

## ساختار داده‌ای HeapSort و الگوریتم مرتب‌سازی Heap

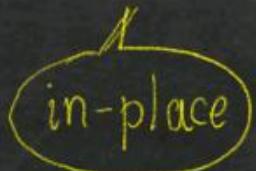
- یک روش برای حل مسئله مرتب‌سازی!

- هر سه زمان اجرای  $\Theta(n \lg n)$  در همه حالت‌ها (every case)

- مرتب‌سازی درجا - بدون نیاز به حافظه ای مسُن از کارایی و روشی -



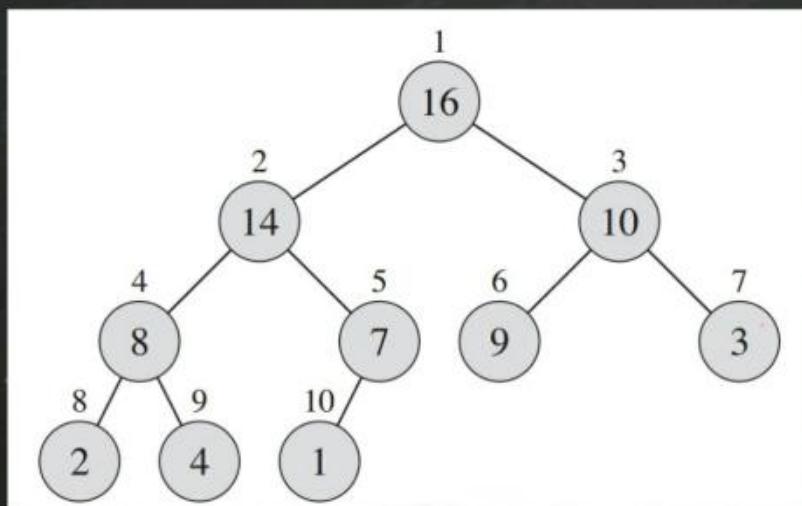
sift



in-place

✓ (نم) heap

## ساختار راده‌ای Heap



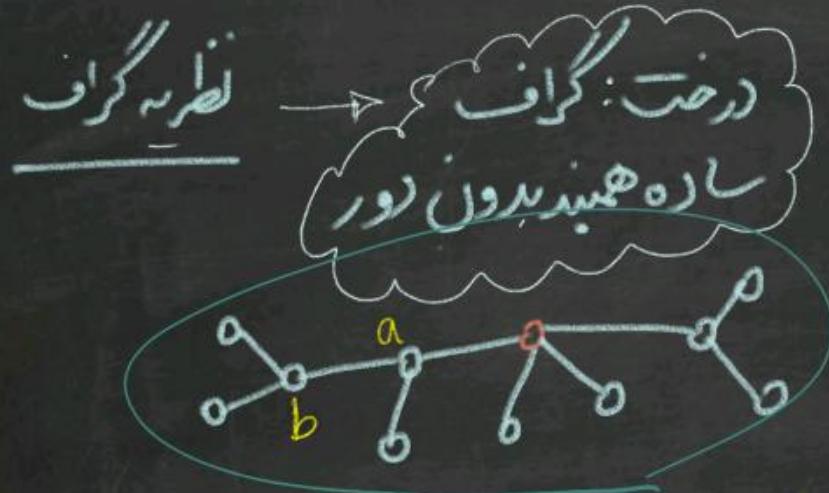
- ساختار Heap

- درخت ریشه دار دودویی

- بالقوه کامل

- قاعده حاکم بر معادله در Heap

- معادله طبقه مسأله با هرگره از معادله مسأله با هرگره والدنس لوحیتر است



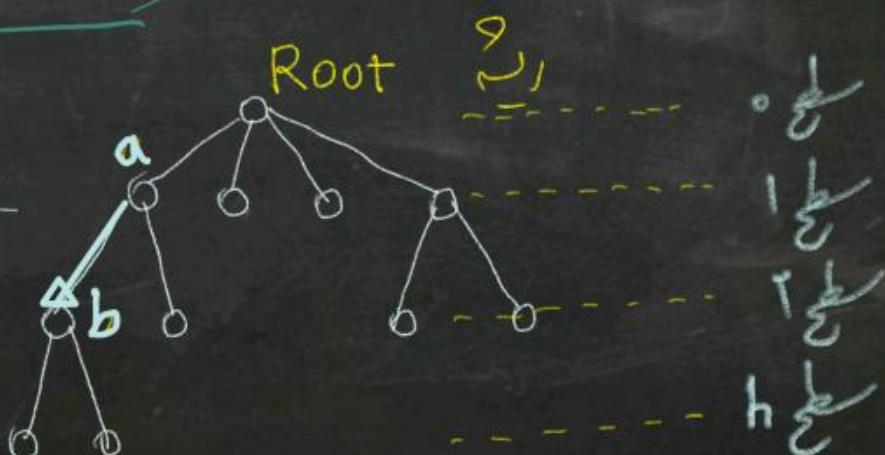
ساختار Heap

- ② درخت ریشه دار دودویی
- ③ بالغه طامل

① Rooted tree

درخت ریشه دار

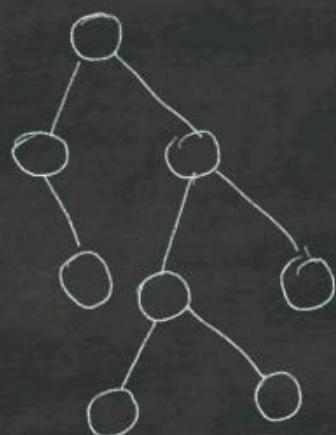
رابطه: a والد b و b فرزند a می باشد



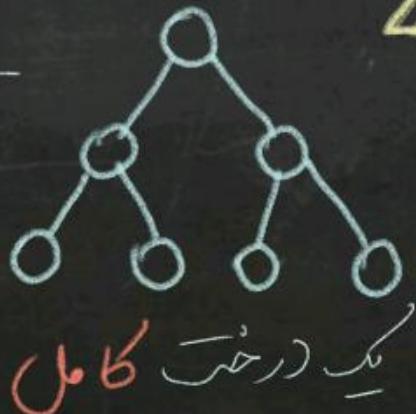
h: ارتفاع درخت

ساختار  
Heap

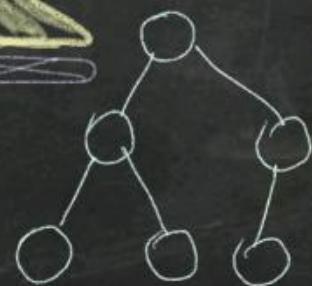
درخت ریشه دار دودویی  
بالقوه کامل



درخت بالقوه کامل

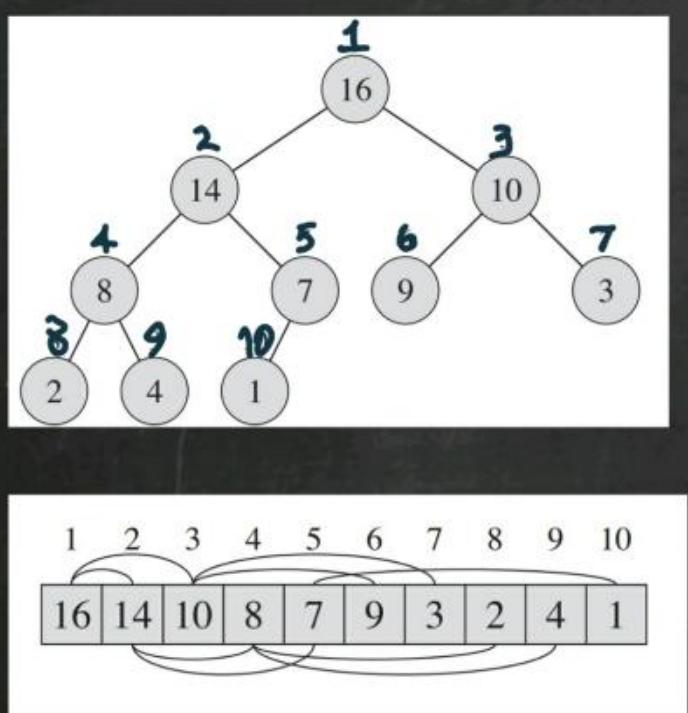


یک درخت کامل



یک درخت بالقوه کامل

# ویژه سازی سختار Heap کو سط آرایه

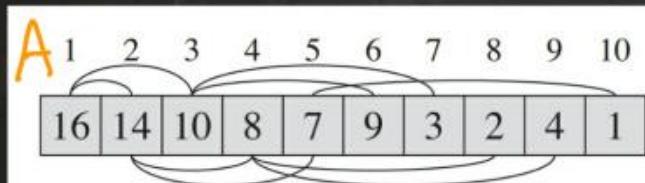
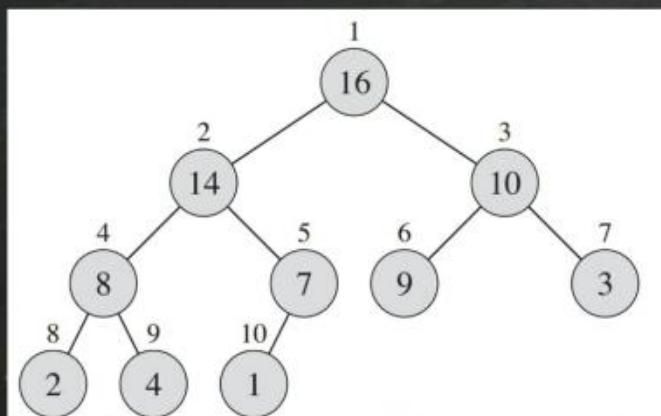


سختار دودویی بالعوہ کامل  
این امکان را فراهم می کند که  
Heap را با سادگی بتوانیم  
در سط آرایه نگذاری کنیم

۴ درواز سختار درختی

بطور خمنی معنا وسایعی کند - در عمل صد هزار در سط آرایه است.

# Heap, with solutions



PARENT( $i$ )

1   **return**  $\lfloor i/2 \rfloor$

LEFT( $i$ )

1   **return**  $2i$

RIGHT( $i$ )

1   **return**  $2i + 1$

فاسد هام بر معادل کنید

$i \in \{2, \dots, n\}$  درم اگر به ارای هر  $\frac{A[i]}{A[\lfloor i/2 \rfloor]}$  max-heap  $\frac{A[i]}{A[\lfloor i/2 \rfloor]}$ -

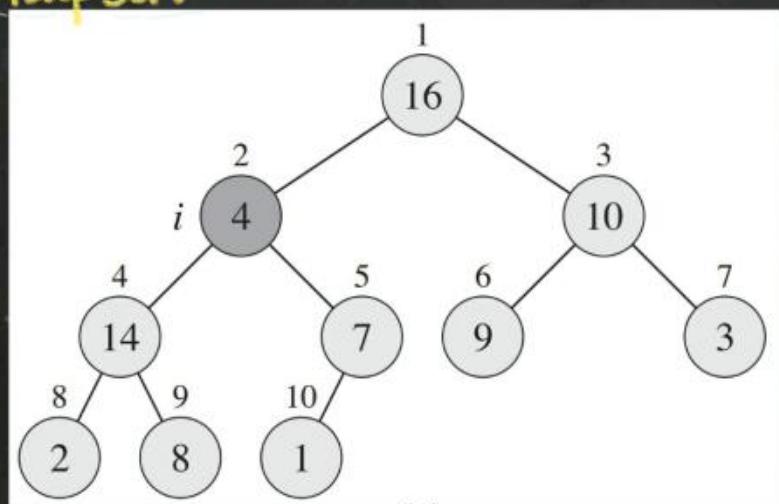
$A[i] \leq A[\lfloor i/2 \rfloor]$

$i \in \{2, \dots, n\}$  درم اگر به ارای  $\frac{A[i]}{A[\lfloor i/2 \rfloor]}$  min-heap  $\frac{A[i]}{A[\lfloor i/2 \rfloor]}$ -

$A[i] \geq A[\lfloor i/2 \rfloor]$

\* بعد از دادی درین سه نظر از heap

Heapify , sift-down  
Bubble-up  
Build Heap  
HeapSort



خط وری

فرض کنیم هر بار که از  
نهاده هایی که در هر دو فرزند  
نهاده هایی هستند، هر دویکی از  
نهاده های خود را بزرگتر می باشد

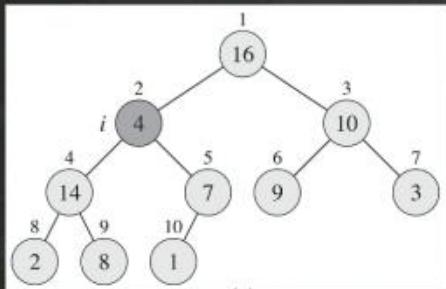
نهاده هایی که در هر دو فرزند  
نهاده هایی هستند، هر دویکی از  
نهاده های خود را بزرگتر می باشد

لذا هر دویکی از

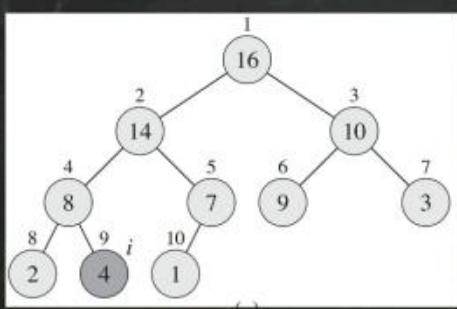
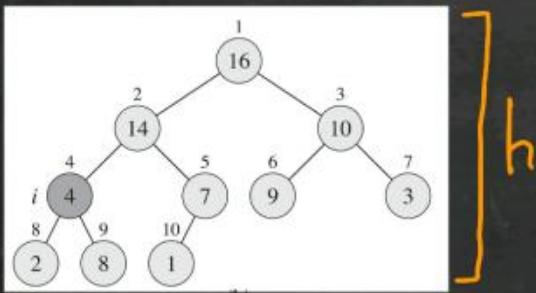
نهاده هایی که در هر دو فرزند  
نهاده هایی هستند، هر دویکی از  
نهاده های خود را بزرگتر می باشد

است نهاده هایی که در هر دو فرزند  
نهاده هایی هستند، هر دویکی از  
نهاده های خود را بزرگتر می باشد

با جایگزینی هر دویکی از  
نهاده های خود را بزرگتر می باشد



Sift-down      Heap  $\frac{1}{2}$  to  $\frac{1}{2}$

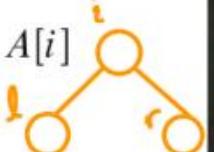


MAX-HEAPIFY( $A, i$ )

```

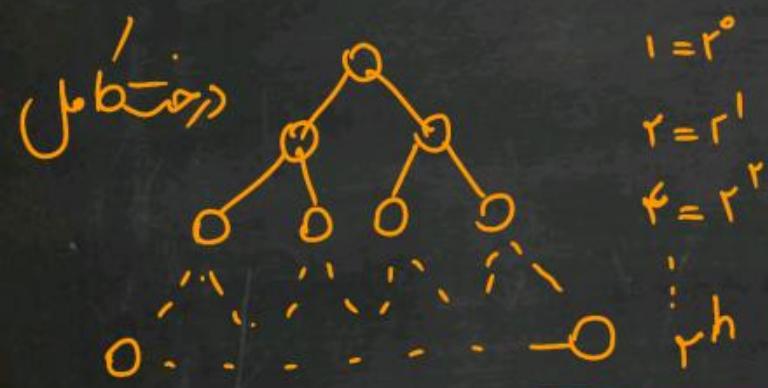
1   l = LEFT( $i$ )
2   r = RIGHT( $i$ )
3   if  $l \leq A.\text{heap-size}$  and  $A[l] > A[i]$ 
4       largest =  $l$ 
5   else largest =  $i$ 
6   if  $r \leq A.\text{heap-size}$  and  $A[r] > A[\text{largest}]$ 
7       largest =  $r$ 
8   if  $\text{largest} \neq i$ 
9       exchange  $A[i]$  with  $A[\text{largest}]$ 
10      MAX-HEAPIFY( $A, \text{largest}$ )
    
```

$O(h) = O(\lg n)$



# رایط، ارتفاع و لعدا کردنی درخت در ساختار Heap

①



$$\Rightarrow r^{h+1} = n + 1 \Rightarrow h+1 = \lg(n+1)$$

(درخت بالقوه مل بکثرین بعرا کرده)

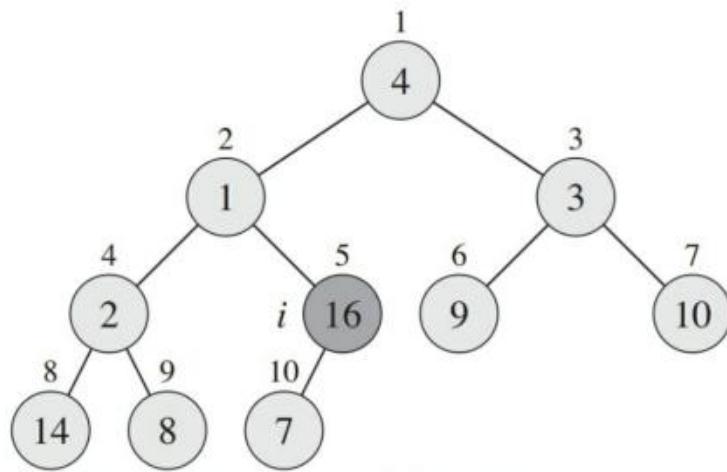
②



$$n = r^h \Rightarrow h = \lg n$$

$$\left. \begin{array}{l} ① \\ + \\ ② \end{array} \right\} \Rightarrow h = \lfloor \lg n \rfloor$$

$A$  [ 4 | 1 | 3 | 2 | 16 | 9 | 10 | 14 | 8 | 7 ]



BUILD-MAX-HEAP( $A$ )

```

1    $A.\text{heap-size} = A.\text{length}$ 
2   for  $i = \lfloor A.\text{length}/2 \rfloor$  downto 1
3     MAX-HEAPIFY( $A, i$ )  $\rightarrow O(\lg n)$ 
```

Heap نهائی

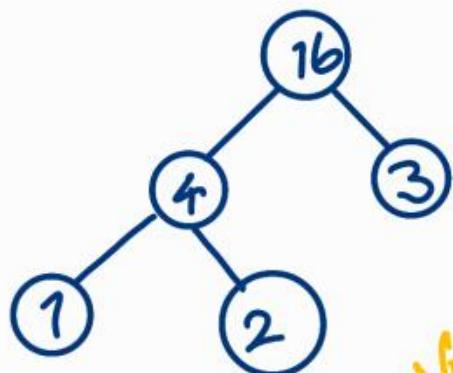
الوثر ایضاً نهائی  
نهائی بجهة نهائی  
نهائی نهائی  
نهائی بجهة نهائی  
نهائی بجهة نهائی

Build Heap  $\leftarrow$

$O(n \lg n)$

$O(n)$

A [ 4 | 1 | 3 | 2 | 16 | 9 | 10 | 14 | 8 | 7 ]



BuildHeap( $A$ )

for  $i = 2$  to  $A.length$

*Version 2* Bubble-up( $A, i$ )

for  $i = \frac{n}{2}$  down to 1

sift-down( $A, i$ )

*Version 1*

example

A [ 4 | 1 | 3 | 2 | 16 | 9 | 10 | 14 | 8 | 7 ]

Trace two implementation of BuildHeap  
on the example by yourself

Build Heap  $\approx \sum_{j=1}^n 2^{h-j} \cdot j = \Theta(n)$

# مرتب سازی HeapSort

HEAPSORT( $A$ )

```

1  BUILD-MAX-HEAP( $A$ )
2  for  $i = A.length$  downto 2
3    exchange  $A[1]$  with  $A[i]$   $\rightarrow O(1)$ 
4     $A.heap-size = A.heap-size - 1$ 
5    MAX-HEAPIFY( $A, 1$ )  $\rightarrow O(1)$ 

```

$\nabla O(lgn) \quad \Theta(nlgn)$

آرایه و دردی

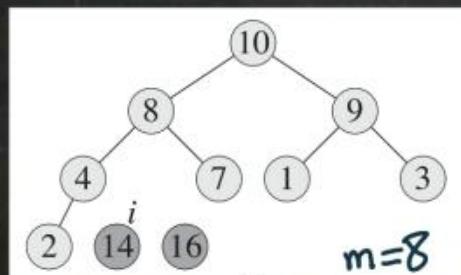
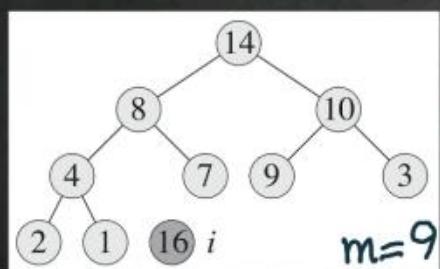
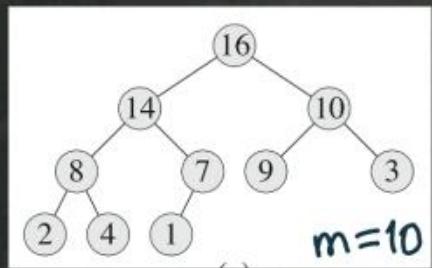
Heap

آرایه  
مرتب شده

- از منطق الگوریتم مرتب سازی SelectSort استفاده می کند:  
 نزولی رین عصر را پیدا کن و به آن سفل کن

- مرتب سازی heap: پیام ردن  
 صفر را کردم تا ب حذف  $O(1)$  دارد.

# مرتب سازی هیپ



*m: heap\_size*

HEAPSORT( $A$ )  $\underline{O(n \lg n)}$

- 1 BUILD-MAX-HEAP( $A$ )  $\Theta(n)$
- 2 **for**  $i = A.length$  **downto** 2  $\underline{\text{برگشت}}(n-1)$
- 3  $O(1)$  exchange  $A[1]$  with  $A[i]$
- 4  $O(1)$   $A.heap-size = A.heap-size - 1$
- 5 MAX-HEAPIFY( $A$ , 1)

