איסוף אשפה (Garbage collection)

כותרת חילופית: ניהול זיכרון דינמי (Dynamic memory allocation)

ספר בתחום: Garbage Collection: Algorithms for automatic dynamic memory management, Richard Jones and Rafael Lins, Wiley, 1999.

כמה זיכרון צורכת התוכנית הבאה ?

```
void insert(int x){}
```

```
void delete(int x){
   for (p = s; p → next → value ≠ x;  p = p → next);
   p → next = p → next → next ;
}
```

```
void main( ){
   for (i=0;  i < 5;  i++)
         insert(i);
   for (i=5;  i<1000;  i++)
         insert(i);
         delete(i-5);
}
```

©Geiger & Itai, 2001

איסוף אשפה (Garbage collection)

כותרת חילופית: ניהול זיכרון דינמי (Dynamic memory allocation)

ספר בתחום: Garbage Collection: Algorithms for automatic dynamic memory management, Richard Jones and Rafael Lins, Wiley, 1999.

כמה זיכרון צורכת התוכנית הבאה ?

```
void insert(int x){}
```

```
void delete(int x){
   for (p = s; p → next → value ≠ x;  p = p → next);
   p → next = p → next → next ;
}
```

```
void main( ){
   for (i=0;  i < 5;  i++)
         insert(i);
   for (i=5;  i<1000;  i++)
         insert(i);
         delete(i-5);
}
```

©Geiger & Itai, 2001

איסוף אשפה (Garbage collection)

כותרת חילופית: ניהול זיכרון דינמי (Dynamic memory allocation)

ספר בתחום: Garbage Collection: Algorithms for automatic dynamic memory management, Richard Jones and Rafael Lins, Wiley, 1999.

כמה זיכרון צורכת התוכנית הבאה ?

```
void insert(int x){}
```

```
void delete(int x){
   for (p = s; p → next → value ≠ x;  p = p → next);
   p → next = p → next → next ;
}
```

```
void main( ){
   for (i=0;  i < 5;  i++)
         insert(i);
   for (i=5;  i<1000;  i++)
         insert(i);
         delete(i-5);
}
```

©Geiger & Itai, 2001

איסוף אשפה (Garbage collection)

כותרת חילופית: ניהול זיכרון דינמי (Dynamic memory allocation)

ספר בתחום: Garbage Collection: Algorithms for automatic dynamic memory management, Richard Jones and Rafael Lins, Wiley, 1999.

כמה זיכרון צורכת התוכנית הבאה ?

```
void insert(int x){}
```

```
void delete(int x){
   for (p = s; p → next → value ≠ x;  p = p → next);
   p → next = p → next → next ;
}
```

```
void main( ){
   for (i=0;  i < 5;  i++)
         insert(i);
   for (i=5;  i<1000;  i++)
         insert(i);
         delete(i-5);
}
```

©Geiger & Itai, 2001

איסוף אשפה

הבעיה: איך לשחרר צמתים שהמתכנת לא שחרר באמצעות פקודת free.

חשיבות הבעיה

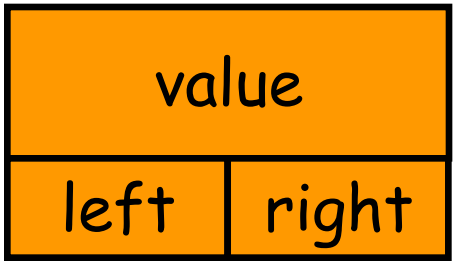
- בתוכניות גדולות עם הרבה מודולים והרבה אובייקטים (בניגוד לתוכניות שראינו בקורס) קשה למתכנתים לדעת מתי באמת משתחרר אובייקט.
- כאשר מיוצרת אשפה (Memory leak) קשה לאתר את התקלה, כלומר ה- debugging קשה.

©cs,Technion

מודל פשוט לשימוש בזיכרון

- הזכרון מאורגן ממערך mem[m] שבו מוקצים צמתים (אובייקטים) מהצורה :

```
typedef struct node {
    DATA  value;
    node *left *right;
} NODE;
```



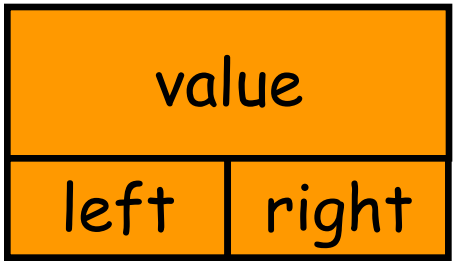
(צמתים מורכבים יותר ושוני גודל אפשריים, אך אנו נתרכז במודל פשוט).

- קיימת רשימה AVAIL של צמתים לא מנוצלים. כל פעם שעושים malloc מקבלים צומת מרשימה זו וכשעושים free מחזירים צומת לרשימה AVAIL.

מודל פשוט לשימוש בזיכרון

- הזכרון מאורגן ממערך mem[m] שבו מוקצים צמתים (אובייקטים) מהצורה :

```
typedef struct node {
    DATA  value;
    node *left *right;
} NODE;
```



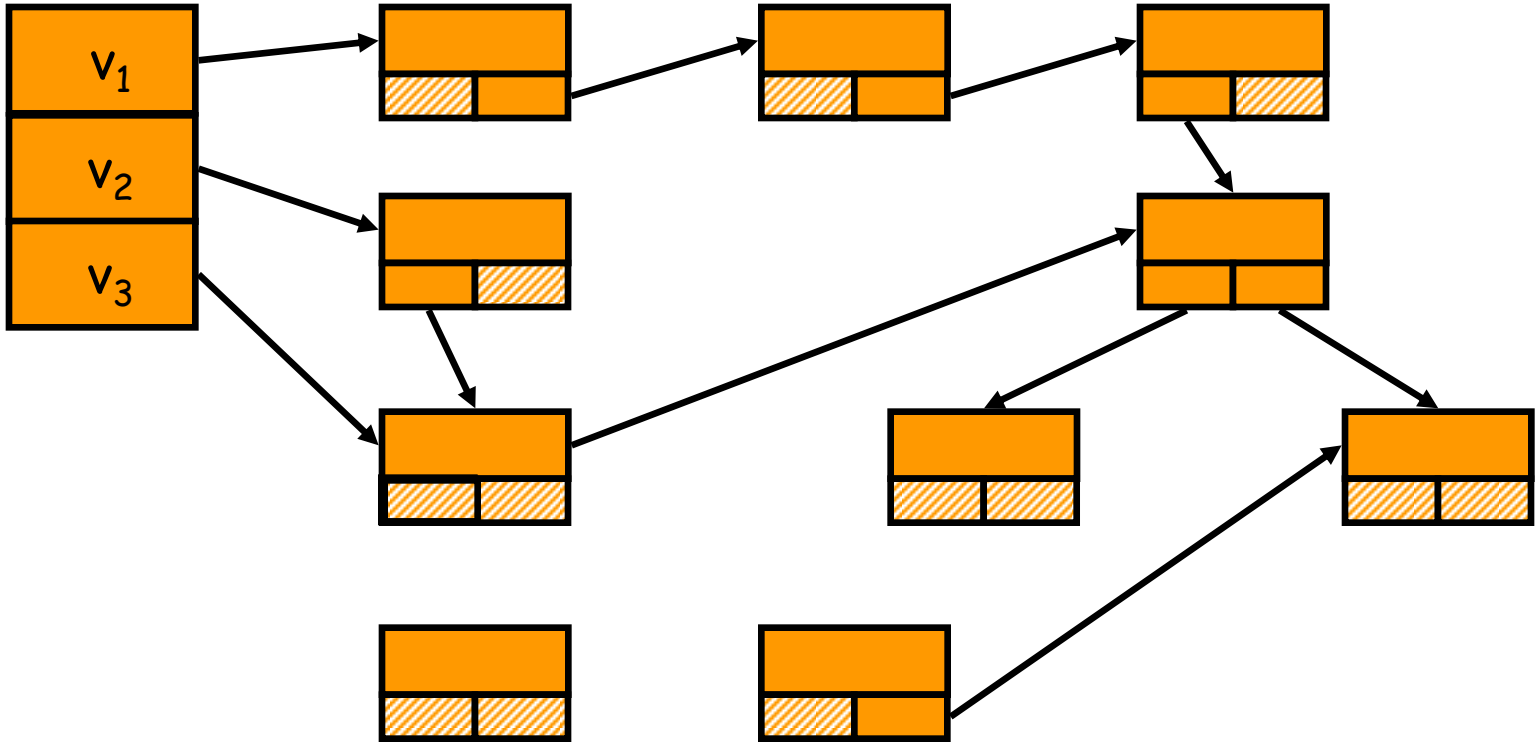
(צמתים מורכבים יותר ושוני גודל אפשריים, אך אנו נתרכז במודל פשוט).

- קיימת רשימה AVAIL של צמתים לא מנוצלים. כל פעם שעושים malloc מקבלים צומת מרשימה זו וכשעושים free מחזירים צומת לרשימה AVAIL.

הזיכרון מחולק פונקציונלית ל"ערימת אובייקטים" לשמירת אובייקטים המיוצרים באופן דינמי ול"מחסנית ריצה" המכילה את "משתני התוכנית".

מודל לשימוש בזיכרון (המשך)

- ניתן לגשת לצמתים בזכרון רק דרך "משתני התוכנית" $v_1,\ldots,v_n$ הנמצאים במחסנית הריצה.



<u>אשפה</u>: צמתים מנותקים, כאלה שאין אליהם מסלול מאף v_i.

כיצד נוצרת אשפה ?

נתאר שלושה פתרונות לבעיית האשפה: ספירה, סימון, ודחיסה.

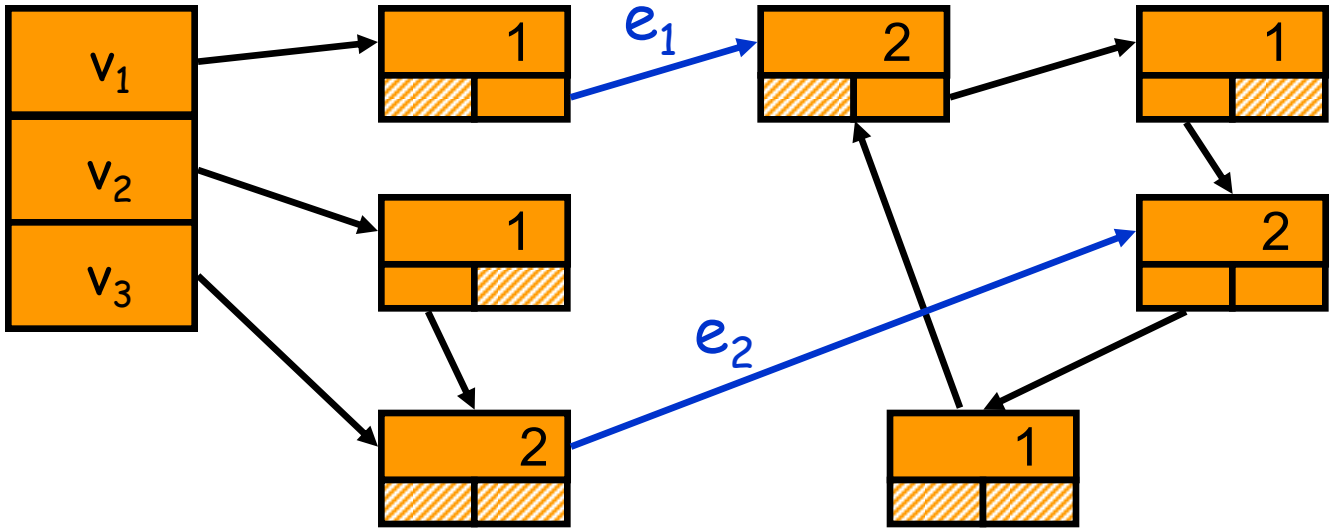
©cs,Technion

פתרון ע"י ספירה (reference counting)

הוסף לכל צומת V שדה count השומר את מספר המצביעים אליו.

עדכן את V.count בכל פעם שמשנים מצביע לצומת V.

כאשר עבור צומת V מתקבל V.count=0 הוסף את V לרשימת הצמתים הלא-מנוצלים AVAIL ועדכן את count לצמתים אליהם V מצביע.

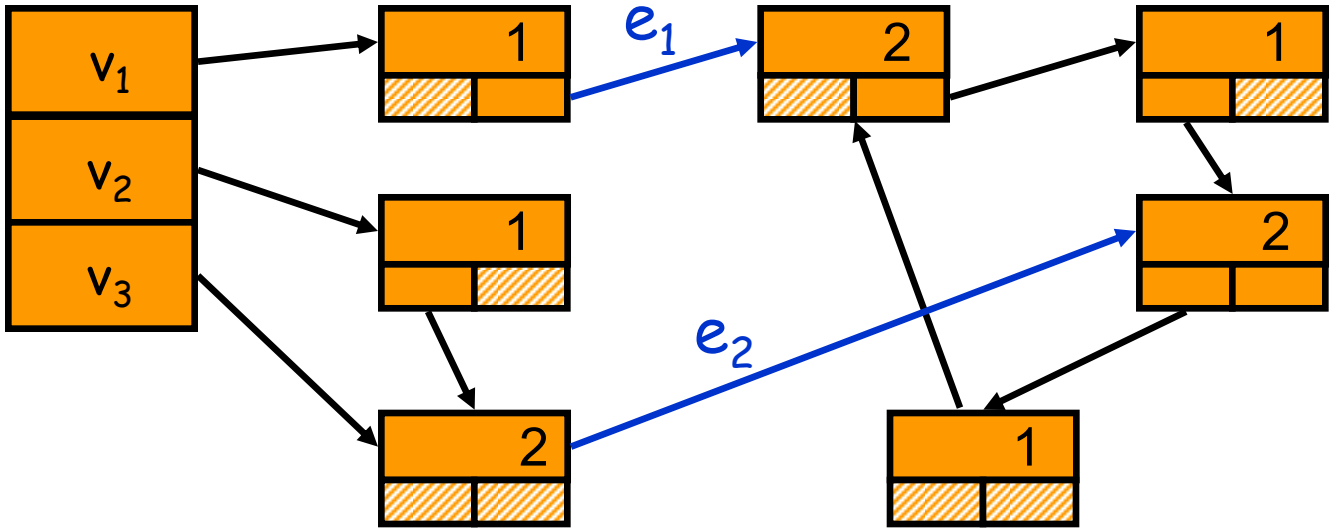


פתרון ע"י ספירה (reference counting)

הוסף לכל צומת V שדה count השומר את מספר המצביעים אליו.

עדכן את V.count בכל פעם שמשנים מצביע לצומת V.

כאשר עבור צומת V מתקבל V.count=0 הוסף את V לרשימת הצמתים הלא-מנוצלים AVAIL ועדכן את count לצמתים אליהם V מצביע.



בעיה: מעגלים תמיד מכילים צמתים עם count חיובי אפילו אם אין גישה למעגל כולו. למשל אם נבטל קשת e_1 וקשת e_2 נקבל מעגל ללא גישה אליו, כלומר אשפה.

©cs,Technion

פתרון ע"י סימון (tracing collector)

הרעיון: כאשר הרשימה AVAIL מתרוקנת, עוברים על הזיכרון כולו ומסמנים את כל הצמתים אליהם יש גישה ממשתני התוכנית. עוברים שנית על הזיכרון ומכניסים לרשימה AVAIL את כל הצמתים שלא סומנו. כדי לאפשר את סימון הצמתים יש להוסיף שדה בולאני label לכל צומת.

האלגוריתם

- עבור על כל צמתי הזיכרון וקבע label=0.
- סמן את הצמתים הקשורים ישירות למשתני התוכנית ע"י label = 1.
- אם סימנת צומת w של השמאלי והימני כדלקמן:

```
void mark(NODE *w){
  if(w!=NULL && w→label = = 0) {
    w→label = 1;
    mark(w→left);
    mark(w→right); }
}
```

פתרון ע"י סימון (tracing collector)

הרעיון: כאשר הרשימה AVAIL מתרוקנת, עוברים על הזיכרון כולו ומסמנים את כל הצמתים אליהם יש גישה ממשתני התוכנית. עוברים שנית על הזיכרון ומכניסים לרשימה AVAIL את כל הצמתים שלא סומנו. כדי לאפשר את סימון הצמתים יש להוסיף שדה בולאני label לכל צומת.

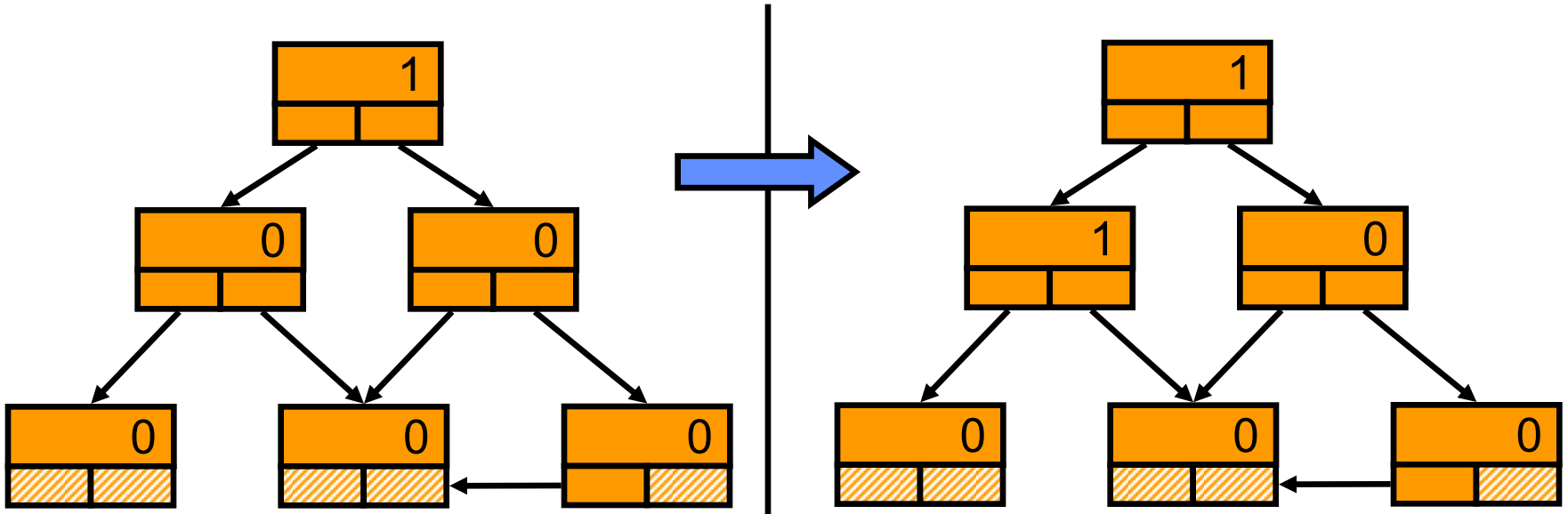
האלגוריתם

- עבור על כל צמתי הזיכרון וקבע label=0.
- סמן את הצמתים הקשורים ישירות למשתני התוכנית ע"י label = 1 .
- אם סימנת צומת w של השמאלי והימני כדלקמן:

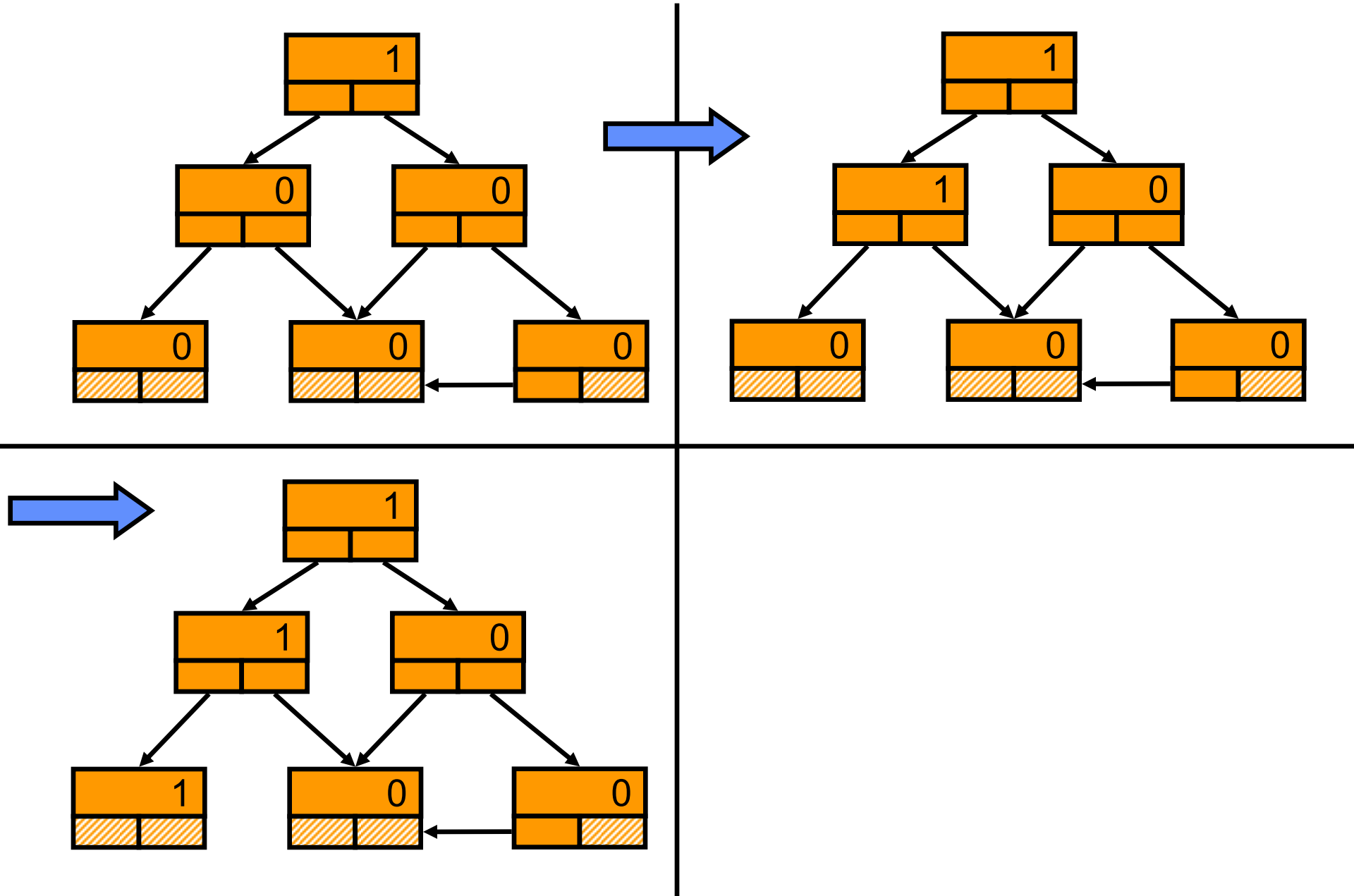
```
void mark(NODE *w){
  if(w!=NULL && w→label = = 0) {
    w→label = 1;
    mark(w→left);
    mark(w→right); }
}
```

הערה: זהו אלגוריתם preorder עבור עצים. אבל הוא מסמן גרפים מכוונים כלשהם.

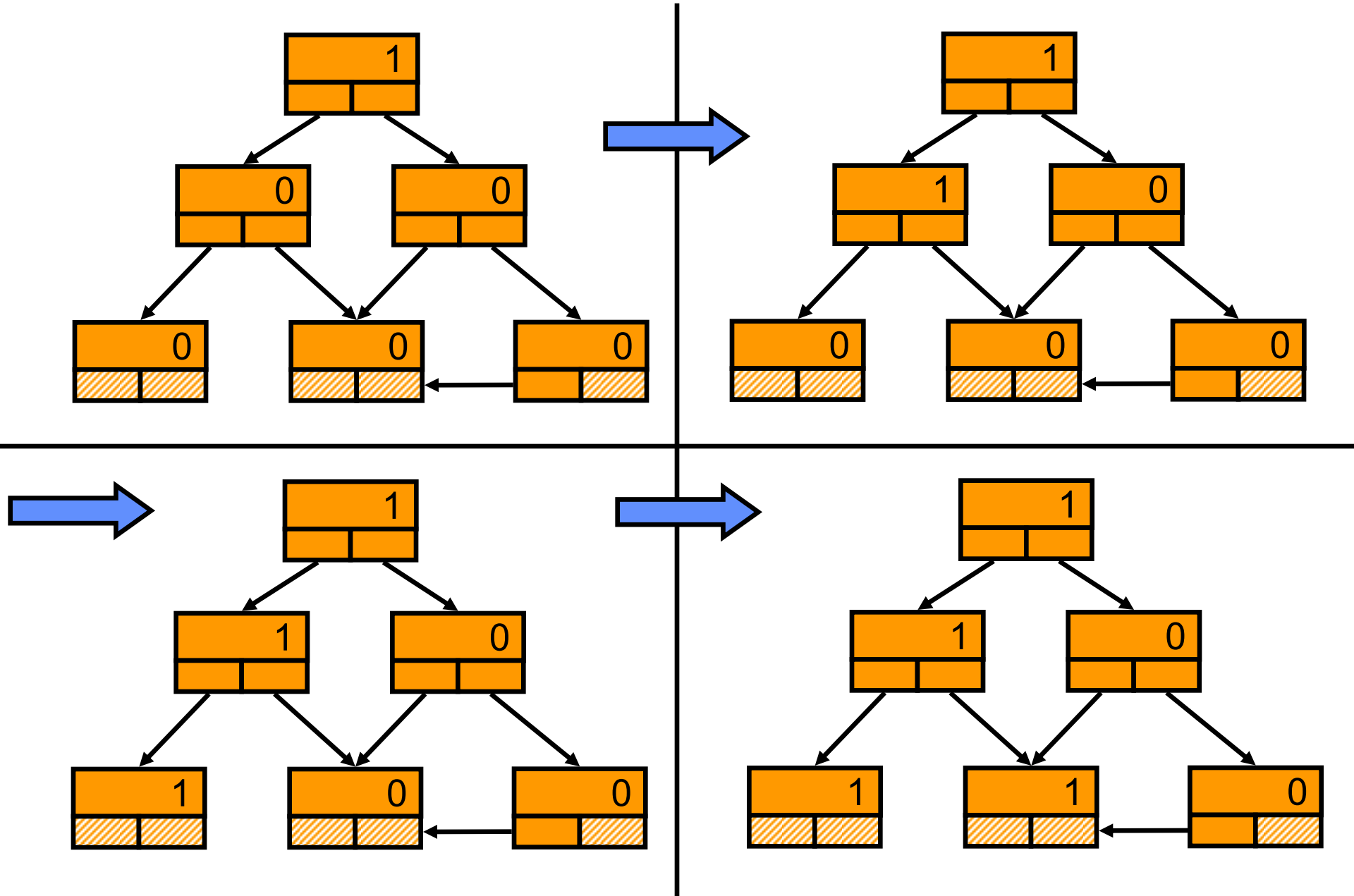
דוגמא על גרף כללי



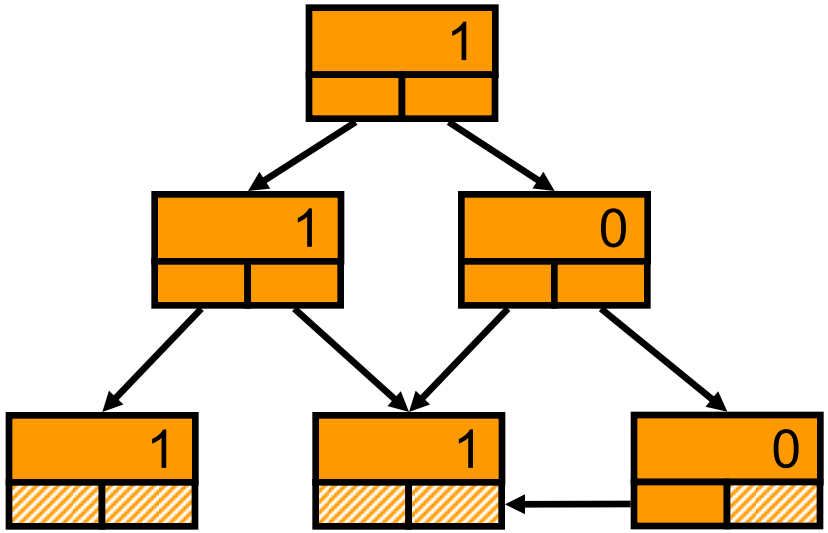
דוגמא על גרף כללי



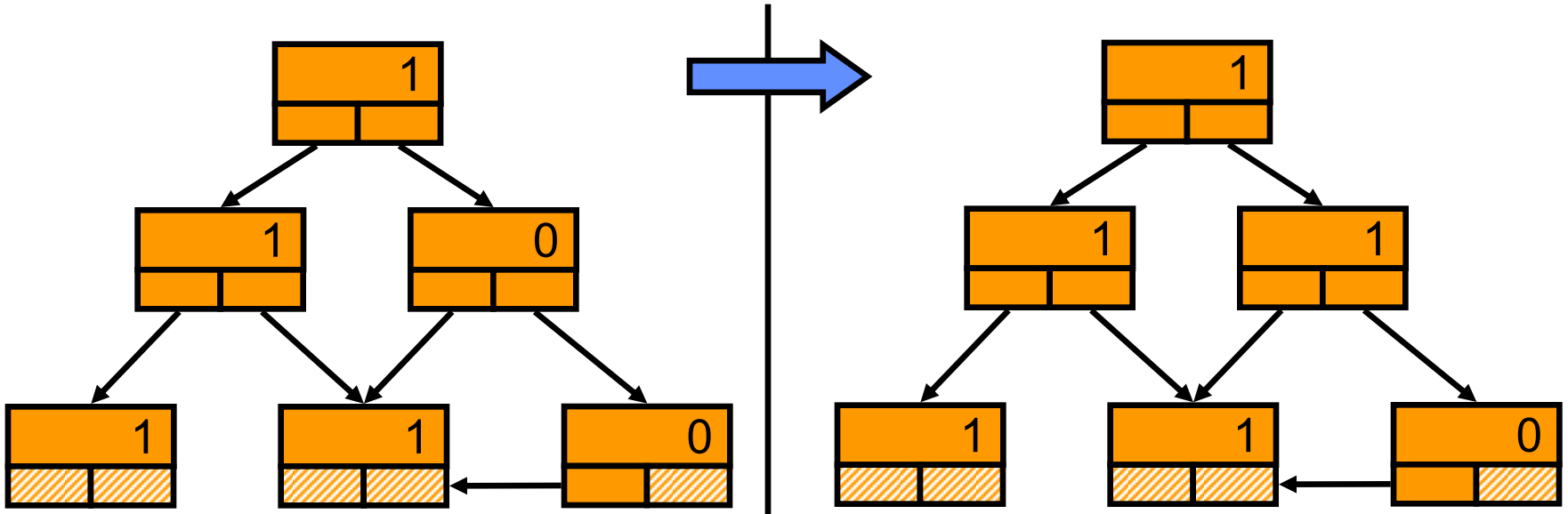
דוגמא על גרף כללי



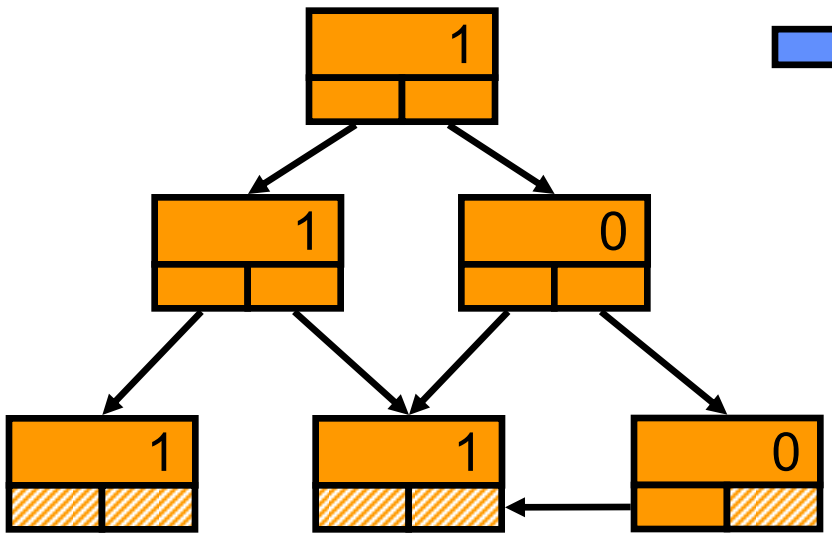
דוגמא על גרף כללי

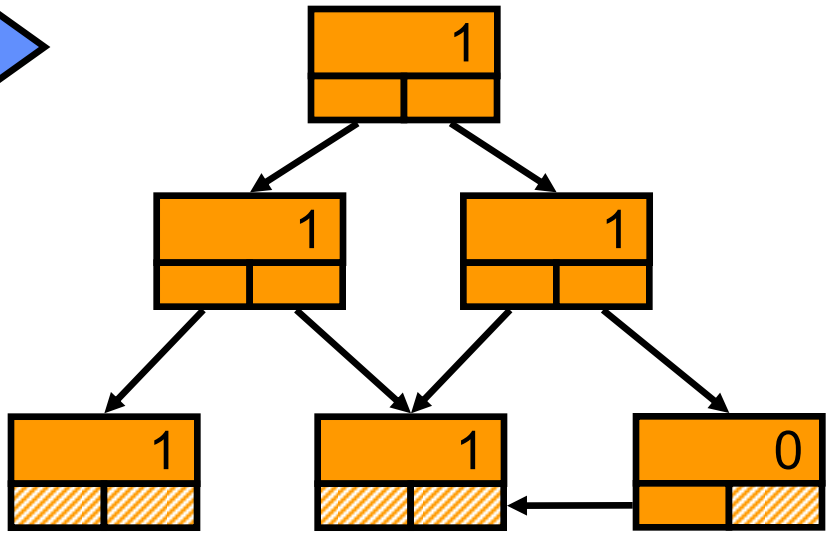


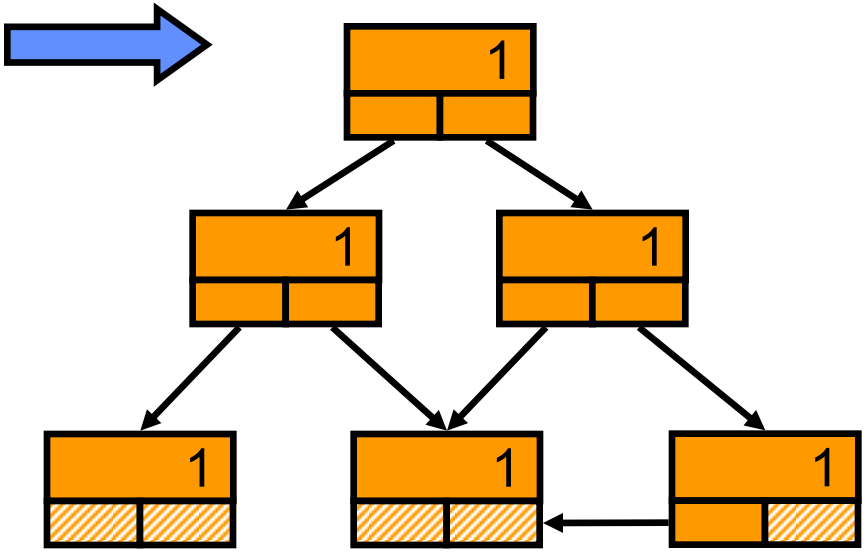
דוגמא על גרף כללי



דוגמא על גרף כללי





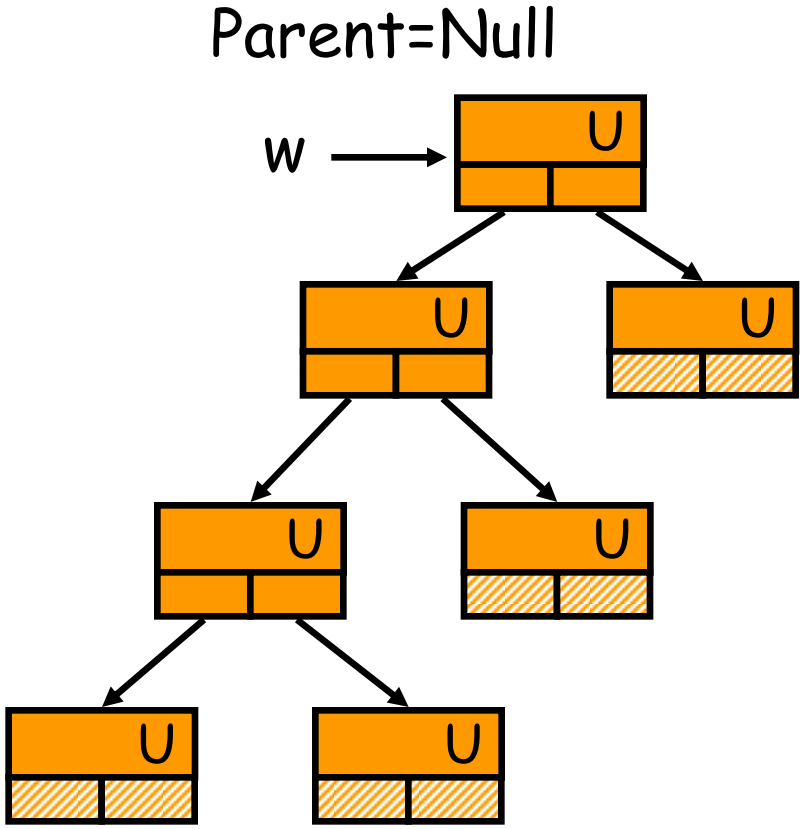


הערות:

- זהו אלגוריתם Depth First Search לסיור בגרפים מכוונים.
- גם סימון וגם ספירה ישימים כאשר לצמתי הזיכרון יותר משני בנים.

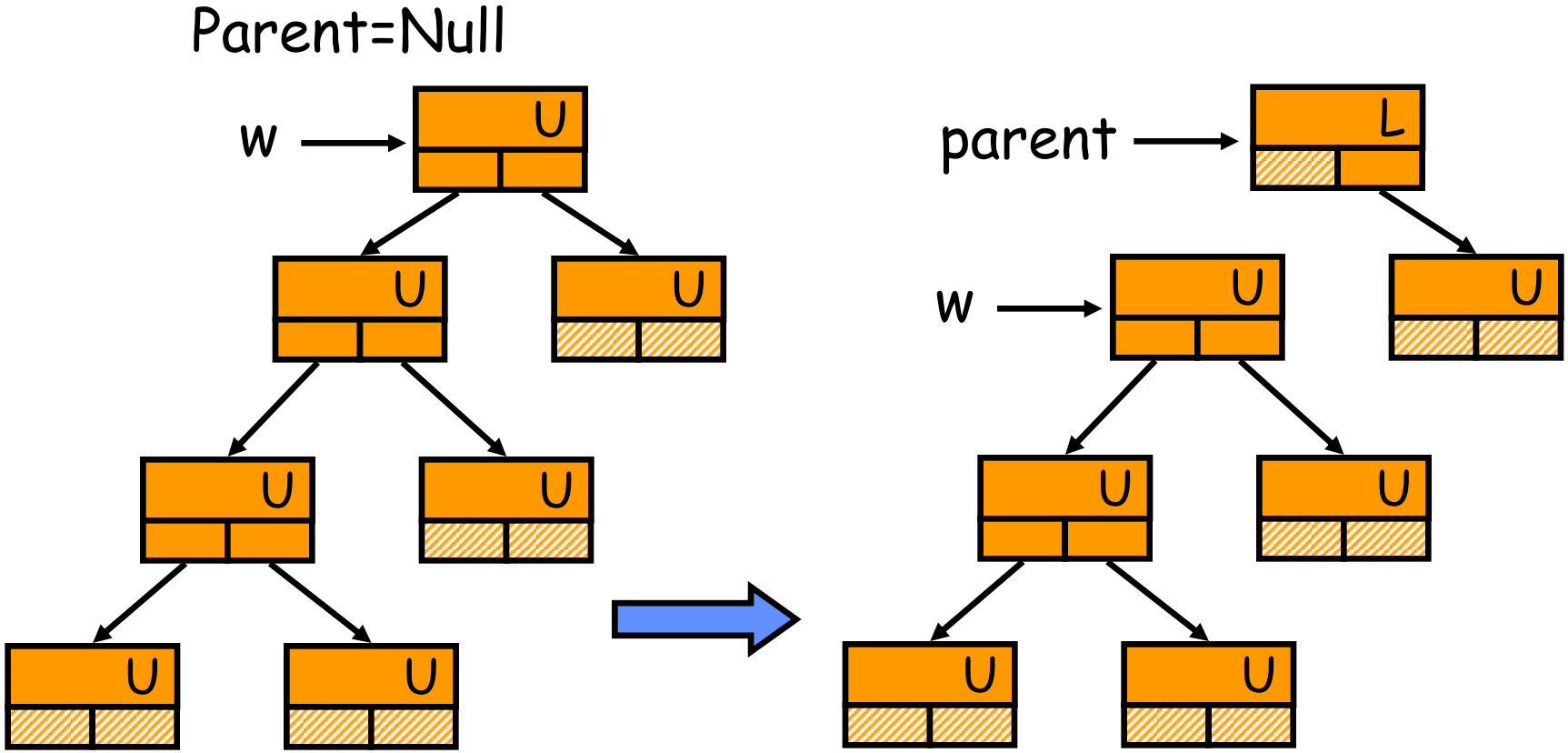
סימון ללא רקורסיה

הרעיון: כאשר נבקר בצומת, נשמור את המכוון להורה שלו בשדה left או right כך שנוכל לחזור להורה בסוף הביקור. לתווית של כל צומת יהיו שלושה ערכים: {u,r,l} שיאפשרו לדעת איזה מכוון מצביע להורה. נשתמש במשתנים הבאים: root עבור שורש העץ ו-w עבור הצומת הנוכחי (מאותחל ל-root). השטה נקראת הפוך מצביעים(Pointer Reversal) : פרק 4.3 בספר (Garbage Collection, Jones & Lins, 96).

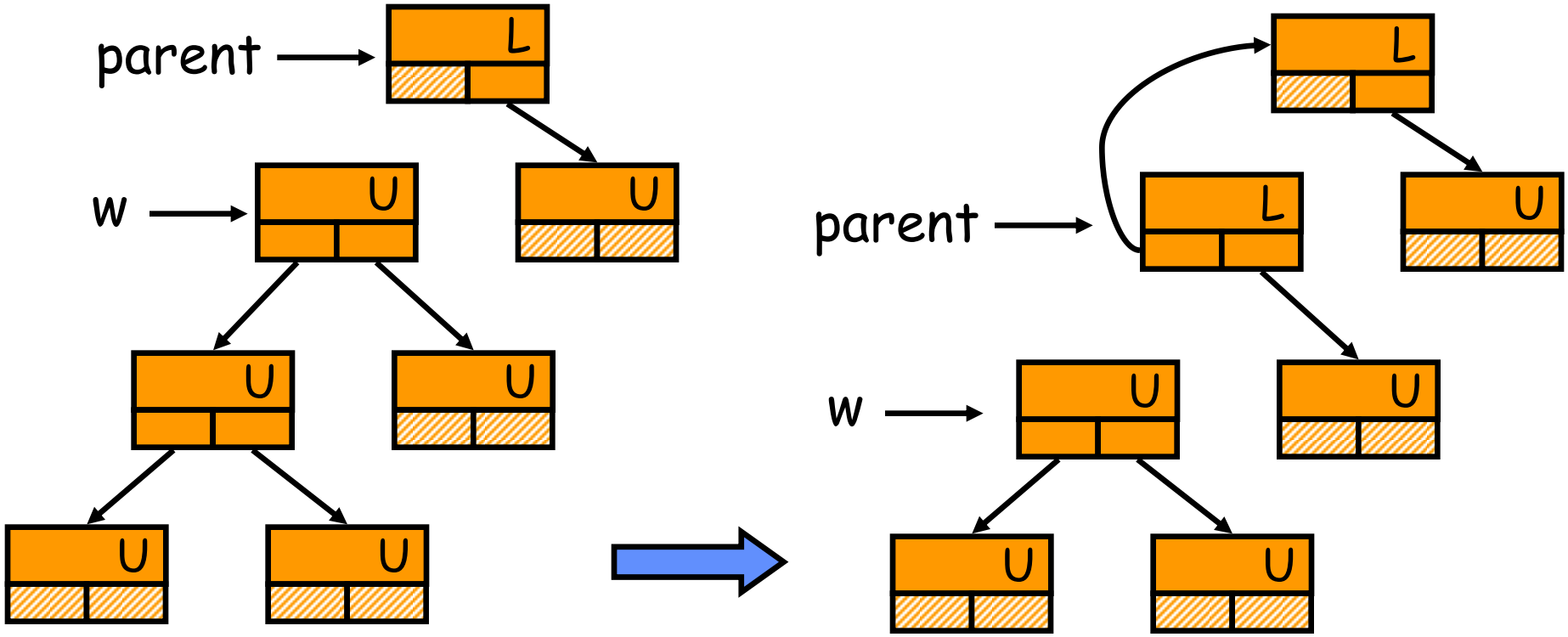


סימון ללא רקורסיה

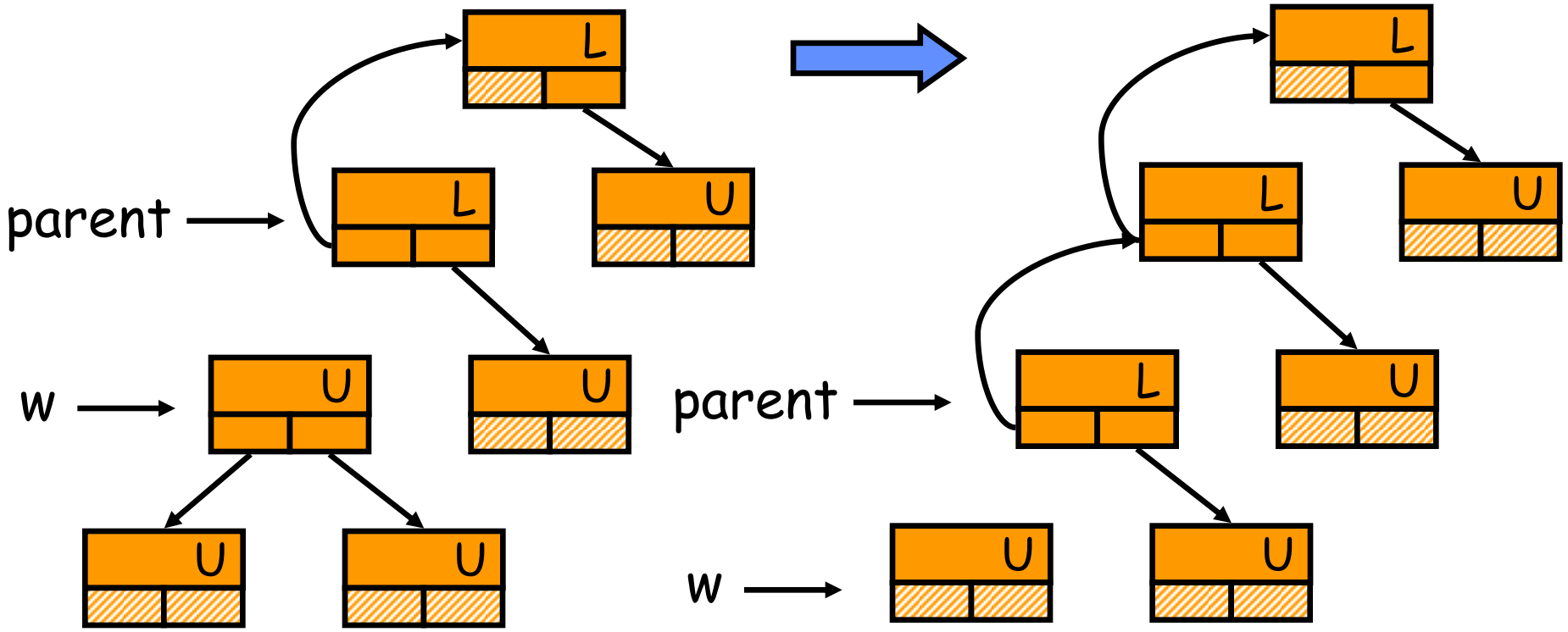
הרעיון: כאשר נבקר בצומת, נשמור את המכוון להורה שלו בשדה left או right כך שנוכל לחזור להורה בסוף הביקור. לתווית של כל צומת יהיו שלושה ערכים: {u,r,l} שיאפשרו לדעת איזה מכוון מצביע להורה. נשתמש במשתנים הבאים: root עבור שורש העץ ו-w עבור הצומת הנוכחי (מאותחל ל-root). השטה נקראת הפוך מצביעים(Pointer Reversal) : פרק 4.3 בספר (Garbage Collection, Jones & Lins, 96).



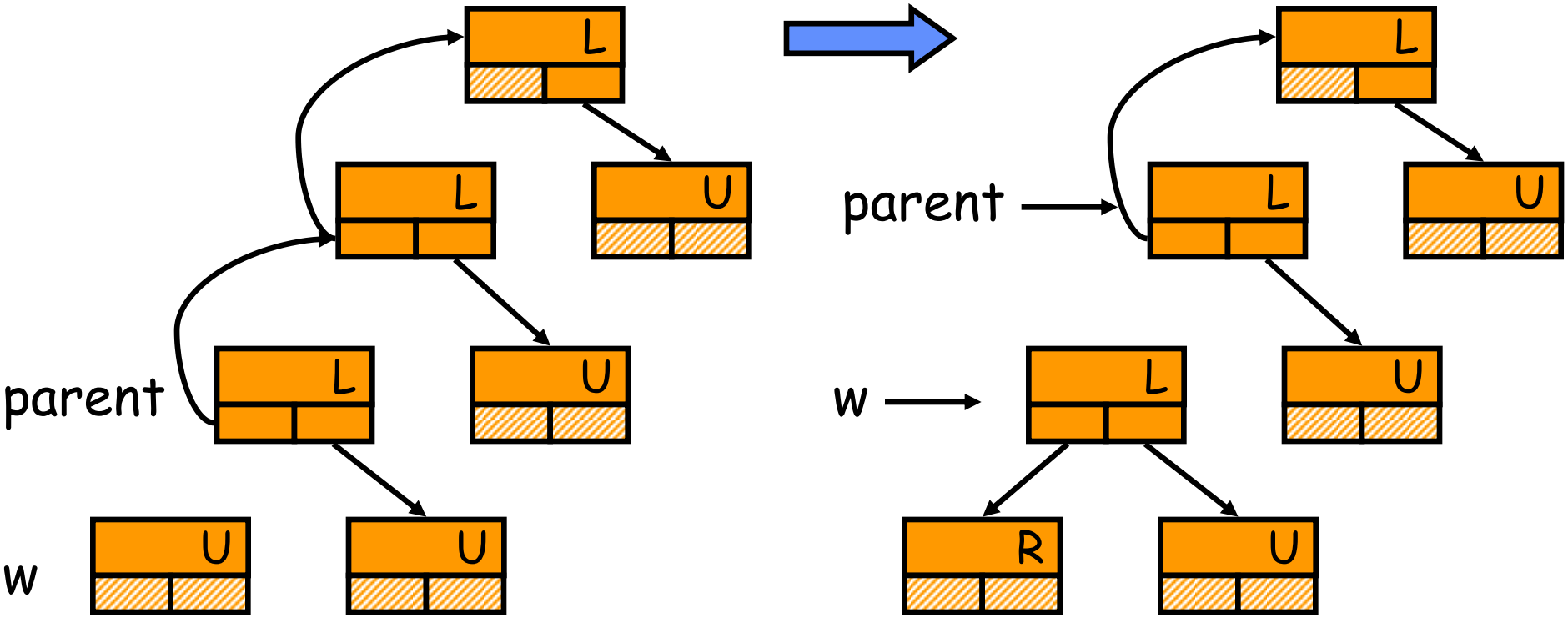
סימון ללא רקורסיה (המשך)



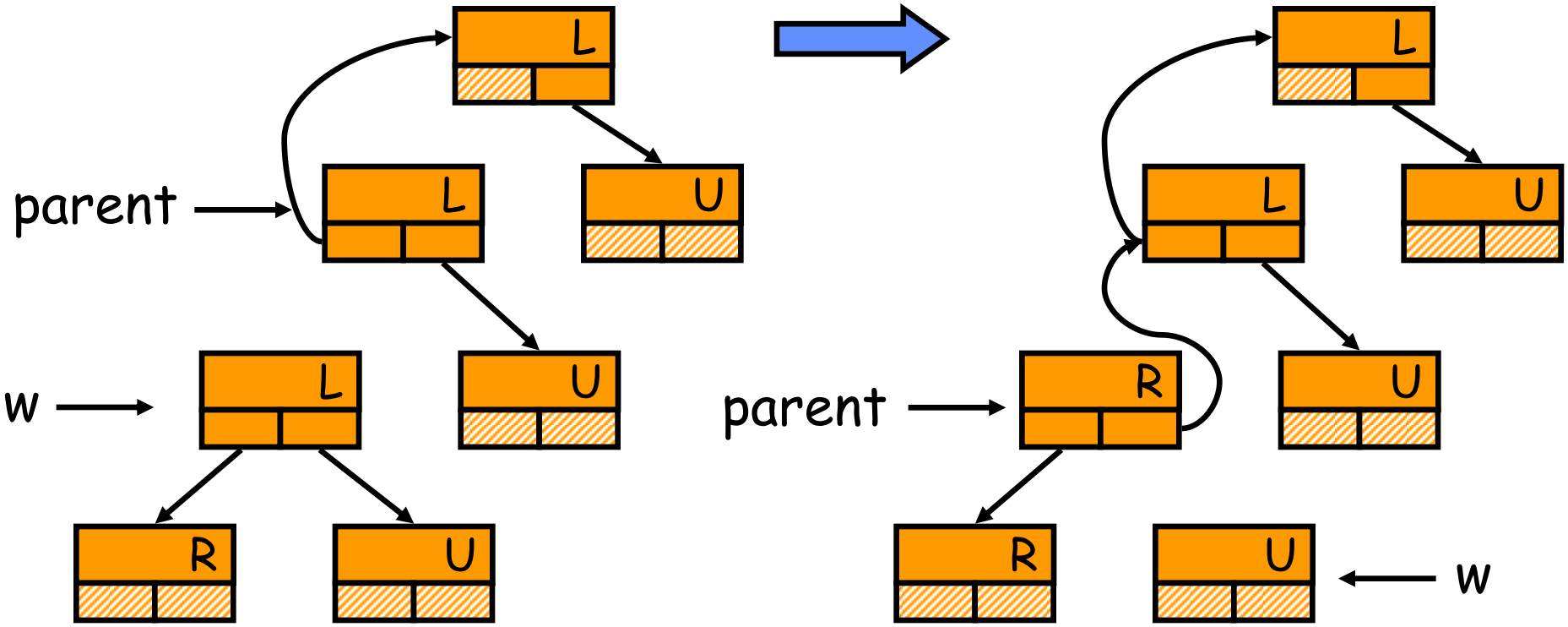
סימון ללא רקורסיה (המשך)



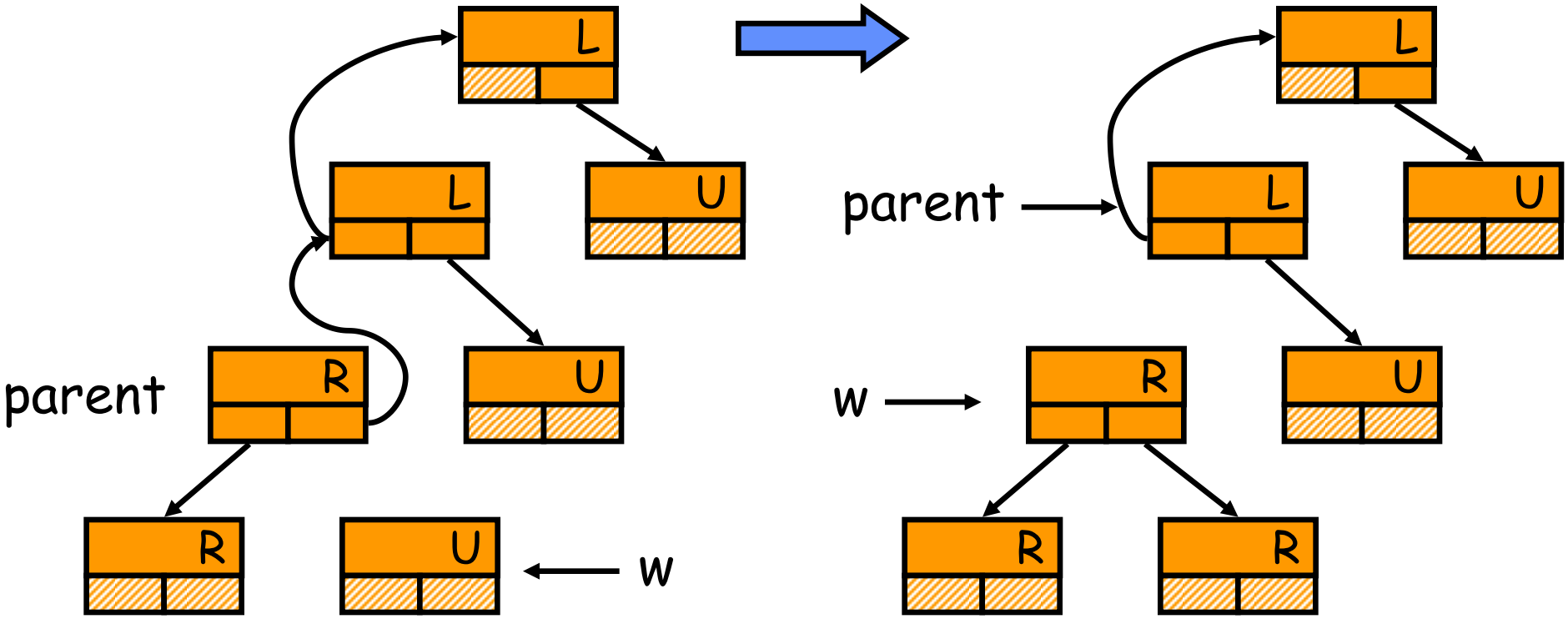
סימון ללא רקורסיה (המשך)



סימון ללא רקורסיה (המשך)

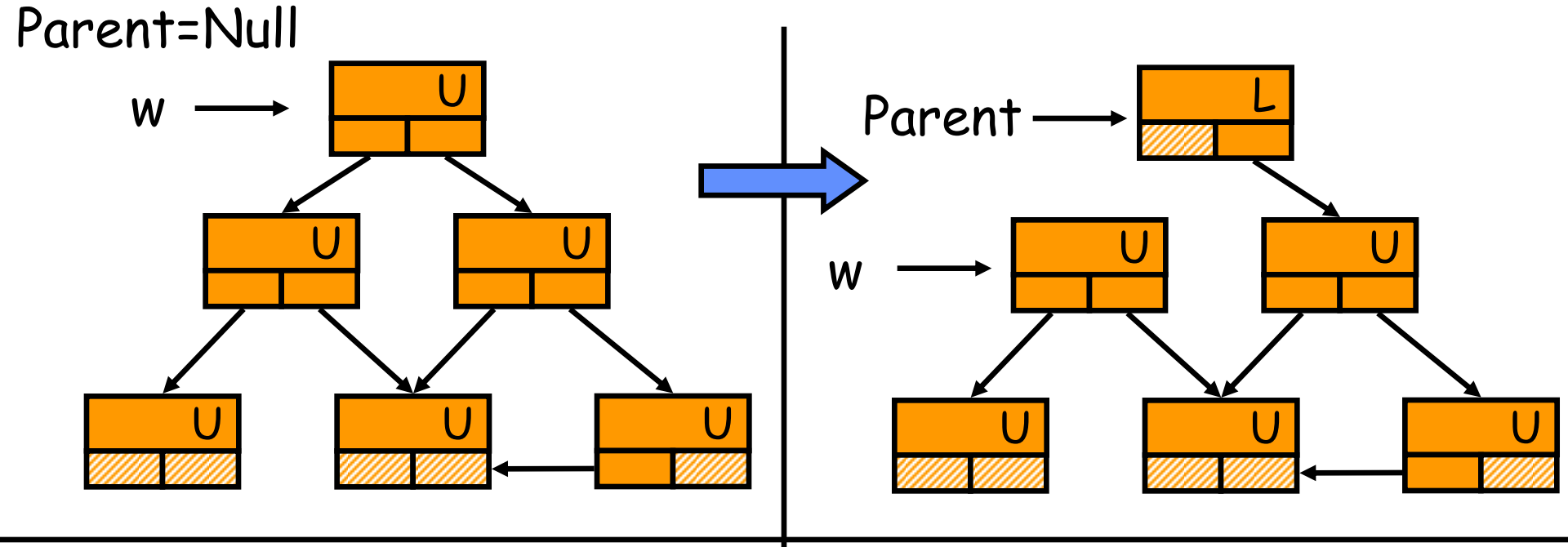


סימון ללא רקורסיה (המשך)

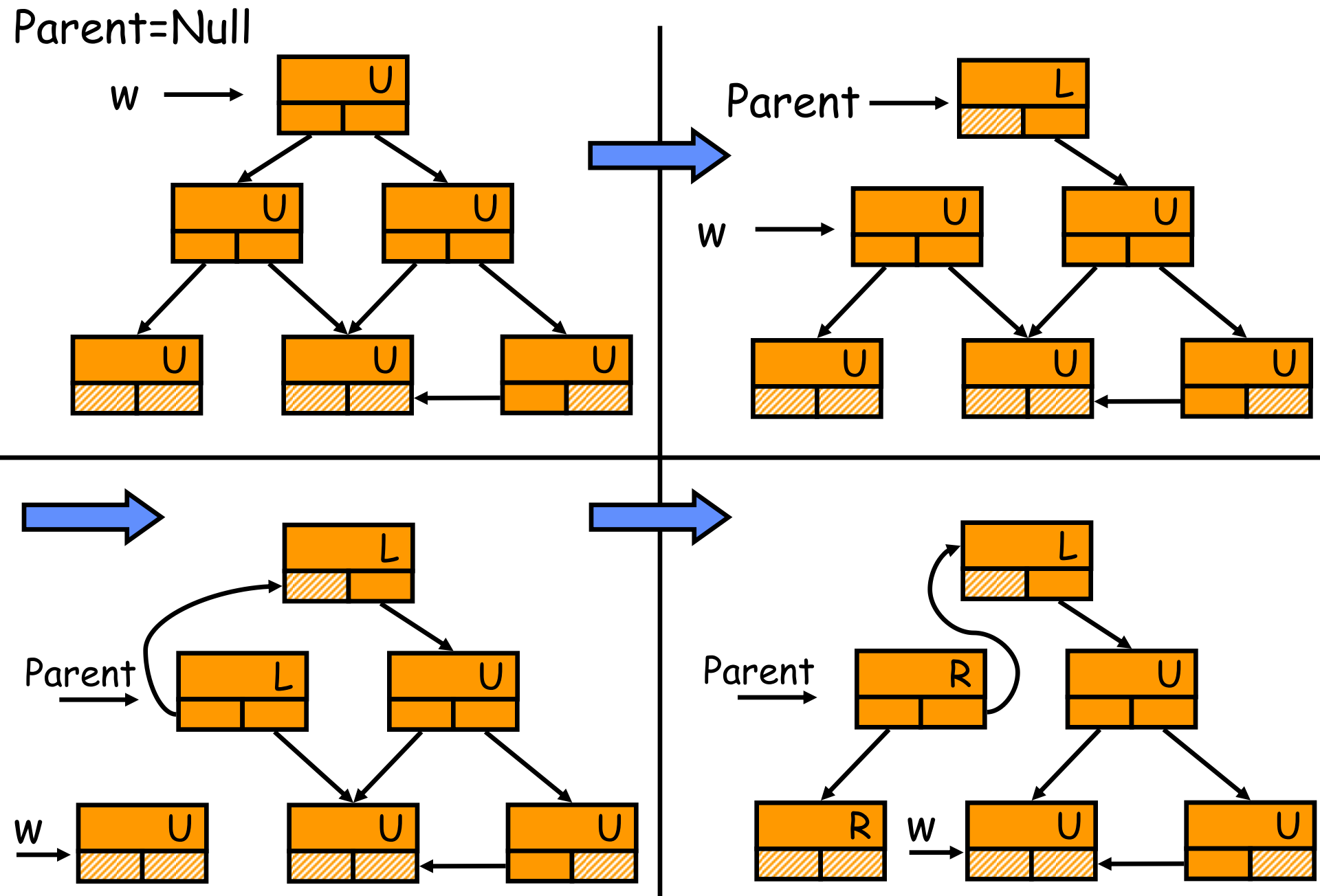


בסוף ריצת הפרוצדורה כל התוויות מסומנות R.

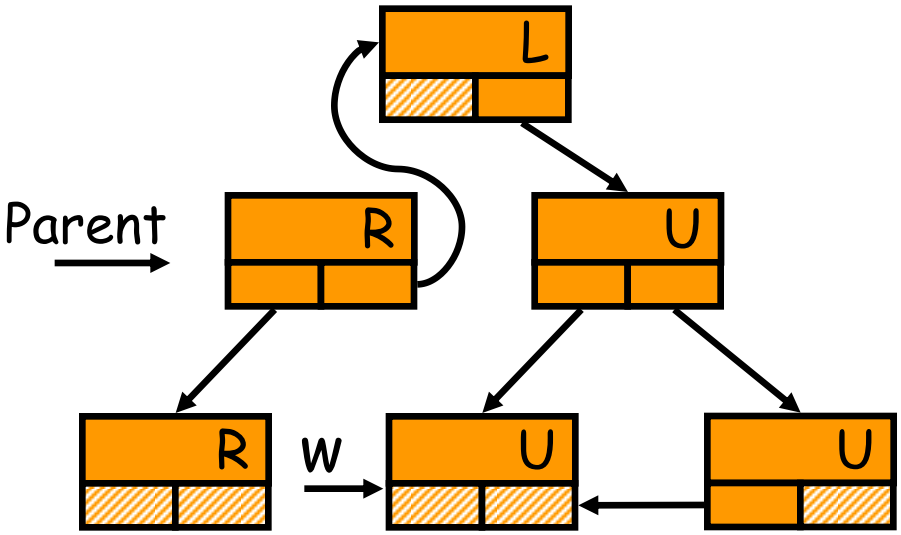
האלגוריתם עובד גם על גרף כללי

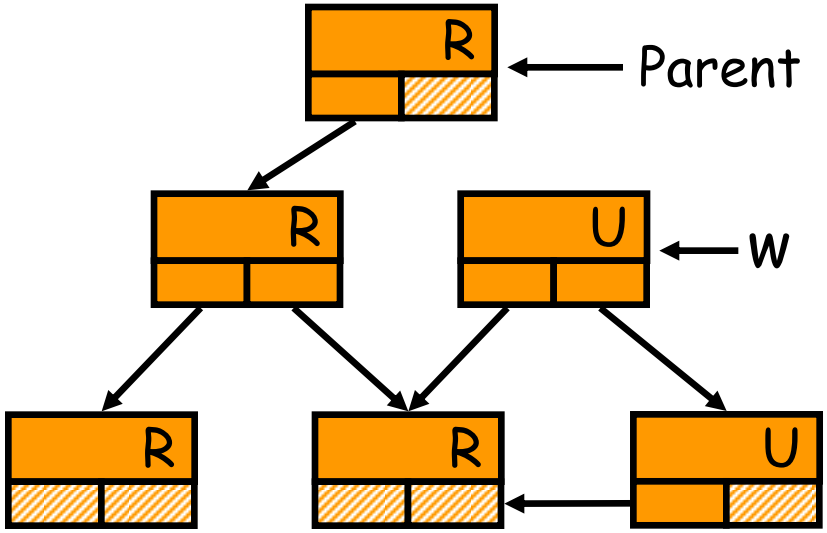


האלגוריתם עובד גם על גרף כללי

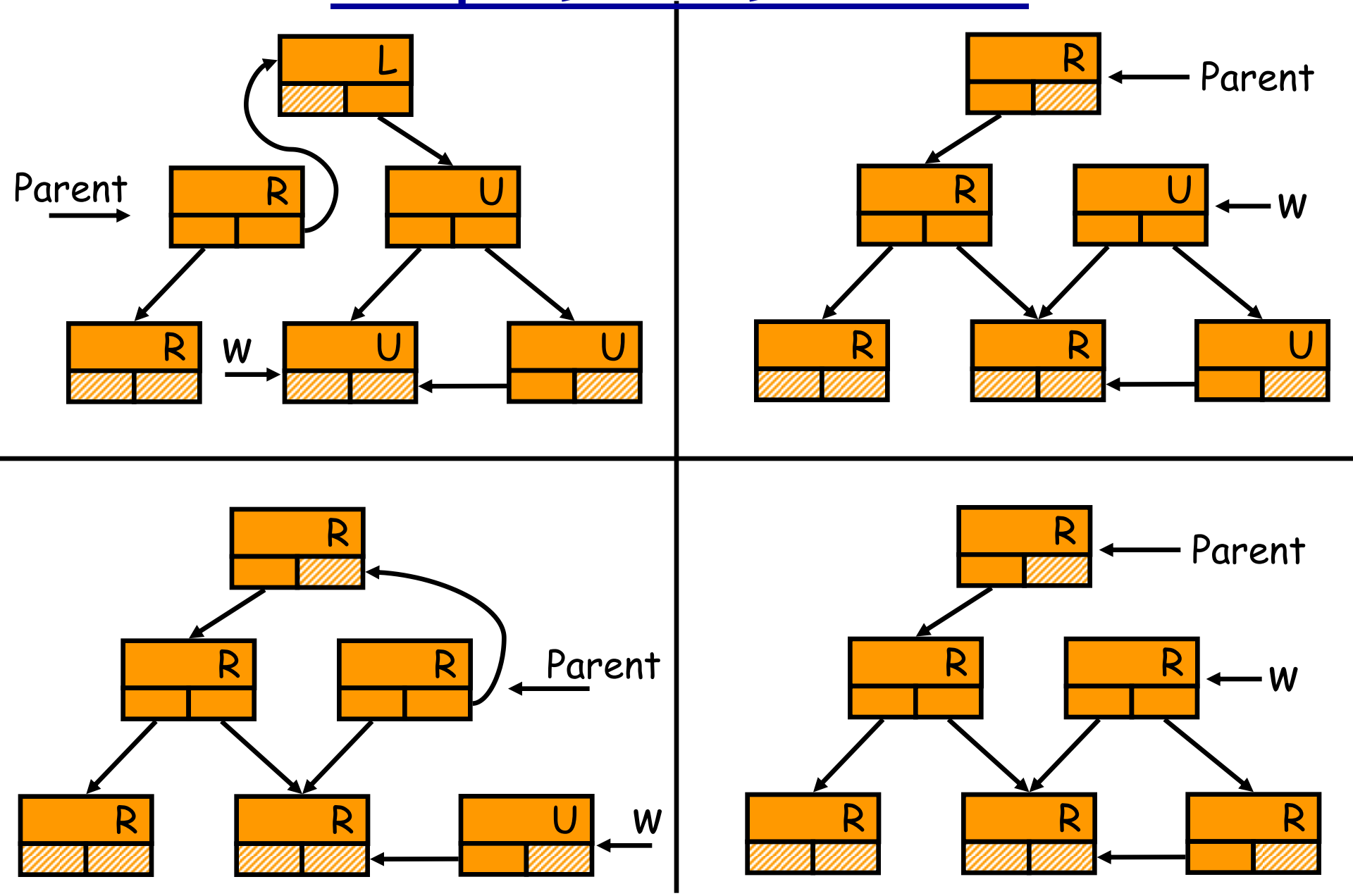


האלגוריתם עובד גם על גרף כללי





האלגוריתם עובד גם על גרף כללי



הערות

רצוי למנוע מצב שמדי פעם נעשית פעולה ארוכה של איסוף אשפה מזיכרון שבו נערמה אשפה רבה.

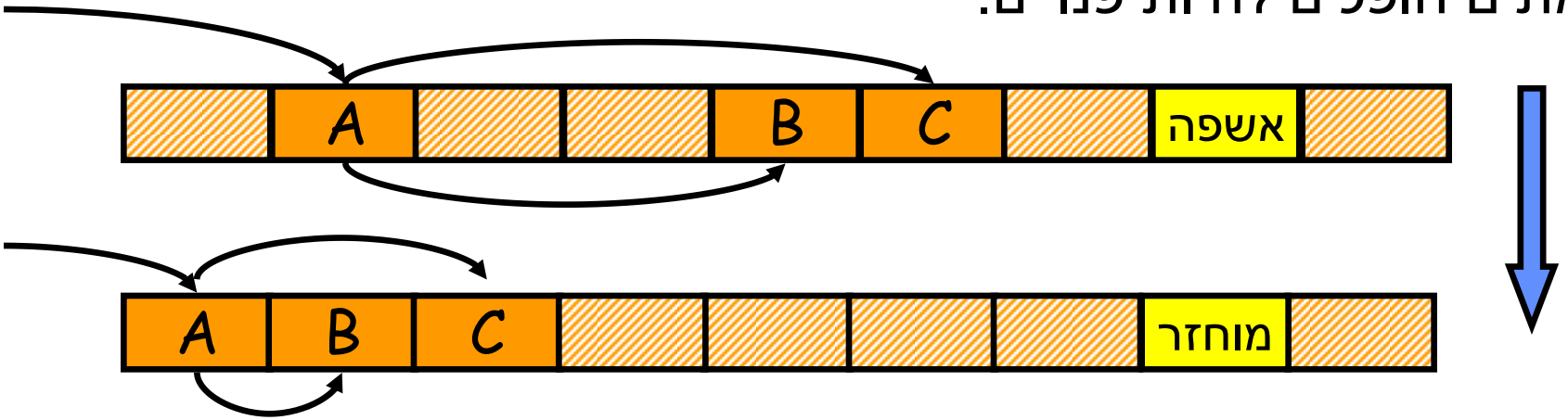
פתרון

- שילוב בן ספירה וסימון. ספירה משחררת צמתים בזמן שהם מתגלים.בדר"כ אין מעגלים רבים וספירה מונעת הצטברות אשפה. ניתן להגביל את גודל השדה count לשלושה-ארבע ביטים מפני שרק לעיתים רחוקות כמות המצביעים על צומת גדולה מ-10. מצב נדיר זה מטופל ע"י הסימון.
- איסוף אשפה גם כאשר רשימת AVAIL אינה ריקה (on the fly), למשל כאשר יש הפוגה בעומס על מערכת ההפעלה.

©cs,Technion

איסוף אשפה ע"י דחיסה (Compaction)

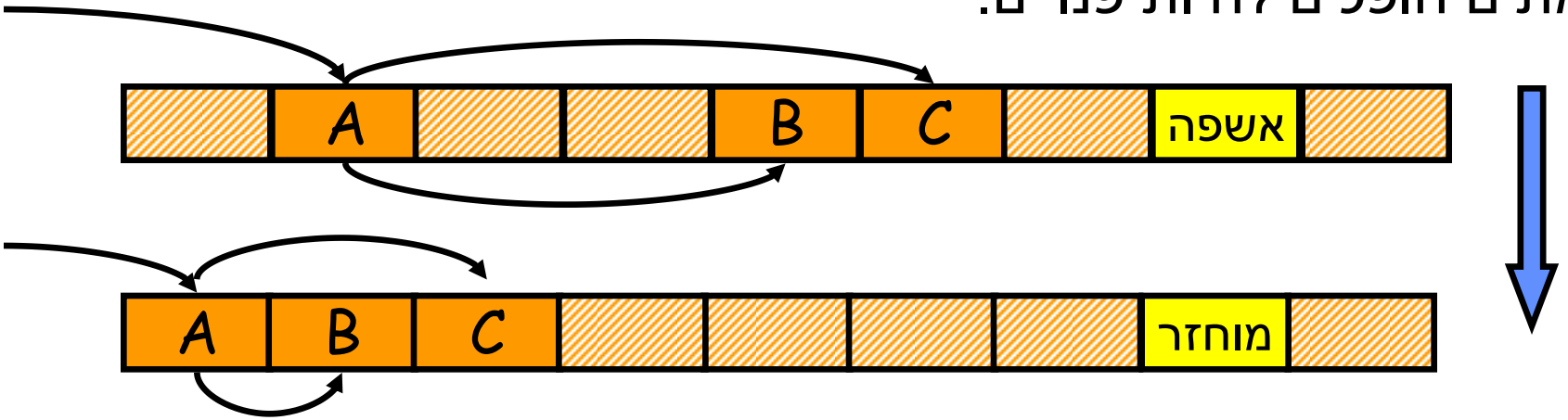
מטרה: ריכוז כל הצמתים הקשורים בתחילת הזיכרון - כל שאר הצמתים הופכים להיות פנויים.



נדרשת זהירות בעדכון המצביעים: העתקה של צמתים למקומם החדש אינה מספיקה. להלן פתרון ראשוני לבעיה:

איסוף אשפה ע"י דחיסה (Compaction)

מטרה: ריכוז כל הצמתים הקשורים בתחילת הזיכרון - כל שאר הצמתים הופכים להיות פנויים.



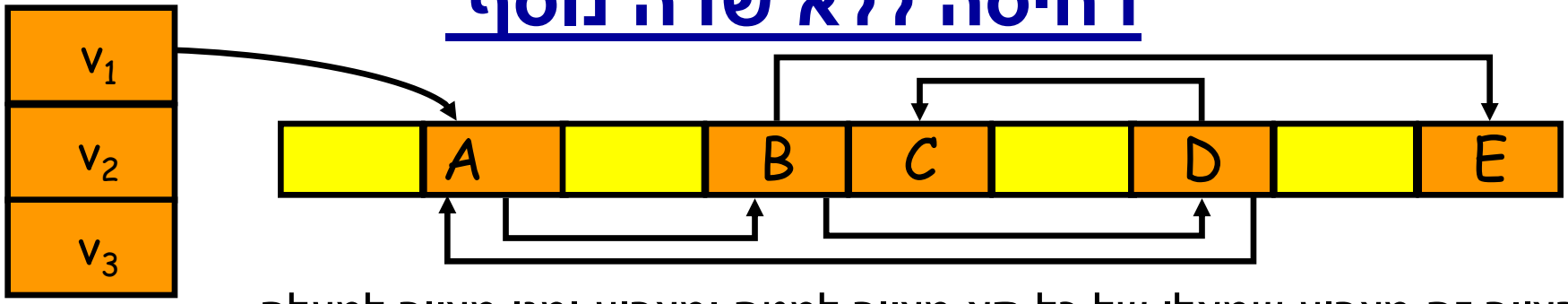
נדרשת זהירות בעדכון המצביעים: העתקה של צמתים למקומם החדש אינה מספיקה. להלן פתרון ראשוני לבעיה:

לכל צומת נוסיף שדה new-addr שישמש אותנו בזמן איסוף האשפה.

- נבצע סימון.
- נחשב לכל צומת את הכתובת החדשה, ונשים אותה בשדה new-addr.
- נעבור על הצמתים הקשורים ונעדכן מצביעים.
- נעתיק את הצמתים.

©cs,Technion

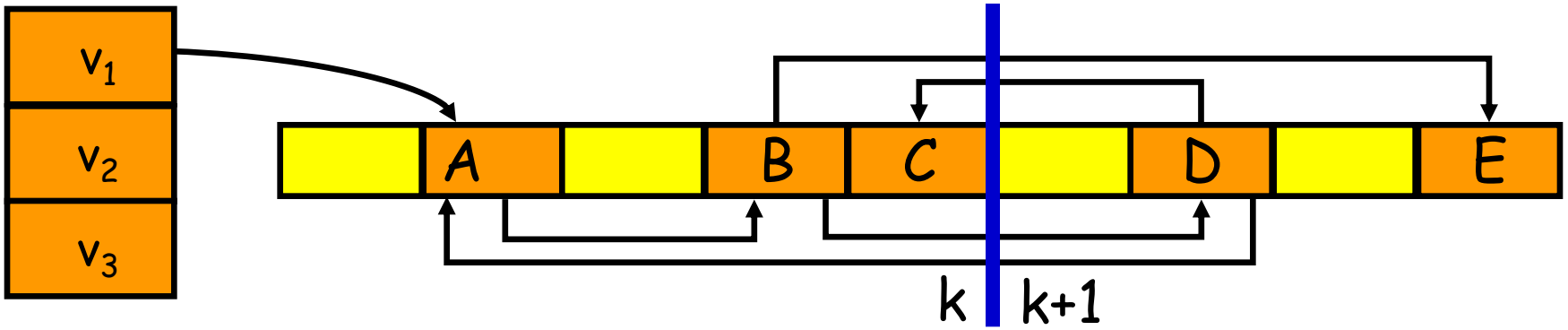
דחיסה ללא שדה נוסף



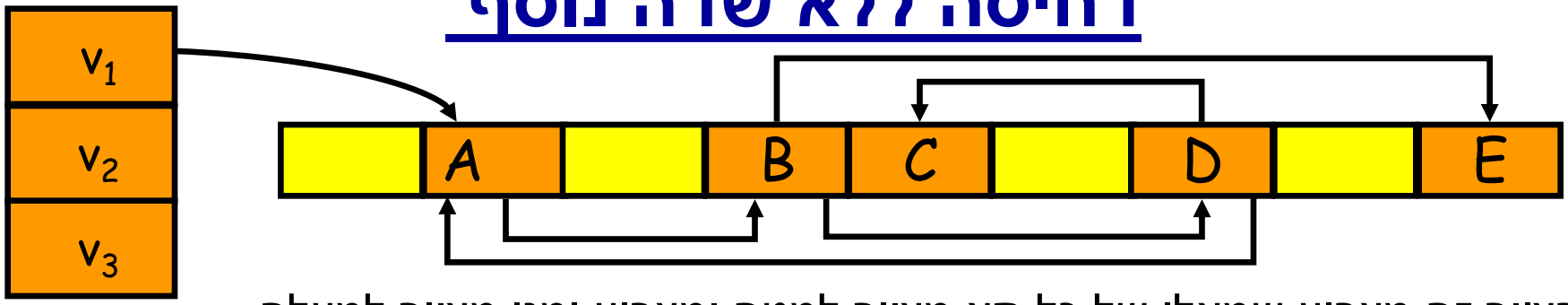
בציור זה מצביע שמאלי של כל תא מצויר למטה ומצביע ימני מצויר למעלה.

<u>הרעיון</u>: נניח שישנם k צמתים נגישים. בדוגמא k=5.

כל הצמתים בכתובות 1...k יושארו במקומם. כל צמתים הקשורים הנמצאים במקומות k+1...m יועתקו למרווחים שנמצאים ב-k הכתובות הראשונות.



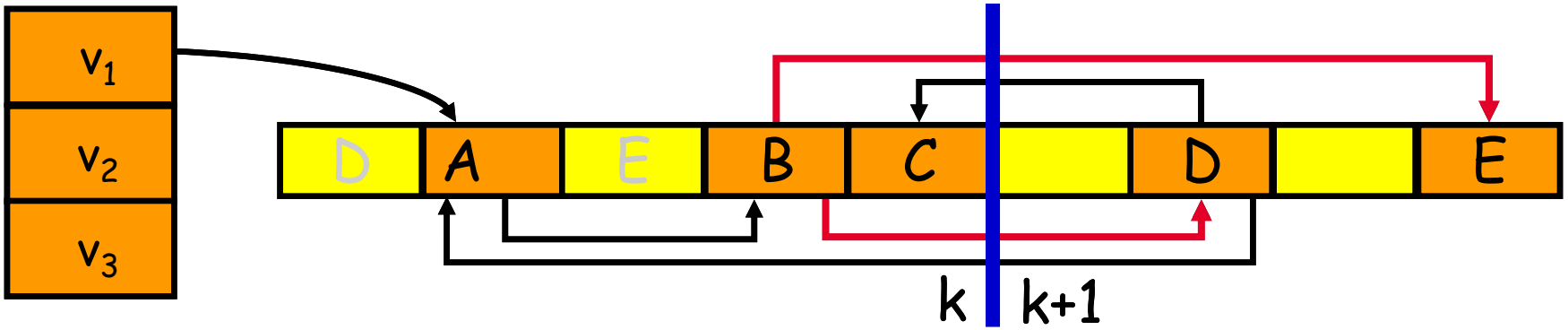
דחיסה ללא שדה נוסף



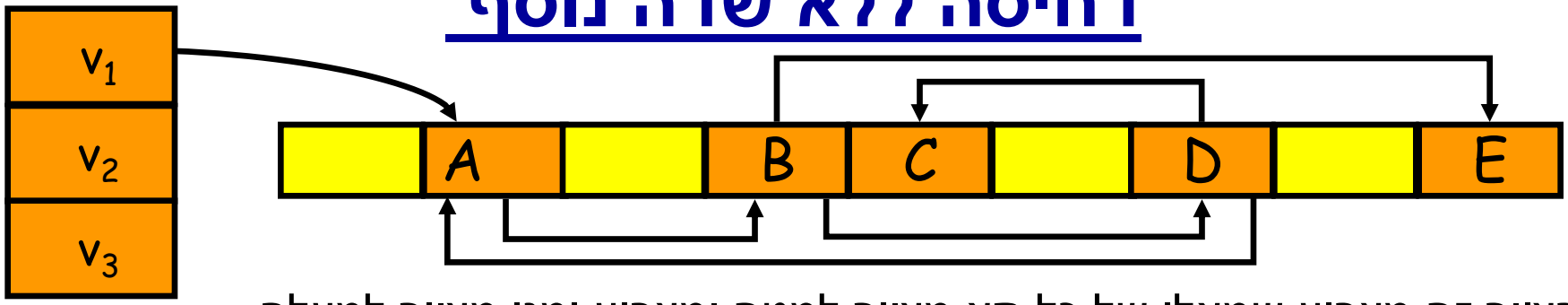
בציור זה מצביע שמאלי של כל תא מצויר למטה ומצביע ימני מצויר למעלה.

הרעיון: נניח שישנם k צמתים נגישים. בדוגמא k=5.

כל הצמתים בכתובות 1...k יושארו במקומם. כל צמתים הקשורים הנמצאים במקומות k+1...m יועתקו למרווחים שנמצאים ב-k הכתובות הראשונות.



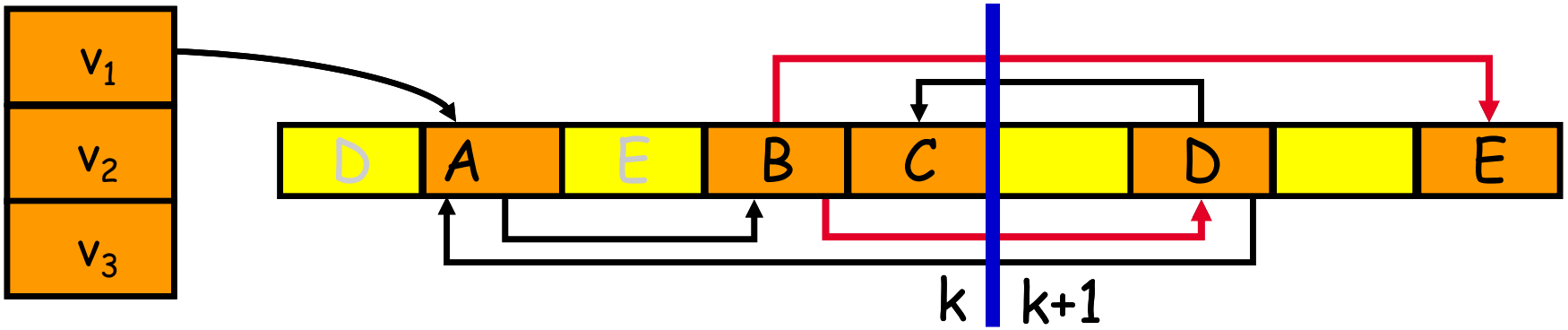
דחיסה ללא שדה נוסף



בציור זה מצביע שמאלי של כל תא מצויר למטה ומצביע ימני מצויר למעלה.

הרעיון: נניח שישנם k צמתים נגישים. בדוגמא k=5.

כל הצמתים בכתובות 1...k יושארו במקומם. כל צמתים הקשורים הנמצאים במקומות k+1...m יועתקו למרווחים שנמצאים ב-k הכתובות הראשונות.

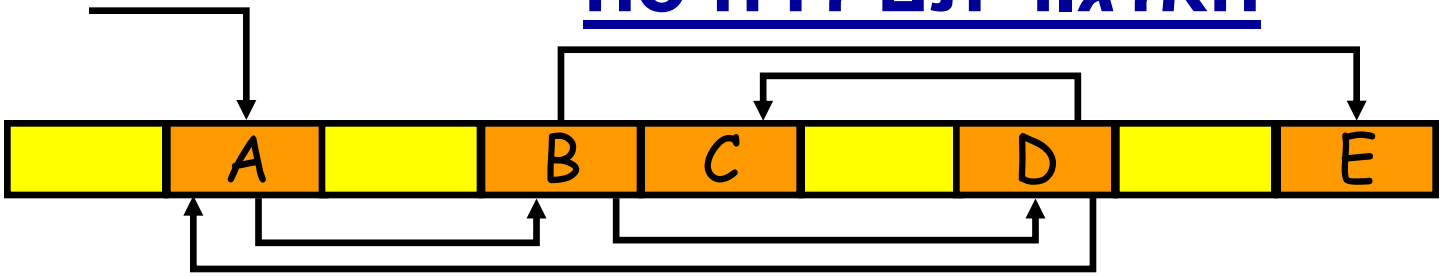


בזמן העתקת צומת V לכתובתו החדשה, נשמור במקומו הישן (נאמר בשדה left) מצביע לכתובת החדשה. מצביע זה ישמש לעדכון המכוונים המצביעים אל V. בדוגמא נעדכן את שני המצביעים של צומת B לכתובות החדשות של D ושל E.

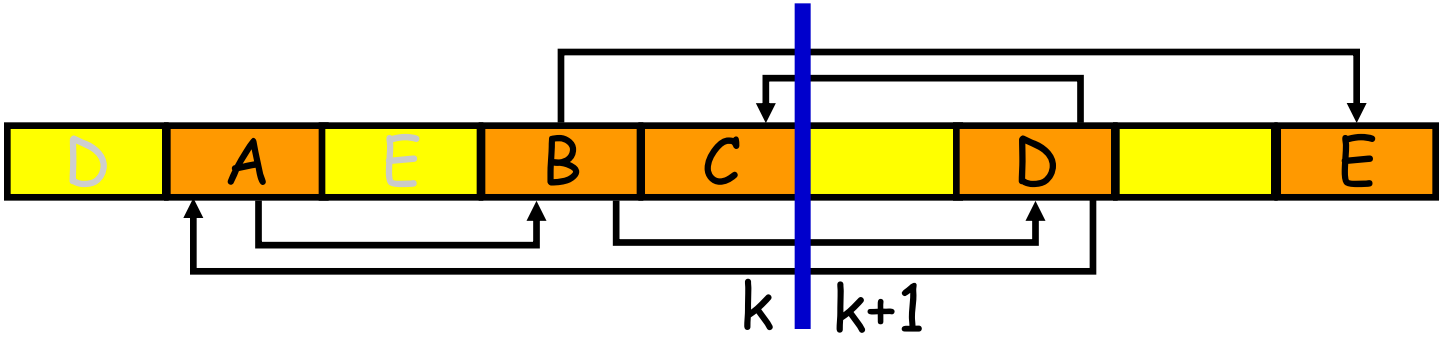
©cs,Technion

האלגוריתם לדחיסה

סימון וספירה:



חישוב כתובות:

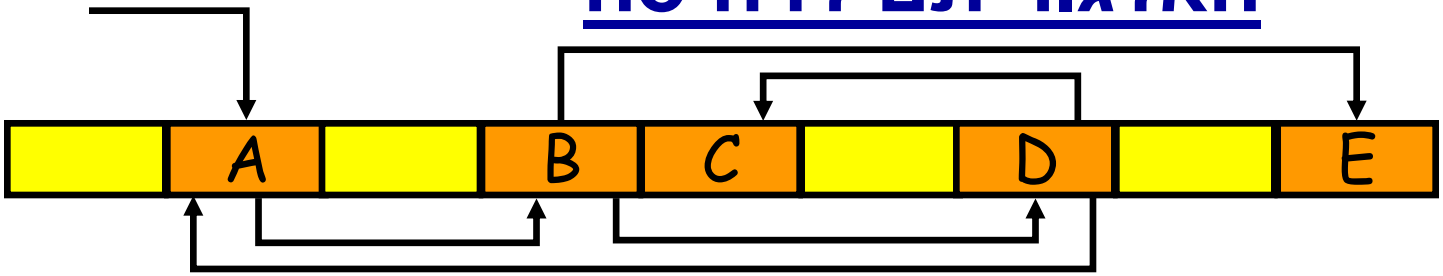


העתקת צמתים:

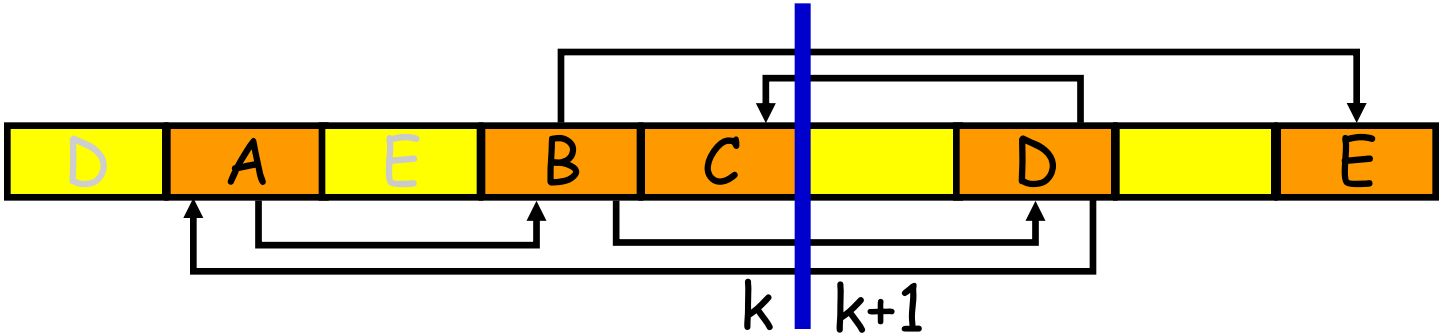
עדכון מצביעים:

האלגוריתם לדחיסה

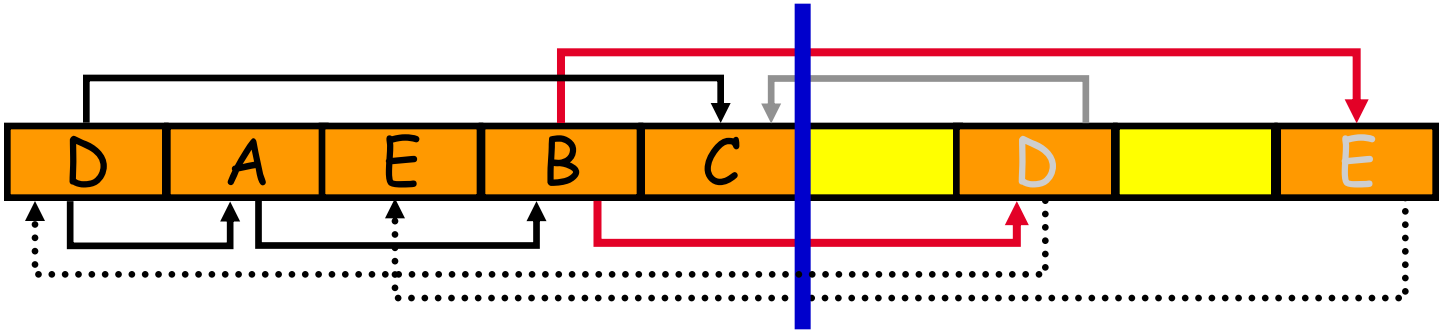
סימון וספירה:



חישוב כתובות:



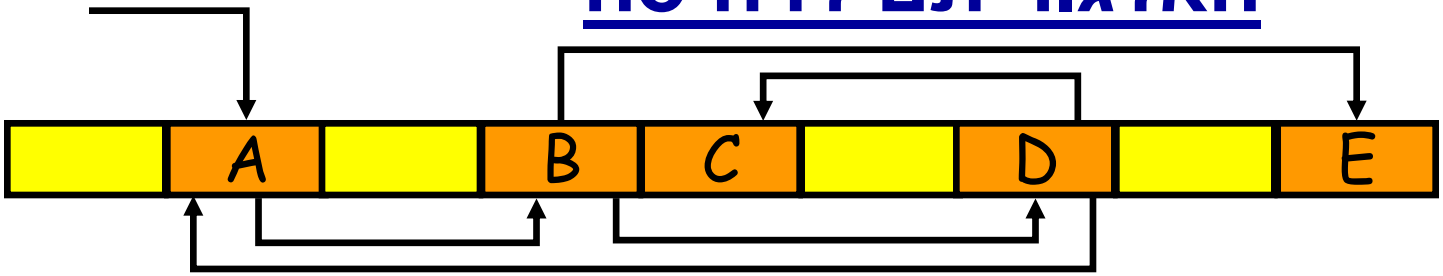
העתקת צמתים:



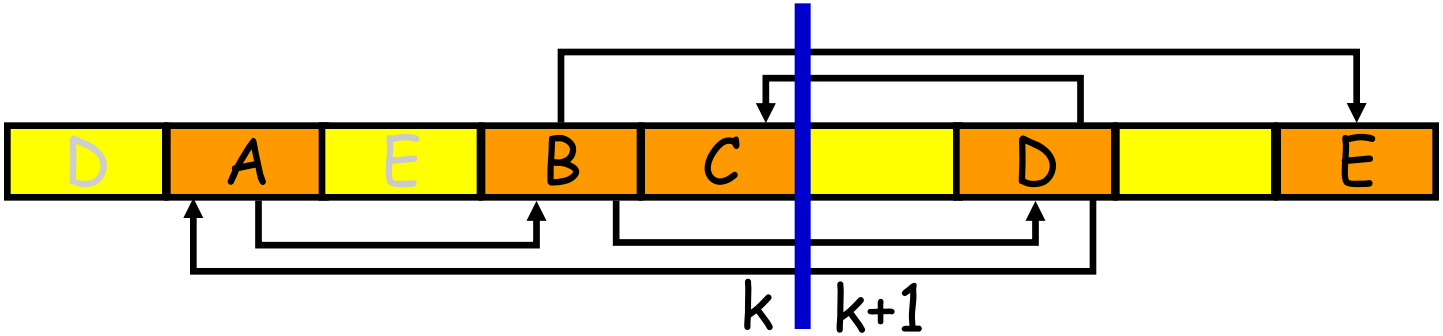
עדכון מצביעים:

האלגוריתם לדחיסה

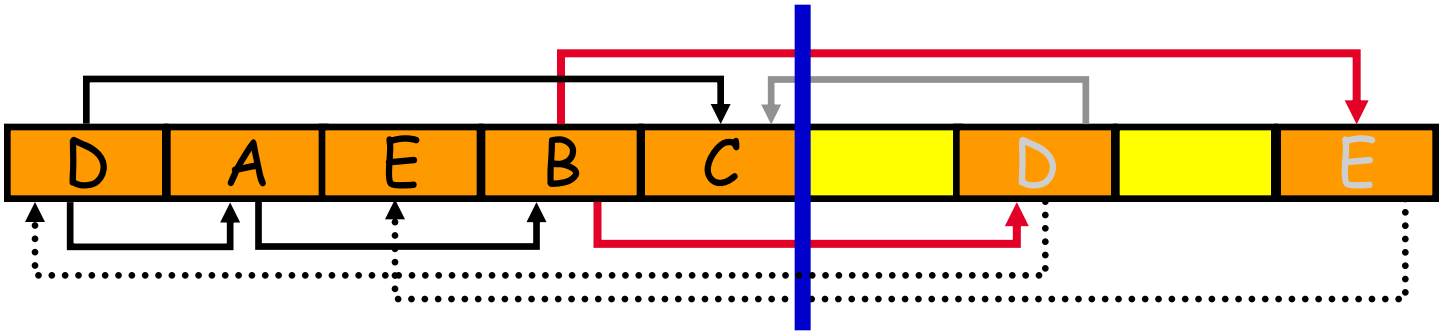
סימון וספירה:



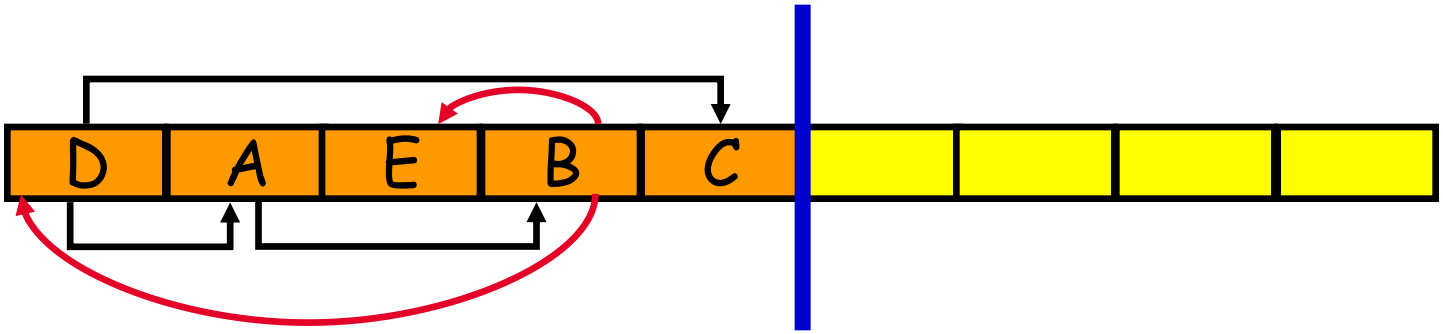
חישוב כתובות:



העתקת צמתים:



עדכון מצביעים:



הערות לסיכום

- בשיטת הספירה משתמשים לדוגמא במערכות ה- unix: awk, perl וכן במערכות חומרה קטנות כמו בקרים בהם פעולת reset מנקה אשפה.
- בשפת MODULA 2+ משתמשים בשיטת ספירה עם backup בעזרת שיטת הסימון.
- בספר Garbage Collection: Algorithms for automatic dynamic memory management, Richard Jones and Rafael Lins, Wiley, 1999.

יש פרקים על איסוף אשפה בשפת C++ ובשפות רבות נוספות.

- כל האלגוריתמים בהם דנו ניתנים להרחבה לזיכרון המכיל אובייקטים מסוגים וגדלים שונים.

©cs,Technion

Garbage Collection