תרגיל בית רטוב 2 מבני נתונים 234218

צפריר ריהן 039811880 tzafrir@cs.technion.ac.il

איליה לסוחין 305930133 silyal@t2.technion.ac.il

להחזיר לתא 88

תיאור מבנה הנתונים

מבנה הנתונים הדרוש DS ממומש באמצעות מחלקה בשם RectangleLand המכילה:

•משתנים מספריים

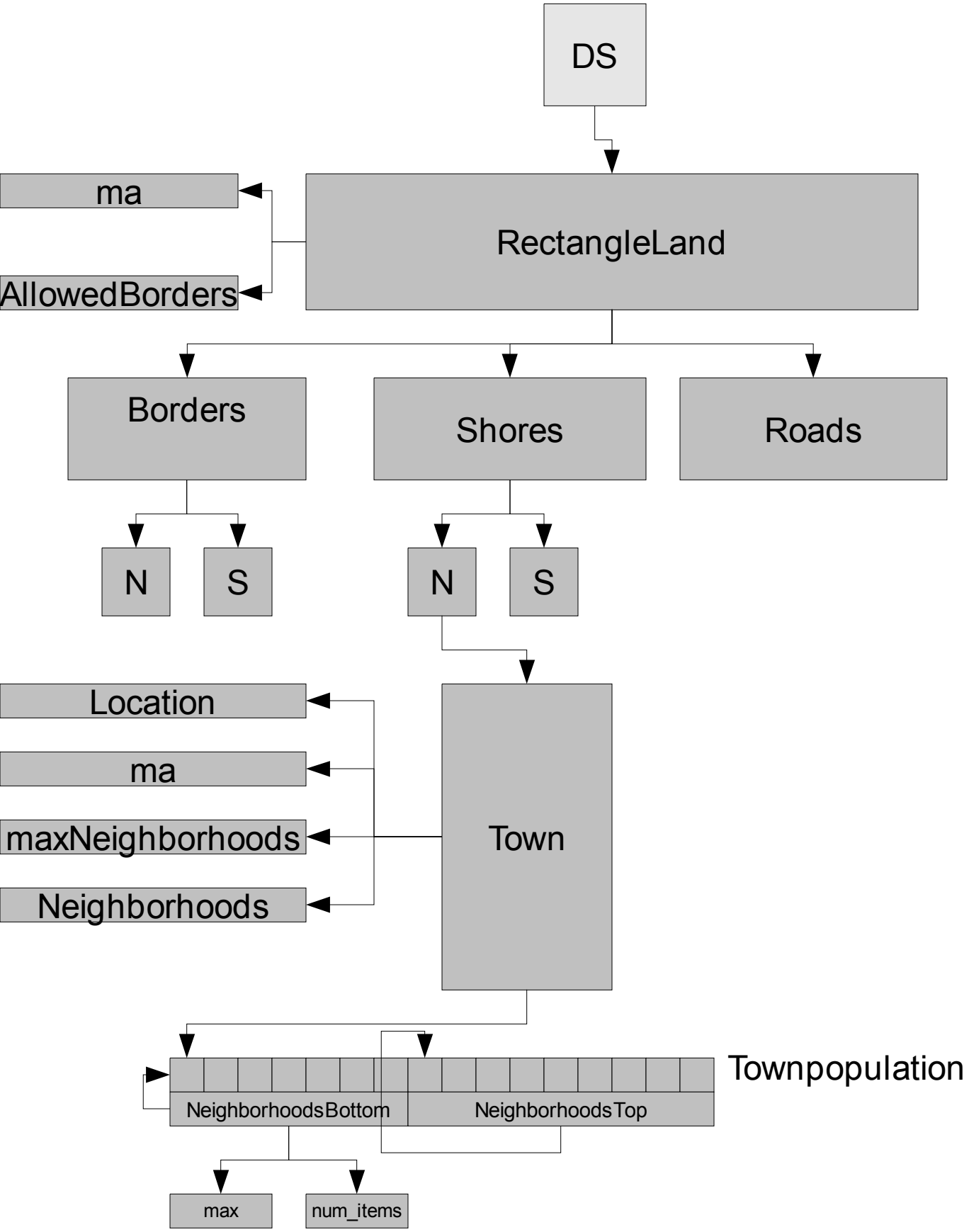
המשתנה ma שנועד עבור המפלצת

מספר הגבולות שניתן עוד להוסיף (ההפרש בין מספר הערים למספר הגבולות)

•עצי AVL המחזיקים מינימום ומקסימום, המחזיקים את הערים על שני החופים (צפוני ודרומי), את הכבישים ואת הגבולות, כאשר יש שני עצים, אחד הממיין את הגבולות לפי החוף הצפוני, והשני לפי הדרומי.

עיר ממומשת על ידי מבנה הנתונים Town, המכיל:

- משתנים מספריים
  - מספר השכונות המקסימלי המותר בעיר
  - המשתנה ma
  - מיקום העיר על החוף שבו היא נמצאת
  - מספר השכונות בעיר
- מערך המחזיק את השכונות בעיר, כאשר שכונה מתוארת על ידי מספר תושביה
- שתי ערימות המנהלות את המערך הנ"ל, כך שכל אחת שומרת את הגודל שלה, ומשמרת את תכונת הערימה עבור תת המערך בגודל זה – ערימה אחת היא ערימת מקסימום, והערימה השנייה מחזיקה את הערך השלילי של מספר התושבים בשכונה, כך שהיא מממשת ערימת מינימום


DS
ma
RectangleLand
AllowedBorders
Borders
Shores
Roads
N
S
N
S
Location
ma
maxNeighborhoods
Town
Neighborhoods
Townpopulation
NeighborhoodsBottom
NeighborhoodsTop
max
num_items

בהוכחות הסיבוכיות, אנו נשתמש בעובדות הבאות:

- מספר הערים בכל חוף קטן שווה למספר הערים הכולל, לכן פעולות המילון על עץ ערים של חוף מבוצעות ב $O(\log \text{מספר הערים בחוף})$, ולכן מבוצעות ב $O(\log n)$
- מספר הגבולות קטן שווה למספר הערים, לכן פעולות המילון על עץ הגבולות תבוצענה ב $O(\log n)$.
- מספר הכבישים קטן ממש מהמקרה בו ישנו כביש מכל עיר לכל עיר - n^2 . פעולות מילון על עץ הכבישים תבוצענה ב $O(\log(n^2))=O(2\log n)=O(\log n)$.
- העצים שאנו משתמשים בהם הם עצי מינימום/מקסימום, ומתחזקים את ערכי המינימום והמקסימום של כל תת עץ, המשתנים רק על מסלול החיפוש, ב $O(\log n)$. בעץ כזה, בחיפוש איבר, אנו מפיקים מידע של מינימום מקסימום.

מבנה הנתונים מממש את הפעולות הבאות:

Init:

תיאור האלגוריתם: מקצים זכרון לאובייקט חדש מסוג RectangleLand, פעולה זו גורמת לשמירת המשתנים המספריים, ולאתחול חמשת העצים.

הוכחת סיבוכיות: הקצאת זכרון מבוצעת ב $O(1)$, שמירת נתונים בזכרון מבוצעת ב $O(1)$, אתחול עץ ריק מבוצע ב $O(1)$, סה"כ: $O(1)$.

AddTown:

תיאור האלגוריתם: נסמן את המיקום המבוקש ב X. ידוע לנו החוף בו המקום המבוקש נמצא. לפי חוף זה, אנו מחפשים בעץ המתאים את הגבול הקרוב ביותר משמאל ל X, ומימין, בודקים שגבולות אלה אינם בנק' X, ואם אכן גבולות אלה אינם בנק' X, נקצה זכרון לעיר חדשה, ונוסיף אותה לעץ הערים של החוף המתאים. נגדיל באחד את מספר הגבולות המותר.
יצירת עיר חדשה מקצה זכרון למערך השכונות שלה, ומבצעת השמה של משתנים מספריים, ומחלקת את המערך הריק בין שתי הערימות.

הוכחת סיבוכיות: נסמן ב n את מספר הערים במבנה הנתונים. חיפוש הגבולות שתיארנו בעצים מבוצע ב $O(\log n)$, בדומה לחיפוש רגיל בעץ AVL. פעולת היצירה של עיר מבוצעות ב $O(1)$, הוספת איבר לעץ הערים של החוף הנוכחי מבוצעת ב $O(\log n)$, הגדלת משתנה מספרי מ בוצעת ב $O(1)$.

סה"כ: $2O(\log n)+O(1)+O(\log n)+O(1)=O(\log n)$.

AddRoad:

תיאור האלגוריתם: באמצעות שתי ערים זמניות, אנו בודקים ששתי הערים המבוקשות אכן קיימות על ידי חיפוש בעצי החופים הצפוני והדרומי. מקצים זכרון לכביש חדש.

מחפשים את הגבול המתחיל על החוף הצפוני בנקודה הקרובה היותר משמאל לעיר הצפונית של הכביש המבוקש, ובודקים האם הוא מגיע לנקודה בחוף הדרומי, הנמצאת מימין לעיר הדרומית של הכביש המבוקש – אם כן, אז הכביש המבוקש חותך את גבול זה. באופן דומה, נבדוק עבור הגבול הקרוב ביותר מימין.

מאחר שגבולות לא יכולים לחתוך זה את זה, מובטח שאם גבולות אלה לא חותכים את הכביש, אז אף גבול לא יחתוך את הכביש.

מוסיפים את הכביש לעץ הכבישים.

הוכחת סיבוכיות: יצירת ערים זמניות מבוצעת ב $O(1)$, חיפוש בעצי AVL מבוצע ב $O(\log n)$, הקצאת

זכרון לכביש מבוצעת ב $O(1)$, חיפוש בעצי הגבולות מבוצע ב $O(\log n)$, הוספת כביש לעץ הכבישים מבוצעת ב $O(\log n)$. סה"כ: $O(1)+2O(\log n)+O(1)+O(\log n)+O(\log n)=O(\log n)$.

:RemoveRoad

תיאור האלגוריתם: מנסים להסיר את הכביש הנתון מעץ הכבישים. אם הוא אכן היה בעץ והוסר, מוחקים אותו מהזכרון.

הוכחת סיבוכיות: הסרה מעץ הכבישים מבוצעת ב $O(\log n)$, מחיקת איבר בזכרון מבוצעת ב $O(1)$, סה"כ $O(\log n)$

:AddBorder

תיאור האלגוריתם: בודקים שלא נכניס יותר גבולות מערים, בודקים שאין ערים בקצוות הגבול.
בודקים שאין גבולות שחותכים את הגבול המיועד, כפי שתיארנו בהוספת כביש.
מבצעים בעץ הכבישים חיפוש עבור הנקודה על החוף הצפוני, ומניבים ממסלול החיפוש את הנקודה הימנית ביותר על החוף הדרומי שמגיע אליה כביש מנקודה כלשהי בצד שמאל של הנקודה על החוף הצפוני,
ובודקים האם נקודה זו נמצאת מימין לנקודה בחוף הדרומי אליה מגיע הגבול המתוכנן. אם כן, אזי ישנו כביש החותך את הגבול, ולא ניתן לבנות את גבול זה. באותו אופן, אנו מחפשים מהי הנקודה הקיצונית משמאל על החוף הדרומי אליה מגיע כביש מצד ימין של הנקודה על החוף הצפוני, והאם קיים כביש החותך את הגבול.

אם אין כביש החותך את הגבול המיועד, מוסיפים את הגבול החדש לעץ הממיין לפי החוף הצפוני, ולעץ הממיין לפי החוף הדרומי, ומקטינים את מספר הגבולות שניתן להוסיף באחד.

הוכחת סיבוכיות: אנו מבצעים שני חיפושים בעצי ערים, שני חיפושים בעצי גבולות, שני חיפושים בעצי כבישים, שתי הוספות לעצי גבולות.

סה"כ: $O(8\,logn)=O(\log n)$

:RemoveBorder

תיאור האלגוריתם: מסירים את הגבול משני עצי הגבולות, מוחקים את גבולות אלה מהזכרון, ומגדילים באחד את מספר הגבולות שניתן להוסיף.

הוכחת סיבוכיות: שתי הסרות מעצי גבולות, ושתי מחיקות מהזכרון, מבוצעות ב $2O(\log n)+2O(1)=O(\log n)$.

:AddNeighborhood

תיאור האלגוריתם: מחפשים את העיר הרלוונטית בחוף המתאים, אם יש פחות מ ma שכונות, מוסיפים את השכונה החדשה לערימה התחתונה, אחרת בודקים מה גודל השכונה ה ma בגודלה על ידי חיפוש האיבר המקסימלי בערימת המקסימום, שמים את הקטנה מביניהם בערמה התחתונה, והגדולה מביניהם בערימה העליונה. מעדכנים את מספר השכונות בעיר.

הוכחת סיבוכיות: חיפוש העיר מבוצע ב $O(\log n)$, ובמקרה הגרוע עלינו לבצע חיפוש איבר מקסימלי בערימת המקסימום המבוצע ב $O(1)$, פעולת הוצאה מערימה, ופעולת הכנסה על שתי ערימות. גודל כל אחת מהערימות קטן שווה ל m, לכן פעולות הכנסה והוצאה עליהן היא $O(\log m)$ כפי שנלמד בכיתה. הגדלת משתנה מספרי מבוצעת ב $O(1)$.

סה"כ: $O(\log n)+O(1)+3O(\log m)+O(1)=O(\log n)+O(\log m)=O(\log n+\log m)$

:AddManyNeighborhoods

תיאור האלגוריתם: בודקים שקיבלנו מערך חוקי, מחפשים את העיר בעץ הערים של החוף המתאים, מבטלים את סימן המינוס על חלק המערך ששיך לערימה העליונה, מעתיקים את השכונות החדשות לסוף המערך, מגדילים את מספר השכונות לפי מספר השכונות שהוספו, ומייצרים שתי ערימות חדשות, כך שבתחתונה נמצאים אברי המערך הקטנים שווים מהאיבר ה ma בגודלו, ובערימה העליונה איבר גדולים שווים לאיבר ה ma בגודלו.
לייצור ערימות אלו אנו משתמשים באלגוריתם Select שתואר בהרצאה על מנת לחלק את המערך באופן זה, מעדכנים את הסימן של האברים במערך המתאימים לערימה העליונה לסימן שליל, על מנת להתאים למימוש ערימת מינימום, ומבצעים makeHeap על שתי הערימות.

הוכחת סיבוכיות: בדיקת מערך הקלט מבוצעת ב $O(\text{גודל המערך})$, כאשר גודל המערך קטן שווה ל m, לכן מבוצע ב $O(m)$. חיפוש עיר מבוצע ב $O(\log n)$, שינוי הסימן למספר אברים הקטן שווה ל m מבוצע ב $O(m)$, Select על m אברים בשימוש עם חציון החציונים מבוצע ב $O(m)$ כפי שהוכח בהרצאה. שחזור הסימן מבוצע ב $O(m)$, בניית הערימות הממומשות באמצעות מערך מבוצעות ב $O(m)$ כפי שהוכח בכיתה, סה"כ: $O(m)+O(\log n)+O(m)+O(m)+O(m)+2O(m)=O(\log n+m)$

:MonsterAttack

מחפשים את העיר המתאימה, אם אכן יש מספיק שכונות, מחזירים את השכונה ה ma שהיא האיבר המקסימלי בערימה התחתונה, מחליפים אותה עם השכונה בגודל העוקב לה, הנמצאת בראש ערימת המינימום העליונה. מקטינים באחד את מספר השכונות בעיר.

הוכחת סיבוכיות: חיפוש עיר מבוצע ב $O(\log n)$. החלפת השכונות מבוצעת ב $O(1)$, הסרת איבר מהערימה העליונה מבוצע ב $O(\log m)$, שינוי משתנה מבוצע ב $O(1)$, סה"כ: $O(\log n+\log m)$.

:ChangeMa

משנים את משתנה ma לערכו החדש. מבצעים סיור inorder על שני עצי הערים, כאשר לכל עיר מבצעים פעולת שינוי ma המבוצעת על ידי ביטול סימן המינוס על אברי הערימה העליונה, שינוי ערך ma של העיר, ובניית ערימות מחדש כפי שתיארנו עבור AddManyNeighborhoods.

הוכחת סיבוכיות: שינוי ma של העיר ה i מבוצע ב $O(m_i)$ כאשר m_i הוא מספר השכונות בעיר.

סיור inorder על שני עצי הערים, כשעוברים על כל עיר פעם אחת ומבצעים עליה שינוי ma, מבוצע ב:

$$O(\sum_{i}^{n}(1+m_i))=O(n+\sum_{i}^{n}m_i)=O(n+M) \quad .$$

:Quit

תיאור האלגוריתם: מבצעים מחיקה על מבנה הנתונים מסוג RectangleLand – מחיקה זו גורמת למחיקת חמשת העצים שאנו מחזיקים על ידי סיור postorder. מחיקת עץ גוררת מחיקה של כל האברים שבו.

סיבוכיות מקום:

עבור מבנה נתונים המחזיק n ערים ו M שכונות, המקום שיתפסו שני עצי החופים הוא $O(n)$, המקום שיתפסו שני עצי הגבולות הוא $O(n)$, וכפי שהראינו, מספר הכבישים קטן שווה מ n^2 , לכן המקום שיתפוס עץ הכבישים הוא $O(n^2)$. בנוסף, כל עיר i מכילה מערך בגודל מספר השכונות המקסימלי האפשרי בה m_i , וסה"כ כל מערכים אלו יתפסו בזכרון $O(\sum_i^n m_i)=O(M)$.

סה"כ: $O(n)+O(n^2)+O(M)=O(n^2+M)$