1 פתרון לבעיה הנתונה הוא פרמוטציה כלשהי של תת קבוצה בגודל k של הקבוצה הנתונה. לכן יש $\binom{n}{k} \cdot k! = n(n-1) \cdot \ldots \cdot (n-k+1)$ פתרונות אפשריים לקבוצה בת n איברים.

אם h הוא גובה עץ החלטה שלו l עלים של אלגוריתם כלשהו שפותר את הבעיה הנתונה אז:

$$(n-k+1)^k < n(n-1) \cdot \ldots \cdot (n-k+1) \leq l \leq 2^h$$

נניח ש-$k \leq n/2$ ונוציא $\lg$ משני האגפים:

$$h > \lg(n-k+1)^k = k\lg(n-k+1) > k\lg(n/2) = \Omega(k \lg n)$$

כלומר אפשר למצוא את $k \leq n/2$ האיברים הקטנים (גדולים) ביותר בסדר ממויין בזמן $\Omega(k \lg n)$.

אם $k > n/2$ תחילה נמצא את $n-k < n/2$ האיברים הגדולים, שלפי החלק הראשון של התשובה אפשר בזמן $\Omega((n-k)\lg n)$, ואת k האיברים הנותרים נמיין (עם מיון מיזוג למשל) בזמן $\Omega(k \lg n/2) = \Omega(k \lg n)$. סה"כ מקבלים $\Omega((n-k)\lg n) + \Omega(k \lg n) \underset{\downarrow}{=} \Omega(k \lg n)$.
כי $n-k < n/2 < k$

2 א האלגוריתם מתחיל בחישוב הסכום של כל זוג איברים במערך ושמירתו במערך עזר (נשמור גם את זוג האינדקסים של הזוג) שאותו ממיינים. על מערך הסכומים רצים עם שני מצביעים, אחד בהתחלת המערך ואחד בסופו. אם סכום האיברים ששני המצביעים מצביעים עליהם קטן מ-z אזי צריך לקדם את המצביע התחתון (כי המערך ממויין והאיבר הבא גדול שווה לו לאיבר הנוכחי). אם הסכום גדול מ-z אז נזיז את המצביע העליון לאיבר הקודם כדי להקטין את הסכום.

נמשיך ככה עד שהמצביעים חוצים אחד את השני או שהסכום שווה ל-z, במקרה זה עלינו לוודא שהאינדקסים של כל האיברים המעורבים שונים (במקרה שהם שווים נזיז את המצביע העליון).

מימוש האלגוריתם מחולק לשתי שגרות (כדי שנוכל לעשות שימוש בשגרות אלו בסעיף הבא): אחת שמחשבת את מערך סכומי העזר, והשניה שמוצאת שני איברים במערך זה (כלומר 4 אינדקסים שונים במערך המקורי) ששווים ל-z.

(לשם פשטות המימוש כדי לבדוק אם שתי קבוצות של אינדקסים חותכות אחת את השניה השתמשתי בסימון המקובל לחיתוך קבוצות - $\cap$)

```
procedure CalcSumsOfPairs(A)
    n ← length[A]
    pairs ← n(n − 1)/2
    create array B[1..pairs]
    c ← 1
    for i ← 1 to n − 1 do
        for j ← i + 1 to n do
            B[c] = A[i] + A[j]
            indices[B[c]] = i, j
            c ← c + 1
    return B
```

במערך בגודל n יש $\binom{n}{2} = \frac{n(n-1)}{2} = \Theta(n^2)$ זוגות שונים, לכן זמן הריצה של השגרה לעיל הוא $\Theta(n^2)$.

```
1: procedure FourSum(sums, z)
2:     i ← 1
3:     j ← length[sums]
4:     while i < j do
5:         sum ← sums[i] + sums[j]
6:         if sum < z then
7:             i ← i + 1
8:         else if sum > z then
9:             j ← j − 1
10:        else if indices[sums[i]] ∩ indices[sums[j]] ≠ ∅ then
11:            j ← j − 1
12:        else
13:            return indices[sums[i]], indices[sums[j]]
14:    return nil
```

הלולאה שמתחילה בשורה 4 רצה עם שני מצביעים על מערך נתון כאשר בכל איטרציה אחד המצביעים זז לכיוון השני. כל שאר הפעולות לוקחות זמן קבוע, לכן עבור מערך בגודל n זמן הריצה של השגרה הוא $\Theta(n)$.

כדי לפתור את הבעיה המקורית, נקרא לשגרות כך:

```
1: sums ← CalcSumsOfPairs(A)
2: if k < lg(length[A]) then
3:     radix sort sums
4: else
5:     merge sort sums
6: return FourSum(sums, z)
```

בשורה 1 אנחנו מחשבים את מערך סכומי הזוגות בזמן $\Theta(n^2)$. בשורות 2-5 ממיינים אותו: בהסתמך על גודל המערך וערכו של k אנחנו מחליטים באיזה מיון להשתמש. את המיון בשורה 3 נבצע לפי האלגוריתם המפורט בפתרון שאלה 18-ז במדריך הלמידה, עמ' 126 שלוקח $\Theta(n^2k)$, והמיון בשורה 5 לוקח $\Theta(n^2 \lg n^2) = \Theta(n^2 \lg n)$. סה"כ שורות 2-5 רצות בזמן $\Theta(n^2 \min(k, \lg n))$. לבסוף נקרא לשגרה $\textsc{FourSum}$ על מערך הזוגות, שלוקחת $\Theta(n^2)$.

לסיכום זמן הריצה הכולל הוא:

$$\Theta(n^2) + \Theta(n^2 \min(k, \lg n)) + \Theta(n^2) = \Theta(n^2 \min(k, \lg n))$$

2 ב נרוץ על המערך הנתון ועבור כל איבר $A[i]$ נחפש 4 איברים אחרים שסכומם $z - A[i]$ בעזרת $\textsc{FourSum}$ מהסעיף הקודם. נבדוק שכל האינדקסים של האיברים שנמצאו שונים, ובמידה וכן, מצאנו 5 איברים שונים שסכומם z.

```
1: procedure FiveSum(A, z)
2:     sums ← CalcSumsOfPairs(A)
3:     merge sort sums
4:     for i ← 1 to length[A] do
5:         indices ← FourSum(sums, z − A[i])
6:         if indices ≠ nil and i ∩ indices = ∅ then
7:             return i, indices
8:     return nil
```

ניתוח זמן הריצה של האלגוריתם:

$$\underset{\text{שורה 2}}{\underset{\downarrow}{\Theta(n^2)}} + \overset{\text{שורה 3}}{\overset{\uparrow}{\Theta(n^2 \lg n)}} + \underset{\text{לולאה שמתחילה שורה 4}}{\underset{\downarrow}{\Theta(n)}} \cdot \overset{\text{שורה 5}}{\overset{\uparrow}{\Theta(n^2)}} = \Theta(n^3)$$

3 נסתכל על הייצוג הבינארי של כל איבר במערך. כדי לייצג מספר בתחום $0..2^{n-1}$ דרושות n סיביות. לפי **למה 8.4** (עמ' 143 בספר הלימוד), בהינתן $r \leq n$ ניתן למיין את המערך ב-$\Theta((n/r)(n+2^r))$.

אם נבחר $r = \lfloor \lg n \rfloor$ אז נוכל למיין את המערך בזמן $\Theta((n/\lg n)(n+2^{\lg n})) = \Theta(\frac{n^2}{\lg n})$.

4 א אם המערך גדל ב-1 בכל פעם שהוא מלא, אז בהכנסה ה-k עלינו להעתיק $k-1$ איברים מהמערך הישן למערך החדש. כלומר זמן הריצה הוא:

$$\sum_{k=1}^{n-1} k = \Theta(n^2)$$

עבור n הכנסות.

```
1: procedure Push(S, x)
2:     if top[S] = length[S] then
3:         create array L[1..length[S] + 1]
4:         for i ← 1 to length[S] do
5:             L[i] ← S[i]
6:         top[L] ← top[S]
7:         S ← L
8:     top[S] ← top[S] + 1
9:     S[top[S]] ← x
```

ההבדל בין הגרסה הזו של השגרה לגרסאות בסעיפים הבאים הוא בשורה 3 שקובעת את גודל המערך החדש.

4 ב אם המערך גדל ב-k איברים בכל פעם שהוא מלא אז כשנכניס את האיבר ה-$k+1$ למחסנית נצטרך להעתיק k איברים, ובהכנסה ה-$2k+1$ נעתיק $2k$ איברים, כלומר זמן הריצה להכנסה של n איברים הוא:

$$k+2k+3k+\cdots+\left\lfloor\frac{n}{k}\right\rfloor k=\sum_{i=1}^{\lfloor\frac{n}{k}\rfloor} ik = k\cdot\frac{\lfloor\frac{n}{k}\rfloor(1+\lfloor\frac{n}{k}\rfloor)}{2}=\Theta(n^2/k)\quad(\underset{\downarrow}{=}\Theta(n^2))$$

עבור k קבוע

4 ג אם בהכנסה ה-k, $k-1$ הוא חזקה של 2 אז המערך מלא ועלינו להעתיק $k-1$ איברים. בהכנסה של n איברים, המערך יתמלא $\lfloor \lg n\rfloor$ פעמים, ולכן זמן הריצה הוא:

$$\sum_{k=0}^{\lfloor \lg n\rfloor} 2^k=\frac{2^{\lfloor \lg n\rfloor}-1}{2-1}=\Theta(n)$$

לפי התוצאות של סעיפים **א-ג**, הגדלת המערך פי 2 היא השיטה הכי יעילה.

5 את האיבר הראשון ב-L נחליף עם האיבר שנמצא באינדקס 1, את האיבר השני נחליף עם 2 וכן הלאה... עד שנגיע לסוף L. לאחר הזזת איבר כלשהו, יש לתקן את מצביע ה-$prev$ של האיבר שבא אחריו, ואת מצביע ה-$next$ של האיבר שבא לפניו שיצביעו למיקום החדש. השגרה FixPointers מטפלת בסידור המצביעים והיא נקראת עבור כל איבר בכל זוג שמוחלף.

בסוף המעבר על L, איבריו יתפסו את המקומות $1,2,\dots,m$ וכל שאר $n-m$ המקומות שייכים לרשימת הפנויים. כל שנותר לעשות זה לעדכן את L,F שיצביעו לראשי הרשימות (שייתכן שזזו כתוצאה מההחלפות).

כל הפעולות מלבד הלולאה בשורות 9-17 לוקחות זמן קבוע. הלולאה מתחילה בראש L ומסתיימת בסופה. לכן אם ב-L יש m פריטים, סיבוכיות השגרה היא $\Theta(m)$.

```
1:  procedure FixPointers(x)
2:      if prev[x] ≠ nil then
3:          next[prev[x]] ← x
4:      if next[x] ≠ nil then
5:          prev[next[x]] ← x
6:  procedure Compactify-List(L, F)
7:      curr ← L
8:      i ← 1
9:      while curr ≠ nil do
10:         if curr ≠ i then
11:             Swap(next[curr], next[i])
12:             Swap(key[curr], key[i])
13:             Swap(prev[curr], prev[i])
14:             FixPointers(curr)
15:             FixPointers(i)
16:         curr ← next[curr]
17:         i ← i + 1
18:     if L ≠ nil then
19:         L ← 1
20:     if F ≠ nil then
21:         F ← i
```