

Due Date: 04-05-2019 (F), BEFORE the class begins

This assignment covers textbook Chapter 6 and Chapter 1~5.

1. Heaps (15 points)

Given the set of integers: {3, 80, 19, 72}, how many different MAX HEAPS can be made using these integers? Justify your answer.

2. Heap and Heap property (15 points)

Array A contains integers from 1 to 2047 (including both of them) exactly once and the array is already built as a min-heap. The depth of a node in the heap is defined as the length of the path from the root of the heap to that node. Therefore, the root is at depth 0. What is the maximum depth at which integer 10 can appear? Justify your answer.

3. Heap and Heap property (15 points)

Provide a tight bound for the running time of finding the smallest element in a binary max-heap with n elements? Justify your answer.

4. Heap Sort (15 points)

Using Figure 6.4 as a model, illustrate the operation of HEAPSORT on the array $A = \langle 6, 13, 1, 45, 7, 19, 20, 8, 5 \rangle$

5. Analysis of d-ary heaps (40 points)

Textbook Problem 6-2 (Page 167). (a) ~ (d) only

Algorithms -- COMP.4040 Honor Statement
(Courtesy of Prof. Tom Costello and Karen Daniels with modifications)

Must be attached to each submission

Academic achievement is ordinarily evaluated on the basis of work that a student produces independently. Infringement of this Code of Honor entails penalties ranging from reprimand to suspension, dismissal or expulsion from the University.

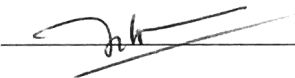
Your name on any exercise is regarded as assurance and certification that what you are submitting for that exercise is the result of your own thoughts and study. Where collaboration is authorized, you should state very clearly which parts of any assignment were performed with collaboration and name your collaborators.

In writing examinations and quizzes, you are expected and required to respond entirely on the basis of your own memory and capacity, without any assistance whatsoever except such as what is specifically authorized by the instructor.

I certify that the work submitted with this assignment is mine and was generated in a manner consistent with this document, the course academic policy on the course website on Blackboard, and the UMass Lowell academic code.

Date: 04/05/2019

Name (please print): DANG NHI NGO

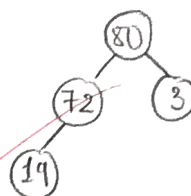
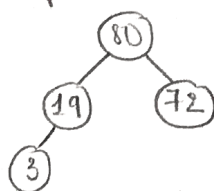
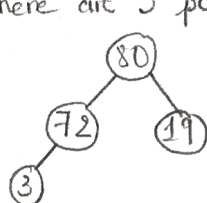
Signature: 

1/ Heaps

Given the set of integers: $\{3, 80, 19, 72\}$

There are 3 possible max heaps:

15



80 is the root, because it is the highest number

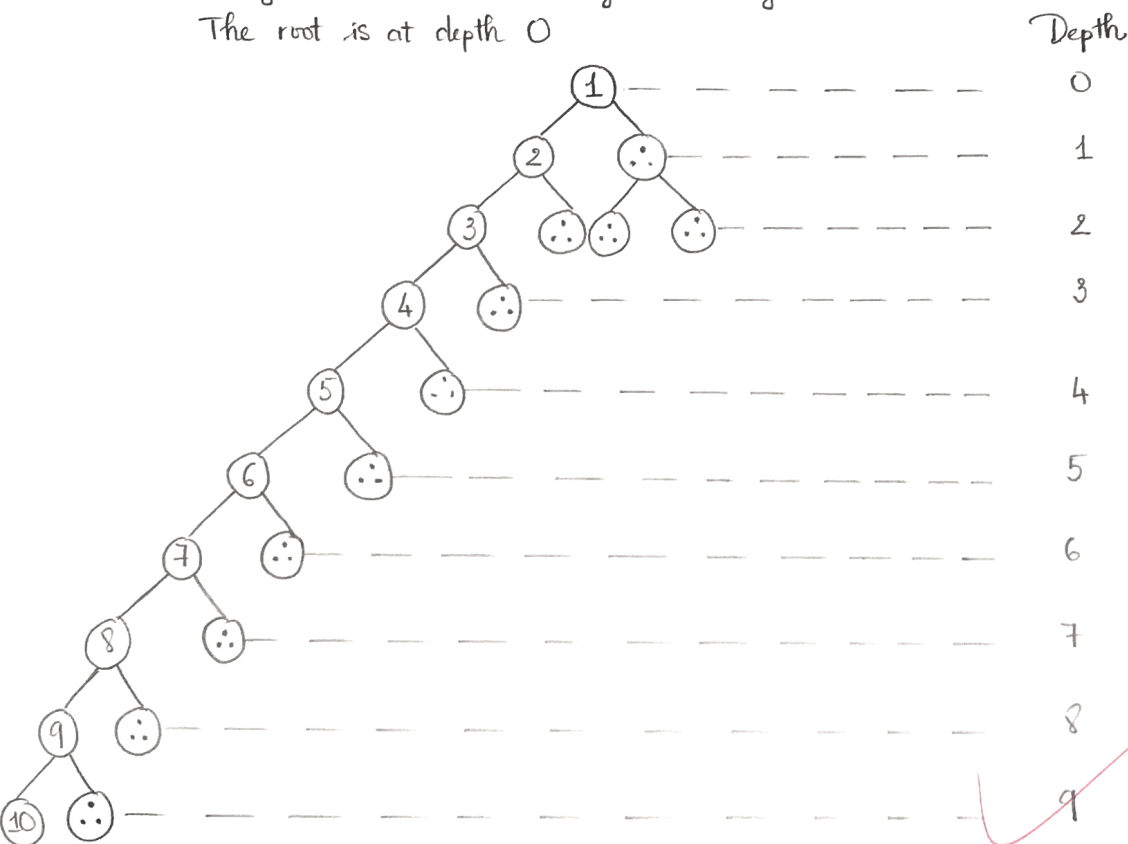
Two elements ^{which} are chosen for the left sub-tree are smaller than the root.

One element in the right sub-tree is also smaller than the root.

15 2/ Heap and heap property

Array A contains integers from 1 to 2047 (including both of them), exactly once and the array is already built as a min-heap.

The root is at depth 0



The maximum depth at which integer 10 can appear is at a depth of 9
 $\log_$

§/ Heap and heap property

Derive a tight bound for the running time of finding the smallest element in a binary max-heap with n elements.

13

To find the smallest element in a binary max-heap is to check all the leaf nodes

FindSmallestElement (A, n)	Cost	# of Execution
Smallest = $A[n]$	C_1	1
for $i = \frac{n}{2} + 1$ to $n-1$	C_2	$n-1 - \frac{n}{2} - 1 + 1 = \frac{n}{2} - 1$
if $A[i] < \text{Smallest}$	C_3	1
Smallest = $A[i]$	C_4	1
return Smallest	C_5	1

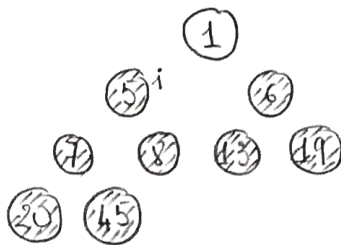
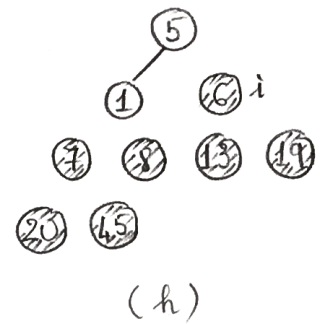
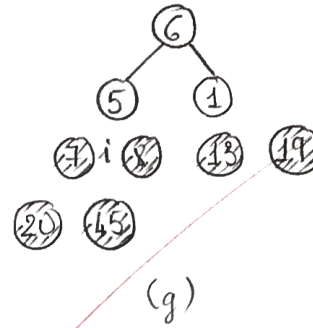
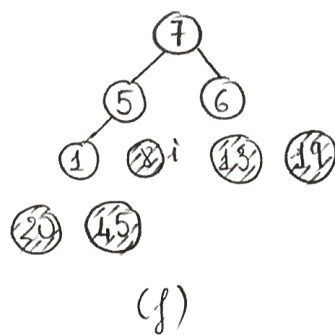
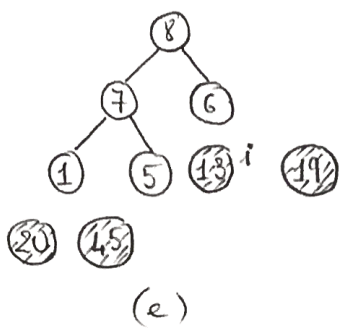
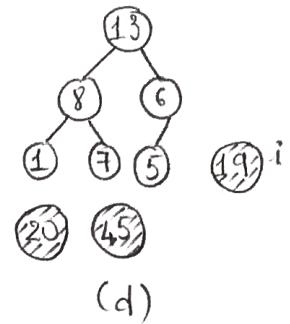
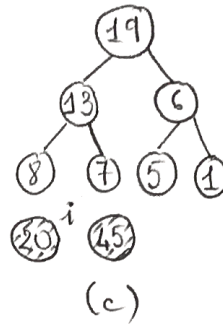
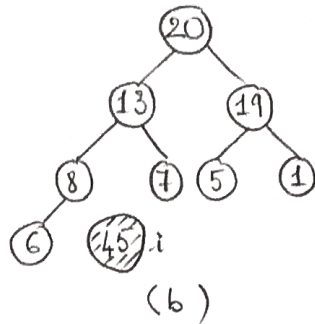
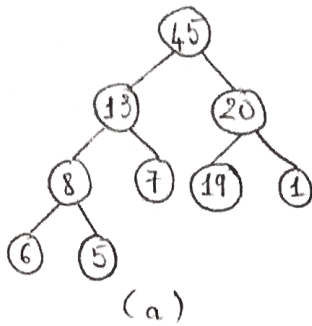
$$T(n) = C_1 \cdot 1 + C_2 \cdot \left(\frac{n}{2} - 1\right) + C_3 \cdot 1 + C_4 \cdot 1 + C_5 \cdot 1$$
$$= \Theta\left(\frac{n}{2}\right) = \Theta(n)$$

Q

4/ Heap Sort

Illustrate the operation of HEAPSORT on the array

15 $A = \langle 6, 13, 1, 45, 7, 19, 20, 8, 5 \rangle$



A

1	5	6	7	8	13	19	20	45
---	---	---	---	---	----	----	----	----

38 5/ Analysis of d-ary heaps

Problem 6-2

a/ The d-ary heap could be represented in memory like the binary heap (enumerating the nodes in top-down and left-to-right manner)

To get the n -th child ($1 \leq n \leq d$) of the node with index i ,

$$\text{CHILD}(i, d, n) = di + (1 + n - d)$$

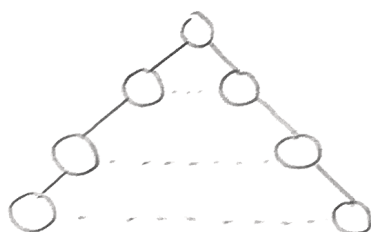
di : do not correspond to the first child anymore, but to the $(d-1)$ -th child.

The parent of the node with index i

$$\text{PARENT}(i, d) = \left\lfloor \frac{i + (d-2)}{d} \right\rfloor$$

$$\text{PARENT}(\text{CHILD}(i, d, n), d) = \lfloor (di + n - 1)/d \rfloor = i \text{ (for every } 1 \leq n \leq d)$$

b/ The height of d-ary heap of n elements in terms of n and d



of nodes

d

d^2

d^i

Total number of nodes: $n = 1 + d + d^2 + \dots + d^i$

$$n = \sum_{j=0}^i d^j$$

$$n = \frac{d^{i+1} - 1}{d - 1}$$

$$\Rightarrow n(d-1) = d^{i+1} - 1 = d(d^i) - 1$$

$$\Rightarrow \frac{n(d-1) + 1}{d} = d^i$$

$$\Rightarrow \log_d \left(\frac{n(d-1) + 1}{d} \right) = i$$

$$\Rightarrow i = \log_d [n(d-1) + 1] - \log_d d = \log_d [n(d-1) + 1] - 1$$

$$\Rightarrow i = \log_d n$$

The height of d-ary heap is $\log_d n$

c/ Give an efficient implementation of EXTRACT-MAX in a d-ary max heap. Analyze its running time in terms of d and n .

HEAPIFY(A, i, n, d)

1. $J \leftarrow i$
2. for $k \leftarrow 0$ to $d-1$
3. if $d \times i + k \leq n$ and $A[d \times i + k] > A[J]$
4. then $J = d \times i + k$
5. if $J \neq i$
6. then
7. Exchange $A[i] \leftrightarrow A[J]$
8. HEAPIFY(A, J, n, d)

EXTRACT-MAX(A, n)

1. $\text{max} \leftarrow A[1]$
2. $A[1] \leftarrow A[n]$
3. $n = n - 1$
4. HEAPIFY($A, 1, n, d$)
5. return max

The running time of this algorithm is constant and depends on the running time of HEAPIFY.

The running time of HEAPIFY is $O(d \times (\log_d n))$, because at each depth we are doing d loops, and we recurse to the depth of the tree.

Answer: $O(d \times (\log_d n))$
 Θ

d/ Give an efficient implementation of INSERT in a d-ary max-heap.

Analyze its running time in terms of d and n

INSERT (A, key)

1. A.heap-size = A.heap-size + 1

2. A[A.heap-size] = -∞

3. INCREASE-KEY (A, A.heap-size, key)

of Executions

Cost

1

C1

1

C2

1

$O(\log_d n)$

INCREASE-KEY (A, i, key)

if key < A[i]

0

C1'

error

0

C2'

A[i] = key

1

C3'

while i > 1 and A[L((i-2)/d)] < A[i]

$\log_d n + 1$

C4'

Exchange A[i] with A[L((i-2)/d)]

$\log_d n$

C5'

i = L((i-2)/d)

$\log_d n$

C6'

Running time for INCREASE-KEY:

$$T(n) = C3' + C4' \cdot (\log_d n + 1) + C5' \cdot \log_d n + C6' \cdot \log_d n$$

$$= O(\log_d n) + \Theta(1)$$

$$= O(\log_d n)$$

Running time for INSERT:

$$T(n) = C1 + C2 + O(\log_d n)$$

$$= O(\log_d n) + \Theta(1)$$

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

0