1. הטענה אינה נכונה. עבור המקרה בו x נמצאת ברשימה אבל רק ברמה אחת הפעולה delete תמחק את x מהרשימה ואז הפעולה insert עלולה (בהסתברות 0.5) להכניס את האיבר בשתי רמות או יותר מה שיגרום לכך שהרשימה תשתנה.
2. הטענה אינה נכונה. במקרה שx לא בS פעולת הinsert תכניס אותו למספר מסוים של רמות ואז פעולת ה delete תסיר אותו מכל אותם רמות. סך הכל שתי הפעולות יחדיו לא ישנו את הרשימה.
3. הטענה לא נכונה, נוכיח בעזרת עצי פיבונאצ'י
נניח כי הבן השמאלי הוא עץ פיבונאצ'י והבן הימני הוא עץ מלא. כאשר שניהם מגובה h. כפי שלמדנו מספר הצמתים בעץ מלא בגובה h הוא $2^{h+1}-1$
בזמן שמספר הצמתים בעץ פיבונאצ'י בגובה h הוא $\frac{\frac{1+\sqrt{5}}{2}^{h+3}-\frac{1-\sqrt{5}}{2}^{h+3}}{\sqrt{5}}$

$$Lim_{n\to\infty}\frac{|T_L|}{|T_R|}=Lim_{n\to\infty}\frac{\frac{\left(\frac{1+\sqrt{5}}{2}\right)^{h+3}-\left(\frac{1-\sqrt{5}}{2}\right)^{h+3}}{\sqrt{5}}}{2^{h+1}-1}<Lim_{n\to\infty}\frac{\phi^{h+3}}{2^{h+1}-1}\leq Lim_{n\to\infty}\frac{\phi^{h+3}}{2^{h}}$$
$$=Lim_{n\to\infty}\,\phi^3\left(\frac{\phi}{2}\right)^{h}=\phi^3\,Lim_{n\to\infty}\left(\frac{\phi}{2}\right)^{h}=0$$

לפי סנדוויץ $Lim_{n\to\infty}\frac{|T_L|}{|T_R|}=0$ ולכן $|T_L|=o(|T_R|)$ משמע לא יכול להתקיים $|T_L|=\Omega(|T_R|)$ במקרה הזה מה שמפריך את הטענה.

4. הטענה לא נכונה
נגדיר עץ חיפוש בינארי שבו לשורש יש רק בן ימני ואז הבן הימני שלו הוא שרוך שנמתח שמאלה. במצב כזה האיבר הראשון בסדר בעץ הוא השורש בגובה n-1 כאשר יש n איברים בעץ והאיבר השני נמצא בתחתית העץ. במצב כזה פעולת הsucc עבור האיבר הראשון דורשת מספר תזוזות שהוא $O(n)$ כדי להגיע לתחתית העץ אל האיבר השני.

5. נוכיח בעזרת אינדוקציה. עבור עץ עם צומת אחד e=i+2n מתקיים כי בעץ הזה e=i=n=0
יהי עץ מלא בגובה כלשהו h, כל עץ מלא שאינו רק צומת אחד מורכב משורש ושתי תתי עצים מלאים, אחד הבן הימני של השורש והשני הבן השמאלי של השורש. נניח שהמשוואה מתקיימת עבור שתי תתי העצים בגובה h-1. נקרא לn,I,e של התתי העצים e',i',n', $e'=i'+2n'$
בגלל שזה עץ מלא שני התתי עצים שווים אחד לשני בכל הפרמטרים.
$$n=2n'+1$$
בגלל שהעץ כולל את כל הצמתים הפנימיים של שתי התתי העצים וגם את השורש
$$i=2*(i'+n')=2i'+2n'$$
כי המסלול של כל צומת פנימי בכל אחד משתי תתי העצים מתארך ב1.
$$e=2(e'+n'+1)=2e'+2n'+2$$
יש n+1 עלים בעץ עם n צמתים בפנימיים והמסלול של כל עלה בכל אחד משתי תתי העצים מתארך ב1
נשתמש בהנחת האינדוקציה
$$e=2e'+2n'+2=2(i'+2n')+2n'+2=2i'+6n'+2=(2i'+2n')+2(2n'+1)$$
$$=i+2n$$
וכך בעצם הוכחנו באינדוקציה שלעץ מלא מכל גובה המשוואה מתקיימת.

א.

נממש את המבנה הנתונים בעזרת עץ AVL ששומר מפתח וערך בכל צומת. בעץ יהיה לכל צומת מצביע להורה שלו.
לכל צומת יהיה גם 4 שדות מספריים מיוחדים אשר יעזרו לנו בביצוע פעולת הquery. השדות הם:

- total - סכום כל האיברים בתת העץ שהצומת הזה הוא השורה שלהם
- **max_segment -** הסכום הכי גדול של הערכים של סדרה רציפה של צמתים (inorder) שמוכלת בתוך תת העץ של הצומת
- **max_left -** הסכום הכי גדול של הערכים של סדרה רציפה של צמתים שמוכלת בתוך תת העץ של הצומת וכוללת את האיבר הכי שמאלי בתת העץ
- **max_right -** הסכום הכי גדול של הערכים של סדרה רציפה של צמתים שמוכלת בתוך תת העץ של הצומת וכוללת את האיבר הכי ימני בתת העץ

להלן נתאר את התהליך שבו נעדכן את הערכים של ארבעת השדות המיוחדים עבור צומת מסויים בהנחה שהשדות המיוחדים של הבנים של הצומת כבר מעודכנים (אם אחד או שני הצמתים לא קיימים ניתן להחליף את הערך של הצומת ב0 עבור החישוב):

- **total -** נסכום את הסכומים של שני הבנים ונוסיף לסכום את הערך של הצומת הזה.

- **max_right -** קיימות שלוש אפשרויות למקסימום האפשרי.

  או שזה הmax_right של הבן הימני של הצומת
  או שזה סדרה שכוללת בתוכה גם איברים מהבן השמאלי של הצומת במקרה הזה המקסימום הזה הוא max_total של הבן הימני + max_right + הערך של הצומת הזה
  או שזה סדרה שכוללת בתוכה את הצומת הזה ולא כוללת צמתים של הבן הימני כלומר max_total של הבן הימני + הערך של הצומת הזה

  נשווה את שלושת המספרים האלה ונשמור את הגבוה מבין השלושה.

- **max_left -** באופן דומה נשווה בין

  הmax_left של הבן השמאלי
  לסכום שבין max_total של הבן השמאלי, הערך של הצומת הזה וmax_left של הבן הימני
  ולבין הסכום של max_total של הבן השמאלי והערך של הצומת הזה
  נבחר את הגדול מביניהם להיות הmax_left החדש.

- **max_segment -** הרצף ערכים עם הסכום הכי גדול בהכרח קיים או בבן הימני או בבן השמאלי או חוצה גבולות ביניהם. לכן הדרך שבה נבחר את max_segment זה על ידי המקסימום שבין חמשת האפשרויות:

  max_segment של בן ימני
  max_segmentשל בן שמאלי
  max_right של בן שמאלי +max_left של בן ימני+ ערך של הצומת הזה
  max_right של בן שמאלי + ערך של הצומת הזה
  max_left של הבן הימני + ערך של צומת זה.

סך הכל ניתן לבצא עדכון לכל השדות האלה בזמן קבוע כלומר ב $O(1)$

מימוש הפונקציות:

- **Init() -** נאתחל עץ ריק בזמן קבוע

- **Insert(x,v) -** נתחיל על ידי כך שנבצע פעולת הכנסת AVL סטנדרטית ב$O(\log n)$ כפי שנלמד. בהנחה

שמפתח לא היה קיים בעץ והכנסנו באמת צומת חדשה נאתחל את הצומת החדש עם 0 בארבעת השדות המיוחדים ואז "נטפס" את הדרך חזרה מהצומת החדש שנוצר ועד לשורש כאשר לאורך כל הדרך אנחנו מעדכנים את ארבעת השדות המיוחדים לכל צומת. הערכים של השדות המיוחדים של צומת שאינו נמצא במסלול הזה לא אמורים להשתנות לכן אין צורך לעדכן את הערכים עבור אף צומת אחר. כיוון שכל עדכון כזה הוא ב$O(1)$ והגובה בעץ AVL הוא $O(\log n)$ אז סך הכל תהליך העדכון הזה יהיה $O(\log n)$ והפעולה סך הכל תיקח $O(\log n)$ כנדרש.

- **delete(x) -** נתחיל על ידי כך שנבצע פעולת הוצאת AVL סטנדרטית ב $O(\log n)$ כפי שנלמד. בהנחה ההמפתח היה קיים בעץ והוצאנו באמת צומת נבצע עדכון של הערכים המיוחדים של הצמתים שנמצאים במסלול שבין הצומת לשורש באופן דומה לפעולת ההכנסה. גם במקרה הזה אין סיבה שהערכים המיוחדים של צמתים מחוץ למסלול ישתנו ולכן נוכל לבצע את כל העדכונים הנדרשים ב$O(\log n)$ ונקבל סיבוכיות כוללת של $O(\log n)$ כנדרש.

- **query(a,b) -** ניעזר בערכים המיוחדים שקבענו בכל צומת על מנת למצוא את הmax_segment בתחום שבין a לבין b. נרד לכיוון של a עד שנגיע לנקודת הפיצול של a וb. כלומר הנקודה הראשונה בעץ שa נמצא איפשהו מימין לה וb איפשהו משמאל לה. נקרא לנקודה זו c. כעת נחשב באופן נפרד משתנים זמניים שהם מקבילים של ארבעת הערכים המיוחדים אבל ספציפית עבור התחום שבין a לc והתחום שבין b לc. אחרי שנחשב את הערכים האלה נוכל למצוא את הmax_segment הכללי שבין a לc באופן דומה לאיך שעשינו כשעדכנו צומת.
הדרך שנמצא את ארבעת המשתנים הזמניים בין a לבין c היא שנעבור על המסלול שעולה מa ועד לc ולאורך העלייה נחשב את הערכים האלה בזמן שאנחנו "מתעלמים" מערכים שמושפעים על ידי צמתים שמשמאל לa.
קודם נאתחל את ארבעת הערכים להיות השדות המקבילים של הבן הימני של a אם הוא קיים ו0 אם הוא לא.
אז נעדכן את כל הערכים הללו כאילו הם עבור a אבל בהתעלמות מהבן השמאלי של a. מהנקודה הזאתי נתחיל לעלות למעלה. אם a הוא בן ימני של ההורה שלו אנחנו נדלג על כל הצמתים שבדרך עד שנגיע לצומת שהמפתח שלו הוא מימינה לזה של a. בנקודה הזאתי נעדכן את הערכים הזמניים שלנו כאילו אנחנו מעדכנים את הערכים של הצומת הנוכחי שאנחנו נמצאים בו אבל הבן השמאלי של הצומת זה הערכים הזמניים הקודמים ששמרנו. ככה בעדכון שלנו אנחנו נתייחס אך ורק לערכים שנמצאים מימינה לa ולא לאלו שנמצאים משמאל לa. נמשיך לטפס בעץ ולעדכן את הערכים ככה כשאנחנו בכל שלב מתייחסים בתהליך העדכון לבן שהגענו ממנו להיות עם הערכים הזמניים הקודמים. ככה כשנגיע עד לנקודת הפיצול c יהיה לנו את ארבעת הערכים הנכונים עבור התחום שבין a (כולל) לבין c (לא כולל).
עבור ארבעת המשתנים הזמניים בין b לבין c נמצאו אותם בדרך "סימטרית" לזו שעבור בין a לc.
לאחר שמצאנו את שתי הסטים של הערכים נמצא את הmax_segment הכולל כפי שתיארנו (נתייחס לשני הסטים כאל הבנים של c ובדרך חישוב של max_segment עבור c נמצא את ה max_segment עבור התחום שבין a לבין b).
כיוון שכל אחד מתהליכי ה"טיפוס" שתיארנו הוא למעשה שוב עדכון של ערכים ב $O(1)$ על כל צומת במסלול הטיפוס, כלומר ב $O(\log n)$ חזרות אז גם פה הסיבוכיות הכוללת היא $O(\log n)$ כנדרש.

ב.

מבנה הנתונים הנדרש במקרה זה דומה לעץ שהגדרנו בסעיף א כאשר ההבדל הגדול הוא שבמקום לשמור רק ערך אחד בשדות של max_left,max_right,max_segment במקום זה נשמור בשדה הזה מערך באורך k שכולל בתוכו את הסכומים הכי גבוהים בכל קטגוריה. עבור כל אחד מהשלושה נשמור גם מספר שהוא מספר המקומות הלא ריקים במערך. אין שינוי בtotal.

להלן נתאר את תהליך העדכון של השדות בצומת מסויים בהנחה שהשדות של הבן של הצומת כבר מעודכנים (אם אחד או שני הצמתים לא קיימים ניתן להחליף את הערך של הצומת ב0 עבור החישוב):

- **total -** זהה לסעיף א, $O(1)$

- **max_right** - נבצע תהליך מאוד דומה לתהליך שבמבנה הנתונים המקורי עם ההבדל שאנחנו עובדים עם רשימות באורך k במקום של המספרים.

ניקח את הלכל היותר k סכומים בmax_right של הבן הימני
ניקח את הלכל היותר k סכומים בmax_right של הבן השמאלי ונסכום אותם עם הערך של הצומת הנוכחי ועם הtotal של הבן הימני
ניקח את הסכום של max_total של הבן הימני והערך של הצומת הזה.

יש לנו כעת לכל היותר 2k+2 סכומים שונים, נמיין אותם לפי גודל ונשמור את הk הגדולים מביניהם במערך max_right.

- **max_left** - תהליך דומה לזה של max_right אבל לכיוון השני.

ניקח לכל היותר k מספרים מmax_left של בן שמאלי
ניקח את הלכל היותר k מספרים מmax_left של בן ימני נחבר אותם לtotal של בן שמאלי ולערך של הצומת הזה
ניקח את total של בן שמאלי ונחבר לערך של הצומת הזה

סך הכל יהיה לנו מערך של לכל היותר 2k+3 איברים, נמיין אותו ונשמור את הk איברים הכי גדולים במערך max_left

- **max_segment -**
ניקח את הלכל היותר k מספרים ב max_segment של הבן הימני
ניקח את הלכל היותר k מספרים בmax_segment של הבן השמאלי
ניקח את הלכל היותר k מספרים בmax_right של הבן השמאלי ונסכום אותם עם הערך של הצומת הזה
ניקח את הלכל היותר k מספרים בmax_left של הבן הימני ונסכום אותם עם הערך של הצומת הזה
וניקח את הk סכומים הכי גדולים של איברים מmax_left של הבן הימני ביחד עם איברים של max_right של הבן השמאלי וביחד עם הערך של הצומת הזה. (הדרך למצוא את הk סכומים הכי גדולים של איברים של 2 קבוצות באורך k ב$O(\text{k})$ קצת מסובבת. נבנה מערך באורך k כך, קודם נכניס את הסכום האיברים הראשונים של כל מערך. אחרי זה נבדוק האם לקדם את הindex של הקבוצה הראשונה או השנייה יניב לנו את הסכום הגדול יותר ואז נכניס אותו, נמשיך ככה עד שנקבל k סכומים הכי גדולים ב$O(k)$)

כעת נמיין את המערך באורך 5k שהתקבל ונשמור את הk הגדולים שביניהם בתורmax_segment

העדכונים כוללים סדרה סופית של פעולות $O(k)$ (העברת k איברים ממקום למקום) ופעולות של $O(klogk)$ (מיון של מספר קטן מ6k איברים). סך הכל כל הפעולות ביחד מסתכמות לסיבוכיות של O(klogk)

מימוש הפונקציות:

- **Init()** - אתחול של עץ ריק ב$O(1)$

- **Insert(x,v), delete(x)** - זהה לסעיף א עם ההבדל שאת עדכון הערכים נבצע באופן שמתואר לעיל. כיוון שכל עדכון כזה הוא
$O(k*\log k)$ ואנחנו מבצעים $\text{O}(\log n)$ שלהם בפונקציות הללו אז הסיבוכיות הכוללת של הפעולות insert וdelete יעלה ל
$O(k*\log k*\log n)$.

- **query(a,b)** - דומה לסעיף א כאשר המשתנים הזמניים גם פה יהיו מערכים באורך k והעדכון שלהם יעשה בשיטה המתוארת בסעיף הנוכחי. בסוף הערך שמוחזר יהיה הערך הk במערך של max_segments שנוצר בסוף עבור התחום של a עד b. ערך זה הוא הסכום הk הכי גדול של רצף ערכים כנדרש. גם כאן הסיבוכיות הכוללת היא $O(k*\log k*\log n)$ כיוון שעדכון המערכים של צומת זה $O(k*\log k)$ וגובה העץ זה $\text{O}(\log n)$.