Homework 2 共 20 題

Chapter 6 ~ Chapter 7

1. Please fill in the blanks below.

   A heap can be stores as an array A:

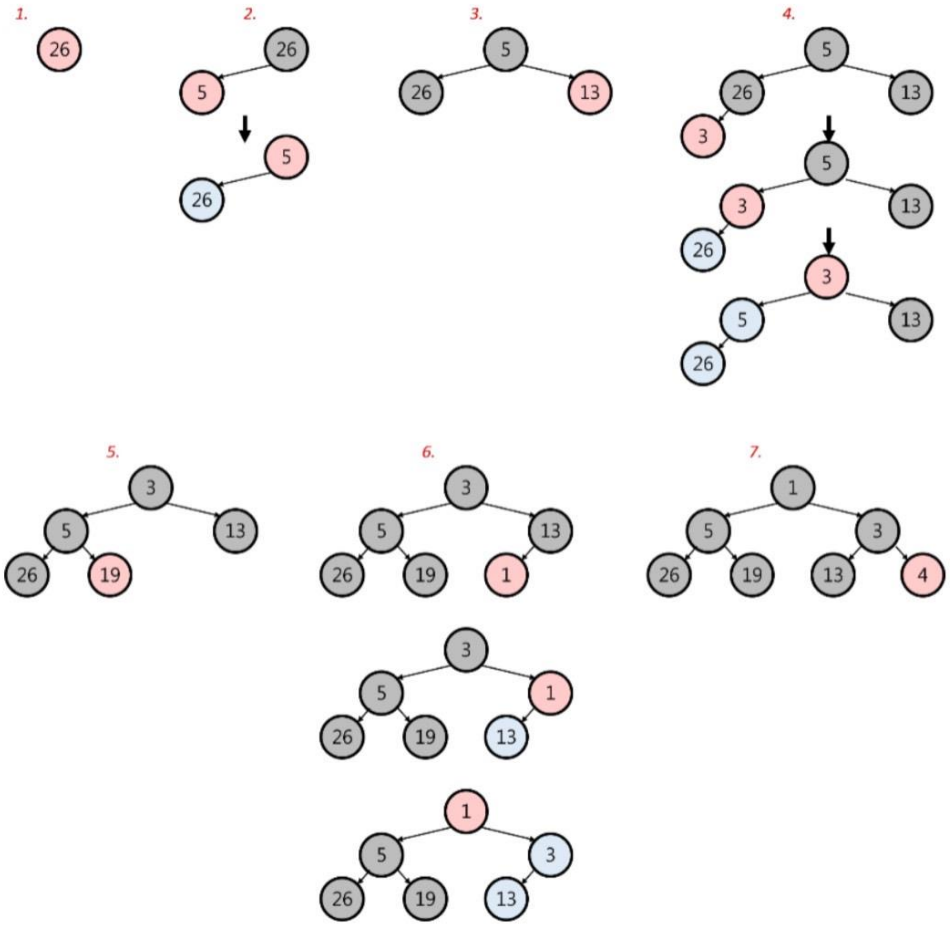
   Root of trees is _____ A[1]

   Parent of A[i] = _____ A[$\lfloor i/2 \rfloor$]

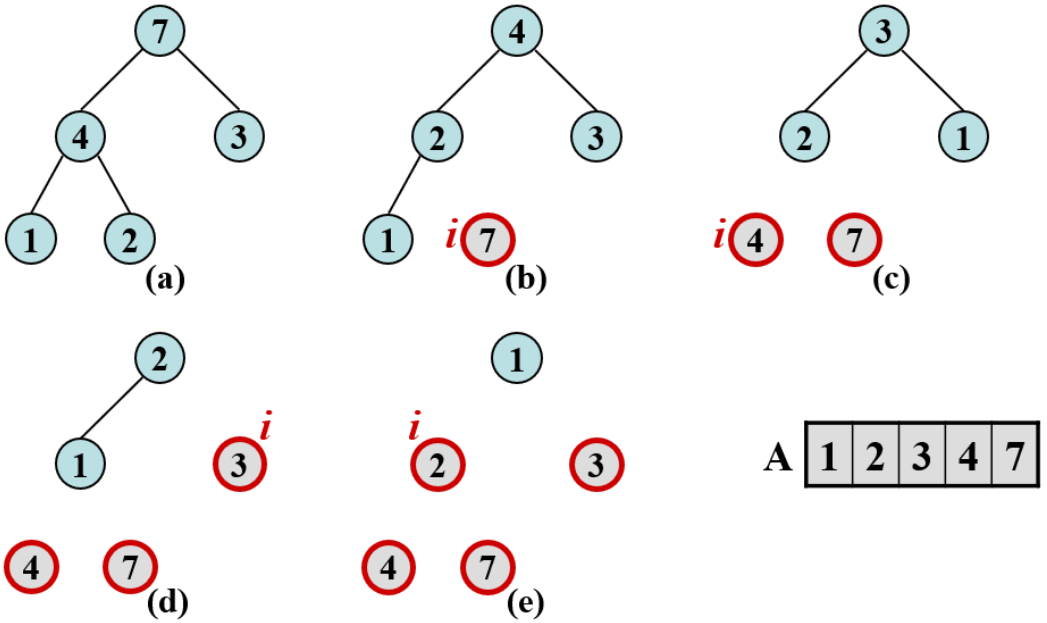
   Left child of A[i] = _____ A[$2i$]

   Right child of A[i] = _____ A[$2i$+1]

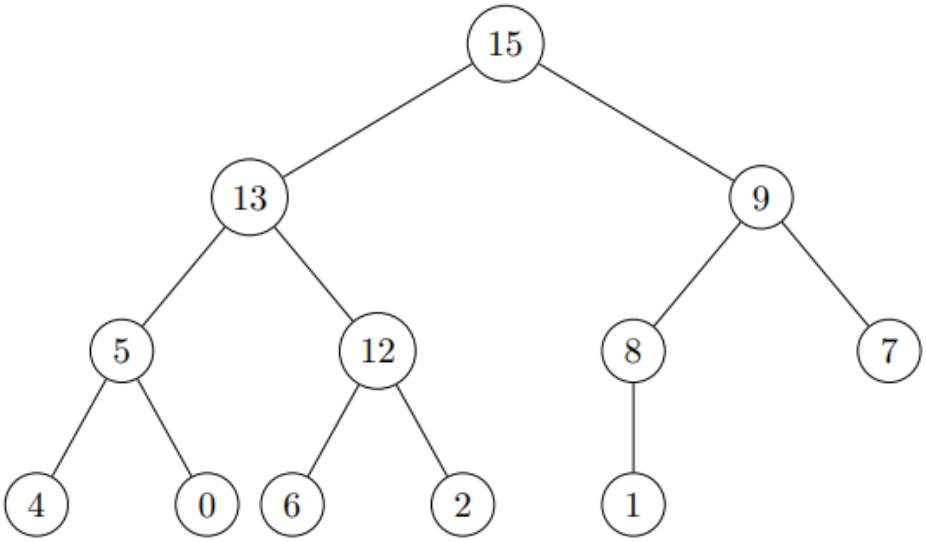
2. Building a min-heap from the following unsorted array A = [26,5,13,2,19,1,4]

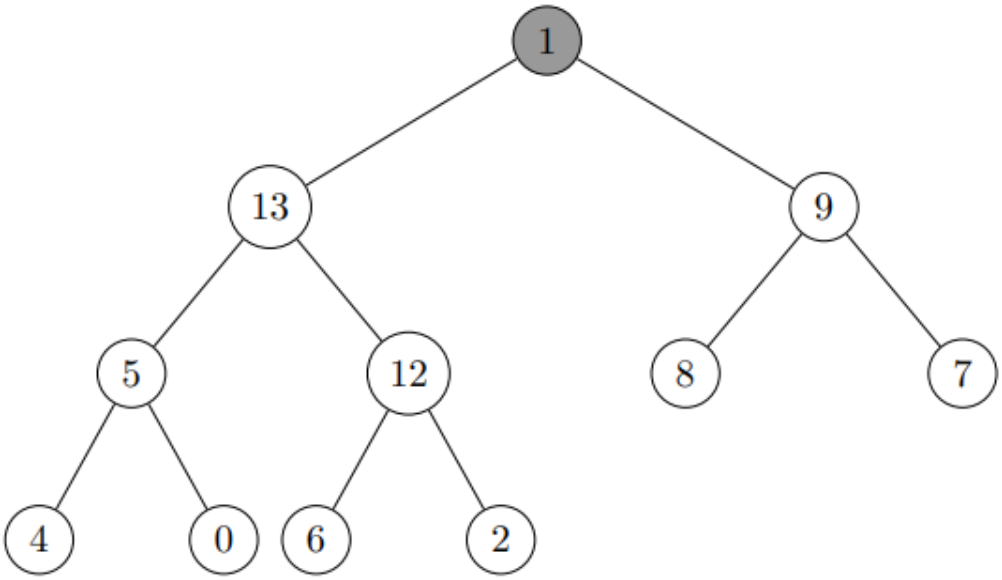


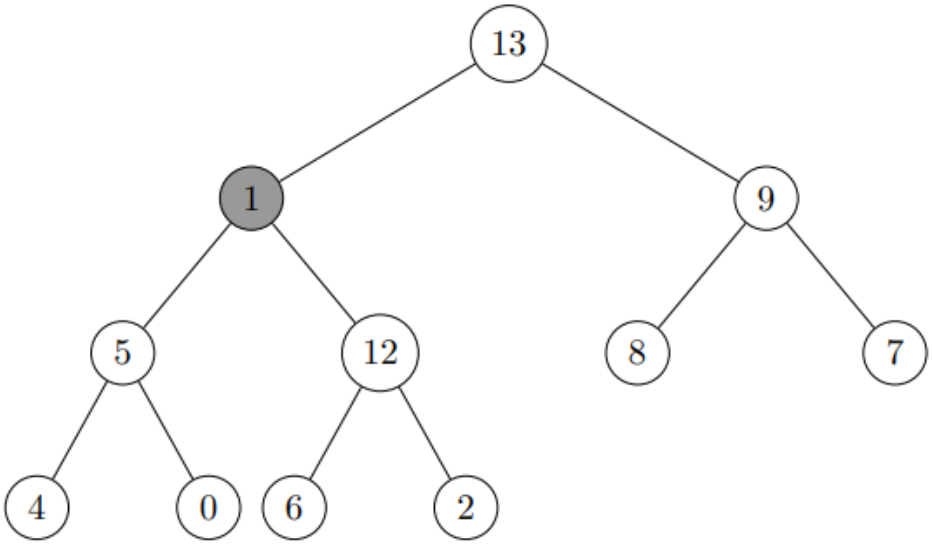
3. Sort the array A = [7,4,3,1,2] using heapsort and sequentially illustrate the steps.

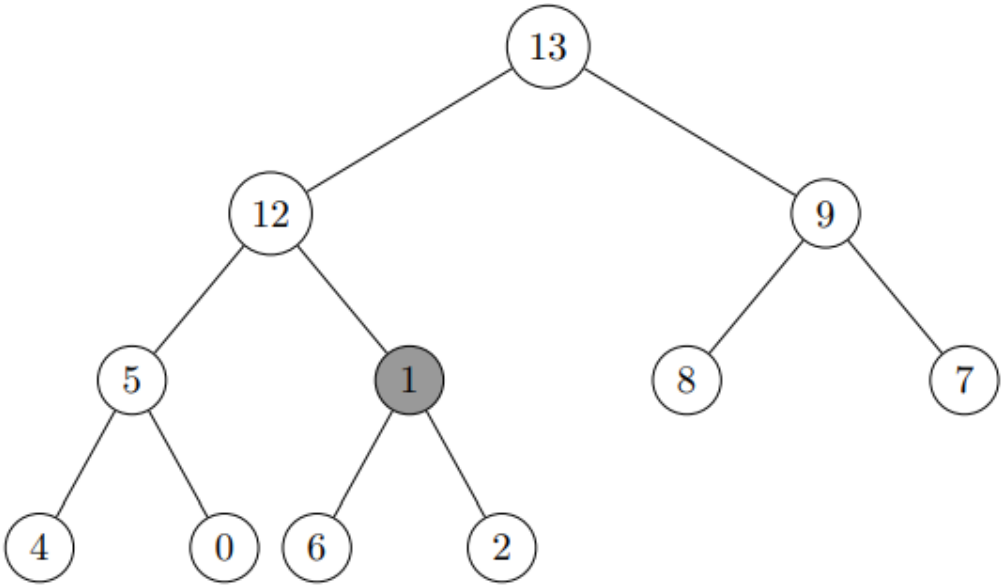


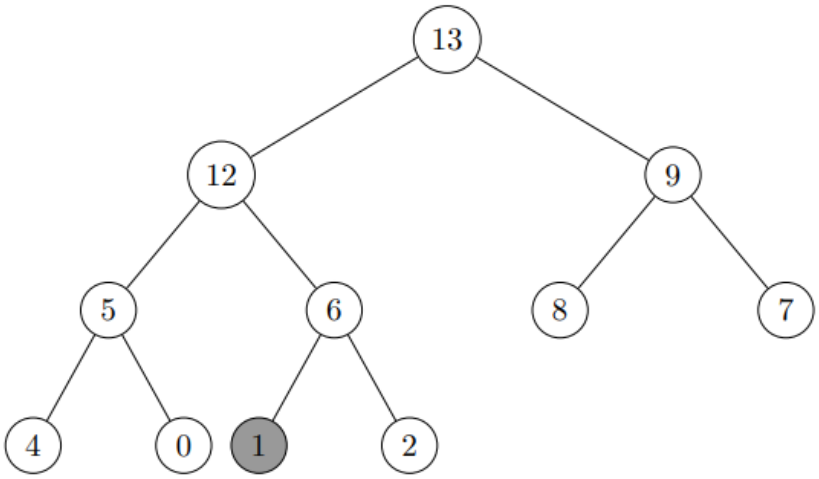
4. Illustrate the operation of HEAP-EXTRACT-MAX on the heap A = [15,13,9,5,12,8,7,4,0,6,2,1]



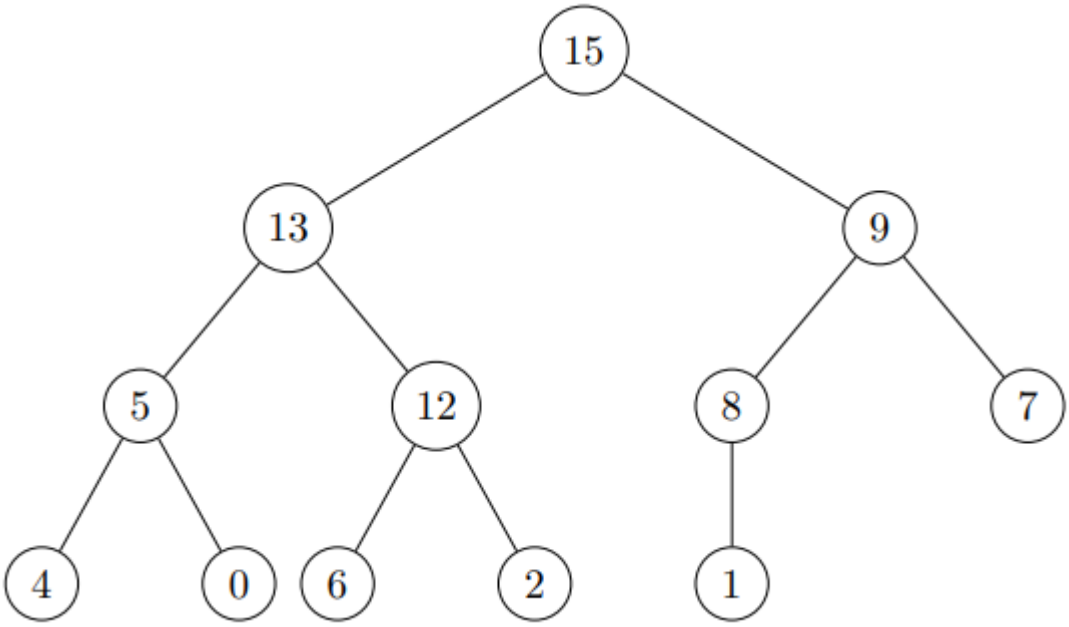

1
13
9
5
12
8
7
4
0
6
2

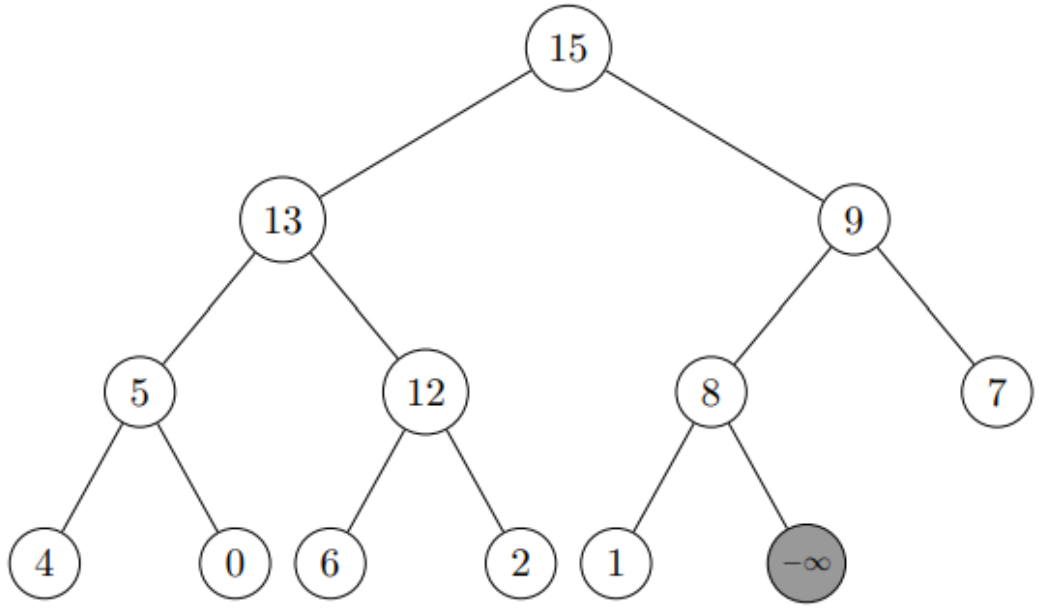

13
1
9
5
12
8
7
4
0
6
2

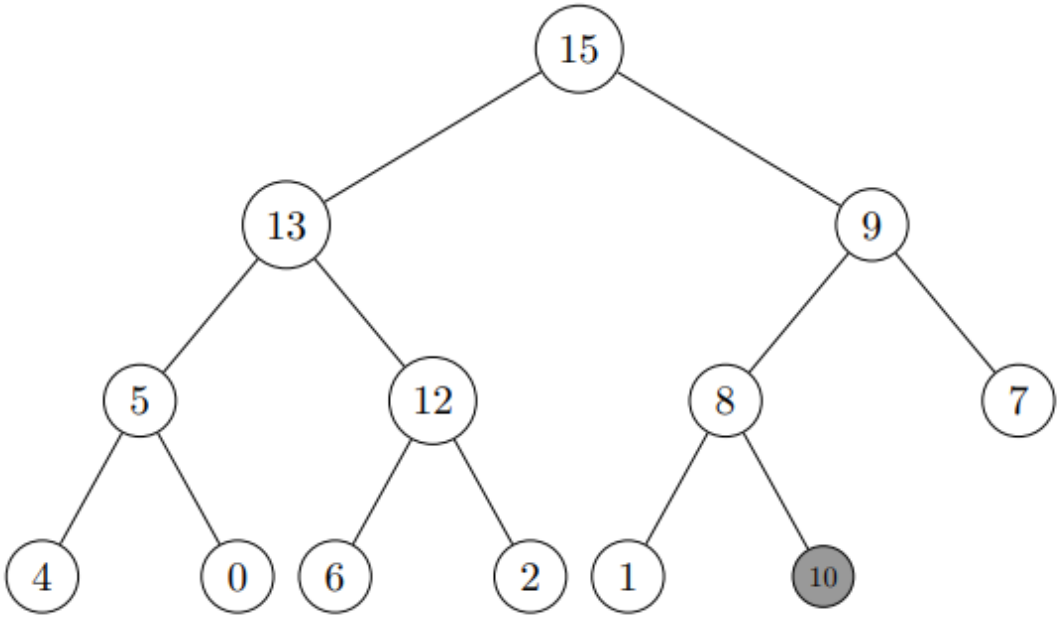

13
12
9
5
1
8
7
4
0
6
2

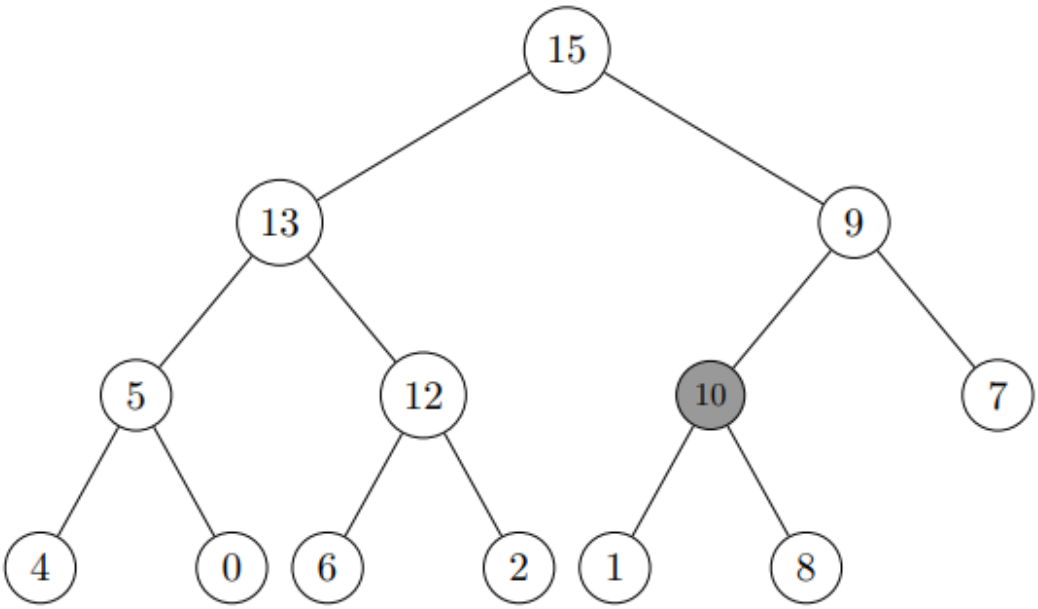


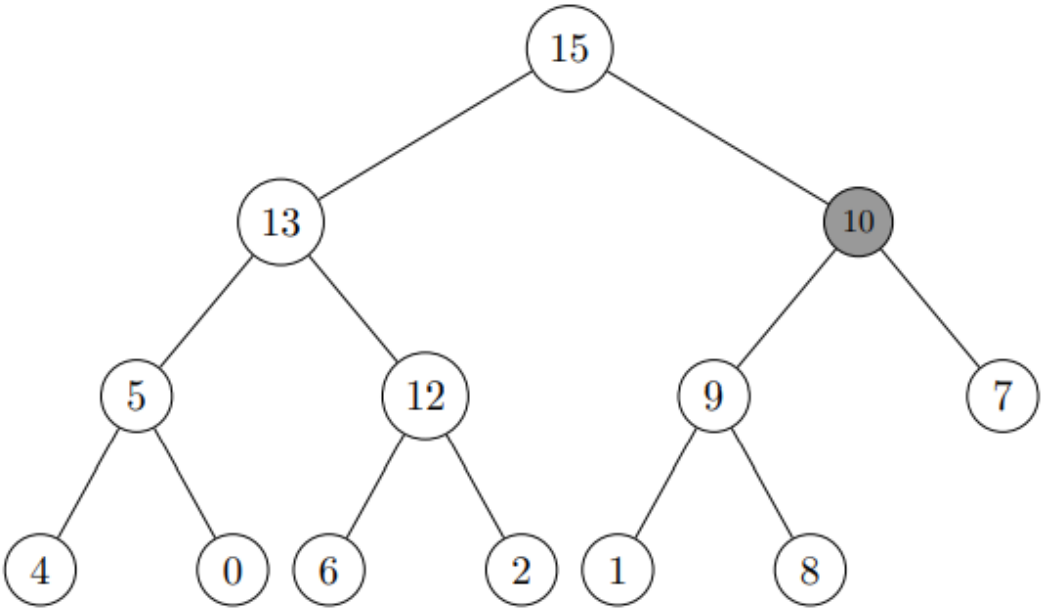
5. Illustrate the operation of MAX-HEAP-INSERT.A; 10/ on the heap A = [15,13,9,5,12,8,7,4,0,6,2,1].



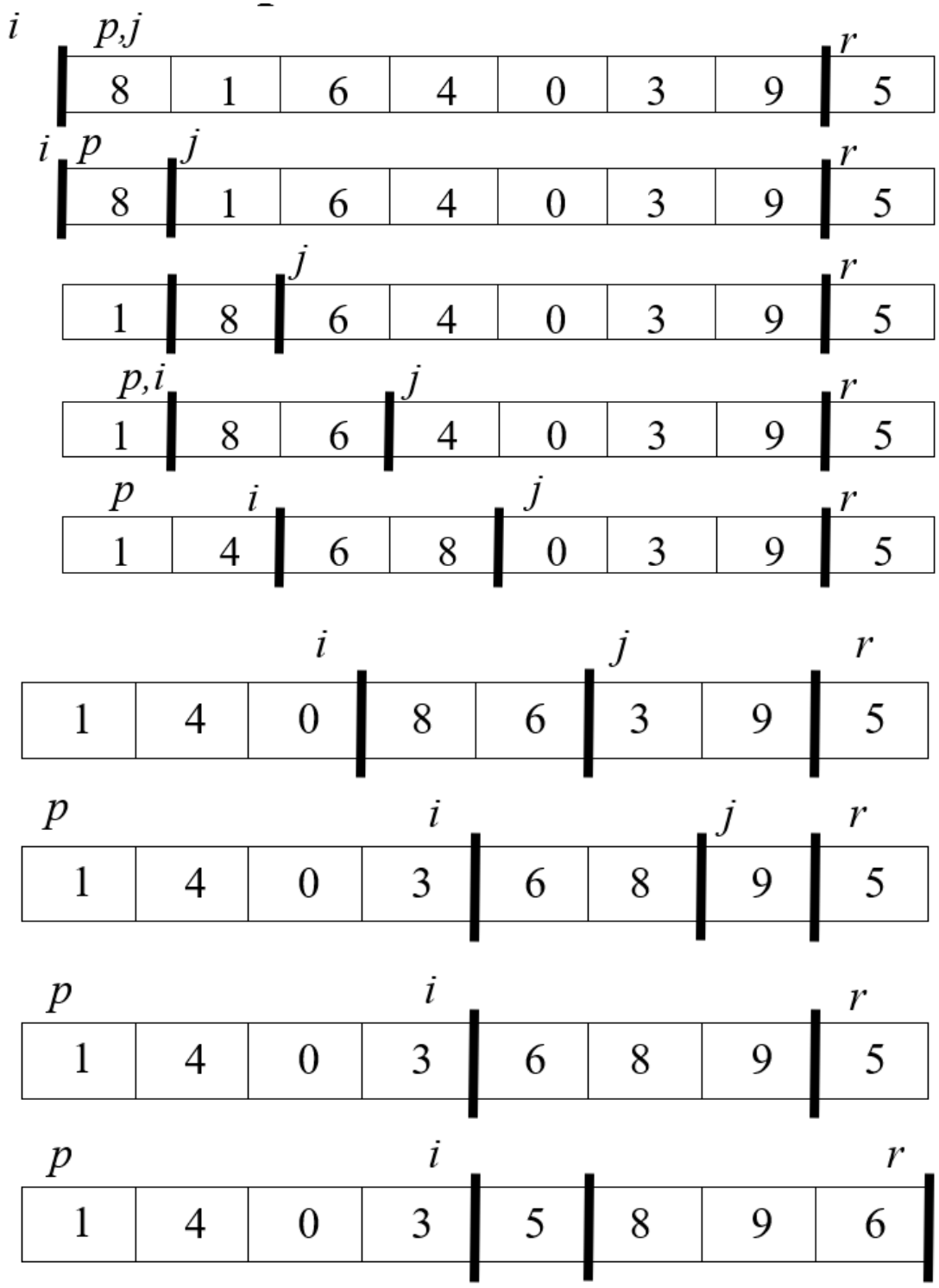

15
13
9
5
12
8
7
4
0
6
2
1
−∞


15
13
9
5
12
8
7
4
0
6
2
1
10





6. Sort the array A = [8,1,6,4,0,3,9,5] using quicksort and sequentially illustrate the steps.



7. Suppose that the following list is the result of the first partition step of quick sort.

   12 9 1 13 19 24 22 20

   What numbers could be pivots? (Multiple answers possible) 13,19

Quicksort is a fast and commonly used comparison-based sorting algorithm. Below you can find a version of quicksort algorithm in pseudo code format.

```
QUICKSORT(A, p, r)
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)

PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

8. True or false: The QUICKSORT() above is an in-place sorting algorithm. T
9. The given version of QUICKSORT above cannot efficiently process the cases where in the input array there are a large number of repeated elements. Analyze the worst-case running time which occurs when all n keys in the input array A are identical. Please give it in big-O notation in terms of n.

$$As\ worst\ case\ T(n) = T(n-1) + T(0) + O(n) = O(n^2)$$

10. Recurrence for the worst-case running time of QUICKSORT:

$$T(n) = \max_{0 \le q \le n-1} \big(T(q) + T(n-q-1)\big) + \theta(n)$$

Because PARTITION produces two subproblems, totaling size n – 1, q ranges from 0 to n – 1.

Using substituting method to proof the worst-case running time is $\theta(n^2)$.

$$T(n) \le \max_{0 \le q \le n-1} (cq^2 + c(n-q-1)^2) + \Theta(n)$$
$$= c \max_{0 \le q \le n-1} (q^2 + (n-q-1)^2) + \Theta(n).$$

The maximum value of $(q^2 + (n-q-1)^2)$ occurs when q is either 0 or $n-1$. (Second derivative with respect to q is positive.) This means that

$$\max_{0 \le q \le n-1} (q^2 + (n-q-1)^2) \le (n-1)^2$$
$$= n^2 - 2n + 1.$$

Therefore,

$$T(n) \le cn^2 - c(2n-1) + \Theta(n)$$
$$\le cn^2 \qquad \text{if } c(2n-1) \ge \Theta(n).$$

Pick c so that $c(2n-1)$ dominates $\Theta(n)$.

Therefore, the worst-case running time of quicksort is $O(n^2)$.

Can also show that the recurrence's solution is $\Omega(n^2)$. Thus, the worst-case running time is $\Theta(n^2)$.

To address the previously mentioned problematic case, we will now develop a new partition algorithm PARITITIONTHREE() that separates the input keys into three groups, the ones that are smaller than, that are smaller than, that are

identical to, and that are larger than the pivot key x. The return values q1 and q2 represent the indices of the first and the last keys of the group that equals the pivot key, respectively. Fill the blanks (A), (B), (C) in PARITIONTHREE() below to complete the implementation, such that QUICKSORTTHREE() runs in O(n)-time when the input array has all n elements identical.

```
QUICKSORTTHREE(A, p, r)
1  if p < r
2       (q1, q2) = PARTITIONTHREE(A, p, r)
3       QUICKSORTTHREE(A, p, q1 − 1)
4       QUICKSORTTHREE(A, q2 + 1, r)

PARTITIONTHREE(A, p, r)
 1  x = A[r]
 2  q1 = p
 3  q2 = r
 4  j = p
 5  while j ≤ q2
 6       if A[j] < x
 7            exchange A[j] with A[q1]
 8            // (A)
 9            j = j + 1
10       elseif A[j] > x
11            exchange A[j] with A[q2]
12            // (B)
13       else
14            // (C)
15  return (q1, q2)
```

q1 = q1 + 1

q2 = q2 – 1

j = j +1

Symmetric Min-Max Heap (SMMH) is defined as a complete binary tree, which supports the implementation of double-ended priority queue. The insert, delete-min, and delete-max are three major operations with complexity O(log n). It satisfies the following properties:

(1) Root contains no data.

(2) Left sibling data ≤ Right sibling data

(3) If node A has grandparent X, the data in the left child of the node X is equal to or smaller than the data in X

(4) If node A has grandparent X, the data in the right child of the node X is equal to or larger than the data in X.

Build an SMMH by inserting elements 13, 25, 18, 30, 40, 50, 22, 16, 19, 96 in order. If SMMH is implemented with the array data structure T[] starting with index 1, which represents the root node.

Please show T[3] in the final SMMH (5%). 96

If we apply extract-min, please show T[4] in the resulted SMMH (5%). 18

Instead, if we apply extract-max to the original SMMH, please show T[5] in the final SMMH (5%). 25

11. What are the maximum numbers of elements in a heap of height h?

    At most $2^{h+1} - 1$.

12. Is the array with values《20, 1, 5, 7, 22, 23, 27, 44, 16,13》a min-heap?

    No, the array is not a min-heap. The value of the parent node must be less than or equal to the values of its children nodes. the parent node 20 is bigger than its children 1 and 5, the parent nodes violate this rule.

13. Which sorting algorithm typically performs best on nearly sorted arrays, where only one or two elements are out of place?

    (A) Heap sort
    (B) Quick sort
    (C) Insertion sort
    (D) Merge sort

    C, Insertion sort takes linear time when input array is sorted or almost sorted. All other sorting algorithms mentioned above will take more than linear time in their typical implementation

14. What is the effect of calling MAX-HEAPIFY（A , i）on element A[i] achieve when it's already larger than its children?

    It remains unchanged because the if statement on line 8 evaluates to false.

15. （Multiple choice）Choose the correct option from the following statements about the heap sort algorithm:

(A) The BUILD-MAX-HEAP procedure creates a max-heap from an unordered input array in O(n) time.

(B) The HEAP-INCREASE-KEY procedure operates in O(lg n) time to adjust the key of a new node to its correct value.

(C) The MAX-HEAPIFY procedure, which runs in O(n lg n) time, is crucial for preserving the max heap property.

(D) The MAX-HEAP-INSERT operation, which operates in O(log n) time, is responsible for implementing the insertion operation.

ABD, (C) The MAX- HEAPIFY procedure, which runs in O(lg n) time, tasked with preserving the max-heap property, not O(n lg n).

16. Briefly describe argument that the running time of PARTITION on a subarray of size $\Theta(n)$.

For loop iterates exactly $r-p$ times, each iteration takes constant time. The section outside the for loop also takes constant time. As $r-p$ represents the size of the subarray, PARTITION takes time proportional to the size of the subarray it's called on.

17. What is the worst-case time complexity of quicksort when the (n/4)th smallest element is selected as the pivot using an O(n) time algorithm?

θ (n*log(n)). recursion expression is T(n) = T(n/4) + T(3n/4) + cn.

upon solving the recursion, determine that the time complexity is θ(n*log(n)).

18. For Quicksort sorting in ascending order using the first element as the pivot:

    (i) n, n-1, n-2, ..., 2, 1

    (ii) 1, 2, 3, ..., n

Let C1 and C2 represent the number of comparisons made for inputs (i) and (ii) respectively, which one has a greater value?

C1=C2. Performs in the worst case when input is sorted in either ascending order or descending order. quick sort will return equal number of comparisons for both given inputs.

19. Why analyze the expected running time of a randomized algorithm rather than it is worst-case running time?

    Analyze expected run time because it's more representative of typical time costs. Additionally, considering randomness in computation prevents adversarial input generation compared to analyzing all possible inputs..

20. Prove that the running time of QUICKSORT is O(n^2) when the array A is sorted in non-increasing order.

    The running time of Quicksort when the array A is sorted in nonincreasing order is indeed O(n^2). This is because, in the worst-case scenario, the partitioning routine produces one region with n - 1 elements and one region with 1 element. This results in a recursion tree with a height of O(log n), which gives a worst-case running time of O(n^2)