בעיית קבוצות זרות – Union/Find

חומר קריאה לשיעור זה

Chapter 22- Data Structures for Disjoint Sets (440 – 462)

קבוצות זרות – בעיית Union/Find

נתון עולם של איברים U, $|U| = n$. לצורך הנוחות נניח $U = \{1,2,\dots,n\}$.

מבנה נתונים לשמירת קבוצות זרות תומך בפעולות הבאות:

1. Makeset(i) <u>מחזיר קבוצה</u> חדשה בעלת איבר בודד i.
2. Find(i) <u>מחזיר</u> את <u>הקבוצה</u> לה שייך האיבר i.
3. Union(p,q) מאחד את הקבוצות p ו- q. כלומר, <u>מחזיר קבוצה</u> חדשה, המכילה את איחוד האיברים בקבוצות p,q. (הקבוצות p,q המקוריות חדלות מלהתקיים).

<u>לדוגמא:</u> נניח כי ברגע נתון הגענו למצב:

$\{1,3\} \quad \{5\} \quad \{2,4,6,7\}$

ואז אנו מבצעים את הפעולות:

p = Find (6)

q = Find (5)

r = Union (p,q)

s = Find (6)

<u>תוצאות:</u>

- הקבוצה שהמשתנה r שומר היא $\{5,2,4,6,7\}$.
- המשתנים p ו- q אינם מחזיקים יותר קבוצות.
- המשתנה s מחזיק את אותה קבוצה כמו r.

דוגמא לשימוש

נתונות n ערים $\{1, \dots, n\}$. בתחילה כל הערים מנותקות. כל זמן מה בונים כביש ישיר בין שתי ערים. יש לאפשר את הפעולות הבאות:

Add-Road(x, y) הוסף כביש ישיר בין שתי ערים x ו- y.

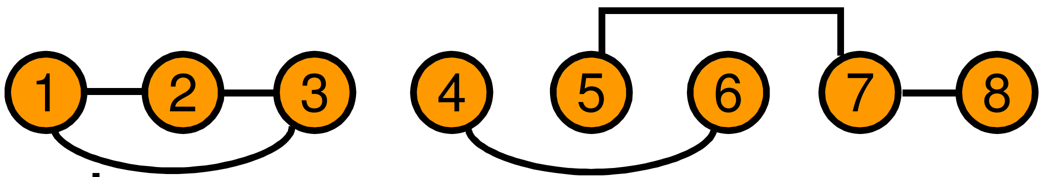
Check-Connectivity(x, y) קבע האם שתי הערים x, y מחוברות במסלול כלשהו.

<u>פתרון:</u>

אתחול: ניצור n קבוצות: $\{1\}, \{2\}, \dots, \{n\}$ ע"י: for ($i = 1$; $i \leq n$; i++) Makeset(i) ;

Add-Road(x, y) ממומשת ע"י Union(Find(x),Find(y)).

Check-Connectivity(x, y) מחזירה "אמת" אם ורק אם Find(x)=Find(y).



הכביש שהוסף	אוסף הקבוצות הזרות							
	{1}	{2}	{3}	{4}	{5}	{6}	{7}	{8}
(1,2)	{1,2}		{3}	{4}	{5}	{6}	{7}	{8}
(1,3)	{1,2,3}			{4}	{5}	{6}	{7}	{8}
(2,3)	{1,2,3}			{4}	{5}	{6}	{7}	{8}
(5,7)	{1,2,3}			{4}	{5,7}	{6}		{8}
(4,6)	{1,2,3}			{4,6}	{5,7}			{8}
(7,8)	{1,2,3}			{4,6}	{5,7,8}			

מימוש נאיבי ראשון

נשתמש במערך A מסוג SET בגודל n ובמונה counter המאותחל ל- 0. בתא $A[i]$ נשמור את שם הקבוצה אליה שייך האיבר i. במימוש זה Set הוא פשוט int.

לדוגמא: $\{2,4,6,7\}$ $\{5\}$ $\{1,3\}$ מיוצגות ע"י המערך הבא :

	1	2	3	4	5	6	7
A	**4**	**12**	**4**	**12**	**5**	**12**	**12**

1. Makeset(i) ממומשת ע"י $A[i]$ =++counter.
2. Find(i) ממומשת ע"י $A[i]$.
3. Union(p,q) ממומשת ע"י הגדלת ה-counter באחד, מעבר על המערך A וכתיבת ערך ה- counter בכל מקום שבו כתוב p או q. הפונקציה מחזירה את הערך של counter.

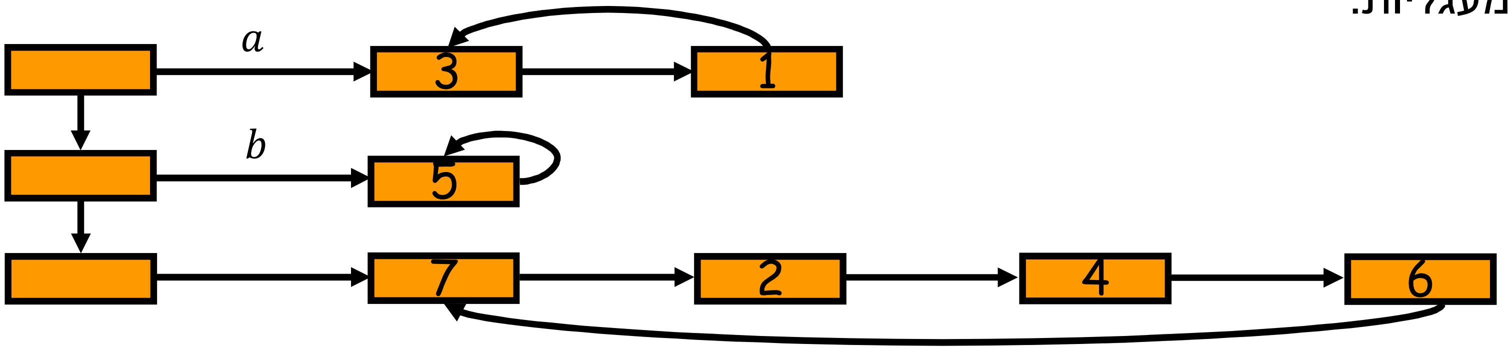
דוגמא: לאחר Union(p,q) עם $p = 4, q = 5$, ו-counter=12, מאוחדות הקבוצות p ו- q ומקבלים:

	1	2	3	4	5	6	7
A	**13**	**12**	**13**	**12**	**13**	**12**	**12**

סיבוכיות הזמן: Find ו-Makeset דורשים $O(1)$ זמן ו-Union דורשת $O(n)$ זמן.

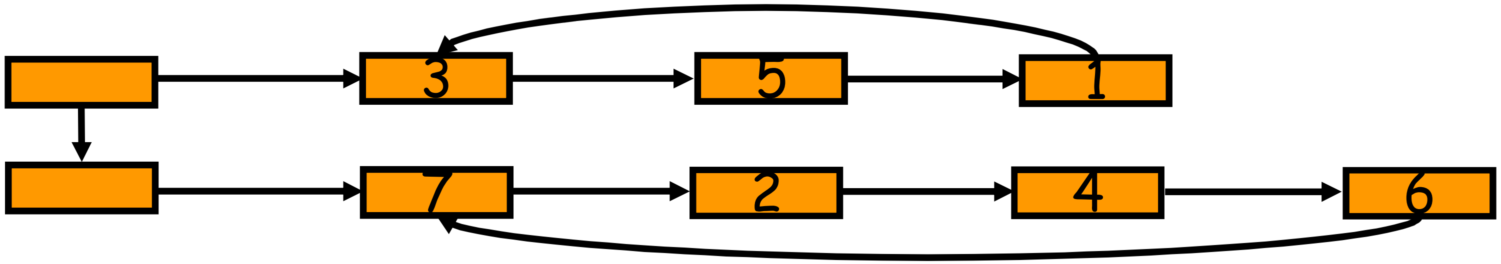
מימוש נאיבי שני

נייצג כל קבוצה כרשימה מעגלית. נחזיק רשימה מקושרת של מצביעים אל הרשימות המעגליות:



Union(p,q) - ממומשת ע"י אחוד הרשימות המוצבעות ע"י p ו- q. זמן $O(1)$.
במימוש זה SET הוא מצביע לצומת מסוג NODE.

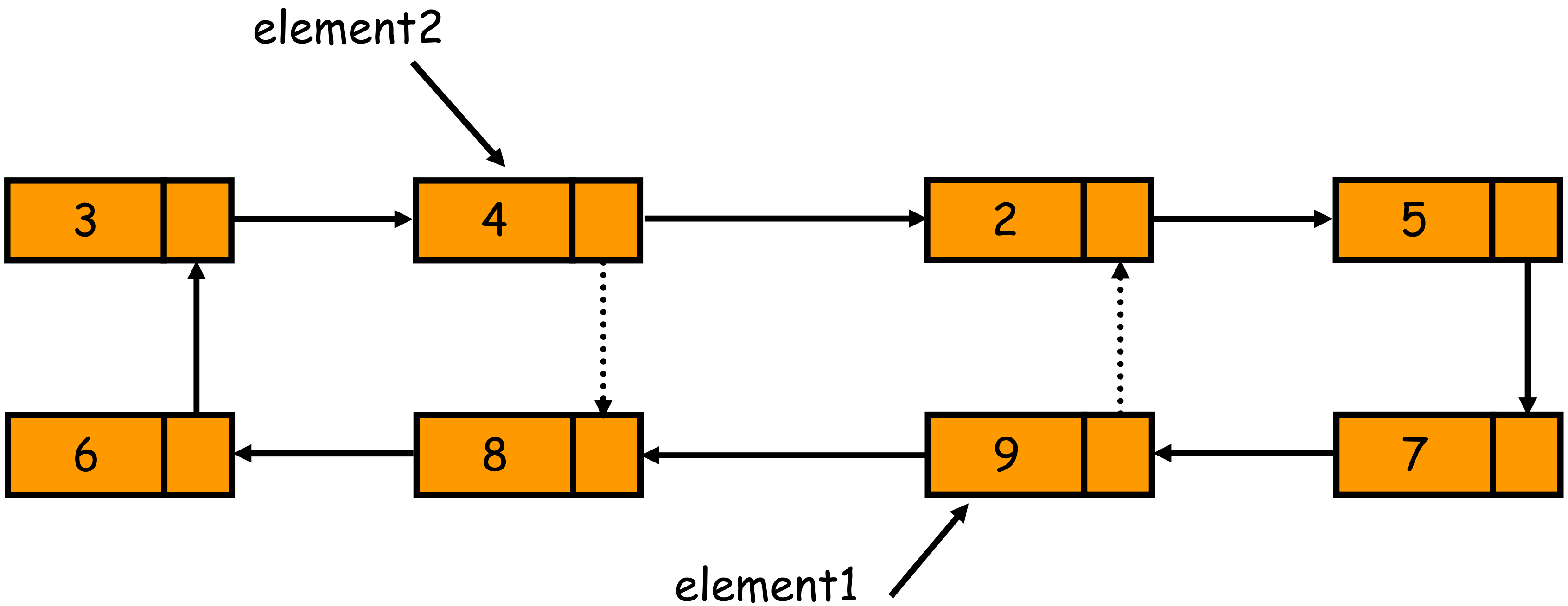
<u>דוגמא:</u> לאחר Union(a, b) נקבל:



Find(i) - ממומשת ע"י מעבר על כל הרשימות עד שנמצא האיבר i. <u>זמן:</u> $O(n)$.

Makeset(i)- ממומשת ע"י הוספת רשימה עם איבר בודד. <u>זמן:</u> $O(1)$.

תזכורת: איחוד רשימות מעגליות



הערות בהקשר לקידוד בשפת C/C++

כותרות הפונקציות והטיפוסים בכל המימושים הניתנים בשיעור זה:

```
SET   Makeset(VALUE  value);
SET   Find(VALUE  value);
SET   Union(SET p,q);

typdef  int  VALUE;
typdef  int  SET;
typdef  NODE* SET;
```

במימוש הראשון ובמימושים נוספים

במימוש השני ובמימושים נוספים

שמות האיברים

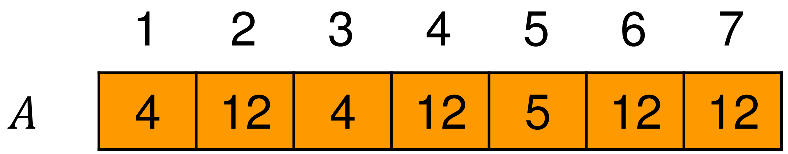
<u>הערה</u>: בהגדרת הבעיה שמות האיברים נבחרו להיות מספרים שלמים בתחום ידוע מראש. ניתן להשתמש בשמות כלליים (כלומר מחרוזות תווים). לשם כך ניתן להשתמש במבנה נתונים הממיר ביעילות בין שמות כלליים ומספרים שלמים, למשל Hash table.

<u>הערה נוספת</u>: במימוש הראשון (ובמימושים הבאים שנראה) הנחנו כי נתון לנו מספר האיברים n מראש ולכן יכלנו להניח קיום מערך בגודל n למימוש שלנו. אם לא כך המצב ניאלץ לממש מערך דינמי במקום המערך.

סיבוכיות זמן המימושים

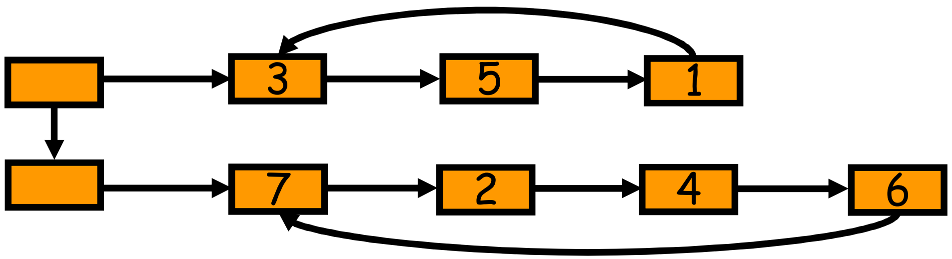
<u>במימוש הנאיבי הראשון:</u>

1. Makeset ב- $O(1)$ זמן.
2. Find ב- $O(1)$ זמן.
3. Union ב- $O(n)$ זמן.



<u>במימוש הנאיבי השני:</u>

1. Makeset ב- $O(1)$ זמן.
2. Find ב- $O(n)$ זמן.
3. Union ב- $O(1)$ זמן.



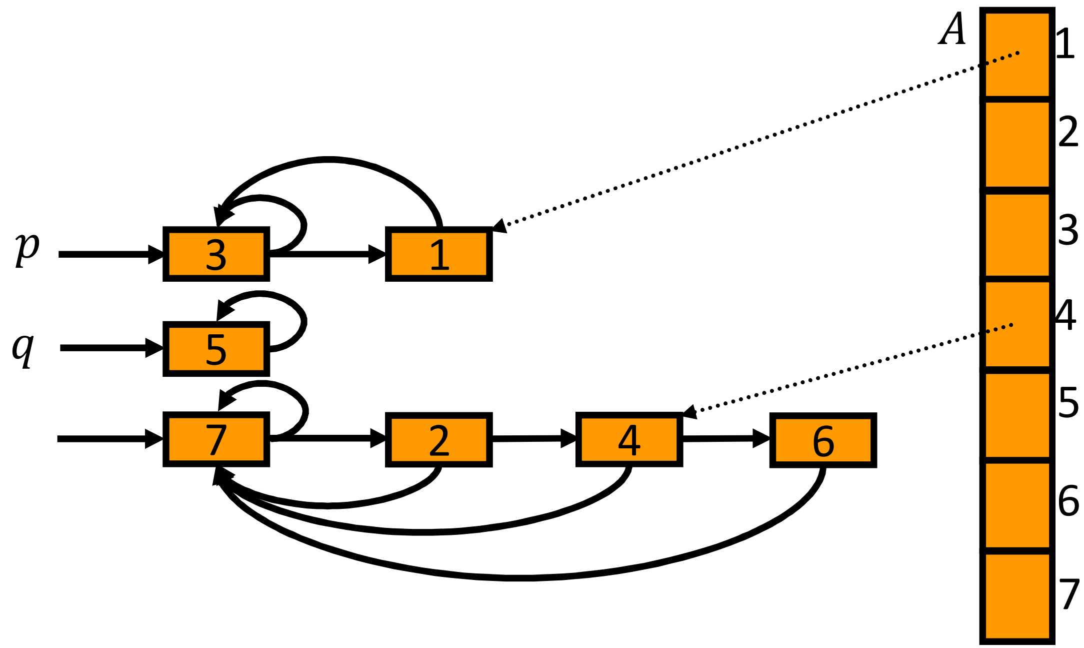
<u>נפתח כעת פתרון המאפשר:</u>

1. Makeset ב- $O(1)$ זמן.
2. Find ב- $O(1)$ זמן.
3. Union ב- $O(\log n)$ זמן משוערך.

<u>ולסיום</u>, נפתח מבנה נתונים בו הזמן המשוערך לפעולות Union ו-Find הוא "כמעט קבוע" וזמן פעולת Union בודדת הוא $O(1)$.

מימוש שלישי

נייצג כל קבוצה כרשימה. כל איבר ברשימה מצביע גם לראש הרשימה וגם לאיבר הבא ברשימה. נחזיק מערך מיפוי איברים $A[i]$ ובו מצביע לאיבר i. קבוצה מיוצגת ע"י מצביע לראש הרשימה המתאימה.



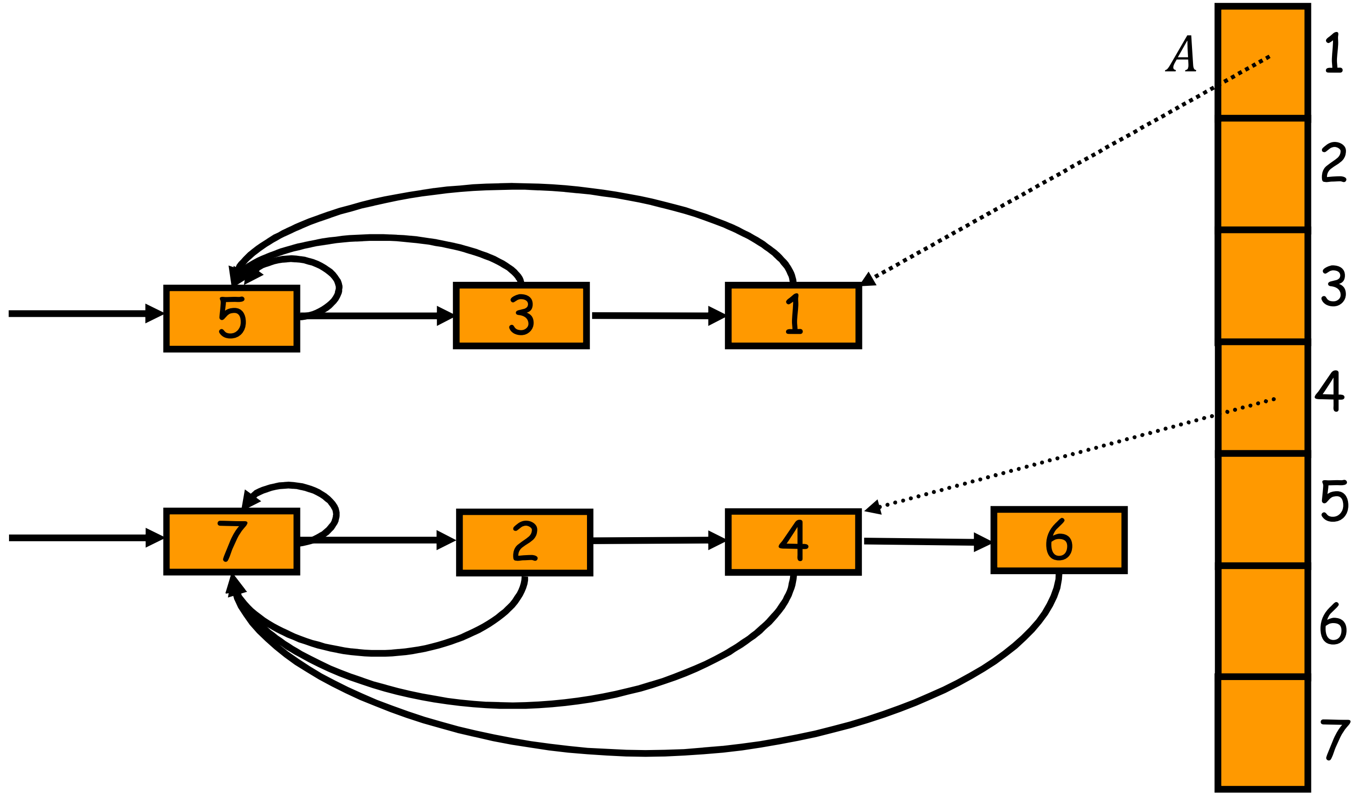
Makeset(i) – ממומשת ע"י יצירת צומת חדש המוצבע ע"י עצמו ו-$A[i]$. <u>זמן:</u> $O(1)$.

Find(i) – ממומשת ע"י מעבר $A[i]$ והחזרת המצביע לראש הרשימה. <u>זמן:</u> $O(1)$.

Union(p, q) - ממומשת ע"י איחוד הרשימות המוצבעות ע"י p, q לרשימה אחת וכן עדכון מצביעי האיברים לראש הרשימה החדשה. הפונקציה מחזירה את ראש הרשימה החדשה. <u>זמן:</u> $O(n)$.

מימוש שלישי - דוגמא

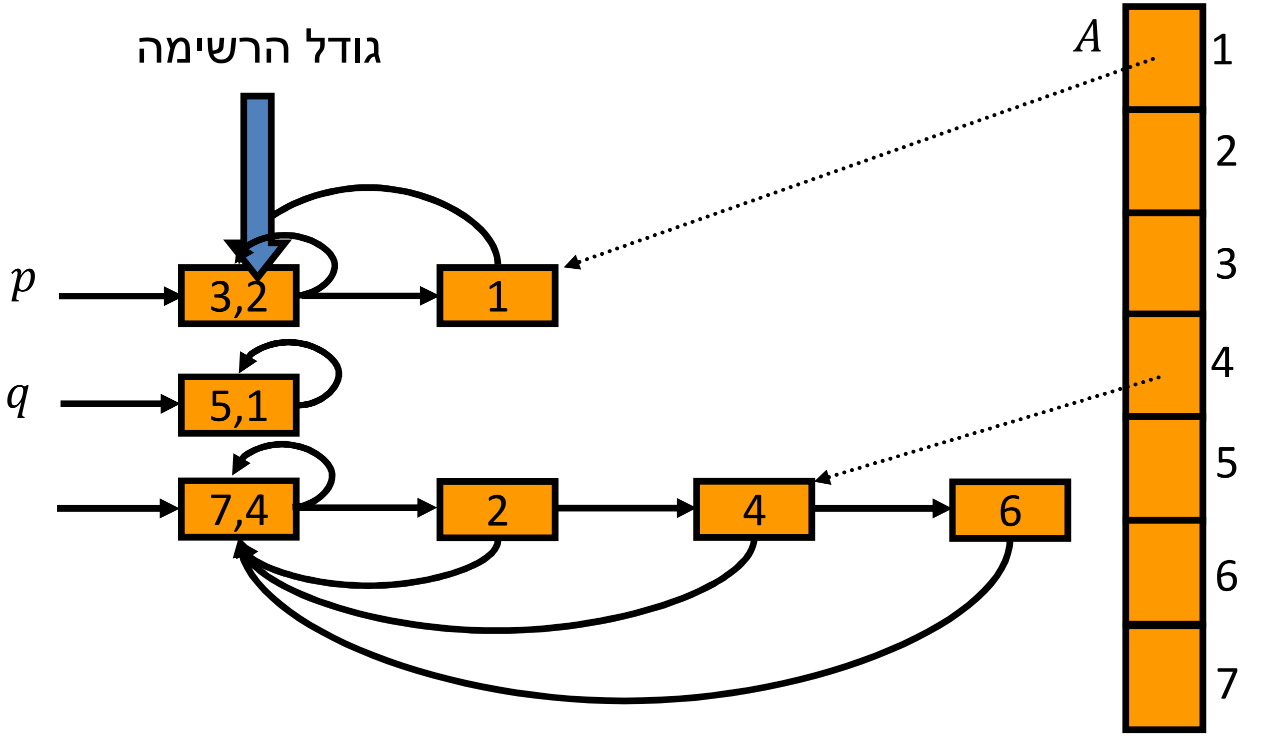
דוגמא: לאחר $\text{Union}(p,q)$, עבור p ו-q מהשקף הקודם, נקבל:



כאשר מאחדים שתי קבוצות יש לשנות באחת הקבוצות את כל המצביעים לכותרת החדשה. אינטואיטיבית, ברור שאת שינוי המצביעים עדיף לעשות בקבוצה הקטנה מבין השתיים (בניגוד לדרך בה פעלנו בדוגמא זו).

מימוש שלישי משופר

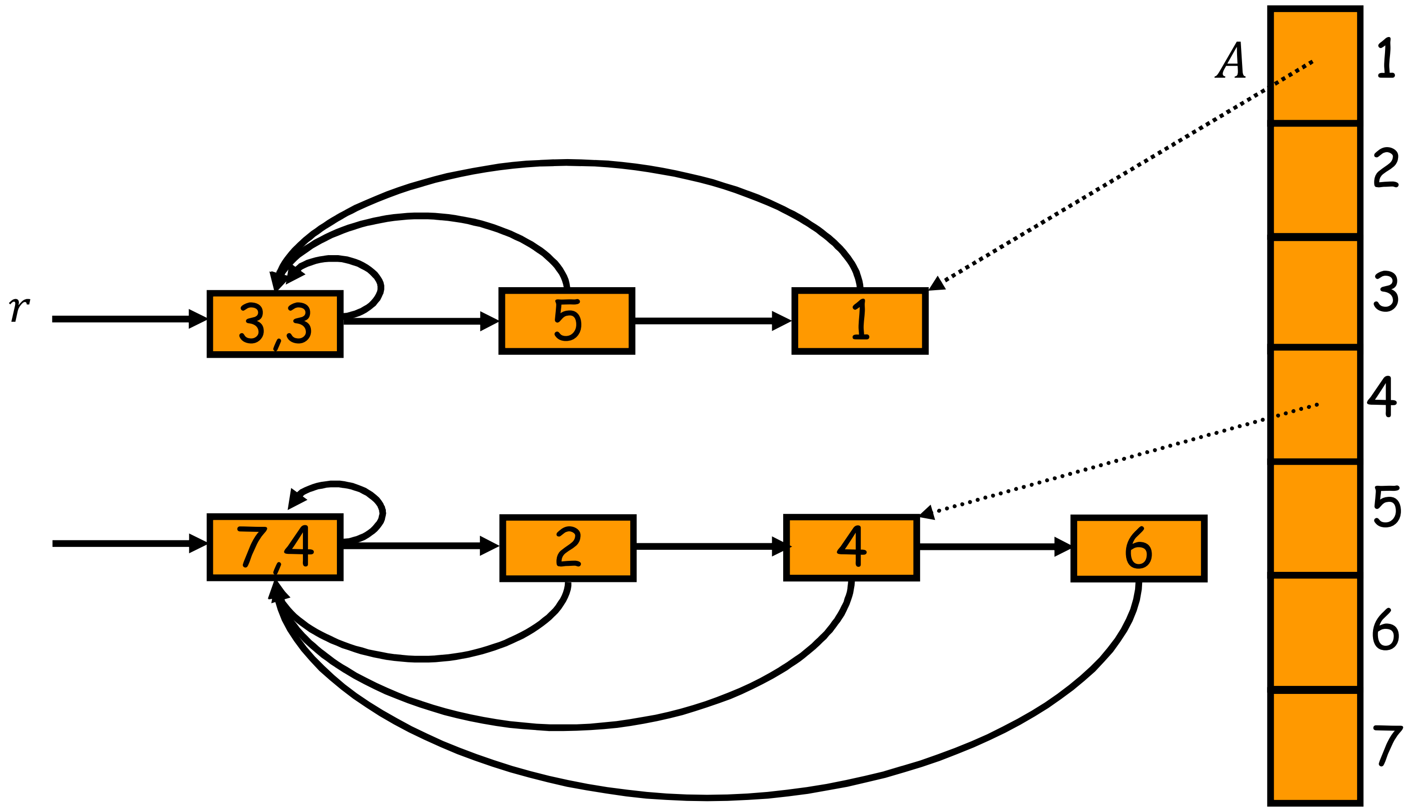
שוב נייצג כל קבוצה כרשימה. <u>בנוסף, בראש הרשימה נשמור את גודלה</u>.
Find יתבצע כמו קודם ו- Union יתבצע ע"י הוספת הקבוצה הקטנה לתוך הקבוצה הגדולה. כלומר ראש הרשימה החדשה יהיה ראש הרשימה הגדולה יותר, וכל המצביעים יצביעו עליו.



כיצד נבצע r = Union(p, q) ?

דוגמא

לאחר r = Union(p, q) נקבל:



שימו לב שהקבוצה $\{5\}$, שהייתה הקטנה מבין שתי הקבוצות שהתאחדו, הוספה לתוך הקבוצה הגדולה יותר, $\{3,1\}$, והקבוצה החדשה שנוצרה מוצבעת ע"י r.

ניתוח השיפור

<u>טענה א</u>: אם בכל איחוד מוסיפים את הקבוצה הקטנה לגדולה אזי כל איבר x משנה את הקבוצה אליה הוא שייך לכל היותר $\log_2 n$ פעמים כאשר n הוא מספר האיברים במבנה.

<u>הוכחה</u>: בכל פעם שאיבר x משנה את הקבוצה אליה הוא שייך הוא עובר לקבוצה חדשה הגדולה לפחות פי 2 מקבוצתו העכשווית. לאחר לכל היותר $\log_2 n$ מעברים כאלו הקבוצה הנוצרת מכילה את כל n האיברים ולפיכך שיוכו של איבר x לא ישתנה יותר.

<u>טענה א (נוסח חזק יותר)</u>: אם בכל איחוד מוסיפים את הקבוצה הקטנה לגדולה אזי כל איבר x <u>שאי פעם משנה את הקבוצה אליה הוא שייך</u> משנה את הקבוצה אליה הוא שייך לכל היותר $\log_2 n$ פעמים כאשר n הוא מספר האיברים במבנה.

ניתוח השיפור

<u>משפט</u>: אם בכל איחוד מוסיפים את הקבוצה הקטנה לגדולה אזי הזמן הכולל לביצוע סדרה כלשהי של m פעולות Union הוא לכל היותר $O(m \log n)$.

<u>הוכחה</u>:

- כל פעולת איחוד עולה $O(1)$ ועוד זמן שתלוי לינארית במספר האיברים ששינו את הקבוצה אליהם הם שייכים.
- כל פעולת Union מוסיפה איבר יחיד לאוסף האיברים שאי פעם שינו את הקבוצה אליה הם שייכים (ראש הקבוצה הקטנה). מכאן שמספר האיברים שאי פעם משנים קבוצה הוא בדיוק m.
- לפי טענה א וההבחנה הנ"ל, נקבל כי מספר שינויי הקבוצה על פני m פעולות Union הוא לכל היותר $O(m \log n)$ ומכאן שהזמן לביצוע m פעולות Union הוא $O(m + m \log n) = O(m \log n)$.

כלומר, כל פעולת Union לוקחת זמן משוערך (Amortized time) של $O(\log n)$.

זמן משוערך - תזכורת

הגדרה: אם m פעולות מתבצעות בתוך זמן M, אזי הזמן המשוערך לכל פעולה מוגדר להיות M/m.

נימוקים להגדרה: כאשר ישנה סדרה ארוכה של פעולות שרובן קלות לביצוע (כגון find במימוש האחרון) וחלקן קשה לביצוע (כגון union באותו מימוש), אז ההשפעה של הפעולות הקשות מתחלקת על סדרת הפעולות כולה. יתכן גם מצב שזמן בצוע הפעולה משתנה כתוצאה מ"עזרה" הניתנת בזמן פעולה קודמת ואז הזמן המשוערך הוא מדד המתחשב בתרומה של פעולה אחת לרעותה.

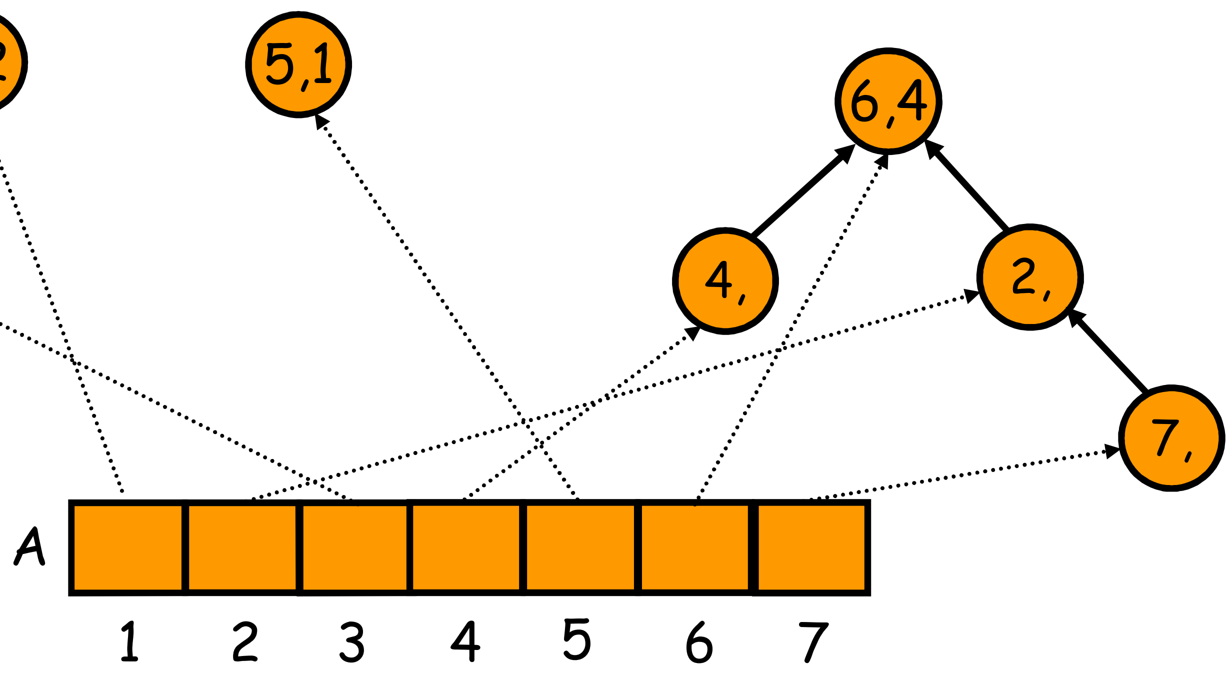
להלן סוגי ממוצעים שראינו בקורס:

1. בשיעור זה ה"ממוצע" נלקח על פני סדרת הפעולות שמשתמש מייצר.
2. באלגוריתם רנדומלי (למשל עבור רשימת דילוגים), הממוצע נלקח על פני תוצאות הגרלה שהאלגוריתם מבצע.
3. בניתוח בניית עץ חיפוש בינרי אקראי, הממוצע נלקח על פני פילוג הקלט.

מימוש נוסף: עצים הפוכים (Up trees)

לכל קבוצה ניצור עץ הפוך (בנים מצביעים להורה) ובו צומת לכל איבר בקבוצה.
שורש העץ יכיל גם את מספר איברי הקבוצה. בנוסף נחזיק מערך גישה לאיברים כמו במימוש השלישי.
בזמן פעולת Union תולים את שורש העץ הקטן יותר מתחת לשורש העץ הגדול יותר ומעדכנים את גודל הקבוצה המאוחדת.

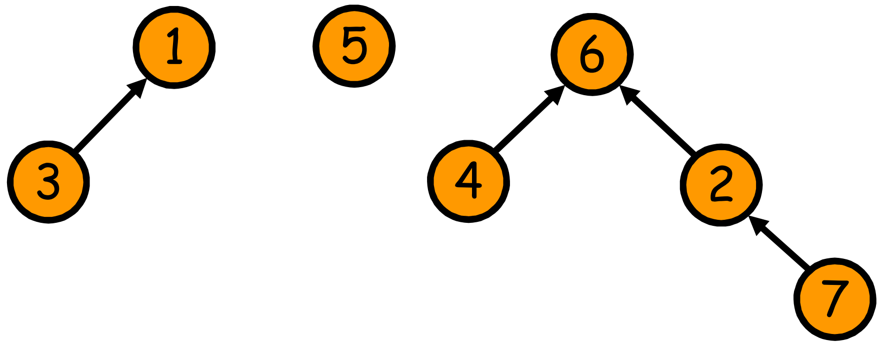
<u>לדוגמא:</u> הקבוצות {2,4,6,7} {5} {1,3} מיוצגות ע"י:



מימוש עצים הפוכים בעזרת מערכים

בהנחה שמספר האיברים ידוע מראש, ניתן לייצג את כל הרשימות במערכים ללא שימוש במצביעים. ניתן לעשות זאת גם בכל המימושים הקודמים.

<u>תאור גרפי</u>:



<u>מימוש</u>:
האיבר בשורש משמש מציין לקבוצה

size	2				1	4	
parent	--	6	1	6	--	--	2
elements	1	2	3	4	5	6	7

Makeset(i) – ממומשת ע"י יצירת עץ בעל צומת יחיד i. <u>זמן</u>: $O(1)$.

Find(i) ממומש ע"י סיור במעלה העץ. <u>זמן</u>: $O(h)$ כאשר h הוא גובה העץ.

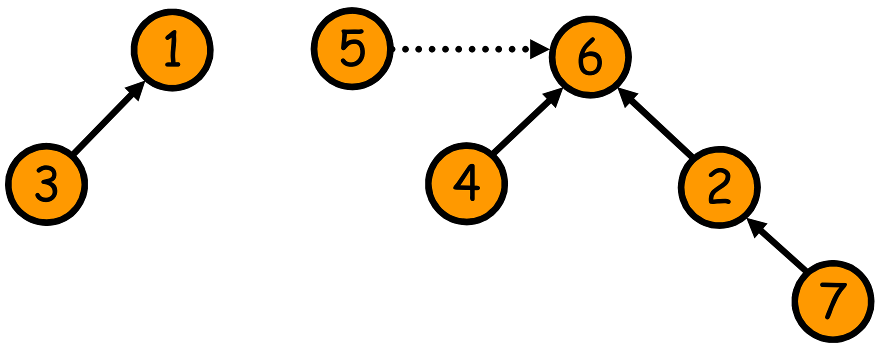
Union(p,q) ממומש ע"י איחוד העצים ששורשיהן p ו- q כך ששורש העץ הקטן יופנה לשורש העץ הגדול. <u>זמן</u>: $O(1)$.

דוגמא

מצב התחלתי (כמו בשקף הקודם):

size	2				1	4	
parent	--	6	1	6	--	--	2
elements	1	2	3	4	5	6	7

לאחר (Union(5,6:



size	2					5	
parent	--	6	1	6	6	--	2
elements	1	2	3	4	5	6	7

זמן לפעולת Union הוא $O(1)$.

תועלת איחוד לפי גודל - ניתוח גובה העץ

טענה: עבור יער של עצים הפוכים בן n צמתים בעל גובה h שנבנה מאיחודים של קבוצות קטנות לתוך גדולות, מתקיים $h \le \log_2 n$.

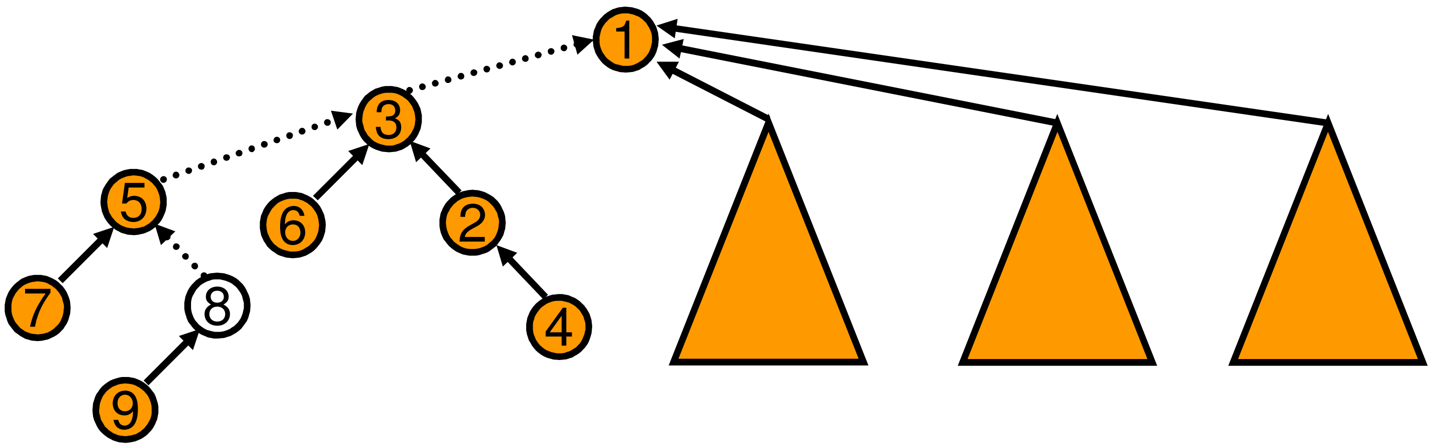
הוכחה:
ניזכר בטענה א עבור ייצוג קבוצות באמצעות רשימות ואיחודן לפי גודל:
אם בכל איחוד מוסיפים את הקבוצה הקטנה לגדולה אזי כל איבר x משנה את הקבוצה אליה הוא שייך לכל היותר $\log_2 n$ פעמים כאשר n הוא מספר האיברים במבנה.

כיוון שכל שינוי קבוצה גורר הגדלת מרחק האיבר משורש קבוצתו ב-1 וכל איבר מתחיל במרחק 0 משורש קבוצתו (כל איבר מתחיל כקבוצה של איבר בודד), נקבל כי מרחק כל צומת משורש קבוצתו הוא לכל היותר $\log_2 n$.

מסקנה: זמן פעולת Find הוא $O(\log n)$ במקרה הגרוע ביותר כאשר n הוא מספר הצמתים הכללי בכל העצים.

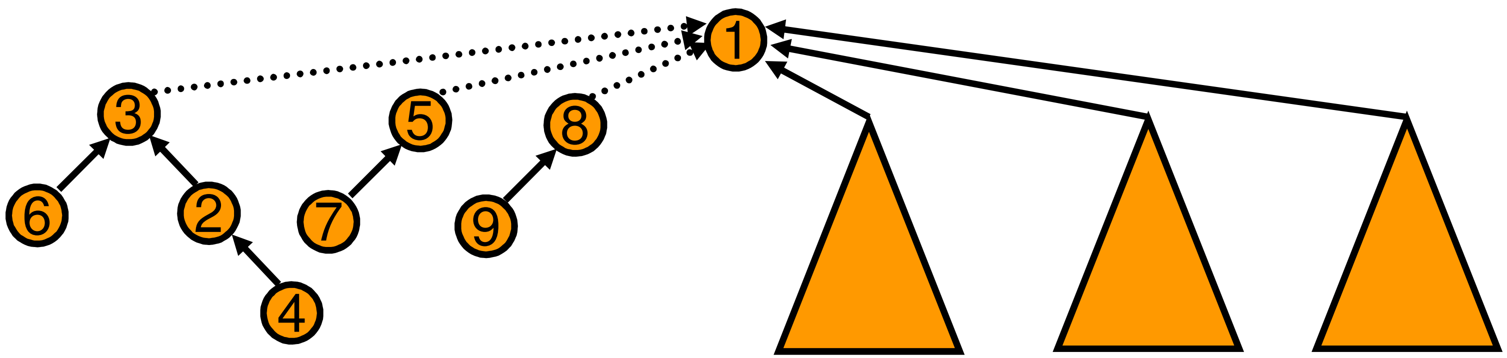
שיפור נוסף: כווץ מסלולים

בזמן ביצוע $\text{Find}(i)$, עדכן את שדה ה-parent של כל הצמתים מ-i ועד השורש כך שיצביעו ישירות לשורש. <u>דוגמא:</u> $\text{Find}(8)$.



המימוש דורש שני סיורים עד השורש: בראשון למציאת השורש. בשני לעדכון המצביעים לכוון השורש. מכאן שהסיבוכיות של Find במקרה הגרוע נשארת $O(h) = O(\log n)$.

<u>תהליך העדכון:</u>



ניתוח זמנים

<u>משפט</u>: במימוש באמצעות אוסף עצים של n איברים עם איחוד לפי גודל וכיווץ מסלולים, סיבוכיות זמן בצוע m פעולות union/find חסומה ע"י $O(m \log^* n)$. לפיכך הזמן המשוערך (amortized time) לכל פעולה חסום ע"י $O(\log^* n)$.

כאשר הפונקציה $\log^* n$ מוגדרת כדלהלן:

$$\log^{(i)}(n) = \overbrace{\log_2 \log_2 \dots \log_2}^{i} n$$

$$\log^*(n) = \min\{i \geq 0 \mid \log^{(i)}(n) \leq 1\}$$

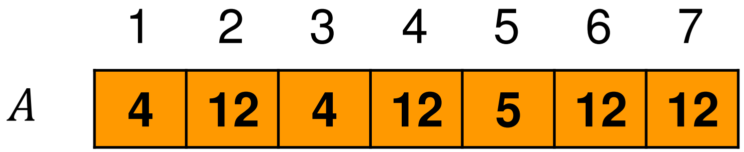
ובמילים, $\log^* n$ הוא מספר הפעמים שצריך לקחת $\log$ כדי לקבל מספר קטן או שווה לאחד. זוהי פונקציה מונוטונית העולה בקצב מאד איטי. לכל מספר n פרקטי $\log^* n \leq 5$. כלומר פעולת union/find לוקחות זמן משוערך שהוא קבוע מבחינה מעשית. לשם המחשה:

$$\log^*(1) = 0 \quad \log^*(2) = 1 \quad \log^*(2^2) = 2 \quad \log^*(2^{2^2}) = 3 \quad \log^*(2^{2^{2^2}}) = 4$$

$$\log^*\left(2^{2^{2^{2^2}}}\right) = \log^*(2^{65536}) = 5$$

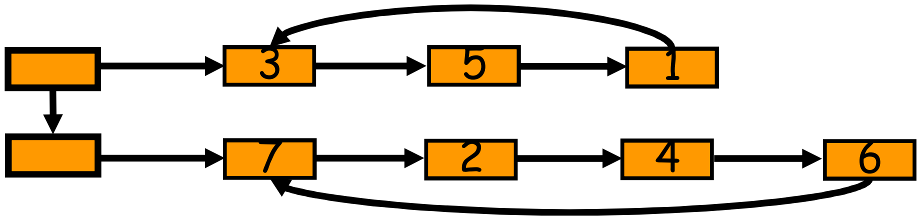
הוכחת המשפט בחוברת. הוכחה אחרת, עם חסם הדוק עוד יותר באמצעות פונקציות אקרמן ניתן למצוא בספר הלימוד בעמודים 450-453.

סיבוכיות זמן המימושים - סיכום



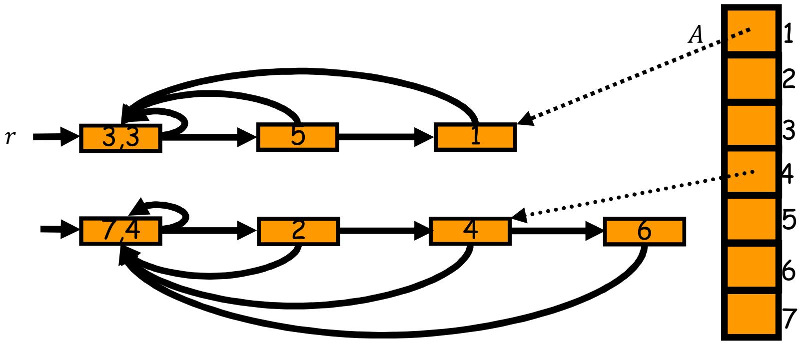
<u>במימוש הנאיבי הראשון:</u>

1. Makeset ב- $O(1)$ זמן.
2. Find ב- $O(1)$ זמן.
3. Union ב- $O(n)$ זמן.



<u>במימוש הנאיבי השני:</u>

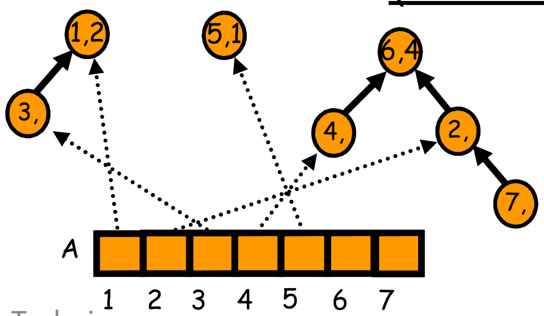
1. Makeset ב- $O(1)$ זמן.
2. Find ב- $O(n)$ זמן.
3. Union ב- $O(1)$ זמן.



<u>במימוש השלישי המשופר (עם איחוד לפי גודל):</u>

1. Makeset ב- $O(1)$ זמן.
2. Find ב- $O(1)$ זמן.
3. Union ב- $O(\log n)$ זמן משוערך.

<u>במימוש הרביעי (עצים הפוכים עם איחוד לפי גודל וכיווץ מסלולים)</u>



1. Makeset ב- $O(1)$ זמן.
2. Find ב- $O(\log n)$ זמן.
3. Union ב- $O(1)$ זמן.

הסיבוכיות המשוערכת של Union ו-Find היא $O(\log^* n)$.

דוגמא – כדורים בשני צבעים

<u>אביב 2010 מועד ב – שאלה 3 (25 נקודות)</u>

בחדר חשוך ישנם n כדורים אשר ממוספרים $1,2,\dots,n$.
כל כדור צבוע בצבע מסוים, אך לא ניתן לראות מה צבעו של כל כדור.
ידוע כי יש <u>בדיוק</u> שני צבעים אפשריים.

תחילה, לא ידוע מהו צבעו של כל כדור.

ממשו מבנה התומך בפעולות הבאות:

Init(n) החדר מאותחל עם n כדורים שצבעם לא ידוע.

Same(i,j) נאמר לכם כי הכדורים i,j צבועים באותו צבע.

Different(i,j) נאמר לכם כי הכדורים i,j צבועים בצבעים שונים.

AreSameColor(i,j) עונה על השאלה: האם הכדורים i,j צבועים באותו צבע?
אם כן, הפונקציה מחזירה TRUE.
אם לא, הפונקציה מחזירה FALSE.
אם לא ניתן להכריע האם הכדורים צבועים באותו צבע או לא, הפונקציה מחזירה UNKNOWN.

כדורים בשני צבעים (המשך)

סעיף א' (3 נקודות)

עבור $n = 5$ הופעלו הפעולות הבאות:

Init(5)
Different(1,2)
Different(3,4)
AreSameColor(1,3) ___UNKNOWN___.
Same(1,3)
AreSameColor(1,3) ___TRUE___.
AreSameColor(2,4) ___TRUE___.
Different(4,5)
AreSameColor(3,5) ___TRUE___.

מה יהיה פלט האלגוריתם בשורה הריקה?

פתרון:

תחילה, הסבר לתשובה השלישית לAreSameColor:

$$color(1) \neq color(2), \qquad color(3) \neq color(4), \qquad color(1) = color(3)$$

וכיוון שיש רק שני צבעים שונים, נסיק כי $color(2) = color(4)$.

הסבר לתשובה האחרונה לAreSameColor:

כיוון שמתקיים $\mathrm{color}(3) \neq color(4) \neq color(5)$ נקבל כי $color(3) = color(5)$.

כדורים בשני צבעים (המשך)

<u>סעיף ב' (22 נקודות)</u>
הציעו מבנה אשר מממש את פעולת Init בסיבוכיות זמן $O(n)$ ואת שלושת הפעולות האחרות (AreSame, AreDifferent, AreSameColor) בסיבוכיות זמן משוערכת $O(\log^* n)$.

<u>דיון מקדים:</u>

נזכיר כי הפעולות המותרות הן:

Init(n) החדר מאותחל עם n כדורים שצבעם לא ידוע.

Same(i,j) נאמר לכם כי הכדורים i,j צבועים באותו צבע.

Different(i,j) נאמר לכם כי הכדורים i,j צבועים בצבעים שונים.

AreSameColor(i,j) עונה על השאלה: האם הכדורים i,j צבועים באותו צבע?

אם כן, הפונקציה מחזירה TRUE.

אם לא, הפונקציה מחזירה FALSE.

אם לא ניתן להכריע האם הכדורים צבועים באותו צבע או לא,

הפונקציה מחזירה UNKNOWN.

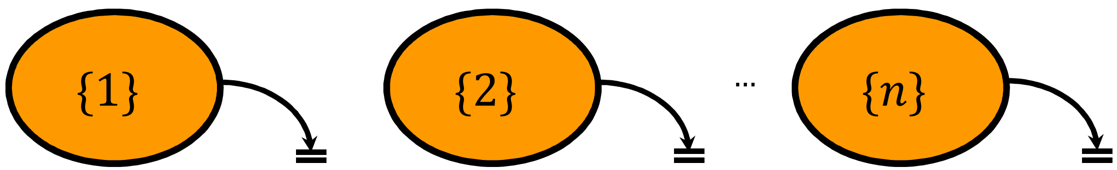
<u>נבחין:</u> פרט לפעולה Different ניתן לפתור את הבעיה בעזרת Union-Find, כאשר הקבוצות יהיו כדורים שידוע שהם מאותו צבע.
בכדי לטפל בפונקציה הנוספת הזו נזכיר שהבחנו כי אם מתקיים $color(i) \neq color(j)$ וגם $color(i) \neq color(k)$ אזי $color(j) = color(k)$.

כדורים בשני צבעים (המשך)

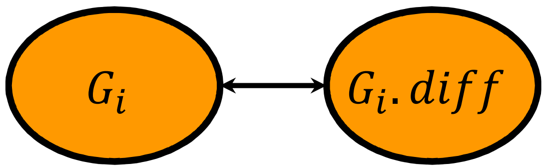
סעיף ב' (22 נקודות)
הציעו מבנה אשר מממש את פעולת Init בסיבוכיות זמן $O(n)$ ואת שלושת הפעולות האחרות (AreSame, AreDifferent, AreSameColor) בסיבוכיות זמן משוערכת $O(\log^* n)$.

פתרון:
הרעיון: נחזיק קבוצות זרות של כדורים שידוע שהם באותו צבע, כשכל קבוצה תשמור מצביע לקבוצה של כדורים שידוע שהם מהצבע השני. למצביע זה נקרא $diff$.

Init(n) נבצע makeset(i) לכל כדור i, עם $diff = NULL$.



AreSameColor(i, j) נמצא את G_i ו- G_j הקבוצות של i ו- j בהתאמה.



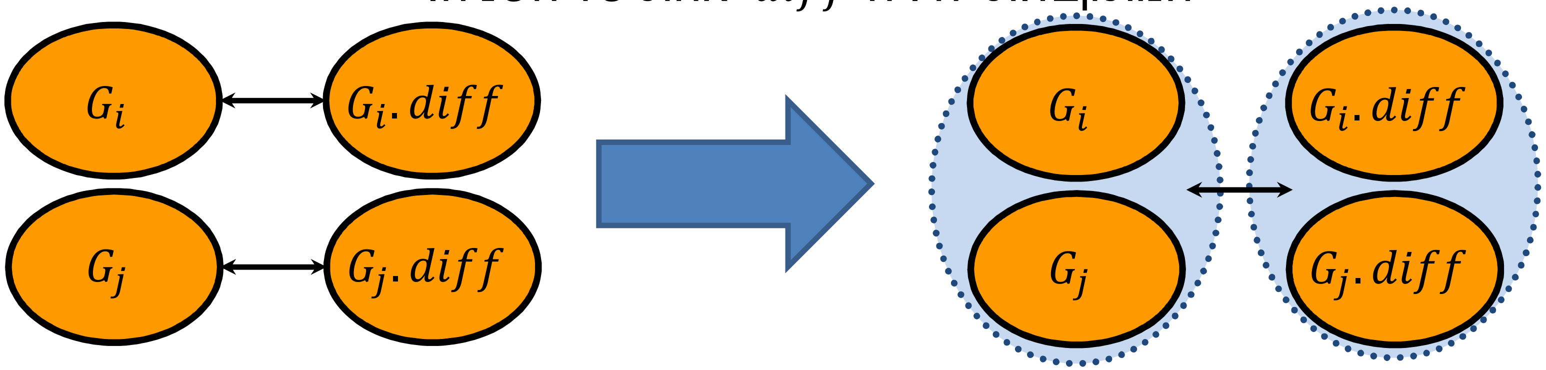
אם $G_i = G_j$ נענה "TRUE".
אם $G_i = G_j.diff$ נענה "FALSE".
אחרת, נענה "UNKNOWN".

כדורים בשני צבעים (המשך)

<u>עדכונים</u>

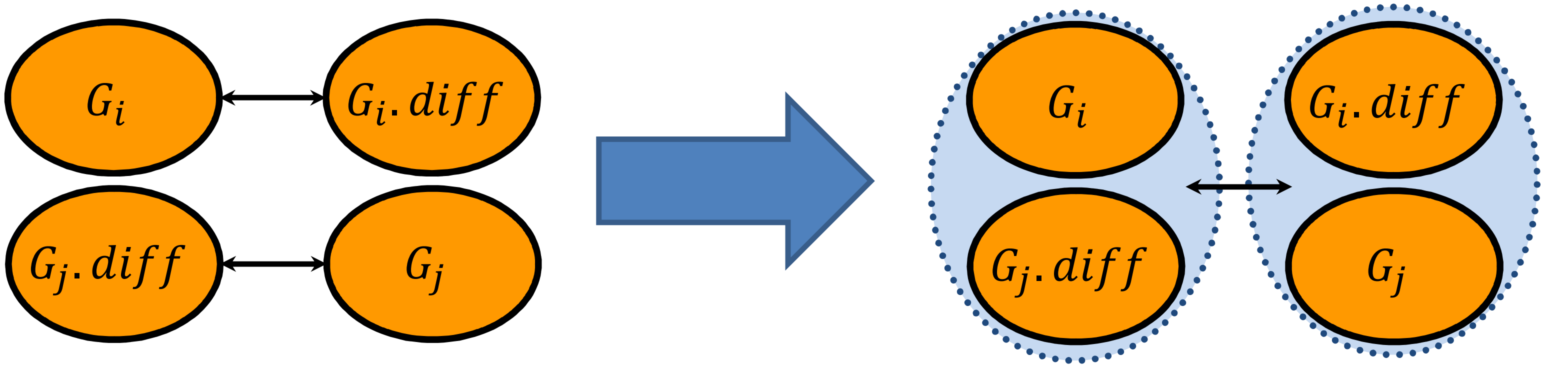
AreSame(i,j)

נמצא את G_i ו- G_j הקבוצות של i ו- j בהתאמה.
נאחד את G_i ו- G_j ונאחד את $G_i.diff$ ו- $G_j.diff$. שתי הקבוצות המתקבלות יהיו ה-$diff$ אחת של השניה.



AreDifferent(i,j)

נמצא את G_i ו- G_j הקבוצות של i ו- j בהתאמה.
נאחד את G_i ו- $G_j.diff$ ונאחד את $G_i.diff$ ו- G_j. שתי הקבוצות המתקבלות יהיו ה-$diff$ אחת של השניה.



כדורים בשני צבעים (המשך)

<u>ניתוח סיבוכיות:</u>
באתחול נבצע n פעולות Makeset ולכל כדור נבצע $O(1)$ עבודה נוספת. <u>סה"כ:</u> $O(n)$.

בשאר הפעולות נבצע מספר קבוע של פעולות Union/Find ועוד $O(1)$ עבודה.
<u>סה"כ:</u> עלות m פעולות כנ"ל היא $O(m)+O(m\log^* n)=O(m\log^* n)$ והעלות המשוערכת של פעולות אלו היא אכן $O(\log^* n)$, כנדרש.

<u>נימוק נכונות:</u>
ניתן להראות באינדוקציה על מספר פעולות המבנה כי לכל אורך הריצה אנו מוודאים כי כל כדור שייך לקבוצת הכדורים שידוע שהם מצבעו, וכל קבוצה כזו מחזיקה מצביע לקבוצת הכדורים שידוע כי הם מהצבע השני. לכן בעת ביצוע שאילתא AreSameColor(i,j) נענה נכונה.