Full Version

Week 8
Sorting, HashTables

LM Data Structures, Algorithms, and Databases (34141)

Dr Ahmad Ibrahim
a.ibrahim@bham.ac.uk
March 04, 2024

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Contents based on lecture notes from Uday, Mirco, Martin, Alan

Topics by Week

Week	Date	Topic
1	15 Jan	Searching algorithms
2	22 Jan	Binary Search Tree
3	29 Jan	Balancing Trees – AVL Tree
4	5 Feb	Databases – Conceptual Design
5	14 Feb	Databases – Logical Design & Relational Algebra
6	19 Feb	Consolidation Week
7	26 Feb	Complexity analysis, Stacks, Queues, Heaps
8	4 Mar	Sorting Algorithms, Hash tables
9	11 Mar	Graph Algorithms
10	18 Mar	Databases – Normalization
		Easter break and Eid break
11	22 Apr	Databases – Concurrency
12	29 Apr	Revision Week

Timetable & Office hours

Day	Time	Event	Location
Monday	4:00-5:00pm	Online support session*	Online*
Tuesday	4:00-5:00pm	Office hour 1 (by appointment)*	Online*
Wednesday	-	-	-
Thursday	4:00-5:00pm	Office hour 2 (by appointment)*	Online*
			Auditorium
			Auditorium

Ramadan Timetable
March 17th and March 24th 2024
Sunday 1:00-4:00pm

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

***Zoom link:** https://bham-ac-uk.zoom.us/j/81310444523?pwd=T01tZlZGdmdUL2lkeHZsVFpjcWxUUT09

Assessments

Assessments (Test 1, Test 2, Test 3): **20%**

Exam: **80%**

Week 10

Test 3

Not available until 20 Mar at 16:00 | Due 21 Mar at 16:00 | -/20 pts

Important Note (Ramadan)

4:00pm UAE Time

(12:00 noon UK Time)

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

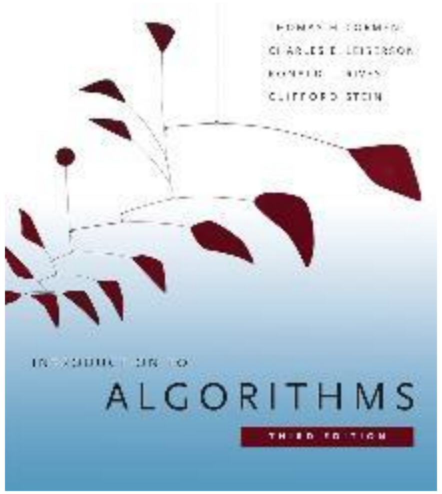
Weekly Reading

Data Structures and Algorithm Analysis by Clifford A. Shaffer (3rd Ed)

Section 7 Sorting
Section 9.4 Hashing
https://people.cs.vt.edu/~shaffer/Book/JAVA3elatest.pdf

Introduction to Algorithms by Cormen (4th Ed)

Section 2.1 Insertion sort
Section 2.3.1 The divide-and-conquer method
Chapter 6 Heapsort
Chapter 7 Quicksort
Chapter 11 HashTables
https://ebookcentral.proquest.com/lib/bham/detail.action?docID=6925615



UNIVERSITY OF BIRMINGHAM | DUBAI دبـي

Recap

Last Week

- Complexity analysis
- Stacks
- Queues
- Heaps
- Sorting Algorithm
  - Insertion Sort
  - Heapsort

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Recap

This Week

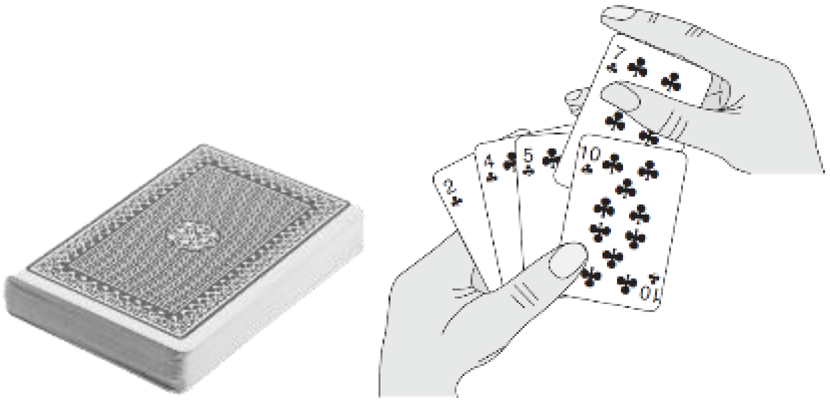
Sorting Algorithm

- Insertion Sort
- Heapsort
- Merge sort (Divide and conquer)
- Quick Sort (Divide and conquer)

HashTables

Insertion sort

A simple, intuitive sorting algorithm.



Iteratively removes one element from the input data → Finds location it belongs in the sorted list → Insert it there until no input elements remain

Main steps in Insertion sort.

Recap

Insertion Sort

Array (A)

index i

1	2	5	7	3	8	6	9	4	10

Sorted

Unsorted

					8	6	9	4	10

Sorted

index i

1	2	3	5	7	8	6	9	4	10

Sorted

Unsorted

Recap

Insertion Sort (Complexity)

```
Insertion-Sort (A):
   for i=2 to n
        Key=A[i]
        j=i-1
        while A[j]>Key and j>=0
               A[j+1] = A[j]
               j=j-1
        A[j+1]= Key
```

O(n)

O(n)

$O(n^2)$

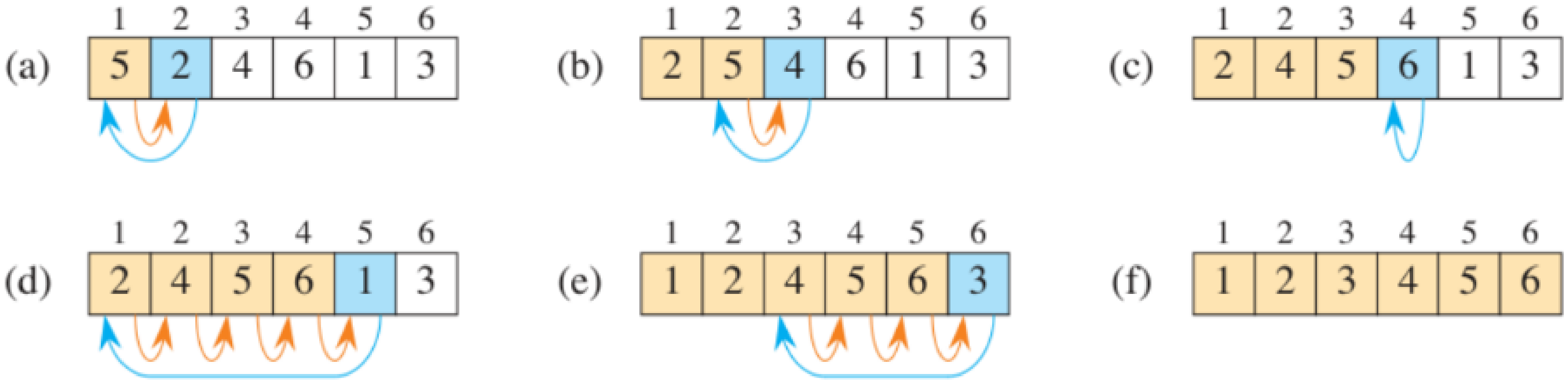
Insertion Sort

Show step by step working of Insertion sort algorithm for the given array A (5, 2, 4, 6, 1, 3)

Insertion Sort

Show step by step working of Insertion sort algorithm for the given array A (5, 2, 4, 6, 1, 3)

Answer



This Week

Sorting Algorithm

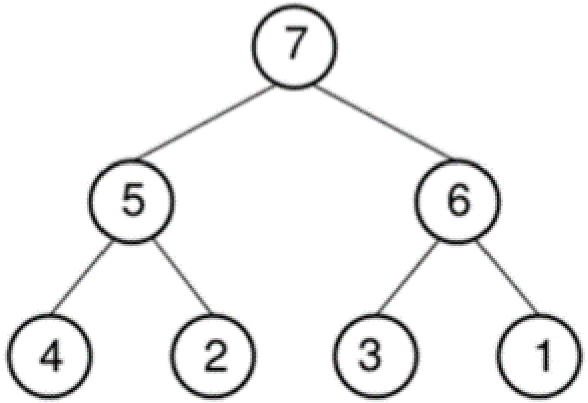
- Insertion Sort
- Heapsort
- Merge sort (Divide and conquer)
- Quick Sort (Divide and conquer)

HashTables

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

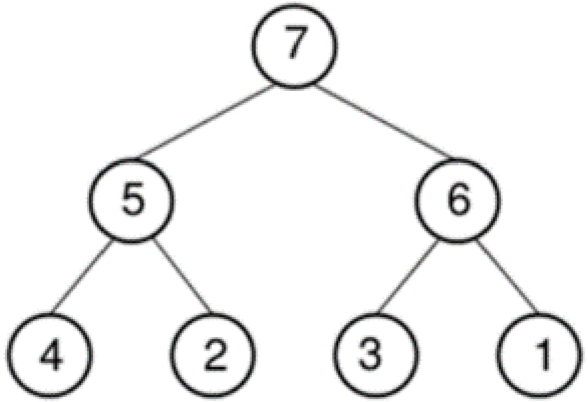
Errata (last week Exercise Sheet 07)

Q6. Show the heap that results from deleting the maximum value from the max-heap of following Figure.

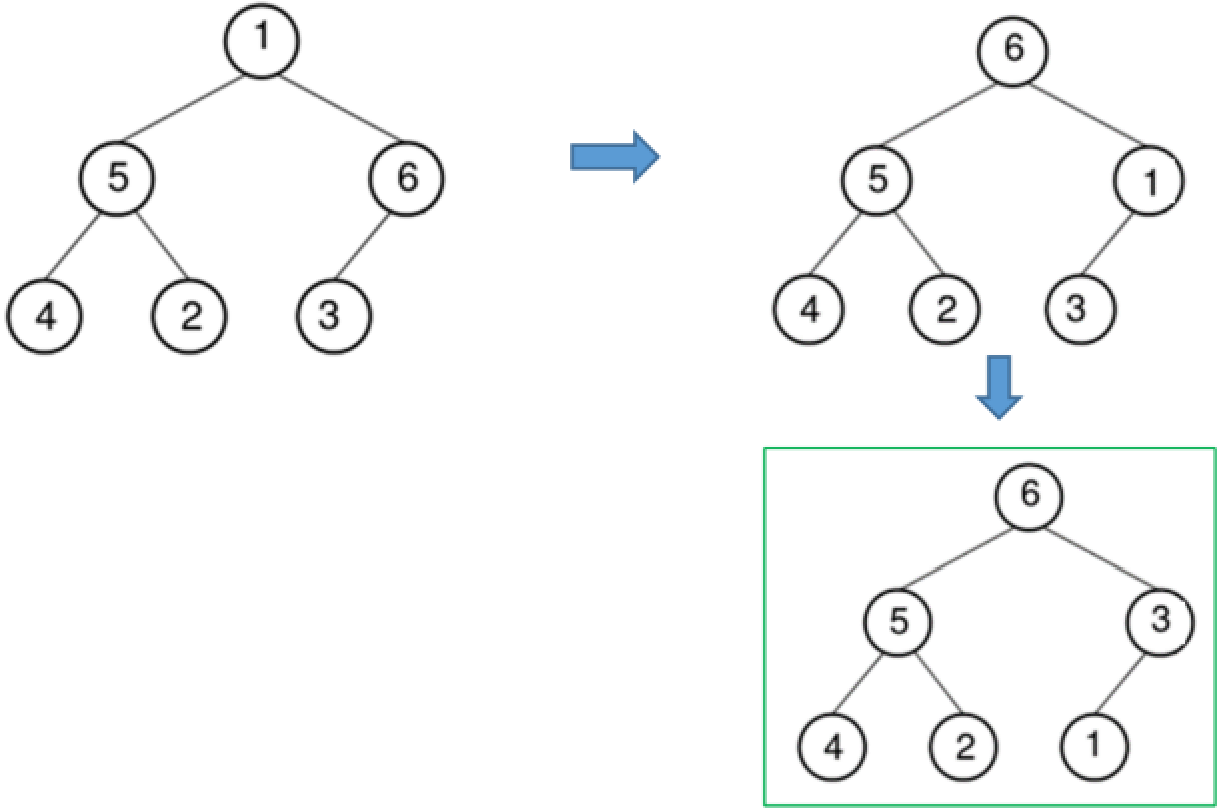


Errata (last week Exercise Sheet 07)

Q6. Show the heap that results from deleting the maximum value from the max-heap of following Figure.



Answer: We can move the element in the last position in the heap (the current last element in the array) to the root position and then order the heap.



Recap

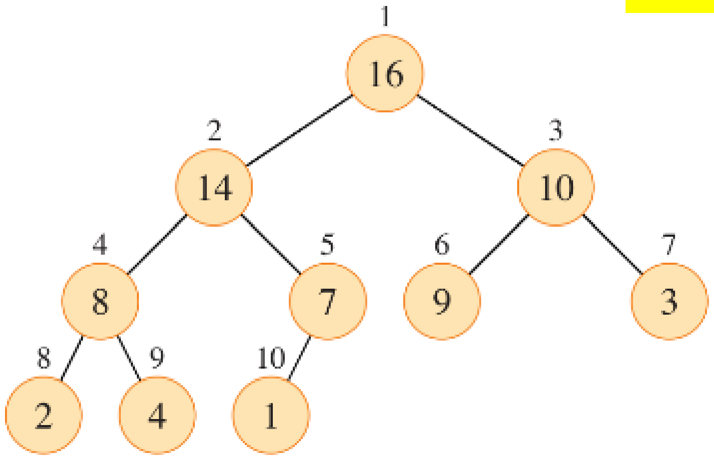
Heapsort

For a Max-Heap, the largest item is stored at the root node.

1.Swap: Remove the root element and put at the end of the array (nth position). Put the last item of the tree (heap) at the vacant place.

2.Remove: Reduce the size of the heap by 1.

3.Heapify: Heapify the root element again so that we have the highest element at root.




```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

https://www.programiz.com/dsa/heap-sort

Recap

1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

heap-size = 10

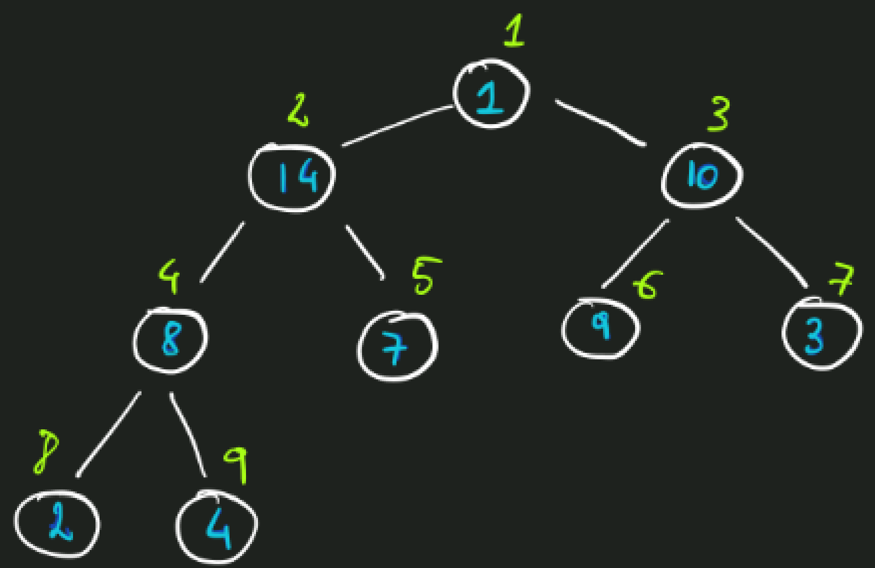
i

Swap

1	2	3	4	5	6	7	8	9	10
1	14	10	8	7	9	3	2	4	16

heap-size = 9

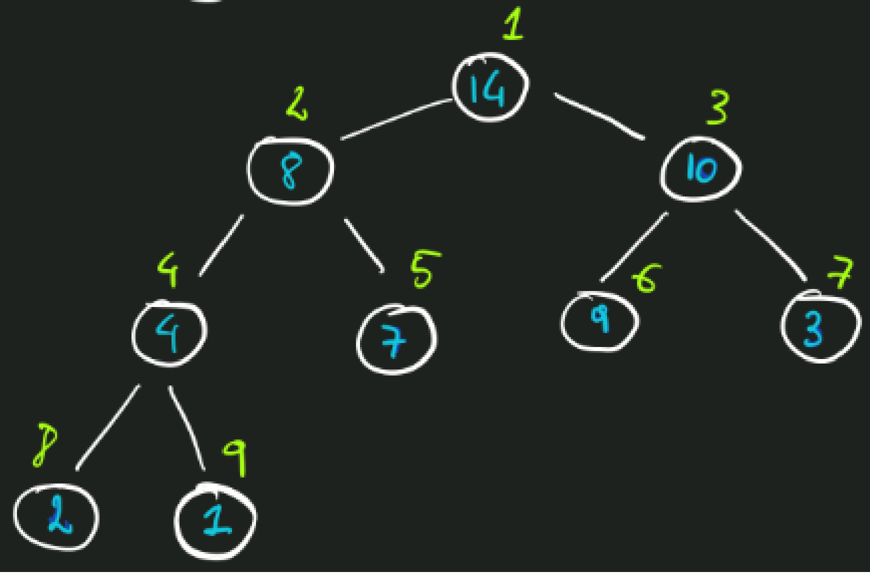
i



Max-Heapify(A, 1)

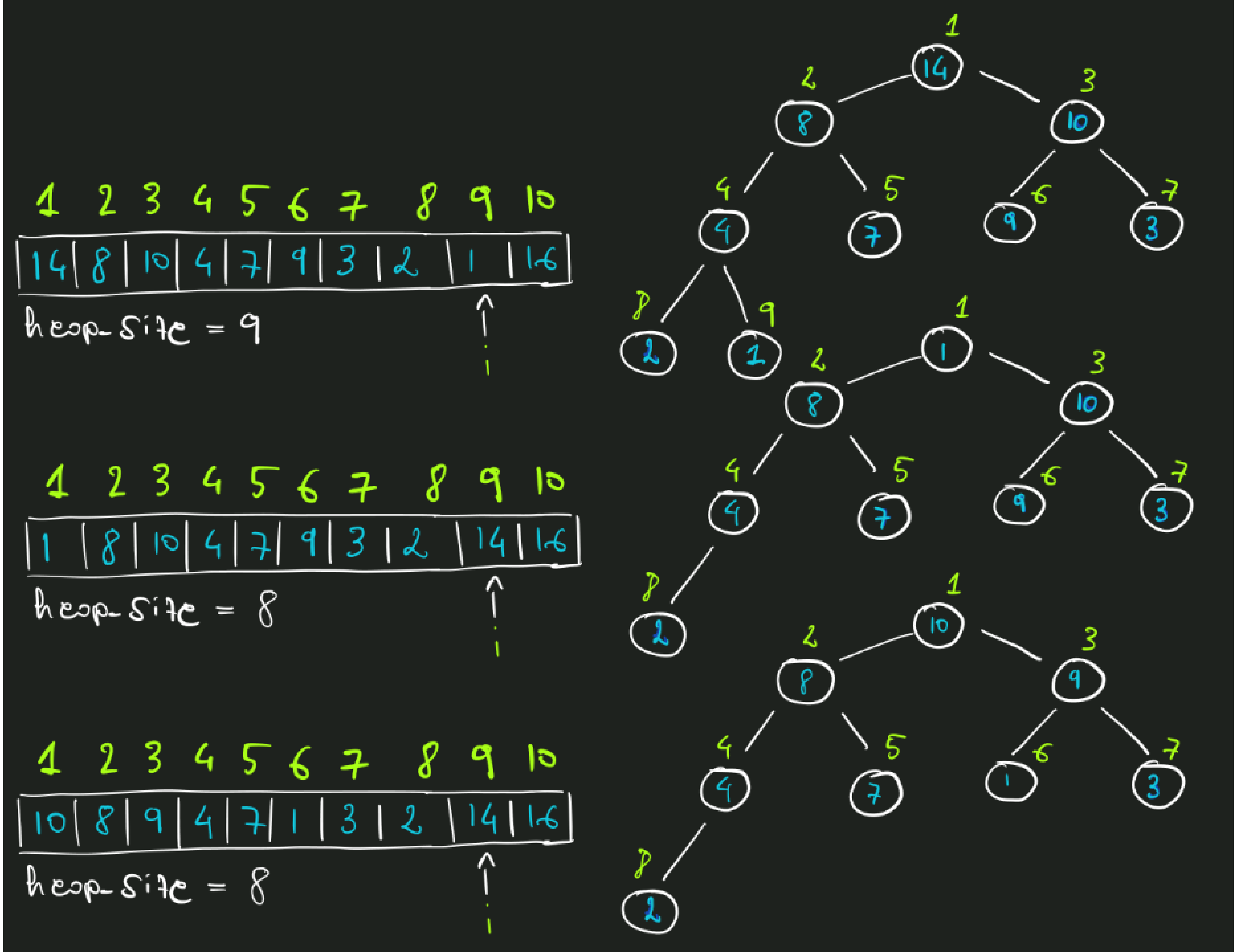
1	2	3	4	5	6	7	8	9	10
14	8	10	4	7	9	3	2	1	16

i

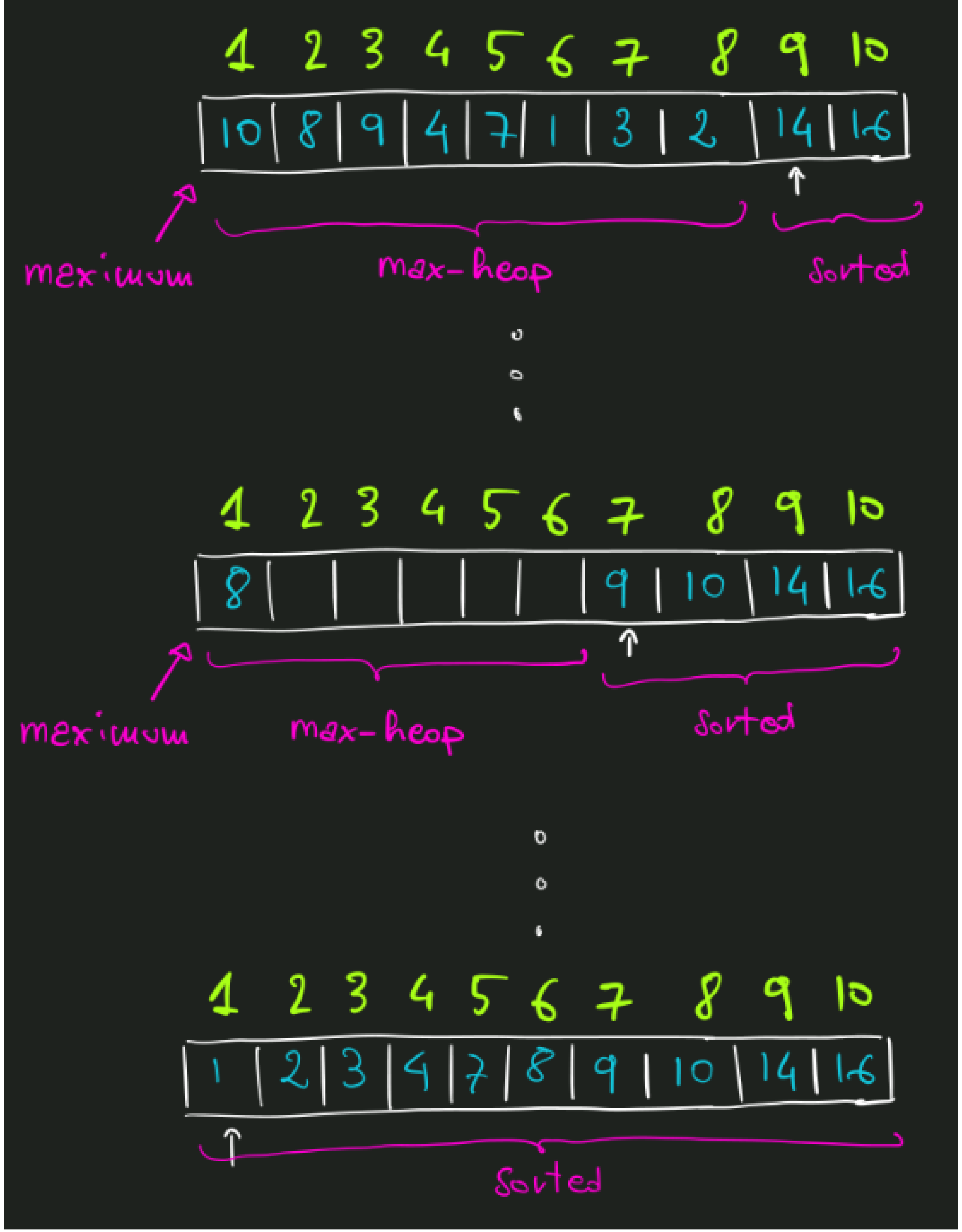


dsadb24_intro-to-sorting.pdf

Recap


1 2 3 4 5 6 7 8 9 10
14 | 8 | 10 | 4 | 7 | 9 | 3 | 2 | 1 | 16
heap-size = 9
1 2 3 4 5 6 7 8 9 10
1 | 8 | 10 | 4 | 7 | 9 | 3 | 2 | 14 | 16
heap-size = 8
1 2 3 4 5 6 7 8 9 10
10 | 8 | 9 | 4 | 7 | 1 | 3 | 2 | 14 | 16
heap-size = 8

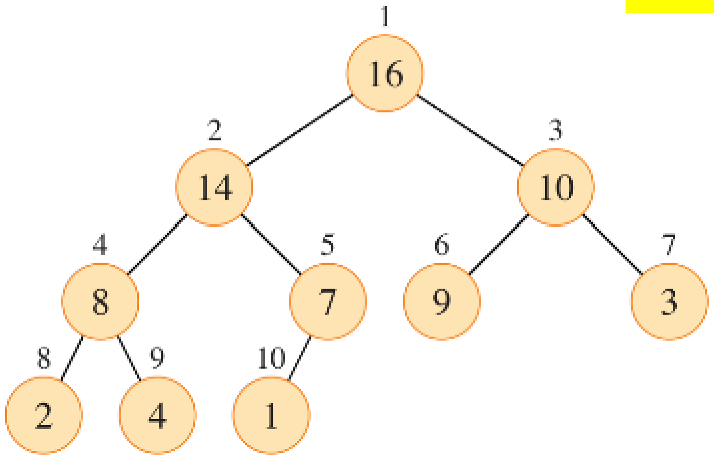
Recap



1 2 3 4 5 6 7 8 9 10
10 8 9 4 7 1 3 2 14 16
maximum
max-heap
sorted
1 2 3 4 5 6 7 8 9 10
8 9 10 14 16
maximum
max-heap
sorted
1 2 3 4 5 6 7 8 9 10
1 2 3 4 7 8 9 10 14 16
sorted

Recap

Heapsort (Complexity)




```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)                      → O (n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)                     → O (log n)
```

O (n log n)

Exercise Heapsort

Consider the following array

Index	1	2	3	4
Value	9	7	5	11

Carry out the steps of sorting in ascending order using HeapSort.

Exercise Heapsort

Index	1	2	3	4
Value	9	7	5	11

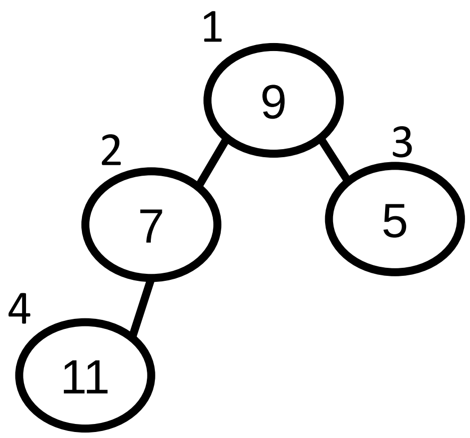

```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

Exercise Heapsort

Index	1	2	3	4
Value	9	7	5	11


```
BUILD-MAX-HEAP(A, n)
  A.heap-size = n
  for i = ⌊n/2⌋ downto 1
      MAX-HEAPIFY(A, i)
```

```
HEAPSORT(A, n)
  BUILD-MAX-HEAP(A, n)
  for i = n downto 2
      exchange A[1] with A[i]
      A.heap-size = A.heap-size − 1
      MAX-HEAPIFY(A, 1)
```



Exercise Heapsort

Index	1	2	3	4
Value	9	7	5	11

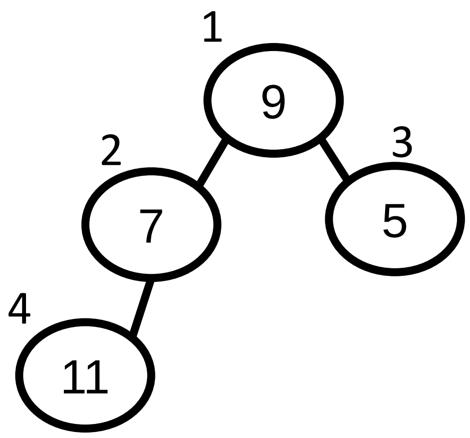
BUILD-MAX-HEAP(A, n)

$A.heap\text{-}size = n$
for $i = \lfloor n/2 \rfloor$ **downto** 1
  MAX-HEAPIFY(A, i)

```
n =4
i=4/2 = 2
for i = 2 downto 1
    Max-Heapify (A, i)
```

HEAPSORT(A, n)

BUILD-MAX-HEAP(A, n)
for $i = n$ **downto** 2
  exchange $A[1]$ with $A[i]$
  $A.heap\text{-}size = A.heap\text{-}size - 1$
  MAX-HEAPIFY$(A, 1)$



Exercise Heapsort

Index	1	2	3	4
Value	9	7	5	11

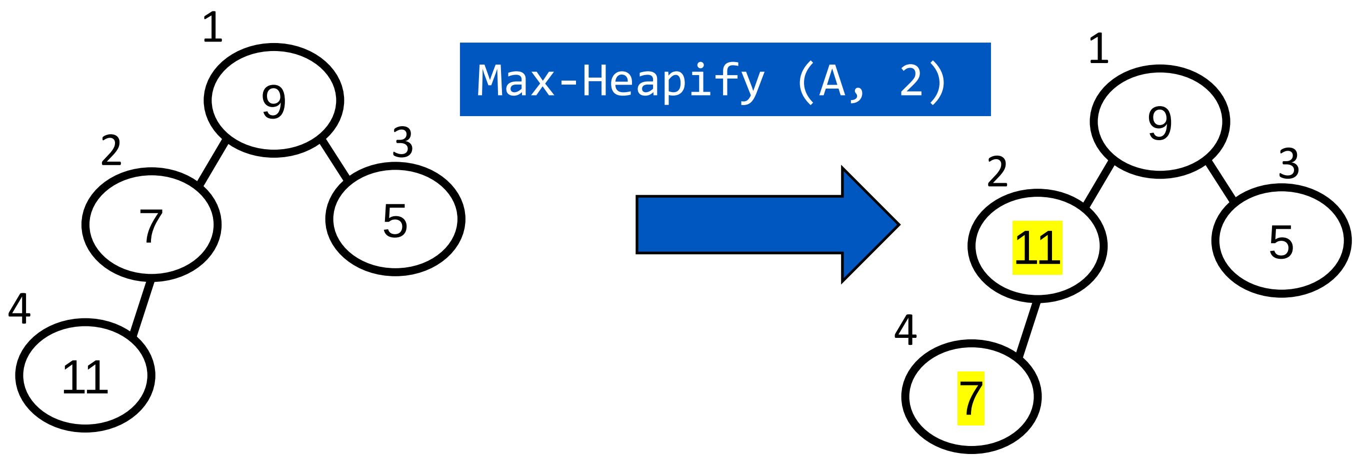
BUILD-MAX-HEAP(A, n)

$A.heap\text{-}size = n$
for $i = \lfloor n/2 \rfloor$ **downto** 1
    MAX-HEAPIFY(A, i)

```
n =4
i=4/2 = 2
for i = 2 downto 1
    Max-Heapify (A, i)
```

HEAPSORT(A, n)

BUILD-MAX-HEAP(A, n)
for $i = n$ **downto** 2
    exchange $A[1]$ with $A[i]$
    $A.heap\text{-}size = A.heap\text{-}size - 1$
    MAX-HEAPIFY$(A, 1)$



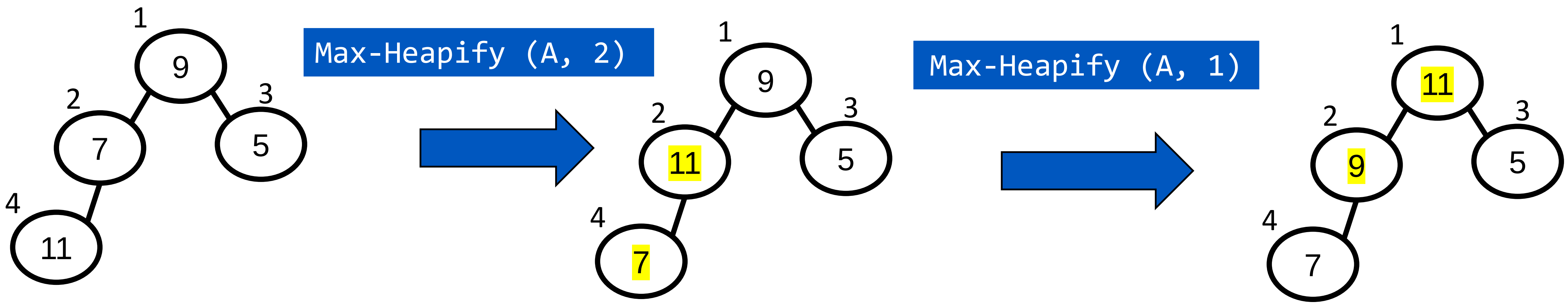
Exercise Heapsort

Index	1	2	3	4
Value	9	7	5	11

```
BUILD-MAX-HEAP(A, n)
    A.heap-size = n
    for i = ⌊n/2⌋ downto 1
        MAX-HEAPIFY(A, i)
```

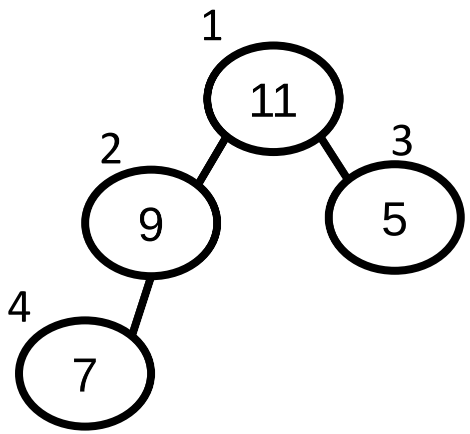
```
n =4
i=4/2 = 2
for i = 2 downto 1
       Max-Heapify (A, i)
```

```
HEAPSORT(A, n)
    BUILD-MAX-HEAP(A, n)
    for i = n downto 2
        exchange A[1] with A[i]
        A.heap-size = A.heap-size − 1
        MAX-HEAPIFY(A, 1)
```



Exercise Heapsort

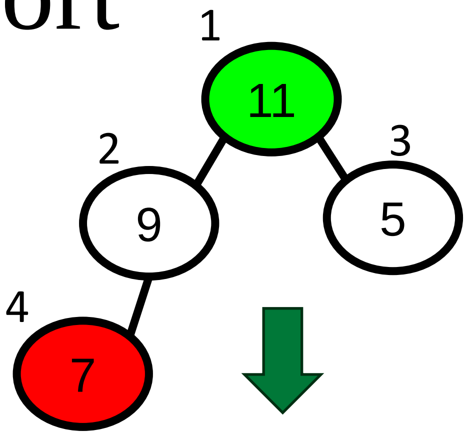

```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```



Array representation

Index	1	2	3	4
Value	11	9	5	7

Exercise Heapsort

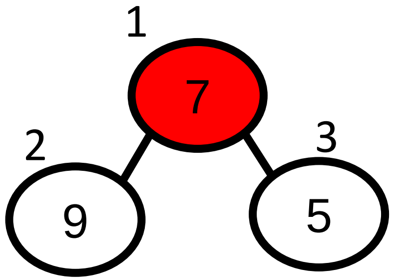



```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

1. Swap the first and last elements: [7, 9, 5, 11]

2. Reduce the heap size: [7, 9, 5]

3. Heapify to maintain the heap property.

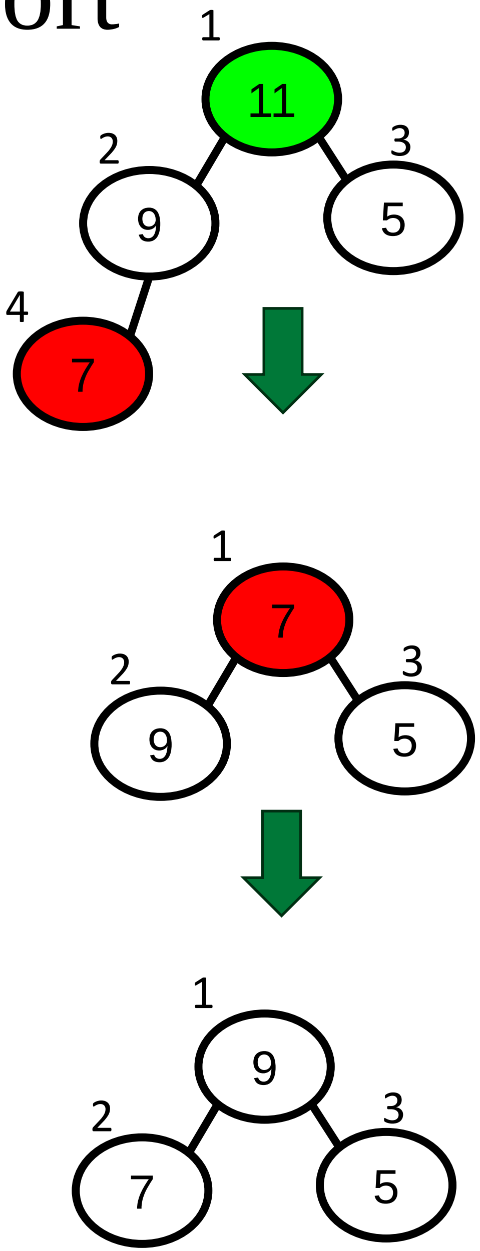


Index	**1**	**2**	**3**	**4**
Value	7	9	5	11

Exercise Heapsort

```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

1. Swap the first and last elements: [7, 9, 5, 11]
2. Reduce the heap size: [7, 9, 5]
3. Heapify to maintain the heap property.



Index	1	2	3	4
Value	7	9	5	11

After Heapify (A, 1)

Index	1	2	3	4
Value	9	7	5	11

heap: [9, 7, 5]

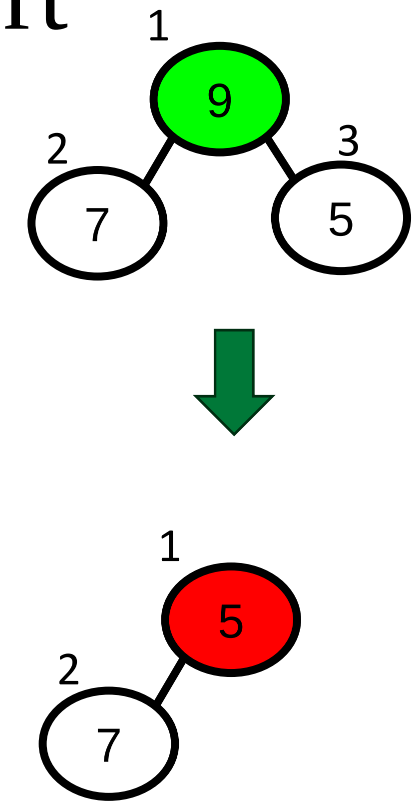
Exercise Heapsort

Repeat the process for the reduced heap.

1. Swap the first and last elements: [5, 7, 9, 11]

2. Reduce the heap size: [5, 7, 9]

3. Heapify to maintain the heap property.




```
HEAPSORT(A, n)
  BUILD-MAX-HEAP(A, n)
  for i = n downto 2
      exchange A[1] with A[i]
      A.heap-size = A.heap-size − 1
      MAX-HEAPIFY(A, 1)
```

Index	1	2	3	4
Value	5	7	9	11

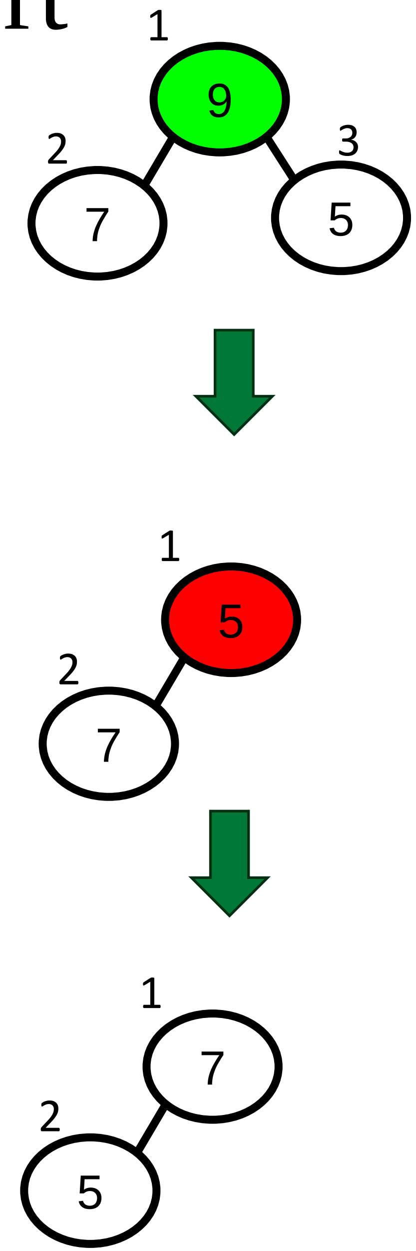
Exercise Heapsort

Repeat the process for the reduced heap.

1. Swap the first and last elements: [5, 7, 9, 11]

2. Reduce the heap size: [5, 7, 9]

3. Heapify to maintain the heap property.




```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size - 1
    MAX-HEAPIFY(A, 1)
```

Index	1	2	3	4
Value	5	7	9	11

After Heapify (A, 1)

Index	1	2	3	4
Value	7	5	9	11

heap: [7, 5]

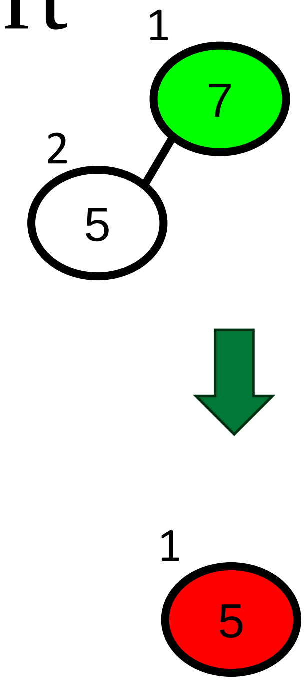
Exercise Heapsort

Repeat the process for the reduced heap.

1. Swap the first and last elements: [5, 7]

2. Reduce the heap size: [5]

3. Heapify to maintain the heap property.




```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size − 1
    MAX-HEAPIFY(A, 1)
```

Index	1	2	3	4
Value	5	7	9	11

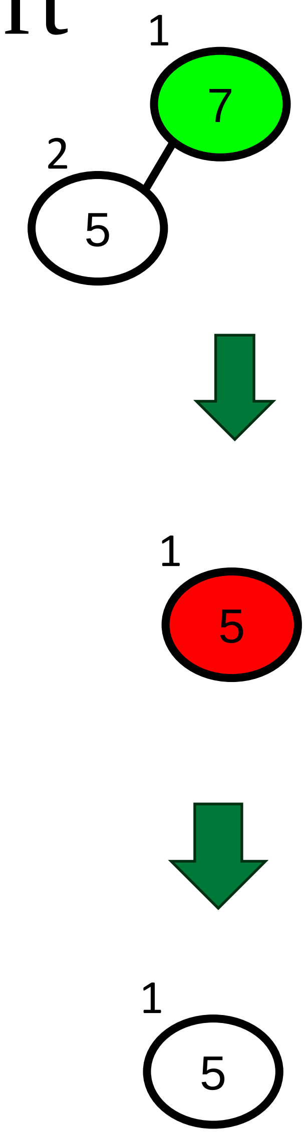
Exercise Heapsort

Repeat the process for the reduced heap.

1. Swap the first and last elements: [5, 7]

2. Reduce the heap size: [5]

3. Heapify to maintain the heap property.




```
HEAPSORT(A, n)
BUILD-MAX-HEAP(A, n)
for i = n downto 2
    exchange A[1] with A[i]
    A.heap-size = A.heap-size - 1
    MAX-HEAPIFY(A, 1)
```

Index	1	2	3	4
Value	5	7	9	11

Index	1	2	3	4
Value	5	7	9	11

This Week

Sorting Algorithm

- Insertion Sort
- Heapsort
- Merge sort (Divide and conquer)
- Quick Sort (Divide and conquer)

HashTables

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Merge sort (Divide and conquer)

Merge Sort (Divide & Conquer)

(Slides from Alan P. Sexton)

Merge-sort.pdf

Exercise Merge Sort

Consider the following array

9	7	5	11

Carry out the steps of sorting in ascending order using Merge Sort.

Exercise Merge Sort

9	7	5	11

9	7

5	11

1. Splitting Phase:

Split the array into two halves:
Left half: [9, 7] Right half: [5, 11]

Exercise Merge Sort

9	7	5	11

9	7

5	11

9

7

5

11

1. Splitting Phase:

Split the array into two halves:
Left half: [9, 7]   Right half: [5, 11]

2. Recursive Sorting:
For the left half, Split further
Left half: [9]   Right half: [7]

Recursively sort each half. Since each half has only one element, they are considered sorted.

Similarly for the right half:
Left half: [5]   Right half: [11]

Exercise Merge Sort

7	9

5	11

3. Merging Phase:
Compare the first elements of the left and right halves:

[9] and [7]: [7, 9]
[5] and [11]: [5, 11]

- We start with two pointers, one for each half: one pointer for the left half (starting at 7) and one pointer for the right half (starting at 5).
- We compare the elements that the pointers are currently pointing to.
- We take the smaller element and add it to the new merged array.

Exercise Merge Sort

7	9

5	11

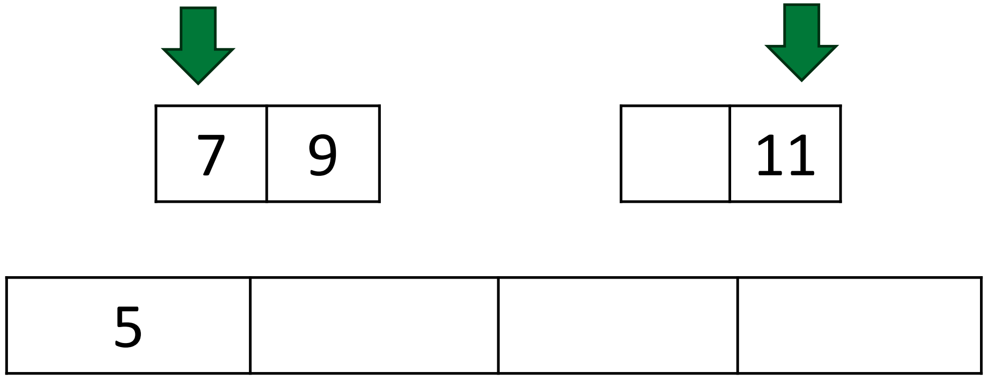
Exercise Merge Sort

7	9

5	11

5			

Exercise Merge Sort



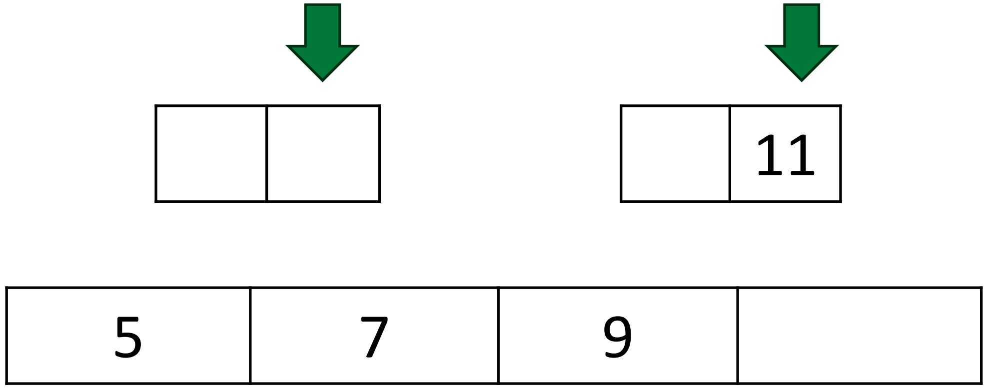
Exercise Merge Sort

	9

	11

5	7		

Exercise Merge Sort



Exercise Merge Sort

5	7	9	11

Done!

This Week

Sorting Algorithm

- Insertion Sort
- Heapsort
- Merge sort (Divide and conquer)
- Quick Sort (Divide and conquer)

HashTables

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Quick sort (Divide and conquer)

Quick Sort (Divide & Conquer)

(Slides from Alan P. Sexton)

Quick-sort.pdf

Exercise Quick Sort

Consider the following array

9	7	5	11

Carry out the steps of sorting in ascending order using quick Sort.

Exercise Quick Sort

9	7	5	11

Choose the leftmost entry 9 as the pivot.

Exercise Quick Sort

5 min

9	7	5	11

Choose the leftmost entry 9 as the pivot.

Lesser elements: [7,5]

Pivot: 9

Greater elements: [11]

Exercise Quick Sort

5 min

9	7	5	11

Choose the leftmost entry 9 as the pivot.

Lesser elements: [7,5]   Pivot: 9   Greater elements: [11]

Lesser elements [5]   Pivot:7   Greater elements: ([])

Exercise Quick Sort

9	7	5	11

Choose the leftmost entry 9 as the pivot.

Lesser elements: [7,5]

Pivot: 9

Greater elements: [11]

Lesser elements [5]

Pivot:7

Greater elements: ([])

Left subarray: [5,7]

9

Right subarray: [11]

Exercise Quick Sort

9	7	5	11

Choose the leftmost entry 9 as the pivot.

Lesser elements: [7,5]

Pivot: 9

Greater elements: [11]

Lesser elements [5]

Pivot:7

Greater elements: ([])

Left subarray: [5,7]

9

Right subarray: [11]

[5,7, 9, 11]

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Done!

This Week

Sorting Algorithm

- Insertion Sort
- Heapsort
- Merge sort (Divide and conquer)
- Quick Sort (Divide and conquer)

HashTables

HashTables

Hash tables

Hashtables.pdf

Exercise HashTable

Create Hashtable for the given input: Input: [120, 130, 241, 253, 367, 446]

Hash Function: studentID mod 10

Each student ID is a three-digit number, ranging from 000 to 999.

Exercise HashTable

Create Hashtable for the given input: Input: [120, 130, 241, 253, 367, 446]

Hash Function: studentID mod 10

Each student ID is a three-digit number, ranging from 000 to 999.

Answer:

Bucket	Value
0	120 -> 130
1	241
2	
3	253
4	
5	
6	446
7	367
8	
9	

Summary

Sorting Algorithm

- Insertion Sort
- Heapsort
- Merge sort (Divide and conquer)
- Quick Sort (Divide and conquer)

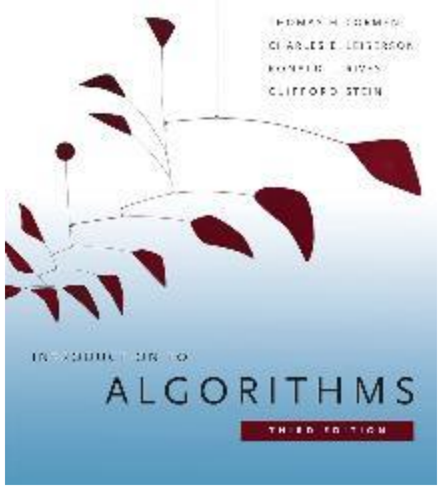
HashTables

Weekly Reading

Data Structures and Algorithm Analysis by Clifford A. Shaffer (3rd Ed)

Section 7 Sorting
Section 9.4 Hashing
https://people.cs.vt.edu/~shaffer/Book/JAVA3elatest.pdf

Introduction to Algorithms by Cormen (4th Ed)



Section 2.1 Insertion sort

Section 2.3.1 The divide-and-conquer method

Chapter 6 Heapsort

Chapter 7 Quicksort

Chapter 11 HashTables

https://ebookcentral.proquest.com/lib/bham/detail.action?docID=6925615

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Next Week

Week	Date	Topic
1	15 Jan	Searching algorithms
2	22 Jan	Binary Search Tree
3	29 Jan	Balancing Trees – AVL Tree
4	5 Feb	Databases – Conceptual Design
5	14 Feb	Databases – Logical Design & Relational Algebra
6	19 Feb	Consolidation Week
7	26 Feb	Complexity analysis, Stacks, Queues, Heaps
8	4 Mar	Sorting Algorithms, Hash tables
9	11 Mar	Graph Algorithms
10	18 Mar	Databases – Normalization
		Easter break and Eid break
11	22 Apr	Databases – Concurrency
12	29 Apr	Revision Week

Thank you.

Questions?

UNIVERSITY OF BIRMINGHAM | DUBAI دبي

Attendance