**ממן 11 – מבני נתונים ומבוא לאלגוריתמים – ברנדס איתי ת.ז.**

בפיתרון התרגילים אני מניח שבRAM שבו אנו עובדים, האינדקס הראשון במערך הוא 1 והאחרון הוא N.

**שאלה 1:**

השגרה הרקורסיבית:

BINARY-DOUBLE-SEARCH(A, p, r, v(

1 if p > r

2 then return NIL, NIL

3 q =

4 if v < A[q]

5 then return BINARY-DOUBLE-SEARCH(A, p, q-1, v)

6 else if v > A[q]

7 then return BINARY-DOUBLE-SEARCH(A, p+1, r, v)

8 else

9 ► calculate i value

10 k 🡨 q

11 while k-1 > 0 and A[k-1] == v

12 do k 🡨 k-1

13 i 🡨 k

14 ► calculate j value

15 k 🡨 q

16 while k+1 <= length[A] and A[k+1] == v

17 do k 🡨 k+1

18 j 🡨 k

19 return i, j

השגרה הראשית (קריאת ההפעלה):

BINARY-DOUBLE-SEARCH)A, 1, length[A], v(

**נוכיח את נכונות האלגוריתם.**

האלגוריתם הוא אלגוריתם חיפוש בינארי הפועל בסיבוכיות של . בשורות 1-2 מתבצע כלל היציאה הרקורסיבית, במידה שחרגנו מגבולות המערך (ומשתמע מכך שהערך לא נמצא במערך, ועל כן אנו מחזירים NIL,NIL).

בשורה 5 אנו מחשבים את אמצע המערך.

בשורות 4-7 אנו בודקים מצב בו הערך שלנו קטן או גדול מהאיבר (האמצעי) שאנו נמצאים בו כרגע. לפי כך אנו מחליטים האם לפעול רקורסיבית על החצי הראשון או החצי השני של המערך (בהתאמה), להוציא את האיבר האמצעי שבו אנו נמצאים כרגע. זה אפשרי משום שהמערך ממוין.

לשורה 8 אנו מגיעים רק במצב שבו הגענו להצלחה – האיבר שבו אנו נמצאים הוא האיבר הדרוש. במקרה זה עלינו לראות האם ישנם איברים נוספים עם אותו ערך. אם יהיו כאלה – הם יהיו סמוכים לאיבר שאנו בו כרגע, משום שהמערך ממוין.

בשורות 9-12 אנו משתמשים במשתנה k כדי לנוע אחורנית במערך. הוא מאותחל להיות האיבר שבו אנו כרגע (שהרי הוא אינדקס שבו הערך נמצא). אנו ממשיכים בלולאה כל עוד אנו ממשיכים לקבל ערכים שזהים לערך אותו אנו מחפשים, או כאשר הגענו לסוף המערך. בסוף הלולאה, ערך k שנמצא הוא אינדקס המערך השמאלי ביותר שבו נמצא האיבר המבוקש. בשורה 13 נכניס את התשובה אל תוך המשתנה i.

בשורות 14-17 נשתמש באותה לוגיקה. נשתמש במשתנה k שיאותחל להיות האיבר בו אנחנו נמצאים כרגע, ואנו נעים קדימה במערך כל עוד נמשיך לקבל ערכים שזהים לערך אותו אנו מחפשים, או כאשר הגענו לסוף המערך. בסוף הלולאה, ערך k שנמצא הוא אינדקס המערך הימני ביותר שבו נמצא האיבר המבוקש. בשורה 18 נכניס את התשובה אל תוך המשתנה j.

בשורה 19 אנו מחזירים את הi, j שנמצאו והשגרה מסתיימת בהצלחה.

**שאלה 2:**

השגרה הרקורסיבית בסיבוכיות של :

LOCAL-MAX-SEARCH(A, p, r(

1 q =

2 ►if A is empty, or has one element, it doesn’t have a local maximum.

3 if length[A] <= 1

4 then return NIL

5 ► if we're in the head, and we have a local minimum

6 if q == 1 and A[q] >= A[q+1]

7 then return A[q]

8 ► if we're in the tail, and we have a local minimum

9 else if q = length[A] and A[q-1] <= A[q]

10 then return A[q]

11 ►if we're in the middle, and we have a local minimum

12 else if A[q-1] <= A[q] and A[q] >= A[q+1]

13 then return A[q]

14 ► otherwise, we're not at the local minimum, but it must exist on the side of the array that is going up.

15 if A[q] <= A[q+1]

16 ► going right

17 then return LOCAL-MAX-SEARCH(A, q+1, r)

18 else

19 ► this happens only when A[q-1] >= A[q]. going left

20 then return LOCAL-MAX-SEARCH(A, p, q-1)

השגרה הראשית (קריאת ההפעלה):

LOCAL-MAX-SEARCH )A, 1, length[A](

**שאלה 3:**

על מנת להראות את נכונות האלגוריתם עלינו להראות שפעולת האלגוריתם איננה חורגת מגבולות המערך, ושהפלט שלה ממוין.

ראשית האלגוריתם מתחיל בהגדרת גבולות המערך. הוא מגדיר את ההתחלה באיבר 0 ואת הסוף באיבר length[A] +1. על פניו יש כאן חריגה מן המערך, אך בראשית הלולאה החיצונית (שורות 5-6) מקודם הפרמטר הראשון ומונמך הפרמטר השני, טרם הגישה למערך, ולכן לא מתבצעת חריגה.

באלגוריתם לולאה חיצונית ו2 לולאות פנימיות. נוכיח את נכונותם בעזרת הגדרת שמורת, אתחול, תחזוק וסיום הלולאות.

הלולאה הפנימית הראשונה (שורות 7-12):

* **שמורת הלולאה** – בתחילת כל איטרציה של לולאת ה-for, תוכן התא A[i] גדול יותר מכל איברי A[left..i-1].
* **אתחול** – לפני האיטרציה הראשונה, i=left, ולכן המערך A[left..left-1] ריק, ולכן A[i] גדול ממנו במובן הריק.
* **תחזוקה** – נניח ששמורת הלולאה מתקיימת בהתחלת האיטרציה ה-k, כלומר, תוכן התא A[left+k-1] גדול יותר מכל איברי A[left..left+k-2]. הלולאה הפנימית מעבירה את האיבר הגדול ביותר ב- A[left..left+k] אל התא A[left+k]. כלומר, שמורת הלולאה מתקיימת גם לפני האיטרציה ה-(k+1).
* **סיום** – אחרי ביצוע האיטרציה האחרונה, האיבר הגדול ביותר בA[left..right] יהיה מצוי בA[right].

הלולאה הפנימית השניה (שורות 16-21):

* **שמורת הלולאה** – בתחילת כל איטרציה של לולאת ה-for, תוכן התא A[i+1] קטן יותר מכל איברי A[i+2..right].
* **אתחול** – לפני האיטרציה הראשונה, i=right-1, ולכן המערך A[right+1..right] ריק, ולכן A[i+1] קטן ממנו במובן הריק.
* **תחזוקה** – נניח ששמורת הלולאה מתקיימת בהתחלת האיטרציה ה-k, כלומר, תוכן התא

A[right-k+1] קטן יותר מכל איברי A[right-k+2..right]. הלולאה הפנימית מעבירה את הקטן ביותר ב- A[right-k+1..right] אל התא A[right-k]. כלומר, שמורת הלולאה מתקיימת גם לפני האיטרציה ה-(k+1).

* **סיום** – אחרי ביצוע האיטרציה האחרונה, האיבר הקטן ביותר בA[left..right] יהיה מצוי בA[left].

הלולאה החיצונית (שורות 3-23):

* **שמורת הלולאה** – בתחילת כל איטרציה של לולאת ה-while, התת-מערכים A[1..left] ו-A[right..length[A]] ממוינים.
* **אתחול** – לפני האיטרציה הראשונה, left=0, right=length[A]+1, ולכן המערכים A[1..left] ו-A[right..length[A]] ריקים, ולכן שמורת הלולאה מתקיימת באופן ריק.
* **תחזוקה** – נניח ששמורת הלולאה מתקיימת בהתחלת האיטרציה ה-k, כלומר המערכים A[1..k-1], A[length[A]-k+2..length[A]] ממוינים. כפי שראינו, הלולאות הפנימיות מעבירות את האיבר הקטן ביותר ב-A[k..length[A]-k+1] אל A[k] ואת הגדול ביותר אל A[length[A]-k+1], כלומר, שמורת הלולאה מתקיימת גם לפני האיטרציה ה- (k+1).
* **סיום** – אחרי הביצוע האיטרציה האחרונה (שתהיה במקרה הרע ביותר האיטרציה ה-, אך עלולה להסתיים גם לפני כן אם המערך ממוין בשלב מוקדם יותר), המערך כולו יהיה ממוין.

בנוסף, בשורות 4,12,13-15,21-23 קיימת אופטימיזציה אשר דואגת לשגרה לחזור כאשר אחד הלולאות הפנימיות לא משנה דבר במערך, משום שאז פירוש הדבר הוא שהמערך כבר ממוין.

**שאלה 4:**

מתקיים:

*נסביר כל צעד בהסתמך על הגדרת הסימונים האסימפטוטים בעמוד 29 במדריך הלמידה:*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*

*כי לכל מתקיים*