ערימה
Heap

1

מה נלמד?

- מבנה נתונים ערימה
  - הגדרה
  - פעולות
  - שימושים

2

3

הגדרה

נתונה קבוצה דינאמית של איברים S , כאשר לכל איבר מצורף מפתח key

key	$Satellite\ data$

מינימום מינימלי

ערימת מקסימום היא מבנה נתונים מבוסס עץ שמאפשר גישה מהירה לערך מקסימאלי ב- S

או מינימום או מקסימום. לא בו זמנית

4

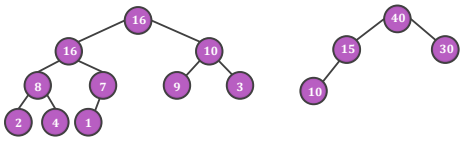
הגדרה - המשך

ערימה מקיימת את תכונת הערימה:

ערימת מקסימום:

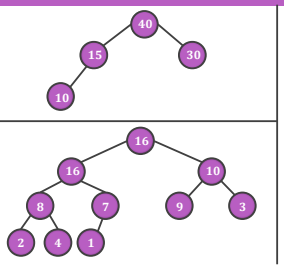
לכל צומת x:

$x.key \geq keys\ of\ its\ children$



5

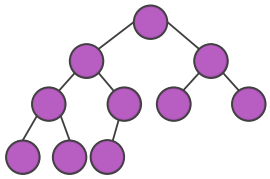
איפה בערימת מקסימום נמצא איבר בעל מפתח מקסימאלי?



1. באחד העלים
2. בשורש
3. לא ניתן לדעת בוודאות

6

הגדרה - המשך



ערימה מקיימת את תכונת הערימה:

ערימת מקסימום:
לכל צומת x:
$x.key \geq keys\ of\ its\ children$

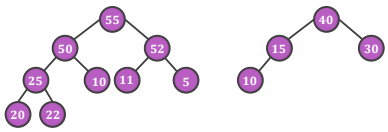
ערימת מינימום
לכל צומת x:
$x.key \leq keys\ of\ its\ children$

ערימה תיוצג על ידי **עץ שלם**

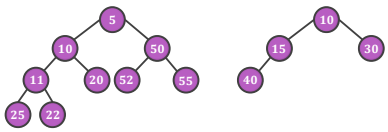
7

דוגמאות

ערימת **מקסימום**:



ערימת **מינימום**:



8

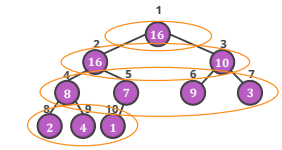
הגדרה - המשך

נממש ערימה באמצעות מערך
שורש העץ – $A[1]$
בהינתן אינדקס i של הצומת

left(i) **return** $2i$ **right(i)** **return** $2i + 1$	**parent(i)** **return** $\lfloor i/2 \rfloor$

למערך A המייצג ערימה שני מאפיינים:
- $A.length$ – גודל המערך
- $A.heap\_size$ – גודל הערימה

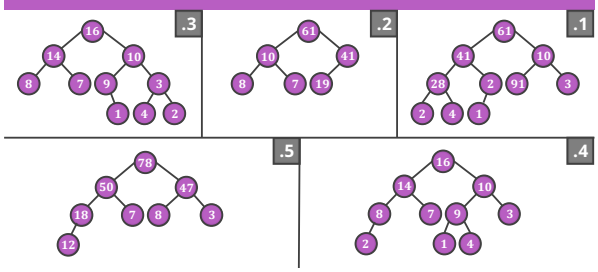
גובה של ערימה בעלת n איברים הוא $\lfloor \log n \rfloor$



	1	2	3	4	5	6	7	8	9	10
A	16	16	10	8	7	9	3	2	4	1

9

אילו מהעצים הבאים הם ערימת מקסימום?



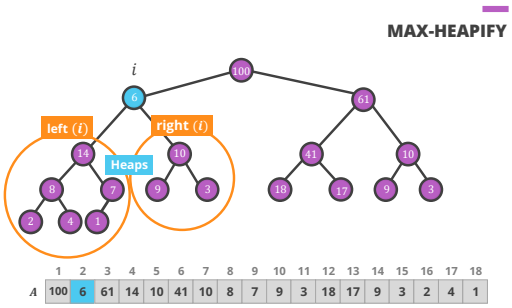
10

11

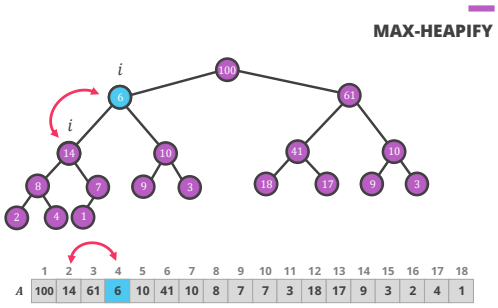
פעולות המוגדרות על ערימה

- *Build-Max-Heap* - יוצרת ערימה ממערך קלט בלתי ממוין
- *Heap-Max* - החזרת איבר בעל מפתח מקסימלי בערימת מקסימום
- *Heap-Extract-Max* - מחיקת איבר בעל מפתח מקסימאלי בערימת מקסימום
- *Heap-Increase-Key* - הגדלת מפתח
- *Max-Heap-Insert* - הכנסת איבר חדש
- *Max-Heapify* - שמירה על תכונת הערימה

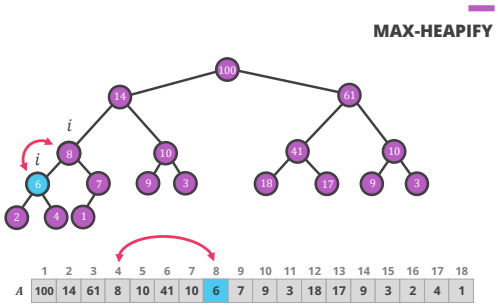
12


MAX-HEAPIFY
i
left (i)
right (i)
Heaps
A
100 6 61 14 10 41 10 8 7 9 3 18 17 9 3 2 4 1

13


MAX-HEAPIFY
i
i
A
100 14 61 6 10 41 10 8 7 7 3 18 17 9 3 2 4 1

14


MAX-HEAPIFY
i
i
A
100 14 61 8 10 41 10 6 7 9 3 18 17 9 3 2 4 1

15

MAX-HEAPIFY

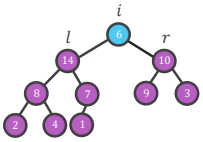
- **קלט:** מערך A ואינדקס i
- **ההנחה:** העצים הבינאריים המושרשים ב- $left\ (i)$ ו-$right\ (i)$ הם ערימות תקינות, אבל אולי $A[i]$ קטן מבניו, ובכך מפר את תכונות הערימה
- **פלט:** העץ ששורשו $A[i]$ הופך לערימת מקסימום

16

MAX-HEAPIFY

$Max-Heapify\ (A, i)$

```
l ← left(i)
r ← right(i)
if l ≤ A.heap_size and A[l] > A[i]
        largest ← l
else largest ← i
if r ≤ A.heap_size and A[r] > A[largest]
        largest ← r
if largest ≠ i
        exchange A[i] ↔ A[largest]
        Max-Heapify(A, largest)
```



17

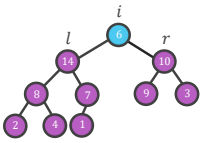
מה זמן הריצה של Max-Heapify

1. $O\ (\log n)$
2. $O\ (n^2)$
3. $O\ (n)$
4. $O\ (n \log n)$

18

MAX-HEAPIFY

```
Max − Heapify (A, i)
l ← left(i)
r ← right(i)
if l ≤ A.heap_size and A[l] > A[i]
        largest ← l
else largest ← i
if r ≤ A.heap_size and A[r] > A[largest]
        largest ← r
if largest ≠ i
        exchange A[i] ↔ A[largest]
        Max–Heapify(A, largest)
```



זמן ריצה:

$T(n) = O(h) = O(\log n)$

19

פעולות המוגדרות על ערימה

- *Build-Max-Heap* - יוצרת ערימה ממערך קלט בלתי ממוין
- *Heap-Max* - החזרת איבר בעל מפתח מקסימלי בערימת מקסימום
- *Heap-Extract-Max* - מחיקת איבר בעל מפתח מקסימאלי בערימת מקסימום
- *Heap-Increase-Key* - הגדלת מפתח
- *Max-Heap-Insert* - הכנסת איבר חדש
- *Max-Heapify* - שמירה על תכונת הערימה

20

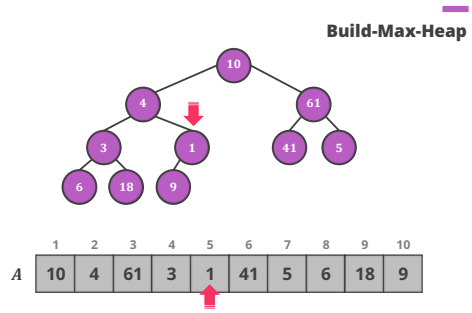
Build-Max-Heap

נתון: מערך A לא ממוין

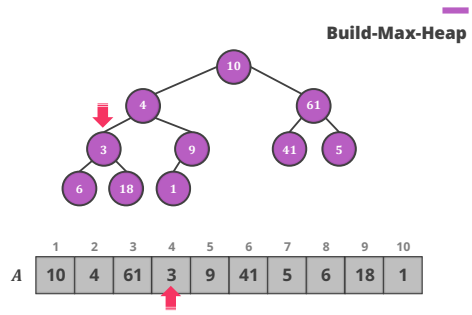
	1	2	3	4	5	6	7	8	9	10
A	10	4	61	3	1	41	5	6	18	9

מטרה: לבנות ערימת מקסימום מ- A

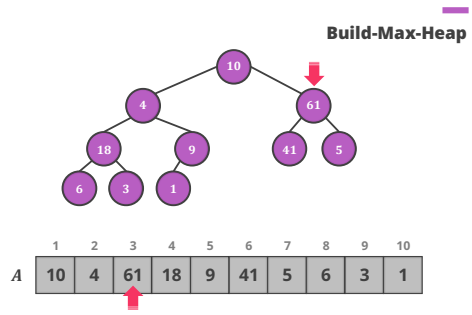
21


Build-Max-Heap
10
4
61
3
1
41
5
6
18
9
1 2 3 4 5 6 7 8 9 10
A
10 4 61 3 1 41 5 6 18 9

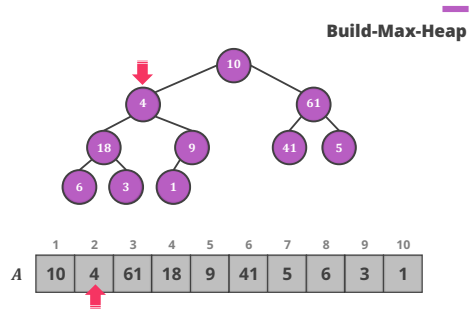
22


Build-Max-Heap
10
4
61
3
9
41
5
6
18
1
1 2 3 4 5 6 7 8 9 10
A
10 4 61 3 9 41 5 6 18 1

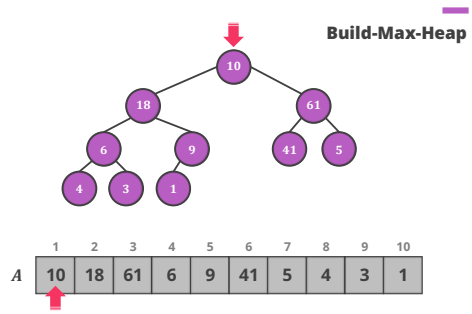
23


Build-Max-Heap
10
4
61
18
9
41
5
6
3
1
1 2 3 4 5 6 7 8 9 10
A
10 4 61 18 9 41 5 6 3 1

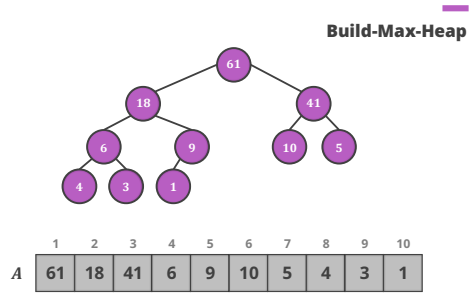
24


Build-Max-Heap
10
4
61
18
9
41
5
6
3
1
1 2 3 4 5 6 7 8 9 10
A
10 4 61 18 9 41 5 6 3 1

25


Build-Max-Heap
10
18
61
6
9
41
5
4
3
1
1 2 3 4 5 6 7 8 9 10
A
10 18 61 6 9 41 5 4 3 1

26


Build-Max-Heap
61
18
41
6
9
10
5
4
3
1
1 2 3 4 5 6 7 8 9 10
A
61 18 41 6 9 10 5 4 3 1

27

Build-Max-Heap

- האיברים $A[\left(\left\lfloor \frac{n}{2} \right\rfloor + 1\right) .. n]$ הם עלים בעץ ולכן כל אחד מהם הוא ערימה בגודל 1

	1	2	3	4	5	6	7	8	9	10
A	10	4	61	3	1	41	5	6	18	9

- נעבור על שאר הצמתים ונפעיל $Max-Heapify$ על כל אחד מהם

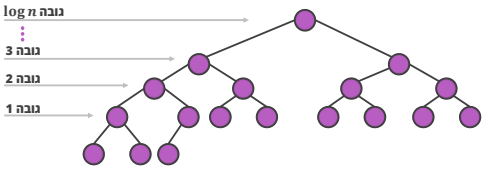
```
Build-Max-Heap(A)
A.heap_size ← A.length
for i ← ⌊A.length/2⌋ downto 1
    Max-Heapify(A, i)
```

28

זמן ריצה של Build-Max-Heap

- כל קריאה ל- $Max-Heapify$ מתבצעת ב- $O(\log n)$, ישנן $O(n)$ קריאות כאלה,
לכן זמן ריצה לכל היותר $O(n \log n)$

חסם זה אינו חסם הדוק אסימפטוטית



29

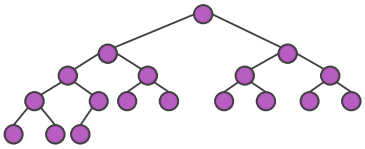
? למה לדעתכם שווה זמן ריצה של הפעולה $Build-Max-Heap$?

30

זמן ריצה של Build-Max-Heap

נוכיח שזמן ריצה של $Build\text{-}Max\text{-}Heap$ הוא $O(n)$

טענת עזר: מספר הצמתים שגובהם h בערימה בת n איברים הוא לכל היותר $\left\lceil \frac{n}{2^{h+1}} \right\rceil$



31

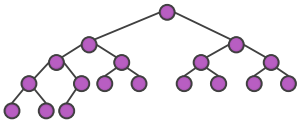
זמן ריצה של Build-Max-Heap

טענת עזר: מספר הצמתים שגובהם h בערימה בת n איברים הוא לכל היותר $\left\lceil \frac{n}{2^{h+1}} \right\rceil$

הוכחה: באינדוקציה על h.

- בסיס: $h = 0$

בערימה יש $\left\lceil \frac{n}{2} \right\rceil$ עלים

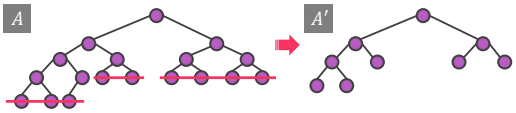


32

זמן ריצה של Build-Max-Heap

הנחה: נניח שהטענה נכונה עבור צמתים בעלי גובה $h \geq 0$

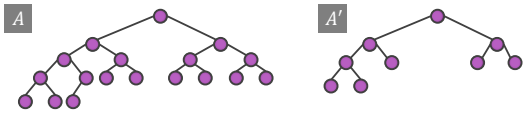
צעד: נוכיח את הטענה עבור צמתים בעלי גובה $h + 1$



צמתים בעלי גובה $h + 1$ ב- A הם צמתים בעלי גובה h ב-A'

33

אם מספר הצמתים בערימה A הוא n, מהו מספר הצמתים בערימה A'?



1. $\left\lceil \frac{n}{2} \right\rceil$
2. $\left\lfloor \frac{n}{2} \right\rfloor$
3. $\lceil \log n \rceil$
4. $\lfloor \log n \rfloor$

34

זמן ריצה של Build-Max-Heap

לפי הנחת האינדוקציה, מספר הצמתים בעלי גובה h ב- A' הוא לכל היותר:

35

זמן ריצה של Build-Max-Heap

זמן ריצה הכולל של *Build-Max-Heap* הוא $O(n)$

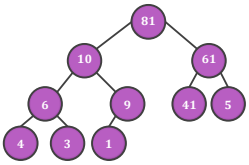
36

פעולות המוגדרות על ערימה

- *Build-Max-Heap* - יוצרת ערימה ממערך קלט בלתי ממוין
- *Heap-Max* - החזרת איבר בעל מפתח מקסימלי בערימת מקסימום
- *Heap-Extract-Max* - מחיקת איבר בעל מפתח מקסימאלי בערימת מקסימום
- *Heap-Increase-Key* - הגדלת מפתח
- *Max-Heap-Insert* - הכנסת איבר חדש
- *Max-Heapify* - שמירה על תכונת הערימה

37

Heap-Max

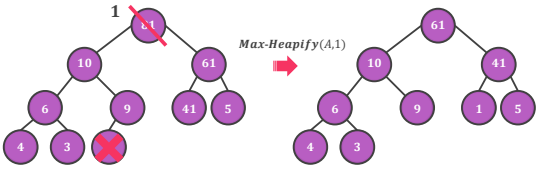



```
Heap-Max(A)
if A.heap_size < 1
    error "heap underflow"
return A[1]
```

זמן ריצה: $T(n) = O(1)$

38

Heap-Extract-Max

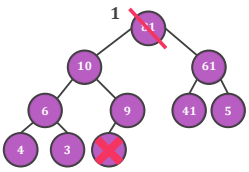


	1	2	3	4	5	6	7	8	9	10
A	81	10	61	6	9	41	5	4	3	1

	1	2	3	4	5	6	7	8	9	10
A	61	10	41	6	9	1	5	4	3	

39

Heap-Extract-Max




```
Heap-Extract-Max(A)
if A.heap_size < 1
    error "heap underflow"
max ← A[1]
A[1] ← A[A.heap_size]
A.heap_size ← A.heap_size − 1
Max-Heapify(A,1)
return max
```

זמן ריצה: $T(n) = O(\log n)$

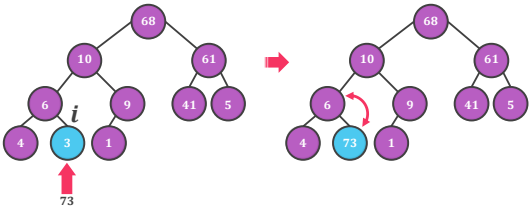
40

פעולות המוגדרות על ערימה

- *Build-Max-Heap* - יוצרת ערימה ממערך קלט בלתי ממוין
- *Heap-Max* - החזרת איבר בעל מפתח מקסימלי בערימת מקסימום
- *Heap-Extract-Max* - מחיקת איבר בעל מפתח מקסימאלי בערימת מקסימום
- *Heap-Increase-Key* - הגדלת מפתח
- *Max-Heap-Insert* - הכנסת איבר חדש
- *Max-Heapify* - שמירה על תכונת הערימה

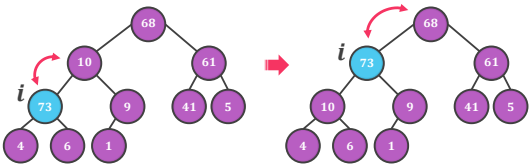
41

Heap-Increase-Key



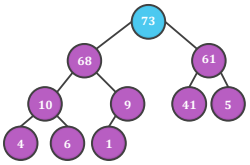
42

Heap-Increase-Key



43

Heap-Increase-Key



44

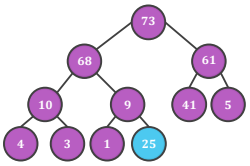
Heap-Increase-Key

```
Heap-Increase-Key(A, i, key)
if key < A[i]
    error "new key is smaller than current key"
A[i] ← key
while i > 1 and A[parent(i)] < A[i]
    exchange A[i] ↔ A[parent(i)]
    i ← parent(i)
```

זמן ריצה: $T(n) = O(\log n)$

45

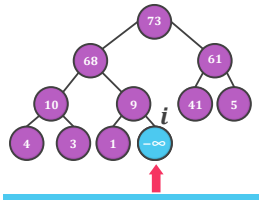
Max-Heap-Insert



$Max\text{-}Heap\text{-}Insert\ (A,25)$

46

Max-Heap-Insert



$Heap\text{-}Increase\text{-}Key(A, i, 25)$

47

Max-Heap-Insert

$\boldsymbol{Max\text{-}Heap\text{-}Insert(A, key)}$

1. $A.heap\_size \leftarrow A.heap\_size + 1$
2. $A[A.heap\_size] \leftarrow -\infty$
3. $Heap\text{-}Increase\text{-}Key(A, A.heap\_size, key)$

זמן ריצה: $T(n) = O(\log n)$

48

שימושים של ערימה

$Heap-Sort$ - מיון ערימה

- מימוש תור עדיפויות

49

מיון ערימה
Heap-Sort

- **נתון:** מערך A לא ממוין

	1	2	3	4	5	6	7	8	9	10
A	10	4	61	3	1	41	5	6	18	19

- **המטרה:** למיין את A

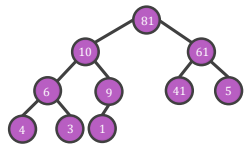
50

מיון ערימה
Heap-Sort

- **רעיון**

	1	2	3	4	5	6	7	8	9	10
A	10	4	61	3	1	41	5	6	81	9

- כל פעם נעביר את האיבר המקסימלי למקום הנכון

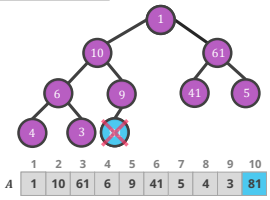


	1	2	3	4	5	6	7	8	9	10
A	81	10	61	6	9	41	5	4	3	1

51

מיון ערימה
Heap-Sort

כל פעם נעביר את האיבר המקסימלי למקום הנכון $Max-Heapify(A,1)$

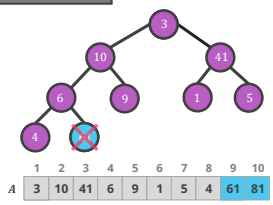


	1	2	3	4	5	6	7	8	9	10
A	1	10	61	6	9	41	5	4	3	81

52

מיון ערימה
Heap-Sort

כל פעם נעביר את האיבר המקסימלי למקום הנכון $Max-Heapify(A,1)$

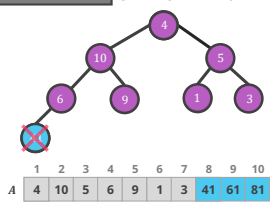


	1	2	3	4	5	6	7	8	9	10
A	3	10	41	6	9	1	5	4	61	81

53

מיון ערימה
Heap-Sort

כל פעם נעביר את האיבר המקסימלי למקום הנכון $Max-Heapify(A,1)$

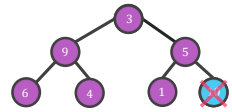


	1	2	3	4	5	6	7	8	9	10
A	4	10	5	6	9	1	3	41	61	81

54

מיון ערימה
Heap-Sort

כל פעם נעביר את האיבר המקסימלי למקום הנכון $Max-Heapify\,(A,1)$

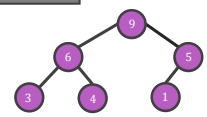


	1	2	3	4	5	6	7	8	9	10
A	3	9	5	6	4	1	10	41	61	81

55

מיון ערימה
Heap-Sort

כל פעם נעביר את האיבר המקסימלי למקום הנכון $Max-Heapify\,(A,1)$



	1	2	3	4	5	6	7	8	9	10
A	9	6	5	3	4	1	10	41	61	81

56

מיון ערימה
Heap-Sort

האיטרציה האחרונה

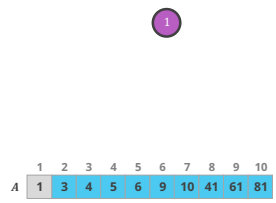
$Max-Heapify\,(A,1)$

	1	2	3	4	5	6	7	8	9	10
A	1	3	4	5	6	9	10	41	61	81

57

מיון ערימה
Heap-Sort

המערך ממוין



58

פסאודו-קוד

$Heap - Sort\ (A)$

1 $Build - Max - Heap(A)$
2 for $i \leftarrow A.length$ downto 2
3 $\quad exchange\ A[1] \leftrightarrow A[i]$
4 $\quad A.heap\_size \leftarrow A.heap\_size - 1$
5 $\quad Max - Heapify(A, 1)$

59

מהו זמן ריצה של מיון ערימה במונחים של O?

תשובה:

$O(n \log n)$

60

פסאודו-קוד

זמן ריצה:

$T(n) = O(n \log n)$

```
Heap − Sort (A)
1  Build − Max − Heap(A)
2  for i ← A.length downto 2
3      exchange A[1] ↔ A[i]
4      A.heap_size ← A.heap_size - 1
5      Max − Heapify(A, 1)
```

61

שימושים של ערימה

- $Heap - Sort$ - מיון ערימה

מימוש תור עדיפויות

62

שימושים של ערימה

- מימוש תור עדיפויות
- תור עדיפויות הוא מבנה נתונים מופשט לטיפול בקבוצה דינאמית של איברים, בה לכל איבר מצורף ערך הנקרא עדיפות (מפתח)

Priority (key)	Satellite Data

- האיבר שנמחק ידוע מראש והוא האיבר בעל העדיפות הגבוהה ביותר

63

64

תור עדיפויות תומך בפעולות

- *MakeEmptyPriorityQueue*
- *EnqueuePriorityQueue*
- *ExtractMax*
- *Maximum*
- *IsEmptyPriorityQueue*
- *IncreaseKey*

65

? במידה ולמימוש תור עדיפויות נשתמש במערך או ברשימה ממוינים לפי עדיפות, מה יהיה זמן הריצה של פעולת *EnqueuePriorityQueue*?

1. $O(\log n)$
2. $O(n^2)$
3. $O(n)$
4. $O(n \log n)$

66

? במידה ולמימוש תור עדיפויות נשתמש במערך או ברשימה **לא ממוינים** לפי עדיפות, מה יהיה זמן הריצה של פעולת ***ExtractMax***?

1. $O(\log n)$
2. $O(n^2)$
3. $O(n)$
4. $O(n \log n)$

67

? במידה ולמימוש תור עדיפויות נשתמש במערך או ברשימה **לא ממוינים** לפי עדיפות, מה יהיה זמן הריצה של פעולת ***ExtractMax***?

1. $O(\log n)$
2. $O(n^2)$
3. $O(n)$
4. $O(n \log n)$

68

תור עדיפויות תומך בפעולות

- *MakeEmptyPriorityQueue*
- *EnqueuePriorityQueue* $O(\log n)$
- *ExtractMax* $O(\log n)$
- *Maximum*
- *IsEmptyPriorityQueue*
- *IncreaseKey*

69

סיכום
ערימה

Heap

70