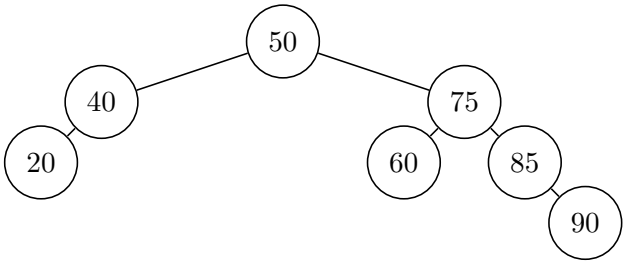
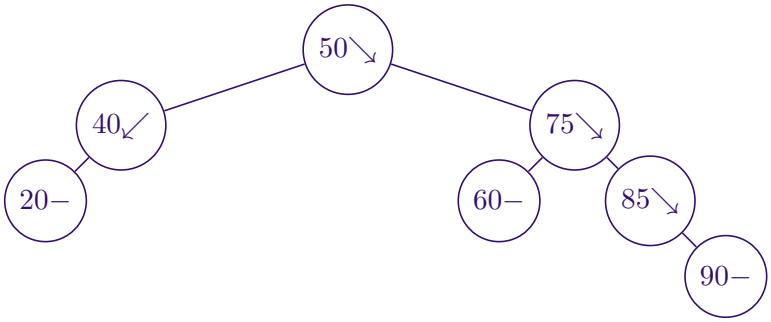
מבני נתונים
תרגיל 6 - פתרונות

1. לכל אחד מהסעיפים הבאים התחילו עם עץ ה AVL הבא:



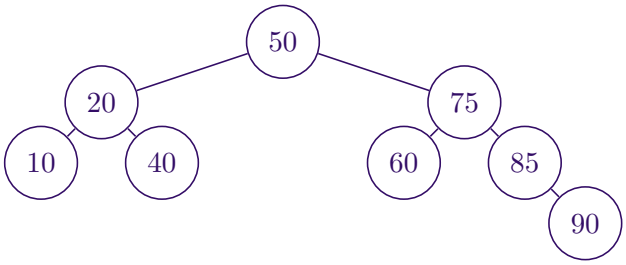
(א) סמנו על כל קודקוד האם הוא מאוזן לגמרי, ואם לא, אז לאיפה הוא נוטה.

פתרון:



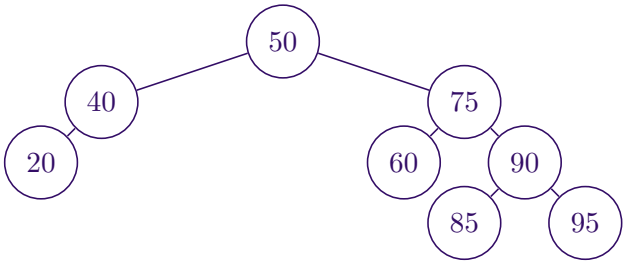
(ב) הוסיפו את 10 עץ המקורי.

פתרון:



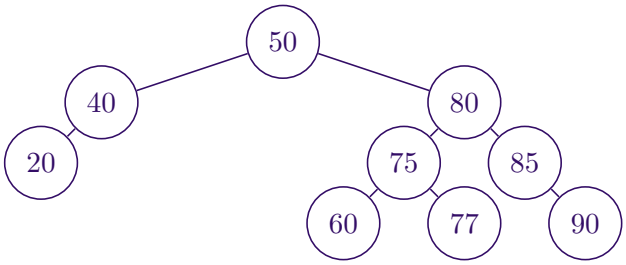
(ג) הוסיפו את 95 לעץ המקורי.

פתרון:



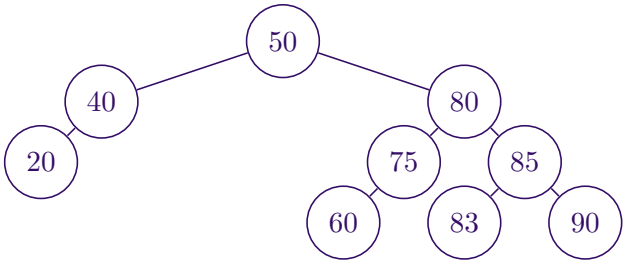
(ד) הוסיפו את 80 ואז את 77 לעץ המקורי.

פתרון:



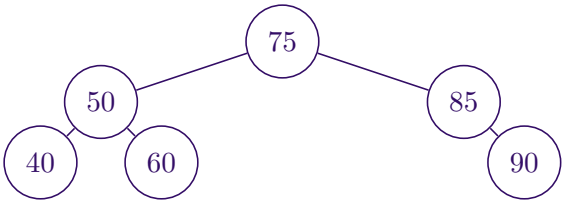
(ה) הוסיפו את 80 ואז את 83 לעץ המקורי.

פתרון:

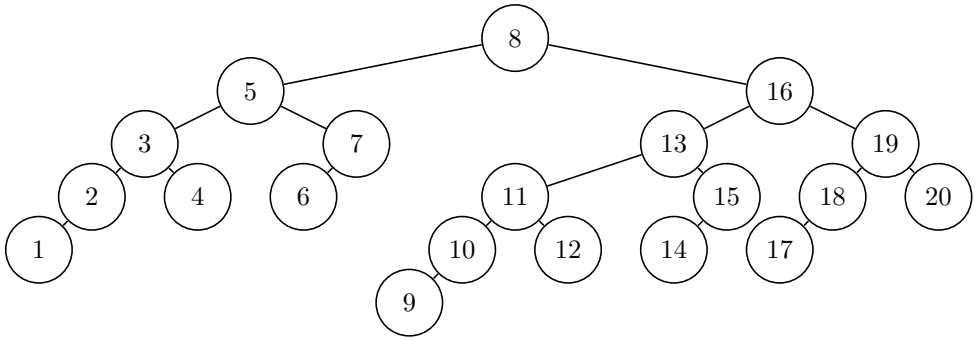


(ו) מחק את 20 מהעץ המקורי.

פתרון:

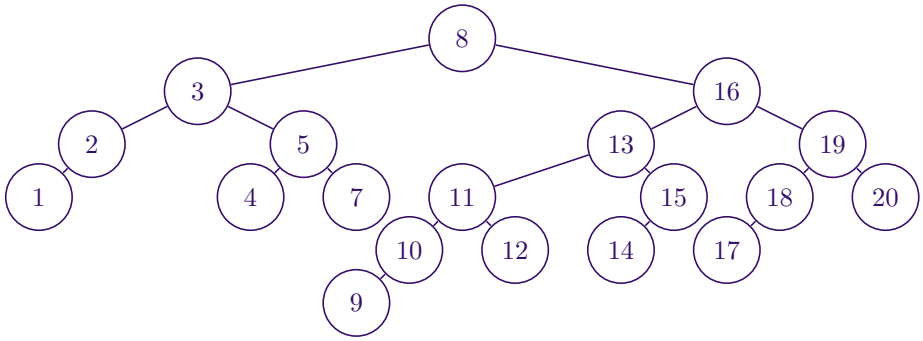


2. נתון עץ ה AVL הבא. מצאו קודקוד שאם תמחקו אותו, הוא יגרום לשני תיקונים בעץ. הכוונה לא לתיקון אחד של רוטציה כפולה, אלא שני תיקונים שונים.

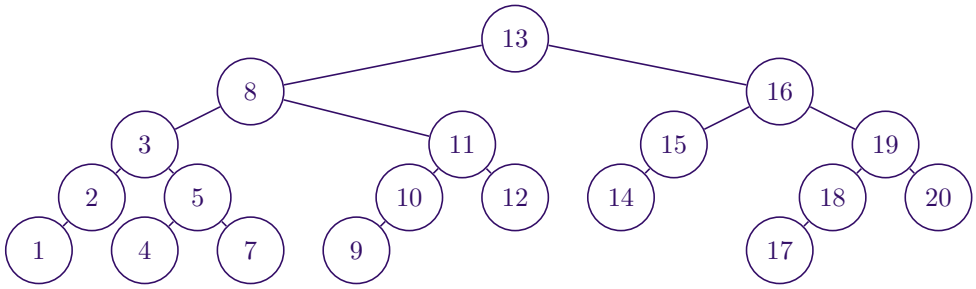


פתרון:

כל אחד מהקודקודים $4, 5, 6, 7, 20$ יעבוד. למעשה גם 3 ו־9 יעבדו אם כשמוחקים קודקוד מחליפים אותו בקודם שלו ולא בעוקב שלו. נדגים את פעולת המחיקה על הקודקוד 6. זה יגרום קודם לרוטציה ימינה סביב 5:



אז יווצר חוסר איזון ב 8, שיגרום לרוטציה ימינה סביב 16 ואז שמאלה סביב 8.



3. כתבו אלגוריתם המקבל עץ חיפוש בינארי ומסמן בתוכו את מצב האיזון לכל קודקוד. לכל קודקוד יהיה בנוסף לשדות הרגילים גם שדה

```
int balance;
```

ערך השדה הזה צריך להיות עומק תת־העץ הימני של הקודקוד פחות עומק תת־העץ השמאלי. אם העץ הוא עץ AVL חוקי אז בכל הקודקודים ערך זה צריך להיות 0 או 1 או -1. דאגו שהאלגוריתם שלכם יהיה יעיל ככל האפשר.

פתרון: כדי להקל על עצמנו, השיטה שלנו תחזיר את עומק העץ שכרגע היא טיפלה בו.

```
int calcBalance(Node t)
{
  if (t == null)
    return 0;
  int d1 = calcBalance(t.getLeft());
  int d2 = calcBalance(t.getRight());
  t.setBalance(d2 - d1);
  return 1 + Math.max(d1, d2);
}
```

זמן הריצה של אלגוריתם זה הוא $\Theta(n)$ כאשר n הוא מספר הקודקודים בעץ.

4. נשנה את ההגדרה של עץ AVL ונדרוש שהבדל העומק בין תת עץ ימין ותת עץ שמאל יהיה מקסימום 2 (ולא 1 כמו בהגדרה המקורית).

הראו כי אם עומק עץ כזה הוא h אז מספר הקודקודים בו הוא לפחות $2^{\frac{h}{3}} - 1$.

הדרכה: סמנו ב $T(h)$ את מספר הקודקודים המינימלי בעץ כזה בעומק h, והוכיחו באינדוקציה:

$$T(h) \geq 2^{\frac{h}{3}} - 1$$

פתרון: אז נוכיח באינדוקציה. אם $h = 0$ אז $T(h) = 0$ ומתקיים

$$T(h) = 0 = 2^0 - 1$$

נעבור למקרה הכללי. לעץ בגובה h יהיה תת־עץ אחד בגובה $h-1$ והשני בגובה לפחות $h-3$ - וזה המקרה הכי גרוע מבחינתנו כי אנחנו רוצים להראות שהעץ שלנו גדול. אז כמה קודקודים יש לנו? לפחות:

$$1 + T(h-1) + T(h-3) \geq 1 + (2^{\frac{h-1}{3}} - 1) + (2^{\frac{h-3}{3}} - 1) =$$
$$2^{\frac{h}{3}-\frac{1}{3}} + 2^{\frac{h}{3}-1} - 1 \geq 2^{\frac{h}{3}-1} + 2^{\frac{h}{3}-1} - 1 =$$
$$\frac{1}{2}2^{\frac{h}{3}} + \frac{1}{2}2^{\frac{h}{3}} - 1 = 2^{\frac{h}{3}} - 1$$

5. לעצי חיפוש בינאריים רגילים. נרצה להיות מסוגלים לענות בצורה יעילה על השאלה: כמה מספרים גדולים מ k ישנם בעץ?

```
int numLarger(Node t, int k);
```

תצטרכו להוסיף לכל קודקוד שדה נוסף size בו הוא שומר כמה קודקודים נמצאים תחתיו בעץ (כולל אותו). ממשו מחדש את שיטת ההוספה לעץ כך שתתמוך בעדכון של הערך הזה. לשם פשטות הניחו תמיד שהוספה של קודקוד היא תמיד של קודקוד חדש ואף פעם לא של ערך שכבר נמצא בעץ. לאחר מכן כתבו את השיטה numLarger.

פתרון: נשנה את השיטה להוספת קודקודים לעץ:

```
public void add(int k)
{
  Node newLeaf = new Node(k, null, null);
  newLeaf.setSize(1);    *
  if (root == null) {
    root = newLeaf;
    return;
  }
  Node t = root;
  while(true) {
    if (k < t.getValue()) {
      t.setSize(t.getSize() + 1);   *
      if (t.getLeft() == null) {
        t.setLeft(newLeaf);
        return;
      } else
        t = t.getLeft();
    }
    else if (k > t.getValue())
      t.setSize(t.getSize() + 1);   *
      if (t.getRight() == null) {
        t.setRight(newLeaf);
        return;
      }
      else
        t = t.getRight();
    } else return;
  }
}
```

נממש את השיטה החדשה - אבל נוסיף שיטת עזר קטנטנה קודם:

```
private int rightSize(Node t)
{
  if (t.getRight() == null)
    return 0;
  else
    return t.getRight().getSize();
}

int numLarger(Node t, int k)
{
  if (t == null)
    return 0;
  if (k == t.getValue())
    return rightSize(t);
  else if (k > t.getValue())
    return numLarger(t.getRight());
  else
    return 1 + rightSize(t) + numLarger(t.getLeft());
}
```

כיון שהשיטה שלנו כל פעם קוראת לרקורסיה רק על צד אחד ויורדת בעץ, אנו מקבלים שזמן הריצה הוא סדר גודל של עומק העץ, כלומר $O(d)$.