פרויקט מבני נתונים 2 – ערמת פיבונאצ'י

יונתן ניצן (yonatann2; 212825715) ואיתי סוניס (itaysonis; 213336977)

תיעוד חיצוני-

insert(int key, String info) - הפונקציה מקבל ערך, מחרוזת, יוצרת מהם צומת ומחברת אותו לשאר העצים בעזרת addToLinkedList. $O(1)$.

findMin()- הפעולה מחזירה מצביע לצומת של המינימום ששמורה בערמה. $O(1)$.

deleteMin()- הפונקציה מנתקת את המינימום מהערמה בעזרת detatchNode, ואז מפעילה את consolidate שימצא את המינימום החדש. $O(\log n)$ משום consolidate.

consolidate()- הפונקציה מחברת בין כל העצים כך שלבסוף יש רק עד עץ אחד לכל היותר מכל דרגה עד $O(\log n)$ דרגות, על ידי מעבר על העצים, מיונם לפי הדרגות ואיחודם כאשר נמצאו 2 עצים בעלי אותה דרגה, ואז העברתם לתא הבא. במקרה הגרוע הפעולה יכולה לקחת $O(n)$ אם יש הרבה עצים קטנים מאוד, אך בפועל כפי שראינו בהרצאה, ניתן להחשיב את הזמן שלוקח "לייצר" את אותם עצים בשאר הפעולות, כך שמדובר ב $O(\log n)$ אמורטייזד.

merge(HeapNode x, HeapNode y)- פעולת עזר לקודמת אשר מאחדת את העצים בפועל. מוצאת את הצומת הקטן מבין השורשים, ומטפלת במצביעים כך שיתקבל עץ חדש מדרגה אחת יותר.$O(1)$.

calculateMaxRank()- פעולת עזר גם היא לconsolidate המחשבת את הדרגה המקסימלית לעצים המאוחדים על מנת לייצר את המערך השומר אותם בעת האיחוד.$O(1)$.

decreaseKey(HeapNode x, int diff)- מורידה את המפתח של צומת בהתאם לפרמטר. לאחר מכן מפעילה את cascadeCut שמטפלת בהמשך ללא פרמטר המחיקה. $O(1)$ מפני שהפעולות שהתבצעו הם כאלה.

cascadeCut(HeapNode x, boolean delete)- הפעולה מקבלת מצביע לצומת ופרמטר מחיקה. היא מנתקת את הצומת מההורה, מוחק את הצומת עצמה אם נדרש, מסמנת את ההורה אם הוא לא מסומן, ואם הוא כן חוזרת את התהליך עם ההורה. כפי שראינו בהרצאה, הפעולה תתבצע בזמן קבוע אלא אם כן היא "משלמת" על סימונים של מחיקות קודמות, ולכן מדובר ב$O(1)$ amortized.

delete(HeapNode x)- הפונקציה מפעילה את deletemin אם מדובר במינימום, או את detatchNode אחרת. ומכך $O(\log n)$.

detatchNode(HeapNode x)- הפעולה מנתקת מההורה בעזרת cascadeCutאך הפעם עם פרמטר מחיקה חיובי, מוחקת את הצומת ממצביעי אחיו וילדיו, וכן מוסיפה את ילדין לרשימת העצים. לכן, היות והדרגה המקסימלית של צומת היא $O(\log n)$, במקרה הגרוע נאלץ לעבור על כל הילדים ולנתק ולחבר אותם בנפרד, ולכן $O(\log n)$.

totalLinks()- מחזירה את סך החיבורים שבוצעו בערמה. $O(1)$

totalCuts()- מחזירה את סך הניתוקים שבוצעו בערמה. $O(1)$

isEmpty()- מחזירה אם הערמה ריקה. $O(1)$

meld(FibonacciHeap heap2)- מחברת את רשימת העצים בפרמטר לרשימת העצים עליה הופעלה הפעולה. $O(1)$.

Size()- מחזיר את גודל העץ. $O(1)$

numTrees()- מחזיר את מספר העצים. $O(1)$

addToLinkedList(HeapNode node, HeapNode list)- פעולת עזר המתחזקת את המצביעים בחיבור צומת חדש לאחיו. $O(1)$

addToRootList(HeapNode node)- פעולת עזר המחברת עץ חדש לרשימת העצים. בפועל היא משתמשת בפעולה הקודמת לחבר את העץ לשאר השורשים, וכן יודעת ליצור מחדש את הערכים הבסיסים של הערמה כשהיא ריקה.$O(1)$

removeFromLinkedList(HeapNode node)- פעולת עזר המנתקת את הצומת מאחיה. $O(1)$

removeFromRootList(HeapNode node)- פעולת עזר המשתמשת בקודמה, אך גם דואגת כי הערמה לא תאבד את המצביע אם נמחק עץ אליו המצביע שלה מחובר. $O(1)$

חלק ניסויי

ניסוי 1

מספר סידורי i	זמן ריצה (מילישניות)	גודל הערמה בסיום	מספר חיבורים	מספר חיתוכים	מספר עצים בסיום
1	0.572625	6559	6550	0	9
2	1.179795	19681	19674	0	7
3	1.85256	59047	59037	0	10
4	5.413765	177145	177133	0	12
5	17.56426	531439	531427	0	12

תחילה, כל האיברים מוכנסים לרשימה, מה שכבר נותן $O(n)$, ולאחר מכן נמחק האיבר המינימלי כדי להפעיל את האיחוד, שגם הוא בפועל פונקציה של מספר העצים, (מפני ש$T > \log n$) ולכן גם הוא בזמן לינארי, ולסיכום הניסוי נמשך כ$O(n)$ זמן, כפי שניתן לחזות בניסויים הגדולים יותר.

כעת, נתבונן בווריאציות. בכל אחד משלבי האיחוד במהלך הconsolidate, הפונקציה לחלוטין אדישה לסדר מבחינת החיבורים. בכל שלב 2 צמתים של 1 מושווים ואחד מהם יחובר לשני, ואין משמעות אם בשלב כלשהו עץ מסדר גדול יחובר מתחת או מעל עץ מסדר גדול אחר. בנוסף, משום שיש מחיקה כאשר העץ עשוי מעצים של 1, אין סיבה לקיום חיתוכים. ולכן, היות ומספר החיתוכים והאיחודים קבוע, גם מספר העצים קבוע. באופן כללי, בניסוי זה מדובר פשוט בעץ בינומי עצל שנמחק וסודר מחדש, וכידוע יש דרך אחת רק לסדר עץ בינומי והיא תלויה רק במספר האיברים. לכן, גם זמן הריצה קבוע בערך.

ניסוי 2

מספר סידורי i	זמן ריצה (מילישניות)	גודל הערמה בסיום	מספר חיבורים	מספר חיתוכים	מספר עצים בסיום
1	2.72577	3280	39849.95	36574.95	5
2	5.14777	9841	135247.5	125413.5	7
3	17.9572	29524	450978.95	421462.95	8
4	81.23521	88573	1498950.15	1410389.15	12
5	382.958385	265720	4887870.35	4622159.35	9

גם כאן כל האיברים מוכנים ב$O(n)$, ואז מופעל deletemin, ובפרט consolidate n/2 פעמים. ההפעלה הראשונה עולה כn, אבל כמו שראינו כל הפעלה מכאן תעלה $O(\log n)$, והיות והיא תופעל n/2 פעמים על ערמה שמכילה מספר גדול מn/2 בכל הרצה, זאת תהיה העלות הדומיננטית והתוצאה תהיה $O(n \log n)$.

מבחינת הווריאציות- כפי שראינו בסעיף הקודם, לאחר מחיקה אחת הצורה קבועה ודטרמיניסטית ותלויה רק בכמות ההכנסות. מכאן ברור שיהיו שינויים- מחיקת מינימום של עץ בעל דרגה גדולה מאלץ לחתוך את כל הילדים שלו, שהם יותר מילדים של עץ בעל דרגה קטנה. על כן, חיתוכים מסוג זה ייצרו עצים רבים יותר. עם זאת, חשוב לשים לב שהיות ועדיין בניסוי זה הערמה מתפקדת לחלוטין בצורה זהה לעץ בינומי עצל, ישנה דרך אחת (מבחינת צורת העצים) להציג כל מספר (לאחר deletemin). מסיבה זו, מספר העצים בסוף

הconsolidate קבוע, ולכן אם יקרו חיבורים מדי מן הנדרש, הם מיד יתחברו חזרה ליצירת מספר (סוג) העצים הדטרמניסטי. כמובן כי זמן הריצה יהיה גדול יותר כאשר הוא ייאלץ לחתוך ולחבר יותר עצים (משמע ככל שהסידור המקורי רחוק ממוין).

ניסוי 3

מספר סידורי i	זמן ריצה (מילישניות)	גודל הערמה בסיום	מספר חיבורים	מספר חיתוכים	מספר עצים בסיום
1	0.750545	31	6550	6549.65	30.65
2	1.33798	31	19674	19673.95	30.95
3	3.32532	31	59037	59036.95	30.95
4	9.216225	31	177133	177133	31
5	82.088855	31	531427	531427	31

נתחיל מהמצב בניסוי 1, ואז נמחק כמעט את כל האיברים, מסדר $O(n)$. נשים לב כי העבודה על מחיקת כל אחד היא קבועה הפעם, היות וכל איבר שהוא מקסימלי בערמה בהכרח עם 0 ילדים, ולכן יוכל להמחק בזמן קבוע. מכאן הסיבוכיות הכוללת היא $O(n)$.

על מנת לנתח את הווריאציות, ההקבלה לעצים בינומיים נופלת, מפאת חוסר הודאות בנוגע למספר האיברים המדויק בעץ. הדבר הבולט ביותר היא אחידות החיבורים, ועובדה זו הגיונית משום שחיבורים מתבצעים רק בעת מחיקת המינימום, מה שקורה רק פעם אחת בתחילת הניסוי, והוסבר למה הוא קבוע בניסוי הראשון.

נתייחס עתה למספר החיתוכים ומספר העצים (שוודאי קשורים זה לזה). נגדיר את האיברים ה"שורדים" 2 עד 32 כקבוצה S. מספר העצים בסוף תלוי בפיזור האקראי של איברי S בתוך סדר ההכנסה, ומכך בעצים לאחר מחיקת המינימום בהתחלה. נבחין כי איבר מS יהפוך להיות שורש (אם לא כבר היה) רק כאשר יש לו שני בנים שאינם מS שנמחקים, ואז הcascadeCut יהפוך את האיבר לשורש, שם הוא ישאר.
כיוון שההכנסות מסודרות באקראי, הפיזור בין מספר האיברים הקטן הזה (S) מתוך מכלול האיברים הגדול מספק על מנת שרוב מוחלט של המקרים נסיים עם 31 שורשים, כפי שניתן לראות.

נשים לב שעבור עץ, חיתוך מפריד בין 2 עצים ובכך יוצר עץ חדש, וחיבור מאחד 2 עצים ובכך מוריד אחד מהכמות. היות ותמיד הבסיס היה n עצים, מספר העצים הכולל הוא-

$$T = n - Links + Cuts$$