הרחבה והעשרה

❖ יעילות ורקורסיה:

- לעבור על פתרונות לשאלות רקורסיה ויעילות שעשיתי (~~בבלו ג'י/ שמרדף/~~ פתרונות של אחרים) שאלות ברקורסיה:
  - שאלת רקורסיה של דוד מ-2012א (להסתכל על פתרון לא שלי)
  - 2016א מועד א2 (פתרון בבלו ג'יי)
  - 2016א מועד א6
  - 2017ב מועד א3
  - 2017ב מועד ב1
  - 2018ב מועד א2
  - K המלכות
- ~~לפתור תרגילי מעקב שיבגני שלח~~ – אין צורך
- רקורסיה ברשימות (מצגת רשימות מקושרות, שקופית 34)
- לענות על השאלות מהמצגות מפגשים של יעילות, רקורסיה, ורשימות
- לצפות בהקלטות מפגשים 11-10 על יעילות ורקורסיה של ג'ודי/ שי תבור
- לעבור על מצגות ממפגשים 11 (דגשים - חיפוש בינארי, מאזניים, מיון מהיר), 12 ו-13

❖ נושאים נוספים:

- לעבור על ההודעות שמסומנות בכוכב בוואטסאפ (בשתי הקבוצות) ולראות אם יש טיפים חשובים/ שאלות קשות ששלחו ולא עשיתי (לפתור/ להסתכל על פתרון)
- לעבור על חומרי עזר מודפסים ומצגות
- לפתור את השאלות במצגת מפגש 15 (שלא היה) של יבגני
- רשימות - לתרגל הפיכת רשימה (מצגת מפגש 14, שקופית 35)

טיפים ודגשים

1. כללי

- לולאת do...while מתבצעת לפחות פעם אחת
- מבני נתונים - לשים לב אם מוסיפים לסוף/להתחלה של המבנה!!! (רשימה/מחסנית/תור) – מתבלבלת במיוחד בתור!!!

2. שאלות "מה הפעולה עושה?"

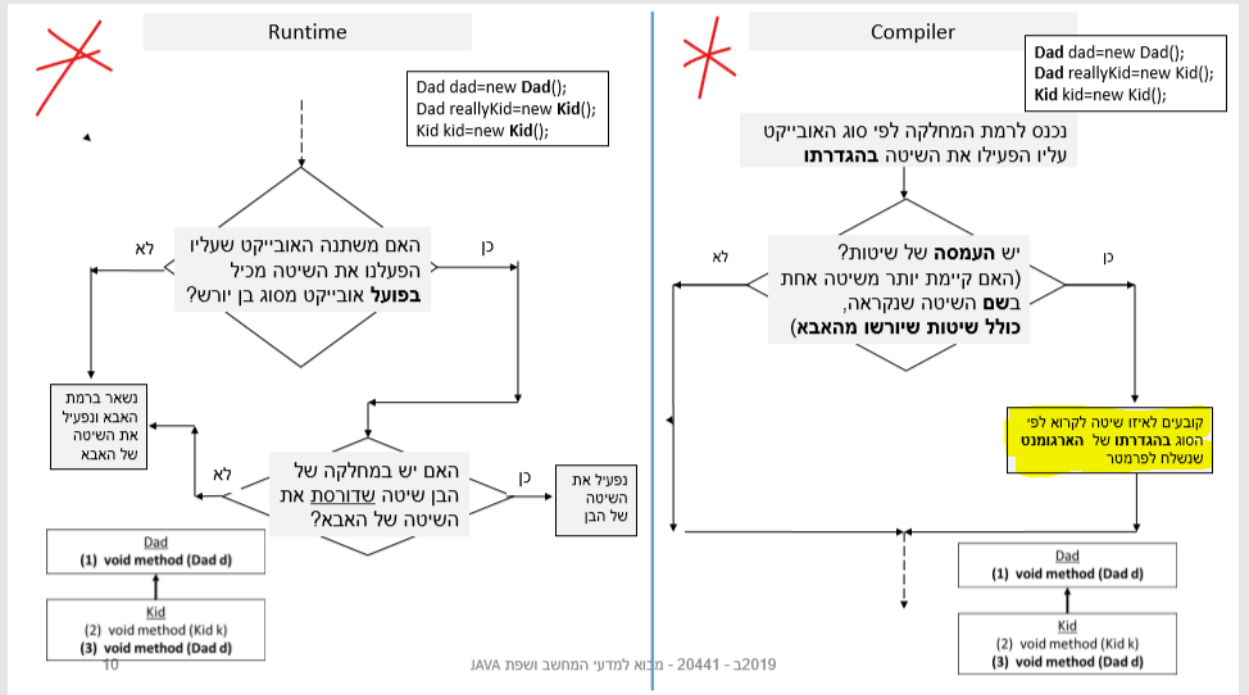
- טענת יציאה של שיטה = מה היא מבצעת בסופו של דבר.
- בניסוח התשובה צריך להתייחס גם למקרי קצה!
- תמר אוהבת שיטות שמחזירות את עומק העץ.
- לזהות דפוסים של שיטות:
  - return (קריאה לרקורסיה) + (קריאה לרקורסיה הבאה) = סכימה
  - הרבה תנאים ו-return אחד בסוף = בדיקה האם הכל?
  - הרבה תנאים ו-returnים בפנים = בדיקות של מקרה פרטני

3. מעקב על שיטה רקורסיבית

- **למספר את הפקודות של הקוד**
- **להקפיד על הזחה ומספור הפקודות במעקב**
- **לא לוותר או להניח שום דבר על חזרה של הרקורסיה הבאה, תמיד להכנס ולעקוב אחריה *עד הסוף!***

4. שאלות מעקב על מחלקות בירושה

- ישר לעשות מעקב עם מספרים וטבלאות משתנים!
- **לא לשכוח שלמחלקה יורשת יש גם את כל השיטות של האבא שלה!!!!**
- **לשים לב להרשאות גישה של השדות של האבא – אם פרייבט, הבן יכול לגשת לאבא רק דרך שיטה שיש לאבא.**
- אפשר לדרוס שיטה גם אם מחזירים ערך אחר שהוא תת טיפוס של הערך המקורי שאמור להיות מוחזר בשיטה של האב.
- Package private היא דיפולט
- **הקומפיילר תמיד יבחר את השיטה "הכי מתאימה"** – אם יש equals(Object obj) ו-equals(Shape s) והאובייקט שמתקבל כפרמטר הוא Shape, השיטה שתבחר היא זו שמקבלת shape ולא object.



- להזהר עם הבנאים, לשים לב אם הבנאי של האבא מדפיס משהו ולא להתעצל ללכת פיזית להסתכל! לא להסתמך על הזיכרון שלי.
- מחלקה שיש בה שיטה אחת אבסטרקטית חייבת להיות אבסטרקטית בעצמה (תופיעה המילה בחתימת המחלקה)
- לא לשכוח שמחלקה אבסטרקטית לא חייבת לממש את השיטות של האבא האבסטרקטי שלה!
- מחלקה שיורשת חייבת לממש את הבנאי של האבא (גם אם מדובר במחלקה אבסטרקטית).

5. פולימורפיזם

- לא לשכוח שקאסטינג נקבע בזמן ריצה, הוא עובר קומ' אם מדובר באותו עץ (לא משנה לאיזה כיוון – אם למעלה או למטה) אבל רק בזמן ריצה יתברר אם הקאסטינג עובר או לא (תלוי על מה המצביע מצביע באמת)
- בחתימה של מחלקה אבס' חובה שתופיע המילה אבסטרקט.
- למרקר את הרשאות הגישה של השדות – במיוחד לשים לב אם הן פרייבט!!!
- לא לשכוח לבדוק האם השיטה הולכת בזמן ריצה לשיטה של המחלקה של האובייקט עצמו ולא של המצביע.
- **לא להתבלבל בין דריסה להעמסה!!!!!!**

לחשוב טוב טוב האם זו דריסה של השיטה שהקומפיילר בחר!!!!

6. שאלות כתיבת קוד – כללי

- **לזכור שממוצע צריך להכנס ל-double!**
- **לשים לב אם תתכן אופציה לשגיאת חילוק ב-0!**
- **להקפיד מאוד על המרות של doubleים**
- **אורך של מחרוזת זה עם סוגריים - ()length.**
- **בשיטות בוליאניות אם נותנים תנאי להחזרת אמת צריך לכתוב שאחרת להחזיר שקר.**
- **לשים לב באיזה סדר אני כותבת את התנאים – באופן כזה שלא יגרום לשגיאה לוגית או שגיאת זמן ריצה.**
- להריץ דוגמא על הקוד שלי אחרי שאני מסיימת לכתוב אותו.
- לעשות בסוף בדיקה שסגרתי את כל הסוגריים המסולסלים ועשיתי להכל פסיק נקודה.
- כל הזמן לחשוב על אופציות לחריגה באינדקסים!!!!!
- כשאני כותבת קוד לא להתבלבל בין "גם" ל"או"!!! לחשוב טוב טוב מה זה צריך להיות

7. יעילות

- אם המערך ממוין – להשתמש בזה.
- פתרון לוגריתמים: אפשר תמיד למיין ב-nlogn
- מעבר על מערך דו מימדי ביעילות ליניארית – שיטת הנחש
- מערך ממוין = חיפוש בינארי!!!!
- בלולאה של מאזניים התנאי עצירה הוא שהאינדקסים לא שווים, כי מקדמים אותם אחד אחד. בלולאה שמקדמים את האינדקסים במקביל התנאי עצירה הוא קטן שווה.
- חיפוש בינארי – לחשוב על כל מיני וריאציות, כמו חיפוש העב"מ שאפשר לפצל לרבעים.
- מיון מערך – עדיף תמיד להחליף בין שני תאים מאשר להזיז את כולם אחד קדימה!
- בחיפוש שלשות במערך חד מימדי אי אפשר פחות מסיבוכיות ריבועית (תלוי במה שמבקשים כמובן!!! לא להסיק את זה אוטומטית), ולהלן הדרך:
  - לולאה באורך n שעוברת תא תא במערך
  - בתוכה לולאה באורך n שבתוכה יש שני מצביעים, אחד הנמוך ביותר ואחד הגדול ביותר, שמתקדמים בהתאם למה שמחפשים (שהוא בהתאם לערך של הלולאה החיצונית).
  - נקדם את הערך של הלולאה החיצונית.
- <u>דרכים להמנע מ-$O(n^2)$</u>
  - שיטת החציון (מיון בינארי), מסתמכת לרוב על מערכים ממוינים
  - שיטת האינדקס הכפול (מאזניים), בה עובדים עם אינדקס מההתחלה ואינדקס מהסוף

8. רקורסיה

```java
        /** Exam 2017a a5 86 question #1 **/
public static int edit (String str1, String str2) {
        return edit(str1, str2, 0, 0);
}

/** Exam 2017a a5 86 question #1 **/
private static int edit (String str1, String str2, int ist1, int ist2) {
    if (ist1 == str1.length() && ist2 == str2.length())
        return 0;
    if ( ist1 == str1.length() || ist2 == str2.length())
        return (str1.length() -ist1) + (str2.length() - ist2) ;

    if ( str1.charAt(ist1) == str2.charAt(ist2))
        return edit (str1, str2, ist1+1, ist2+1);

    int st1 = 1 + edit(str1,str2, ist1+1, ist2);
    int st2 = 1 + edit(str1, str2, ist1, ist2+1);


    return Math.min(st1,st2);

}
```

```java
public static int howManySorted(int n, int max){
    //if the recursive method finished placing values(no more cells) it should return a 1 success.
    if(n==0)
        return 1;
    //if the method ran out of values to place it should return a fail.
    if(max==0)
        return 0;

    //we are checking for each value(placed on the right) possible series until we ran out of values to place their,
    //and until we placed the minimum value for each cell in the series.
    return howManySorted(n-1,max) + howManySorted(n,max-1);
}
```

- לכתוב בלולאה ואז להחליף לרקורסיה
- לבדוק שאין קריאות מיותרות!
- לולאה רקורסבית – לקרוא להעמסה פלוס לרקורסיה ללולאה
- לכתוב את סדר הפרמטרים ככה שיהיה לי נוח.
- אם יש X מקסימלי לפעמים יותר נוח להוריד ממנו כדי לבדוק הישארות בתחום מאשר להוסיף לו (אפשר לוותר כך על פרמטר, לא צריך "לזכור" מה X – בתנאי עצירה במקום הגעה ל-X בודקים הגעה ל-0).
- לא "לחשוב הלאה" בשביל הרקורסיה! להתעסק בבעיה הנוכחית – או לקחת, או לא לקחת (בעיית המקל באורך k).
- לנסות לחפש חוקיות לפתרון (כמו השאלה עם number המינימלי)
- בשיטות רקורסביות עם String - תמיד משווים את התו במקום הראשון (באמצעות chatAt(0)),
  - **אסור להשתמש ב-substring!**
  - **בהשוואה בין מחרוזות התו האחרון לא יוצא דופן, לעשות פשוט תנאי לכל סיטואציה בה הגענו ל"null" באחת המחרוזות והרקורסיה הזאת כבר תחזיר 0 או מה שצריך.**

- סכימה של סך הכל האפשרויות:

```
//count the different options to add elements from weights starting at index i that
sums to sum.
public static int subsetSumCount(int[] weights, int sum, int i){
    int ans = 0;
    if(sum == 0)
        ans = 1;
    else if (sum < 0 | i >= weights.length)
        ans = 0;
    else
        ans = subsetSumCount(weights, sum - weights[i], i+1)
                + subsetSumCount(weights, sum, i+1);
    return ans;
}
```

תנאי עצירה

קריאות רקורסיביות

- בסכימה האם אם קיים איתי או בלעדיי אפשר להחזיר את הרקורסיה הבאה איתי *או* בלעדיי:

```
//check if the sum can be taken from weights, starting at index i.
public static boolean subsetSum(int[] weights, int sum, int i){
    boolean ans = false;
    if(sum == 0)
        ans = true;
    else if (sum < 0 | i >= weights.length)
        ans = false;
    else
        ans = subsetSum(weights, sum - weights[i], i+1) ||
              subsetSum(weights, sum, i+1);
    return ans;
}
```

תנאי עצירה

קריאות רקורסיביות

- לשמור שהסדר של בדיקת המצאות בגבולות המערך היא ראשונה כדי למנוע שגיאת זמן ריצה.
- מינימלי – צריך שמה שלא הגיוני יהיה ערך גדול שבלתי אפשרי לקבל,
  מקסימלי – מה שלא הגיוני יהיה 0 (או מינוס אחד).
- בשאלות מערכים דו-מימדיים לחשוב טוב אם צריך להחזיר 0 או 1 כשמוצאים את מה שחיפשנו!
  (תלוי אם סופרים צעדים/ אורך מסלול)
- **שלושת הכללים לבניית שיטה רקורסיבית:**

1. **תנאי בסיס – תנאי עצירה שניתן לענות עליו ללא קריאה רקורסיבית.**
2. **הקטנת הבעיה – קריאה רקורסיבית עם קלט הקרוב יותר לתנאי העצירה.**
3. **השלמת הפתרון – שימוש בתוצאת הקריאה הרקורסיבית לחישוב התוצאה המוחזרת.**

9. רשימות מקושרות

- לבדוק כל הזמן אם הרשימה ריקה. לתת טיפול מיוחד להכנס במקום הראשון
- לא להחריג באופן אוטומטי את ה-tail – כשעושים את אותה פעולה על כל האיברים לבדוק עד ש-node!=null. אם מכניסים/מוציאים איבר אז צריך טיפול מיוחד לקצוות.
- להשתמש ב-aliasing ולהשוות בין מצביעים!! (node != tail)

10. עצים

- לשים לב האם מדובר בעץ חיפוש בינארי או לא!
- לא להתבלבל בין rightSon ל-leftSon!
- אם אין גישה לערך המספרי בצמתים, מדובר בשאלה שקשורה למבנה העץ בלבד.
- || בין קריאות רקורסיביות = האם קיים מצב מסוים בעץ
- && בין קריאות רקורסיביות = האם מצב מסוים קיים לכל אורך העץ
- + צמוד לקריאות רקורסיביות = ספירה כמה פעמים קיים מצב מסוים/ סכימת איברים מסוימים בעץ