**תקציר**

בעיית הבחירה הינה בעיה אשר מעסיקה חוקרים בענף מדעי המחשב מזה זמן רב. בהינתן פריטים אשר שייכים לקבוצה בעלת יחס סדר מלא, נרצה למצוא את האיבר ה- הקטן ביותר מבין הפריטים. האלגוריתם הפשוט ביותר לפתרון הבעיה למעשה נעזר באלגוריתמים של מיון על ידי מציאת האיבר במקום ה- לאחר מיון הפריטים. אך, הוכח כי חסם תחתון לבעיית המיון הינו וכי אין צורך בהכרח למיין את הפריטים על מנת לפתור את הבעיה. כלומר, עלתה השערה כי ניתן לפתור את בעית הבחירה על ידי חסם תחתון נמוך יותר, ואכן נמצא אלגוריתם אשר פותר את הבעיה ב- . בעבודה זו, בחנו מספר אלגוריתמים שונים לפתרון בעיית הבחירה, ואת סיבוכיות אלגוריתמים אלו ביחס לחסם התחתון הנ"ל. נמצא כי אלגוריתמים המשתמשים בבעיית המיון לצורך פתרון בעיית הבחירה כדוגמת selectRandQuickSort ו selectInsertionSort, עושים זאת בסיבוכיות גבוהה מ וכי קיימים אלגוריתמים נוספים אשר מצליחים לשפר את הסיבוכיות ולהגיע לסיבוכיות המינימלית , כדוגמת medOfMedQuickSelect.בנוסף נמצא כי מבחינה מעשית האלגוריתם שהציג את הסיבוכיות הנמוכה ביותר לפתרון הבעיה הוא randQuickSelect, אשר זמן הריצה שלו הינו גם במקרה הגרוע.

**הפונקציה** - **selectRandQuickSort**

ניתוח תיאורטי🡨

* אלגוריתם זה מוצא את האיבר הk בגודלו על ידי מיון n הפריטים שנעשה בסיבוכיות ולאחר מכן מחזיר את האיבר שנמצא במקום הk בסדר הממוין ב-.
* מיון n הפריטים כולל כמה שלבים: בחירת pivot, חלוקת n הפריטים לפי הpivot (Lemuto’s או Hoare’s) לשני תתי מערכים וקריאה רקורסיבית לפונקציה בעבור שני תתי המערכים הללו.

לכן סיבוכיות ריצת הפונקציה תהיה תלוי בכל אחד מהגורמים האלה.

* במימוש זה הpivot נבחר באופן רנדומלי ולכן בחירת הpivot תהיה ב- .
* לאחר מכן , החלוקה לוקחת בסה"כ השוואות (שכן כל איבר מושווה פעם אחת לpivot שנבחר).
* כדי שהאלגוריתם ירוץ בBC נשאף בכל פעם שהpivot שיבחר יהיה ה**חציון** של הפריטים כך שהקריאה הרקורסיבית על תתי המערכים תוכל לקצץ את מספר ההשוואות פי 2 בכל פעם.

הWC של אלגוריתם זה יהיה אם בכל פעם יחתך המערך באיבר אחד בלבד(בחירת המקסימום/ המינימום בכל קריאה) והוא יהיה .

אין באפשרותנו לבחור כל פעם את החציון ולכן נפנה לפתרון של בחירת pivot רנדומלי.

* Pivot רנדומלי מאפשר מצב בו הסיכוי לWC הוא נמוך מאוד בהינתן **כל** קלט.
* נראה כיצד בחירת pivot רנדומלי משפיעה על ה-Average Case:

1. כל איבר מושווה לכל היותר פעם אחת עם איבר אחר (אם שני איברים מושווים אז אחד מהם הוא הpivot והם לא יושוו יותר)  
   לכן, אם נסמך את להיות מספר ההשוואות באלגוריתם וב את הפריטים בסדר הממוין שלהם - .
2. בחירת הpivot הרנדומלי מביאה למצב בו הסיכוי ששני איברים יושוו אחד לשני הוא פרופורציוני למרחק בינהם (בסדר הממוין) .

הוכחנו בכיתה בעזרת אינדוקציה כי -

כלומר קיבלנו שלא משנה איזה pivot יבחר, הסיכוי ששני מפתחות יושוו יהיה .

כפי שראינו בכיתה הצבת הערך במשוואה שקיבלנו בסעיף (1) ופתירת המשוואה בחישוב מדויק מראה כי (האלגוריתם הרנדומלי גרוע ב39% בלבד מהBC).

* סה"כ אלגוריתם זה מאפשר פתרון לבעית הבחירה ב 🡨

*תוצאות אמפיריות🡨*

קיבלנו כי – עבור מספר ההשוואות בעבור חיפוש איבר k הוא בממוצע (על עשרת איברי הk השונים שהיינו צריכים למצוא) הוא **154980**.1333 השוואות.  
עבור מספר ההשוואות הוא **337721**.1111. בעבור מספר ההשוואות הוא **533365**.9778 וכן הלאה...

ניתן לראות כי לא מדובר בגידול ליניארי בn אך כי קיימת תלות בn.

חישבנו מתאם פירסון בין מספר ההשוואות לכל n וk לבין הפונקציות .

קיבלנו כי המתאם עם הפונקציה הוא **0.999111789** והמתאם עם הוא **0.999496042**.

כלומר יש מתאם גבוה בין מספר ההשוואות לפונקציות הללו והמתאם חזק יותר בעבור nlogn.

(ביצענו גם מתאמים עם פונקציות אחרות התלויות בk ולא נמצא מתאם חזק).

ניתן אם כן לומר כי התוצאות האמפיריות מתיישבות יחסית עם הניתוח התיאורטי.

**הפונקציה** - **selectInsertionSort**

ניתוח תיאורטי🡨

* אלגוריתם זה מוצא את האיבר הk בגודלו על ידי מיון n הפריטים שנעשה בסיבוכיות בWC ולאחר מכן מחזיר את האיבר שנמצא במקום הk בסדר הממוין ב-.
* מיון n הפריטים נעשה בn איטרציות אשר בכל איטרציה מתבצעות לכל היותר n-1 השוואות ולכן סיבוכיות המיון הינה .
* נציין כי הWC של האלגוריתם יקרה כאשר המערך ממוין מהגדול לקטן וה-BC של האלגוריתם יקרה כאשר המערך כבר ממוין מהקטן לגדול ואז המיון יתרחש ב .
* למדנו בשיעור כי אלגוריתם זה יעיל מאוד למערכים קטנים בלבד או מערכים "כמעט" ממוינים.
* סה"כ אלגוריתם זה מאפשר פתרון לבעיית הבחירה ב 🡨 .

*תוצאות אמפיריות*🡨

קיבלנו כי – עבור מספר ההשוואות בעבור חיפוש איבר k הוא בממוצע (על עשרת איברי הk השונים שהיינו צריכים למצוא) הוא **25055690**.11 השוואות.  
עבור מספר ההשוואות הוא **100071851**.4. בעבור מספר ההשוואות הוא **224880074**.8 וכן הלאה...

ניתן לשים לב בבירור לגידול **ריבועי** במספר ההשוואות ואכן הצגה גרפית מראה זאת בבהירות, וודאי שלא גידול לינארי.

ניתן אם כן לומר כי התוצאות האמפיריות מתיישבות עם הניתוח התיאורטי.

**הפונקציה** - **selectHeap**

ניתוח תיאורטי🡨

* אלגוריתם זה מוצא את האיבר הk בגודלו על ידי בניית ערימה מ- n הפריטים ב ולאחר מכן k פעולות Delete-Min כשהאיבר האחרון שהוצא הינו האיבר הk המבוקש.
* בניית ערימת מינימום מn פריטים הינה תהליך שדורש מעבר על כל הצמתים ה"פנימיים"( בערך צמתים) ועבור כל אחד מתבצעת הפקודה Heapify-Down אשר בעלת סיבוכיות .

לכאורה נראה כי תהליך הבניה דורש O(nlogn) השוואות אך ניתוח עדין שעשינו בשיעור מראה כי ככל שצומת יותר רחוק מהשורש כך מספר הHeapify-Down האפשריים שנעשה לו ילך ויקטן. בסופו של דבר נתון זה עוזר למצוא חסם הדוק יותר לסיבוכיות הבניה והוא .

לאחר מכן, בכל מחיקת של האיבר המינימלי , נעביר את האיבר האחרון להיות השורש ונעשה עליו Heapify-Down כדי לשמור על תכונות הערימה. נעשה זאת כאמור k פעמים.

* סה"כ אלגוריתם זה מאפשר פתרון לבעית הבחירה ב 🡨 .

*תוצאות אמפיריות*🡨

עלה קושי בניתוח התוצאות האמפיריות ביחס לחסם התיאורטי שהתקבל שכן, יש תלות של חסם זה גם ב-n וגם ב-k והצגה גרפית דו ממדית פשוטה אינה מאפשרת להבין את דפוס הגידול במספר ההשוואות שהתקבל.

לכן פנינו לחישוב מתאם פירסון בין ערכי הביטוי כאשר מציבים בהם את ערכי ה-n וה-k הרלוונטיים לבין מספר ההשוואות שהתקבל עבור כל צירוף של n ו-k כזה.

התקבל מתאם דיי גבוה - **0.99957658** . נציין כי חישבנו מתאמים גם של מספר ההשוואות שיצאו לנו עם פונקציות שאינן תלויות בk והתקבלו מתאמים נמוכים.

(הסבר חישוב המתאם: כל זוג ערכים שמופיעים כנקודה בגרף הם בעלי k ו-n זהים , כאשר הערך של נקודה על תיר הy משקף את מספר ההשוואות וערכה על ציר הx מציין את הערך המתקבל עבור הk וה-n הרלוונטים בפונקציה n+klogn).

אם כן, מסקנה ברורה מניתוח זה היא כי התוצאות האמפיריות משקפות תלות של האלגוריתם בk וב-n וכי קיים מתאם יחסית גבוה בין הניתוח התיאורטי לתוצאות בפועל.

**הפונקציה** - **selectDoubleHeap**

ניתוח תיאורטי🡨

* אלגוריתם זה מוצא את האיבר הk בגודלו על ידי בניית ערימת מינימום מ- n הפריטים ב . לאחר מכן, מתבצעת הכנסה של השורש אל ערימה עזר חדשה שהיא בגודל לכל היותר k ואז באופן איטרטיבי מוחקים מערמת העזר את המינימום ומכניסים אליה את שני בניו של המינימום כפי שהם מופיעים בערמת המינימום המקורית.
* מחיקה והכנסה אל ערימת העזר תהיה בסיבוכיות שכן גודל ערימה זו לא עולה על k ומכאן מגיע השיפור בסיבוכיות של אלגוריתם זה על פני selectHeap.   
  אם כן, מתבצעות k איטרציות אשר בכל אחת מתבצעת Delete-Min מערימת העזר ב הודות לפעולת הHeapify-Down הכרוכה בכך ומתבצעות לכל היותר 2 פעולות Insert ב הודות לפעולת Heapify-Up לאחר הכנסה.
* סה"כ אלגוריתם זה מאפשר פתרון לבעית הבחירה ב 🡨 .

*תוצאות אמפיריות*🡨

גם כאן עלה קושי בניתוח התוצאות האמפיריות ביחס לחסם התיאורטי שהתקבל .

פנינו לחישוב מתאם פירסון בין ערכי הביטויים , בדומה לתהליך שעשינו בניתוח הקודם לבין מספר ההשוואות שקיבלנו.

התקבלו מתאם דיי גבוהים.

המתאם של מספר ההשוואות עם הוא - **0.999825668** .

המתאם של מספר ההשוואות עם הוא - **0.998859481** .

למעשה קיבלנו כי דווקא קיים מתאם חזק יותר בין החסם שניתן לselectHeap לבין אלגוריתם זה.

כאשר השוונו את הנתונים הגולמיים של מספרי ההשוואות בין שני האלגוריתמים (selectHeap וselectDoubleHeap) עדיין מצאנו כי עבור כל צירוף של k ושל n מספר ההשוואות באלגוריתם selectDoubleHeap נמוך יותר ממספר ההשוואות באלגוריתם השני.

כלומר, מסקנה מניתוח התוצאות האמפיריות היא כי אלגוריתם זה מהווה שיפור לאלגוריתם selectHeap, אך יתכן כי בפועל שיפור זה מזערי עד כדי כך שמספר ההשוואות בעבור כל צירוף של n ו-k יותר מתאים לקצב גידול של .

**הפונקציה** - **randQuickSelect**

ניתוח תיאורטי🡨

* אלגוריתם זה מוצא את האיבר הk בגודלו על ידי שימוש דומה בפונקציה quickSort שהזכרנו, אך, במקום להמשיך עם שני תתי מערכים, ממשיך בקריאה הבאה עם תת המערך הרלוונטי בו נמצא האיבר ה-k, ו"מתעלם" מתת המערך שאינו רלוונטי לפתרון הבעיה.
* מיון n הפריטים כולל כמה שלבים: בחירת pivot**רנדומלי**, חלוקת n הפריטים לפי הpivot (Lemuto’s או Hoare’s) לשני תתי מערכים, הצבת ה-pivot באינדקס המתאים ולאחר מכן בדיקה:
  + אם אינדקס ה-pivot קטן מ-k, קריאה רקורסיבית לתת המערך שמימין ל-pivot.
  + אם אינדקס ה-pivot גדול מ-k, קריאה רקורסיבית לתת המערך שמשמאל ל-pivot.
  + אם אינדקס ה-pivot לאחר החלוקה שווה בדיוק ל-k, מצאנו את האיבר ה-k בגודלו.

לכן סיבוכיות ריצת הפונקציה תהיה תלויה בכל אחד מהגורמים האלה, ובעיקר במספר האיברים ש"איבדנו" במהלך ריצת הפונקציה, ואותם אין צורך למיין.

* במימוש זה הpivot נבחר באופן רנדומלי ולכן בחירת הpivot תהיה ב- .
* החלוקה הראשונה לוקחת בסה"כ השוואות (כל איבר מושווה לpivot שנבחר). לא נוכל לקבוע כמה השוואות מתבצעות בשאר החלוקות, שכן יש תלות בכמות האיברים ש"איבדנו" בכל שלב.
* כדי שהאלגוריתם ירוץ בBC נשאף כי לאחר החלוקה הראשונה, וביצוע השוואות, אינדקס ה-pivot שווה בדיוק ל-k, כלומר ישנם בדיוק איברים הקטנים מה-pivot ולכן ה-pivot הוא האיבר ה-k בגודלו. מקרה BC אחר הוא כאשר ה-pivot שייבחר יהיה בדיוק ה**חציון** של הפריטים כך שהקריאה הרקורסיבית תמשיך בדיוק עם חצי מהמערך (ותזרוק את החצי השני), כך שנקבל מספר השוואות של: .
* הWC של אלגוריתם יתרחש כאשר בכל פעם יחתך המערך באיבר אחד בלבד (בחירת המקסימום/ המינימום בכל קריאה) ונקבל מספר השוואות של .
* אין באפשרותנו לבחור כל פעם את החציון ולכן נפנה לפתרון של בחירת pivot רנדומלי.
* Pivot רנדומלי מאפשר מצב בו הסיכוי לWC הוא נמוך מאוד בהינתן **כל** קלט אפשרי.
* בכיתה ניתחנו את ה-Average Case ואת תוחלת מספר ההשוואות עבור בחירת pivot רנדומלי:
  + ראינו כי כדי לחשב את תוחלת מספר ההשוואות, נחשב את סכום התוחלות עבור כל שלב.
  + ראינו כי לפי למה: Eni+1 Eni, כלומר תוחלת מספר ההשוואות בשלב מסוים חסומה על ידי ממספר ההשוואות בשלב הקודם.
  + על ידי הלמה, הוכחנו כי תוחלת מספר ההשוואות בשלב ה- חסומה על ידי , ועל ידי זה יכולנו להוכיח כי **תוחלת מספר ההשוואות של כל האלגוריתם חסומה על ידי , כלומר .**
  + כלומר, האלגוריתם מאפשר פתרון לבעיית הבחירה ב 🡨

*תוצאות אמפיריות🡨*

קיבלנו כי – עבור מספר ההשוואות בעבור חיפוש איבר k הוא בממוצע (על עשרת איברי הk השונים שהיינו צריכים למצוא) הוא השוואות.  
עבור מספר ההשוואות הוא  **.** בעבור מספר ההשוואות הוא וכן הלאה. ניתן לראות מדובר בגידול ליניארי ב-n.

חישבנו מתאם פירסון בין מספר ההשוואות שקיבלנו בעקבות הרצת הפונקציה לכל n וk לבין הפונקציות  
 .

קיבלנו כי המתאם עם הפונקציה הוא **0.949171564**, המתאם עם הוא **0.948288948** וכי המתאם עם הוא **0.918611528.**

כלומר יש מתאם גבוה בין מספר ההשוואות לפונקציות והמתאם חזק יותר בעבור n.

ניתן אם כן לומר כי התוצאות האמפיריות מתיישבות יחסית עם הניתוח התיאורטי.

**הפונקציה** - **medOfMedQuickSort**

ניתוח תיאורטי🡨

* אלגוריתם זה זהה לאלגוריתם שראינו, אך הפעם בחירת ה-pivot הינה **דטרמיניסטית**, כלומר, ה-pivot נבחר על ידי תהליך מחושב ומטרתו לבחור ב-pivot שיהיה קרוב בערכו לחציון המערך.
* במימוש זה, נחלק את הפריטים ל**חמישיות**, נמצא את החציון עבור כל חמישייה ונכניס למערך נפרד את כל החציונים. על מערך החציונים נקרא רקורסיבית לפונקציה medOfMedQuickSort על מנת למצוא את חציון המערך- **חציון החציונים.** ערך זה יהיה הפיבוט שלנו בכל שלב של חלוקה.
* כפי שראינו בפונקציה הקודמת, ה-BC הוא כאשר ה-pivot שייבחר יהיה בדיוק ה**חציון** של הפריטים כך שהקריאה הרקורסיבית תמשיך בדיוק עם חצי מהמערך (ותזרוק את החצי השני), כך שנקבל מספר השוואות של: .
* ה-WC לעומת הפונקציה הקודמת הוא גם כן , ונראה זאת על פי הניתוח הבא:
  + מציאת החציון של כל חמישייה לוקחת זמן ריצה של
  + לאחר מכן, נחפש את החציון עם קריאה רקורסיבית על בערך חמישית מאיברי המערך- .
  + בכיתה ראינו כי בכל שלב, בחלוקה על פי "חציון החציונים", אנחנו נפטרים **לפחות** מ מאיברי המערך, **ללא תלות בקלט**, לכן, אנחנו ממשיכים עם **לכל היותר** מאיברי המערך.
  + ביצוע החלוקה על פי הפיבוט הדטרמיניסטי לוקחת זמן ריצה של .

**בסה"כ זמן הריצה של האלגוריתם:**

כלומר, גם ה-WC בדומה ל-BC הוא , ולכן גם ה-Average Case הוא .

*תוצאות אמפיריות🡨*

קיבלנו כי – עבור מספר ההשוואות בעבור חיפוש איבר k הוא בממוצע (על עשרת איברי הk השונים שהיינו צריכים למצוא) הוא **79,527** השוואות.  
עבור מספר ההשוואות הוא **162,133.** בעבור מספר ההשוואות הוא **245,424** וכן הלאה. ניתן לראות מדובר בגידול ליניארי ב-n.

חישבנו מתאם פירסון בין מספר ההשוואות שקיבלנו בעקבות הרצת הפונקציה לכל n וk לבין הפונקציות  
 .

קיבלנו כי המתאם עם הפונקציה הוא **0.999805** וכי המתאם עם הוא **0.999736.**

כלומר יש מתאם גבוה בין מספר ההשוואות לפונקציות והמתאם חזק יותר בעבור .

ניתן אם כן לומר כי התוצאות האמפיריות מתיישבות עם הניתוח התיאורטי.