

**פתרון שאלה 1 –**

תחילה נציין שאנחנו עוסקים בערימת **מקסימום** לכן כל השגרות הרלוונטיות יהיו לערימת מקסימום, באופן די דומה אפשר לבצע את אותן פעולות לערימת מינימום.

כמו כן נגדיר :

ייצוג חדש – בשורש שומרים את הערך האמיתי ובכל צומת אחר נחזיק את ההפרש בין ערך אביו לבין ערכו.

ייצוג ישן – הערכים האמתיים של כל איבר בערימה.

**סעיף א' –**

אחרי מעבר לייצוג החדש כל ערך בעץ מלבד השורש מוגדר להיות   
כלומר כל ערך מחזיק את ההפרש בין ערך אביו לבין ערכו.  
אם ברצוננו לעדכן במספר פעולות קבוע את ערך **כל** התאים עלינו לעדכן את .  
נוכיח כי בחזרה לייצוג הישן כל הערכים יתעדכנו כך ש לכל מתקיים   
באינדוקציה על ערך התא :  
**בסיס** – בעבור מתקיים היות ונפעיל את השגרה ADD-TO-KEYS(A,1).  
**הנחה** – לכל מתקיים   
**צעד** – נוכיח שבעבור A[n] מתקיים -  
ערך התא במקום ה היות ו Parent[n] במיקום נמוך יותר(מבחינת אינדקס) מהנחת האינדוקציה נובע כי Parent[n] +C ולכן מתקיים כי

בסדר

**סעיף ב' –**

השגרות שיצרתי עובדות כמכלול, ותלויות אחת בשנייה. כל שגרה שמימשתי באופן שונה כתבתי את המימוש החדש שלה, שגרות שלא נעשה מימוש חדש לא נכתבו כלל.

כמו כן אציין שהשגרה Heapsort בפועל צריכה להחזיר מערך ממוין מהגדול לקטן, אך היות ואנחנו עוברים לייצוג הפרשים, הערכים שיכילו התאים במערך לא יהוו אינדיקציה אמיתית האם המספר הוא מקסימלי או מינימלי מלבד הערך האמיתי של השורש. לכן הנחתי כי מהות המערך הממוין אינה משמעותית בשאלה זו ולכן את השגרה HEAPSORT עצמה לא מימשתי באופן שונה.

**Build-MAX-HEAP** – שגרה זו אחראית ליצור ערימה מקסימלית, לכן תחילה השגרה תיצור ערימה מקסימלית ורק לאחר מכן נהפוך אותה לערימת הפרשים. לולאת ה- for בשורות 4-10 אחראית להפוך את הערמה לערימת הפרשים. זמן הריצה של השגרה לא יפגע שכן זמן הריצה המקורי הוא והוספנו לולאה שזמן הריצה שלה תלוי בגודל הקלט ולכן; זמן הריצה -

**BUILD-MAX-HEAP(A)**

1. Heap-size[A]
2. **For**
3. **do**
4. **For**

7. **If**
9. **If**

**MAX HEAP INSERT** -   
השגרה לא תעבור שינוי מהותי, היא מקבלת פשוט ערמה B המיוצגת ע"י ערכים בוליאניים (T/F) בלבד, וקוראת לשגרה HEAP INCREASE KEY.

**HEAP INCREASE KEY** -  
השגרה מקבלת ערך מפתח, ואינדקס שאותו אנחנו רוצים לשנות או לחילופין להוסיף איבר לערימה. שורה מס' 3 קוראת לשגרה Restore Route שאחראית לאחזר את כל המסלול מאינדקס עד לראש העץ(השורש). אחר שהסתיימה השגרה Restore-Route בוודאות כל הצמתים והבנים הישירים שלהם מאוחזרים לערכיהם המקוריים(הוכחת נכוחות למטה).  
לאחר מכן לולאת ה while בשורות 5-7 זהות לשגרה המקורית, אנחנו מסדרים את הערימה מחדש כך שאם הערך החדש גדול מערך אביו נחליף ביניהם.  
ולבסוף בלולאה בשורה 9-16 מאחזרים את הערכים ואת הבנים הישירים שלהם לייצוג ערימת ההפרשים.

זמן ריצה -

1. **If**
2. **then error** “new key is smaller then current key”
3. **while**
4. **Do** exchange
6. **while**

9. **If**
11. **If**

**RESTORE-ROUTE -**

אחראית לאחזר את ערכי העץ מנקודה מסוימת עד השורש –  
שגרה זו פועלת בשיטת הפרד ומשול נוכיח נכונות באינדוקציה על גובה העץ -   
**בסיס** – על גובה עץ 1, כלומר h=1 , נסמן ראש העץ ב- X ונניח כי אזי התנאי בשורה אחד יעבור ונקרא רקורסיבית ל- ולכן כעת התנאי בלולאה אחת לא יעבור היום ו- יתבצע שחזור לבנים בשורות 3-8 ונתקפל חזרה כאשר כעת בגלל שאנחנו ברמת העלים שורות 3-8 לא יתבצעו ובסיום נקבל עץ בגובה 1 שאיבריו מכילים את ערכם המקורי.  
**הנחה** – נניח בעבור עץ בגובה h-1 ונוכיח בעבור עץ בגובה h.  
**צעד -** מהנחת האינדוקציה נובע שכל ערכי העץ ברמות h-1 מאוחזרים ולכן ערכו של השורש ברמה h-1 הוא ערכו המקורי, לפיכך כל שנותר הוא לבצע את שורות 3-8 ולאחזר את השורשים של העץ.  
בסיום מתקבל מסלול שכל ערכיו מאוחזרים לערכיהם האמיתיים כולל הבנים הישירים של כל צומת.



**RESTORE-VAL**

שגרה זו אחראית להחזיר את הערך האמיתי של הצומת שהוזן לה, ע"י חיסור כל הערכים מהצומת כלפי מעלה עד שנגיע לשורש שמחזיק את ערכו האמיתי, ולבסוף נחזיר את התוצאה בערך מוחלט (היות ואנחנו מאחזרים בכיוון ההפוך נקבל את הערך הנגדי).

זמן ריצה –

נכונות האלגוריתם נובעת מכך שערך המפתח הוא ההפרש בינו לבין אביו, וערך המפתח של אביו של הוא ההפרש בינו לבין אביו, ולכן ערך המפתח יכול להיות מיוצג ע"י ההפרשים בין שורש העץ למסלול שמוביל ל- כלומר –

נסמן ב- (1).  
נניח בה"כ כי כל האבות הן בנים ימניים (כלומר המסלול מתקדם ימינה בכל רמה) ולכן ניתן לייצג את הערך לעיל ע"י   
 נסמן ב – (2).  
ערכו בייצוג הפרשים של הוא תוצאתה של משוואה מס' (2), נשים לב כי משוואה מס' 1 היא הנגדית של משוואה מס' 2 ולכן משוואה מס' (1) בערכה המוחלט שווה לתוצאת משוואה מס' 2 הלוא היא הייצוג בהפרשים של

1. **While**

4. Return

**Heap-Extract-Max** -  
השגרה תאחזר את הערך האחרון בערימה אותו אנחנו נחליף בהמשך עם שורש העץ, לאחר מכן נאחזר את הבנים של שורש העץ, נבדוק מי מהם יותר גדול שהוא יהיה שורש העץ אחרי שנסיר את שורש העץ הנוכחי. נבצע 2 החלפות כך שבסוף ההחלפה המפתח שהיה בערך התחתון ביותר יהיה בן ימני או שמאלי של שורש העץ, ושורש העץ יהיה הגדול מבין 2 הבנים. נקרא לשגרה Extract-Asisstant שאחראית להחזיר את הערך למקומו ולמנוע הפרה בערימה.  
בסיום נחזור לייצוג הפרשים ונחזיר את המקסימלי.

זמן ריצה -

1. **If**
2. **Then** **error** “heap underflow”
3. **If**
4. **Then**
5. **Else**

8. **return** max

**EXTRACT ASSISTANT** –

שיטה זו מקבלת מ HEAP-EXTRACT-MAX אינדקס (2\3) בערימה שנמצא בו הערך שהיה ב heap-size[A] שעבר לראש הערימה. השיטה אחראית לתקן את ההפרה(אם יש) שנוצרה בעקבות הצבת האיבר שערכו לראש העץ. נשתמש בהפרד ומשול על-מנת לרדת מלמעלה למטה ולאחזר בחזרה מעלה את הערכים כנדרש.

זמן ריצה -

1. **If**





8. **Else if**

נכונות האלגוריתם -   
נוכיח באינדוקציה כי השגרה מקימת את הדרוש ובסופה הערמה היא ערימת הפרשים כך שאם נאחזר אותה תהיה ערמת מקסימום, בה"כ נוכיח על שורות 3-9 שכן שורות 10-16 זהות רק לבן שני.  
**בסיס –** בעבור (גובה ערמה) , אם קטן מהבן הימני תתבצע החלפה בשורה 6 וקריאה רקורסיבית ל- Extract-Max שלא תמשיך כי אנחנו ברמת העלים, בהתקפלות נבצע אחזור ערכים בשורות 8,9 ולבסוף העץ הנתון הוא ערמה מקסימלית שהפכה לערימת הפרשים.  
**הנחה –** נניח נכונות בעבור גובה ערימה קטן ממש מ- h ונוכיח עבור גובה h .  
אזי , מהנחת האינדוקציה בעבור h-1 רמות (כאשר h רמת עלים) הערכים משוחזרים, ואין הפרה.  
הבדיקה כעת תתבצע בין ראש העץ ברמה ה- h-1 לבין בניו ברמה ה-h, אם כך נבחן את היחסים ביניהם ע"פ שורות 3-9 ונבצע החלפה. נסמן ב- X את ראש העץ הנוכחי ברמה ה- h-1 כך ש וגם אז ניקח את המקסימלי מבין ונחליף אותו עם X.  
כך שלבסוף בראש העץ ברמה ה- h-1 נמצא שגדול בערכו מהאיברים ברמה התחתונה h.  
נקרא רקורסיבית לרמת העלים, ברמה זו נתקפל חזרה כי אף תנאי לא עובר, הרי לרמת העלים אין בנים כלל.  
בהתקפלות נהיה שוב פעם בעץ ששורשו הוא איבר ברמה h-1 ובניו ברמה h, ובשורות 8,9 נחזיר את העלים לייצוג של ערמת הפרשים.  
כך שלבסוף העץ ברמת h-1 שבניו בגובה h הוא עץ מקסימלי שהוסב לעץ הפרשים.  
בסדר

**סעיף ג' –**

נחשב את זמן הריצה באופן פרטני יותר לכל אחת מהשגרות –

**BUILD MAX HEAP (**תוך שימוש בנוסחא בעמ' 111) **-**

**MAX-HEAPIFY –** לא בוצע כלל שינוי, ולכן זמן הריצה זהה לזו של השגרה המקורית.

**HEAP SORT**   **-**  לא בוצע שינוי מעבר למס' פעולות קבוע לכן זמן הריצה זהה לשגרה המקורית.

**HEAP-EXCTRACT-MAX –**

**MAX HEAP INSERT–** לא ביצענו שינוי מעבר לפעולה קבועה ולכן זמן הריצה זהה לשגרה המקורית.

**HEAP INCREASE KEY –**

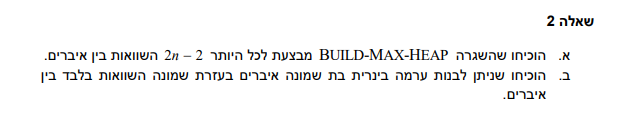
**RESTORE ROUTE –** שגרת תחזוקה נוספת שהוספנו –

**RESTORE VAL -** שגרת תחזוקה נוספת

**EXTRACT-ASSISTANT –** זהה ל- Max-Heapify

תשובה טובה מאוד, יש לציין,

סדר הגודל של הסיבוכיות, או הסיבוכיות האסימפטוטית אינה משתנה.



**פתרון שאלה 2 –**

1. בכל עץ קיימים צמתים שיש להם בנים, ולכן אך ורק בעבור הצמתים הללו מתקיימות השוואות, היות ולכל עלה לא מתקיימת אף לא השוואה אחת.  
   לכן נתמקד בצמתים בלבד.  
   במקרה הגרוע ביותר המפתח בכל צומת צריך לרדת לרמה התחתונה ביותר כלומר לרמת העלים ולכן מתבצעות 2 השוואות כפול ההפרש בין גובה העץ(הגבוהה ביותר) לבין גובה הצומת שהערך מוחזק בה בתחילת הקריאה. כלומר ;  
   אם גובה הצומת הוא 0 ( שורש העץ הגבוהה ביותר ) , וערכו של השורש צריך לרדת לרמה התחתונה ביותר, מתבצעות השוואות עד אשר מגיעים לרמה התחתונה(בתרגיל 6.1-2 הוכח כי גובה ערמה בת n איברים הוא ).   
   נשתמש בנוסחה שמוצגת בעמוד 111, כאשר הוא זמן הריצה של בעבור צומת בגובה h, נחליף את זה במספר ההשוואות בכל קריאה כפול הגובה ולכן מתקבל הסכום הבא -

בסדר

1. נוכיח את הטענה :  
   יהי S - מערך בעל 8 משתנים(עשויים להיות זהים) כך שכל אחד מהם מייצג מספר כלשהו, כך שאיננו יודעים מהו היחס בין המספרים.  
   נבצע **4** השוואות בין המשתנים כך ש מהשוואות אלו נוכל לבנות 2 עצים בינריים(בה"כ נניח כי אלו היחסית שהתקבלו בין כל השוואה, איננו יודעים עדיין מה היחס בין A ו- C וגם בין E ל- G).

A

C

H

G

F

E

D

B

כעת מבצעים זוג השוואות נוסף בין אם ו - העצים כמו באיור 1 לעיל, אחרת נשנה את העצים להצגה הבאה ,

H

F

E

G

D

B

A

C

נניח כי איור 1 הוא ההצגה המתקבלת הוספנו 2 ההשוואות לעיל, וסה"כ ביצענו 6 השוואות.

עם השוואה נוספת אנחנו בודקים האם כדי להחליט מי יהיה שורש העץ , בה"כ נניח .  
כעת נותרה בידינו בדיקה אחרונה, נשתמש בה כדי לבדוק האם B>C כדי לדעת מי נמצא מעל מי. בה"כ נניח כי ולכן העץ המתקבל לבסוף –

A

H

G

F

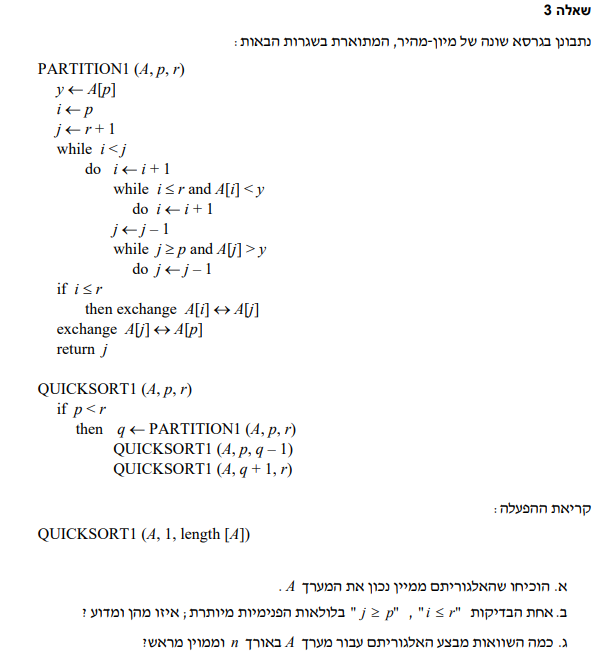
E

D

CC

B

לסיכום הוכחנו כי באמצעות 8 השוואות על קלט לא ידוע אפשר לסדר אותו כך שיהיה ערימה מקסימלית.בסדר





**פתרון שאלה 3–**

1. תיאור האלגוריתם -

מבחינתם נכונות האלגוריתם, השגרה QUICKSORT1 זהה לשגרה של QUICKSORT שמופיעה בספר בעמ' 122, ההבדל במיון הוא בשגרה PATRITION1 ולכן , די בכך שנוכיח שהאלגוריתם Partition1 מקיים את הנדרש בעבור קלט בגודל n על מנת להוכיח שהאלגוריתם quicksort1 כולו ממיין את המערך A. אם כך, נוכיח ע"י שמורת לולאה כי Partition1 מיין –

**אתחול** – לפני האיטרציה הראשונה מוגדרים המשתנים שהוא מכיל את המפתח של A[p] , יהווה האינדקס של p , ו- שמאותחל להיות אורך המערך הנתון +1.   
הגדרה זו מבטיחה לנו שהכניסה ללולאה הראשונה תמיד תתבצע, היות ואם התנאי הראשון של QuickSort1 לא יעבור ולכן Partition1 לא תופעל אם"ם .  
ולכן בכל קריאה לPartition1 נדע כי .  
כמו כן , לפני כל איטרציה נדע שכאשר אנחנו קוראים ל- Partition1 , כל מי שנמצא מימין ל- j גדול מ y) – ולכן עלינו לבדוק רק את הערכי מפתח שנמצאים באינדקסים . גם באיטרציה הראשונה כאשר r = A[length] וגם j = r+1 נגיד כי התנאי מתקיים במובן הריק.

**תחזוקה** – נחלק את התחזוקה לשלוש לולאות – מסומנות בהתאם על גבי הקוד בשאלה לעיל.  
ונדון מתקיים כי -   
1. כל מי שנמצא מימין ל-j מקיים כי לכל מתקיים .   
נדון בכל לולאה באופן פרטני -   
1. הלולאה הראשונה תופעל בכל קריאה ל- partition1 היות ומההשמה בשורות 1-3 נובע כי ולכן משורה 3 מתקיים כי ולכן התנאי של לולאה מס' 1 יעבור.  
בנוסף בכל פעם שנצא משתי הלולאות 2-3 ומתקיים נבצע החלפה בין האיברים, כלומר איבר נחליף בין איבר גדול לבין איבר קטן.

2. לולאה 2 – לולאה זו אחראית לקדם את i כל עוד ולכן בסוף הלולאה הזו אחד משני המקרים הבאים מתקיים – או שA[i] > y או ש (מחוץ לגבולות המערך). כלומר אחרי שלולאה 2 תסתיים נדע כי לכל מתקיים .

3. לולאה 3 – לולאה זו אחראית לשנמך את j כל עוד ולכן בסוף לולאה זו מתקיימים אחד משני המקרים – או ש j = i-1 או ש כלומר קיים A[j] < y. ולכן אחרי שלולאה מס' שלוש תסתיים נדע כי לכל מתקיים כי .  
  
לולאה 2-3 ושורה ראשונה של הלולאה הראשונה מבטיחות לנו כי המקרה בו יתקיים היות ובלולאה 2 ובלולאה 3 מובטח לנו כי ולכן אם עדיין , שורה מס' 5 – תבצע כלומר לכל אורך הקוד מתקיים כי i גדל ואילו j קטן וכאשר שורה 5 תגרור את ולכן התנאי של לולאה אחת לא יתקיים.  
  
**סיום** – אם כך בסיום הלולאות מתקיים אחד מהתנאים הבאים בוודאות אנו יודעים כי לכל מתקיים . וכי לכל מתקיים וגם כי ולכן -   
במקרה בו שורה 13 תתבצע ונדע כי המפתח במקום ה ונבצע חילוף ביניהם ע"פ שורה 13.  
אם נדע כי מה שגורר ולכן נבצע חילוף ביניהם ע"פ שורה 12 היות ובמערך ממוין לכל אינדקס מתקיים כי ולכן החלפה זו מקיימת את הרעיון של מערך ממוין.   
לבסוף כלומר המוחזר לשגרה quicksort מקיים כי כל מי שמימין לו גדול מ- A[p]. וכל מי שמשמאלו מקיים כי קטן שווה מ-A[p] אשר נמצא במקום של A[j].

בסדר

1. הבדיקה המיותרת היא הבדיקה של לולאה 3 , .  
   לפני הכניסה ללולאה מס' 1, p הוא האינדקס השמאלי ביותר של המערך הנתון ולכל אורך השגרה איננו מזיזים את p, כי הוא האינדקס הדינמי. ולכן כל עוד אנחנו באחת מהלולאה 1-3 מובטח לנו ש הוא הערך של האינדקס השמאלי ביותר במערך.  
   כלומר הבדיקה איננה רלוונטית כי אם אז A[p] = A[y] אז לא מתקיים כי ולכן בהכרח כל עוד הלולאה השלישית רצה. כלומר מכסה גם את התנאי .  
   כמו כן הבדיקה של הלולאה השנייה הכרחית היות ו- (שורה 5) לכל איטרציה של לולאות 1-3, ולכן אם לא קיים x במערך כך ש התנאי ימשיך לרוץ ונעוף בשגיאת ריצה כאשר שלא קיים במערך. בסדר

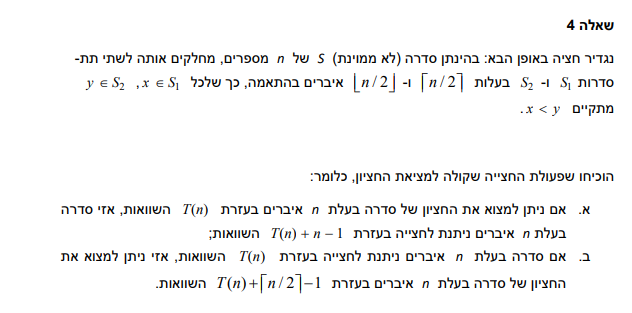
1. נחשב את מס' ההשוואות שמבצע האלגוריתם עבור מערך A באורך n וממוין מראש כלומר לכל A[i…..n] מתקיים כי .   
   שורות ההשוואה – .

3. שורה מס' 6 בשגרה Partition1 - השוואה אחת.

4. שורה מס' 9 בשגרה Partition1 – השוואה אחת.  
השגרה PARTITION1 - מההשמה בשורות 1-3 מתקיים כי תמיד ולכן הבדיקה תתקיים **ותעבור** ולכן בכל איטרציה מתקיים כי (שורה 5) והיות והמערך ממוין מתקיים כי ולכן לולאה מס' 2 תבצע אך ורק בדיקה אחת כל עוד ניכנס לשגרה Partition1. **לולאה 3** תבצעה כל פעם השוואה אחת כפול גודל המערך הנתון ששווה ל- היות ובכל כניסה מחודשת ל- Partition1 גודל המערך קטן ב-1, הסכום המתבקש הוא גודל המערך בכל כניסה ל- Partition1 כאשר הבדיקה המקסימלית היא מערך בגודל בעוד הבדיקה המינימלית היא 2. כלומר   
סה"כ יתקיימו (באיטרציה ה- n מתקיים כי q=r ולכן לא ניכנס ל- Partition1) איטרציות כאלו – בפרט אלו האיטרציות שהקריאה הרקורסיבית השנייה מפעילה, לכל שאר האיטרציות בהן הקריאה הרקורסיבית הראשונה מתקיימת, מתקיים כי היות ומהערך ממוין בכל פעם הערך שיוחזר מהשגרה PARTITION1 יהיה 1 (האינדקס הראשון במערך בקריאת רקורסיבית ספציפית) כלומר הקריאה הרקורסיבית הראשונה תקיים תמיד QuickSort1(A,q,q-1) מה שגורר שהתנאי הראשון לאחר הקריאה לא יעבור והיא תתקפל.  
  
לפיכך מס' ההשוואות של כל השגרות אם המערך המתקבל כקלט ממוין הוא -

בסדר, ופיתוח הביטוי?

+====

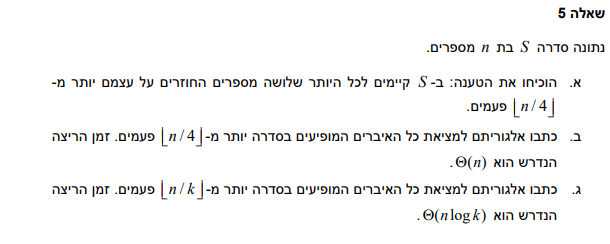


**שאלה 4 –**

1. לאחר מציאת החציון , כלומר האיבר "נקודת האמצע" של הקבוצה, בעלות של השוואות, נותר לנו להשוות את שאר איברי אל ערכי של החציון כדי לסווג אותם האם הם קטנים ממנו\גדולים ממנו.   
   אזי בהינתן מערך בגודל , נמצא את החציון שלו בעלות של לאחר מכן, נשווה את כל שאר האיברים (n-1) ביחס אליו, כל איבר שקטן מערכו של החציון יהיה בסדרה וכל איבר שגדול מהחציון יהיה בסדרה , החציון עצמו יוכנס לסדרה .  
   ולכן באמצעות השוואות נוכל לחצות את הסדרה הנתונה כמוגדר בשאלה.

בסדר

1. נסמן סדרה בעלת איברים ב- אז; עלות החצייה(השוואות) כפי שנתון הוא ולכן לאחר פעולת החצייה בידינו הסדרה כך שהתת סדרות ו- מקיימות את ההגדרה של השאלה. בלי הגבלת הכלליות נניח כי החציון נמצא בסדרה שיש בה מאיברי הסדרה ונגדיר אותה כ- בעוד הסדרה מכילה איברים. ידוע לנו שלאחר פעולת החצייה לכל מתקיים כי ולכן כל שנותר לנו הוא לחפש את האיבר הגדול ביותר בקבוצה המכילה, כפי שמצוין בספר בעמ' 152 עלות מציאת מקסימום או מינימום של סדרה בגודל היא השוואות ולכן, אם גודל הסדרה הוא אזי כדי למצוא את המקסימום שלה נדרש השוואות, ולסיכום הוכחנו כי ניתן למצוא את החציון שלה באמצעות השוואות. בסדר



**פתרון שאלה 5 –**

1. **נניח בשלילה כי קיימים 4 מספרים החוזרים על עצמם יותר מ- פעמים כלומר קיימים ארבעה מספרים כך שכל אחד מהם לפחות חוזר על עצמו פעמים. לכן מתכונות הערך השלם מתקיים(בין אם n זוגי או אי זוגי) -**

**כלומר אם קיימים 4 מספרים החוזרים על עצמם יותר מ- פעמים אזי מתקבלת סתירה כי סה"כ כמות האיברים היא . ולכן בהכרח מס' האיברים החוזרים על עצמם מס' פעמים קטן שווה ל- .**

**מאידך אם קיימים עד 3 איברים כאלו אז ;**

**ולכן לכל היותר קיימים 3 איברים המקיימים יותר מ .  
בפרט מקרה זה הוא מקרה פרטי לכך שלכל חלוקה קיימים לכל היותר איברים החוזרים על עצמם יותר מ- פעמים באופן דומה להוכחה הקודמת -  
יהי X כמות האיברים החוזרים על עצמם יותר מ פעמים בסדרה המכילה n איברים, נניח בשלילה כי אז מס' ההופעות המינימלי של כל איבר כזה הוא כלומר מס' ההופעות הכולל הוא –**

**וזו סתירה לכך שבסדרה המכילה n איברים יופיעו איברים ).** בסדר

1. **הרעיון – להשתמש באלגוריתם select למציאת ערכי המיקום אז לכל**

**מתקיים כי ולכל מתקיים . אם כך המערך מחולק סביב נק' הציר כך שמשמאל לה איברים קטנים שווים ומימין איברים גדולים.**

**לאחר מכן להשתמש בערכי המיקום של ולבדוק האם יש יותר מ הופעות לכל ערך כזה, נסביר מדוע בהכרח אלה ערכי המיקום שאנחנו מחפשים;**

**נוכיח את הטענה הבאה :   
כל איבר המופיע יותר מ- פעמים בסדרה, אחד ממופעיו יהיה בערך מיקום   
באינדוקציה -**בסיס **– אם גודל המערך 1 , הרי שאיבר נמצא בערך מיקום n, הוא הערך במיקום אחד ואכן מתקיים כי האיבר במיקום ה- 1 בסדרה בגודל 1 יופיע פעם אחת כך שהוא מקיים .**

הנחה – **נניח כי קיים ערך -נסמן ב- X - המופיע יותר מ מופעים בסדרה בגודל**

צעד – **בלי הגבלת הכלליות נתבונן ברבע הראשון של הסדרה בגודל n, ונניח כי ערך המיקום של מופעו הראשון של X נמצא במיקום Y. כלומר החל מ- Y קיימים לפחות עוד מופעים של X – ולכן היות והמופעים של X יהיו כולם משמאל לערך המיקום שלו(היות והגדרת ערך מיקום היא שכל איבר שמשמאל לו קטן שווה לו ולכן בהכרח קיים ערך של X המופיע במיקום .**

**זמן הריצה של האלגוריתם –**

1. **השגרה(דטרמיניסטית) Select**
2. **לולאת While**

**סה"כ -**

**האלגוריתם -** בסדר

**FIND-MOST-OCCURENCES)A)**

1. **IF**



6. 0
7. **While**
8. **IF**
10. **IF**
12. **IF**

15. **IF**  //if found one of them, print it.
17. **IF**
18. **IF**
20. **רעיון האלגוריתם -  
    כמו בסעיף ב' עלינו לחפש את ערכי המיקום של כך ש לפיכך נרצה להשתמש בשגרה Select ( דטרמיניסטית - כאשר בשונה מהשגרה הרגילה שמחזירה את הערך של המיקום שמחפשים, אנחנו לא נצטרך שתחזיר כלום ופשוט תמיין ע"פ ערך מיקום).כך שבסופה במיקומים יהיו האיברים ה"חשודים" שמספר הופעתם גדול מ , נבצע זאת בשיטת הפרד ומשול ונעצור כאשר נגיע למערך בגודל . האלגוריתם יפעל באופן דומה מאד לשגרה quicksort אך במקום תנאי עצירה שהוא מערך בגודל 1, התנאי עצירה שלנו הוא מערך בגודל ולכן סה"כ הפעמים שהשגרה תרוץ יהיה פעמים.  
    האלגוריתם Select בתנאי הראשון יפעל על מערך בגודל כדי למצוא את חציון החציון החציונים, ולאחר מכן כל קריאה רקורסיבית הוא ימצא חציון של התת מערכים הגדולים מ- .   
    מבחינת זמן ריצה – השגרה SORT-K פועלת באופן דומה לשגרה QuickSort אך השינוי המרכזי הוא שתנאי העצירה הוא מערך בגודל ולא מערך בגודל 1, לכן מס' הקריאות הרקורסיביות תלוי בגודל של (כי עץ הקריאות יהיה בגובה lgk) ולכן סה"כ מס' הקריאות הרקורסיביות יהיו בנוסף זמן הריצה של השגרה SELECT הוא . כלומר סה"כ זמן הריצה של הוא .**

**הלולאה בשורה 6 של השגרה רצה על המערך 2n פעמים לכל היותר היות ובכל פעם קיים שטח חפיפה בין 2 התת מערכים בגודל מכיוון שאנחנו מחפשים מימין**

**ומשמאל ע"פ ערך המיקום.  
נוכיח את האמור לעיל –**

**נציין כי הוא מס' ערכי המיקום ה"חשודים" ולכן בעבור כל k הריצה על המערך תהיה בעלות נמוכה מ- (לינארי).** בסדר

1. **If**
2. )