1. What is the time complexity of inserting an element into a binary min-heap with 'n' elements?

**A) O(log n)**

B) O(n)

C) O(n log n)

D) O(1)

2. What will be the output of the following C++ code?

#include <iostream>

#include <vector>

#include <algorithm>

void heapify(std::vector<int>& heap, int idx) {

int largest = idx;

int left = 2 \* idx + 1;

int right = 2 \* idx + 2;

if (left < heap.size() && heap[left] > heap[largest])

largest = left;

if (right < heap.size() && heap[right] > heap[largest])

largest = right;

if (largest != idx) {

std::swap(heap[idx], heap[largest]);

heapify(heap, largest);

}

}

void insertToHeap(std::vector<int>& heap, int value) {

heap.push\_back(value);

int currentIdx = heap.size() - 1;

int parentIdx = (currentIdx - 1) / 2;

while (currentIdx > 0 && heap[parentIdx] < heap[currentIdx]) {

std::swap(heap[currentIdx], heap[parentIdx]);

currentIdx = parentIdx;

parentIdx = (currentIdx - 1) / 2;

}

}

int main() {

std::vector<int> maxHeap;

insertToHeap(maxHeap, 50);

insertToHeap(maxHeap, 30);

insertToHeap(maxHeap, 70);

insertToHeap(maxHeap, 10);

std::cout << maxHeap.front();

return 0;

}

a) 50

b) 30

**c) 70**

d) 10

3. In a binary max-heap, the new element to be inserted is placed at the \_\_\_\_\_ of the heap.

A) Root

B) Leftmost leaf

C) Rightmost leaf

**D) Bottom-right position**

4. What will be the output of the following C++ code?

#include <iostream>

#include <vector>

#include <algorithm>

void heapify(std::vector<int>& heap, int idx) {

int largest = idx;

int left = 2 \* idx + 1;

int right = 2 \* idx + 2;

if (left < heap.size() && heap[left] > heap[largest])

largest = left;

if (right < heap.size() && heap[right] > heap[largest])

largest = right;

if (largest != idx) {

std::swap(heap[idx], heap[largest]);

heapify(heap, largest);

}

}

void insertToHeap(std::vector<int>& heap, int value) {

heap.push\_back(value);

int currentIdx = heap.size() - 1;

int parentIdx = (currentIdx - 1) / 2;

while (currentIdx > 0 && heap[parentIdx] < heap[currentIdx]) {

std::swap(heap[currentIdx], heap[parentIdx]);

currentIdx = parentIdx;

parentIdx = (currentIdx - 1) / 2;

}

}

int main() {

std::vector<int> maxHeap = {80, 60, 50, 45, 30, 40, 20};

insertToHeap(maxHeap, 70);

for (int num : maxHeap) {

std::cout << num << " ";

}

return 0;

}

a) 80 60 50 45 30 40 20 70

b) 80 60 70 45 30 40 20 50

**c) 80 70 50 60 30 40 20 45**

d) 80 60 70 50 45 20 30 40

5. Which step is essential to maintain the heap property after inserting an element into a binary heap?

**A) Swapping the new element with its parent until the heap property is satisfied.**

B) Swapping the new element with its left child until the heap property is satisfied.

C) Swapping the new element with its right child until the heap property is satisfied.

D) No step is required; the heap property is automatically maintained.

6. If the initial min-heap is {5, 10, 15, 20}, what will be the min-heap after inserting the element 8?

a) {5, 10, 15, 20, 8}

**b) {5, 8, 10, 15, 20}**

c) {5, 8, 15, 10, 20}

d) {5, 8, 10, 20, 15}

7. When inserting an element into a binary max-heap, which operation is used to compare the new element with its parent?

**A) Greater than (>)**

B) Less than (<)

C) Greater than or equal to (>=)

D) Less than or equal to (<=)

8. If the initial min-heap is {1, 3, 5, 7}, what will be the min-heap after inserting the element 2?

a) {1, 3, 5, 7, 2}

b) {1, 2, 5, 7, 3}

c) {1, 2, 5, 3, 7}

**d) {1, 2, 3, 5, 7}**

9. In an array-based representation of a binary min-heap, the index of the parent of element 'i' is given by:

**A) (i - 1) / 2**

B) (i + 1) / 2

C) i / 2

D) 2i

10. If the initial input to the program is changed to:

minHeap.insert(5);

minHeap.insert(10);

minHeap.insert(20);

minHeap.insert(15);

What will be the output of the program?

**a) 5**

b) 10

c) 15

d) 20

11. While inserting a new element in a binary max-heap, if the new element violates the heap property, what action is taken?

A) The new element is removed from the heap.

B) The new element is placed at the root position.

**C) The new element is swapped with its largest child until the heap property is satisfied.**

D) The heap is restructured from scratch.

12. What will be the output of the program after inserting the elements 50, 30, 70, and 60 into an empty min-heap?

#include <iostream>

#include <vector>

#include <algorithm>

void insertIntoHeap(std::vector<int>& heap, int num) {

heap.push\_back(num);

std::push\_heap(heap.begin(), heap.end(), std::greater<int>());

}

int main() {

std::vector<int> minHeap;

insertIntoHeap(minHeap, 50);

insertIntoHeap(minHeap, 30);

insertIntoHeap(minHeap, 70);

insertIntoHeap(minHeap, 60);

for (int num : minHeap) {

std::cout << num << " ";

}

return 0;

}

**a) 30 50 70 60**

b) 70 50 30 60

c) 30 60 70 50

d) 30 50 60 70

13. Which type of binary heap allows duplicates among its elements?

A) Binary Max-Heap

B) Binary Min-Heap

**C) Both Binary Max-Heap and Binary Min-Heap**

D) Neither Binary Max-Heap nor Binary Min-Heap

14. After inserting elements 25, 10, 35, and 5 into an empty max-heap, what will be the value at the root node?

#include <iostream>

#include <vector>

#include <algorithm>

void insertIntoHeap(std::vector<int>& heap, int num) {

heap.push\_back(num);

std::push\_heap(heap.begin(), heap.end());

}

int main() {

std::vector<int> maxHeap;

insertIntoHeap(maxHeap, 25);

insertIntoHeap(maxHeap, 10);

insertIntoHeap(maxHeap, 35);

insertIntoHeap(maxHeap, 5);

std::cout << maxHeap.front();

return 0;

}

**a) 35**

b) 25

c) 10

d) 5

15. When inserting an element into a binary min-heap, the element is placed in such a way that it is smaller than or equal to its \_\_\_\_\_\_.

**A) Parent**

B) Left child

C) Right child

D) Both Parent and Children

16. What will be the output of the program after inserting the elements 50, 20, 70, and 10 into an empty min-heap?

#include <iostream>

#include <vector>

#include <algorithm>

void insertIntoHeap(std::vector<int>& heap, int num) {

heap.push\_back(num);

std::push\_heap(heap.begin(), heap.end(), std::greater<int>());

}

int main() {

std::vector<int> minHeap;

insertIntoHeap(minHeap, 50);

insertIntoHeap(minHeap, 20);

insertIntoHeap(minHeap, 70);

insertIntoHeap(minHeap, 10);

for (int num : minHeap) {

std::cout << num << " ";

}

return 0;

}

**a) 10 20 70 50**

b) 70 50 20 10

c) 10 70 50 20

d) 20 50 10 70

17. What is the worst-case time complexity to insert 'k' elements consecutively into a binary heap of size 'n' using the standard heap insertion algorithm?

**A) O(k log n)**

B) O(n log k)

C) O(k + n)

D) O(n^2)

18. Consider the following C++ program:

#include <iostream>

#include <vector>

void heapifyUp(std::vector<int>& heap, int index) {

int parent = (index - 1) / 2;

while (index > 0 && heap[parent] < heap[index]) {

std::swap(heap[parent], heap[index]);

index = parent;

parent = (index - 1) / 2;

}

}

void insertIntoHeap(std::vector<int>& heap, int value) {

heap.push\_back(value);

heapifyUp(heap, heap.size() - 1);

}

int main() {

std::vector<int> maxHeap;

insertIntoHeap(maxHeap, 20);

insertIntoHeap(maxHeap, 10);

insertIntoHeap(maxHeap, 30);

insertIntoHeap(maxHeap, 40);

for (int value : maxHeap) {

std::cout << value << " ";

}

return 0;

}

What is the output of the program?

a) 10 20 30 40

**b) 40 30 20 10**

c) 20 10 30 40

d) 40 20 30 10

19. In which data structure is the underlying implementation of a binary heap used for efficient insertion and deletion of elements?

A) Queue

B) Linked List

C) Stack

**D) Priority Queue**

20. insertIntoHeap(heap, 10);

insertIntoHeap(heap, 20);

insertIntoHeap(heap, 5);

insertIntoHeap(heap, 15);

insertIntoHeap(heap, 30);

After the first call to insertIntoHeap(), what is the value of heap.size()?

a) 0

**b) 1**

c) 2

d) 3

21. A \_\_\_\_\_\_\_\_ is a special Tree-based data structure in which the tree is a complete binary tree.?

a) Graph

**b) Heap**

c) List

d) Stack

22. Given the code, choose the correct option that is consistent with the code. (Here A is the heap)

build(A,i)

left-> 2\*i

right->2\*i +1

temp- > i

if(left<= heap\_length[A] ans A[left] >A[temp])

temp -> left

if (right = heap\_length[A] and A[right] > A[temp])

temp->right

if temp!= i

swap(A[i],A[temp])

build(A,temp)

**a) It is the build function of max heap**

b) It is the build function of min heap

c) It is general build function of any heap

d) It is used to search element in any heap

23. How many type of heap are there?

**a) 2**

b) 3

c) 4

d) 5

24.State the complexity of algorithm given below.

int function(vector<int> arr)

int len=arr.length();

if(len==0)

return;

temp=arr[len-1];

arr.pop\_back();

return temp;

a) o(n)

b) O(logn)

**c) O(1)**

d) O(n logn)

25.In which heap the root node must be greatest among the keys present at all of it’s children?

a) min-heap

**b) max-heap**

c) Both A and B

d) None of the above

26. For construction of a binary heap with property that parent node has value less than child node. In reference to that which line is incorrect. Line indexed from 1.

add(int k)

{

heap\_size++;

int i = heap\_size - 1;

harr[i] = k;

while (i != 0 && harr[parent(i)] < harr[i])

{

swap(&harr[i], &harr[parent(i)]);

i = parent(i);

}

}

a) Line – 3

b) Line – 5

**c) Line – 6**

d) Line – 7

27.What is the complexity of adding an element to the heap?

a) O(log n)

b) O(log h)

c) O(h)

**d) Both A and C**

28. What does this pseudo\_code return ?

int myfun(heap\_arr[])

{

int mini=INF;

for(int i=0;i<tot\_node;i++)

mini=min(mini,heap\_arr)

return mini;

}

a) Last added element to heap

b) First element added to heap

**c) Root of the heap**

d) Leftmost node of the heap

29. Heap can be used as \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**a) Priority queue**

b) Stack

c) A decreasing order array

d) Normal Array

30. What is wrong with the following code of insertion in fibonacci heap.

Choose the correct option

FIB-INSERT(H, x)

degree[x]= 0

p[x]= NIL

child[x] =NIL

left[x] =x

right[x] =x

mark[x] =FALSE

concatenate the root list containing x with root list H

if min[H] = NIL or key[x] > key[min[H]]

then min[H]= x

n[H]= n[H] + 1

a) Line -11

b) Line -3

**c) Line 9**

d) Line 7

31. An array consists of n elements. We want to create a heap using the elements. The time complexity of building a heap will be in order of

a) O(n\*n\*logn)

**b) O(n\*logn)**

c) O(n\*n)

d) O(n \*logn \*logn)

32. What will be the order of new heap created after union of heap H1 and H2 when created by the following code. Initially both are of the order n.

FIB\_UNION(H1,H2)

{

H =MAKE\_HEAP()

min[H]= min[H1]

concatenate the root list of H2 with the root list of H

if (min[H1] = NIL) or (min[H2]!= NIL and min[H2] < min[H1])

then min[H] = min[H2]

n[H]= n[H1] + n[H2]

free the objects H1 and H2

return H

}

**a) n+1**

b) n+n/2

c) nlogn

d) 2\*n

33. Min heap can be used to implement selection sort.

a) True

**b) False**

34. Which one of the following array elements represents a binary min heap?

a) 12 10 8 25 14 17

**b) 8 10 12 25 14 17**

c) 25 17 14 12 10 8

d) 14 17 25 10 12 8

35. Given an array of element 5, 7, 9, 1, 3, 10, 8, 4. Which of the following is the correct sequences of elements after inserting all the elements in a min-heap?

**a) 1,3,4,5,7,8,9,10**

b) 1,4,3,9,8,5,7,10

c) 1,3,4,5,8,7,9,10

d) 1,3,7,4,8,5,9,10

36. What is the amortized cost per operation of a skew heap?

a) O(N)

b) O(N log N)

c) O(N2)

**d) O(log N)**

37. Suppose we are sorting an array of eight integers using heapsort, and we have just finished some heapify (either maxheapify or minheapify) operations. The array now looks like this: 16 14 15 10 12 27 28 How many heapify operations have been performed on root of heap?

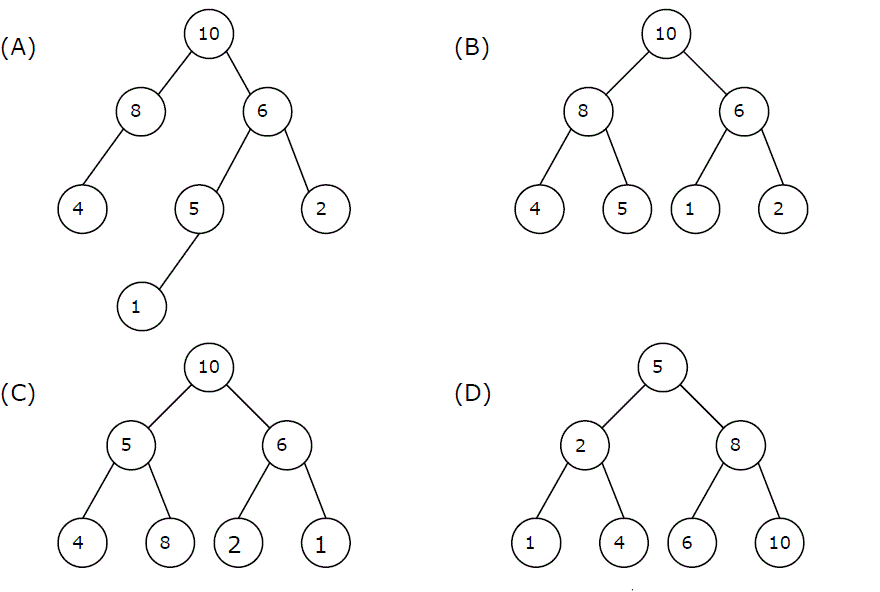
a) 1

**b) 2**

c) 3 or 4

d) 5 or 6

38. A max-heap is a heap where the value of each parent is greater than or equal to the values of its children. Which of the following is a max-heap?



a) A

**b) B**

c) C

d) D

39. A priority queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is: 10, 8, 5, 3, 2. Two new elements 1 and 7 are inserted into the heap in that order. The level-order traversal of the heap after the insertion of the elements is:

**a) 10, 8, 7, 3, 2, 1, 5**

b) 10, 8, 7, 2, 3, 1, 5

c) 10, 8, 7, 1, 2, 3, 5

d) 10, 8, 7, 5, 3, 2, 1

40. Consider a max heap, represented by the array: 40, 30, 20, 10, 15, 16, 17, 8, 4. Now consider that a value 35 is inserted into this heap. After insertion, the new heap is

a) 40, 30, 20, 10, 15, 16, 17, 8, 4, 35

**b) 40, 35, 20, 10, 30, 16, 17, 8, 4, 15**

c) 40, 30, 20, 10, 35, 16, 17, 8, 4, 15

d) 40, 35, 20, 10, 15, 16, 17, 8, 4, 30

41. Heap Sort is a sorting algorithm based on the concept of a \_\_\_\_\_\_\_.

a) Linked list

**b) Priority queue**

c) Hash table

d) Binary search tree

42. What will be the output of the following code snippet?

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

int n = sizeof(arr) / sizeof(arr[0]);

make\_heap(arr, arr + n);

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

a) 13 12 7 5 6 11

**b) 13 11 12 5 6 7**

c) 13 11 12 6 5 7

d) 7 6 5 13 11 12

43. Which of the following statements is true about Heap Sort?

**a) It has a worst-case time complexity of O(n log n).**

b) It is an in-place sorting algorithm.

c) It is a stable sorting algorithm.

d) It is suitable for sorting large linked lists.

44. What will be the output of the following code snippet?

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

int n = sizeof(arr) / sizeof(arr[0]);

make\_heap(arr, arr + n);

sort\_heap(arr, arr + n);

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

**a) 5 6 7 11 12 13**

b) 13 12 11 7 6 5

c) 5 6 7 12 11 13

d) 13 11 12 7 6 5

45. Heap Sort works by first building a \_\_\_\_\_\_\_ from the given array.

a) Binary search tree

b) Min-heap

**c) Max-heap**

d) AVL tree

46. What will be the output of the following code snippet?

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

int n = sizeof(arr) / sizeof(arr[0]);

make\_heap(arr, arr + n);

pop\_heap(arr, arr + n);

n--;

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

**a) 12 11 7 5 6**

b) 13 11 12 7 6

c) 12 11 6 5 7

d) 11 5 6 7 12

47. During the heapification process in Heap Sort, the maximum element is moved to the \_\_\_\_\_\_\_.

**a) Root of the heap**

b) Rightmost leaf node

c) Leftmost leaf node

d) Middle element of the heap

48. What will be the output of the following code snippet?

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

int arr[] = {12, 11, 13, 5, 6, 7};

int n = sizeof(arr) / sizeof(arr[0]);

make\_heap(arr, arr + n);

pop\_heap(arr, arr + n);

n--;

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

sort\_heap(arr, arr + n);

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

return 0;

}

A) 11 6 7 5

5 6 7 11

B) 13 11 12 7

7 11 12 13

**C) 12 11 7 5 6**

**5 6 7 11 12**

D) 11 5 6 7

5 6 7 11

49. What is the time complexity of Heap Sort for sorting n elements in the worst case?

a) O(n)

**b) O(n log n)**

c) O(log n)

d) O(n^2)

50. Choose the correct option to fill? X so that the code given below implements the Heap sort.

#include <stdio.h>

void heapify(int arr[], int n, int i)

{

int largest = i; // Initialize largest as root

int l = 2\*i + 1; // left = 2\*i + 1

int r = 2\*i + 2; // right = 2\*i + 2

if (l < n && arr[l] > arr[largest])

largest = l;

if (r < n && arr[r] > arr[largest])

largest = r;

if (largest != i)

{

swap(arr[i], arr[largest]);

heapify(arr, n, largest);

}

}

void heapSort(int arr[], int n)

{

for (int i = n / 2 - 1; i >= 0; i--)

heapify(arr, n, i);

for (int i=n-1; i>=0; i--)

{

X;

heapify(arr, i, 0);

}

}

void printArray(int arr[], int n)

{

for (int i=0; i<n; ++i)

printf(“%d”,arr[i]);

printf(“\n”);

}

int main()

{

int arr[] = {12, 11, 13, 5, 6, 7};

int n = sizeof(arr)/sizeof(arr[0]);

heapSort(arr, n);

printf(“Sorted array is \n");

printArray(arr, n);

}

a) swap(arr[0], arr[n])

b) swap(arr[i], arr[n])

**c) swap(arr[0], arr[i])**

d) swap(arr[i], arr[2\*i])

51. In a Max-heap, the element with the highest value is always located at the \_\_\_\_\_\_\_.

**a) Root of the heap**

b) Rightmost leaf node

c) Leftmost leaf node

d) Middle element of the heap

52. What is the output of the following code snippet?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(10);

pq.push(20);

pq.push(5);

pq.pop();

cout << pq.top();

return 0;

}

a) 5

**b) 10**

c) 20

d) Compilation Error

53. When performing a deletion in a Max-heap, after removing the root element, which element replaces it to maintain the heap property?

a) Largest element from the left subtree

b) Smallest element from the right subtree

**c) Largest element from the right subtree**

d) Smallest element from the left subtree

54. What will be the output of the following C++ code?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(50);

pq.push(20);

pq.push(30);

while (!pq.empty()) {

cout << pq.top() << " ";

pq.pop();

}

return 0;

}

a) 20 30 50

**b) 50 30 20**

c) 30 20 50

d) Compilation Error

55. After deleting an element from a Max-heap, the remaining elements need to be rearranged to maintain the \_\_\_\_\_\_\_.

a) Binary search tree property

b) Min-heap property

**c) Max-heap property**

d) Balanced tree property

56. What is the output of the following C++ code?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(10);

pq.push(15);

pq.push(5);

pq.push(25);

pq.pop();

cout << pq.top();

return 0;

}

a) 5

b) 10

**c) 15**

d) 25

57. What is the time complexity of deleting an element from a Max-heap with n elements?

**a) O(log n)**

b) O(n)

c) O(n log n)

d) O(1)

58. What will be the output of the following C++ code?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(30);

pq.push(10);

pq.push(20);

cout << pq.top() << " ";

pq.pop();

cout << pq.top();

return 0;

}

a) 10 20

b) 20 10

**c) 30 20**

d) Compilation Error

59. Heap Deletion is commonly used to efficiently implement the \_\_\_\_\_\_\_ operation in priority queues.

a) Insertion

b) Deletion

**c) Peek**

d) Search

60. What is the output of the following code snippet?

#include <iostream>

#include <queue>

using namespace std;

int main() {

priority\_queue<int> pq;

pq.push(5);

pq.push(10);

pq.push(15);

pq.push(10);

while (!pq.empty()) {

cout << pq.top() << " ";

pq.pop();

}

return 0;

}

a) 5 10 10 15

**b) 15 10 10 5**

c) 5 10 15 10

d) Compilation Error