מבני נתונים ואלגוריתמים

חזרה, מיונים
פולינה לוצקר

שיעורי בית- תזכורת

- יש להגיש את התרגיל במערכת submit עד השעה 23:55 בתאריך 20.11.
- יש לממש את הערימה בחלק השני לבד- אני אבדוק!
- יש להשתמש אך ורק במיון ערימה ויש לממש את המיון בעצמכם.
- תהיה בדיקת העתקה.
- יש לתכנת בpython3 ואך ורק בפייתון 3.
- **לכתוב שם ות"ז!**

תזכורת-עצים בינאריים

עץ: גרף קשיר ללא מעגלים.

עץ בינארי: עץ מכוון, שבו לכל קודקוד יש לכל היותר שני בנים.

עץ חיפוש בינארי: מקיים את התכונה שלכל צומת, כל ערכי הצמתים בתת העץ הימני שלו גדולים מערך הצומת, וכל ערכי הצמתים בתת העץ השמאלי שלו קטנים מערך הצומת.

מעבר על עצים:

ln-order,posr-order,pre-order

שאלה: באיזה מעבר נרצה לעבור על עץ חיפוש בינארי כדי להדפיס את הערכים ממוינים?

תזכורת-עצים בינאריים

עץ: גרף קשיר ללא מעגלים.

עץ בינארי: עץ מכוון, שבו לכל קודקוד יש לכל היותר שני בנים.

עץ חיפוש בינארי: מקיים את התכונה שלכל צומת, כל ערכי הצמתים בתת העץ הימני שלו גדולים מערך הצומת, וכל ערכי הצמתים בתת העץ השמאלי שלו קטנים מערך הצומת.

מעבר על עצים:

In-order,posr-order,pre-order

שאלה: באיזה מעבר נרצה לעבור על עץ חיפוש בינארי כדי להדפיס את הערכים ממוינים? In-order

עצים מאוזנים

משפחה של עצים בינאריים תקרא מאוזנת אם כל עץ במשפחה המכיל n קודקודים גובהו $O(logn)$.

שאלה לתזכורת: למה עדיף לעבוד עם עצים מאוזנים?

עצים מאוזנים

משפחה של עצים בינאריים תקרא מאוזנת אם כל עץ במשפחה המכיל n קודקודים גובהו $O(logn)$.

שאלה לתזכורת: למה עדיף לעבוד עם עצים מאוזנים?

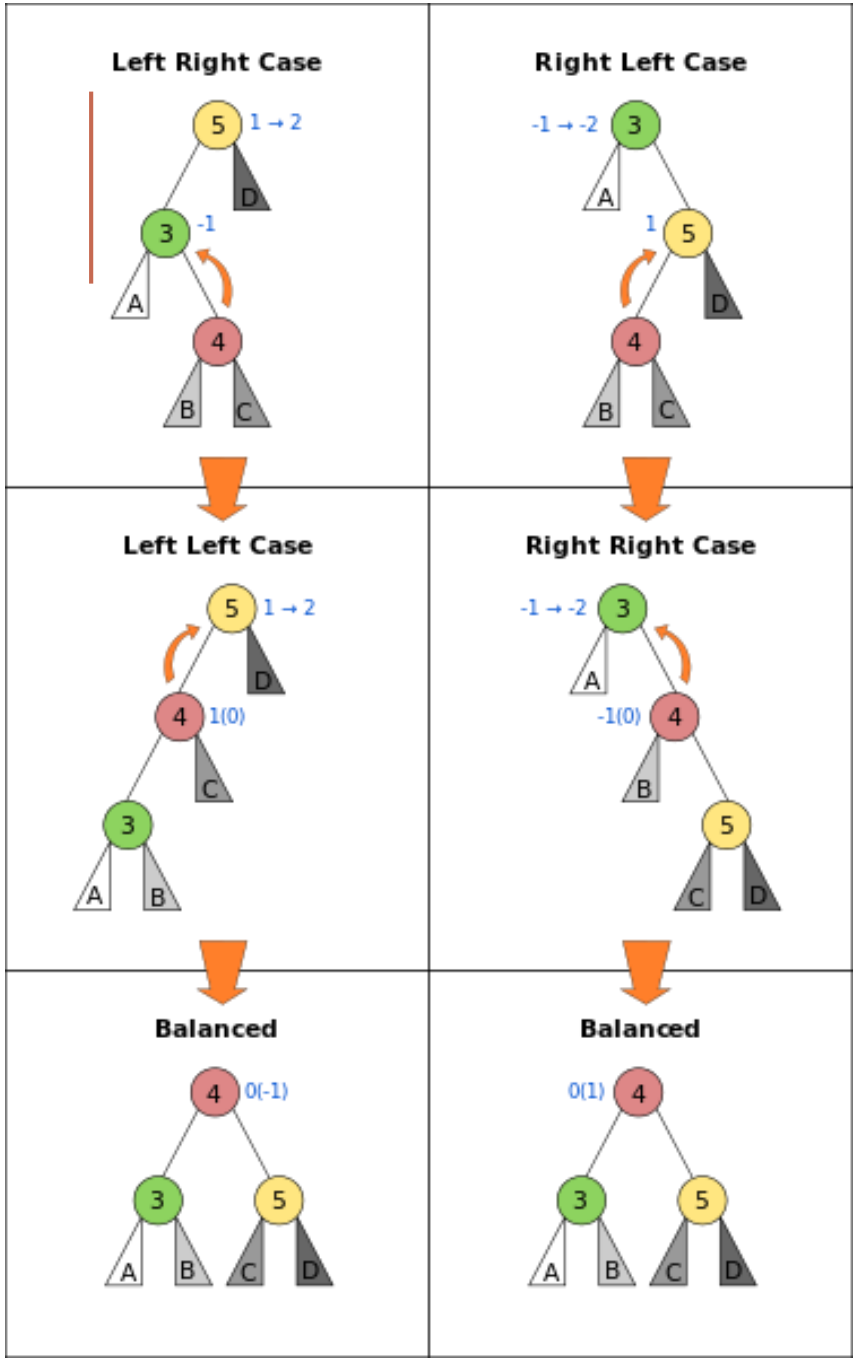
תכונה זו מבטיחה שניתן יהיה לחפש בעץ, להכניס ולהוציא ממנו נתונים בסיבוכיות $O(logn)$ כאשר n הוא מספר הצמתים בעץ.

עצים מאוזנים

עצים מאוזנים

עצי 2-3

עצי AVL



עצי AVL – תזכורת

שאלה

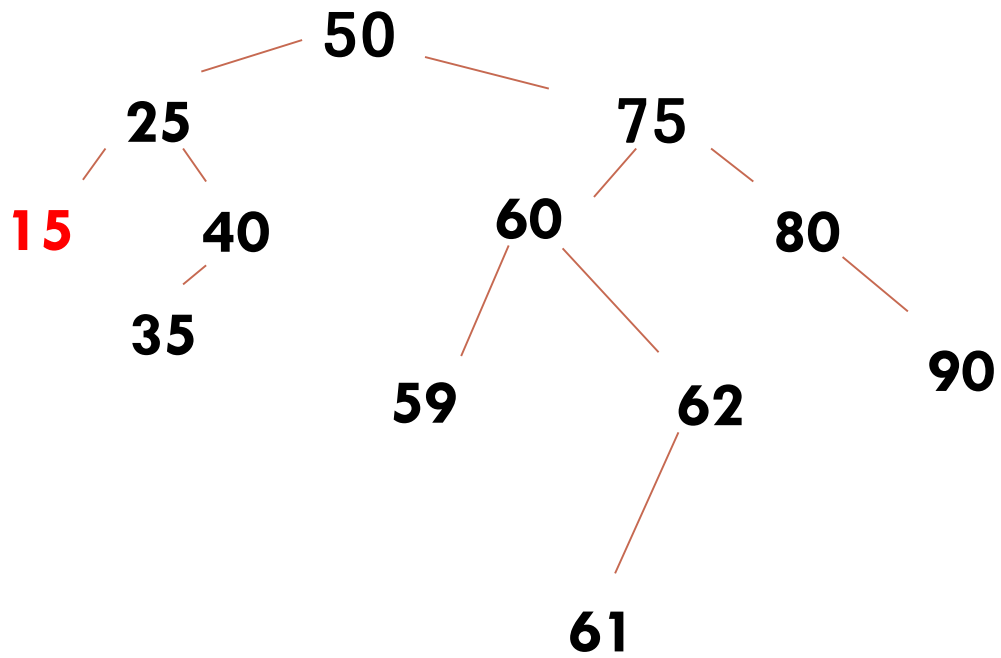
הוכח/הפרך:

1. בעת הכנסת איבר חדש לעץ AVL מבוצע איזון של העץ בצומת אחד לכל היותר.

2. בעת מחיקת איבר קיים מעץ AVL מבוצע איזון של העץ בצומת אחד לכל היותר

שאלה

1. הוכחה- בבית.
2. הפרכה- אם נמחק את 15 מהעץ מהבא: (שב)



מיונים

מיונים

מיוני השוואה
המקרה הכי טוב - $\Theta(n \log n)$

מיוני לא השוואה-
מיונים עם הנחות
נוספות על הקלט

שאלה

הגדר "מיון יציב" ותן דוגמה למיון יציב ודוגמה למיון לא יציב.

שאלה

הגדר "מיון יציב" ותן דוגמה למיון יציב ודוגמה למיון לא יציב.

מיון נקרא מיון יציב אם הוא שומר על הסדר של הנתונים לאחר המיון גם כשיש שני נתונים זהים.

דוגמה למיון יציב: מיון מיזוג.

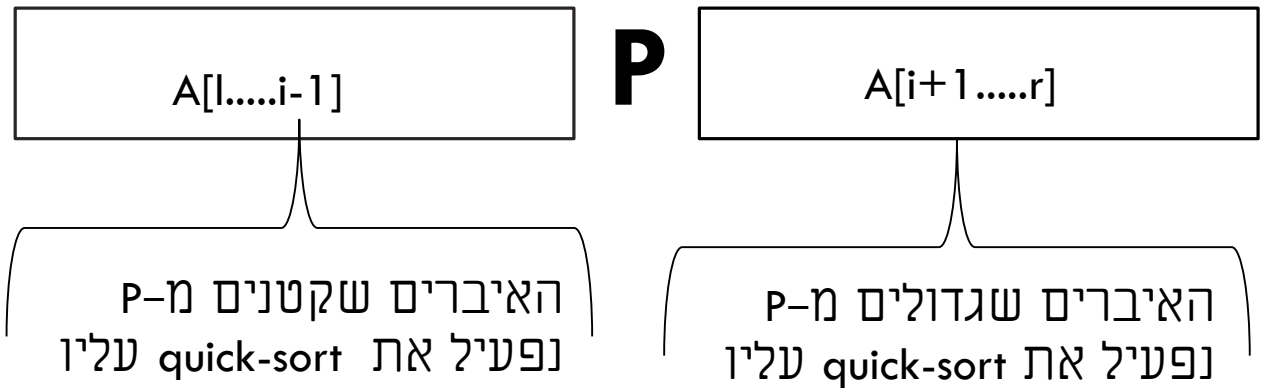
דוגמה למיון לא יציב: מיון ערימה.

מיוני השוואה

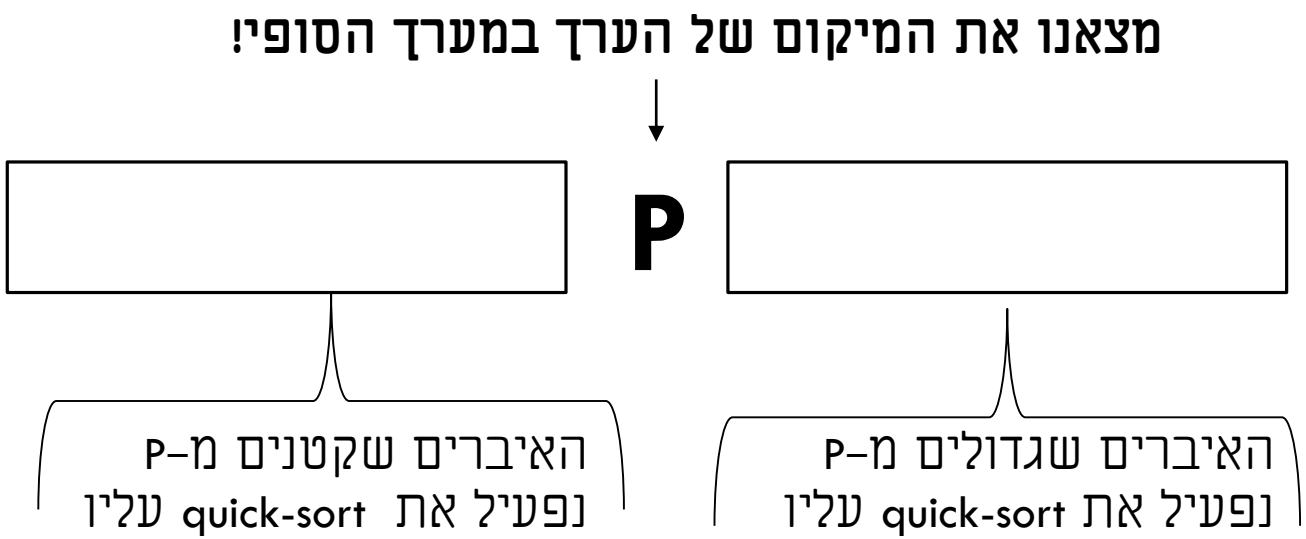
- מיון בועות $\Theta(n^2)$
- מיון Insertion-Sort $\Theta(n^2)$
- מיון Quick-Sort $\Theta(n\log(n))$
- מיון Merge-Sort $\Theta(n\log(n))$
- מיון Heap-Sort $\Theta(n\log(n))$

QUICK-SORT – הרעיון הבסיסי

- עובד באופן רקורסיבי– גישה של "הפרד ומשול".
- הרעיון של האלגוריתם:
  - בכל שלב ממיינים מערך A[l....r]
  - נבחר איבר מהמערך שאנחנו מימנים A[p] שיהיה איבר הציר (pivot)
  - נחלק את המערך ל3 חלקים: האיבר שקטנים מאיבר הציר שהם בעצם A[l..i-1], איבר הציר יהיה במקום ה A[i], האיברים שגדולים מאיבר הציר A[i+1...r] **(הפרד).**
  - נמיין רקורסיבית את A[l...i-1] ואת A[i+1..r] **(משול)**



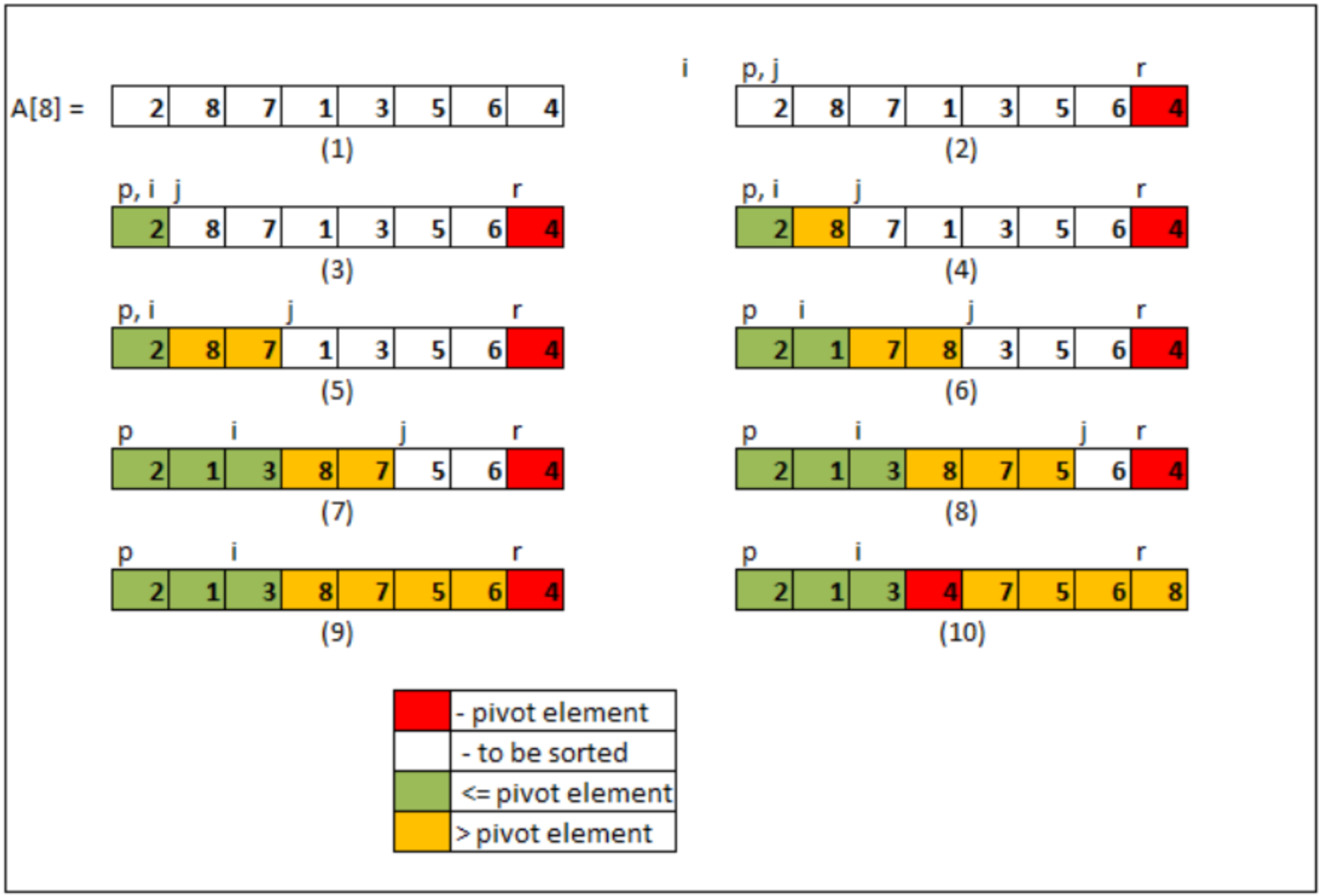
נשים לב –QUICK-SORT



QUICK-SORT

- איך בוחרים את איבר הציר?
- איך נסדר את האיברים כך שמי שקטן יהיה לפני איבר הציר?
- נהוג לקרוא לפעולה Partition

PARTITION-


A[8] =
2 8 7 1 3 5 6 4
(1)
i
p, j r
2 8 7 1 3 5 6 4
(2)
p, i j r
2 8 7 1 3 5 6 4
(3)
p, i j r
2 8 7 1 3 5 6 4
(4)
p, i j r
2 8 7 1 3 5 6 4
(5)
p i j r
2 1 7 8 3 5 6 4
(6)
p i j r
2 1 3 8 7 5 6 4
(7)
p i j r
2 1 3 8 7 5 6 4
(8)
p i r
2 1 3 8 7 5 6 4
(9)
p i r
2 1 3 4 7 5 6 8
(10)
- pivot element
- to be sorted
<= pivot element
> pivot element

PARTITION – פסאדו קוד

```
PARTITION(A, p, r)
  pivot = A[r]  - picking the last element as the pivot.
  i  = p -1
  for j = p to r-1
    if A[j] <= pivot
    i = i+1
    swap A[i] and A[j]
  swap A[i+1] with A[r] - this will put the pivot in right place
return i+1
```

פסאדו קוד QUICK-SORT

```
Quicksort(A, low, high)
{
// If we are not in the base case
If(high>low):
        // Partition the array using pivot value
        pivot = partition(A, low, high)

        // Quicksort left partition recursively
        Quicksort(A,low,pivot-1)

        // Quicksort right partition recursively
        QuickSort(A,pivot+1,high)
}
```

הקריאה תראה כך:

A is an array

Quicksort(A, 0,len(A)-1)

QUICK-SORT

3 1 4 5 9 2 8 6
1 2 3 4 5 9 8 6
1 2 3 4 5 9 8 6
1 2 3 4 5 9 8 6
1 2 3 4 5 8 6 9
1 2 3 4 5 6 8 9
1 2 3 4 5 6 8 9

זמני ריצה QUICK-SORT

נסתכל על נוסחת הנסיגה:

$$T(n) = \theta(n) + T(i) + T(n-i-1)$$
$$T(0) = T(1) = 1$$

כאשר i הוא גודל הבלוק הראשון לפי ההפרדה לפי איבר הציר.

נשאל את עצמנו:

- מהו המקרה הממוצע?
- מהו המקרה האופטימלי?
- מהו המקרה הגרוע?

שאלה: במה זה תלוי?

זמני ריצה QUICK-SORT

נסתכל על נוסחת הנסיגה:

$$T(n) = \theta(n) + T(i) + T(n-i-1)$$
$$T(0) = T(1) = 1$$

כאשר i הוא גודל הבלוק הראשון לפי ההפרדה לפי איבר הציר.

נשאל את עצמנו:

- מהו המקרה הממוצע?
- מהו המקרה האופטימלי?
- מהו המקרה הגרוע?

שאלה: במה זה תלוי?

בבחירה של איבר הציר!

QUICK – SORT המקרה הגרוע

במידה ונתון מערך ממוין מהקטן גדול, ונבחר כל פעם את איבר הצירלהיות האיבר הקטן ביותר (או הגדול ביותר)

בכל קריאה רקורסיבית המערך A[l,r] נבצע קריאה רקורסיבית למערך A[l,l-1] ולמערך A[l+1,n]

וסך הכול נקבל את הנוסחה:

$$T(n) = cn + T(n-1) = cn + c(n-1) + T(n-2) = c\sum_{i=1}^{n} i = \theta(n^2)$$

מיונים שאינם מיוני השוואה

1) *Counting sort*: מניחים שכל איברי הקלט הם מספרים בתחום מ-1 עד *k*, עבור *k* שלם כלשהו. כאשר *k=O(n)*, הסיבוכיות היא *O(n)*. הרעיון – סופרים כמה ערכים במערך הקלט קטנים או שווים לכל ערך מ-1 עד *k*, ואז מכניסים את האיברי הקלט למערך פלט ישר למקום הנכון. המיון יציב.
2) *Bucket sort*: גם כאן יש הנחה על הקלט – איברי הקלט מגיעים מהתפלגות אחידה בקטע מסוים (לדוג' - *[0,1)*). מחלקים את הקטע ל-*n* דליים ומפזרים את הערכים בכל דלי. בגלל שהם מתגיעים מהתסלגות אחידה – מצפים שהחלוקה תהיה בערך שווה בין הדליים. ממיינים את המספרים בכל דלי ועוברים על כל הדליים כדי לקבל את הפלט. תוחלת זמן הריצה היא *O(n)*.
3) *Radix sort*: ממיינים לפי הספרה הכי פחות משמעותית (*Least significant digit*) (באמצעות מיון יציב כגון *counting sort*) ואז לפי הספרה הלפני האחרונה וכך הלאה. סיבוכיות – *O(logn)*. יש גם *Most significant digit* (הפוך).

COUNTING SORT

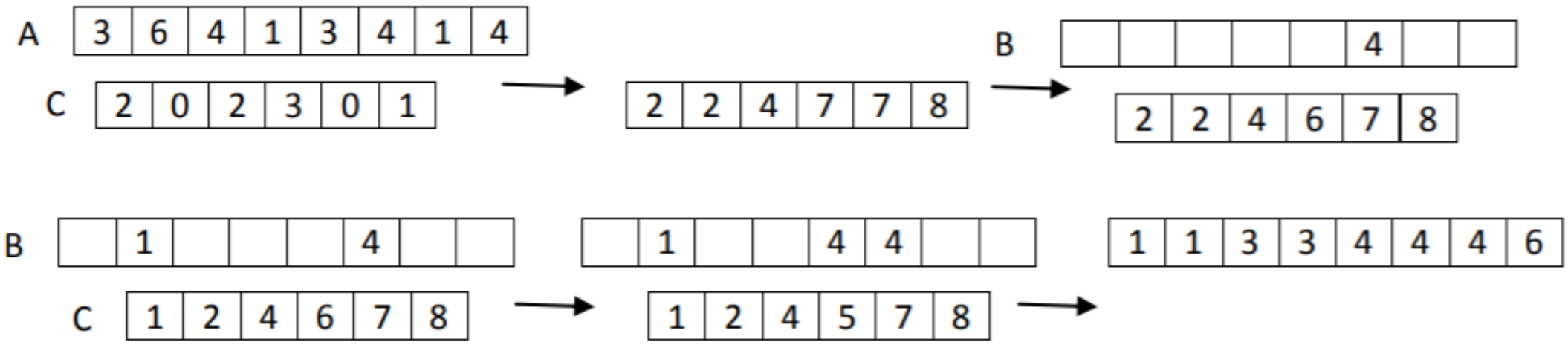
Counting sort (A, B, k) :

// A – input array (size n), B- output array (size n), k – range of values (1-k) . C – array of size k

```
for  i=1 to k
        C[i] = 0;
for j=1 to length(A)
        C[A[j]] = C[A[j]]+1;
// C[i] contains the number of elements equal to i
for i=2 to k
        C[i] = C[i] + C[i-1];
// C[i] contains the number of elements less than or equal to i
for j=length(A) downto 1
        B[C[Aj]]] = A[j];
        C[A[j]] = C[A[j]] – 1
// reduce by 1 for equal values (the next equal value will be inserted in the previous slot)
```

COUNTING SORT

דוגמאת ריצה:



תרגיל

נניח שמשנים את לולאת ה- forהאחרונה באופן הבא:

For j=1 to len(A)

האם לאחר השינוי האלגוריתם עדיין עובד? האם הוא יציב?

תרגיל

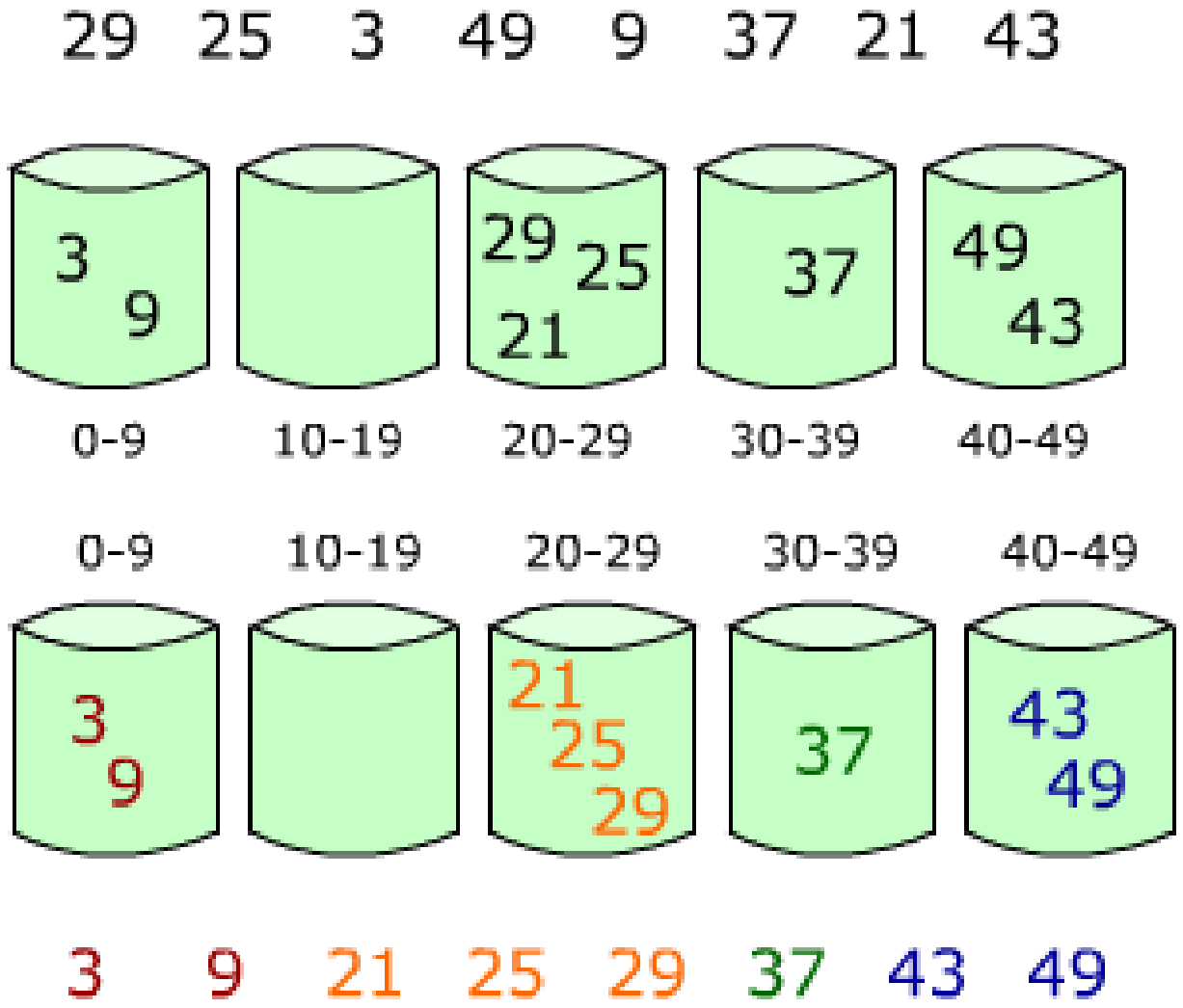
נניח שמשנים את לולאת ה- forהאחרונה באופן הבא:

For j=1 to len(A)

האם לאחר השינוי האלגוריתם עדיין עובד? האם הוא יציב?

פיתרון: האלגוריתם עדיין עובד אך לא יציב, מכיוון שכאשר יש 2 ערכים זהים, והכנסו כבר את הראשון, הבא יוכנס לפניו. לכן אם הלולאה פועלת בסדר עולה במערך איבר "מאוחר" יותר יוכנס לפני איבר "מוקדם" יותר.

BUCKET SORT



דוגמה מויקפדיה

BUCKET SORT

מיון דלי רץ בזמן ריצה ממוצע לינארי , הוא מניח שהקלט מורכב ממספרים המפולגים התפלגות אחידה.

למה אנו דורשים התפלגות אחידה?

BUCKET SORT

מיון דלי רץ בזמן ריצה ממוצע לינארי , הוא מניח שהקלט מורכב ממספרים המפולגים התפלגות אחידה.

למה אנו דורשים התפלגות אחידה?

על מנת לשמור על יעילות לינארית. במידה ולתא אחד נכנסו $O(n)$ איברים, מיונם ייקח $O(nlog(n))$. המעבר על שאר התאים הינו $O(m)$ ולכן נגיע לזמן ריצה $O(nlog(n)+m)$

בהנחה והפיזור אחיד ,נצטרך למיין m תאים שבכל אחד מהם $O\left(\frac{n}{m}\right)$ איברים.

למיין כל אחד מהם לוקח $O\left(\left(\frac{n}{m}\right)log\left(\frac{n}{m}\right)\right)$ וגם צריך לקחת בחשבון את המעבר על התאים (במידה וm יותר גדולה בסדר גודל מn)

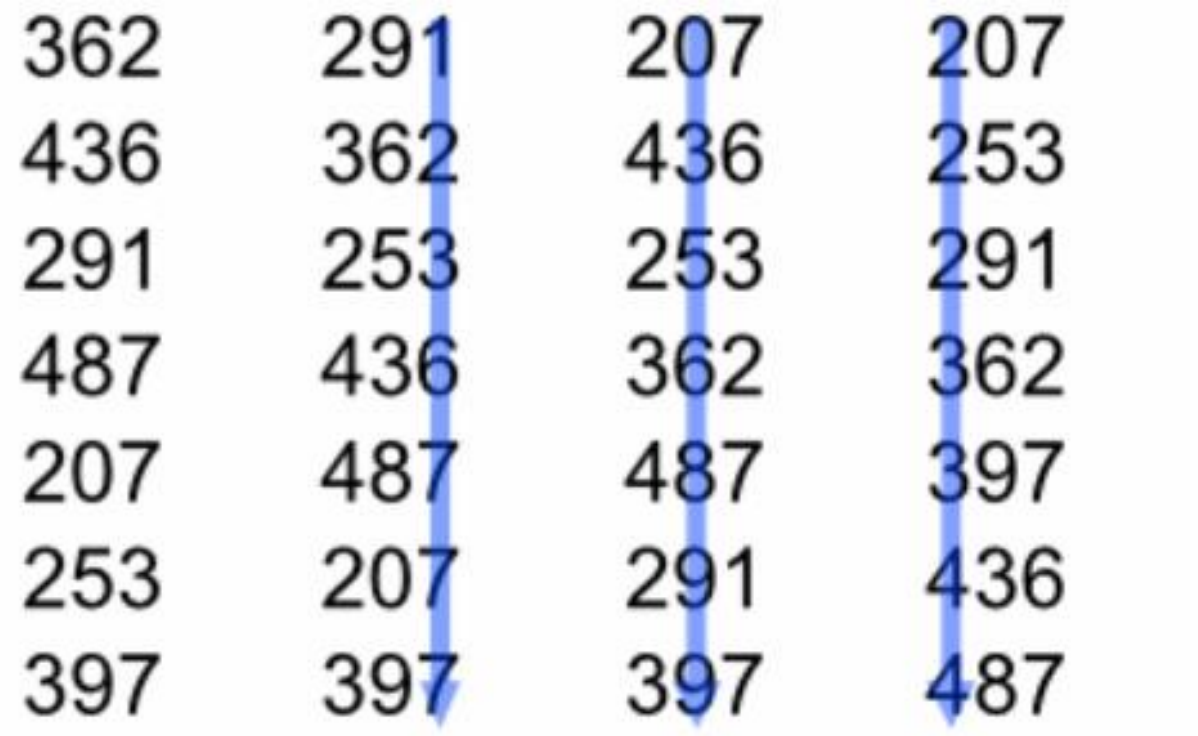
וודאו שאתם יכולים להגיע מפה ליעילות לינארית.

LSD RADIX SORT
LEAST-SIGNIFICANT-DIGIT RADIX SORT

מיון זה פותר בעיית מיון מספרים בעלי מספר ספרות , באופן הפוך לדרך בה אנו ממינים את המספרים, הוא מתחיל למיין לפי הספרה הפחות משמעותית ולאחר מכן ממיין לפי הספרות השונות כך שמערך ימוין לפי הספרה המשמעותית רק בסוף התהליך.

המיון בו משתמשים לצורך מיון כל סיפרה חייב להיות יציב, כך שאינו יחליף את הסדר היחסי בין איברים שווים , בעת המיון.

RADIX SORT LSD



RADIX SORT MSD
RADIX SORT MOST SIGNIFICANT DIGIT

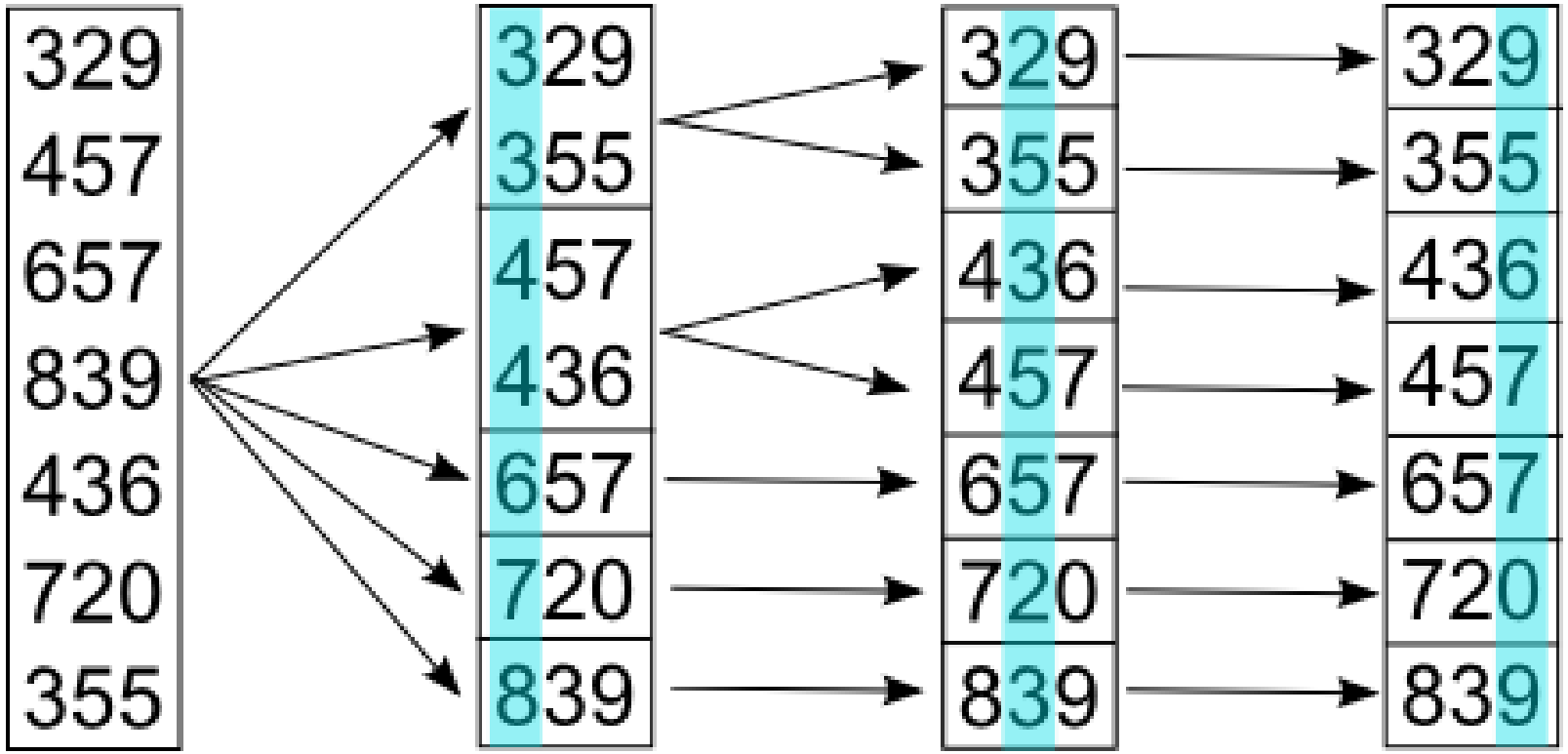
מיון זה יוצר מיון לקסיקוגרפי.

נמיין לפי הספרה הראשונה- נחלק לקבוצות.

ואז בכל אחת מהקבוצות מבצעים שוב פעם.

בניגוד ל LSD זהו מיון רקורסיבי!

RADIX SORT MSD



תרגיל

בהינתן רשימת מספרים טבעיים בין 1 ל100 עם 5000 ערכים,

אם המערך ממוין– באיזה אלגוריתם מאלגוריתמי המיון שלמדנו יהיה היעיל ביותר?

תרגיל

בהינתן רשימת מספרים טבעיים בין 1 ל100 עם 5000 ערכים,

אם המערך ממוין– באיזה אלגוריתם מאלגוריתמי המיון שלמדנו יהיה היעיל ביותר?

Insertion sort

המיון הינו $O(n)$ לכל מערך ממוין.

תרגיל– בעיית הבחירה–SELECT

נתונה רשימה `L` באורך `n` רוצים למצוא את האיבר ה- `i` בגודלו.

תרגיל– בעיית הבחירה

נתונה רשימה L באורך n רוצים למצוא את האיבר ה– i בגודלו.

הצעה לפתרון:

1. מיין את המערך.
2. החזר את איבר ה–i.

יעילות:$O(nlogn)$

תרגיל– בעיית הבחירה

נתונה רשימה L באורך n רוצים למצוא את האיבר ה– i בגודלו.

פתרון 2 (הסתברותי):

- בוחרים איבר $k \in L$ באקראי.
- מחלקים את L ל-2 חלקים – האיברים שגדולים מ-k (L_2) ואיברים שקטנים מ-k.
- אם $len(L_1) > i$ מחזירים $Select(L_1, i)$.
- אם $len(L_1) = i$ מחזירים את k.
- אחרת מחזירים $Select(L_2, k - len(L_1) - 1)$.

תרגיל–ניתוח זמן ריצה

במקרה הטוב: כל פעם שנבחר איבר ציר זה יהיה החציון ואז נקטין את המערך פי 2:

$$T(n) = \Theta(n) + T(\frac{n}{2})$$

לפי משפט המאסטר מקבלים ש- $T(n) = \Theta(n)$.

במקרה הגרוע: בכל שלב נקטין את המערך רק ב-1:

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

תרגיל– המשך ניתוח זמן ריצה

אורך תת הרשימה הארוכה ביותר עבור כל k שנבחר מתפלג בין $[\frac{n}{2}, n-1]$ לכן נקבל:

$$T(n) \leq \frac{1}{n}\left(2\sum_{k=\lceil\frac{n}{2}\rceil}^{n-1} T(k)\right) + O(n)$$

- אם n זוגי, אז כל אורך של קטע הכי ארוך מופיע פעמיים (לדוגמא אם n=6 אז כאשר k=2 וגם כאשר k=5 אורך הקטע הארוך יותר הוא 4). אם n אי-זוגי, כולם יופיעו פעמיים פרט לאחד מהם.

נראה ש- $\boldsymbol{T(n) = O(n)}$ באינדוקציה:

המשך

אורך תת הרשימה הארוכה ביותר עבור כל k שנבחר מתפלג בין $[\frac{n}{2}, n-1]$ לכן נקבל:

$$T(n) \leq \frac{1}{n}\left(2\sum_{k=\lceil\frac{n}{2}\rceil}^{n-1} T(k)\right) + O(n)$$

- אם n זוגי, אז כל אורך של קטע הכי ארוך מופיע פעמיים (לדוגמא אם n=6 אז כאשר k=2 וגם כאשר k=5 אורך הקטע הארוך יותר הוא 4). אם n אי-זוגי, כולם יופיעו פעמיים פרט לאחד מהם.

המשך ההוכחה יופיע כתרגיל בית!

גרפים

גרפים-הגדרות

גרף $G = (V, E)$ הוא קבוצת קודקודים V (Veritces, nodes) ואוסף של זוגות לא סדורים של קודקודים E (Edges).
אוסף = רב-קבוצה, כלומר מותרות חזרות, למשל $\{\{a, a, b\}\}$. כל זוג בE נקרא צלע.

גרף $G = (V, E)$ הוא גרף מכוון אם V היא קבוצת קודקודים וE אוסף זוגות סדורים של קודקודים.
(כל זוג כזה נקרא צלע או צלע מכוונת).

גרף $G = (V, E)$ לא מכוון הוא גרף פשוט אם E קבוצה של זוגות לא סדורים $(u, v) \in V^2$ של קודקודים
וגם לכל זוג כזה $u \neq v$.

יהי $G = (V, E)$ גרף כללי לא מכוון.
צלע $(u, u) \in E$ כאשר $u \in V$ נקראת לולאה, כלומר צלע מהקודקוד אל עצמו.

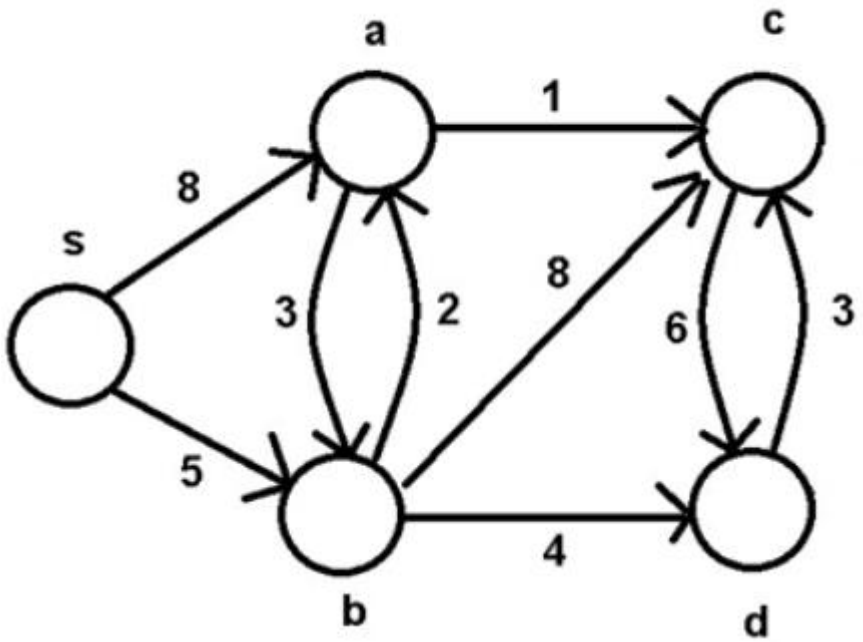
גרפים- המשך הגדרות

גרף הוא פשוט אם אין בו ריבוי צלעות ואין בו לולאות.
ריבוי צלעות פירושו שישנם שני קודקודים $u, v \in V$ כך שהצלע (u, v) מופיעה יותר מפעם אחת בE.
למשל, הגרף הסופי:

$$\begin{aligned} V &= \{u, v\} \\ E &= \{\{(u, v), (u, v)\}\} \end{aligned}$$

גרף ממושקל

גרף ממושקל: $G = (E, V, W)$ כאשר $W: E \to \mathbb{R}$.



ייצוג גרפים–רשימת שכנויות

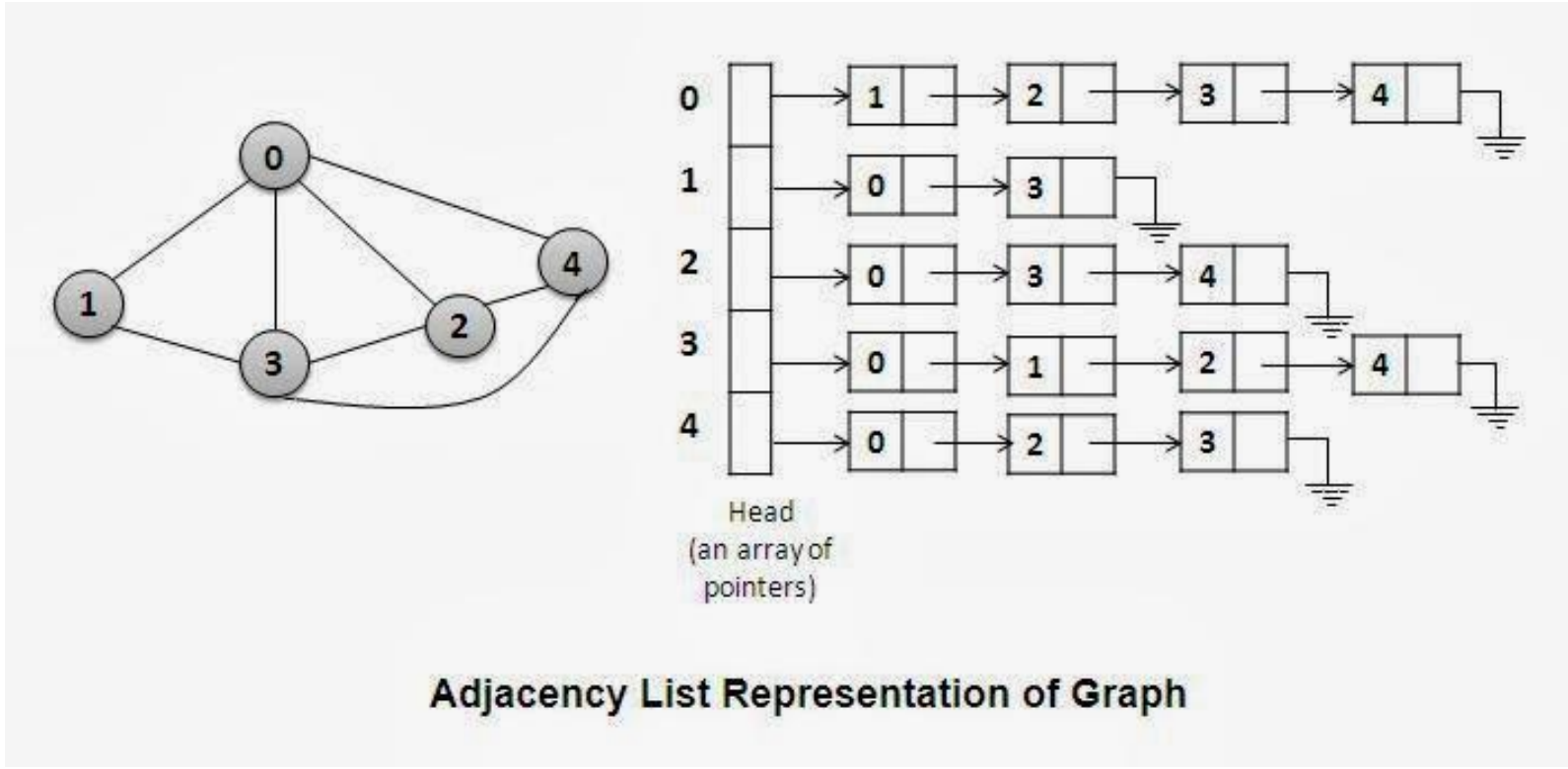
רשימת שכנויות – טוב לגרפים דלילים. מערך של קודקודים ולכל אחד יש מערך של מצביעים לקודקודים אליהם יש קשת.

ברשימת שכנויות:

- בגרף מכוון – סכום אורכי הרשימות הוא $|E|$. דוגמא – (u,v) מיוצג ברשימה של u בלבד.

- בגרף לא מכוון – סכום אורי הרשימות הוא $2|E|$. לדוגמא – (u,v) מיוצג ברשימה של u וגם של v.
- בגרף ממושקל – המשקל $w(u,v)$ של קשת מאוחסן עם קודקוד v ברשימה של u.
- חיסרון – יש צורך לעבור על כל הרשימה כדי לחפש האם קשת קיימת.

ייצוג גרפים- רשימת שכנויות



Adjacency List Representation of Graph

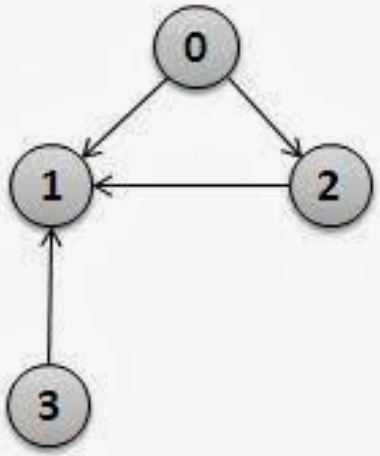
ייצוג גרפים- מטריצה

מטריצת שכנויות – טוב לגרפים צפופים.

במטריצת שכנויות:

- הקודקודים ממוספרים בצורה שרירותית $1,2,\ldots,|V|$. המטריצה A שמימדיה $|V|x|V|$ מייצגתאת גרף G וערכיה הם: $a_{ij} = \begin{cases} 1 & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$.
- נגדיר את A^T להיות המטריצה המשוחלפת של A. בגרף לא מכוון $A = A^T$.
- בגרף לא מכוון – אפשר לשמור רק חצי מטריצה (חוסך מקום).
- ייצוג בגרף ממושקל: $a_{ij} = \begin{cases} w(i,j) & (i,j) \in E \\ 0 \; or \; \infty & (i,j) \notin E \end{cases}$.

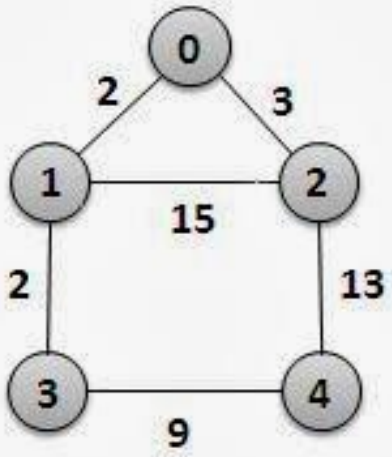
ייצוג גרפים- מטריצה



	0	1	2	3
0	0	1	1	0
1	0	0	0	0
2	0	1	0	0
3	0	1	0	0

Adjacency Matrix Representation of
Directed Graph

ייצוג גרפים- מטריצה



	0	1	2	3	4
0	0	2	3	0	0
1	2	0	15	2	0
2	3	15	0	0	13
3	0	2	0	0	9
4	0	0	13	9	0

Adjacency Matrix Representation of
Weighted Graph

תרגיל

נתונה רשימה לא ממוינת של קשתות.

נתונות m קשתות כאשר כל קשת היא זוג מספרים בטווח 1...n.

בנו רשימות סמיכויות ממוינות בזמן $O(n+m)$

תרגיל

נתונה רשימה לא ממוינת של קשתות.

נתונות m קשתות כאשר כל קשת היא זוג מספרים בטווח 1...n.

בנו רשימות סמיכויות ממוינות בזמן $O(n+m)$

פתרון: Radix sort

תרגיל– דוגמה

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

מיון לפי "ספרה" ימנית

(3,3)

(4,3)

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

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

מיון לפי "ספרה" שמלאית

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

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