פתרון מבחן מבני נתונים סמסטר א' תשס''ח

שאלה 2

חלק ראשון

הפעולות בהן אנו צריכים לתמוך הן

- Push – הכנס איבר למחסנית בזמן לוגריתמי
- Pop – הוצא את האבר האחרון שהוכנס למחסנית בזמן לוגריתמי
- Min – המחזיר את האבר הקטן ביותר במחסנית בזמן קבוע
- Delete-Min – מוחק את האבר הקטן ביותר במחסנית בזמן לוגריתמי

נממש את מבנה הנתונים בעזרת רשימה מקושרת דו-כיוונית וערמה בינארית (heap). ברשימה המקושרת נממש מחסנית רגילה כאשר בכל תא יש מצביע למיקום האבר בערמה. בערמה גם כן יש מצביע למיקום ברשימה בו נמצא האבר. בפעולת push/pop נבצע גם דחיפה/הוצאה מהרשימה בזמן קבוע וגם הכנסה/מחיקה מהערמה (בזמן לוגריתמי). פעולת min לוקחת זמן קבוע (ראש הערמה) ופעולת delete-min לוקחת זמן לוגריתמי כיוון שנמחק את הערך מן הערמה ומן הרשימה.

חלק שני

במקרה זה נצטרך לתמוך בפעולות pop,delete-min בזמן לוגריתמי (amortized). ואילו בפעולות push,min בזמן קבוע (amortized).

לצורך הפתרון נשתמש בערמה בינומית או ערמת פיבונאצ'י במקום ערימה בינארית. בנוסף נחזיק כמו בחלק הקודם רשימה מקושרת לצורך המחסנית. יש קישור דו-כיווני בין הערמה למחסנית (רשימה מקושרת).

נלמד בכיתה כי סדרת פעולות insert על ערמה בינומית/פיבונאצ'י נעשית ב-O(1) amortized.

פעולת min תתבצע ע"י min על הערימה. פעולת delete-min תתבצע ע"י מחיקה של אבר זה (זמן לוגריתמי) ומחיקתו מן המחסנית (ידוע מקומו שם ע"י מצביע מהערמה). פעולת push כוללת הכנסה לרשימה בזמן קבוע והכנסה לערמה. פעולת pop כוללת מחיקת ראש הרשימה בזמן קבוע ומחיקת האבר מן הערמה (זמן לוגריתמי).

הערה: חלק מן התלמידים השתמשו בעץ אדום-שחור כדי לתחזק את האברים הממוינים והסתמכו על כך שבעץ מסוג זה סדרת פעולות insert/delete נעשית בזמן amortized קבוע. אך שימו לב שלצורך כך צריך לדעת את מיקום ההכנסה! ובהכנסת מספר כללי למחסנית אין אנו יודעים היכן להכניס אותו. לכן לא נוכל להשיג סיבוכיות amortized קבועה באמצעות עץ אדום שחור.

חלק שלישי

נניח שניתן לבצע את כל הפעולות בזמן קבוע עבור סדרת פעולות. נתור את בעיית המיון במודל ההשוואות.

בהינתן n מספרים לא ממוינים נבצע את האלגוריתם הבא:

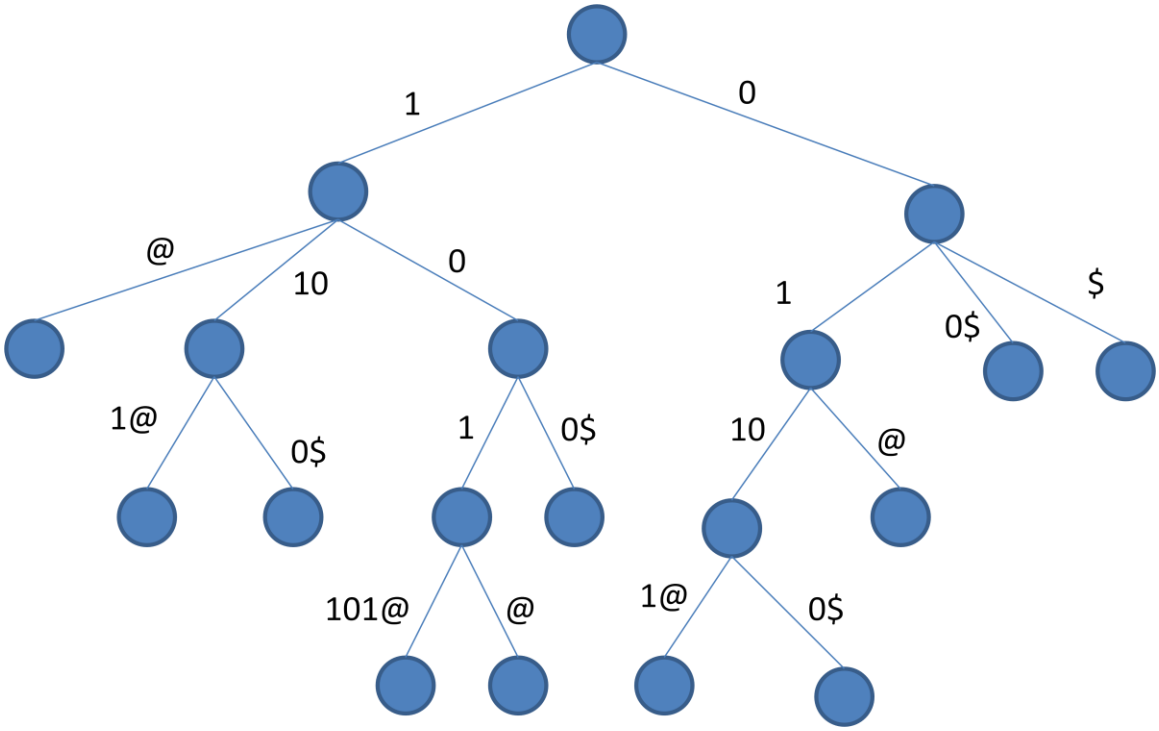
1. נבצע n פעולות push
2. נבצע n פעולות min ואחרי כל אחת delete-min.

סה"כ ביצענו 3n פעולות וקיבלנו סדרת אברים ממוינים. אנו יודעים שהחסם התחתון על מיון הוא O(nlogn) ולכן זו סתירה להנחה.

שאלה 5

חלק א'

צייר generalized suffix tree של 01100 ו-101101



חלק ב'

בהינתן מחרוזת s באורך n ניצור מחרוזת s' המשלימה ל-s (כל 0 הופך ל-1, כל 1 הופך ל-0). ניצור generalized suffix tree לשתי המחרוזות בזמן לינארי. מציאת המחרוזת הארוכה ביותר p כך שגם P וגם המשלימה לה הן תתי מחרוזות של s שקולה למציאת תת המחרוזת המשותפת הארוכה ביותר ל-s ו-s'. ולכן נחפש את הצומת העמוקה ביותר כך שבעלים היוצאים ממנה יש את סימני הזיהוי של s ושל s'. תת מחרוזת זו הינה תת מחרוזת p המבוקשת.

הערה: מספר תלמידים רשמו את האלגוריתם הבא:

נעבור רקורסיבית על העץ כאשר בכל צומת נרד במסלול רק אם יש לו מסלול מקביל מצומת זה עם ערכים הפוכים על הקשת. נפסיק את הרקורסיה כשאין יותר מחרוזת משלימה מתאימה. בסופו של דבר נחזיר את המחרוזת לאורך המסלול הארוך ביותר שצעדנו לאורכו.

האלגוריתם עובד ולכאורה הוא לינארי כי עוברים על צומת פעם אחת. אך שימו לב שכדי להשוות שתי קשתות צריך או לפתוח כל קשת לתווים בודדים או למצוא את ה-substring הכי ארוך (שתי האפשרויות שקולות). משמעות הדבר היא שכל השוואה יכולה לקחת O(n) ולכן סה"כ הסיבוכיות תהיה O(n^2). אלגוריתם זה אינו יעיל.