חלק ג'- עבודה מספר 2 במבני נתונים:

ננתח את זמני הריצה של שלושת הפעולות הבאות:

1)lookup:

עד הנקודה הזאת O(1) פעולות. מכאן ואילך, מתחיל התיאור כבהמשך

```
public FloorsArrayLink lookup(double key) {
   int index=maxArrFilled;
   if(size==0){
      current= negativeInfinity;
      return null;
   }
   current=negativeInfinity;

   boolean stillNotInfinity=true;
   while (index>0)
   {
      Double nextKey=current.getNext(index).getKey();
      if(nextKey>key)
      {
         if(index-1>0)
         {
            //com.O++;
            index--;
         }
         else
         {
                  if(current.getNext(index).getKey()==positiveInfinity.getKey())
                  {
                     //com.O++;
                        return null;
                  }
                  else {
                        //com.O++;
                        current = current.getNext(index);
                        index = current.getArrSize();
                  }
            }

      }
      else if(nextKey==key)
      {
               //com.O++;
               return current.getNext(index);
      }
      else
      {
            //com.O++;
            current=current.getNext(index);
      }

   }
   return null;

}
```

A

B

C

במחלקת ה-list יש משתנה שתפקידו לשמור את מספר התא הגבוה ביותר במערך של החוליה *המינוס אינסוף*, שמצביע לחוליות ברשימה שאינה *פלוס אינסוף*.

לכן החיפוש מתחיל מהתא הזה.

מאחר ואנחנו מניחים שההכנסה בוצעה לפי המתואר בהמשך, המשתנה הנ"ל יהיה במקסימום $\log_2 n$.

בכל תא במערך של החוליה אנחנו בודקים, אם ערך החוליה אליו התא מקשר:

1. **גדול** מהערך שאנחנו רוצים לחפש(key)
   a. סימן שאנחנו מצביעים לתא רחוק מדי (היות והמערך מסודר לפי הגודל) ולכן נקטין את ה-index שמייצג את התא במערך החוליה אותו אנו בודקים
   b. אם אי אפשר להקטין יותר (הגענו לסוף המערך), נמשיך לתא הבא מיד בתור, ונמשיך באותה סריקה לפי גודל המערך שלו
   c. אם התא הבא הוא *פלוס אינסוף* הגענו לסוף הרשימה והחוליה המבוקשת לא קיימת
2. **שווה** לערך אותו אנחנו מחפשים
   a. מצאנו את הערך המבוקש, נחזיר אותו
3. **קטן** מהערך שאנו מחפשים
   a. אנחנו יכולים לעבור לתא אליו אנו מצביעים, היות ונישאר בטווח שקטן מהערך המבוקש.

בקוד מתחילים מהחוליה מינוס אינסוף ועוברים מהתא

*נגדיר: גובה= גודל המערך

<u>באופן כללי:</u>

- אנחנו עוברים מהרמה הגבוהה ביותר של המערכים עד לרמה התחתונה
- אם הערך המבוקש (key) גדול מהחוליה הבאה "באותו גובה", נעבור לחוליה הבאה באותו הגבוה
- עם הערך המבוקש קטן מהחוליה הבאה, נקטין את הגובה בו אנו מחפשים
- את המעבר הזה נבצע עד שנגיע לחוליה שאותה אנו מחפשים.

<u>גודל המערך של כל חוליה ברשימה מוגדר להיות:</u>

- גודל המערך של החוליה ה- i $(1 \le i \le n)$ הינו x+1, כאשר x הינו המספר הטבעי הגדול ביותר שמקיים $i \bmod 2^x = 0$ $(0 \le x \le \log n)$. n מספר המפתחות ברשימה (במערך הקומות).
- ניתן לראות שגודל המערך המקסימלי יכול להיות log n.
- גודל המערך של כל חוליה נקבע לפי המיקום של החוליה ברשימה.
- לחוליה עם מפתח k שגודל המערך שלה הוא arrSize, האיבר ה- i+1 במערך (אינדקס i, $0 \le i \le arrSize$)הוא מצביע לחולייה הקרובה ביותר משמאל/מימין שגודל המערך שלה גדול ממש מ-i, ובמילים אחרות מצביע לחוליה עם המפתח הגדול ביותר שקטן מ-k וגודל המערך של החוליה גדול מ-i.

- החלוקה הזאת של המערכים לפי מיקום החוליה ברשימה, "מחלק" מבחינה לוגית את איברי הרשימה כולה לחלקים לפי חזקות של 2.
- בתא הראשון, כל חוליה מצביע לחוליה הבאה בתור או לזאת שלפניה.
- בתא השני, חוליות מצביעות למערך שנמצא במרחק 2 חוליות ממנה.
- בתא השלישי, חוליות מצביעות למערך שנמצא במרחק 4 חוליות ממנה וכן הלאה.
- לכן כאשר אנחנו סורקים המערכים לפי התהליך שתואר לעיל, אנחנו עוברים בכל פעם מרחק של 2^{index} על אברי הרשימה כולה.
- בהתחלה עוברים על הרשימה בקפיצות מקסימליות, אחר כך בקפיצות שקטנות פי 2, עד שמגיעים לחוליה המבוקשת.

<u>ניתוח:</u>

- בכל רמה i ניתן לנוע ככמות הפעמים ש- 2^{i} נכנס בטווח שבו אנחנו מסתכלים.
- עבור הרמה העליונה, שהיא במקסימום $\log \mathrm{n}$, כמות הפעמים שניתן לנוע ימינה היא 1.
- לאחר מכן בתוך הטווח החדש הוא בגודל $\frac{n}{\log n}$
- עבור הרמה הבאה, ניתן לנוע פעמיים ימינה.
- עבור הרמה הבאה ניתן לנוע 3 פעמים ימינה.
- סה"כ, במקרה הגרוע ביותר, קיבלנו:

מסכום של סדרה חשבונית קיבלנו:

$$S_{\mathrm{n}} = \frac{\mathrm{n}(\mathrm{a}_1 + \mathrm{a}_{\mathrm{n}})}{2} = \frac{\log \mathrm{n}\,(1 + \log \mathrm{n})}{2} = O(\log n)$$

2)insert:

```
public void insert(double key, int arrSize) {
    FloorsArrayLink toInsert = new FloorsArrayLink(key, arrSize);

    int i=0;
    if(arrSize>maxArrFilled)
    {
        maxArrFilled=arrSize;
    }

    ///Fixing arrays from -Infinity foward
```

A

```
    int curr_index=maxArrFilled;
    FloorsArrayLink fixingNow= negativeInfinity;
    while (curr_index>0)
    {
        if(fixingNow.getNext(curr_index).getKey()<key)
        {
            //com.O++;
            fixingNow=fixingNow.getNext(curr_index);
        }
        else
        {
            if(curr_index>arrSize)
            {
                curr_index--;
            }
            else
            {
                //com.O++;
                fixingNow.setNext(curr_index,toInsert);
                toInsert.setPrev(curr_index,fixingNow);
                curr_index--;
            }

        }
    }
```

```
    ///Fixing arrays from +Infinity backward
    curr_index=arrSize;
    fixingNow= positiveInfinity;
    while (curr_index>0)
    {
        if(fixingNow.getPrev(curr_index).getKey()>key)
        {
            //com.O++;
            fixingNow=fixingNow.getPrev(curr_index);
        }
        else
        {
            //com.O++;
            fixingNow.setPrev(curr_index,toInsert);
            toInsert.setNext(curr_index,fixingNow);
            curr_index--;
        }
    }
    size++;
    first=negativeInfinity.getNext(1);
}
```

- נבדוק הכנסה של חוליה חדשה
- בהתאם לקביעה המופיעה לעיל, גודל המערך של החוליה ה- i ($1 \le i \le n$) הינו x+1, כאשר x הינו המספר הטבעי הגדול ביותר שמקיים $i \bmod 2^x = 0$ ($0 \le x \le \log n$). n מספר המפתחות ברשימה (במערך הקומות).
- אנחנו תחת ההנחה שההכנסה עד כה התבצעה לפי הסדר ובהתאם לקביעה הנ"ל.
- במערך אנחנו נעים באותה צורה כמו בחיפוש, לפי הרמות.
- בחלק שסומן כחלק A בקוד של insert, אנחנו מגיעים מהתא *מינוס אינסוף* ברשימה ועד התא החדש אותו אנחנו רוצים להכניס (forward).
- מתחילים ברמה הגבוה ביותר ומתקדמים בגובה הזה עד לחוליה שקרובה ביותר בערכה לחוליה החדשה שצריכים להכניס
- לאחר מכן יורדים בגובה הסריקה (curr_index) עד לגובה של המערך החדש שרוצים להכניס
- בכל פעם נעים שוב ימינה עד לחוליה הקרובה ביותר בערכה לחוליה החדשה, בגובה שבו אנחנו נמצאים
- מבצעים זאת עד שמגיעים לגובה המערך החדש.
- המצביעים שיש לעדכן כאשר מכניסים חוליה חדשה הם המצביעים מגובה החוליה החדשה ומטה.
- לכן נתחיל מהמצביע בגובה המערך החדש, אותו נשנה שיצביע על החוליה החדשה שהכנסנו.
- לאחר מכן נמשיך לרדת בגובה המערך החדש ונשנה בהתאם את המצביעים.
- בסיום כל התהליך הזה נבצע תהליך דומה מהצד השני, כלומר נתאים את כל המצביעים *ממינוס אינסוף* לחוליה החדשה

<u>ניתוח-</u>

- גובה המערך המקסימלי: log n, בהתאם לקביעה
- ראינו קודם (בניתוח זמן חיפוש) שכמות המעברים שניתן לבצע כדי להגיע לחוליה היא O(log n).
- כל שינוי של המצביעים הוא O(1) פעולות
- סה"כ קיבלנו O(log n) מעברים ימינה ולמטה, ובכל מעבר O(1) פעולות, סה"כ סיבוכיות זמן הריצה היא O(log n).

3)Remove:

```
public void remove(FloorsArrayLink toRemove) {
    int arrSize=toRemove.getArrSize();
    FloorsArrayLink prev = toRemove.getPrev(1);
    FloorsArrayLink next = toRemove.getNext(1);

    int i=0;


    for (i=0;i<arrSize;i++)
    {

        //com.O++;
        toRemove.getNext(i+1).setPrev(i+1,toRemove.getPrev(i+1));
        toRemove.getPrev(i+1).setNext(i+1,toRemove.getNext(i+1));
    }
    size--;
    first=negativeInfinity.getNext(1);


}
```

- בעת הסרת חוליה צריך בעצם להוריד את הטווח, ועבור תא i במערך, לקשר את החוליה שהמצביע אחורה במערך מצביע אליה במיקום i, לחוליה שהמצביע קדימה במערך מציע אליה במיקום i.
- סה"כ בהתאם לקביעות קודם, גודל המערך של חוליה הוא במקסימום $\log n$.
- לכן יש $\log n$ תאי מערך שעבורה $O(1)$ פעולות, לכן סה"כ סיבוכיות זמן הריצה היא $O(\log n)$

4) סיבוכיות מקום

נחשב את הסיבוכיות כך:

לכל חוליה במערך יש $O(1)$ שדות שקיימים לה מהגדרת החוליה.

לכן בתוך התחלה עבור n איברים יש $O(n)$ מקום בזיכרון.

לזה יש להוסיף לכל חוליה במקום זוגי עוד $O(1)$ תאים, למערך מצביעים קדימה ואחורה.

נוסיף לכל חוליה במקום שמתחלק ב-4 עוד $O(1)$ תאים, למצביע קדימה ואחורה, וכן הלאה.

קיבלנו טור כזה:

$$n + \frac{1}{2}n + \frac{1}{4}n + \cdots + \frac{1}{2^{\log n}}n = n\sum_{i=0}^{\log n}\frac{1}{2^i}$$

עבור $n=2$ נקבל שסך המקום הוא $1.5n$, עבור $n=2$, נקבל $1.75n$ וכן הלאה.

ובסך הכל קיבלנו סכום של סדרה הנדסית:

$$n \cdot S_n = n \cdot \frac{a_1 \cdot (q^n - 1)}{q - 1} = n \cdot \frac{1 \cdot \left(\left(\frac{1}{2}\right)^{\log n+1} - 1\right)}{\frac{1}{2} - 1} =$$

$$\frac{n((\frac{1}{2})^{\log n} \cdot \frac{1}{2}) - 1)}{-\frac{1}{2}} = \frac{n((\frac{1}{2})^{\log n} - 2)}{-1} \le \frac{n\frac{1}{2} - 2n}{-1} = 1.5n = O(n)$$

לכן מצאנו שהמקום שמבנה הנתונים הנ"ל תופס הוא $O(n)$, כאשר n- מספר האיברים ברשימה.