מבני נתונים – תרגיל 2

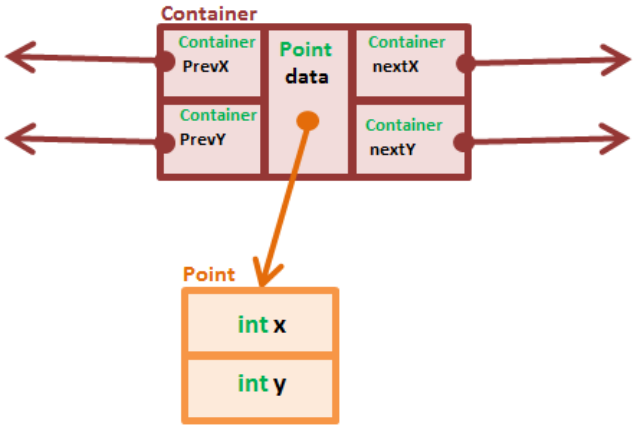
מגישים:
אסלן אסלן 302962493
עידן קלבו 201632163

תיאור מבנה הנתונים שבחרנו:

מבנה הנתונים שבחרנו היינו **רשימה מקושרת** עם תוספות אשר עזרו לנו לממש את השיטות הנדרשות עם אילוצי הזמנים.

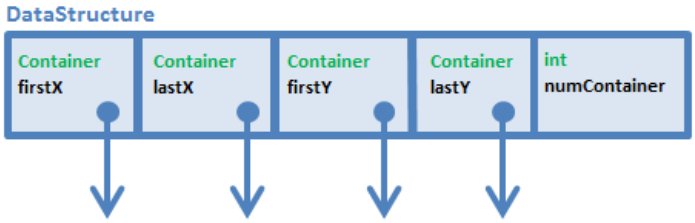
מחלקת Container מתנהגת כמו איבר ב**רשימה מקושרת** (link)

מחלקת DataStructure מתנהגת כמו הרשימה עצמה (linkList)



כל איבר (Container) ברשימה מכיל 5 שדות:

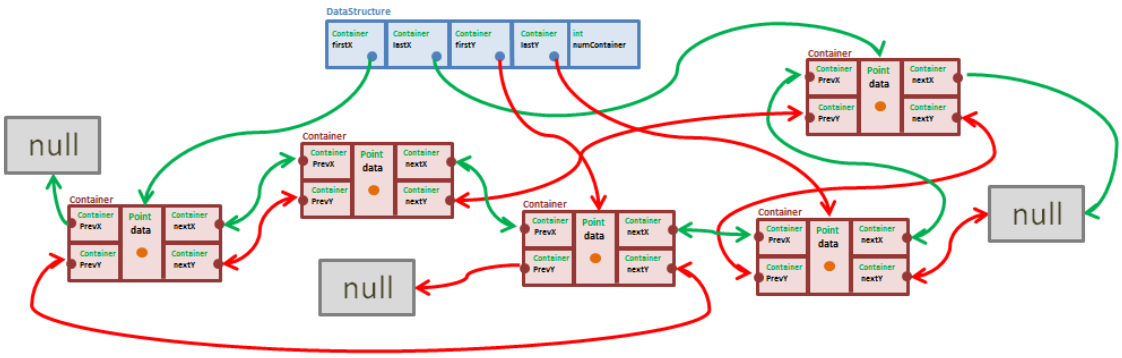
- שדה ה data בו שמורה הנקודה.
- 4 שדות של Container אשר מקשרים אותנו לאיברים הסמוכים אליו, לפי מיון של X ושל Y.



האובייקט (DataStructure) מכיל 5 שדות:

- שדה של numContainer אשר מונה את מספר האיברים ברשימה.
- 4 שדות Container אשר מקשרים אותנו לאיברים הראשונים והאחרונים, לפי מיון של X ושל Y.

רשימה לדוגמא : (בירוק מיון לפי x ובאדום מיון לפי y)



מימוש השיטות של הממשק DT:

void addPoint (Point point);

תיאור:
השיטה יוצרת איבר חדש ברשימה עם - point ב- data, ומעדכנת את השדות של המצביעים 2 לפי מיון של x
ו 2 לפי מיון של y. ואת השדות המצביעים של השכנים של הנקודה.
מציאת המיקום בו נכנס האיבר החדש נעשת על ידי פונקציית עזר אשר מחזירה קישור לאיבר שלפני האיבר שנרצה להכנים.
private Container LocationFront (Container newCont, **int** m ,Boolean axis)
כאשר אם הנקודה נכנסת לתחילת הרשימה (או לרשימה ריקה) ההכנסה נעשת על ידי פונקציית עזר:
void addPointfirst (Container newCont , Boolean axis)

זמן הריצה:
במקרה הגרוע נצטרך להוסיף את הנקודה בסוף הרשימה ולכן בכדי למצוא את המיקום נרוץ על כל איברי הרשימה ולכן זמן הריצה הוא **O(n)**

Point[] getPointsInRangeRegAxis (**int** min, **int** max, Boolean axis);

תיאור:
בעזרת פונקיות עזר אשר רצות על הרשימה מההתחלה לסוף ומהסוף להתחלה
private Container LocationFront (Container newCont, **int** m ,Boolean axis)
private Container LocationBackwards (Container newCont, **int** m ,Boolean axis)
ומחזירות מצביעים לאיברים שהם הקצוות ל min ו max.
לאחר מכן רצים בלולאה על האיברים שבין min ל max בכדי לדעת מהו גודל המערך
יצירת מערך בגודל מתאים, וריצה בלולאה להכנסת האיברים.

זמן הריצה:
במקרה הגרוע min = max לכן נעבור על כל איברי הרשימה.
ולכן זמן הריצה הוא **O(n)**

Point[] getPointsInRangeOppAxis (**int** min, **int** max, Boolean axis);

תיאור:
רצים בעזרת לולאה על כלל הרשימה (לפי מיון של axis!) ומונים את מספר האיברים שנמצאים בין min ל max.
יוצרים מערך בגודל המתאים. ועוברים שנית עם לולאה על כלל האיברים (לפי מיון של axis!) ומכניסים אותם למערך.

זמן הריצה:
שיטה זו תמיד רצה על כל איברי הרשימה פעמיים ולכן **O(2n)** אך ב **n** מספיק גדול אנו מתעלמים מקבועים
ולכן זמן הריצה הוא **O(n)**

double getDensity();

תיאור:
בעזרת שיטת עזר של Container
public int delta (Container other, Boolean axis)
אשר מחזירה את המרחק בין שני קורדינטות לפי axis
ובעזרת השדה numContainer מחשבים את הצפיפות.

זמן הריצה:
שיטה זו דורשת 5 פרמטרים בגישה מהירה ומספר חישובים פשוטים, ולכן תמיד היא רצה ב**O** של קבוע.
ולכן זמן הריצה הוא **O(1)**

void narrowRange (**int** min, **int** max, Boolean axis) ;

תיאור:
רצים על הרשימה משתי הקצוות לפי axis וכל איבר שמוחקים מעדכנים את המצביעים שלו כדי להשאיר את הרשימה ממוינת כמו צריך לפי הציר הנגדי.

זמן הריצה:
שיטה זו רצה בשתי לולאות שונות ונפרדות ששתיהן ביחד הוא מספר האיברים שנמחקים, ומעדכנת כתובות. ולכן זמן הריצה הוא מותאם לקבוע כפול מספר האיברים שנמחקו.
ולכן זמן הריצה הוא **O(|A|)**

Boolean getLargestAxis () ;

תיאור:
השיטה בודקת את המרחק בין שני האיברים הקיצוניים לפי כל שיטת מיון. מחזירה לפי התוצאה מי יותר גדול.

זמן הריצה:
שיטה זו דורשת 4 פרמטרים בגישה מהירה ומספר חישובים פשוטים, ולכן תמיד היא רצה בO של קבוע.
ולכן זמן הריצה הוא **O(1)**

Container getMedian (Boolean axis);

תיאור:
השיטה מקדמת מצביעים משני הקצוות עד שהם נפגשים באותה הנקודה ומחזירה את הנקודה הזאת.

זמן הריצה:
שיטה זו רצה על כלל הרשימה מני הקצוות עד מפגש המצביעים ולכן בסך הכל היא רצה פעם אחת על כל איבר.
ולכן זמן הריצה הוא **O(n)**

Point[] nearestPairInStrip (Container container, **int** width, Boolean axis);

תיאור:
השיטה מחשבת את הערך הקורדינטות של min ושל max ובעזרת הפונקציות:
private Container LocationFront (Container newCont, **int** m ,Boolean axis)
private Container LocationBackwards (Container newCont, **int** m ,Boolean axis)
היא מוצאת את האיברים הקיצוניים בתחום של min ושל max. החיפוש מתחיל מהנקודה במרכז כלפי חוץ.
בעזרת לולאה רצים על כל האיברים בתחום כדי לקבוע את גודל המערך הדרוש.
חוזרים שנית על הלולאה כדי להעתיק את הנקודות למערך.
ממינים את המערך לפי הציר הנגדי. ומחפשים את הנקודות הכי קרובות תוך השוואה של כל נקודה ל7 נקודות שאחריה.

זמן הריצה:
שיטה זו מקדמת שני מצביעים מהמרכז עד לגבולות של min ושל max ורצה פעם נוספת כדי לקבוע את גודל המערך ופעם שלישית כד להעתיק את הנקודות **O(3|B|)** , לאחר מכן מיון המערך לפי ציר נגדי לוקח **O(nlog(n))** שבמקרה שלנו שווה ל **O(|B|log(B))**, לבסוף מבצעים לכל נקודה 7 השוואות בזמן ריצה של **O(7|B|)** .
ולכן זמן הריצה הוא **O(|B|log(B))**

Point[] nearestPair();

תיאור:
*כמפורט בעבודה.

זמן הריצה:
נחשב את זמן הריצה לפי סעיפים:

1. אם יש רק זוג נקודות, החזר אותם. אם יש פחות, החזר זוג ריק (או null).
מספר קבוע של בדיקות O(1)

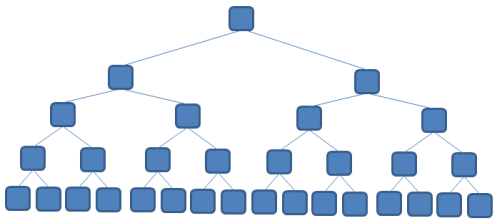
2. מוצאת את הציר הגדול ביותר axis נניח בלי הגבלת הכלליות ציר X
משתמש בפונקציה getLargestAxis () אשר רצה בזמן של O(1)

3. מוצאת את החציון median בציר X
משתמש בפונקציה getMedian (axis) אשר רצה בזמן של O(n)

4. מחשבת רקורסיבית את הזוג הקרוב ביותר עבור כל הנקודות הגדולות מ median (כולל (נקודות שקורדינטת ה-X שלהם גדולה מקורדינטת ה-X של ה- median) ורקורסיבית את הזוג הקרוב ביותר עבור כל הנקודות הקטנות מ median לפי ציר X (נקודות שקורדינטת ה-X שלהם קטנה מקורדינטת ה-X של ה- median)
זמן הריצה של שלב זה מחושב רקורסיבית, קל לחשוב על זמן הריצה כ"ש שלב הנוקאאוט" בטורנירים. כך שבכל שלב מספר ההשוואות קטן בחצי.

$$\left(\frac{n}{2}\right)+\left(\frac{n}{4}\right)+\left(\frac{n}{8}\right)+\left(\frac{n}{16}\right)+\cdots+\left(\frac{n}{n}\right)$$

$$= n\cdot\sum_{1}^{n}\left(\frac{1}{2}\right)^{n} = n => O(n)$$



5. בוחרת את הזוג הקרוב יותר מבין שתי הזוגות בצעד 4 ומחשבת את המרחק ביניהן minDist
מספר קבוע של בדיקות O(1)

6. בודקת האם ברצועה (בציר X) ברוחב 2*minDist אשר האמצע שלה זה ערך ה X של הנקודה median, יש זוג נקודות שמרחקן קטן מ minDist
משתמש בפונקציה nearestPairInStrip(Container container, **double** width, Boolean axis)
אשר רצה בזמן של O (| B | log (B)).
ועוד מספר קבוע של בדיקות O(1).
   a. אם קיימות זוג נקודות כאלו אזי הפונקציה מחזירה את הזוג הזה
   b. אחרת, הפונקציה מחזירה את הזוג מצעד 5

לסיכום:

O(1) + O (| B | log (B)) + O(1)+O(n) + O(n)+ O(1)+O(1).

O(| B | log (B)+n) אם B בסדר גודל של n <= O (n log (n) אחרת O(n)

הפונקציה: split (int value , Boolean axis)

הפונקציה מחזירה מערך של Container בגודל 4
כך שבכל תא מצוי מצביע על איבר ברשימה.
arryContainer [0] - מצביע על האיבר הראשון באוסף הקטן
arryContainer [1] - מצביע על האיבר האחרון באוסף הקטן
arryContainer [2] - מצביע על האיבר הראשון באוסף הגדול
arryContainer [3] - מצביע על האיבר הראשון באוסף הגדול

```
arryContainer [ 0 ] ← getFirst (axis);
arryContainer [ 3 ] ← getLast(axis);
Boolean  Continued ← true;
HelpCont1  ←  getFirst (axis);
HelpCont2  ←  getLast(axis);
while (Continued){
      if (axis){
            kay1 ← HelpCont1  . getData().getX()
            kay2  ← HelpCont2  . getData().getX()
      }else{
            kay1 ← HelpCont1  . getData().getY()
            kay2  ← HelpCont2  . getData().getY()
      }
      If  (value <= kay1){
            Continued =  !Continued
            arryContainer [1 ] ← HelpCont1
            arryContainer [2 ] ←HelpCont1.getPrev(axis);
            brack;
      }
      If   (value >= kay2){
            Continued =  !Continued
            If (value > kay2)
                arryContainer [1 ] ← HelpCont2;
                arryContainer [2 ] ← HelpCont2. getnext (axis);
            else{
                arryContainer [1 ] ← HelpCont2 .getPrev (axis);
                arryContainer [2 ] ← HelpCont2;

            }
            brack;
      }
      HelpCont1 ← HelpCont1.getnext (axis);
      HelpCont2 ← HelpCont2.getPrev (axis);
}
```

זמן הריצה:

הפונקציה רצה בלולאה משני צידי הרשימה ככה שהיא תמצא את המקום של value לאחר שהיא תרוץ על 2*|c| איברים מהרשימה. כאשר |c| הוא מספר האיברים ברשימה הקטנה מבין השניים.
ולכן זמן הריצה של הפונקציה היינו O (| c |) .