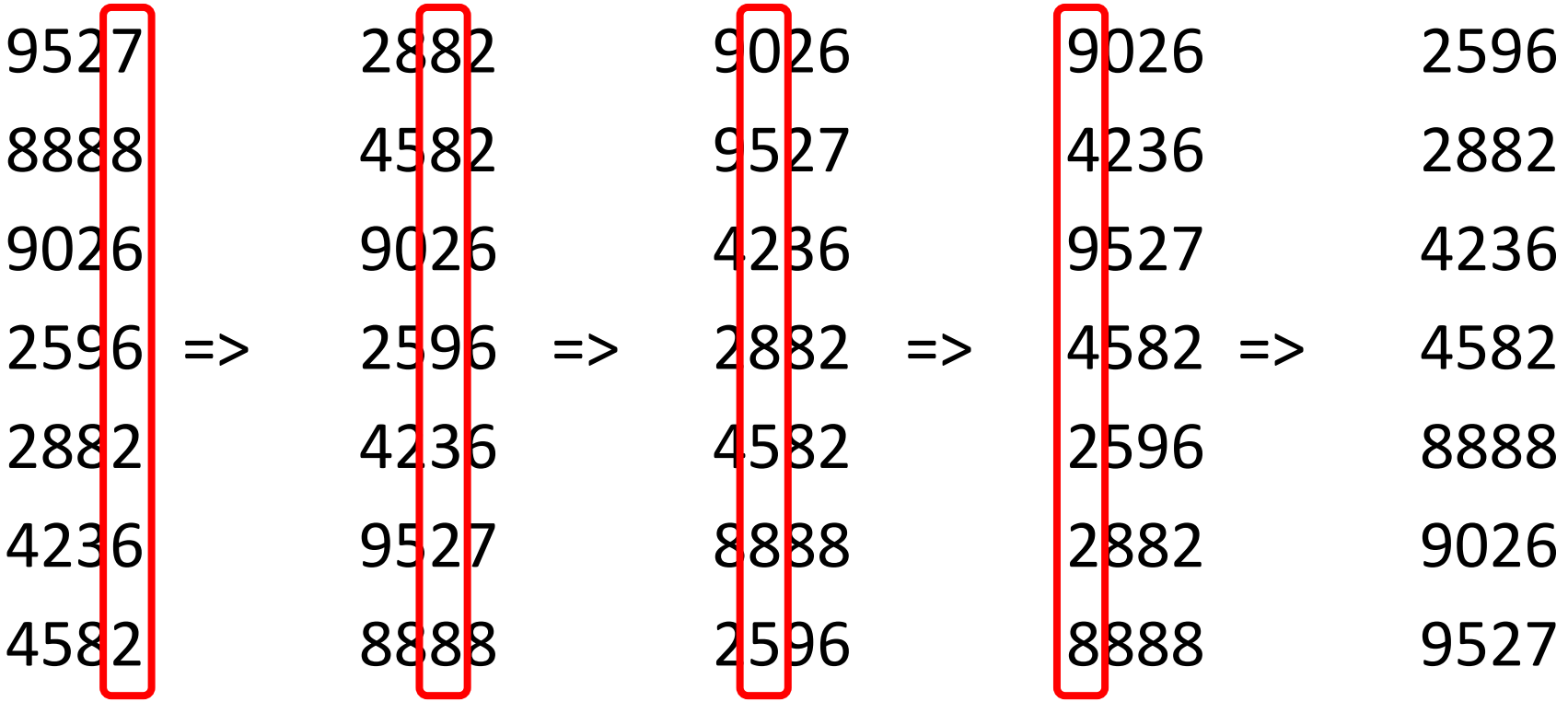
Algorithm 2019

HW 2 Solution

教授: 謝孫源 講座教授
助教: 姚凱勛 李昀 鄭力瑋 楊奕正

1.Sort the given list of numbers by radix sort with LSD to ascending order {9527, 8888, 9026, 2596, 2882, 4236, 4582}.

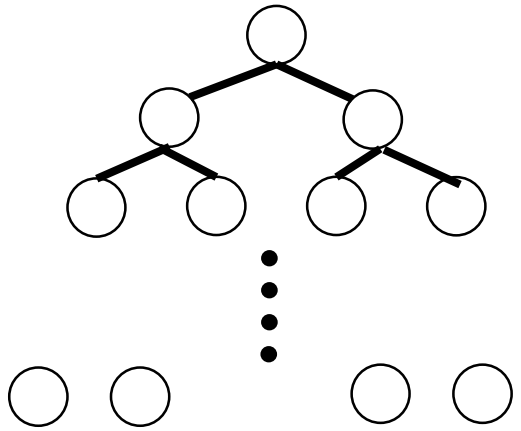


2.What situation is worst-case for quicksort? Why?
Please also derive the time complexity of worst-case.

Occurs when the subarrays are completely unbalanced.(pivot is max/min)
Have 0 elements in one subarray and n-1 elements in the other subarray.
Get the recurrence

$$T(n) = T(n-1) + T(0) + \Theta(n)$$
$$= T(n-1) + \Theta(n)$$
$$= \Theta(n^2).$$

3.(a)What are the minimum numbers of elements if the height is h? Show your solution process.
(b) What are the maximum numbers of elements if the height is h? Show your solution process.



number	height
$2^0=1$	0
$2^1=2$	1
$2^2=4$	2
2^h	h

Min:
There is only one node at the lowest layer
Total number is
$=2^0+2^1+2^2+\ldots+2^{h-1}+1$
$= 2^h$

Max:
There are full nodes at the lowest layer
Total number is
$=2^0+2^1+2^2+\ldots+2^h$
$= 2^{h+1}-1$

If you define the height of root is 1:
Min = 2^{h-1} Max = 2^h-1

3.(c) Show that an n-element heap has height $\lfloor \log n \rfloor$.

$$2^h \leq n \leq 2^{h+1} - 1 < 2^{h+1}$$

$$\Rightarrow h \leq \lg n < h+1$$

Because h is integer , $h = \lfloor \lg n \rfloor$

If you define the height of root is 1:

$h = \lfloor \lg n \rfloor + 1$

4.(10pts) We know that it is important to how to choose a good pivot in Quick-Sort. Median-of-3 is one way to deal with this problem. Please understand the Median-of-3 by yourself and illustrate the operation of Median-of-3 on the array A(you just need to explain how you choose the pivot) = {13, 19, 9, 5, 12, 8, 7, 4, 11, 2, 6, 21, 35, 8, 13, 2, 5, 6, 37, 12, 24, 26, 3, 8, 9, 10, 54, 56, 10}.

explain how you choose the first pivot

A[0],A[n-1],A[(n-1)/2] which mean the first, the last, and the middle of the array. choose the middle of them to be the pivot

or you can choose random 3 elements of array A, the choose the middle of them to be the pivot

5.(10pts) Please show how to sort n integers in the range 0 to n3-1 in O(n) time, but the space complexity is in O(n).

use radix sort
1:所有數字%n 進行radix sort 分配跟合併各花O(n)
2:所有數字/n %n 進行radix sort 分配跟合併各花O(n)
3:所有數字/n/n %n 進行radix sort 分配跟合併各花O(n)

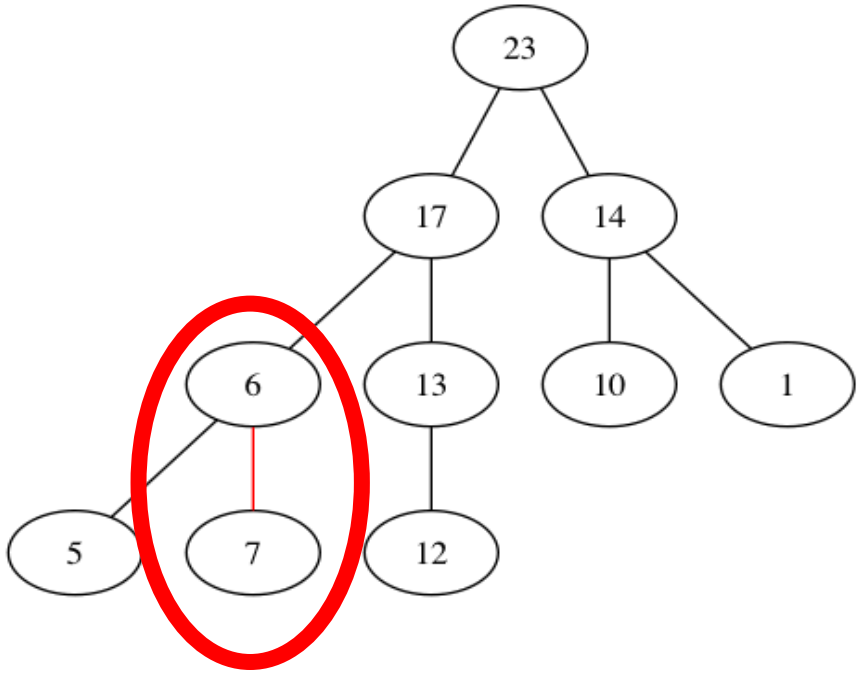
下次最好寫出為什麼是在O(n)時間複雜度跟空間複雜度內
以及每個phase在幹嘛 盡量不要只說base n時有3個bits就結束

6. (10pts) Is the array with values [23, 17, 14, 6, 13, 10, 1, 5, 7, 12] a max-heap? Please answer “Yes” or “No” and explain your reason.

- No。在max-heap中，parent的值要大於child的值，6為7的parent，但6沒有大於7。

評分標準:

- No: 5分
- 原因(只要有圈出6、7): 5分



7.(10pts) There is an array which implements a heap.

[2, 5, 3, 11, 7, 13, 17, 19, 23, 29]

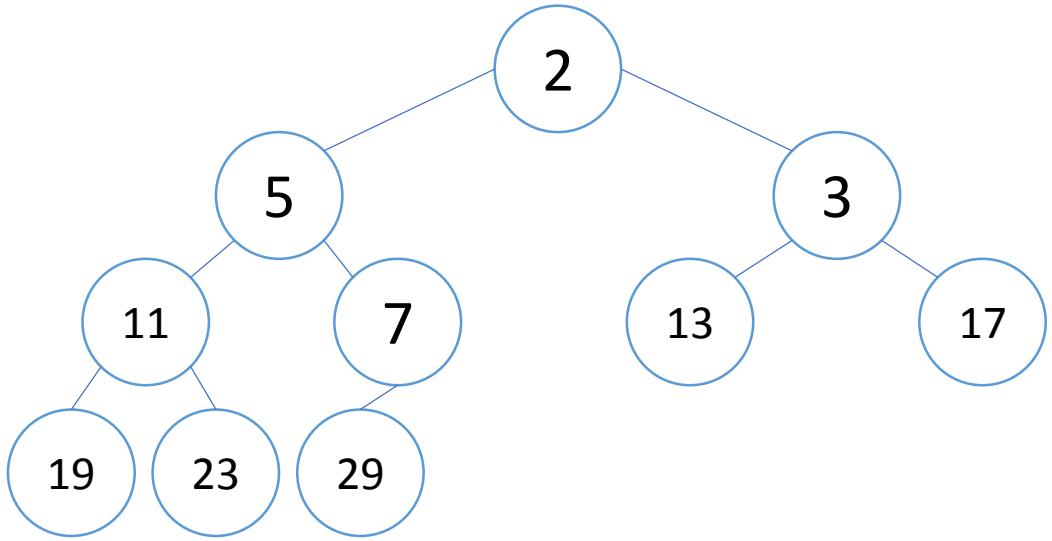
Please do the following steps and maintain the heap (you can either draw or explain) and give the min you extract.

(a) Insert 1 to the heap

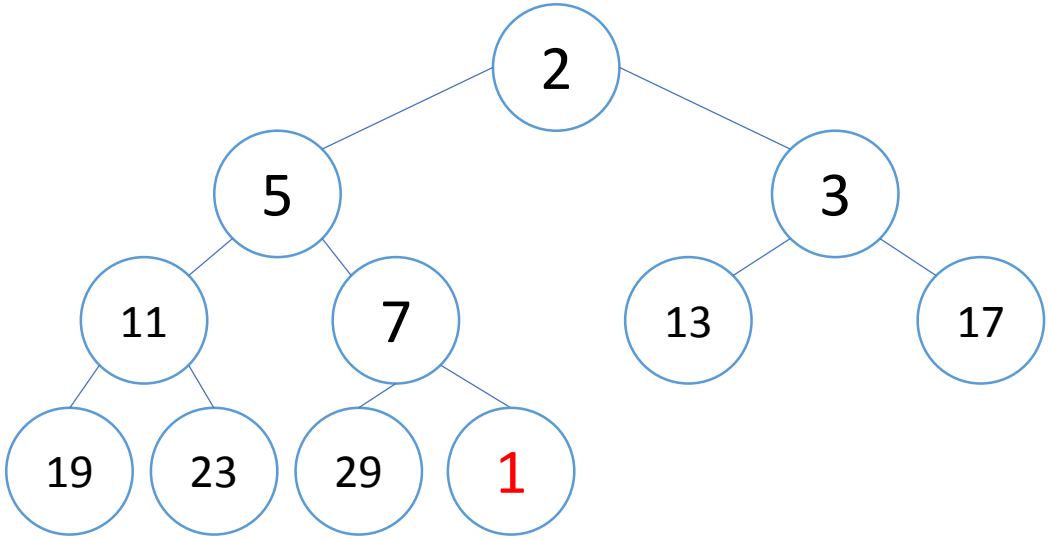
(b) Extract min

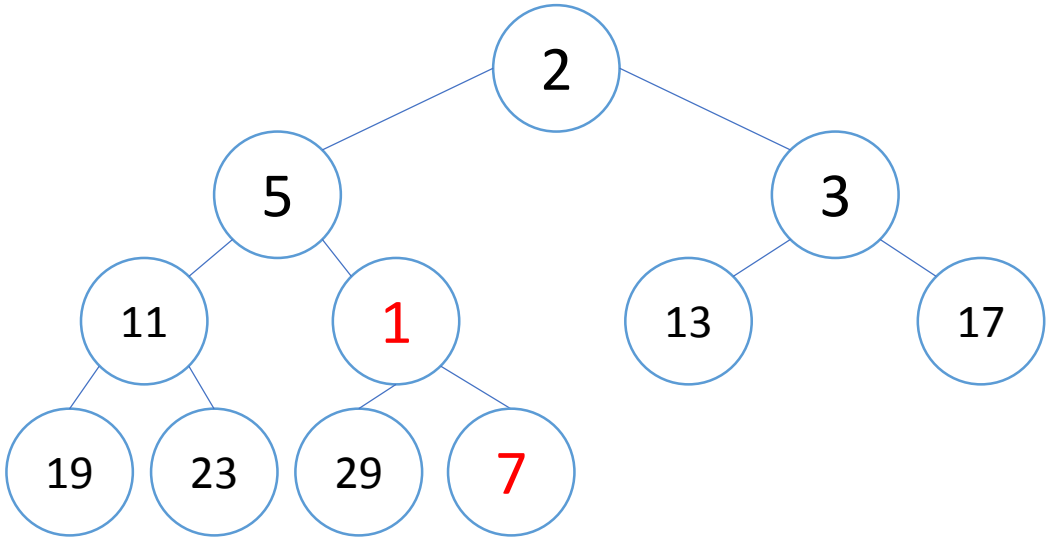
(c) Change 19 to 8

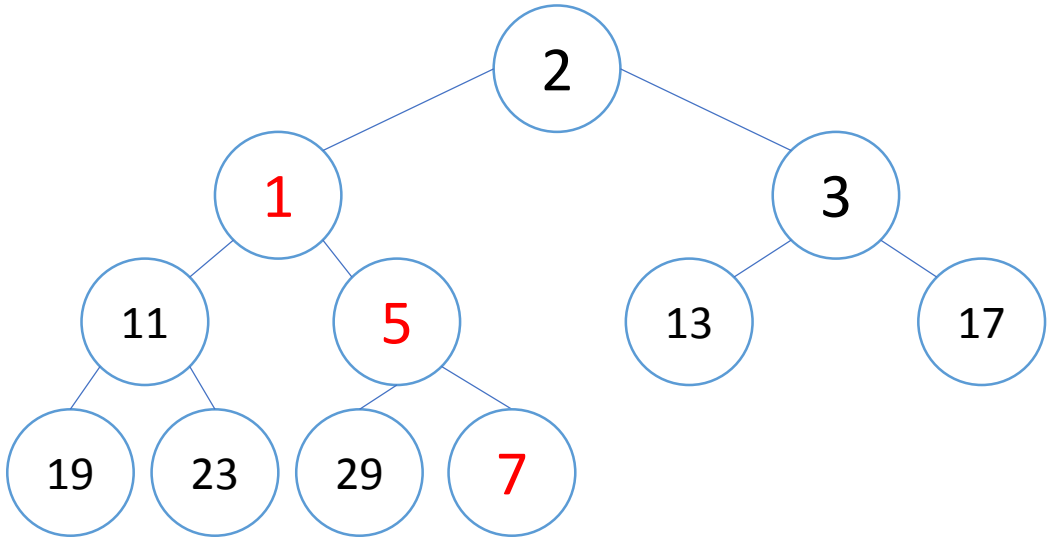
(d) Extract min

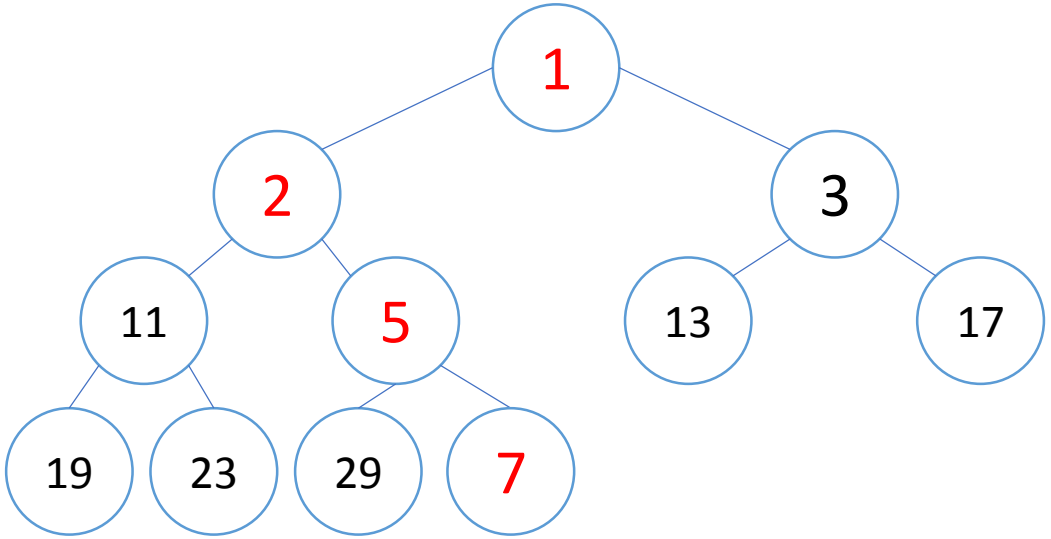


(a) Insert 1 to the heap

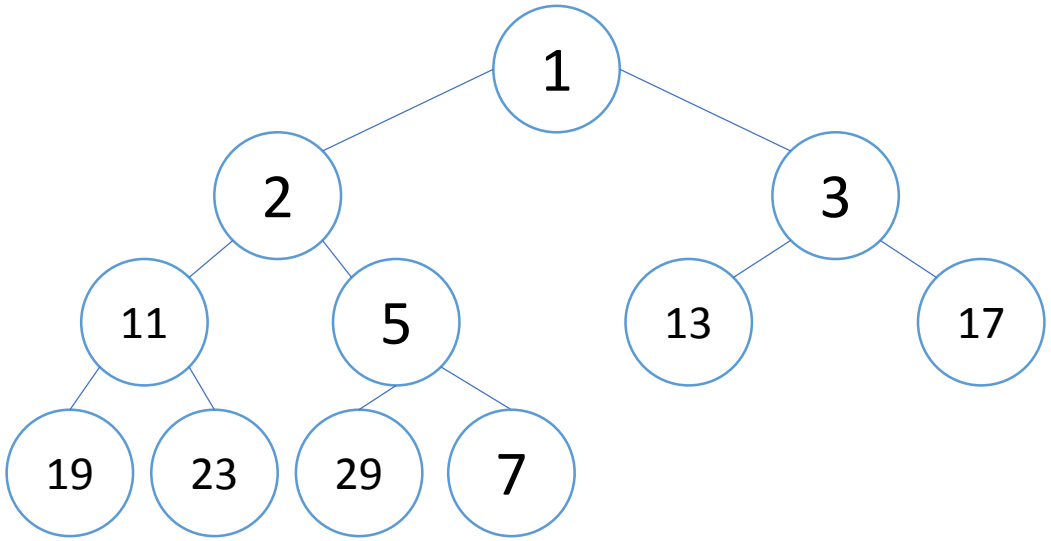




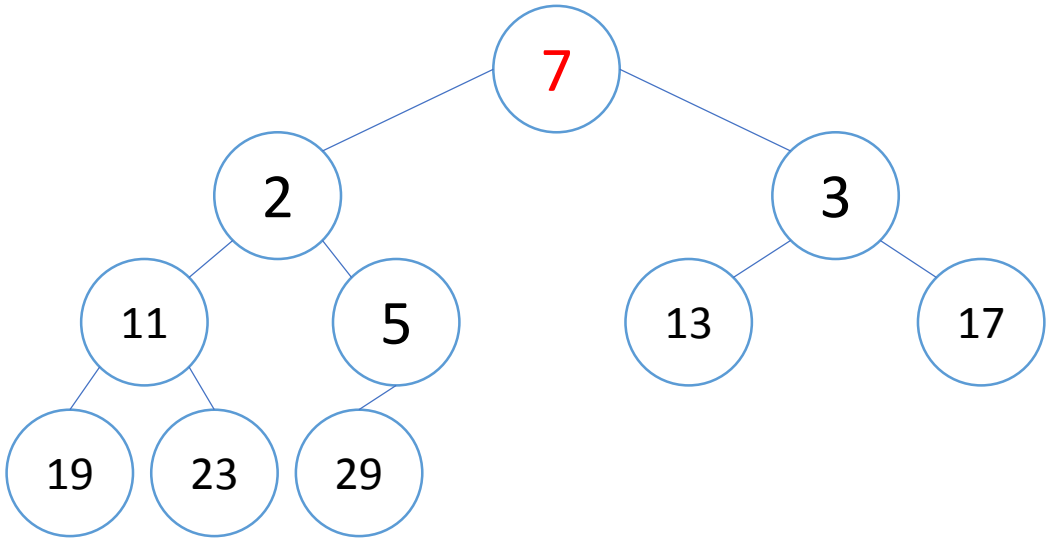


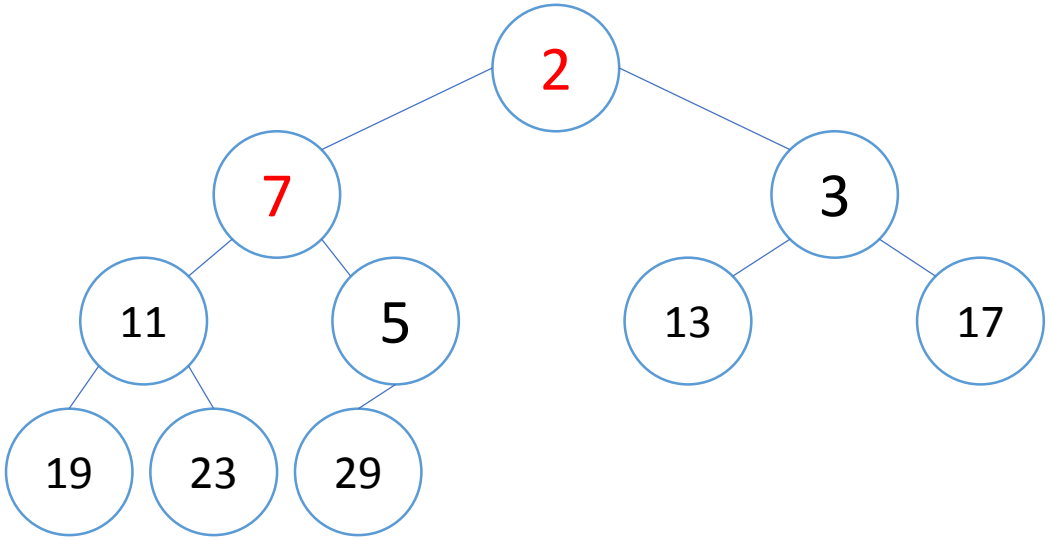


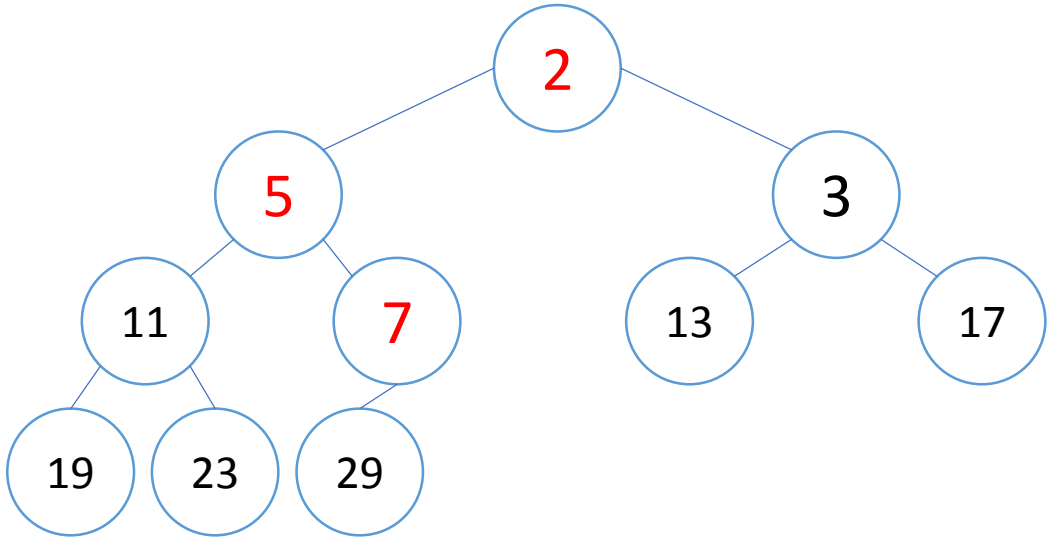
(b) Extract min



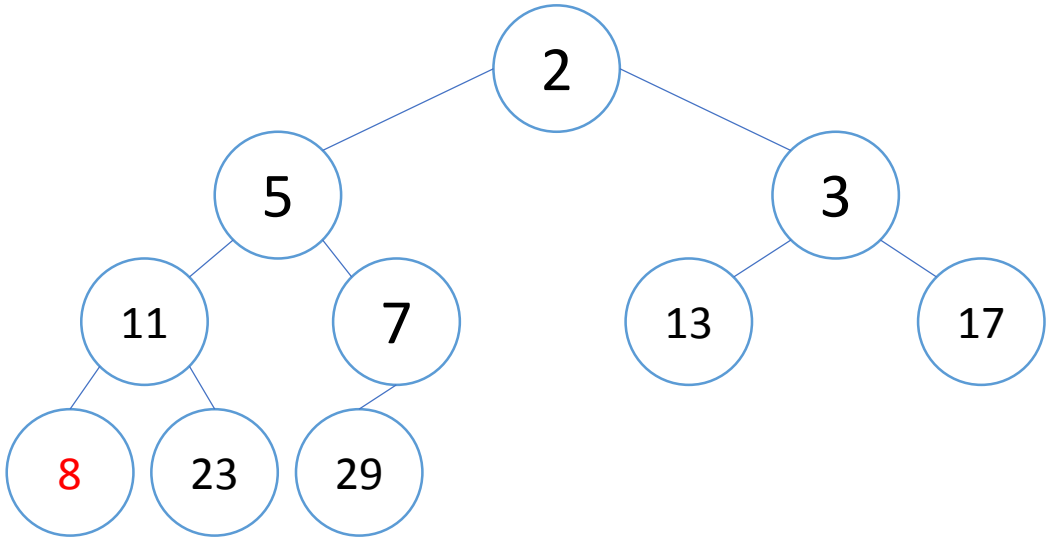
Min = 1

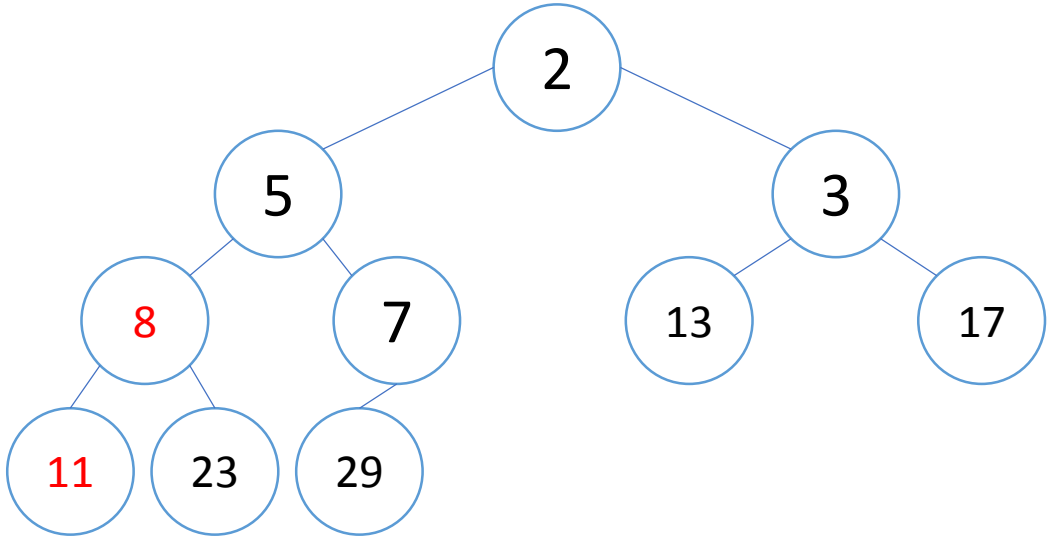




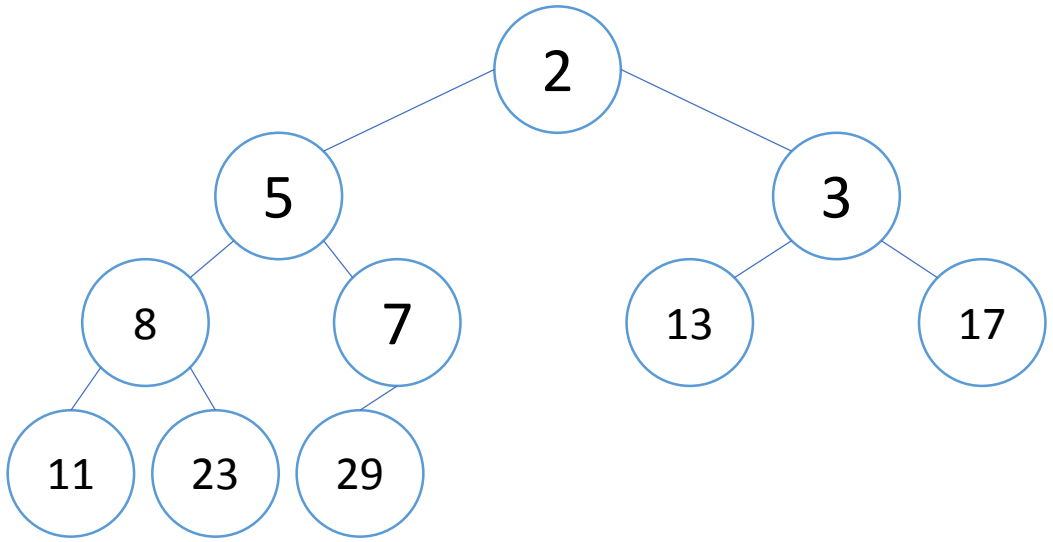


(c) Change 19 to 8

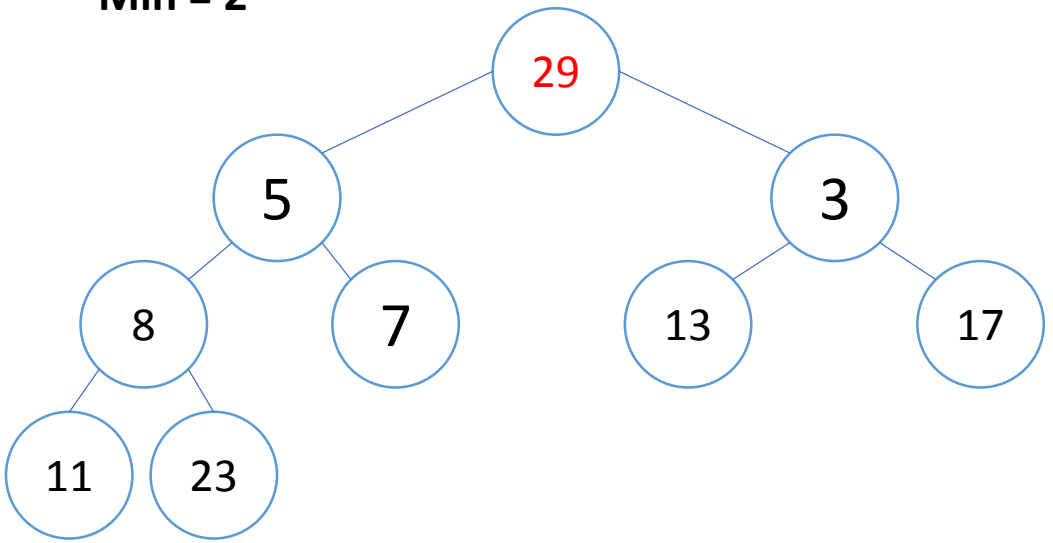


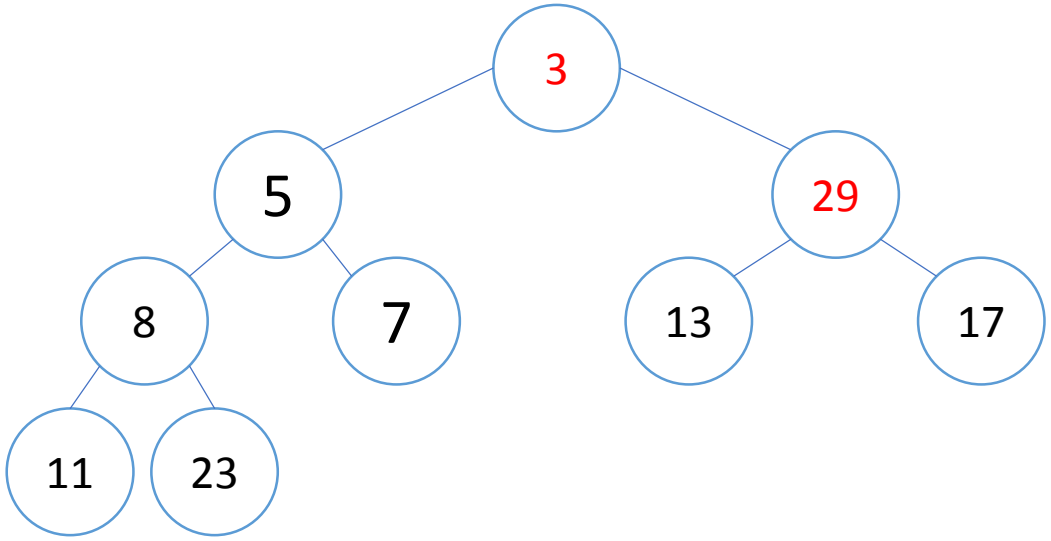


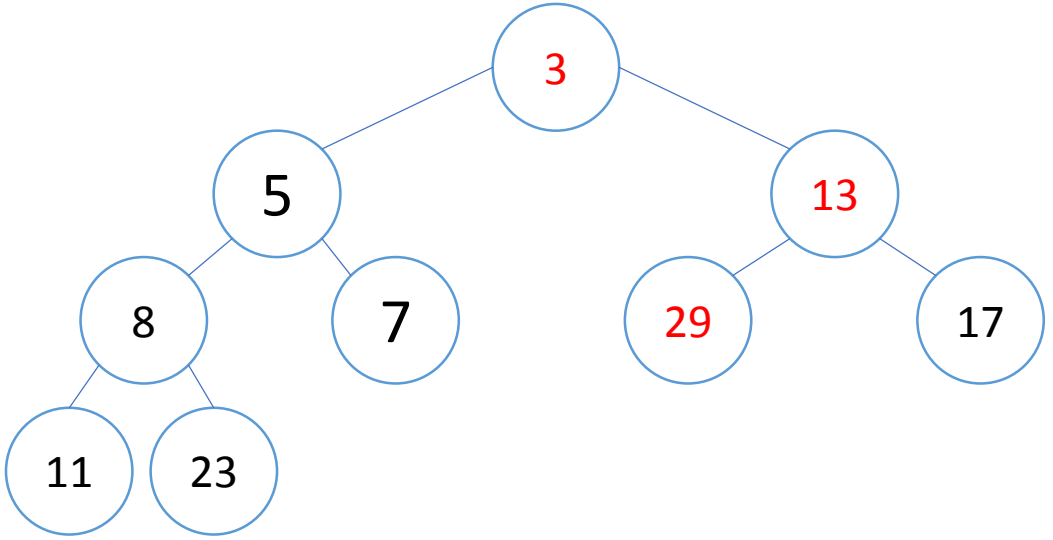
(d) Extract min



Min = 2







- Should maintain after each step
- Extract：是從最後面抓來跟root交換
- 上到下的heapify 會是跟比較小的child交換

8.(10pts) Please fill in the rest of the table and assume they will sort n things.

For counting sort, the numbers to be sorted are between 0 to *k*.

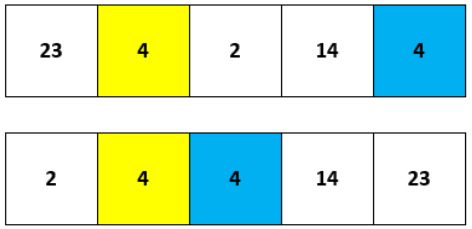
For radix sort, the digits are in range (0 to *k*) per pass, and there is *d* pass.

	Time complexity			Additional space complexity	Is it stable?
	Best case	Avg. case	Worst case		
Bubble sort	$\theta(n)$	$\theta(n^2)$	$\theta(n^2)$	$\theta(1)$	Y
Insert sort	(1)	$\theta(n^2)$	(2)	(3)	(4)
Merge sort	$\theta(n \log n)$	$\theta(n \log n)$	(5)	(6)	Y
Quick sort	$\theta(n \log n)$	$\theta(n \log n)$	(7)	$\theta(\log n)$ $\sim\theta(n)$	(8)
Heap sort	(9)	$\theta(n \log n)$	(10)	(11)	(12)
Counting sort	(13)			$\theta(n+k)$	Y
Radix sort	(14)			$O(kn)$	(15)

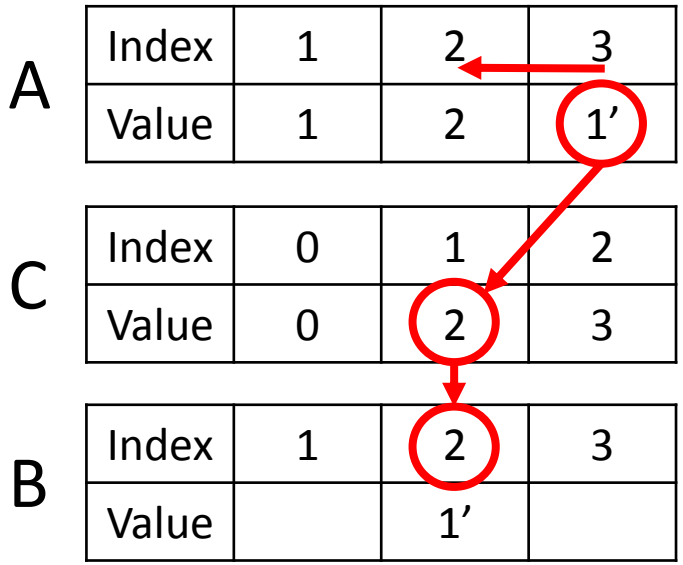
	Time complexity			Additional space complexity	Is it stable?
	Best case	Avg. case	Worst case		
Bubble sort	$\theta(n)$	$\theta(n^2)$	$\theta(n^2)$	$\theta(1)$	Y
Insert sort	$\theta(n)$	$\theta(n^2)$	$\theta(n^2)$	$\theta(1)$	Y
Merge sort	$\theta(n \log n)$	$\theta(n \log n)$	$\theta(n \log n)$	$\theta(n)$	Y
Quick sort	$\theta(n \log n)$	$\theta(n \log n)$	$\theta(n^2)$	$\theta(\log n)$ $\sim\theta(n)$	N
Heap sort	$\theta(n \log n)$	$\theta(n \log n)$	$\theta(n \log n)$	$\theta(1)$	N
Counting sort	$\theta(n+k)$			$\theta(n+k)$	Y
Radix sort	$\theta(d \times (n+k))$			$O(kn)$	Y

9. (a) (5pts) Explain what is a stable sorting algorithm, that is, what does it mean for a sorting algorithm to be "stable"?
(b) (5pts) Explain how come COUNTING-SORT is a stable sorting algorithm.

- (a) 如果鍵值相同之資料，在排序後相對位置與排序前相同時，即為stable sort algorithm。



- (b) 因C array中存放的是該數值的最後一個在B array中出現的位置，所以要從A的最後一個element開始放入B，放入後將C陣列數值-1，下個有相同數值的element就會擺在前一位。



評分標準:

- (a) 說明不完整，+3分。
- (b) 有講到是因為最後的for loop，+5分。原因不充分或只說counting sort排序後順序不變，+3分。

9. (c) (10pts) Is the modified algorithm stable? Please answer "Yes" or "No" and explain your reason.

- (c) No，承(b)的解釋，若改為順向for loop會使相對位置相反，因從A的頭開始拿，原本在前面的會被放到後面。

A

Index	1	2	3
Value	1	2	1'

C

Index	0	1	2
Value	0	2	3

B

Index	1	2	3
Value		1	

評分標準:

- (c) 寫No，因順序會相反，+10分。
  - 如果寫到c要變+1，卻沒有先處理c變成"各數值開始的位置"(原本是"各數值結束的位置")，因為會蓋到別人的位置，+8分

> Prove that `COUNTING-SORT` is stable.
>
> An informal argument will suffice.
>
> Let's say that two elements at indices $i_1 < i_2$ are equal to each other. In the sorted array, they take place at indices $j_1 + 1 = j_2$. Since the `COUNTING-SORT` processes the input array in reverse order, $A[i_2]$ is put in $B[j_2]$ first and then $A[i_1]$ is put in $A[j_2]$. Since the two elements preserve their order, the algorithm is stable.

$A[i_1]$ is put in **$B[j_1]$**