תיעוד - DHeap

DHeap:

לאיבר במחלקה 4 שדות פרטיים:
- int size – מספר האיברים בערימה.
- int max_size – מספר האיברים המקסימלי בערימה, גודל המערך.
- int d – מספר הילדים המקסימלי של איבר.
- DHeap_Item[] array – מערך שמייצג את מבנה הערימה.

- **DHeap(int m_d, int m_size):**
  - בנאי המחלקה. מבצע השמה של שדה size ל-0, של שדה max_size ל-m_size, של דשה d ל-m_d ושל שדה array למערך חדש של DHeap_Item בגודל max_size.
  - סיבוכיות זמן ריצה O(1)

- **getSize()**:
  - מחזירה את השדה size שמייצג את מספר האיברים בערימה.
  - סיבוכיות זמן ריצה O(1)

- **arrayToHeap(DHeap_Item[] array1)**:
  - מקבלת מערך של איברים ומייצרת ערימה חדשה עם איבריו כפי שלמדנו בהרצאה. מחזירה את מספר ההשוואות הכולל שקרו בפעולות ה-HeapifyDown.
  - פונקציות עזר:
    - HeapifyDown
  - סיבוכיות זמן ריצה – O(n)

- **isHeap()**:
  - עוברת על כל הצמתים בערימה ובודקת שערך הצומת גדול מהערך של האב שלה. מחזירה true אם כן, אחרת false.
  - סיבוכיות זמן ריצה – O(n)

- **static parent(int i, int d):**
  - מקבלת אינדקס של איבר בערימה ואת d ומחזירה את האינדקס של אביו לפי החישוב שנלמד.
  - סיבוכיות זמן ריצה - O(1)

- **static child(int i, int k, int d):**
  - מקבלת אינדקס של איבר בערימה ואת d ומחזירה את האינדקס של הבן ה-k לפי החישוב שנלמד.
  - סיבוכיות זמן ריצה – O(1)

- **insert(DHeap_Item item):**
  - מקבלת איבר ומכניסה אותו לסוף המערך ו"מפעפעת" אותו למקום הנכון בעזרת HeapifyUp. מחזירה את מספר ההשוואות הכולל שקרו בפעולות ה-HeapifyUp.
  - פונקציות עזר:
    - HeapifyUp
  - סיבוכיות זמן ריצה – $O(\log_d n)$

- **Delete_Min():**
  - מוחקת את השורש של הערימה בעזרת Delete ומחזירה את מספר ההשוואות הכולל שקרו בפעולות ה-HeapifyDown/HeapifyUp.
  - פונקציות עזר:
    - Delete
  - סיבוכיות זמן ריצה – $O(d \log_d n)$

- **Get_Min():**
  - מחזירה את שורש הערימה (האיבר הראשון במערך) שהוא בעל המפתח הקטן ביותר.
  - סיבוכיות זמן ריצה – O(1)

- **Decrease_Key(DHeap_Item item, int delta):**
  - מקבלת איבר ומספר delta ומקטינה את המפתח של האיבר ב-delta. "מפעפעת" אותו למקומו בעזרת HeapifyUp.
  - פונקציות עזר:
    - HeapifyUp
  - סיבוכיות זמן ריצה – $O(\log_d n)$

- **Delete(DHeap_Item item):**
  - מקבלת איבר בערימה, מחליפה בינו לבין האיבר האחרון במערך ומקטינה את השדה size ב-1, בכך "מוחקת" את האיבר.
    מבצעת "פעפוע" של האיבר המוחלף למקום הנכון בעזרת heapifyUp ו-heapifyDown (רק אחד מהם יקרה) ומחזירה את מספר ההשוואות הכולל שקרו.
  - פונקציות עזר:
    - HeapifyUp
    - HeapifyDown
  - סיבוכיות זמן ריצה - $O(d\log_d n)$

- **DHeapSort(int[] array1, int d):**
  - מקבלת מערך של מספרים ומספר d, יוצרת ערימה d-ארית חדשה שמפתחות איבריה הם מספרי המערך ואז אחד אחד מכניסה את המספרים לפי סדרם למערך הקלט ע"י getMin ו-deleteMine כמו שלמדנו בהרצאה. מחזירה את מספר ההשוואות הכולל שקרו בפעולות arrayToHeap ו-deleteMin.
  - פונקציות עזר:
    - arrayToHeap

- Get_Min
- Delete_Min
- סיבוכיות זמן ריצה – $\Theta(dn\log_d n)$

- **HeapifyUp(DHeap_Item item):**
  - מקבלת איבר item שבערימה ומוודאת שהוא מקיים את תנאי הערימה עם אביו. אם לא והמפתח שלו קטן משל אביו, מחליפה בינם בעזרת swap ומבצעת את אותה בדיקה ברקורסיה עבור האבא החדש של item. מחזירה את מספר ההשוואות הכולל שהתבצעו עד לערימה תקנית.
  - פונקציות עזר:
    - swap
  - סיבוכיות זמן ריצה – $O(\log_d n)$

- **HeapifyDown(DHeap_Item item):**
  - מקבלת איבר item שבערימה ומוודאת שהוא מקיים את תנאי הערימה עם כל בניו. אם לא והמפתח שלו גדול משל הבן עם המפתח המינימלי, מחליפה בינם בעזרת swap ומבצעת את אותה בדיקה ברקורסיה עבור הבנים החדשים של item. מחזירה את מספר ההשוואות הכולל שהתבצעו עד לערימה תקנית.
  - פונקציות עזר:
    - swap
  - סיבוכיות זמן ריצה -$O(d\log_d n)$

- **swap(DHeap_Item item, DHeap_Item minChild):**
  - מקבלת שני איברים בערימה ומחליפה בין המקומות שלהם. מעדכנת את שדות pos של שניהם.
  - סיבוכיות זמן ריצה - O(1)

מדידות:

DHeapSort

d	m	ממוצע מספר ההשוואות
2	1,000	9,424.1
2	10,000	127,665.6
2	100,000	1,609,285.2
3	1,000	1,1935.2
3	10,000	161,046.0
3	100,000	2,030,999.8
4	1,000	14,209.6
4	10,000	192,056.3
4	100,000	2,407,483.6

ננתח את סיבוכיות זמן הריצה WC:

הפונקציה מבצעת לולאה ראשונה להעברת האיברים למערך ב-O(n), אחר כך נעזרת ב-arrayToHeap() שבכיתה למדנו שהיא גם O(n) ולבסוף מבצעת n פעמים getMin() (כל פעולה O(1) לכן סה"כ O(n)) ו-n פעמים deleteMin(). נראה חסם עליון ותחתון זהה ל-deleteMin():

ברור שלאחר כל קריאה, גודל הערימה קטן ב-1, ובכל קריאה יש לכל היות $\log_d n$ השוואות ב-heapifyDown וכל אחת מהן היא עבור d בנים. לכן מספר ההשוואות הוא (כאשר את המעבר הימני למדנו בהרצאה):

$$\Omega(dn\log_d n) = d\frac{n}{2}\log_d\left(\frac{n}{2}\right) \le d\sum_{k=\frac{n}{2}}^{n}\log_d k \le d\sum_{k=1}^{n}\log_d k = d\log_d(n!) = \mathrm{O}(\mathrm{dn}\log_d n)$$

סה"כ רצף פעולות deleteMin() הוא $\Theta(dn\log_d n)$.

סה"כ זמן ריצה WC של DHeapSort הוא $O(n) + O(n) + O(n) + \Theta(dn\log_d n) = \Theta(dn\log_d n)$

Decrease-Key

ממוצע מספר ההשוואות	x	d
100,000.0	1	2
152,947.0	100	2
303,277.3	1,000	2
100,000.0	1	3
130,918.3	100	3
213,265.7	1,000	3
100,000.0	1	4
122,951.5	100	4
181,200.9	1,000	4

ננתח את סיבוכיות זמן הריצה WC:

המקרה הגרוע ביותר הוא כאשר נבצע Decrease-Key() על עלה ונשנה את המפתח שלו להיות קטן משל השורש של הערימה. כך העלה יצטרך לעלות בעזרת HeapifyUp ולהחליף את השורש, מה שייקח $\Theta(\log_d n)$ (כפי שלמדנו בהרצאה).

אנו מבצעים פעולה זאת n פעמים לכן סה"כ זמן הריצה של סדרת הפעולות הוא $\Theta(n\log_d n)$.