שאלה 1 -

במדעי המחשב בעיית הסכום החלקי (Subset Sum) היא בעיה חשובה בתורת

הסיבוכיות ובקריפטוגרפיה.

להלן הבעיה:

בהינתן קבוצה של מספרים שלמים, האם קיימת תת קבוצה לא ריקה שלה, שסכום

איבריה הוא אפס? לדוגמה: בהינתן הקבוצה {8,7,5,-2,-3} כקלט לבעיה,

התשובה תהיה חיובית, והקבוצה שסכומה אפס תהיה {5,-2,-3}.

נתבונן בבעיה פשוטה יותר של SUSU (subset sum).

נתון מערך ובו n מספרים שלמים, יש למצוא האם קיים זוג מספרים x,y כך

שסכומם הוא k.

אילו המערך היה ממוין, אזי זמן הריצה של הפתרון היה יכול להיות O(nlogn)

במקרה הגרוע (חשבו מדוע).

1. הציעו אלגוריתם שמשתמש בטבלת גיבוב ומוצא זוג מספרים {x,y} שסכומו k בזמן O(n) צפוי (בממוצע).
למשל, עבור המערך arr = {1,2,3,4,5,6,7,8,9,10} יש לכתוב אלגוריתם המשתמש בטבלת גיבוב המוצא האם קיים זוג מספרים שסכומם הוא 14.
יש לתאר את האלגוריתם ולנמק למה הוא עומד בזמן ריצה הדרוש.

1. נגדיר טבלת גיבוב בגודל של המערך T.

2. נרוץ על המערך arr וכל איבר בו x ניקח אותו   first loop

ונכניס לטבלת הגיבוב שלנו הערך x. ונשתמש בבדיקה לינארית
כדי לטפל בהתנגשות.

$h(v) = (v + i \mod) m$

אורך המערך arr

3. נרוץ עוד פעם על המערך arr וכל איבר בו x

נעשה פעולת חיפוש בפונקציה found = search(T, k-x)

נבדוק אם found != Null

נדפיס {x, T[found.key]}

ונחזיר True

4. אם הריצה הסתיימת ב- לולאה ה 3.

אז נחזיר false

2. הציעו אלגוריתם שמדפיס את כל הזוגות במערך בעלי אותו סכום נתון k.
למשל, עבור המערך arr הנתון בסעיף הקודם, האלגוריתם ידפיס את הזוגות {4,10} , {5,9} , {6,8} .

1. נגדיר טבלת גיבוב בגודל של המערך T.
2. נרוץ על המערך arr וכל איבר בו $\underline{x}$ ניקח אותו — first loop
ונכניס לטבלת הגיבוב שלנו הערך x. ונשתמש בבדיקה ליניארית
כדי לטפל בהתנגשות.

$h(v) = (v + i \bmod)\, m$

אורך המערך arr

3. נרוץ עוד פעם על המערך arr וכל איבר בו x
נעשה בטבלת חיפוש בפונקציה found = search(T, k-x)
נבדוק את found != Null
נדפיס {x, T[found.key]}

שאלה 2 -

נתונות שתי קבוצות מספרים A ו-B בגודל בהתאמה. הקבוצות אינן ממוינות.

הציעו אלגוריתם שבודק האם B היא תת קבוצה של A בזמן O(n) צפוי (ממוצע), כאשר n הוא מספר האיברים הכולל בשתי הקבוצות.

פתרון צריך להיכל הסבר במילים של הרעיון, המלווה בפסאודו-קוד.

הראו שאלגוריתם שהצעתם עומד בדרישות הזמן.

רמז: היעזרו בטבלת גיבוב.

נוסיף כל מספר ב A לטבלת גיבוב
ונרוץ על כל מספר בקבוצה B ונבדוק
אם הוא נמצא בטבלת הגיבוב אם
כן נמשיך למספר הבא אם לא נחזיר false
אם עברנו על איברי הקבוצה B הנמצאים בטבלה
נחזיר true.

```
function if_included (A, B):

    T [len(A)] <- null      // נגדיר טבלת גיבוב באורך
                            // של המערך A ונאתחל
                            // אותה ב null.
    for i <- 0 to len(A):
        insert (T, A[i])

    for i <- 0 to len(B):
        found = search (T, B[i])
        if (found == null)
            return false

    return true.
```

שאלה 3 -

נתונה טבלת הגיבוב הבאה המנוהלת בשיטת מיעון פתוח (open addressing) עם גיבוב כפול (double hashing).

0	1	2	3	4	5	6	7	8	9
		39	75		35	42	23		45

המפתחות הוכנסו לטבלה ריקה מלכתחילה תוך שימוש בפונקצית הגיבוב הבאה:

$h(k, i) = (h1(k) + i \cdot h2(k)) \bmod 10$

כאשר

$h1(k) = k \bmod 10$

$h2(k) = \lfloor k/10 \rfloor \bmod 10$

באיזה סדר הוכנסו המפתחות לטבלה? נמקו את תשובתכם.

k	h_1	h_2
39	9	3
75	5	7
35	5	3
42	2	4
23	3	2
45	5	4

35, 45, 42, 75, 23

39

35, 45, 39, 42, 75, 23

שאלה 4 -

הדגימו את הכנסתם של המפתחות {36, 19, 57, 7, 22, 4, 26, 41, 16} משמאל

לימין לטבלת גיבוב,

בגודל 11 בשיטת מיעון פתוח, כאשר $h_1(k) = k \bmod 11$

1. באמצעות בדיקה ליניארית
2. באמצעות בדיקה ריבועית כאשר $c_1 = 1$, $c_2 = 2$
3. באמצעות גיבוב כפול, כאשר $h_2(k) = 1 + k \bmod 10$

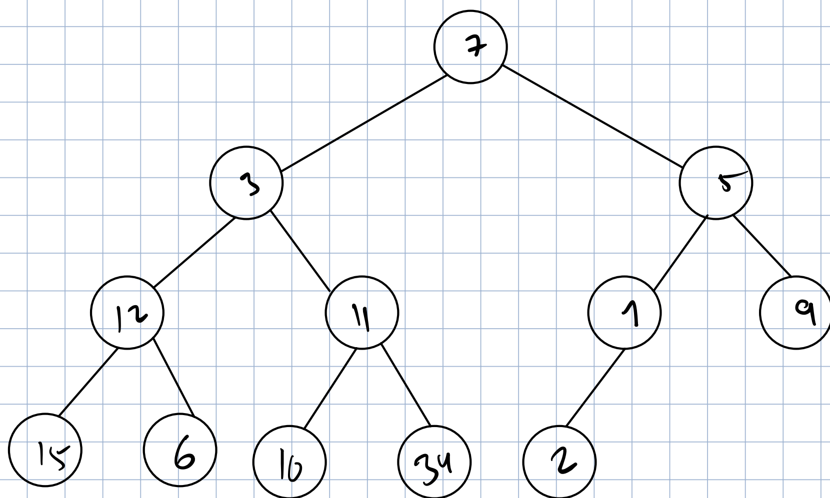
	בדיקה לינארית	בדיקה ריבועית	גיבוב כפול
0	22	22	22
1		7	
2	57	57	57
3	36	36	36
4	26	26	26
5	16	16	16
6	4	19	19
7	7	4	7
8	41	41	41
9	19		4
10			

שאלה 5 –

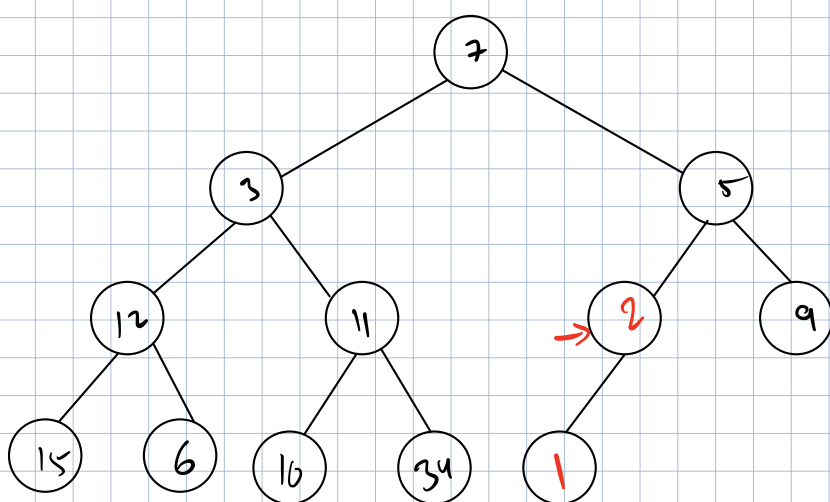
בנו ערמת מקסימום (Max-Heap) ממערך הקלט הבא:

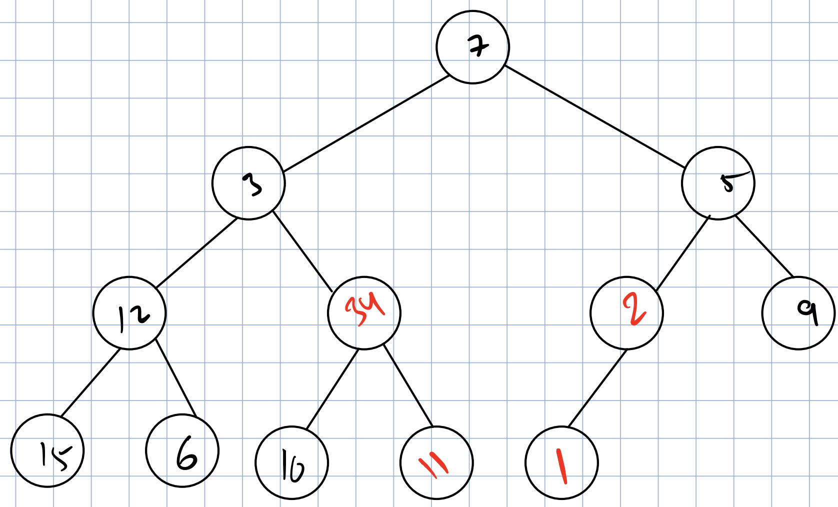
1	2	3	4	5	6	7	8	9	10	11	12
7	3	5	12	11	1	9	15	6	10	34	2

הראו הכנסה שלב אחר שלב להצגת ערימה סופית.

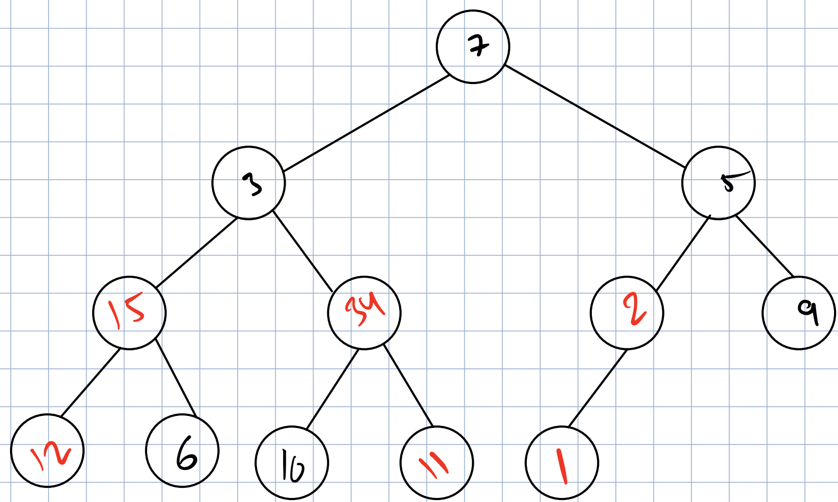


$$6 = \left\lfloor \frac{13}{2} \right\rfloor$$

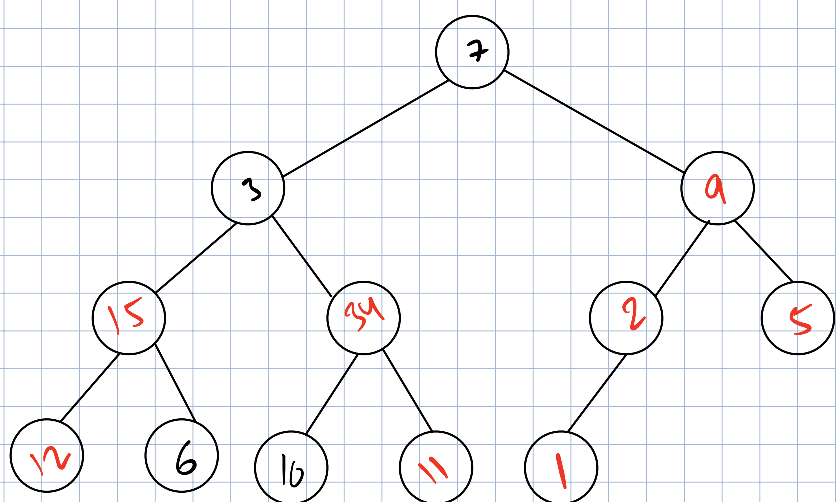




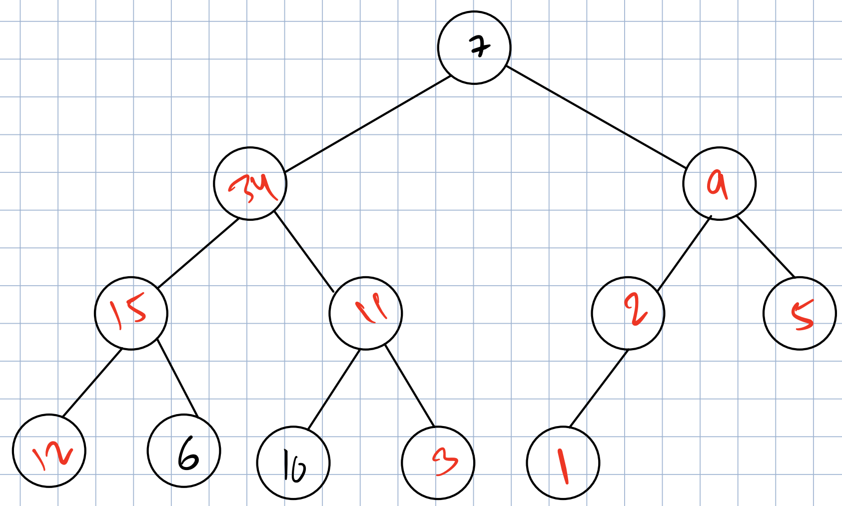
7
3
5
12
34
2
9
15
6
10
11
1

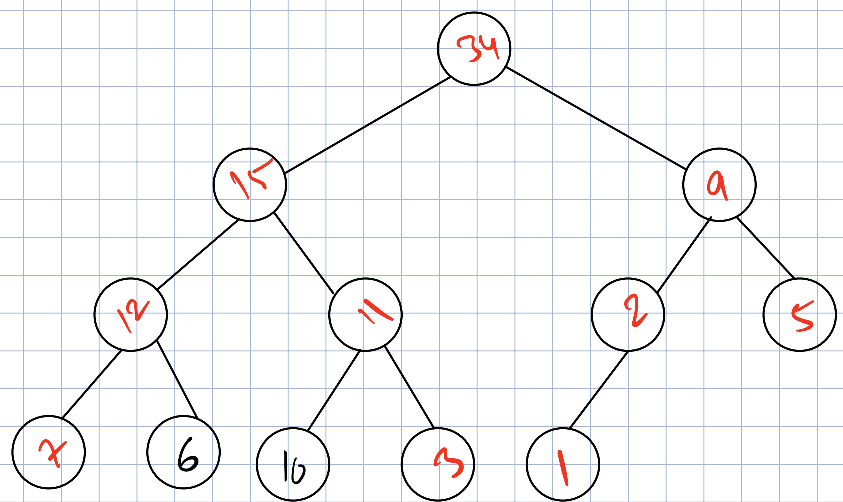


7
3
5
15
34
2
9
12
6
10
11
1



7
3
9
15
34
2
5
12
6
10
11
1


7
34
9
15
11
2
5
12
6
10
3
1


34
15
9
12
11
2
5
7
6
10
3
1