קובץ תיעוד – יותם מנה 204717862, דניאל פאר 206283442

מחלקת DHeap:

שדות:

private int size: שדה מופע מטיפוס int השומר את מספר האיברים הנוכחי בערימה.
Private int max_size: שדה מופע המגדיר את מספר האיברים המקסימלי בערימה.
Private int d: שדה מופע המגדיר את מספר הילדים המקסימלי של כל איבר בערימה.
Public Dheap_Item array: שדה מופע המכיל מערך המשמש כמבנה הנתונים של הערימה.

מתודות:

public DHeap(int d, int maxSize) (בנאי): בנאי המחלקה, מאתחל את שדה size לאפס ואת שאר השדות לפי הארגומנטים שהוא מקבל.

Public int getSize(): המתודה מחזירה את ערך השדה size – מספר האיבר הנוכחי הערימה. סיבוכיות O(1).

Public boolean isHeap(): המתודה עוברת על איברי הערימה ובודקת שכל איבר קטן מילדיו. במקרה שאחד האיברים לא מקיים כלל זה, המתודה עוצרת את הבדיקה ומחזירה false. מכיוון שהמתודה עוברת על כל איבר בערימה ומשווה אותו ל-d הילדים שלו, סיבוכיות הריצה היא O(dn).

Public static int parent(int i, int d): המתודה מחזירה את האינדקס של ההורה של האיבר הנמצא במקום ה-i במערך המייצג את הערימה. סיבוכיות O(1).

Public static int child(int i, int k, int d): המתודה מחזירה את האינדקס של הילד ה-k של האיבר הנמצא במקום ה-i במערך המייצג את הערימה. סיבוכיות O(1).

Private int heapifyUp(int child): המתודה מקבלת אינדקס של איבר הנמצא בערימה ו"מקדמת" אותו במעלה הערימה כל עוד הוא קטן מההורה שלו. תוך כדי ריצת המתודה היא סופרת את מספר ההשוואות המבוצעות ומחזירה את הספירה. במקרה הגרוע נבצע heapifyUp לאיבר המהווה "עלה" בערימה והמפתח שלו הוא הקטן ביותר בערימה – נצטרך להעלות אותו לשורש העץ, כלומר לבצע $O(\log_d n)$ פעולות. המתודה מחזירה את מספר ההשוואות שביצעה בתהליך.

Public int insert(Dheap_Item item): המתודה מקבלת עצם מסוג Dheap_Item, מכניסה אותו בתור האיבר האחרון בערימה ולאחר מכן קוראת למתודה heapifyUp על מנת למקם אותו היכן שנדרש כדי לשמור על משתמר הערימה. פעולת ההכנסה למערך עולה O(1) אך פעולת heapifyUp "מייקרת" את השימוש לסיבוכיות $O(\log_d n)$. המתודה מחזירה את מספר ההשוואות שבוצעו במהלך ההכנסה.

Public int arrayToHeap(DHeap_Item[] array1): המתודה מקבלת מערך המכיל איברי ערימה ויוצרת ממנו ערימה חוקית. המתודה מחליפה את המערך הנוכחי שמייצג את הערימה בזה שהיא מקבלת כארגומנט, מעדכת את size בהתאם ומבצעת heapifyDown לכל איבר במערך החל מההורה של האיבר במיקום size-1 במערך ועד האיבר במיקום ה-0. סיבוכיות הריצה שלה היא $O(n)$ בהתאם להוכחה שהוצגה בכיתה. המתודה מחזירה את מספר ההשוואות שבוצעו במהלך בניית הערימה.

public int [] findMinChildIndex(int item): המתודה מקבלת אינדקס של איבר בערימה, עוברת על כל ילדיו ושומרת את האינדקס של הילד עם המפתח המינימלי באיבר הראשון במערך בגודל 2. במקום השני במערך המתודה שומרת את מספר ההשוואות שביצעה לצורך מציאת הילד המינימלי. במקרה הגרוע הילד המינילי הוא הילד ה-d של האיבר ולכן סיבוכיות הריצה היא O(d).

Private int heapifyDown(int item): המתודה מקבלת אינדקס של איבר בערימה, שולחת אותו לפונ' findMinChildIndex על מנת לדעת עם איזה מילדיו הוא צריך להתחלף ולאחר החילוף מבצעת קריאה רקורסיבית לביצוע חילופים נוספים עד שהאיבר מקיים את כלל הערימה. במהלך הריצה המתודה סופרת את מספר ההשוואות שבוצעו ומחזירה אותו. במקרה הגרוע נקרא למתודה עבור איבר הנמצא בראש העץ ומפתחו הוא הגדול ביותר בערימה, ובנוסף הילד המינימלי של כל איבר הוא הילד ה-d שלו, ולכן סיבוכיות המתודה היא $O(d\log_d n)$.

Public int deleteMin(): המתודה מעתיקה את האיבר האחרון במערך המייצג את הערימה למקום הראשון (דורסת את האיבר הראשון), מוחקת את האיבר האחרון ממקומו המקורי, וקוראת ל heapifyDown על האיבר החדש שנמצא במיקום הראשון המערך. המתודה מחזירה את מספר ההשוואות שבוצעו במהלך ריצתה. סיבוכיות המתודה נקבעת לפי סיבוכיות heapifyDown ולכן היא $O(d\log_d n)$.

public DHeap_Item Get_Min(): המתודה מחזירה את האיבר הראשון במערך המייצג את הערימה – סיבוכיות O(1).

public int Decrease_Key(DHeap_Item item, int delta): המתודה מקבלת פוינטר לאיבר בערימה ומספר שלם דלתא, ומחסירה מהמפתח של האיבר את דלתא. כתוצאה מכך משתמר הערימה עלול להיות מופר ולכן המתודה שולחת את item לאחר הקטנת המפתח למתודה heapifyUp. לבסוף המתודה מחזירה את מספר ההשוואות שבוצעו. במקרה הגרוע item הוא בתחתית הערימה והורדת המפתח גורמת לכל שהמפתח החדש של item הוא המינימלי בערימה ולכן מיקומו הסופי יהיה שורש העץ – לכן הסיבוכיות היא $O(\log_d n)$.

public int Delete(DHeap_Item item): המתודה מקבלת פוינטר לאיבר בערימה ומוחקת אותו ע"י הקטנת מפתחו למינוס אינסוף (Integer.MIN_VALUE) כך שבוודאות מיקומו הסופי יהיה בראש הערימה ולאחר מכן מבצעת קריאה ל deleteMin. המתודה מחזירה את מספר ההשוואות שבוצעו. במקרה הגרוע נרצה למחוק איבר הנמצא בתחתית הערימה ולכן הסיבוכיות היא

$$O(\log_d n) + O(d\log_d n) = O(d\log_d n)$$

public static int DHeapSort(int[] array1, int d): המתודה מקבלת מערך מספרים שלמים ומספר שלם d. ראשית יוצרת מערך Dheap_Item כאשר המפתח של כל איבר הוא המספר שהיה במיקום המקביל במערך array1. לאחר מכן המתודה יוצרת ערימה d-ארית ע"י קריאה למתודה arrayToHeap, ולבסוף מבצעת deleteMin כמספר האיברים בערימה, כאשר בכל מחיקה מפתח האיבר שנמחק נשמר בחזרה במערך array1. כתוצאה מכך נקבל מערך ממוין. את סיבוכיות המתודה נחשב לפי הסיבוכיות של כל שלב המתואר:

$$O(n) + O(n\log_d n) + O(nlog_d n) = O(nlog_d n)$$

public String toString(): מציגה כמחרוזת את הערימה – שימש אותנו בדיבאגינג. סיבוכיות O(n).

מחלקת Dheap_Item:

שדות:

private int key: מפתח האיבר

private String name: שם האיבר (הערך)

private int pos: מיקום האיבר במערך המייצג ערימה

מתודות:

בנאי: DHeap_Item(String name1,int key1): מאתחל את שדות האיבר לפי הארגומנטים, ואת pos ל 1-.

Getters , Setters

מדידות:

ניסוי ראשון: מיון ערימה

	M=1,000	M=10,000	M=100,000
d=2	16848	235333	3018443
d=3	16396	226570	2896439
d=4	17621	242732	3076779

בניסוי זה ביצענו מיון המשתמש במודל ההשוואות ולכן במקרה הגרוע נקבל מערך "ממוין הפוך" ובעץ ההשוואות נקבל את הפרמוטציה שנמצאת בעומק n log n ולכן חסם הדוק למספר ההשוואות הוא $\Theta(nlogn)$.

ניסוי שני: decrease key

	X=1	X=100	X=1000
d=2	99999	153028	303027
d=3	99999	130742	212854
d=4	99999	122948	181140

במקרה הגרוע נוכל לבנות ערימה שברמה התחתונה בעץ שמייצג אותה יש איברים שההפרש בינם לבין השורש קטן מדלתא. אם נבצע decrease-key(delta) על איברי הרמה התחתונה בסדר יורד, כל אחד מאיברים יגיע לראש הערימה בסוף תהליך ה heapifyUp. ברמה התחתונה של ערימה יש $\Theta(n)$ איברים וכל אחד מהם מבצע $\Theta(\log_d n)$ השוואות, לכן חסם הדוק למספר ההשוואות הוא $\Theta(n\log_d n)$.