מבני נתונים -משימה2

הקובץ כולל בתוכו את הבונוס

שמות המגישים: בנימין פרסצקי 310727771

נוי משט 205835275

חלק ב'

תיאור מבנה הנתונים

מבנה הנתונים בו השתמשנו הוא רשימה דו מקושרת ממוינת.
לרשימה 5 שדות- שדה size מייצג את גודל הרשימה, כלומר מספר הcontainer-ים ברשימה.
בנוסף, ישנם 4 שדות הנותנים גישה ל4 מצביעים, שניים לכל ציר: מצביע למינימום ולמקסימום בציר הX ומצביע למינימום ולמקסימום בציר הY.

כל "חוליה" ברשימה היא container השומר בתוכו מידע, ולו 4 מצביעים- 2 מצביעים לcontainer הקודם והבא לפי ציר הX, ו2 מצביעים לcontainer הקודם והבא לפי ציר הY.

המצביעים מאפשרים שהרשימה תהיה ממוינת גם לפי ערכי הX וגם לפי ערכי הY בסדר עולה.

תיאור השיטות הממשות את הממשק DT וזמן הריצה שלהן

add point- השיטה מוסיפה נקודה חדשה למבנה הנתונים, בתור מידע השמור בתוך container.
אם מבנה הנתונים ריק) size=0(, ניצור container חדש ונבצע השמות של המצביעים.
אם size=1, נקרא לשיטת העזר insert2ndByAxis הבודקת האם הנקודה החדשה צריכה להופיע לפני או אחרי הנקודה הקיימת)בכל ציר(, ומוסיפה אותה ע"י יצירת container חדש ושינוי ההשמות של המצביעים.
אם גודל מבנה הנתונים גדול או שווה ל2, נקרא לשיטת העזר insertToList המשווה בין ערכי הנקודה הנתונה והנקודות הקיימות במבנה הנתונים, ומכניסה אותה תוך שמירה על הסדר הממוין של הרשימה.

insert2ndByAxis- זמן הריצה של שיטת עזר זאת הוא O)1(מכיוון שמתבצעות בה פעולות יסודיות בלבד)השוואות, קריאה לבנאי וקריאה לשיטות המבצעות השמות.(

insertToList- זמן הריצה של שיטת עזר זאת הוא O)n(כאשר המקרה הגרוע ביותר הוא שערך הנקודה החדשה הוא מקסימלי וההוספה צריכה להתבצע בסוף הרשימה.

לכן, זמן הריצה של השיטה הוא O)n(.

getPointsInRangeRegAxis- השיטה עוברת על מבנה הנתונים ובודקת בכל נקודה האם הערך של נמצא בטווח המינימום והמקסימום הנתונים לפי הציר הנתון. אם כן, הנקודה מועתקת לתוך מערך חדש.

במקרה הגרוע אין נקודות הנמצאות בטווח זה ולכן נעבור על כל הרשימה בזמן ריצה של O)n(.

getPointsInRangeOppAxis- השיטה עוברת על מבנה הנתונים ובודקת בכל נקודה האם הערך של נמצא בטווח המינימום והמקסימום הנתונים לפי הציר הנגדי. אם כן, הנקודה מועתקת לתוך מערך חדש.

במקרה הגרוע אין נקודות הנמצאות בטווח זה ולכן נעבור על כל הרשימה בזמן ריצה של O(n).

getDensity- מחשבת את צפיפות הנקודות שבמבנה הנתונים.

מתבצעות פעולות יסוד)השמות ופעולות

חשבון.(זמן הריצה של השיטה הוא O(1).

narrowRange- השיטה קוראת לפונקציות העזר removeByMin ו removeByMax.

ב removeByMin הפונקציה מסירה את כל האיברים החל מהאיבר הראשון ברשימה עד לאיבר האחרון שקטן מהערך המינימלי הנתון. ב- removeByMax הפונקציה מסירה את כל האיברים הגדולים מהערך המקסימלי הנתון החל מהאיבר האחרון. הפונקציה מסירה את הנקודות אשר לא נמצאות בטווח בשני הצירים. הפונקציה עוברת רק על כמות האיברים שצריך להסיר ולכן זמן הריצה של הפונקציה הוא O(|A|) .

getLargestAxis – הפונקציה בודקת את גדלי הצירים בעזרת פעולה חשבונית פשוטה ומחזירה את הציר הגדול יותר ולכן זמן הריצה הוא O(1).

getMedian – הפונקציה עוברת על הרשימה עד התא האמצעי ברשימה ומחזירה את הערך החציוני של הרשימה לפי הציר נתון, ולכן זמן הריצה שלה הוא o(n).

nearestPairInStrip- פונקציה זו מתחילה מאמצע מבנה הנתונים ומתקדמת אחד קדימה)/אחורה(ובודקת האם ערך הקואורדינטה לפי הציר הנתון נמצאת בטווח שחישבנו.
ברגע שהערך הבא לא נמצא בטווח, הפונקציה מסמנת את הנקודה הנוכחית להיות האחרונה)/הראשונה(ברצועה.
תוך כדי כך, סופרים את מספר הנקודות שנמצאות בטווח של הרצועה.

לפי מספר הנקודות ברצועה והגודל של מבנה הנתונים נחשב לפי איזה זמן ריצה הפתרון יתבצע

O(min(n, |B|log|B|).

נרצה למיין את הרשימה כך שתהיה ממוינת לפי הציר השני.

עבור פתרון ב O(n)- נשתמש בפונקציה getPointsInRangeOppAxis שזמן הריצה שלה הוא O(n) על מנת למיין את המערך לפי הציר השני.

עבור פתרון ב O(|B|log|B|) - נעתיק למערך את הנקודות הנמצאות ברצועה)זמן הריצה של פעולה זו הוא O(|B|)). לאחר מכן נשתמש בפונקציה sort.Arrays על מנת למיין את המערך לפי הציר השני .
הפונקציה sort.Arrays משתמשת בcomparator שמשווה את ערכי הנקודות לפי הציר השני.
לפי הנחה זמן הריצה של הפונקציה הוא O(|n|log|n|), כאשר |n| הוא מספר האיברים במערך ולכן הפונקציה ממיינת מערך בגודל B בזמן O(|B|log|B|), ולכן זהו זמן הריצה של הפתרון.

במערך הממוין לפי הציר השני שקיבלנו מהפתרון המתאים, נחשב את המרחק בין כל 2 נקודות במערך, ובכל פעם שנמצא מרחק קצר יותר נעדכן את 2 ערכי המערך המוחזר, להיות 2 הנקודות עם המרחק המינימלי.
לצורך כך, השתמשנו באלגוריתם מוכר לחישוב זוג נקודות הכי קרובות, לפיו די להשוות כל נקודה עם 7 הנקודות הבאות אחריה על מנת להבטיח שמצאנו את שתי הנקודות שמרחקן זו מזו הוא מינימלי.

-nearestPair זוהי פונקציה רקורסיבית למציאת שתי הנקודות הקרובות ביותר ברשימה. כל עוד גודל הרשימה גדול מ 2- נעתיק את מבנה הנתונים ל2 מבני נתונים חדשים וכך נקבל 2 רשימות חדשות- רשימה שמאלית)מהאיבר הראשון לאחד לפני החציון(ורשימה ימנית)מהחציון ועד לאיבר האחרון(.בכל פעם נפעיל את הפונקציה על כל צד שמאלי וצד ימני של הרשימה. נחשב עבור כל שתי נקודות שחוזרות מהרקורסיה את המרחק המינימלי בין שני הזוגות. כעת המרחק המנימלי השמור הוא מרחק של שתי נקודות באותו צד של המבנה הנתונים.
תנאי העצירה- כאשר הרשימה התפצלה לגודל הקטן מ2 נחזיר null. כאשר הרשימה התפצלה לגודל השווה ל2, נחזיר את זוג הנקודות שקיבלנו .
הפונקציה בודקת גם את המקרה שבו שתי הנקודות הקרובות ביותר נמצאות בצדדים שונים של מבני הנתונים. זאת עושים ע"י הפונקציה nearestPairInStrip. לפי אלגוריתם של חישוב שתי נקודות הכי קרובות, אם שתי הנקודות נמצאות בצדדים שונים, מספיק לבדוק בטווח של החציון ועוד המרחק המינימלי ששמרנו ופחות המרחק המנימלי ששמרנו, כדי לגלות האם קיימות 2 נקודות כאלה. נחזיר את שתי הנקודות בעלות המרחק המינימלי מבין המרחק השמור)כאשר 2 הנקודות נמצאות באותו צד של הרשימה(והמרחק שקיבלנו משתי הנקודות שהפונקציה nearestPairInStrip מחזירה.

הפונקציה split

- הפונקציה מחזירה מערך דו מימדי מטיפוס Container[][] בגודל 2x2.
  בתא 0[]0[ישמר הקונטיינר שהוא האיבר הראשון באוסף הקטן.
  בתא 0[]1[ישמר הקונטיינר שהוא האיבר האחרון באוסף הקטן.
  בתא 1[]0[ישמר הקונטיינר שהוא האיבר הראשון באוסף הגדול.
  בתא 0[]1[ישמר הקונטיינר שהוא האיבר האחרון באוסף הגדול.

הגישה לכל איבר באחד משני האוספים תהיה באותו אופן בו עוברים על מבנה הנתונים המקורי- בעזרת שימוש במצביע next או prev על הקונטיינר השמור במערך.

- פסואדו קוד

```
split (int value, Boolean axis)

.1 Container [][] array = new Container [2][2]

.2 Container start = getFirstByAxis(axis)

.3 Container end = getLastByAxis(axis)

.4 array[0][0] = start

.5 array[1][1] = end

.6 While (getByAxis(axis,start.getData)<value && getByAxis(axis,end.getData)>=value)

.7         start = start.getNext(axis)
```

```
.8        end = end.getPrev(axis)
.9 If (getByAxis(axis,start.getData)>value)
.01                array[0][1] = start.getPrev(axis)
.11                array[1][0] = start
.21 return array.
.31 If (getByAxis(axis,end.getData)<=value)
.41                array[0][1] = end
.51                array[1][0] = end.getNext(axis)
.61 return array
```

•**<u>ניתוח זמן הריצה</u>**

זמן הריצה של הפונקציה split הוא O)|C|(כאשר| C| הוא מספר הנקודות באוסף הקטן יותר.

יש שני מצביעים- אחד לתחילת הרשימה שמתקדם קדימה ואחד לסוף הרשימה שמתקדם אחורה.

לולאת הwhile ממשיכה לקדם את המצביעים מההתחלה ומהסוף כל עוד שהמצביע הראשון קטן מvalue וגם המצביע השני גדול או שווה לvalue.

בצורה זאת, המצביעים משווים בו זמנית את הערכים ממבנה הנתונים לvalue הנתון. ברגע שאחד מהם מוצא את הvalue)זאת אומרת שהתנאי מלולאת הwhile מפסיק להתקיים(, נפצל את מבנה הנתונים .

כל מצביע עובר על| C| איברים ומבצע מס' כלשהו a של פעולות יסוד שהסיבוכיות שלהן היא O)1(עד שמתבצע הפיצול ומכאן שזמן הריצה הוא a*O)1(=C*)**O|C|(** .

<u>ניתוח זמן הריצה של הפונקציה nearestPair</u>

בפונקציה מתבצעות פעולות יסוד של השמה, תנאים, אריתמטיקה וכו'.. הן מתבצעות בזמן קבוע של O)1(.

בנוסף מתבצעות קריאות לפונקציות הבאות:

getMedian- זמן ריצה O)n(

DistanceCalculator- זמן ריצה O(1)

copyStructure- זמן ריצה של $O(n^2)$

מכיוון שזמן הריצה של copyStructure הוא המשמעותי ביותר ,זמן הריצה של הפונקציה ללא הקריאות הרקורסיביות הוא $O(n)^2$.

קריאה רקורסיבית לnearestPair- בכל פעם שולחים חצי ממבנה הנתונים לפונקציה ולכן כל חלק מתבצע ב- $T\left(\frac{n}{2}\right)$, ומכיוון שהפעולה הרקורסיבית פועלת עבור שני מבני נתונים שכל אחד מהם מכיל $\frac{n}{2}$ איברים, זמן

הריצה של שתי הקריאות הוא $T2(\frac{n}{2})$.

על מנת לחשב את זמן הריצה של הפונקציה כולה נחבר את זמני הריצה של הקריאות הרקורסיביות עם שאר הפונקציה ונקבל:

$$T(n) = 2T(\frac{n}{2}) + O(n^2)$$

נפתור בשיטת המאסטר:

- אם3לפי מקרה $f(n) = \Omega(n^{\log_b a+\epsilon})$ וגם קיים $c < 1$ כך ש $af(\frac{n}{b}) < cf(n)$ לכל n מספיק גדול אז $T(n) = \theta(f(n))$.

נבדוק עבור הנוסחה שקיבלנו:

a=2, b=2, f(n)=n^2

$$n^2 = \Omega(n^{\log_2 2+\epsilon}) = \Omega(n^{1+\epsilon})$$

כלומר, נראה שקיימים c,n0>0 כך ש:

$$> c * n^{1+\epsilon} \leq n^2 \ ; n0 > n, c < 0$$

עבור 1c= 0 וגם n0= 1 אי השוויון מתקיים.

בנוסף, קיים $c > 1$ כך ש

$$2f\left(\frac{n}{2}\right) < c * n^2$$

$$2 * ((\frac{n}{2})^2) < c * n^2$$

$$\frac{n^2}{2} < c * n^2$$

מתקיים עבור c=0.75, n=1

ולכן, זמן הריצה של הפונקציה nearestPair הוא $T(n)= \theta(n^2)$.

חלק ג – GUI

כל ילד בשנות התשעים וכמובן גם כיום מכיר את המשחק, אנימציה, סדרה פוקימון.

וכמו הרבה ילדים בשנות ה90 הרבה היו משחקים בקונסולות גיימבוי וגם במחשב במשחקי פוקימון.

אחד מהמשחקים המפורסמים שפוקימון היו מוציאים זה פוקימון זהב ופוקימון כסף. לכל משחק היה את המקומות הייחודים שלו ולכל מקום היה את הסוד או הפוקימון הנדיר שבו. מי כמוני יודע שלפעמים זה נחמד לחזור לעבר ולשחק שוב עם מההתחלה במשחקים שריגשו פעם אותנו. אך, עבר כל כך הרבה זמן ואין ממש כח או רצון כמו פעם לחפש את אותם מקומות כי במפה מתוארים רק המקומות שבהם עברת.

וגם לכל פוקימון שאתה מחפש יש תיאור באילו מקומות הוא נמצא, אך אינך תמיד יודע.

אז אמרתי מה אם יהיה לנו מפה שמביאה את כל המיקומים של כל הערים, שבילים שונים, מנהרות או בניינים מיוחדים. במקרה כזה, כל מי שירצה לשחק שוב ידע תמיד איפה הפוקימון שהוא רוצה נמצא או פשוט לאן הוא צריך ללכת שוב והאם יש קיצורי דרך.

ולכן תמצאו בקובץ תיאור קורדינטות של כל הקיצורי דרך מקומות מיוחדים ועוד עם המפה המקורית של המשחק. לדוגמא תמונת מסך: