Elementary Data Structures and Algorithms

Binary Heaps

If the heap is stored in an array, it is stored in the standard way.
Arrays use zero-based indexing, unless otherwise indicated.
Assume integer division.
Always choose the best or most general answer, unless otherwise instructed.

Concept: heap shapes

1. In a heap, the upper bound on the number of leaves is:

   (A) $O(1)$
   (B) $O(n \log n)$
   (C) $O(n)$
   (D) $O(\log n)$

2. In a heap, the distance from the root to the furthest leaf is:

   (A) $\theta(n)$
   (B) $\theta(1)$
   (C) $\theta(\log n)$
   (D) $\theta(n \log n)$

3. In a heap, let d_f be the distance of the furthest leaf from the root and let d_c be the analogous distance of the closest leaf. What is $d_f - d_c$, at most?

   (A) 0
   (B) 1
   (C) 2
   (D) $\theta(\log n)$

4. What is the most number of nodes in a heap with a single child?

   (A) 0
   (B) 2
   (C) $\Theta(n)$
   (D) 1
   (E) $\Theta(\log n)$

5. What is the fewest number of nodes in a heap with a single child?

   (A) one per level
   (B) 0
   (C) 2
   (D) 1

6. **T** or **F**: There can be two or more nodes in a heap with exactly one child.

7. **T** or **F**: A heap can have no nodes with exactly one child.

8. **T** or **F**: All heaps are perfect trees.

9. **T** or **F**: No heaps are perfect trees.

10. **T** or **F**: All heaps are complete trees.

11. **T** or **F**: No heaps are complete trees.

12. **T** or **F**: A binary tree with one node must be a heap.

13. **T** or **F**: A binary tree with two nodes and with the root having the smallest value must be a min-heap.

14. **T** or **F**: If a node in a heap is a right child and has two children, then its sibling must also have two children.

15. **T** or **F**: If a node in a heap is a right child and has one child, then its sibling must also have one child.

Concept: heap ordering

16. In a min-heap, what is the relationship between a parent and its left child?

    (A) the parent has a larger value
    (B) the parent has a smaller value
    (C) there is no relationship between their values
    (D) the parent has the same value

17. In a min-heap, what is the relationship between a left child and its sibling?

    (A) both children cannot have the same value
    (B) there is no relationship between their values
    (C) the right child has a larger value
    (D) the left child has a smaller value

18. **T** or **F**: A binary tree with three nodes and with the root having the smallest value and two children must be a min heap.

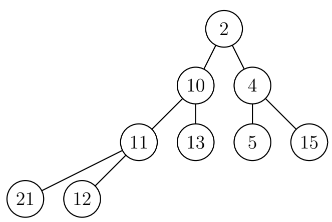
19. **T** or **F**: The largest value in a max-heap can be found at the root.

20. **T** or **F**: The largest value in a min-heap can be found at the root.

21. **T** or **F**: The largest value in a min-heap can be found at a leaf.

Concept: heaps stored in arrays

22. How would this heap be stored in an array?



    (A) `[2,10,11,21,12,13,4,5,15]`
    (B) `[2,10,4,11,13,5,15,21,12]`
    (C) `[21,11,12,10,13,2,5,4,15]`
    (D) `[2,4,5,10,11,12,13,15,21]`

23. Printing out the values in the array yield what kind of traversal of the heap?

    (A) post-order
    (B) level-order
    (C) pre-order
    (D) in-order

24. Suppose the heap has n values. The root of the heap can be found at which index?

    (A) 0
    (B) n-1
    (C) n
    (D) 1

25. Suppose the heap has n values. The left child of the root can be found at which index?

    (A) n
    (B) n-2
    (C) n-1
    (D) 2
    (E) 0
    (F) 1

26. Left children in a heap are stored at what kind of indices?

    (A) all odd
    (B) all even
    (C) all odd but one
    (D) all even but one
    (E) a roughly equal mix of odd and even

27. Right children in a heap are stored at what kind of indices?

(A) all even
(B) all even but one
(C) all odd but one
(D) a roughly equal mix of odd and even
(E) all odd

28. The formula for finding the left child of a node stored at index i is:

(A) $i * 2 + 1$
(B) $i * 2$
(C) $i * 2 - 1$
(D) $i * 2 + 2$

29. The formula for finding the right child of a node stored at index i is:

(A) $i * 2 + 2$
(B) $i * 2$
(C) $i * 2 + 1$
(D) $i * 2 - 1$

30. The formula for finding the parent of a node stored at index i is:

(A) $i/2$
(B) $(i + 2)/2$
(C) $(i + 1)/2$
(D) $(i - 1)/2$

31. If the array uses one-based indexing, the formula for finding the left child of a node stored at index i is:

(A) $i * 2 - 1$
(B) $i * 2 + 1$
(C) $i * 2 + 2$
(D) $i * 2$

32. If the array uses one-based indexing, the formula for finding the right child of a node stored at index i is:

(A) $i * 2 + 2$
(B) $i * 2$
(C) $i * 2 + 1$
(D) $i * 2 - 1$

33. If the array uses one-based indexing, the formula for finding the parent of a node stored at index i is:

(A) $(i - 1)/2$
(B) $(i + 2)/2$
(C) $i/2$
(D) $(i + 1)/2$

34. Consider a trinary heap stored in an array. The formula for finding the left child of a node stored at index i is:

(A) $i * 3 + 2$
(B) $i * 3 - 2$
(C) $i * 3$
(D) $i * 3 + 1$
(E) $i * 3 + 3$
(F) $i * 3 - 1$

35. Consider a trinary heap stored in an array. The formula for finding the middle child of a node stored at index i is:

(A) $i * 3 - 1$
(B) $i * 3$
(C) $i * 3 - 2$
(D) $i * 3 + 2$
(E) $i * 3 + 1$
(F) $i * 3 + 3$

36. Consider a trinary heap stored in an array. The formula for finding the right child of a node stored at index i is:

(A) $i * 3 + 2$
(B) $i * 3 - 1$
(C) $i * 3 - 2$
(D) $i * 3$
(E) $i * 3 + 1$
(F) $i * 3 + 3$

37. Consider a trinary heap stored in an array. The formula for finding the parent of a node stored at index i is:

(A) $i/3 - 1$
(B) $(i - 1)/3$
(C) $(i - 2)/3$
(D) $(i + 1)/3$
(E) $(i + 2)/3$
(F) $i/3 + 1$

Concept: *heap operations*

38. In a max-heap with no knowledge of the minimum value, the minimum value can be found in time:

    (A) $\theta(\log n)$
    (B) $\theta(1)$
    (C) $\theta(n \log n)$
    (D) $\theta(n)$

39. Suppose a min-heap with n values is stored in an array a. In the *extractMin* operation, which element immediatelyreplaces the root element (prior to this new root being sifted down).

    (A) the minimum of `a[1]` and `a[2]`
    (B) `a[1]`
    (C) `a[2]`
    (D) `a[n-1]`

40. The *findMin* operation in a min-heap takes how much time?

    (A) $\Theta(\log n)$
    (B) $\Theta(n \log n)$
    (C) $\Theta(n)$
    (D) $\Theta(1)$

41. The *extractMin* operation in a min-heap takes how much time?

    (A) $\Theta(\log n)$
    (B) $\Theta(n \log n)$
    (C) $\Theta(1)$
    (D) $\Theta(n)$

42. Merging two heaps of size n and m, $m < n$ takes how much time?

    (A) $\Theta(\log n * \log m)$
    (B) $\Theta(m \log n)$
    (C) $\Theta(n \log m)$
    (D) $\Theta(\log n + \log m)$
    (E) $\Theta(n + m)$
    (F) $\Theta(n * m)$

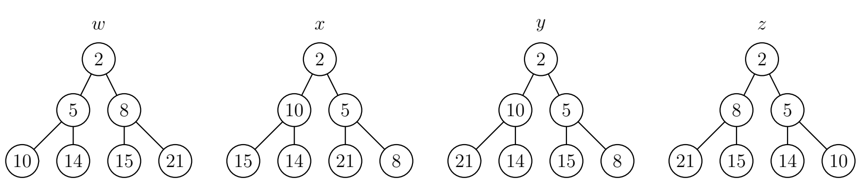
43. The *insert* operation takes how much time?

    (A) $\Theta(\log n)$
    (B) $\Theta(n)$
    (C) $\Theta(n \log n)$
    (D) $\Theta(1)$

44. Turning an unordered array into a heap takes how much time?

    (A) $\Theta(1)$
    (B) $\Theta(\log n)$
    (C) $\Theta(n)$
    (D) $\Theta(n \log n)$

45. Suppose the values 21, 15, 14, 10, 8, 5, and 2 are inserted, one after the other, into an empty *min*-heap. What does the resulting heap look like? Heap properties are maintained after every insertion.



    (A) w
    (B) y
    (C) x
    (D) z

46. Using the standard *buildHeap* operation to turn an unordered array into a *max*-heap, how many parent-child swaps are made if the initial unordered array is `[5,21,8,15,25,3,9]`?

    (A) 7
    (B) 4
    (C) 6
    (D) 3
    (E) 5
    (F) 2