהיר של מספרים ומטריצות



כפל של מספרים ארוכים

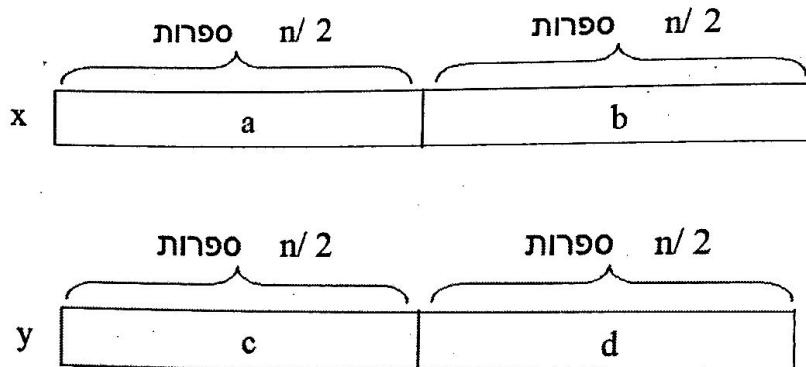
ע,א שני מספרים טבעיים גדולים באורך מ' כל אחד,
שנשמרים בשני מערכימ', כר שהספרה ה- נמצאת בתא ה-
ן.

$$x = [3,4,5,6] \quad y = [1,7,8,9]$$

תרגיל: מה ייעילות האלגוריתם שמכפיל את ע,א כמו תלמידים בבית ספר?

כפל רקורסיבי

נחלק את x ואת y לשני חלקים באורך $n/2$.



$$x = a \cdot 10^{n/2} + b, \quad y = c \cdot 10^{n/2} + d$$

מתכוניות זהויות:

$$x = a \cdot 10^{n/2} + b, \quad y = c \cdot 10^{n/2} + d$$

מתכבלת הנוסחה הרקורסיבית הבאה:

$$\begin{aligned} x \cdot y &= (a \cdot 10^{n/2} + b) \cdot (c \cdot 10^{n/2} + d) \\ &= (a \cdot c) \cdot 10^n + (a \cdot d + b \cdot c) \cdot 10^{n/2} + (b \cdot d) \end{aligned}$$

נוסחת נסיגה שמתארת את זמן הרכיצה:

פתרון הנוסחה:

Karatsuba האלגוריתם של

$$x \cdot y = (a \cdot c) \cdot 10^n + ((a+b) \cdot (c+d) - a \cdot c - b \cdot d) \cdot 10^{n/2} + (b \cdot d)$$

נוסחת נסיגה שמתארת את זמן הריצה:

פתרון הנוסחה:

וקיבלנו כפל מספרים לפחות מ- $O(n^2)$

כפל מטריצות

יהיו A , B שתי מטריצות של מספרים שלמים, בגודל $n \times n$,
ותהי C המטריצה השווה למינימלטן.

$$\begin{pmatrix} c_{11} & c_{12} & \cdots & c_{1n} \\ c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & \cdots & c_{nn} \end{pmatrix} =$$
$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \times \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \cdots & b_{nn} \end{pmatrix}$$

נחלק את A, B ו- C לארבעה חלקים שווים:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

כל חלק הוא תת-מטריצה בגודל $(n/2) \times (n/2)$.

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \quad \leftarrow$$

נחשב רקורסיבית את המכפלות!

$$\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \times \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

$C_{11} = A_{11} \times B_{11} + A_{12} \times B_{21}$
$C_{12} = A_{11} \times B_{12} + A_{12} \times B_{22}$
$C_{21} = A_{21} \times B_{11} + A_{22} \times B_{21}$
$C_{22} = A_{21} \times B_{12} + A_{22} \times B_{22}$

נחשב רקורסיבית את
4 החלקים של C:

נוסחת נסיגה שמתארת את זמן הריצה:

פתרון הנוסחה:

האלגוריתם של Strassen

$$\begin{aligned}
 M_1 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\
 M_2 &= (A_{21} + A_{22}) \times B_{11} \\
 M_3 &= A_{11} \times (B_{12} - B_{22}) \\
 M_4 &= A_{22} \times (B_{21} - B_{11}) \\
 M_5 &= (A_{11} + A_{12}) \times B_{22} \\
 M_6 &= (A_{21} - A_{11}) \times (B_{11} + B_{12}) \\
 M_7 &= (A_{12} - A_{22}) \times (B_{21} + B_{22})
 \end{aligned}$$

מחשב רקורסיבית את
המכפלות הבאות:



$$\begin{aligned}
 C_{11} &= M_1 + M_4 - M_5 + M_7 \\
 C_{12} &= M_3 + M_5 \\
 C_{21} &= M_2 + M_4 \\
 C_{22} &= M_1 - M_2 + M_3 + M_6
 \end{aligned}$$

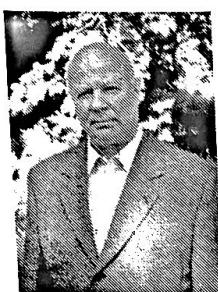
ארבעה חלקים של C הם:

מה ייעילות האלגוריתם הרקורסיבי המתקבל?

נוסחת נסיגה שמתארת את זמן הריצה באלגוריתם של Strassen:

פתרון הנוסחה:

וקיבלנו כפל מטריצות בפחות מ- $O(n^3)$



(2008 – 1937) Anatoly Karatsuba: מתמטיקאי רוסי.

ב- 1960 קולמגורוב שאל האם נתן להכפל מספרים בפחות מ- n^2 פעולות ושיער שלא.

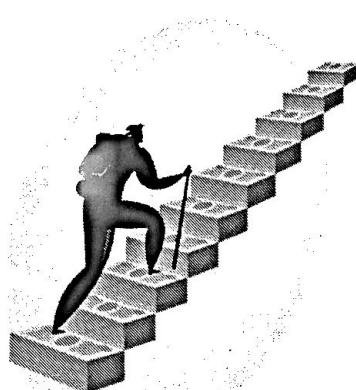
תווך שבוע גיליה קרטסובה שהיא אז סטודנט בן ה- 23 שזה אפשרי, והציג את האלגוריתם הראשון הרקורסיבי לכפל של מספרים בפחות מ- $O(n^2)$ פעולות.



(1936) Volker Strassen: מתמטיקאי גרמני.
גיליה את האלגוריתם הראשון לכפל מטריצות בפחות מ- $O(n^3)$.

המציא גם אלגוריתם מהיר לחישוב המטריצה ההפכית, ולבדיקה הסתברותית לראשוניות.
זכה בפרסים רבים (מדליית קנטור, פרס קנות' ועוד).

תכנות דינמי



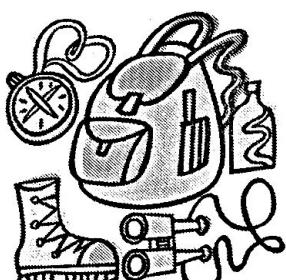
בעיה Knapsack

גłów:

1. $\{s_1, s_2, \dots, s_n\} = S$ – קבוצה של n עצמים,
כאשר לכל עצם s_i מותאמים שני ערכים:

- w_i – הנפח של s_i
- b_i – הערך של s_i

2. W – נפח מקסימלי לכל החפצים בתרגיל



פלט:

תת-קבוצה של S אשר נפח הכלול אינו עולה את W
וסכום ערכי העצמים בה מקסימלי.
(ביחס לשאר תת הקבוצות של S שנפחם אינו עולה את W).



לוגמה ל- Knapsack

	נפח (w_i)	ערך (b_i)
s_1	6	30
s_2	3	14
s_3	4	18
s_4	2	9

נפח התרמיל: $W=10$

הפתרון: בחירת s_1 ו- s_3 אינו עבר את הנפח הכלול 10, וערכם הכולל הוא 48.

הפתרון הרקורסיבי

הרעיון: או שמכניסים את s_n לתרמיל או לא.

Knapsack({ s_1, s_2, \dots, s_n }, W)

```

if (n = 1)
    if ( $w_1 > W$ )
        return 0;
    else
        return  $b_1$ ;
else
    if ( $w_n > W$ )
        return Knapsack({ $s_1, s_2, \dots, s_{n-1}$ }, W);
    else
        with_last  $\leftarrow b_n + \text{Knapsack}(\{s_1, s_2, \dots, s_{n-1}\}, W - w_n)$ ;
        without_last  $\leftarrow \text{Knapsack}(\{s_1, s_2, \dots, s_{n-1}\}, W)$ ;
        return max(with_last, without_last);
    
```

הרעיון של התכנות הדינמי

נגידר את (i, w) ב- B להיות סכום הערכיהם המקסימלי שאפשר להרכיב מ- $s_i, s_1, s_2, \dots, s_n$ שונפחים הכלול לא עובר את w .

נחפש את $B(n, W)$: מותר להשתמש בכל העצמים ונפח התרמייל W .

מתקיים:

$$w < w_i$$

$$w \geq w_i$$

$$B(i, w) = \begin{cases} B(i-1, w) \\ \max \left\{ B(i-1, w), \underbrace{b_i + B(i-1, w - w_i)}_{\text{א: לא נכנס לתרמייל}} \right\} \end{cases}$$

א: אין כניסה לתרמייל

$$B(1, w) = \begin{cases} 0 & w < w_1 \\ b_1 & w \geq w_1 \end{cases}$$

תנאי התחלה:

פתרון בעזרת תכנות דינامي

	0	1	2	3	4	5	6	7	8	9	10
$w_1=6$	0	0	0	0	0	0	30	30	30	30	30
$b_1=30$											
$w_2=3$	0	0	0	14	14	14	30	30	30	44	44
$b_2=14$											
$w_3=4$	0	0	0	14	18	18	30	32	32	44	48
$b_3=18$											
$w_4=2$	0	0	9	14	18	23	30	32	39	44	48
$b_4=9$											

Knapsack({s₁, s₂, ..., s_n}, W, B)

```
int V[1...n, 0...W]

for w ← 0,..., W do
    if (w < w1)
        V[1, w] ← 0
    else
        V[1, w] ← b1

for i ← 2,...,n do
    for w ← 0,...,W do
        if (w < wi)
            V[i,w] ← V[i-1,w]
        else
            valWithith ← bi + V[i-1, w-wi]
            valWithoutlth ← V[i-1,w]
            V[i,w] ← max(valWithith, valWithoutlth)

return V[n,W]
```

סיבוכיות

סיבוכיות זמן: $\Theta(nW)$

סיבוכיות מקום: $\Theta(nW)$

איך ניתן לחסוך במקומות?

הערה: היעילות תלויה גם ב- W ולא רק ב- n , ולכן אינה פולינומית בגודל הקלט,

כי דרישים $W \log n$ ביטים לשמר את W .

זו בעיה NP-שלמה ב- n , לא ידוע אלגוריתם שרצה בזמן פולינומי בגודל הקלט.

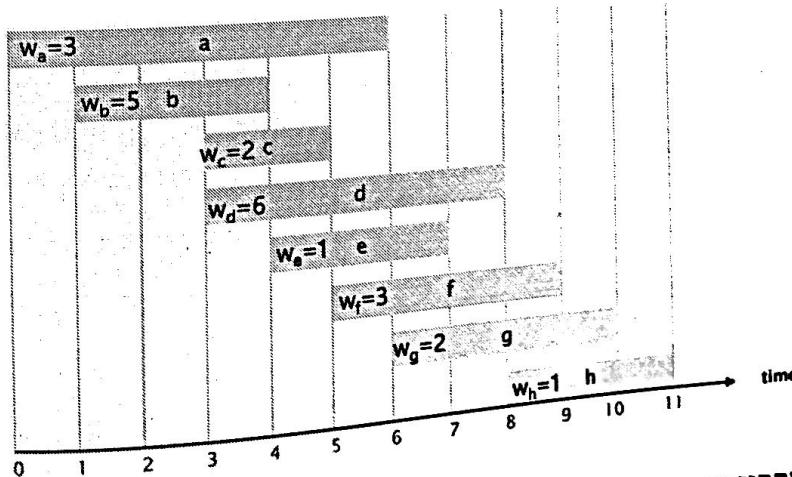
זמן קטעים ממושקלים (Weighted interval scheduling)

קלט: קבוצה I של n קטעים, כשל קטע מוצג ע"י זוג סדור (s_i, t_i) .
שמייצג את נקודת התחלה והסיום שלו, ולכל קטע משקל w_i .

פלט: תת-קובוצה של קטעים מתוך I שורם זה לזה ומשקלם הכולל מקסימלי.
ערך הפתרון = המשקל הכולל של פתרון מקסימלי.

דוגמה: איך לבחור קורסים לא חופפים בשעות כך שנלמד מקסימום נ"ז בסמסטר?
המשקל הוא נ"ז של כל קורס.

דוגמה: זמן קטעים ממושקלים



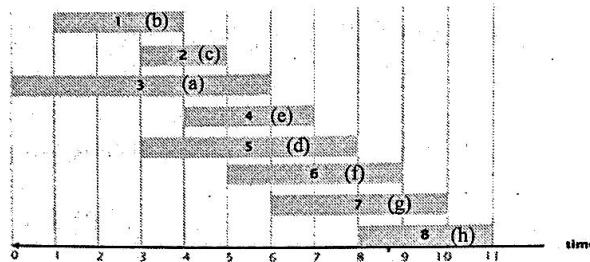
פתרון אופטימלי: {b, f}

ערך הפתרון האופטימלי: 8

$$w_b + w_f = 5 + 3 = 8$$

תזמון קטעים ממושקלים: המשך

נמיין את הקטעים לפי זמני הסיום שלהם (t_i), ומספר אותם בהתאם.



נדיר (i) $p =$ האינדקס של הקטע האחרון לפני הקטע ה- i שאינו חופף לו
(אם אין קטע כזה אז $p(i) = 0$).

כלומר: $j = p(i)$ אם j הוא האינדקס המקסימלי הקטן מ- i , כך ש- $s_j \leq t_i$.

דוגמאות: $p(4) = 1, p(6) = 2, p(7) = 3, p(2) = 0$

הרעיון הרקורסיבי

נדיר (i) $opt =$ ערך הפתרון האופטימי עבור קבוצת הקטעים $\{i, \dots, 1\}$.

נחשף את $.opt(n)$

תהי $\{i, \dots, 1\} = I$ קבוצה של קטעים ממושקלים (מסודרים לפי זמני הסיום)
ותהי $I \subseteq S$ תת-קבוצה המהווה פתרון אופטימי עבור I .

יש שתי אפשרויות: $S \in i$ או $S \notin i$.

הרענון הרקורסיבי

$S \in i$: הפתרון האופטימלי כולל את הקטע ה- i וחת-קבוצה אופטימלית של הקטעים $.opt(p(i)) + w_i + \{1, \dots, p(i)\}$

$S \notin i$: הפתרון האופטימלי לא כולל את הקטע ה- i ולכון ערך הפתרון האופטימלי $.opt(i - 1)$.

$$opt(i) = \max\{opt(p(i)) + w_i, opt(i - 1)\}$$



$$opt(0) = 0$$

תנאי עצירה:

האלגוריתם רקורסיבי

אלגוריתם (טיאור עלי):

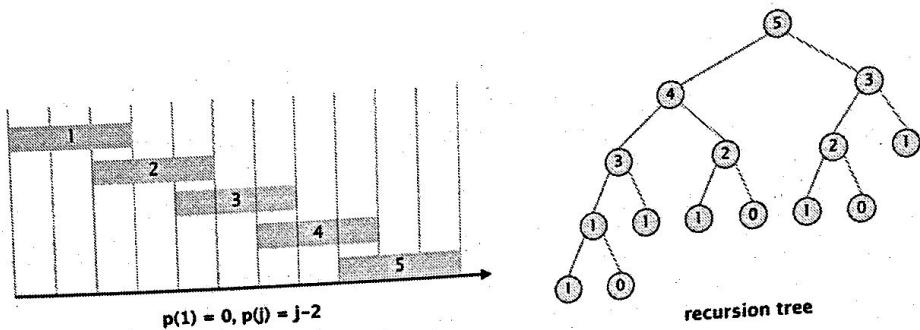
1. נמיין את הקטעים לפי זמני הסיום שלהם (t_i).
- 2.-Calculates $p(1), \dots, p(n)$.
3. Calculates $opt(n)$ using opt as a "call" to opt .

COMPUTE-OPT(n)

```
COMPUTE-OPT(i)
if  $i = 0$  then
    return 0;
else
     $c_1 \leftarrow \text{COMPUTE-OPT}(p[i]) + w[i];$ 
     $c_2 \leftarrow \text{COMPUTE-OPT}(i - 1)$ 
    return  $\max(c_1, c_2);$ 
```

האלגוריתם רקורסיבי לא יעיל

בעיה:יעילות האלגוריתם הרקורסיבי היא מעריכית בשל חישוב חזר של תת-בעיות.



פתרון: תכנות דינמי (bottom-up).

אלגוריתם תכנות דינמי

אלגוריתם (תיאור עליי):

1. נמיין את הקטעים לפי זמני הסיום שלהם (t_i).
2. נחשב את ($p(1), \dots, p(n)$).
3. נחשב את $opt(n)$ ע"י קרייה ל-

COMPUTE-OPT(n)

```

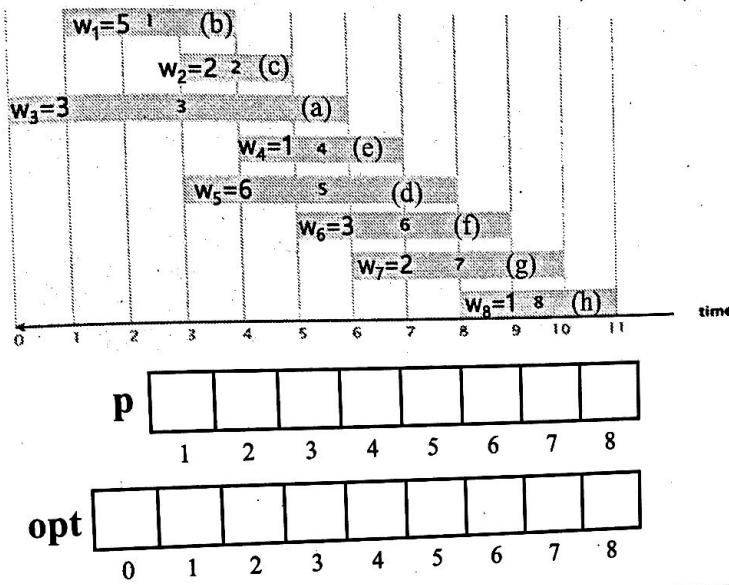
COMPUTE-OPT(n, p[1...n], w[1...n])
  opt[0] ← 0;
  for i ← 1,...,n do
    opt[i] ← max(opt[p[i]] + w[i], opt[i - 1])
  return opt[n]
  
```

יעילות כוללת:
 $\Theta(n \log n)$
 זיכרון:
 $\Theta(n)$

יעילות :
 $\Theta(n)$:**COMPUTE-OPT**

אלגוריתם תכנות דינמי – דוגמת הרצאה

הקטעים לאחר המילוי לפי זמני הסיום:



תזמון קטעים ממושקלים מציאת תת-קובוצה האופטימלית

נשתמש במערך שיצרנו לצורך חישוב ערכי opt .

• אתחול $\emptyset, S = \emptyset, i = n, opt[i] = 0$.

• כל עוד $i > 0$:

$$opt[i] = opt[i - 1] -$$

 ↳ נעדכן: $i \leftarrow i - 1$

- אחריה ($opt(i) = opt[p[i]] + w[i]$)

 ↳ נעדכן: $S \leftarrow S \cup \{i\}, i \leftarrow p[i]$

↳ הקטע ה- i -י לא נבחר

↳ הקטע ה- i -י נבחר

יעילות (ללא חישוב ערכי opt): $\Theta(n)$

מרחוק עריכה - Edit Distance

קלט: שתי מחרוזות של תווים (y_1, \dots, y_m) , (x_1, \dots, x_n) .
פלט: מרחק העריכה בין X ל- Y .

פעולות עריכה על מחרוזת X היא החלפה / מחיקה / הוספה של תו ב- X .

דוגמה: $X = (T, G, C, A, C, C, A)$

$X' = (T, G, G, A, C, C, A)$ החלפת תו:

$X' = (T, G, A, C, C, A)$ מחיקת-תו:

$X' = (T, G, C, G, A, C, C, A)$ הוספת תו:

מරחיקת העריכה

מරחיקת העריכה של מחרוזות (y_1, \dots, y_m) , (x_1, \dots, x_n) .
 הוא מספר פעולות העריכה המינימלי הנדרש לעבור מ- X ל- Y .

דוגמה: $X = (T, G, C, A, C, C, A)$, $Y = (T, C, C, C, G, A)$

מראק העריכה ≥ 3 \leftarrow $X = X_0 = (T, G, C, A, C, C, A)$,
 $X_1 = (T, C, C, A, C, C, A)$,
 $X_2 = (T, C, C, C, C, A)$,
 $X_3 = (T, C, C, C, G, A) = Y$

שימושים: תיקון שגיאות, השוואת בין מחרוזות DNA.

הרענון הרקורסיבי

סימוניים: $X_{[1..0]} = X$, והמחוזות הראיקה היא $X_{[1..i]} = (x_1, \dots, x_i)$

נסתכל על סדרת פעולות מינימלית להמרת X ל- Y :

1. אם מוחקם את התו האחרון של X : צריך עדין להמיר את $X_{[1..n-1]}$ ל- Y .

2. אם מוסיפים את התו האחרון של Y ל- X : צריך להמיר את $X_{[1..n]}$ ל-

3. אם לא מוחקם את התו האחרון של X ולא מוסיפים את התו האחרון ב- Y :
צריך להמיר את $X_{[1..n-1]}$ ל- $Y_{[1..m-1]}$ ולהחליףתו אם $x_n \neq y_m$.

נוסחת הנסיגת הרקורסיבית

יהי $c(i,j)$ מרחק העריכה בין $X_{[1..i]}$ ל- $Y_{[1..j]}$. נחשב את $c(n,m)$.

$$c(i,j) = \min \begin{cases} c(i-1,j) + 1 \\ c(i,j-1) + 1 \\ c(i-1,j-1) + \delta(x_i, y_j) \end{cases}$$



$$\delta(x_i, y_j) = \begin{cases} 0 & x_i = y_j \\ 1 & x_i \neq y_j \end{cases}$$

כאשר:

תנאי עצירה:

- אם $i = 0$ או $j = 0$: $c(i,j) = 0$ נדרשות j פעולות להמיר את המחרוזות הראיקה $X_{[1..0]}$ ל- $Y_{[1..j]}$.
- ואם $i = j = 0$: $c(i,j) = 0$

בעיה Edit Distance

אלגוריתם תכונות דינامي

- נגידר מערך c בגודל $(n + 1) \times (m + 1)$.
- נמלא את המערך החל מהשורה הראשונה בעזרת נוסחת הנסיגה:

$$c(i, j) = \min \begin{cases} c(i - 1, j) + 1 \\ c(i, j - 1) + 1 \\ c(i - 1, j - 1) + \delta(x_i, y_j) \end{cases}$$

- נחזיר את $c[n, m]$.

סיבוכיות זמן: $\Theta(n \cdot m)$

סיבוכיות זיכרון: $(n \cdot m) \Theta(m)$ (ניתן ליעיל ל- $\Theta(n)$)

$$c(i, j) = \min \begin{cases} c(i - 1, j) + 1 \\ c(i, j - 1) + 1 \\ c(i - 1, j - 1) + \delta(x_i, y_j) \end{cases}$$

		B	D	C	A	
		0	1	2	3	4
X ↓	0	0	1	2	3	4
	1	1				
A	2	2				
A	3	3				
D	4	4				

דוגמה:

$$Y = (B, D, C, A), \\ X = (D, A, A, D)$$

$$Y = (B, D, C, A), X = (D, A, A, D) \quad \underline{\text{דינמי}}$$

		B	D	C	A	
		Y →				
X ↓	c[i,j]	0	1	2	3	4
	0	0	1	2	3	4
D	1	1	1	1	2	3
A	2	2	2	2	2	2
A	3	3	3	3	3	2
D	4	4	3	3	4	3

EditDistance (X,Y)

n ← length[X]

m ← length[Y]

for i ← 1 to n do

 c[i,0] ← i

for j ← 1 to m do

 c[0,j] ← j

for i ← 1 to n do

 for j ← 1 to m do
 $\min \{c[i-1,j-1]+\delta(x_i, y_j), c[i,j-1]+1, c[i-1,j]+1\}$

return c[n,m]

תרגיל

חישוב האיבר ה- n -י של סדרת פיבונacci:

$$a_0 = a_1 = 1, a_n = a_{n-1} + a_{n-2}$$

0 1 2 3 4 5 6 ...

1	1	2	3	5	8	13	...
---	---	---	---	---	---	----	-----

• מהי הנוסחה הרקורסיבית?

• כיצד ניתן באמצעות תכנות דינמי?

תרגילים ותרגילים

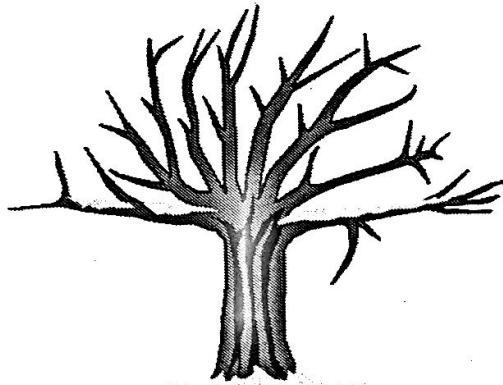
1. מילוי טבלאות סדרת פיבונacci. סדרת פיבונacci היא סדרה של מספרים טבעיים, בה כל איבר הוא סכום האיברים הקודמים. סדרת פיבונacci מוגדרת כך: $a_0 = 0, a_1 = 1$, ו- $a_n = a_{n-1} + a_{n-2}$ עבור $n \geq 2$.

2. מילוי טבלאות סדרת פיבונacci. סדרת פיבונacci היא סדרה של מספרים טבעיים, בה כל איבר הוא סכום האיברים הקודמים. סדרת פיבונacci מוגדרת כך: $a_0 = 0, a_1 = 1$, ו- $a_n = a_{n-1} + a_{n-2}$ עבור $n \geq 2$.



3. מילוי טבלאות סדרת פיבונacci. סדרת פיבונacci היא סדרה של מספרים טבעיים, בה כל איבר הוא סכום האיברים הקודמים. סדרת פיבונacci מוגדרת כך: $a_0 = 0, a_1 = 1$, ו- $a_n = a_{n-1} + a_{n-2}$ עבור $n \geq 2$.

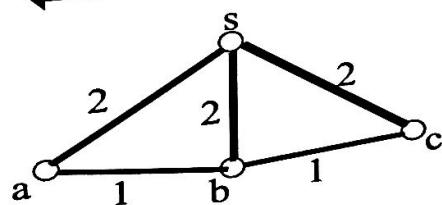
עץ פורש מינימלי



מוטיבציה: רשת תקשורת

סצנrio 1: s הוא שרת מידע. לבקשתו של כל לקוח (c, b, a) עליו לשולח אליו הודעה בדרך הזרלה ביותר. עلينו לחת בידיו "מפת דרכי".

עץ המק"בים



סצנrio 2: מפעילי הרשת צריכים לשוכר מראש מראש את הקווים שישתמשו בהם. רוצים למזער את הוצאות.

עץ פורש מינימלי

עצים פורשים

הגדרה: עץ פורש של גרף לא-מכוון וקשרי G הוא תת-graf קשור אציקלי T שכולל את כל קדקודי הגרף.

משפט: יהיו T תת-graf של G שכולל את כל קדקודיו. כל הטענות הבאות שකולות:

1. T הוא עץ פורש.
2. T קשר, וכל צלע בו היא גשר (כלומר הסרתה פוגעת בקשרות של T).
3. T חסר מעגלים, אך ככל צלע נוסף לו תסגור מעגל.
4. T קשר ומכל בדיקת-a צלעות (a – מספר הקדקודים ב-G).
5. T חסר מעגלים ומכל בדיקת-a צלעות.

מסקנה (חשובה): יהיו T עץ פורש. אם מוסיפים קשת ל-T ומסירים את אחת הקשתות של המעגל שנוצר, מקבל שוב עץ פורש.

עצים פורשים מינימליים

הגדרה: יהיו G גרף לא-מכוון עם משקלים לצלעות, יהיו T עץ פורש של G.

$$\text{משקלו של } T \text{ הוא: } w(T) = \sum_{(u,v) \in T} w(u,v)$$

הגדרה: T הוא עץ פורש מינימי (UPM / MST) אם משקלו מינימי מבין המשקלים של כל העצים הפורשים של G.

בעיית העץ הפורש המינימי:

קלט: גרף קשר לא-מכוון G, עם משקלים.

פלט: עץ פורש מינימי של G.

אבחןות:

- יתוכנו כמה עצים פורשים מינימליים לאותו גרף.
- מכיוון שהקדקודים קבועים, ניתן לייצג את T כאוסף צלעות.

אלגוריתם המדיון

אלגוריתם המדיון בסיסי:

- נחזיק קבוצת צלעות שתגדל בהדרגה.
 - כל עוד אין בידינו עץ פורש:
- נסיף צלע שאינה סוגרת מעגל עם הצלעות הקודמות.
בחירת הצלע תעשה בצורה חמדנית.

האלגוריתם של ק魯וסקל : Kruskal
במהלך האלגוריתם T עשוי להיות עיר. בוחרים צלע מינימלית מכל הצלעות שטרם ניסינו.

האלגוריתם של פרימ Prim
ובונים עץ אחד ובכל שלב מוסיפים לו צלע מינימלית.

קבוצה מבטיחה

הגדרה: יהיו G גרף לא-מכוון קשור עם משקלים. קבוצה $E \subseteq F$ נקראת מבטיחה אם ניתן להשלים אותה לעץ פורש מינימלי.

אבחן 1: הקבוצה משרה תת-גרף $(V, F) = H$, אשר אינו בהכרח קשור.

אבחן 2: הקבוצה הריקה \emptyset היא קבוצה מבטיחה בכל גרף.

שיטת למציאת עפ"מ:

$F \leftarrow \emptyset$

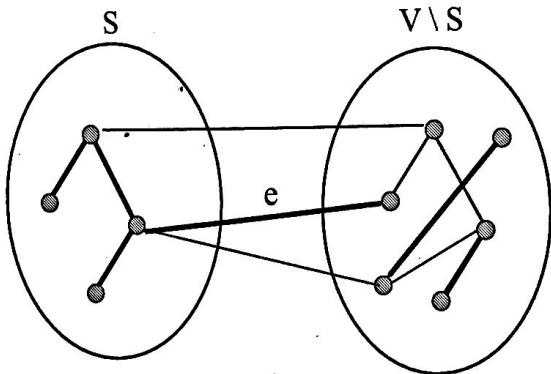
while (V, F) is not a spanning tree
choose "good edge" $e \notin F$ and let $F \leftarrow F \cup \{e\}$

צלע טובה = צלע שהוספה יוצרת קבוצה מבטיחה.

איך נדע איזו צלע לבחור?

למה הבחירה החמדנית

למה: יהיו G גרף לא-מכוון קשור עם משקלים,andi $S \subseteq F$ קבוצה מבטיחה, ותהי קבוצת קזקודות המהווה רכיב קשרות ב- $H = (V, F)$. תהי e צלע מינימלית במשקלה בין S ל- $V \setminus S$. אז גם $\{e\} \cup F$ קבוצה מבטיחה.

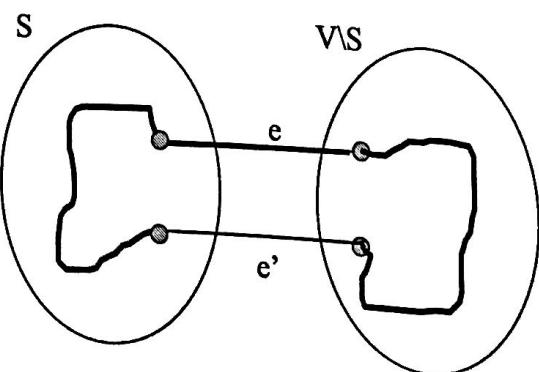


הוכחה: יהיו T עץ פורש מינימלי כך ש- $F \subseteq T$
אם $e \in T$ אז סימנו, שכן $T \subseteq \{e\} \cup F$. אחרת, $\{e\} \cup T$ מכיל מעגל.
↳ יש צלע אחרת $T' \in T$ בין S ל- $V \setminus S$ שסוגרת את המעגל.

נתבונן בו: $\{e\} \setminus \{e'\} \cup T' = T$ הוא עץ פורש (לאו דווקא מינימלי) ⇐

$$w(T') = w(T) + w(e) - w(e') \leq w(T)$$

מתקיים:



e מינימלית בין S ל- $V \setminus S$

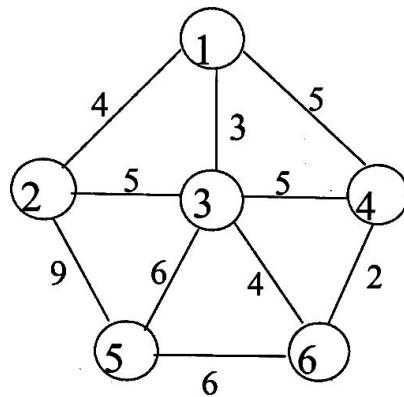
מהמינימליות של T , מתקיים
 $w(T) \leq w(T')$,
ולכן $w(T) = w(T')$ כולם T הוא עץ פורש מינימלי.

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

אלגוריתם קרווסקל

האלגוריתם של קרווסקל:

- מתחילה עם עיר המורכב מ- τ קודקודים הגרף (ללא קשתות)
- עוברים על קשתות הגרף בסדר עולה של משקל
- מוסיפים קשת לעיר אם היא אינה סוגרת מעגל



משפט: בהינתן גראף קשיר G , האלגוריתם של קרווסקל מוצא עץ פורש מינימלי.

הוכחה: נוכיח באינדוקציה על מספר הצלעות ב- F , שבו כל שלב F מבטיחה. מטענה זו נובעת נכונות האלגוריתם (תרגיל!).

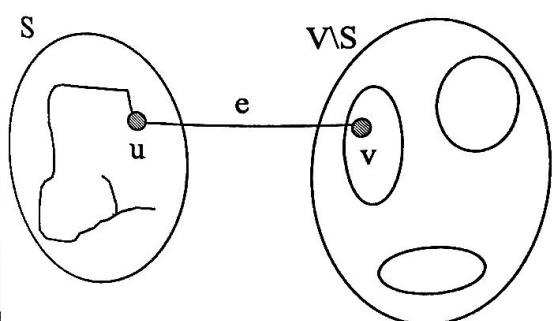
בסיס: $F = \emptyset$ ברור שזו קבוצה מבטיחה.

צעד: נניח ש- F מבטיחה עד כה ותהי $(v, u) = e$ הקשת שהאלגוריתם בחר להוסיף. ככלומר, u, v שייכים לרכיבי קשריות שונים של העיר.

יהי S הרכיב שמכיל את u .

e צלע מינימלית שיוצאת מ- S , כי אם הייתה צלע יותר קלה כזו, האלגוריתם היה נתקל בה קודם ובודח בה.

לכן, לפי למת הבחירה החמדנית, גם $\{e\} \cup F$ מבטיחה.



ייצוג רכיבי הקשרות

הבעיה: לתחזק רכיבי קשרות של גרף שבתחלתו הוא ריק מצלעות, ומעודכן מדי פעם ע"י הכנסת צלע נוספת (שאינה סוגרת מעגל).

בכל שלב, יש לתמוך גם בפעולת בדיקה – האם שני קדוקדים שייכים באותו רכיב.

מבנה הנתונים המופשט המתאים: מבנה Union-Find

תזכורת:

המבנה מייצג אוסף של קבוצות זרות מתוך $\{1, 2, \dots, n\}$. לכל קבוצה יש נציג שנבחר בצורה חופשית ע"י האלגוריתם.

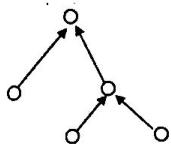
הפעולות הבסיסיות:

MakeSet(x), Find(x), Union(s₁, s₂)

KRUSKAL(WeightedGraph G)

```
EdgeSet F ← Ø // Empty forest
DisjointSets UF;
List L ← SORT(E) // Sort edges by weight
for each v ∈ V do
    S.MakeSet(v) // Enter v into Union-Find structure
for each (u,v) ∈ L do // in ascending order
    u' ← UF.Find(u)
    v' ← UF.Find(v)
    if u' ≠ v' then
        F ← F ∪ { (u,v) }
        UF.Union(u',v')
return F
```

Union-Find ייעיל של מימוש



- מבנה הנחות:
- כל קבוצה תיווצר על ידי עץ עם מצביעים להורים.
 - נציג הקבוצה הוא שורש העץ.

- шиפוריים:
- מאחדים קבוצה קטנה לתוך גדולה Union by Rank
 - ציוץ מסלולים בפועל Find

יעילות הפעולות:

$\Theta(1)$	MakeSet •
$\Theta(1)$	Union •
$O(\log n)$	Find •

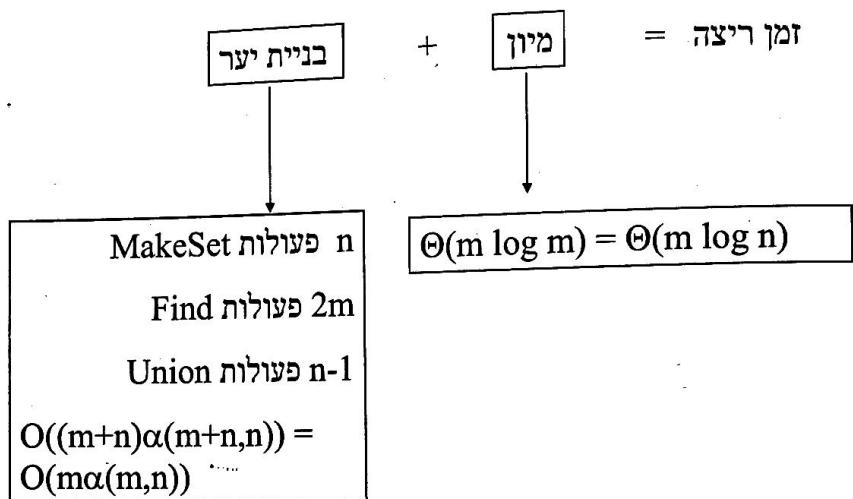
משפט (Tarjan): היעילות הכלולות של k פעולות MakeSet, Find, Union שמתוכן יש α פעולות MakeSet היא $\Theta(k\alpha(k,n))$, כאשר $\alpha(k,n)$ היא הפונקציה ההפוכה לפונקציית אקרמן.

הפונקציה $\alpha(k,n)$

n	$\alpha(2n,n)$	$\alpha(3n,n)$
2	1	1
4	1	1
16	2	1
64	2	2
256	2	2
1024	2	2
4096	2	2
16384	2	2
65536	3	2
2^{32}	3	2

הפונקציה עולה לאט מאוד!
 $\Theta(k\alpha(k,n)) \leftarrow \Theta(n, \Theta(k, \Theta(k)))$. אין אינטואיטיביון לכך ההבדל זעום.
http://en.wikipedia.org/wiki/Ackermann_function

ניתוח היעילות של אלגוריתם קראוסקל



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

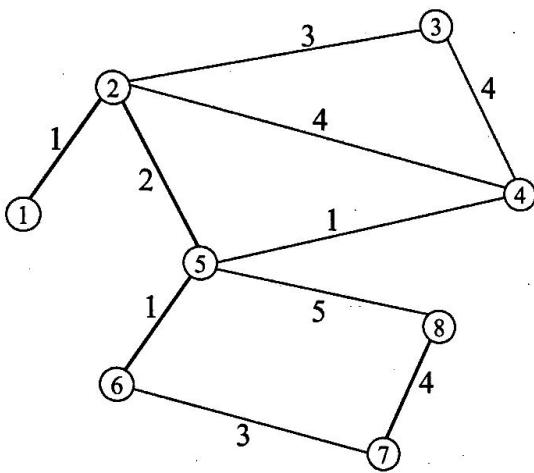
האלגוריתם של פרים

רעיון: בונים עץ אחד T. מתחילה עם קדקוד שריורתי. בכל שלב מוסיפים קדקוד אשר עלות חיבורו לעץ הקיים היא מינימלית.

הוכחת נכונות: עפ"י למת הבחירה החמדנית (+ אינדוקציה).

יישום: לכל קדקוד שאינו בעץ, נשמר את הצלע הקלה ביותר שמחברת אותו לעץ הנוכחי. כמו כן, נזקק תור קדיימות שמאפשר לשלווף את המינימלית בינהן.

דוגמת הרצאה



PRIM(WeightedGraph G)

PriorityQueue Q

Boolean InT[n] = { false ... }

Weight min[n]

```

min[v0] ← 0                                //arbitrary initial vertex
p[v0] ← nil
for each vertex v ≠ v0 do
    min[v] ← ∞
    p[v] ← nil
Q.Build(V,min)                                // Build Priority Queue
while Q ≠ ∅ do                                  // Grow Tree
    u ← Q.DeleteMin()
    InT[u] ← true
    for each v ∈ Adj[u] do
        if (not InT[v]) and w(u,v) < min[v] then
            min[v] ← w(u,v)
            p[v] ← u
            Q.DecreaseKey(v,min[v])
return p // represents the tree
    
```

:min[v]

משקל צלע מינימלית
שמחברת את v לעצם
(מעודכן עבור צמתים
שאינם בעצם).

:p[v]

הקודוד בעצם
שאליו v יתחבר
(או כבר התחבר).

יעילות: $\Theta(m+n\log(n))$
(בניה שהטור ממומש
ע"י עירימת פיבונצ'י)

קרוסקל ופרים



: (1928-2010) Joseph Kruskal
מתמטיקאי ומדען מודיעי המחשב אמריקני.
חוקר בתחום האלגוריתמים והקומבינטוריקה.



: (נולד ב- 1921) Robert Prim
מתמטיקאי ומדען מודיעי המחשב אמריקני.
פרים גילה את האלגוריתם לעץ פורש מינימלי בשנת 1957,
אבל למשה מתמטיקאי צ'כי Vojtěch Jarník
חשב על האלגוריתם כבר ב- 1930.



תכונות של עצים פורשים מינימליים

למאת הפונקציה המונוטונית
יהי G גרף לא מכוון, תהי w פונקציית משקל על הצלעות, ותהי f פונקציה עולה ממש.
נגידר פונקציית משקל חדשה w' באופן הבא:
$$(w'(e)) = (w(e))$$

או עץ פורש של G הינו מינימלי ביחס ל- w' אם והוא מינימלי ביחס ל- w .

- דוגמאות:
- נוסף קבוע לכל הצלעות.
 - כפיף בקבוע חיובי.

הערה: תכונה זו אינה מתקינה בבעיית המק"בים...

כל שני עצים מינימליים זהים במשקלים

משפט (Cayley): מספר העצים הפורשים בגרף שלם בן n קדקודים הוא 2^{n-2} .

משפט:

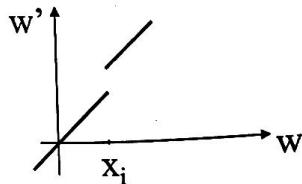
- יהיו T_1 ו- T_2 עצים פורשים מינימליים של גרף G ויהיו:
- המשקלים של צלעות, T_1 , $x_1 \leq x_2 \leq \dots \leq x_{n-1}$
 - המשקלים של צלעות, T_2 , $y_1 \leq y_2 \leq \dots \leq y_{n-1}$
 - או $x_i = y_i$ לכל i . כלומר, שתי הסדרות זהות.

מסקנה: אם כל המשקלים ב- G שונים זה מזה אז יש עץ פורש מינימי יחיד.

הוכחת המשפט: נניח בשיילה שהסדרות שונות.

יהי i האינדקס הראשון כך ש- $x_i < y_i$. נניח בה"כ כי $x_i < y_i$.

נפעיל על המשקלים את הפונקציה הבאה:



$$f(w) = \begin{cases} w, & w \leq x_i \\ w+1, & w > x_i \end{cases}$$

f פונקציה מונוטונית עולה ממש.

לפי למת הפונקציה המונוטונית, T_1, T_2 מינימליים גם ביחס ל- w' ומשקלם החדש הוא:

$$w'(T_2) = y_1 + y_2 + \dots + y_{i-1} + (y_i + 1) + (y_{i+1} + 1) + \dots + (y_{n-1} + 1)$$

$$= w(T_2) + (n - i)$$

$$w'(T_1) \leq x_1 + x_2 + \dots + x_{i-1} + x_i + (x_{i+1} + 1) + (x_{i+2} + 1) + \dots + (x_{n-1} + 1)$$

$$= w(T_1) + (n - i - 1)$$

$$w'(T_1) < w'(T_2) \iff w(T_1) = w(T_2) \iff T_2 \text{ אינו מינימי ביחס ל- } w. \text{ סתירה.}$$

תרגיל: העדפת צלעות

נתון גרפּ לא מכון עם משקלים שבו לכל צלע יש גם צבע: כחול או אדום. מצאו אלגוריתם יעיל שモצא עץ פורש מינימלי שבו יש מקסימום צלעות כחולות.

תרגיל: האם זה עובד? (א)

האם האלגוריתם הבא מוצא עץ פורש מינימלי בגרף?

אתחול: יעד של ח' עצים כשל עץ מכל לבדוק קזקודה אחד.

כל עוד יש ביעד יותר מעץ אחד:

- נבחר עץ כלשהו; T_i ביער (למשל, העץ; T_i עם האינדקס i הקטן ביותר).
- נמצא צלע e מינימלית במשקל שיצאת מהעץ; T_i לעץ אחר ביער.
- נוסיף את e ליער, ונאחד את העצים שהוא מחברת לעץ אחד.

תרגיל: האם זה עובד? (ב)

האם האלגוריתם הבא מוצא עץ פורש מינימלי בגרף?

אתחול: יעד של מעצים כשל עץ מכיל בדיקת קדוק אחד.

כל עוד יש ביער יותר מעץ אחד:

- נבחר קדוקו כלשהו ואשר יוצאות ממנו צלעות לעץ אחד או יותר מלבד העץ שהוא שייך אליו.
- תהי e צלע מינימלית במשקל מבין הצלעות הנ"ל.
- נוסיף את e ליער, ונאחד את העצים שהוא מחברת לעץ אחד.

שאלות מבחנים

הוכחו או הפריכו:

יהי G גרף לא מכון פשוט עם משקלים על הקשתות.

תהי f הקשת הכבידה ביותר בגרף, כך שמשקלה גדול ממש ממשקל כל הקשתות האחרות או הקשת f לא תשתתף אפילו ufm של G .

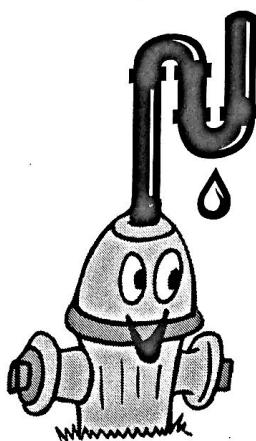
הוכחו או הפריכו:

יהי (V,E) G גרף לא מכון פשוט עם משקלים על הקשתות ותהי $(u,v) = e$ קשת בגרף. נסתכל על הגרף $(V,E') = G - \{(u,v)\}$

$$E' = \{(i,j) \in E \wedge w(i,j) < w(u,v)\}$$

הגרף G' מכיל את כל קדוקיו G ורק את הקשתות שמשקלן קטן ממש ממשקל e . אם u ו- v נמצאים ברכיבי קשרות שונים של G' או קיים ufm של G' שמכיל את e .

זרימה ברשות



תרגיל רכבת תחתית תל אביב-יפו

משימה 1: תכנון מסילות

נתונים:

- נקודות ציון.
- מסילות פוטנציאליות להיבור זוגות נק' ציון.
- עלות הקמה לכל מסילה פוטנציאלית.

מטרה:

- לבחור מסילות שישילו כדי לספק מסלול נתיחה בין כל שתי נק' ציון,
- ובסכום עליות הקמה מינימלי.

פתרון (השלימו!):

_____ • תרגום לבעה שלמדו בכתיבה:

_____ • אלגוי לפתרון הבעה:

_____ • ייעילות :

תרגיל רכבת תחתית תל אביב-יפו

משימה 2: בחירת נתיב נסעה מהיר

קלט:

- גרף המיצג את מפת הרכבת התתית:

- קדקוד = נק' ציון.

- קשת = מסילה פעילה.

- משקל קשת = זמן נסעה במסילה.

מטרה:

- בהתנחת נק' ציון א' ו-ב', החזר מסלול מ-א' ל-ב' עם זמן נסעה מינימלי.

פתרון (השלימו):

תרגם לבעה שלמדו בכיתה:

אלגו לפתרון הבעיה:

יעילות:

תרגיל רכבת תחתית תל אביב-יפו

משימה 3: ניתוב תנועת הנוסעים

קלט:

- גרף המיצג את מפת הרכבת + קיבולת של כל מסילה:

- קדקוד = נק' ציון

- קשת = מסילה פעילה

- קיבולת הקשת = מספר נוסעים שניין לשנע בקשה במשך שעה

- קדקוד מוצא + קדקוד יעד

מטרה:

- הגיחות לניתוב זרימת הקהיל:

מהו המספר הרצוי של אנשים שנעו בכל קשת
(במהלך שעה) כדי ש-

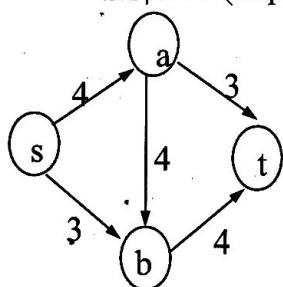
מקסימום אנשים יגיעו מהמוצא ליעד?

פתרון:

- בעית זרימה ברשות - הנושא הבא!

זרימה ברשתות

הגדרה: רשת זרימה \mathcal{N} היא גרף מכוון G (לאו דווקא DAG), עם קודקודים מיוחדים s, t הנקראים מקור (source) ובור (sink). לכל קשת (v, u) יש ערך $0 \leq C(v, u) \leq \infty$ שהוא הקיבול (Capacity) של הקשת.



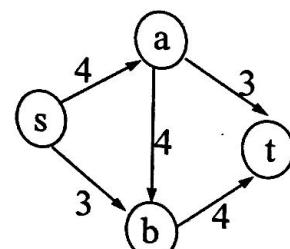
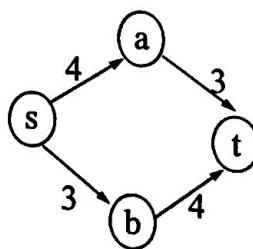
בעיית הזרימה המקסימלית: מה הקצב המקסימלי
שבו אפשר להזרים חומר ברשת מ- s אל t
ambilי לחזור מקיבולי הצלעות?

שימושים:

- רשת כבישים – מהו קצב ההתקדמות המקסימלי של מכוניות, והיכן ייווצרו פקקים.
- רשת מים – מהו הזרם המקסימלי שיכל לעבור ברשת, הקיבול הוא קוטר הצינור.
- רשת חשמל.
- העברת מידע ברשת תקשורת.

בעיית הזרימה המקסימלית

מהו הקצב המקסימלי בו אפשר להזרים חומר מ- s אל t , בכפוף לקיבולי הצלעות?



הגדרות פורמליות

$C(u,v) = 0$ או $(u,v) \notin E$, כאשר אם $C: V \times V \rightarrow [0, \infty]$, פונקציית קיבול היא פונקציה

פונקציית זרימה $f: V \times V \rightarrow R$ נקראת פונקציית זרימה אם מתקיימות שלוש הדרישות:

1. **אילוץ הקיבול**: $f(u,v) \leq C(u,v)$:Capacity Constraint (קשת רוויה)
אם $f(u,v) = C(u,v)$

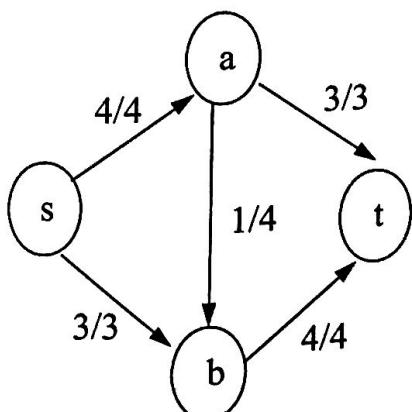
2. **אנטי-симטריה**: $f(u,v) = -f(v,u)$:Skew Symmetry

3. **שמור הזרימה**: $\sum_{v \in V} f(u,v) = 0$:Flow Conservation

ערך הזרימה $|f|$: הזרימה שיצאה מ- s והיא

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

דוגמה לפונקציית זרימה



$$f(a,s) = -4$$

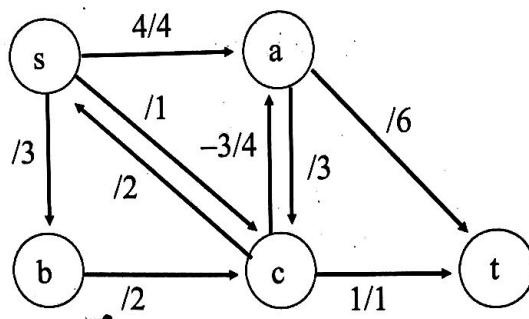
$$f(s,t) = f(t,s) = 0$$

כל מה שלא מצויר נקבע לפי אנט-סימטריה:

או לפי אילוץ הקיבול:

תרגיל

על גבי רשת הזרימה הבאה רשומים כל הקיבולים ורק חלק מערכי הזרימה.
השלימו את ערכי הזרימה.



תכונות של פונקציית זרימה

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

טענה:

הוכחה: נסתכל על כל זוגות הנקודות ונסכם אותם:

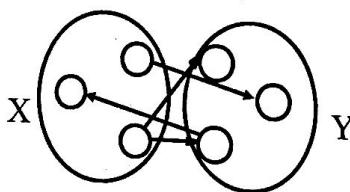
$$0 = \sum_u \sum_v f(u, v) = \sum_v f(s, v) + \sum_v f(t, v) = \sum_v f(s, v) - \sum_v f(v, t)$$

↑ ↑ ↑
 אנטיא-סימטריה שימור הזרימה אנטיא-סימטריה

הרחבת הגדרות של קיבול ושל זרימה

נרחיב את הגדרת פונקציית הזרימה ופונקציית הקיבול, כך שניתן יהיה להפעיל אותו על קבוצות.

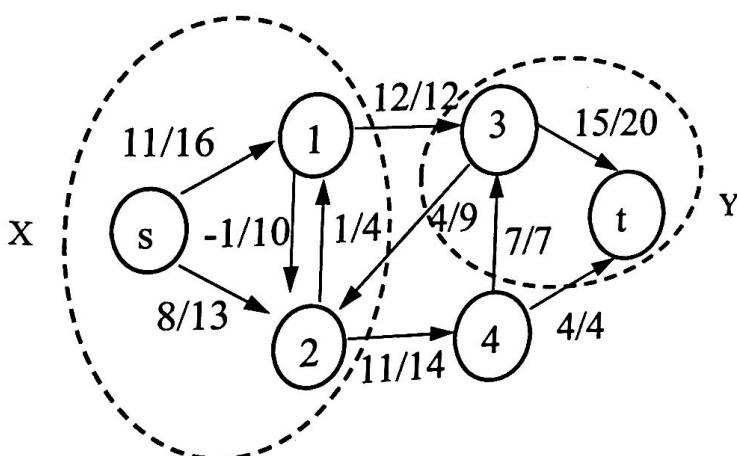
תהיינה $X, Y \subseteq V$ קבוצות. פונקציית הזרימה ופונקציית הקיבול מ- X ל- Y תוגדרנה:



$$f(X, Y) = \sum_{u \in X} \left(\sum_{v \in Y} f(u, v) \right)$$

$$C(X, Y) = \sum_{u \in X} \left(\sum_{v \in Y} C(u, v) \right)$$

דוגמה



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

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

תכונות של פונקציית זרימה

משפט: תהי 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:

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

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

$$f(X, Y) = \sum_{u \in X} \left(\sum_{v \in Y} f(u, v) \right) \xrightarrow{\text{אילוץ הקיבול}} \leq \sum_{u \in X} \left(\sum_{v \in Y} c(u, v) \right) = c(X, Y) \quad \underline{\text{הוכחה 1}}$$

קומוטטיביות של החיבור

$$f(X, Y) = \sum_{u \in X} \left(\sum_{v \in Y} f(u, v) \right) = \sum_{v \in Y} \left(\sum_{u \in X} f(u, v) \right) = \quad \underline{\text{הוכחה 2}}$$

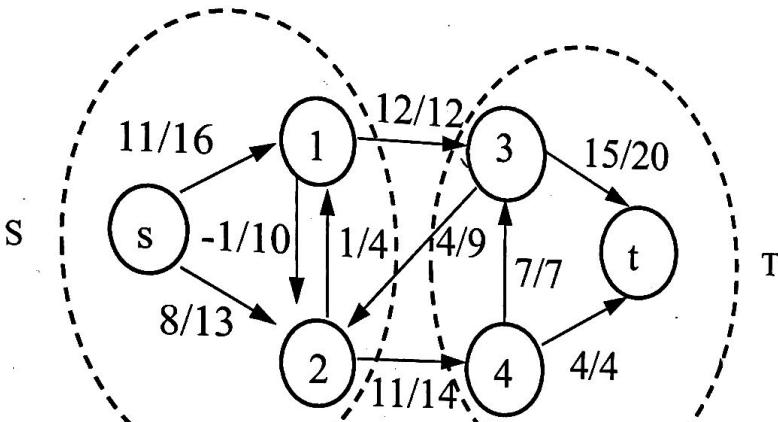
$$\xrightarrow{\text{אנטי-סימטריה}} = \sum_{v \in Y} \left(\sum_{u \in X} -f(v, u) \right) = -\sum_{v \in Y} \left(\sum_{u \in X} f(v, u) \right) = -f(Y, X)$$

הוכחה 3: עפ"י חלק 2 $f(X, X) = -f(X, X)$ ולבן $2f(X, X) = 0$, מכאן $-2f(X, X) = 0$

הוכחה 4: (תרגיל!)

מושג החתך

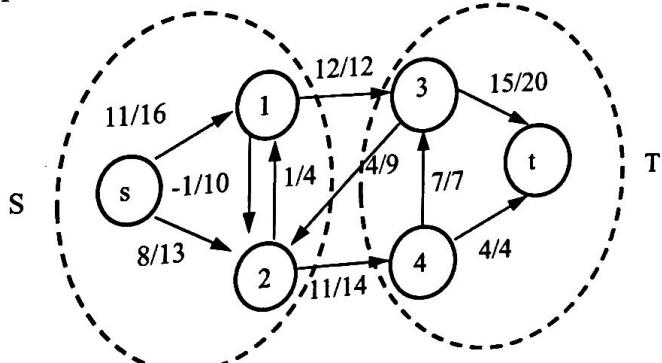
הגדרה: חוג (S, T) יקרא חתך ברשף אם הוא חלוקה של V לשתי קבוצות זרות,
 $t \in T$ כאשר $V = S \cup T$



מושג החתך

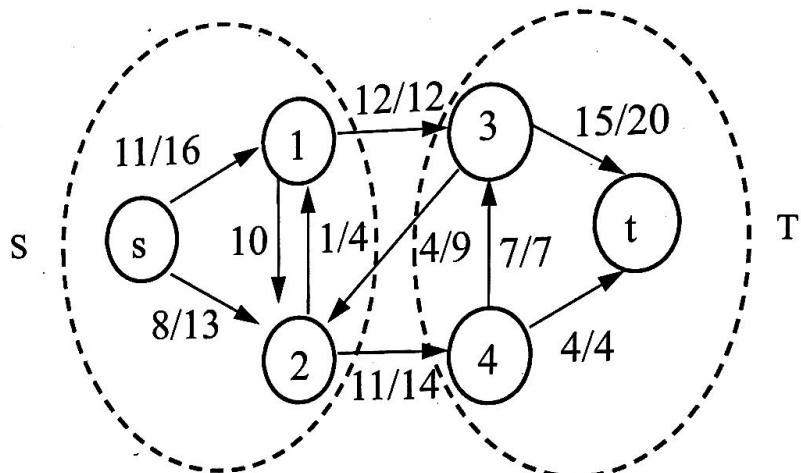
$$C(S, T) = \sum_{u \in S} \sum_{v \in T} C(u, v)$$

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



אבחן: לכל חתך (S, T) מתקיים $f(S, T) \leq C(S, T)$

דוגמה



$$C(S, T) = C(1, 3) + C(2, 4) = 12 + 14 = 26$$

$$f(S, T) = f(1, 3) + f(2, 4) + f(2, 3) = 12 + 11 + (-4) = 19$$

למה החתך

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

הוכחה:

נראה שני הטענות שווים ל-

$$1. \quad f(S, V) = f(S, T) + f(S, S) = f(S, T)$$

$$2. \quad f(S, V) = f(s, V) + f(S \setminus \{s\}, V) = |f| + \sum_{u \in S \setminus \{s\}} \left(\underbrace{\sum_{v \in V} f(u, v)}_0 \right) = |f|$$

מסקנה: $\text{MaxFlow} \leq \text{MinCut}$
 (כלומר הקיבול המינימלי בין כל קיבולי החתכים חוסם את הזרימה המקסימלית)

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

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

↑
מתקיים לכל חתך ↓
אבחנה

שיטת פורד-פלקרסון

רעיון: אלגוריתם שיפור הדרגי.

מושגי יסוד

קשת משפרת: קשת (v, u) שאפשר "להזירם" לאורכה עוד זרימה.
מסלול משפר: מסלול מ- s ל- t שמורכב כולו מקשתות משפרות.

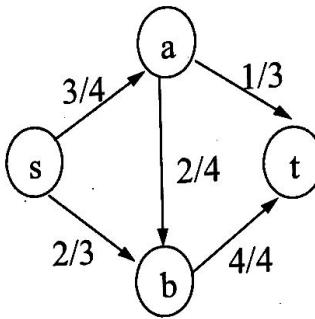
שאלות:

כיצד נמציא מסלול משפר?
 בכמה גודיל את הזרימה במסלול?

השיטה הבסיסית:

- אתחול: $f = 0$
- כל עוד יש מסלול משפר P :
 גודיל את הזרימה לאורך P

לוגמה



האלגוריתם בפירוט: הגדרות

קיבול שיורי C_f : כמות הזרימה שאפשר עדין להזרים לאורך קשת:

$$C_f(u,v) = C(u,v) - f(u,v)$$

אבחנות: תמיד $0 \geq C_f(u,v) > C(u,v)$. $C_f(u,v) = 0$.

הקיבול השינוי של מסלול P : זהו הקיבול השינוי המינימלי לאורך P ,

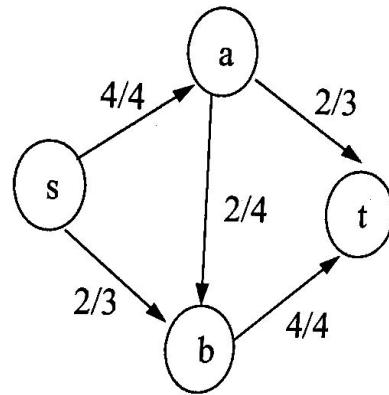
$$C_f(P) = \min_{(u,v) \in P} C_f(u,v)$$

מסלול משפר הוא מסלול שקיבולו השינוי חיובי.

גרף השינוי G_f : מכיל את כל הקשתות המשיפורות (קשנות שקיבולן השינוי חיובי).

כל מסלול מ- s ל- t ב- G_f הוא מסלול משפר.

לוגמה



זהו G . מהו הגרף השוויי כרגע?

האלגוריתם של פורד-פלקרסון:

- אתחול:

זרימה 0 בכל הקשתות: $f(u,v) = f(v,u) = 0$
הגרף השוויי G_f שווה לגרף המקורי G : $C_f(u,v) = C(u,v)$

- כל עוד יש מסלול משפר (מסלול מ- s ל- t בגרף G_f)
א. נמצא מסלול כזה P ונחשב את קיבולו השוויי
 $C_f(P)$
- ב. נגדיל את הזרימה ב- G לאורך P ב- $C_f(P)$ באופן הבא:
 לכל קשת (v,u) במסלול P נעדקן:

$$f(u,v) = f(u,v) + C_f(P)$$

$$f(v,u) = -f(u,v)$$

ג. נעדקן את הגרף השוויי לאורך המסלול P :

לכל קשת (v,u) במסלול P נעדקן:

$$C_f(u,v) = C(u,v) - f(u,v)$$

$$C_f(v,u) = C(v,u) - f(v,u)$$

- נחזיר את f

$c_f(P)$	הגרף G_f והמסלול המשפר P	הגרף G והזרימה הנוכחית קלט:
4		
7		

8		
4		
אין מסלול משפר		

ארבע קושיות

האם האלגוריתם תמיד מסתים?



אם כן, האם הוא מוצא זרימה מקסימלית?



כמה איטרציות ידרשו?



כיצד ובאיזה ייעילות נוכל לישם איטרציה?



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:

$$(1) \Rightarrow (2) \quad (\text{בעצם } (1) \rightarrow (2) \text{ נניח שב-} G_f \text{ יש מסלול משפר } P.$$

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

$(1) \Rightarrow (3)$ לכל חתך (S, T) מתקיים:

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

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

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).

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

$$T = V \setminus S \quad S = \{v \in V \mid G_f \text{ יש מסלול מ-} s \text{ ל-} v \text{ ב-} G_f\}$$

חתך כי $S \in \mathcal{S}$ וגם $S \notin \mathcal{T}$ (כי אין מסלול מ- s ל- t ב- G_f).

לכל $T \subsetneq S$ מתקיים $f(u, v) = C(u, v)$ לאחרת הקשת (u, v) שייכת ל- G_f וזו גם $v \in S$.

$$|f| = f(S, T) = C(S, T) \quad \leftarrow$$

פורד-פלקרסון – נכונות חלקית

משפט: אם אלגוריתם פורד-פלקרסון עוצר, אז הזרימה f היא מקסימלית

הוכחה:

אם האלגוריתם עוצר או אין בגרף השינוי G_f מסלולים משפרים. לפि משפט Max-Flow, Min-Cut מקסימלית.

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

אך האם פורד-פלקרסון תמיד עוצר? ובכמה איטרציות?

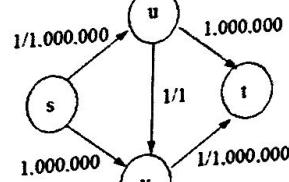
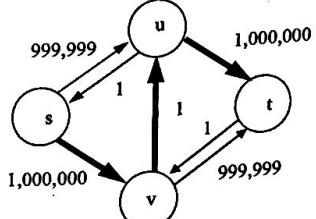
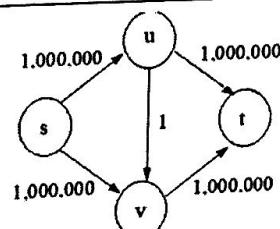
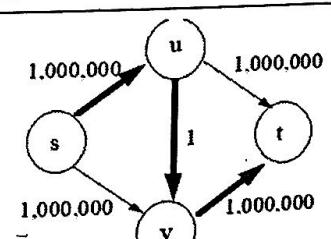
רשתות עם קיבולים שלמים

סימון: $MF = (\text{ערך הזרימה המקסימלית}).$

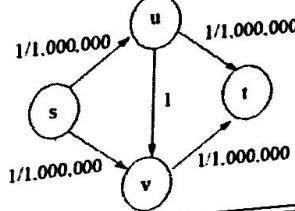
טענה: אם כל הקיבולים שלמים אז יתכנו לכל היותר MF איטרציות.

הוכחה: בכל איטרציה, ערך הזרימה גדול לפחות ב- 1.

טענה: החסם הדוק, כלומר יש רשת שבה דרישות MF איטרציות, ל- MF גדול מרצוינו.

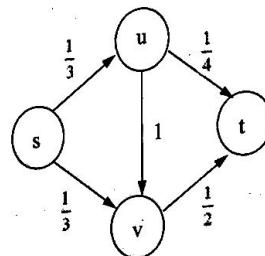


וכך הלאה...



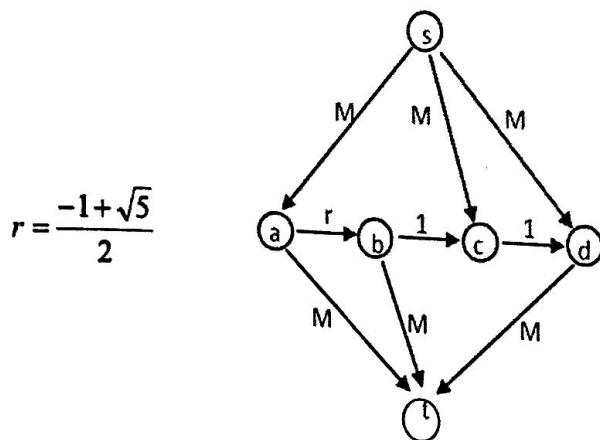
תרגיל: קיבולים רצינגולים

האם תוכלו למצוא תחליף לחסם MF על מסך האיטרציות, שיהיה נכון לרשות עם קיבולים רצינגולים?



קיבולים אי-רצינגולים

ברשת שיש בה קיבולים אי-רצינגולים, יתכן ששיטת פורד-פלקרטון לא הסתאים.



איך כдאי לבחור מסלול משפר?

משפט Edmonds, Karp: אם בוחרים מסלול משפר בעזרת BFS, מספר האיטרציות אינו עולה על nm .
הערה: טענה זו נכונה עבור גרפים עם קיבולים ממשיים כלשהם.

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

פתרונות ידועות נוספות

לגרפים עם קיבולים שלמים:

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

1. היה ש- $nC_{\max} \leq MF$, אזי מספר האיטרציות שיבצע האלגוריתם הוא לכל היותר nC_{\max} .
2. אם בוחרים תמיד מסלול משפר בעל קיבול שיורי מירבי, מספר האיטרציות אינו עולה על $(C_{\max})^{2m} \log(C_{\max})$ (מה ייעילות האלגוריתם?)

לגרפים עם קיבולים ממשיים כלשהם:

ראינו אלגוריתם שreq בזמן: $O(nm^2)$

קיים אלגוריתם נוספת, למשל דיניז $O(mn^2)$ גולדברג-טרזון $O(n^3)$ וועד.

Fulkerson and Tarjan



: (1924-1976) Delbert Ray Fulkerson
מתמטיקאי ומדען מודיעי המחשב אמריקני. עסוק בתחוםים של תוכנות לינארית ואופטימיזציה קומבינטורית.
פיתח יחד עם פורד את אלגוריתם פורד-פלקרסון בשנת 1956.



: (נולד ב-1948) Robert Tarjan
מתמטיקאי ומדען מודיעי המחשב אמריקני. המציא בין היתר את אלגוריתם החמשות (עם Blum, Floyd, Pratt, Rivest), ערימות פיבונצ'י (עם Fredman), והניתוח הייעיל של קבוצות זרות.
זכה בפרס טיוריינג ב-1986.

אדמונדס וקארפ



: (נולד ב-1934) Jack Edmonds
מדען מודיעי המחשב אמריקני. עסוק בתחוםים של אופטימיזציה קומבינטורית, זוגים בגרפים.
הגדיר לראשונה שבעיות ניתנות לפתרון יעיל על מחשב רק אם הן פולינומיות.



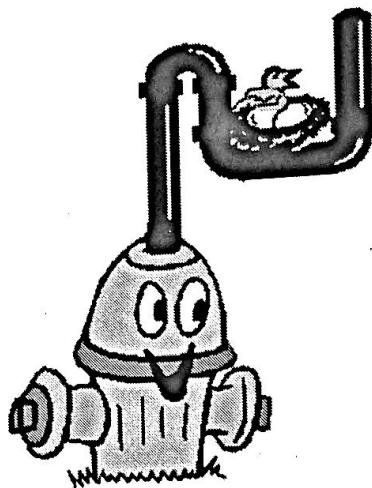
: (נולד ב-1935) Richard Karp
מדען מודיעי המחשב אמריקני. חוקר בתחוםי האלגוריתמים. הוכיח לראשונה שיקילות של בעיות NP-שלמות על ידי רדוקציה.
זכה בפרס טיוריינג בשנת 1985.

שאלות מבחנים

- נתונה רשות תקשורת ובה מחשבים שמחוברים ביניהם בקווים חד ציווניים.
לכל קו יש עלות כספית נתונה. מי שרכוש קו שולט על התעבורה בקו זה.
מחשב A מעוניין לשולח הודעה למחשב B.
חברה מסוימת מעוניינת למנוע מההודעה להגיע ליעדה, ולכן רוצה לרכוש קווים
כך שלא תאפשר העבורה בקווים שלה, ולכן A לא יוכל להעביר הודעה ל- B.
כתבו אלגוריתם המוצא אילו קווים כדי לחברה לרכוש כך שתשיג את מטרתה
בעלות מינימלית.

- נתונה רשות זרימה שבה יש קבוצת מקורות s_1, s_2, \dots, s_n וקבוצת בורות
 t_1, t_2, \dots, t_q .
מצאו זרימה חוקית שערכה מקסימלי כשערך הזרימה הוא סכום הזירימות מכל
המקורות.
הסבירו כיצד ניתן לפתור בעיה זו ע"י רדוקציה לביעית הזרימה המקסימלית.

שימושים נוספים ברשותות זרימה



זרימה בשלמים ורשותות 1-0

טענה: אם כל הקיבולים הם ערכים שלמים או אלגוריתם פורד-פלקרסון מוצא זרימה שערכיה שלמים.

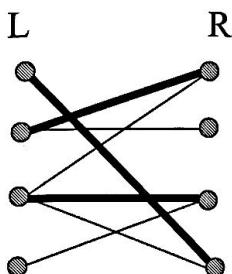
מסקנה: אם כל הקיבולים הם 0 או 1 או אלגוריתם פורד-פלקרסון מוצא זרימה שכל ערכיה הם 0 או 1.

תרגיל: כיצד כדאי לישם את פורד-פלקרסון ברשות 1-0, ומה תהיה הייעילות?

זיווג מרבי בגרף דו-צדדי

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

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



דוגמאות בגרפים דו-צדדיים:

- הרכבת זוגות לריקודים
- השנת עובדים למטלות

זיווג מושלם (Perfect matching) הוא זיווג שבו משתתפים כל הקדוקדים.

בזיווג מושלם בגרף דו-צדדי מתקיים $|R| = |L| = |M|$.

אלגוריתם למציאת זיווג מרבי בגרף דו-צדדי

הרענון: רזוקציה לביעית הזרימה ברשות 1-0.

הקלט: גרף דו-צדדי לא מכוון $G = (L \cup R, E)$

האלגוריתם:

1. בניית רשות זרימה N באופן הבא:

- נוסיף שני קדוקדים חדשים t, s .

- נוסיף קשתות מכוננות מ- s לכל הקדוקדים בקבוצה L .

- נוסיף קשתות מכוננות מכל קדוקד בקבוצה R ל- t .

- נכון כל צלע מ- L ל- R .

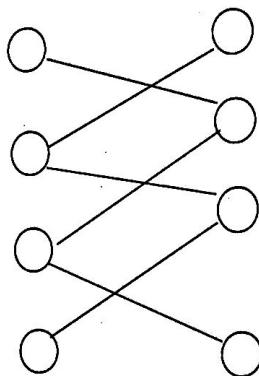
- הקיבולים של כל הקשתות יהיו 1.

2. נמצא זרימה מקסימלית f ברשות הגדולה בעזרת אלגוריתם פורד-פלקרסון.

3. הזיווג M יהיה: $M \in \{(u, v) \in L \times R \mid f(u, v) > 0\}$ וגם $f(u, v) = 1$.

דוגמה

L R



מבנה הוכחת הנכונות

ידוע שהאלגוריתם מוצא זרימה מקסימלית. כלומר $|f|_b = N$ יהיה מרבי.

1) נראה קשר בין זוגים בגרף G לזרימות בראשת N:

(זיווג M ב- G) \longleftrightarrow (זרימה f ב- N)

2) נראה שגודל היזוג שווה תמיד לערך הזרימה. כלומר: $|M| = |f|$

זרימה מקסימלית נותנת זיווג מרבי. \longleftrightarrow

הוכחת נכונות

טענה 1: לכל זרימה f בערכים שלמים ב- N מתאים זיוג M ב- G , שגודלו הוא $|f|$

הוכחה:

בhinתן זרימה f , נגדיר זיוג M באופן הבא:

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

nocih sh- M ziog:

לכל קזקود $L \subseteq u$ נכנסת רק קשת אחת (u, s) שקיבולה 1.

לכן הזרימה המקסימלית שתיתכן לתוך u היא 1.

בזרימה עם ערכים שלמים יש לכל היוטר קשת אחת עם זרימה חיובית שיוצאת מ- u .

← כל קזקוד $L \subseteq u$ מופיע לכל היוטר בקש את אחת של M .

בדומה ניתן להראות שככל קזקוד $R \subseteq v$ מופיע לכל היוטר בקש את אחת של M .

← M הוא זיוג.

nocih sh- $|M| = |f|$: הזרימה בחתך $(s \cup L, t \cup R)$

שווה מצד אחד לזרימה $|f|$, ומצד שני - זה מספר הצלעות שימושות בזיוג M .

טענה 2: לכל זיוג M ב- G , יש זרימה f בערכים שלמים ב- N שערכה הוא $|M|$

הוכחה:

בhinתן זיוג M , נגדיר זרימה f ב- N באופן הבא:

$$\begin{aligned} \text{אם } (v, u) \text{ ב- } M \text{ אז: } \\ f(s, u) = f(u, v) = f(v, t) = 1 \\ (\text{וכמובן } f(t, v) = f(v, u) = f(u, s) = -1) \\ \text{לכל יתר הקשתות נגדיר } f(u, v) = 0. \end{aligned}$$

M הוא זיוג ולכן כל המסלולים $t \rightarrow v \rightarrow u \rightarrow s$ זרים בקדוקודים (פרט לקצוות t, s).

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

יעילות האלגוריתם

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

הוכחה: בניית הרשת N לוקחת זמן ליניארי.

$$m_N = m + n$$

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

היות-שלא תהינה יותר מ- m איטרציות (מדובר?), אזי זמן מציאת הזרימה יהיה $O((m+n)n)$.

היות שבגרף אין קודקודים מבודדים, מתקיים $m \leq 2n$. ולכן $O(mn)$.

תרגיל: מינוי לוועדות

במוסד אקדמי יש n חברי סגל ו- c ועדות שהמוסד רוצה לאייש. לכל חבר סגל יש רשימה של ועדות שבה הוא/היא מעוניין להיות חבר, כשבכל ועדה אמורים להיות לכל היוטר 4 חברי סגל, והחברי סגל יכולים להשתתף בכמה ועדות שירצו.

כתבו אלגוריתם יעיל שモצא האם אפשר לשבץ את כל חברי הסגל לוועדות לפי העדפותיהם, כך שכל חבר סגל ישובן לפחות לוועדה אחת (לא חייבים לאייש את כל הוועדות).

Committee



נתחום הייעילות: נסמן ב- m את מספר האיברים הכללי ברשימות העדפות הפוטנציאליות של חברי הסגל. הביעו את זמן הריצה כפונקציה של n ו- m .

תרגיל: רשימות נסעים

N משפחות יוצאות לטיול ב- M מכוניות.
במשפחה ה- i יש j_i נפשות ובמכונית ה- j יכולים לשכט j_i אנשים.

כתבו אלגוריתם הבודק האם אפשר לשכט את האנשים למכונית
כך שבשם מכונית לא יהיה יותר מאדם אחד מאותה המשפחה.

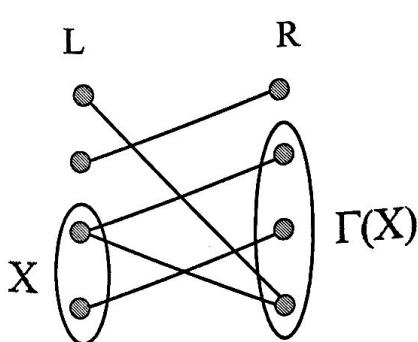


מה ייעילותו?

משפט Hall

יהי $G = (V, E)$ גרף לא-מכoon

קבוצת השכנים של קדוק $V \in X$ היא:



קבוצת השכנים של קבוצת קדוקים X היא:

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

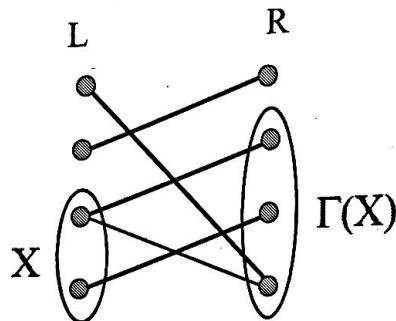
משפט Hall: יהי $G = (L \cup R, E)$ גרף דו-צדדי לא-מכoon.

ב- G יש זיווג שלם ל- L אם ורק אם לכל $X \subseteq L$ מתקיים $|X| \leq |\Gamma(X)|$

הוכחת הכרחיות

אם יש זיוג שלם ל- L ואם $L \subseteq X$, אז לפחות קדקוד ב- X יש בן זוג שונה ב- R .

כל בני הזוג שייכים ל- $(X) \Gamma$ ולכן $|\Gamma(X)| \leq |X|$



הוכחת מספיקות

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

נראה שהאלגוריתם למציאת זיוג מרבי מהזיר ת ובכך נוכיה שיש זיוג שלם ל- L .

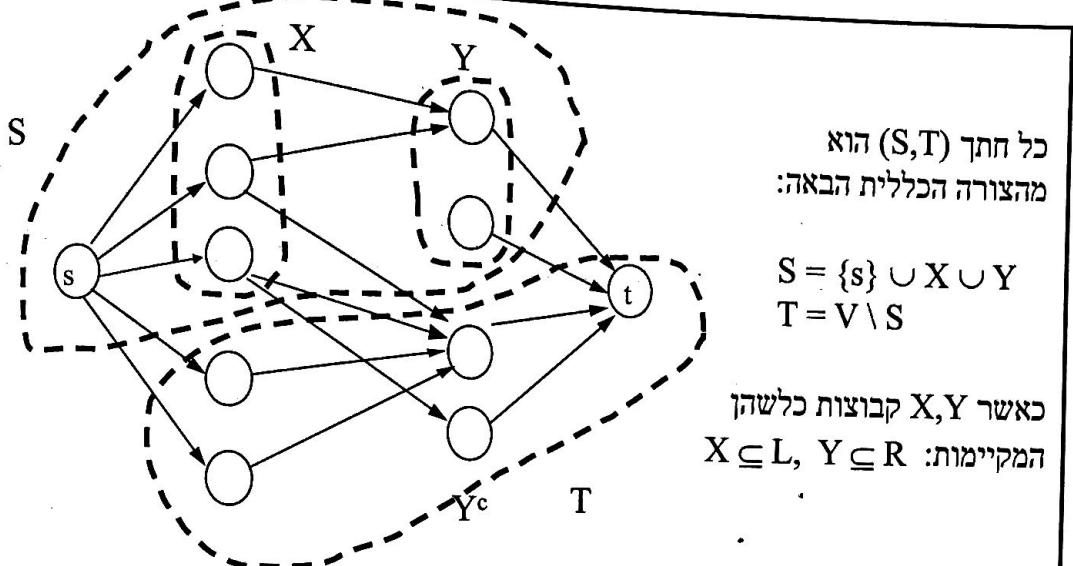
נתבונן ברשת זרימה שמייצר האלגוריתם לזיוג מרבי.

ונוכיה שקיבול החתך המינימלי ברשת הוא n .

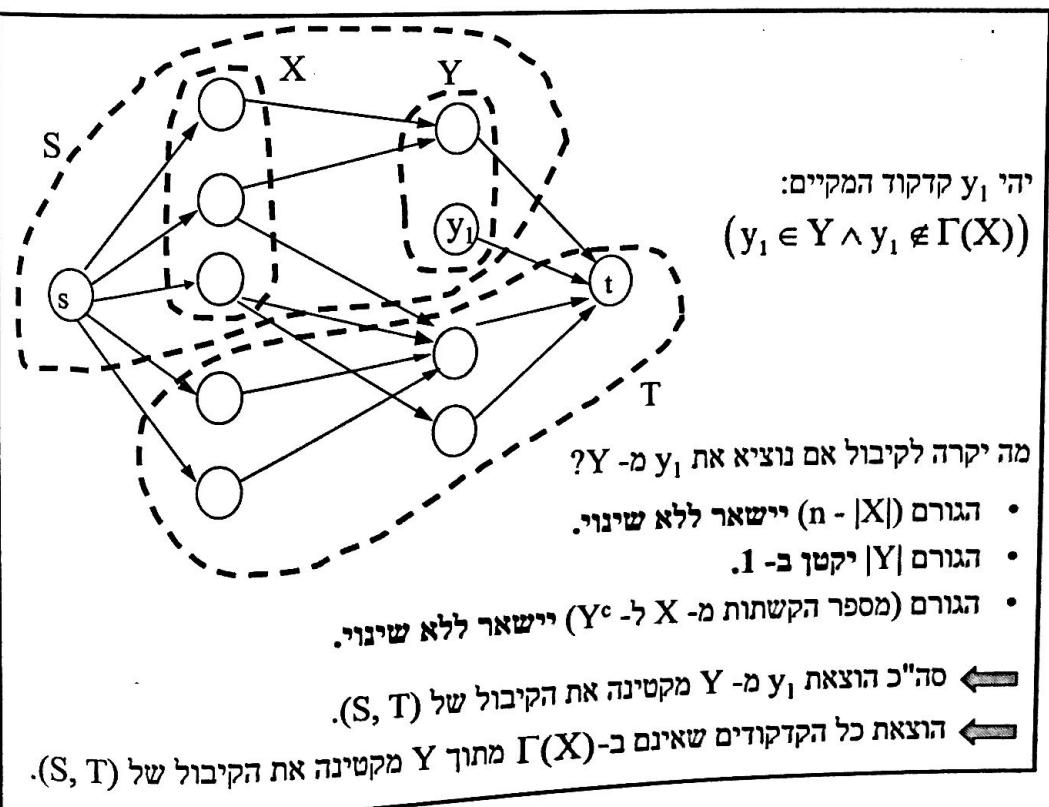
↳ הזרימה המקסימלית שווה בערכה ל- n .

↳ הזיוג המרבי הוא בגודל n .

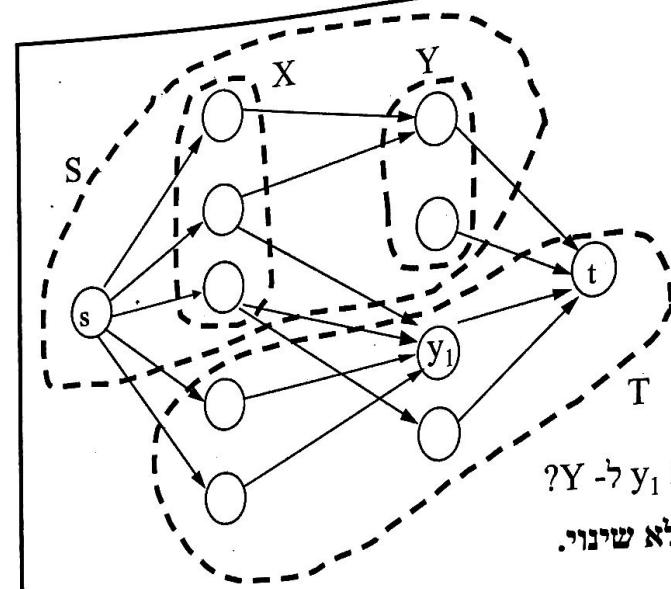
היות שלא תיחסן זרימה גדולה מ- n , די להראות שקיבולו של כל חתך $\leq n$



היות שקיובל כל הקשתות הוא 1,
 $C(S, T) = |T| - |S| =$
 $= (n - |X|) + |Y| + |Y^c|$



יהי y_1 קדקוד המקיים:
 $(y_1 \notin Y \wedge y_1 \in \Gamma(X))$



- הגורם $|X| - n$ יישאר ללא שינוי.
- הגורם $|Y|$ יגדל ב-1.
- הגורם (מספר הקשتوות מ- X ל- Y^c) יקטן לפחות ב-1.

הכנסת y_1 ל- Y לא תגדיל את הקיבול של (S, T) , יתכן שאפילו תקטין אותו.

הכנסת כל הקדקודים ב- $\Gamma(X)$ ל- Y לא תגדיל את הקיבול של (S, T) .

$Y = \Gamma(X)$



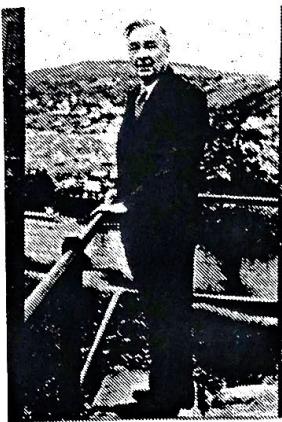
$$C(S, T) \geq (n - |X|) + |\Gamma(X)| + (\Gamma(X)^c \text{ ל- } X \text{ נספחים})$$

$$= (n - |X|) + |\Gamma(X)| \underbrace{\geq n}_{= 0}$$

$$\uparrow \\ |X| \leq |\Gamma(X)|$$

ולכן $n \geq C(S, T)$, כנדרש.

Philip Hall



פיליפ הול (1904-1982):

מתמטיקאי אנגלי.

חקר בעיקר בתחום האלגברה ותורת החבורות.
הוא כיהן את משפט הול לגביו זיווגים בשנת 1935
בדרך קומבינטורית.

דוגמה

טענה: אם בבעיית השידוכים ידוע שמספר הגברים המקבילים על כל גברת הוא תמיד 2, וכן לכל גבר מספר הגברות המעוניינות בו גם הוא 2, אז קיים זיוג מושלם.
הוכחה:

תהי X קבוצה של קדוקדים המייצגים נשים.

נשים לב שמתקיים:

(מספר הצלעות שיצאו מ- X) \geq (מספר הצלעות שנכנסות ל- $(X \cap \Gamma)$).

1. מספר הצלעות היוצאות מ- X הוא בדיק $|X|$.

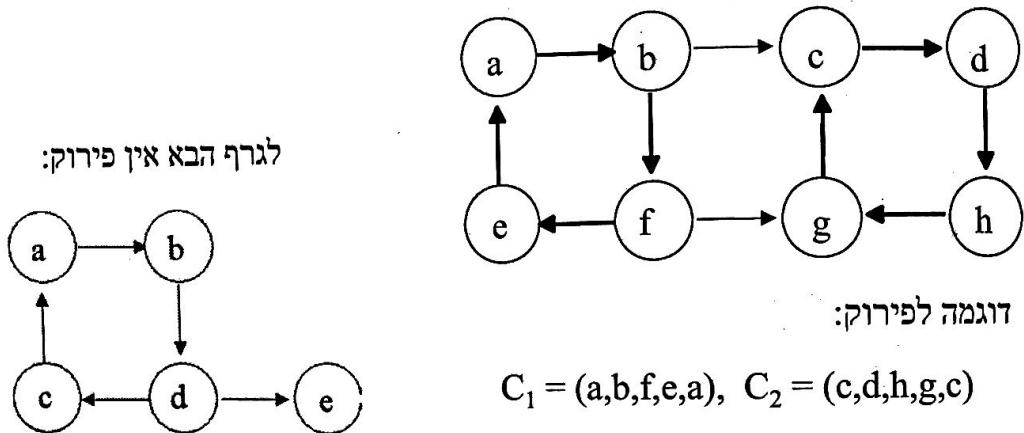
2. מספר הצלעות הנכנסות ל- $(X \cap \Gamma)$ הוא בדיק $|\Gamma(X)|$.

לכן: $|X| \geq |\Gamma(X)|^2 \iff |X| \geq |\Gamma(X)|$.

עפ"י משפט הול קיים זיוג מושלם.

תרגיל - פירוק גרף למעגלים זרים

לגרף מכובן (V, E) יש פירוק למעגלים אם קיים אוסף $C = \{C_1, \dots, C_k\}$ של מעגלים פשוטים זרים בקדקודים הכלול את כל קדודי הגרף. כלומר, כל קדוק $V \in C$ שיחד בדיקת מעגל אחד באוסף C .



תרגיל - פירוק גרף למעגלים זרים

הבעיה: כתבו אלגוריתם המתקבל קלט גרף מכובן G ומカリע האם יש לו פירוק למעגלים

במנז': לגרף G יש פירוק למעגלים אם ורק אם יש תת-קבוצה $E' \subseteq E$ בגודל n של קשותות, כך שבגרף (V, E') דרגת הכניסה ודרגת היציאה של כל קדוק היא 1 בדיקות.

תרגיל מבחינה

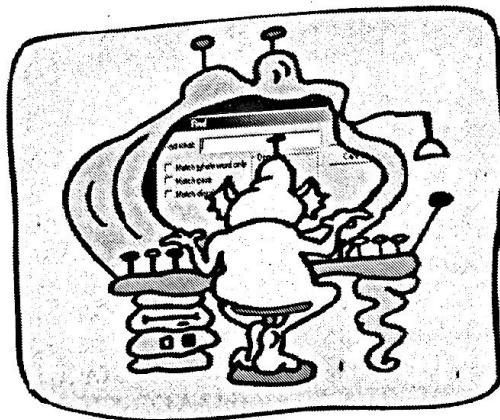
יהיו P, Q שתי חלוקות של הקבוצה $\{1, 2, \dots, n\}$ כשבן חלוקה כוללת בדיקות r חלקיים ורים. החלוקה P כוללת את החלקים P_1, P_2, \dots, P_r , והחלוקה Q את החלקים Q_1, Q_2, \dots, Q_r . נאמר שאייר σ הוא נציג של חלקו כלשהו אם הוא שייך לחלק זהה. ברצוננו למצוא קבוצה S של r איברים מתחום $\{1, 2, \dots, n\}$ כך שלכל אחד מהחלקים של P ו- Q יהיה נציג בקבוצת S , או להזכיר שאין קבוצה כזו של נציגים.

דוגמה: עבור $n=8, r=3$.

- החלוקה P היא: $P_1 = \{1, 2, 3\}, P_2 = \{4, 6\}, P_3 = \{5, 7, 8\}$
 $Q_1 = \{1, 2\}, Q_2 = \{3, 5\}, Q_3 = \{4, 6, 7, 8\}$
הנציגים שנבחרו: $\{1, 5, 6\}$
- החלוקה P היא: $P_1 = \{1\}, P_2 = \{2\}, P_3 = \{3, 4, 5, 6, 7, 8\}$
 $Q_1 = \{1, 2\}, Q_2 = \{3, 4, 5\}, Q_3 = \{6, 7, 8\}$
במקרה זה אין 3 נציגים שיקיימו את תנאי השאלה.

כתבו אלגוריתם לבנייה שמקבל כקלט שני מספרים טבעיים r, n ואת החלוקות P, Q וモץ באhor פלט קבוצה כזאת של נציגים אם יש כזאת, או מודיע שאיו קבוצה כנדש.

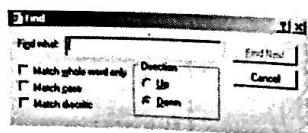
התאמת מחרוזות, אוטומטים סופיים ושפות רגולריות



בעיית התאמת מחרוזות

גָלוּט:

- טקסט $[1, \dots, n]$
- תבנית $[1, \dots, m]$
- סימון: $\Sigma = \{a, b, c, \dots, z\}$ או $\Sigma = \{0, 1\}$
- למשל $T = aba$



- פָלֶט:
- המיקום הראשון שבו מופיעה התבנית P בטקסט T
 - כל המופיעים של P ב- T

$$T = \underline{ab} \underline{ab} \underline{aca}, \quad P = aba$$

של ברווז DNA

CTACCC ?



Duck rearranged beta-globin gene (hemoglobin alpha-2 beta-2)

CAGGGAGCTG CAGCACTGG AATCAAAGAC GGGGGACAGG GTGGTGGCGG
ATCTGGGCAC CGTGCCTGG GGCCCCACCC TGACACCGCA TCCCCTCCCT
TGGGGTGCCA GGCTGGGGGG GCCCCTCCCG GGACCCAGCC AATGGAGGGG
TGCAGGGGGA AGAGGAGGGG CCCTGCAGG GCTATAAAAG TGGGGTCGGG
ACAGCCGCTC GCCAGCGTGC GATCCCCGCG GGAGCAAGAG CCGAGACCTC
CTCCGTACCT GCAGCCACAC GCTACCCTCC ACCCGACACC ATGGTGCACT
GGACAGCCGA GGAGAACAG CTCATCACCG GCCTCTGGGG CAAGGTCAAT
GTGGCCGACT GTGGAGCTGA GGCCCTGGCC AGGTAGGTTG GGCTCCCGGG
CATCGGCTCG GCTGCACCTG CATGGAGAAA CCGCTGCAGT CACAGGAGGG
AGCATTAAACG GGGTGTGTGT GCTTCTCCCC CCTTCTCT GCAGGCTGCT
GATCGTCTAC CCCTGGACCC AGAGGTTCTT CGCCTCCTTC GGGAACCTGT ...

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
```

פעולות ההשוואה: $P[1..m] = T[i..i+m - 1]$ מתבצעת כך:

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

$O((n - m)m) = O(nm)$ עילוות:

אלגוריתמים מתקדמים יותר

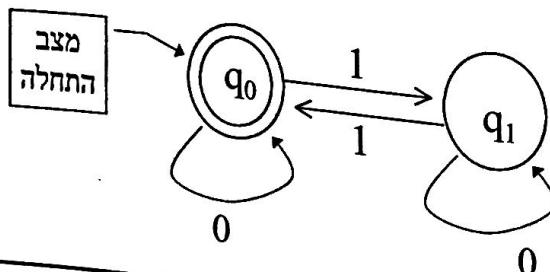
- אלגוריתמים המבוססים על אוטומטים סופיים.
- אלגוריתם קנות-מוריס-פראט.
- אלגוריתמים אחרים – Boyer-Moore, רבין-קארפ ועוד.
- חיפוש תבניות מורכבות, התאמות חלקיות וכו'.

אוטומט סופי – תיאור לא פורמלי

אוטומט סופי הוא מכונה שכוללת מספר סופי של מצבים ועוברת ממצב למצב על קלט חיצוני. נתיחס לאוטומט כמודל לסוג מסוים של חישובים על מחרוזות.

המכונה מתוארת לעתים קרובות ע"י דיאגרמת מצבים.

דוגמה: מכונה שקולטת מחרוזות בינהarity ומחשבת את הזוגיות (parity) שלה.



סימונים מקובלים בקשר למילים/מחרוזות

Σ – אלף-בית: קבוצה סופית של אותיות.

$x \cdot w$ שרשור של המילה x אחרי המילה w .

a^n – קיצור למילה $\underbrace{a \cdot a \cdot \dots \cdot a}_n$

ϵ – המילה הריקה.

$|w|$ – האורך של המילה w .

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

$$\bigcup_{i=1}^{\infty} \Sigma^i = \Sigma^+ \quad \bigcup_{i=0}^{\infty} \Sigma^i = \Sigma^*$$

אוטומט סופי דטרמיניסטי

(DFA) Deterministic Finite Automaton

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

Σ – א"ב

Q – קבוצה סופית של מצבים

$q_0 \in Q$ – מצב תחيلي

F – קבוצת מצבים סופיים $F \subseteq Q$ (נקראים גם "מזהים")

$\delta: Q \times \Sigma \rightarrow Q$ – פונקציית מעברים. המצב הבא = (תו, מצב הנוכחי) δ

הчисוב של אוטומט

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

מצב הנוכחי: q_0

סרט
קלט

מצב הנוכחי: q_0

$$\delta(q_0, 0) = q_0$$

0 1 1 0

מצב הנוכחי: q_1

$$\delta(q_0, 1) = q_1$$

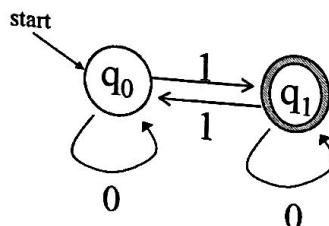
0 1 1 0

אם בסוף הקלט האוטומט במצב מזוהה/סופי או המילה בשפה שהאוטומט מזוהה, ואחרת לא.

תיאור גרפי של אוטומט

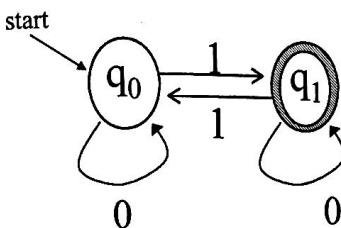
נייצג אוטומט כגרף מכונן באופן הבא:

- המצבים של האוטומט יהיו קדוקדי הגרף.
- הקדוקדים שמייצגים מצבים מזוהים יסומנים באופן מיוחד על ידי שני עיגולים.
- הקדוקוד שמייצג את המצב ההתחלתי יסומן באופן מיוחד.
- את פונקציית המעברים יביעו קשתות הגרף:
אם $q_j = \delta(q_i, c)$ אז ניציר קשת מהקדוקוד q_i לקדוקוד q_j ומעליה יופיע הסימן c .



דוגמה

ייצוג גרפי:



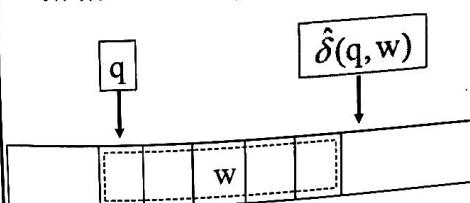
ייצוג טקסטואלי:

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

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

הגדרת עזר - הפונקציה $\hat{\delta}$

נגדיר פונקציה $\hat{\delta}: Q^* \times \Sigma \rightarrow Q$ כך שלכל $w \in \Sigma^*$, $\hat{\delta}(q, w) =$ המצב אליו הגיע האוטומט אם הוא מתחילה במצב q וקורא את המילה w .



הפונקציה תוגדר באופן אינדוקטיבי:

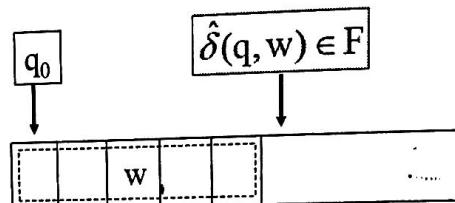
$$\begin{aligned} \hat{\delta}(q, \varepsilon) &= q \\ \text{צעוד: } \text{תהי } w \in \Sigma^* \text{ ותהי } a \in \Sigma \text{ ויהי } q' = \hat{\delta}(q, w) & \quad - \text{ביסיס: המילה הריקה} \\ \hat{\delta}(q, wa) &= \delta(q', a) \end{aligned}$$

או נגדיר: $\hat{\delta}(q, wa) = \delta(\hat{\delta}(q, w), a)$

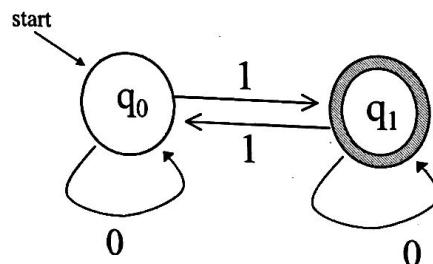
זיהוי מילים ע"י אוטומט

הגדרה: נאמר שאוטומט מקבל/مزזה מהרוות w אם: $\hat{\delta}(q_0, w) \in F$

הגדרה: קבוצת המילים שהאוטומט מזזה נקראת השפה שהאוטומט מזזה.

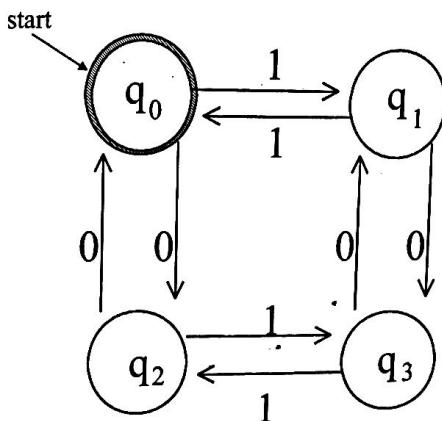


לוגמה 1



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

לוגמה 2



טענה: האוטומט מזהה את המחרוזות של 0,1 שבחן יש מספר האפסים זוגי ומספר האחדים זוגי, כלומר את השפה:

$$L = \{w \in \{0,1\}^* \mid \#0 \text{ is even} \wedge \#1 \text{ is even}\}$$

הוכחה: נוכיח באינדוקציה על אורך המילה w שמתקיים:

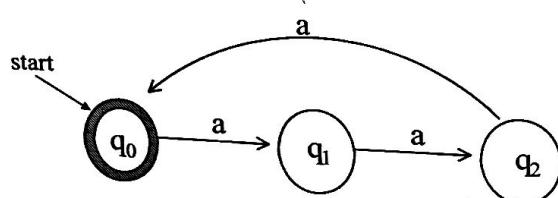
$$\hat{\delta}(q_0, w) = \begin{cases} q_0 & \#0 \text{ is even} \wedge \#1 \text{ is even} \\ q_1 & \#0 \text{ is even} \wedge \#1 \text{ is odd} \\ q_2 & \#0 \text{ is odd} \wedge \#1 \text{ is even} \\ q_3 & \#0 \text{ is odd} \wedge \#1 \text{ is odd} \end{cases}$$

מבחן זו נובע שהגדרת q_0 כ מצב מזהה יחיד מביאה ליזיהוי השפה L .

איך מוכחים שאוטומט מזהה שפה L?

1. מגדירים את תפקידי המ מצבים:
לכל מצב מאפינים את קבוצת המילים שגיעה אליו.
2. מוכחים שמלים מוגיעות למצב הנכון (באינדוקציה על אורך המילים):
 - בסיס – הוכחת נכונות למילה הריקה. היה ש- $q_0 = \hat{q}(\epsilon)$,
עלינו להראות ש- q_0 הוא המצב הנכון למילה זו.
 - צעד – לכל $S \in a^*$, בהנחה ש- S מביאה את האוטומט למצב המתאים
עבורה, יש להוכיח שפונקציית המעברים גורמת למילה wa לעבור
למצב הרצוי עבורה.
3. מוכחים את נכונות המ מצבים המזהים:
יש להראות שהמלים בשפה L הן בדוק המילים המוגיעות למצבים ב- F.

דוגמה 3



טענה: האוטומט מזהה את קבוצת המחרוזות מהצורה a^n כאשר $n \bmod 3 = 0$.

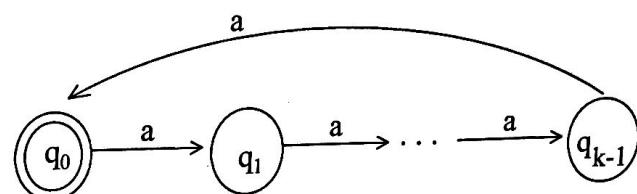
בניה של משפחת אוטומטים

יהי $k > 0$ קבוע כלשהו, ויהי $\{a\} = \Sigma$.
נבנה אוטומט לזיהוי קבוצת המחרוזות a^n עבורן $n \mod k = 0$.

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

הגדרת תפקידים: $|w| \mod k = i$ יהיה q_i אם ורק אם i מצב מזוהה: q_0 .

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



תרגיל

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

L_1 : כל המילים עם a אחד לפחות

L_2 : כל המילים עם לפחות b אחד

L_3 : כל המילים עם a אחד בדיק

תרגיל

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

L_1 : המחרוזות המתחילה ברכף $.aba$.

L_2 : המחרוזות המשתיימות ברכף $.aba$.

L_3 : המחרוזות המכילות את הרכף $.aba$.

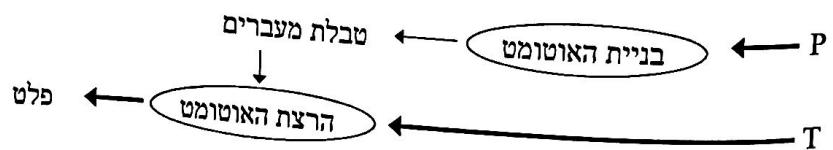
פתרון לבניית התאמת המחרוזות על ידי אוטומט

אלגוריתם לזיהוי המופעים של תבנית P בטקסט T:

שלב א: בניית אוטומט שמצוה מחרוזות שמשתיימות בתבנית P.

שלב ב: נרץ את האוטומט עם קלט T.

בכל פעם שהאוטומט מגע למצבמצוה, נרץ על התאמת.



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
    
```

יעילות:

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

\Leftarrow כל צעד לוקח זמן קבוע

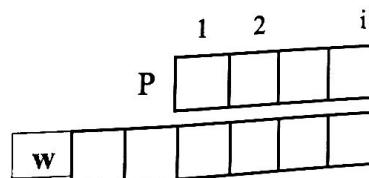
\Leftarrow זמן הריצה הכלול: $\Theta(n)$

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

אוטומט לזיהוי מחרוזות מסוימות ב-P

מצבים: $Q = \{q_0, \dots, q_m\}$

הגדרת תפקדים: עבור $0 \leq i \leq m$, w יהיה q_i אם ורק אם i הוא המספר המקסימלי כך ש- w מסתיימת ב- $P[1..i]$ (תחילה של המילה P).

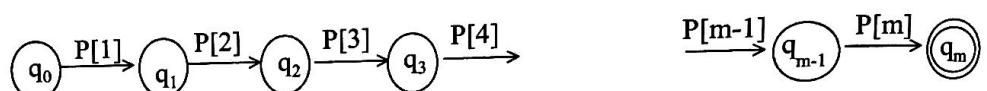


תרגום לארכית מדוורת: יש סיפה של w השווה לרשא $[i..1]p$ של p .

המצב יהיה q_m אם "מ" w מסתיימת במופיעשלם של P . \leftarrow

בנייה האוטומט

1. גדר $F = \{q_m\}$, $Q = \{q_0, \dots, q_m\}$
 2. גדר טבלה δ ובה שורה לכל מצב, ועמודה לכל אות בא"ב.
 2. נמלא את שורה 0 (תרגיל).
 3. לכל $m > i > 0$
- נמלא את מעבר ה- $\text{match } \delta[i, P[i+1]] \leftarrow i+1$ (האות הבאה תואמת לתבנית):

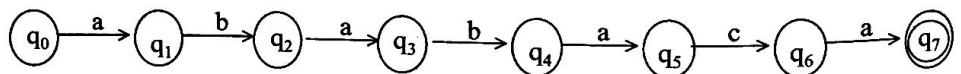


- לכל $c \in \Sigma$: נמלא את מעברי ה- $\delta[i, c]$ mismatch
- 4. לכל $c \in \Sigma$: נמלא את $\delta[m, c]$

דוגמה

$$\Sigma = \{a, b, c\}$$

$$P = a \ b \ a \ b \ a \ c \ a$$

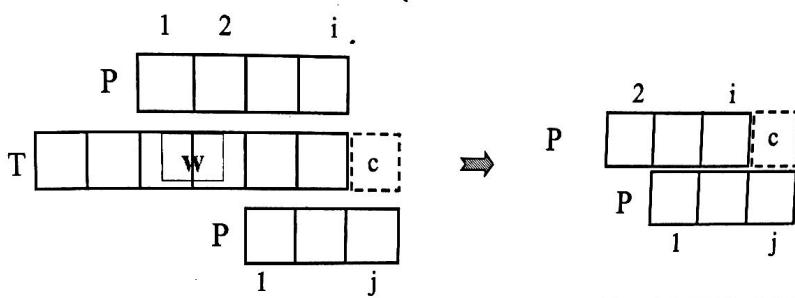


לא כל הצלעות מצוירות

$T = b \ a \ c \ b \ a \ b \ a \ b \ a \ a \ b \ c \ a \ b$: טקסט

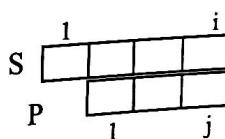
חישוב מעברים $\delta[i,c]$ mismatch

- בעוד הקודם הגיענו ל מצב q . לנוכח $P[1..i]$ הייתה הירשא המקסימלית של P שהיא גם סיפא של הקלט w .
 - c הינו הבא בקלט, $P[i+1] \neq c$.
- מבקש: ה- j המקסימלי כך שהירשא $[j..1]P$ היא סיפא של המחרוזת wc .
- אבחן: הירשא המקסימלית $[j..1]P$ שהיא סיפא של המחרוזת wc , היא גם הירשא המקסימלית של P שהיא סיפא של המחרוזת $c[i..2..j]$.



בעית עזר

קלט: בניית P ומחרוזת S (באורך $i \geq m$)
פלט: אורך הירשא המקסימלית של P המופיעיה כסיפא של S .



פתרונות נאייבי:

SuffixMatch (P, S)

```

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

```

מה ייעילות?

אלגוריתם לבניית האוטומט

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

```

יעילות האלגוריתם

COMPUTE- δ (String p[1...m])

```

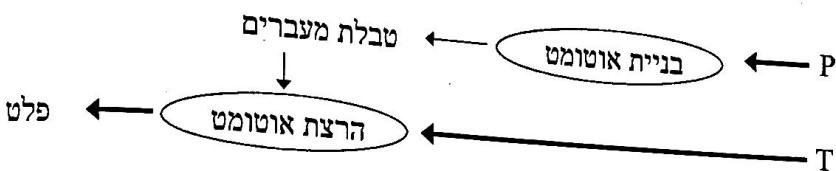
 $\delta[0, P[1]] \leftarrow 1$ 
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0, c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i, P[i+1]] \leftarrow i+1$ 
    for each  $c \in \Sigma, c \neq P[i+1]$  do
         $\delta[i, c] \leftarrow \text{SuffixMatch}(P, P[2 \dots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

```

$\Theta(i^2)$

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

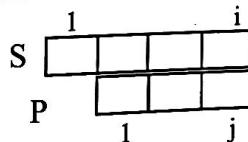
יעילות הפתרון לבעיית התאמת המחרוזות



$$\Theta(|\Sigma| \cdot m^3 + n)$$

פתרון יעיל יותר לבעיית העזר

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



תזכורת:

באוטומט שלנו, עבור $m \leq j \leq 0$, $(w, q_0, \delta) \hat{\rightarrow} q_j$ אם ורק אם
 זה הוא אורך הרישא המקסימלית של P המופיע כסיפה של w .

מסקנה:

אם נריץ את האוטומט שבנו על המילה S , אז המצב $(S, q_0, \delta) \hat{\rightarrow} q_j$ אלין יגיע לאוטומט
 יהיה בדיק הערך j המבוקש.

אלגוריתם לבניית האוטומט

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])

```

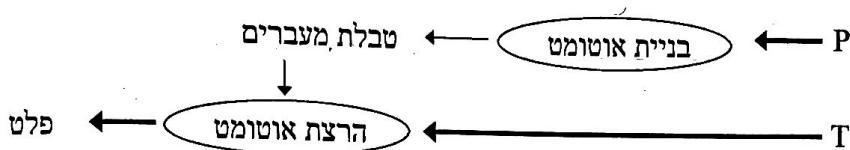
 $\delta[0, P[1]] \leftarrow 1$ 
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0, c] \leftarrow 0$ 
for  $i \leftarrow 1, \dots, m-1$  do
     $\delta[i, P[i+1]] \leftarrow i+1$ 
    for each  $c \in \Sigma, c \neq P[i+1]$  do
         $\delta[i, c] \leftarrow \text{RUN}(\delta, P[2 \dots i]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

```

$\Theta(i)$

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

יעילות הפתרון לבעיית התאמת המחרוזות



$$\Theta(|\Sigma| \cdot m^2 + n)$$

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:
הויצאת חישובים חוזרים
(איינוריאנטיים)
אל מחוץ ללולאה הפנימית.

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
     $\delta[m, c] \leftarrow \delta[\text{state}, c]$ 

```

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

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:
העברת מידע מאיטרציה
לאיטרציה הבאה.

נסמן ב- $state_i$ את ערכו של המשתנה state בתחילת האיטרציה ה- i . אז:

$$state_i = \hat{\delta}(q_0, P[2..i])$$



$$state_{i+1} = \hat{\delta}(q_0, P[2..i+1]) = \hat{\delta}(q_0, P[2..i] \cdot P[i+1]) = \delta(state_i, P[i+1])$$



$state \leftarrow \delta[state, P[i+1]]$ נעדכן בסוף איטרציה:

שיפור הייעילות (2)

COMPUTE- δ (String p[1..m])

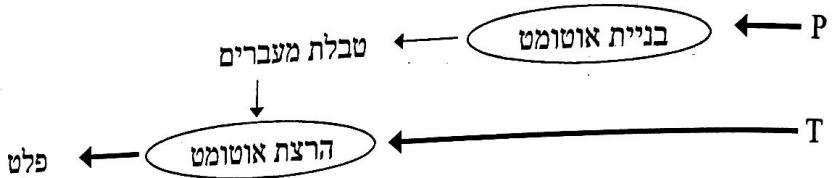
```

 $\delta[0, P[1]] \leftarrow 1$ 
for each  $c \in \Sigma, c \neq P[1]$  do
     $\delta[0, c] \leftarrow 0$ 
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
         $\delta[i, c] \leftarrow \delta[state, c]$ 
    state  $\leftarrow \delta[state, P[i+1]]
for each  $c \in \Sigma$  do
     $\delta[m, c] \leftarrow \delta[state, c]$$ 
```

זמן הריצה:

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

יעילות הפתרון לבעיית התאמת המחרוזות



$$\Theta(|\Sigma| \cdot m + n)$$

אלגוריתם Knuth, Morris, Pratt

הרעיון:

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

במונח הבניה:

נשמר בטבלה δ את ערכי המשתנה state state [1, ..., m] את ערכיו המשתנה state state (מהגרסה הקודמת)
 $\hat{\delta}$ = ערכו של המשתנה state state בהתחלת האיטרציה ה- $i = (q_0, P[2..i])$

כשצריך את δ :

נקרא לפונקציה שמחשבת את $(c, i)\delta$ במקום לשלוף ערך זה מטבלה.

מימוש הפונקציה δ

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

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

```
δ(State i, Character c) // Transition Function
if c = p[i+1] then
    return i + 1
else if i = 0 then
    return 0
else
    return δ(state[i],c)
```

הערה: רואים ש-state נחוץ רק במקרה של אי-התאמה (failure).

עדכון המערך state

תזכורת:

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

אבל כשבנינו את האוטומט כולם, עדכנו בסוף האיטרציה ה- i :

$$\text{state} \leftarrow \delta[\text{state}, P[i+1]]$$

```
SetState(String p[1..m]) // Fill state array.
state[1] ← 0
for i ← 1,...,m-1 do
    state[i+1] ← δ(state[i], p[i+1])
```

אלגוריתם KMP:

קלט: תבנית P וטקסט T

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

```
SetState(String p[1...m]) // Fill state array.
state[1] ← 0
for i ← 1,...,m-1 do
    state[i+1] ← δ(state[i], p[i+1])
```

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

```
δ(State i, Char c)
if c = p[i+1] then
    return i + 1
else if i = 0 then
    return 0
else
    return δ(state[i],c)
```

יעילות כוללת: $\Theta(n+m)$

KMP יעילות אלגוריתם

יעילות Search(δ, T)

- נספר קריאות ל- δ לפי סוגים.
- מספר הקריאות שאינן מבוצעות רקורסיבית הוא τ (אורך הטקסט T).
- מספר הקריאות הרקורסיביות:
 - כל קריאה רקורסיבית מקטינה את המצב.

נשתמש בכלל חשבון הבנוק: אם ידוע שהחישוב תמיד אי-שלילי, או סכום המשיכות אינו עולה על סכום ההפקדות.

המצב תמיד אי-שלילי וסכום ה"הפקדות" אינו עולה על τ .
 \Leftarrow מספר ה"משיכות" (הקריאות הרקורסיביות) אינו עולה על τ .

לסיכום: זמן הריצה הוא $\Theta(n)$.

KMP –יעילות אלגוריתם

יעילות של $\text{SetState}(P)$

- מספר הקריאה לפונקציה δ מתוך SetState הוא m (אורך התבנית P).
 - מספר הקריאה רקורסיביות חסום ע"י, מספר הקריאה ללא-רקורסיביות (אותו טיעון).
- זמן הריצה הוא $\Theta(m)$.

יעילות כוללת: $\Theta(n+m)$

Knuth, Morris, Pratt

Donald Knuth (נולד 1938):

חוקר אמריקני בתחום מדעי המחשב. תרם לפיתוח של תורת הסיבוכיות והכניס לשימוש פופולארי במדעי המחשב את O, Ω .
זכה בפרס טירינג בשנת 1974.



Vaughan Pratt (נולד 1944):

חוקר אמריקני בתחום מדעי המחשב שנולד באוסטרליה.
פיתח אלגוריתמים למילון, חישוב ובדיקה ראשונית.



James H. Morris (נולד 1941):

חוקר אמריקני בתחום מדעי המחשב, בתחום של שפות תכנות.



תרגיל

בנו את מערך state עבור המחרוזת popopipopol מעל
 $\Sigma = \{p,o,i\}$
הרייצו על הטקסט: pppopopopopipopo

```
SetState(String p[1...m]) // Fill state array.
state[1] ← 0
for i ← 1,...,m-1 do
    state[i+1] ← δ(state[i], p[i+1])
```

```
δ(State i, Char c)
if c = p[i+1] then
    return i + 1
else if i = 0 then
    return 0
else
    return δ(state[i],c)
```

	1	2	3	4	5	...	6
state							

שאלות מבחינות

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

0 0 0 0 0 0 0 0

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

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

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

איזה תנאי מספיק והכרחי מקיימת המחרוזת P?

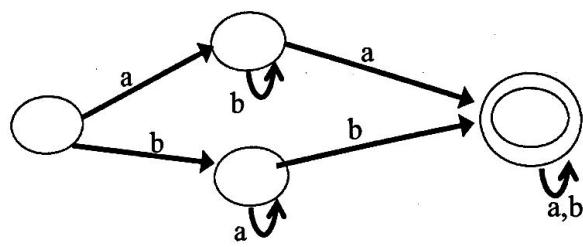
כתבו אלגוריתםיעיל שמקבל:

- קבוצת תווים Σ
- פרמטר m
- מטריצה בגודל $(m+1) * |\Sigma|$

ומחזר כפלט האם המטריצה היא אוטומט סופי ליזהו מחרוזת,
 ואם כן מהי המחרוזת.

- בנו אוטומט סופי דטרמיניסטי מעל $\{a,b,c\}$ שמקבל מחרוזות בהן לפחות אחד מהרצפים ab או bc מופיע.

• תארו את השפה שהאוטומט הבא מזוהה.



ועד תרגיל...

בנו אוטומט שמזוהה את השפה של כל המילים מעל הא"ב $\{y, x\}$ כך שbullet רישא הפרש בערך מוחלט בין מספר ה- x למספר ה- y אינו עולה על 2.

מילים בשפה: $yyxx$, $xyxx$,

מילה שאינה בשפה: $xxxxx$.

ומה לגבי אוטומט שמזוהה את השפה של כל המילים מעל הא"ב $\{y, x\}$ כך שbullet רישא הפרש בערך מוחלט בין מספר ה- x למספר ה- y אינו עולה על 2?

שפות רגולריות

הגדרה: פעולות רגולריות על קבוצות:

$$S_1 \cup S_2 = \{ u \mid u \in S_1 \text{ or } u \in S_2 \}$$

$$S_1 \cdot S_2 = \{ u \cdot v \mid u \in S_1 \text{ and } v \in S_2 \}$$

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

• סגור קליני:

הגדרה – שפה רגולרית:

- הקבוצה $\{\}$ היא שפה רגולרית.
- הקבוצה $\{\epsilon\}$ היא שפה רגולרית.
- לכל $S \in \Sigma$ הקבוצה $\{a\}$ היא רגולרית.
- אם S_1 ו- S_2 רגולריות אז גם $S_1 \cup S_2$ ו- $S_1 \cdot S_2$ רגולריות.
- אם S רגולרית אז גם S^* רגולרית.

ביטויים רגולריים

אפשר לתאר שפות רגולריות ע"י ביטויים רגולריים.

הגדרה – ביטוי רגולרי:

- \emptyset הוא ביטוי רגולרי שמתאר את השפה $\{\}$.
- ϵ הוא ביטוי רגולרי שמתאר את השפה $\{\epsilon\}$.
- a הוא ביטוי רגולרי שמתאר את השפה $\{a\}$.
- אם E ביטוי רגולרי שמתאר את השפה S אז E^* הוא ביטוי רגולרי שמתאר את השפה S^* .
- אם E_1 ו- E_2 ביטויים רגולריים שמתארים את השפות S_1 , S_2 או $(E_1 + E_2)$ הוא ביטוי רגולרי שמתאר את השפה $S_1 \cup S_2$ ו- $(E_1 \cdot E_2)$ הוא ביטוי רגולרי שמתאר את השפה $S_1 \cdot S_2$.

דוגמאות לשפות ולביטויים רגולריים

הביטוי $(d \cdot c) + (a \cdot b)$ מתאר את השפה $\{abd, cd\}$

נרשום בקיצור גם $d(ab+c)$

תרגיל: מהי השפה המתוארת ע"י הביטוי הרגולרי?

1. $(a+b)^*$

2. $a(a+b)^*b$

3. $(aaa)^*$

תרגילים

בנו ביטויים רגולריים עבור השפות הבאות כאשר $\Sigma = \{a, b, c\}$:

1. כל המילים שמסתיימות ב- b .
2. כל המילים שלא מסתיימים ב- b , לא כולל את ϵ .
3. כל המילים שלא מסתיימים ב- b , כולל ϵ .

תרגילים נוספים לבניית אוטומטים

בנו אוטומטים שמצוים את השפות הבאות כאשר $\Sigma = \{a,b,c\}$:

1. כל המילים עם a אחד לפחות.
2. כל המילים עם לפחות b אחד.
3. כל המילים עם a אחד בלבד.

כתבו ביטוי רגולרי עבור כל שפה.

הקשר בין שפות רגולריות לאוטומטים

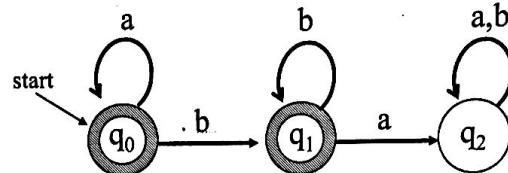
משפט: L שפה רגולרית אם ורק אם קיים אוטומט סופי דטרמיניסטי שמצווה את L

דוגמה

(כוון ←)

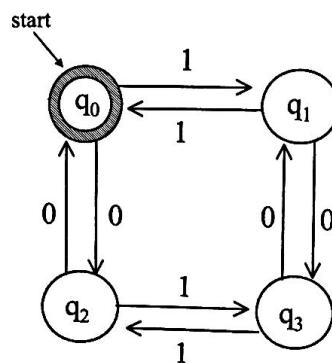
נתבונן בשפה הרגולרית המתוארת ע"י הביטוי: a^*b^*

אוטומט שמצווה אותה:



(כוון →)

נתבונן באוטומט הבא:



הביטוי הרגולרי המתאים לשפה שמצווה האוטומט:

$$(11 + 00 + (10 + 01)(11 + 00)^*(10 + 01))^*$$

שפות לא רגולריות

משפט: השפה הבאה אינה רגולרית: $L = \{a^n b^n \mid n \geq 0\}$

הוכחה: נניח בsvilleה שקיים אוטומט סופי שמצווה את L ונתבונן בקבוצת התחליות:

$$\{ a^n \mid n \geq 0 \}$$

קבוצה זו אינסופית. אולם היה שקבוצת המ מצבים סופית, קיימים $n_1 \neq n_2$ שעבורם:

$$\hat{\delta}(q_0, a^{n_1}) = \hat{\delta}(q_0, a^{n_2})$$

$$\hat{\delta}(q_0, a^{n_1} b^{n_1}) = \hat{\delta}(q_0, a^{n_2} b^{n_1})$$

זו סטירה כי האוטומט-Amor לקל את $a^{n_1} b^{n_1} a$ ולדוחות את $a^{n_2} b^{n_1} a$.



תבנית להוכחה של אי-רגולריות

משפט: השפה הבאה אינה רגולרית: ...

הוכחה: נניח בsvilleה שקיים אוטומט סופי שמצווה את L ונתבונן בסדרת תחליות:

$$u_1, u_2, \dots, u_n, \dots$$

מכיוון שהסדרה אינסופית, בהכרח קיימים $n_1 \neq n_2$ כך ש:

$$\hat{\delta}(q_0, u_{n_1}) = \hat{\delta}(q_0, u_{n_2})$$

כך נובע שלכל המשך v שנבחר,

$$\hat{\delta}(q_0, u_{n_1} v) = \hat{\delta}(q_0, u_{n_2} v)$$

ונקבל סטירה ע"י בחירת המשך v כך שרק אחת מamilim אלו תהיה בשפה.

תרגיל מבחינה

האם השפה

$$L = \{a^n \mid \exists i, n = 2^i\}$$

(כלומר שפת כל המילים מעל $\{a\}$ שאורכם הוא חזקה של 2)
רגולרית? הוכחו תשובהכם במדויק.

אוטומטים עם זיכרון

התכוונה המהותית של שפות רגולריות היא שניתן להוותן ללא שימוש
בזיכרון (למעט המצב של האוטומט).

כדי להזוהות שפות לא רגולריות מוסיפים לאוטומט זיכרון נוסף.

42

МОНА

automat

memor

b
a
c
a

מחסנית

אוטומט מחסנית – מהחסנית מכילה תווים.

$L = \{a^n b^n \mid n \geq 0\}$ תרגיל: איזה מודל חישובי יודע לזהות את השפה

שפות חסרות הקשר

השפות שמצוות על ידי אוטומטי מהסנית נקראות שפות חסרות הקשר.

Context Free Languages

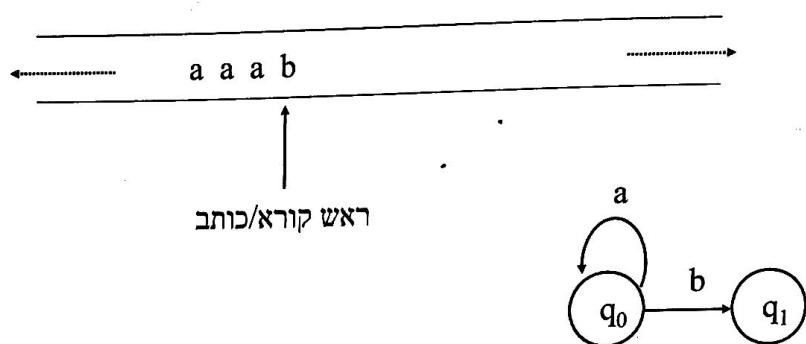
משפט: L שפה חסרת הקשר אם ורק אם קיימ אוטומט מהסנית שמצוות את L .

משפט: השפה הבאה אינה חסרת הקשר $L = \{a^n b^n c^n \mid n \geq 0\}$

- לאוטומטי מהסנית תפרקן חשוב בפתרון בעיות של עיבוד טקסטים.
- קיים כלים נוספים להוכחה ששפה אינה רגולרית או שאינה חסרת הקשר.

מכונת טיורינג

כדי לזהות שפות שאין חסרות הקשר מוסףים לאוטומט סרט אינסופי עם ראש קורא/כותב. המכונה נקראת מכונת טיורינג.



מחלקות של שפות

שפות רגולריות



שפות שניתן לזהות ע"י אוטומט מונה



שפות שניתן לזהות ע"י אוטומט מחסנית



שפות שניתן לזהות ע"י אלגוריתם כלשהו
(מכונת טיורינג)

דוגמאות

מה נחוץ כדי לזיהות את השפות הבאות?

$$\text{BAL} = \{w \mid \#0(w) = \#1(w)\}$$

שפה המחרוזות המאוזנות

$$\text{PAL} = \{w\$w^{\text{reverse}} \mid w \in \Sigma^*\}$$

שפה הפלינדרומיים המסומנים

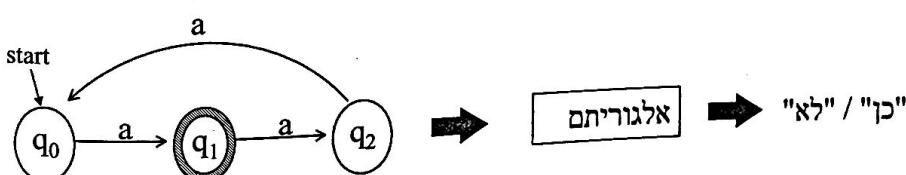
$$\text{DBL} = \{w\$w \mid w \in \Sigma^*\}$$

שפה המחרוזות הכפולות

תרגיל אינטגרציה

כתבו אלגוריתם המזהה אוטומט סופי דטרמיניסטי A ועונה על השאלה:

האם A מזהה שפה לא ריקה?



• מהו מבנה הקלט לאלגוריתם?

• מהי הייעולות?

אלן טיורינג

(1912-1954) Alan Turing



מתמטיקאי בריטי שנחשב לאבי מדעי המחשב.

הגדיר את מכונת טיורינג ואת המושג הפורמלי של אלגוריתם.

במהלך מלחמת העולם השנייה סייע לצבא הבריטי

לפצח את הצפנים של הגרמנים.

על שמו נקרא פרס טיורינג שנחשב לנובל של מדעי המחשב.

טיורינג נשפט והורשע ב- 1952 על היותו הומוסקסואל. והוא מת כתוצאה מהרעליה.



בשנת 2009 התנצל ראש ממשלת בריטניה על היותו המחפיר אליו

ובשנה 2013 מלכת אנגליה נתנה לו חנינה.

בנק אנגלי הוחלט ב- 2019 להוציא לכבודו שטר חדש.