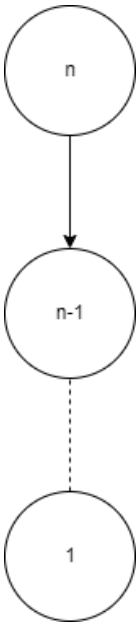
תרגיל בית 04 אביתר שמש 322623182.



שאלה 1 סעיף א 1) הרקורסיה פועלת בכל פעם על רשימה שקטנה באורכה ב-1 מהקריאה הקודמת. ננתח סיבוכיות של קריאה אחת:

```
def max_v1(L):
    if len(L) == 1:  O(1)
        return L[0]

    without_left = max_v1(L[1:]) O(n)

    return max(without_left, L[0])  O(1)
```

בסה"כ כל העבודה שנעשית בכל צומת היא לבדוק האם אורך הרשימה הוא 1(איבר יחיד והאחרון ברשימה המקורית). אם כן להחזיר אותו. אחרת, נעשה סלייסינג(במקרה הגרוע O(n), כאורך הרשימה, אך באופן כללי (O(size of slice)). לכן במקרה הגרוע בכל צומת עבודה של O(n).

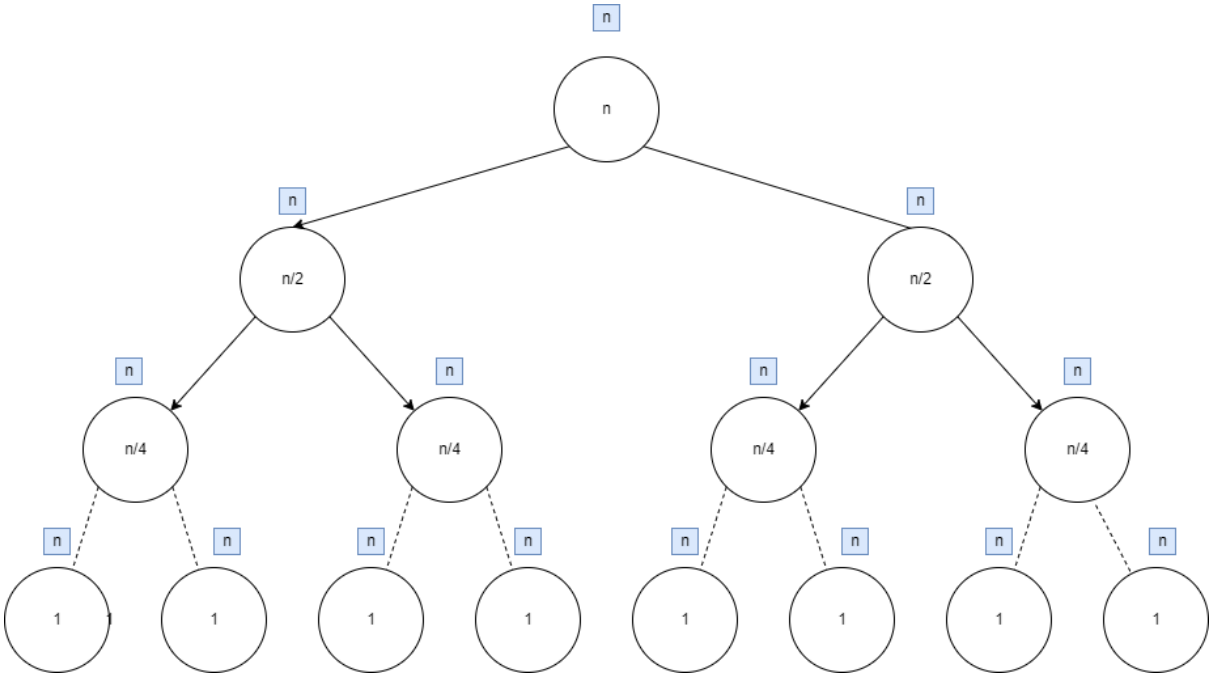
ובכל פעם כשנעלה מלמטה למעלה, נעשה מקסימום בין שני מספרים, המקסימום לאחר העלייה הראשונה הוא מקסימום בין L[-1] ל L[-2]. הגדול מבניהם יועלה לבדיקה אל מול L[-3] וכן הלאה.

לכן העבודה שנעשית בכל צומת היא O(n).

כמות הצמתים היא n.

לכן, סיבוכיות זמן הריצה היא $n*O(n) = O(n^2)$.

שאלה 1 סעיף א 2)



עומק העץ הוא log(n), כי כל צומת מתפצל ל-2 צמתים. לכן,log(n).

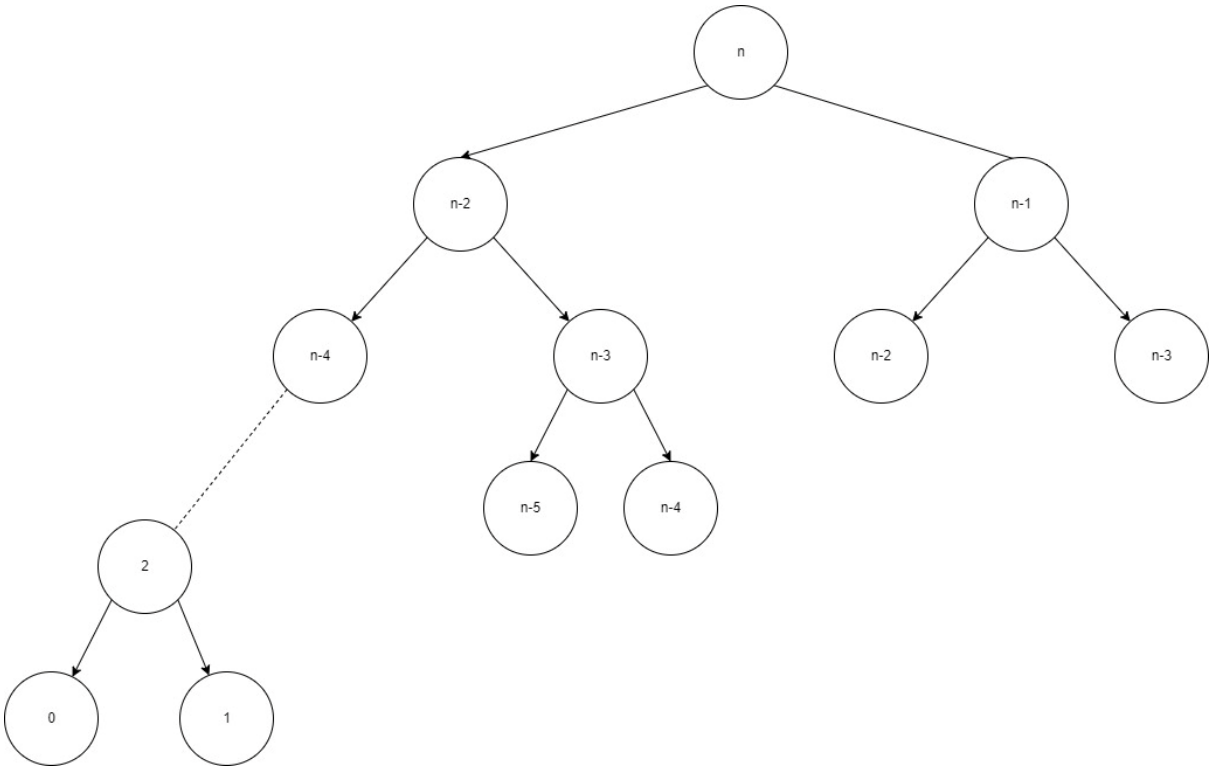
ננתח את הסיבוכיות בכל צומת.

```
def max_v2(L):
    if len(L) == 1:        O(1)
       return L[0]

    mid = len(L) // 2      O(1)
    first_half = max_v2(L[:mid])      O(n)
    second_half = max_v2(L[mid:])     O(n)

    return max(first_half, second_half)   O(1)
```

הסיבוכיות בכל שלב בעץ(שורה)

סיבוכיות זמן הריצה: log(n) * O(n) = O(nlog(n))

שאלה 1 סעיף ב 1)



שאלה 1 סעיף ב 2)

כמו שניתן לראות בציור הסכמטי, הענף שיורד עד לשורש העץ (השמאלי בכל צומת) יורד בקפיצות של 2.

לכן, אם n זוגי, עומק העץ הוא n/2.

הרצה עבור n=1000 עובדת ומתקבל הפלט:
7033036771142281582183525487718354977018126983635873274260490508715453711
8196933579742249494562611733487750449241765991088186363265450223647106012
0533741212738673391111981393731255987676900919022452453234035 01

אבל, החל מ-n מסויים ההרצה כבר לא תעבוד כי עומק הרקורסיה חוצה את הגבול שמוגדר במערכת ההפעלה(ניתן לשנותו כמו שראינו בהרצאה, אך רצוי שלא).

שאלה 1 סעיף ב 3)

```
def fib2_reverse(n, fib_dict):
    if n not in fib_dict: O(1)
        res = fib2_reverse(n-2, fib_dict) + fib2_reverse(n-1, fib_dict)   O(1)
        fib_dict[n] = res O(1)
    return fib_dict[n] O(1)
```

ופונקציית המעטפת:

```
def fibonacci2_reverse(n):
    fib_dict = {0:1, 1:1} # initial dictionary O(1)
    return fib2_reverse(n, fib_dict) O(1)
```

הסיבוכיות של הפונקציה ה"פנימית" fib2_reverse() היא: O(1)+O(1)+O(1)+O(1) = O(1).

לכן, הסיבוכיות של פונקציית המעטפת fibonacci2_reverse() היא O(1)+O(1) = O(1).

סך העבודה המתבצעת בכל צומת במקרה הגרוע: O(1). (בפיאצה נכתב להתייחס לבדיקת האם ערך במילון כ O(1) ולא O(n).

בכל דרגה של העץ, יש 4 צמתים לכל היותר (הודות לממואיזציה), לכן סיבוכיות זמן הריצה הכוללת:

O(1) * 4 * (n/2) = O(n)

שאלה 1 סעיף ג 1)

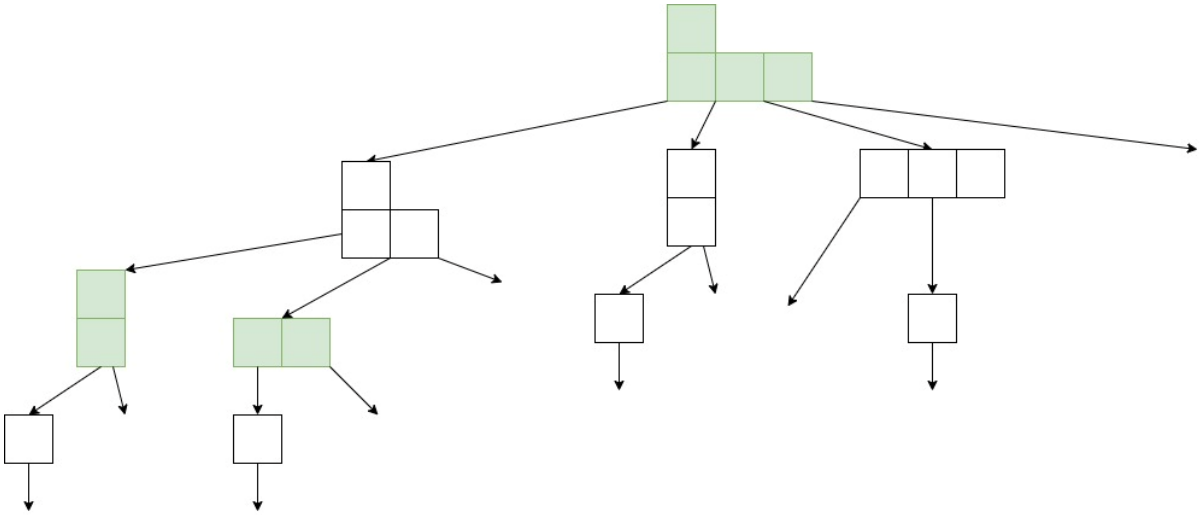
הפלט שיודפס עבור הפקודה בתרגיל הוא:

recommended move: [1,1,0] → [1,0,0]

recommended move: [2,0,0] → [1,0,0]

recommended move: [2,1,1] → [2,1,0]

ויוחזר True.



הרצת הפקודה תגרור הרצת סימולציה של המשחק עבור השחקן 0, כלומר, אילו מהלכים יש לבצע על מנת לגרום לשחקן 1 להפסיד.

המהלכים שהודפסו, הם אלו שעבור שחקן 0, יובילו להפסד וודאי של שחקן 1.

כמובן, שלא יודפס כל שלב עבור הלוח המקורי, אלא רק עבור לוח נוכחי.

שאלה 1 סעיף ג 3)

Input	winnable_mem()	winnable()
[5]*2	9.76541799999886e-05	0.002431689100000012
[5]*4	0.0006559518099999914	0.3180397490000002
[5]*8	0.017355747999999997	ההרצה לא הסתיימה(כמות קריאות רקורסיביות עצומה, ללא ממואיזציה)

שאלה 1 סעיף ג 4 1)

Input	Dict searches
[5]*4	454
[5]*8	8571
[5]*16	251125

שאלה 1 סעיף ג 4 2)

Input	Dict Successful Searches
[5]*4	378
[5]*8	7813
[5]*16	238351

שאלה 2 סעיף ב)

```
# 2a
def H_local(n, i, j):
    num = pow(2,n) O(n)
    if (n==0):  O(1)
        return 0
    if(i>=num//2 and j>=num//2): O(n)
        if(H_local(n-1,i-num//2,j-num//2) ==1):
            return 0
        return 1
    if i>=num/2: O(n)
        i = i-num//2 O(n)
    if j>=num//2: O(n)
        j = j-num//2  O(n)
    return H_local(n-1,i,j) O(n)
```

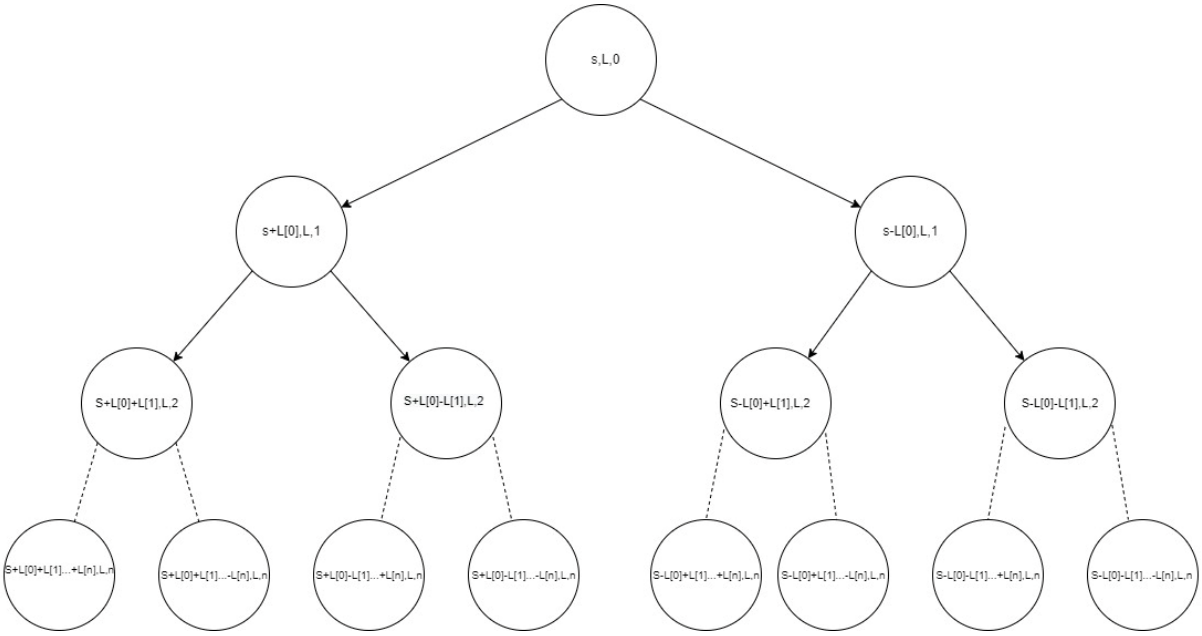
כל קריאה רקורסיבית דורשת O(n) עבודה(במקרה הגרוע).

כמות הקריאות הרקורסיביות: n קריאות(מ n עד 0).

לכן, הסיבוכיות הכוללת: $n*O(n) = O(n^2)$.

שאלה 3 סעיף א 2)

עץ רקורסיה:



ניתוח סיבוכיות:

```
# 3a
def can_create_once(s, L):
    def q3a_rec(s,L,i):
        if i == len(L) and s==0:    O(1)
            return True
        elif i== len(L): O(1)
            return False
        return q3a_rec(s+L[i],L, i+1) or q3a_rec(s-L[i], L, i+1)   O(1)

    return q3a_rec(s,L,0) O(2^n)
```

בפונקציה q3a_rec(), הסיבוכיות עבור צומת בודד: O(1).

כמות הצמתים: כל צומת מתפצל ל-2 , ולכן 2^n צמתים.

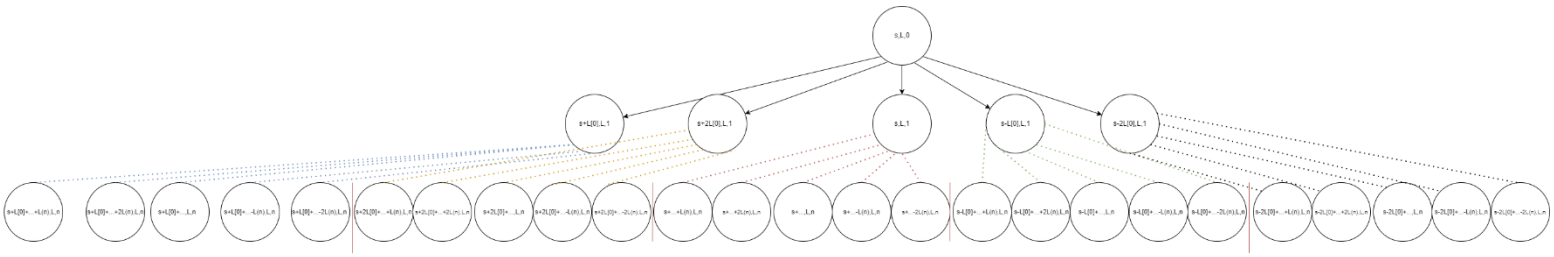
לכן סיבוכיות זמן הריצה הכוללת של q3a_rec(): $O(1)*2^n = O(2^n)$

לכן, סיבוכיות זמן הריצה של הפונקציה can_create_once() : הרצה אחת של הפונקציה q3a_rec() כלומר $O(2^n)$.

לכן, עונה על דרישות סיבוכיות זמן הריצה בשאלה.

שאלה 3 סעיף ב 2)

עץ רקורסיה:



(לא הצלחתי לצרף את זה כאן ושיראה טוב, מצרף כקובץ, יש ללחוץ כאן)

ניתוח סיבוכיות:

```
# 3b
def can_create_twice(s, L):
    def q3b_rec(s,L,i):
        if i == len(L) and s==0:   O(1)
            return True
        elif i == len(L): O(1)
            return False
        return q3b_rec(s+L[i],L, i+1) or q3b_rec(s+2*L[i],L, i+1) or q3b_rec(s-L[i], L, i+1) or q3b_rec(s-2*L[i], L, i+1) or q3b_rec(s,L, i+1)  O(1)

    return q3b_rec(s,L,0)  O(5^n)
```

בפונקציה q3b_rec(), כמות העבודה בכל צומת: O(1).

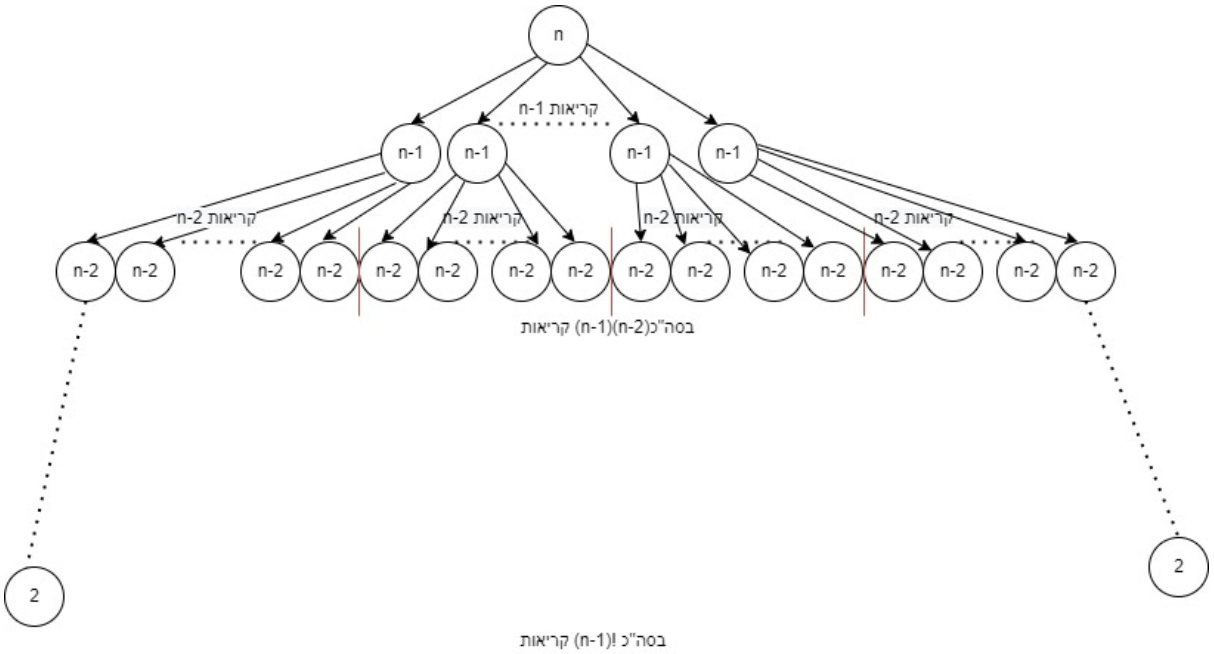
כמות הצמתים בפונקציה q3b_rec(): 5^n(כל צומת מתפצל ל 5 צמתים שונים).

לכן, סיבוכיות זמן הריצה הכוללת של q3b_rec(): $5^n*O(1) = O(5^n)$

לכן, סיבוכיות זמן הריצה הכוללת של can_create_twice(): $O(5^n)$.

לכן, עונה על דרישות סיבוכיות זמן הריצה בשאלה.

שאלה 3 סעיף ג 2)



שאלה 3 סעיף ג 3) מתחילים עם n איברים, ושולחים n-1 קריאות רקורסיביות. מכל קריאה כזו, שולחים n-2 קריאות רקורסיביות, וכן הלאה.

לכן, כמות האיברים בקריאה יורדת ב-1, וכמות הקריאות יורדת.

חשוב לציין כי עומק העץ הוא n-1, לכן !(n-1) צמתים.

סך העבודה בכל צומת הוא O(1).

לכן, העבודה הכוללת היא, O(1) * (n-1)! ≤O(n!)

שאלה 5)

המקרה הגרוע הוא המקרה בו b הוא מספר עם m ביטים שבכל ביט נמצא המספר 1(מכיוון ובמקרים אלו, בכל איטרציה תהיה לנו פעולת כפל). ניקח את מקרה זה וננתח אותו.

השמה של result = 1, O(1).

לולאת ה while תרוץ m פעמים, כי בכל פעם מורידים את הביט הימני מ-b.

במקרה הגרוע שלנו, ההשמה של result תתבצע m+1 פעמים, (כי בכל ביט, נמצא המספר 1).

באיטרציה הראשונה ב a יש n ביטים וב result יש ביט 1, לכן סיבוכיות המכפלה היא O(n+1) = O(n).

לכן, לאחר איטרציה זו, מספר הביטים ב result הוא n.

בעוד, ב a יש 2n ביטים. לאחר איטרציה זו, מספר הביטים ב result הוא 3n, וסיבוכיות המכפלה היא $o(n^2)$.

לכן, ניתן לרשום את הסיבוכיות הכוללת של פעולה זו לאחר m איטריציות כ:

$$\sum(2^i-1)n^2*(2^i)n^2= n^2\sum(2^i-1)(2^i)<= n^2\sum 2^i*2^i = n^2\sum 2^{2i} = n^2\sum 4^i = O(n^2*4^m)$$

ננתח את הפעולה a = a*a. באיטרציה הראשונה, ב a יש n ביטים, ולאחר ההכפלה יהיו 2n ביטים.

סיבוכיות הזמן של הפעולה היא $n*n = O(n^2)$.

באיטרציה השנייה, ב a יש 2n ביטים, ולאחר ההכפלה יהיו 4n ביטים.

סיבוכיות הזמן של הפעולה היא: $2n*2n = O(n^2)$.

ניתן לרשום את הסיבוכיות כסכום סדרה חשבונית, מ 1 עד m:

$$\sum (2^i)*n^2 * (2^i)*n^2 = n^2 * \sum 2^{2i} = n^2 * \sum 4^i = O(n^2* 4^m)$$

סיבוכיות הפעולה b= b//2 היא O(1) מהנחיות השאלה, ומבוצעת m פעמים, לכן O(m).

סיבוכיות הריצה הכוללת: $O(1)+O(n^2*4^m) + O(n^2*4^m) + O(m) = O(n^2*4^m)$