Name: Ori Glassman, ID: 311453427, User Name: origlassman

Name: Itay Levi, ID: 312485386, User Name: itaylevi

מסמך תיעוד פרוייקט d-ary heap

הערימה DHeap מיוצגת ע"י איברי DHeap_Item. לכל DHeap ישנו שדה גודל מקסימלי, גודל נוכחי, איזה סוג ערימה (d) ואת הערימה העצמה.

לכל DHeap_Item ישנם 3 שדות: מפתח, שם והמיקום הנוכחי שלו במערך של הערימה.

- getSize()- תפקידה להחזיר את מספר האיברים בערימה, משמע להחזיר את השדה size של המחלקה.
- arrayToHeap(DHeap[] array1)- תפקידה לבנות ערימה חוקית בהינתן מערך כלשהו. לשימוש פונק' זו נעזרו במתודת העזר HeapifyDown(Dheap h, int i) אשר ממומשת בדומה לאלגוריתם בשקפים של ההרצאה. המתודה הסופית תחזיר את מספר ההשוואות שבוצעו בתהליך ההמרה לערימה.
- isHeap() – תפקידה להחזיר האם המערך היא ערימה חוקית, אשר מקיימת את כל המשתמרים ההוגדרו עבור מערך d-ary (לכל אבא יש d בנים, לא כולל הרמה האחרונה. כמו כן, כל אבא קטן מבניו). לכן, נעבור על האיברים בערימה ונבדוק שהמשתמר מתקיים.
- Parent(int I, int d) – מחזיר את המיקום של ההורה. מכיוון שישנה חוקיות במערך (שהרי הוא מייצג את הערימה) ניתן לדעת בכל רגע את האינדקס של ההורה.
- Child(int I, int k,int d) – בדומה ל parent(), מחזירה את האינדקס של הילד ה k של ההורה.
- Insert(DHeap_Item item) – תפקידה להכניס איבר DHeap_Item לערימה ולוודא שיווצר ערימה חוקית, לצורך כך הפונקציה נעזרת במתודת העזר heapifyUp אשר ממומשת בדומה לאלגוריתם בשקפים של ההרצאה. המתודה הסופית תחזיר את מספר ההושוואות שבוצעו בתהליך ההכנסה.
- Delete_Min() – תפקידה למחוק את האיבר המינימלי בערימה, ולהחזיר את מספר ההשוואות שבוצעו במהלך המחיקה. לצורך המחיקה הנ"ל, נשתמשת ב heapifyDown על מנת לוודא שהערימה חוקית.
- Get_min() – מחזירה את האיבר שערכו הוא המינימלי בערימה.
- Decrease_Key(DHeap_Item item, int delta) – תפקידה להוריד ב delta צומת item כלשהי, ולהחזיר את מספר ההשוואות שבוצעו על מנת לוודא שזו נשארה ערימה חוקית (ולכן משתמשת ב heapifyUp).
- Delete(DHeap_Item item) – תפקידה למחוק את האיבר Item מהערימה ולשמור על המשתמר של הערימה, ובסופה להחזיר את מספר ההשוואות שבוצעו בהמלכה. ע"מ לשמור על המשתמר, מתבצע heapifyDown. האלגוריתם של המתודה Delete הינה להחליף את האיבר הנמחק עם האיבר האחרון בערימה, ולבצע heapifyDown.

- DHeapSort(int[] array1, int d) – תפקידה למיין מערך של מספרים בעזרת הערימה, ומחזירה את מספר ההשוואות שבוצעו בתהליך זה. האלגוריתם הינו לבנות ערימה חוקית עם המספרים מהמערך, דבר זה מתבצע בעזרת המתודה arrayToHeap(). לאחר מכן, מתבצעת מחיקת האיבר המינימלי והכנסתו למערך.

מדידות

arrayToHeap-DHeapSort comparisons count			
m\d	2	3	4
1000	16845.90	18085.09	19427.31
10000	235331.30	250143.73	267853.47
100000	3018685.90	3198359.79	3396461.18

Decrease-key comparisons count			
x\d	2	3	4
1	180.90	104.49	61.25
100	49651.30	30320.93	19869.09
1000	190649.10	119280.61	80429.76

ArrayToHeap ואז DHeapSort:

מספר השוואות אסימפטוטי: חסם תחתון – נשים לב שאנו נמצאים במודל ההשוואות. מספר העלים בערימה הוא n, כמו כן מספר הפרמוטציות עליהם הוא !n ועל כן גובה עץ ההשוואות וחסם תחתון לזמן ריצת האלגוריתם הוא $\Omega(nlogn)$ (חסם תחתון לזמן הריצה של כל מיון השוואות). חסם עליון – האלגוריתם שלנו פותר את הבעיה ב - $O(nd\log_d n)$ כפי שמוסבר בהמשך, ולכן זהו חסם עליון לזמן הריצה. סה"כ **נקבל סיבוכיות זמן ריצה $\Theta(nd\log_d n)$.**

כפי שניתן לראות בתוצאות המצורפות לעיל, קצב הגידול תואם את זה האסימפטוטי -מספר פעולות ההשוואה גדל כתלות ב-d וב-n בהתאם לקצב שתואר קודם.

Decrease-Key:

ביצוע Decrease-Key במקרה הגרוע על איבר בודד יעלה $O(\log_d n)$ (כגובה העץ) כפי שמתואר בטבלה מטה. מכיוון שיש n איברים, כל איבר יבצע heapifyUp כגובה העץ, לכן **הסיבוכיות הנ"ל תהא $\Theta(\mathbf{nlog}_d n)$.**

כפי שניתן לראות בתוצאות המצורפות לעיל, מספר פעולות ההשוואה קטן כאשר d גדל כצפוי, וכן מספר פעולות ההשוואה כתלות ב –n תואם לקצב הגידול האסימפטוטי שצוין קודם.

טבלת סיבוכיות

שם המתודה	סיבוכיות	הסבר
*heapifyUp(DHeap h, int i)	$O(\log_d n)$	ב W.C נעבור במסלול מהעלה עד השורש וכל פעם נבצע החלפה בין האבא לאיבר, לכן סה"כ פעולות כגובה העץ.
*heapifyDown(DHeap h, int i)	$O(d \log_d n)$	ב W.C נעבור מהשורש לעלה, כל פעם נבדוק d איברים האם הם גדולים מהאבא שלהם, דבר זה מתבצע עבור כל רמה עד העלה ולכן הסיבוכיות הנ"ל.
getSize()	O(1)	החזרת השדה size
arrayToHeap(DHeap[] array1)	O(n)	ברמה 1 נבצע הכי הרבה n/d פעמים heapifyDown, ברמה השנייה n\d^2 וכן הלאה. לכן נקבל : $\sum_h^H h\frac{n}{d^h} < \sum_{h=1}^H h\frac{n}{2^h} =$ $O(n)$ כאשר השוויון האחרון הוכח בהרצאה.
isHeap()	O(d*n)	נעבור על כל ההורים וכל פעם נבדוק את כלל הערימה, קרי האם קיימים בנים הקטנים מההורה.ישנם O(n) צמתים שבכל צומת d בנים לכן הסיבוכיות המצויינת.
Parent(int I, int d)	O(1)	מחזיר את האינדקס המתאים של ההורה (הידוע מראש לפי כללי הערימה)
Child(int I, int k,int d)	O(1)	בדומה לparent
Insert(DHeap_Item item)	$O(\log_d n)$	הכנסה פיזית למערך ב(O(1, וקריאה ל heapifyUp, ולכן הסיבוכיות של ההכנסה תהיה זו של heapifyUp.
Delete_Min()	$O(d \log_d n)$	החלפת שני איברים במערך, שינוי של השדה גודל המערך, כל זה ב(O(1. לאחר מכן, קריאה ל heapifyDown ולכן הסיבוכיות של המחיקת המינימלי תהיה זו של heapifyDown.
Get_min()	O(1)	החזרת האיבר באינדקס הראשון במערך.
Decrease_Key(DHeap_Item item,int delta)	$O(\log_d n)$	הורדת ערך ממפתח (O(1, אך בדיקת תקינות המערך ע"י קריאה ל heapifyUp ולכן הסיבוכיות שלה תהיה זו של heapifyUp.
Delete(DHeap_Item item)	$O(d \log_d n)$	נבצע decrease_key לאיבר הרצוי כך שהוא יהיה מינימלי $O(\log_d n)$. לאחר מכן נבצע delete_min הלוקח $O(d \log_d n)$. מכיוון שאלו בטור, הסיבוכיות תהיה של זו הגבוהה יותר
DHeapSort(int[] array1, int d)	$O(n * d \log_d n)$	מציאת האיבר המינימלי (O(1. מחיקת האיבר המינימלי n פעמים תביא אותנו לסיבוכיות הנ"ל.

* מייצגת מתודת עזר