Binary Search MCQs

Part 1: Definition-Based Questions

Q1. [Definition] What is the time complexity of binary search on a sorted array?
A. O(n)
B. O(log n)
C. O(n log n)
D. O(1)
Answer: B
Explanation: Binary search reduces the search space by half each time, leading to O(log n) time complexity.

Q2. [Definition] What is the required condition for applying binary search on an array?
A. Array must be unsorted
B. Array should be sorted in descending order
C. Array should be sorted
D. Array must contain only positive integers
Answer: C
Explanation: Binary search works only on sorted arrays, either in ascending or descending order.

Q3. [Definition] Which technique is used in binary search?
A. Divide and Conquer
B. Dynamic Programming
C. Greedy
D. Backtracking
Answer: A
Explanation: Binary search divides the array and searches in the half where the target may lie.

Q4. [Definition] In binary search, what is the typical role of the 'mid' index?
A. Checks if the array is sorted
B. Acts as a pointer to iterate from start
C. Splits the search space
D. Stores the target value
Answer: C
Explanation: The mid index divides the array into two halves to decide the direction of the search.

Q5. [Definition] What will happen if binary search is applied on an unsorted array?
A. It will give correct result
B. Time complexity becomes O(n)
C. It may return incorrect results
D. It will always return -1
Answer: C
Explanation: Binary search assumes sorted order; applying it on unsorted data may give wrong results.

Part 2: Moderate Difficulty (Error Identification & Debugging)

Q6. [Moderate] What will be the output of the following code?

```
int[] arr = {1, 3, 5, 7, 9};
int low = 0, high = arr.length - 1, target = 7;
while (low <= high) {
    int mid = (low + high) / 2;
    if (arr[mid] == target) {
        System.out.println("Found at " + mid);
        break;
    } else if (arr[mid] < target) {
        low = mid + 1;
    } else {
        high = mid - 1;
    }
}
```

A. Found at 2
B. Found at 3
C. Found at 1
D. Not Found
Answer: B
Explanation: Binary search finds 7 at index 3 in the sorted array.

Q7. [Moderate] What is the logical error in this binary search implementation?

```
int mid = (low + high) / 2;
```

A. Division by zero
B. May cause overflow for large values
C. Incorrect comparison
D. Infinite loop
Answer: B
Explanation: When low and high are large, (low + high) can overflow. Prefer low + (high - low)/2.

Q8. [Moderate] What happens if low = 0, high = arr.length, instead of high = arr.length - 1?
A. Search will skip last element
B. Search will be incorrect
C. IndexOutOfBoundsException
D. Works fine
Answer: C
Explanation: arr.length is out of bounds (last valid index is arr.length - 1).

Q9. [Moderate] Which change will make binary search work on descending sorted arrays?
A. Use linear search instead
B. Swap low and high
C. Reverse array
D. Reverse comparison logic
Answer: D
Explanation: Modify comparison: if arr[mid] < target → go left, else right.

Q10. [Moderate] What's the output of this binary search if target is not found?

```
int[] arr = {2, 4, 6, 8, 10};
int low = 0, high = arr.length - 1, target = 5;
boolean found = false;
while (low <= high) {
    int mid = (low + high) / 2;
    if (arr[mid] == target) {
        found = true;
        break;
    } else if (arr[mid] < target) {
        low = mid + 1;
    } else {
        high = mid - 1;
    }
}
System.out.println(found);
```

A. true
B. false
C. 1
D. Error
Answer: B
Explanation: 5 is not in the array; the loop completes without finding, so found remains false.

Part 3: Difficult (Code Output / Scenario-Based)
Q11. [Difficult] What will this code output for arr = {1,2,3,4,5} and target = 6?

```
int[] arr = {1, 2, 3, 4, 5};
int low = 0, high = arr.length - 1;
while (low <= high) {
    int mid = (low + high) / 2;
    if (arr[mid] == 6) {
        System.out.println("Found");
        return;
    } else if (arr[mid] < 6) {
        low = mid + 1;
    } else {
        high = mid - 1;
    }
}
System.out.println("Not Found");
```

A. Found
B. Not Found
C. Error
D. Infinite loop
Answer: B
Explanation: 6 is not in the array, so loop ends and "Not Found" is printed.

Q12. [Difficult] What does this recursive binary search return for target = 7?

```
int binarySearch(int[] arr, int low, int high, int target) {
    if (low > high) return -1;
    int mid = low + (high - low) / 2;
    if (arr[mid] == target) return mid;
    else if (arr[mid] > target)
        return binarySearch(arr, low, mid - 1, target);
    else
        return binarySearch(arr, mid + 1, high, target);
}
```
Input: arr = {1, 3, 5, 7, 9}, low = 0, high = 4
A. 3
B. 2
C. 4
D. -1
Answer: A
Explanation: Recursive binary search finds target 7 at index 3.

Q13. [Difficult] How many times will mid be computed in binary search on an array of size 1024?
A. 10
B. 11
C. 9
D. 8
Answer: B
Explanation: In worst-case, binary search on n items takes $\lceil \log_2(n + 1) \rceil \rightarrow \log_2(1025) \approx 10.01 \Rightarrow 11$ steps.

Q14. [Difficult] What is returned by this binary search if target = 8?
```
int[] arr = {2, 4, 6, 8, 10};
int low = 0, high = arr.length - 1;
int res = -1;
while (low <= high) {
    int mid = low + (high - low) / 2;
    if (arr[mid] == 8) {
        res = mid;
        break;
    } else if (arr[mid] < 8) {
        low = mid + 1;
    } else {
        high = mid - 1;
    }
}
System.out.println(res);
```
A. 2
B. 3
C. 4
D. -1
Answer: B
Explanation: 8 is found at index 3, so res becomes 3.

Q15. [Difficult] Which scenario can cause infinite loop in binary search?
A. Forgetting to update low or high
B. Using incorrect mid formula
C. Using while (low < high) instead of (low <= high)
D. All of the above

Answer: D
Explanation: All these mistakes can lead to infinite loops by not narrowing the search space.

Part 1: Definition-Based Questions
Q16. [Definition] Which of the following problems can be solved using binary search?
A. Finding an element in a sorted array
B. Finding square root of a number
C. Finding smallest element satisfying a condition
D. All of the above
Answer: D
Explanation: Binary search is versatile and can solve problems involving ordered structure or monotonicity.

Q17. [Definition] What is the main advantage of binary search over linear search?
A. It can handle unsorted data
B. It's faster for small arrays
C. It requires less memory
D. It reduces time complexity from O(n) to O(log n)
Answer: D
Explanation: Binary search reduces the number of comparisons by halving the array in each step.

Q18. [Definition] What happens when the array size is 1 in binary search and the target does not match?
A. Infinite loop
B. ArrayIndexOutOfBoundsException
C. The condition low <= high fails
D. The array is split further
Answer: C
Explanation: If target doesn't match, low > high after one iteration, ending the loop.

Part 2: Moderate Difficulty
Q19. [Moderate] Identify the mistake in this code snippet:
```
int[] arr = {1, 3, 5, 7, 9};
int low = 0, high = arr.length - 1;
while (low < high) {
    int mid = (low + high) / 2;
    if (arr[mid] == target) return mid;
    else if (arr[mid] < target) low = mid + 1;
    else high = mid - 1;
}
```
A. Incorrect mid calculation
B. Missing case when low == high
C. Incorrect return statement
D. Wrong comparison
Answer: B
Explanation: The condition low < high skips the case where only one element is left (low == high).

Q20. [Moderate] What is the result of binary search when applied to an empty array?
```
int[] arr = {};
int target = 5;
```

A. Index 0
B. Index -1
C. Exception
D. Not found
Answer: D
Explanation: An empty array has no elements to search; function should return "not found" or -1.

Q21. [Moderate] Which of the following mid formulas is safest against integer overflow?
A. (low + high) / 2
B. low + (high - low) / 2
C. (low - high) / 2
D. high / 2
Answer: B
Explanation: This avoids overflow by subtracting before adding.

Part 3: Difficult (Scenario-Based / Applications)
Q22. [Difficult] Which of the following problems use binary search on the answer space?
A. Allocate Minimum Pages
B. Aggressive Cows
C. Koko Eating Bananas
D. All of the above
Answer: D
Explanation: These are classic binary search problems where we apply BS on a numeric answer range.

Q23. [Difficult] What will be the output of this code if target = 4?
```
int[] arr = {1, 3, 4, 4, 4, 5, 6};
int low = 0, high = arr.length - 1, res = -1;
while (low <= high) {
    int mid = low + (high - low) / 2;
    if (arr[mid] == 4) {
        res = mid;
        high = mid - 1;
    } else if (arr[mid] < 4) {
        low = mid + 1;
    } else {
        high = mid - 1;
    }
}
System.out.println(res);
```
A. 4
B. 2
C. 5
D. 3
Answer: B
Explanation: This code finds the **first occurrence** of 4, which is at index 2.

Q24. [Difficult] What will be the output if we modify the above to find the last occurrence?
Change:

high = mid - 1;
To:
low = mid + 1;
A. 3
B. 4
C. 5
D. 2
Answer: C
Explanation: The modified logic tracks the last index where target is found, which is at index 5.

Q25. [Difficult] Binary search is used in which of the following Leetcode/CP patterns?
A. Search in Rotated Sorted Array
B. Median of Two Sorted Arrays
C. Find Peak Element
D. All of the above
Answer: D
Explanation: All listed problems are solved efficiently with variations of binary search.

Q26. [Difficult] Rahul is solving a problem where he is given a sorted array of book pages. He must distribute the books among m students such that the maximum number of pages assigned to a student is minimized. Which technique should he use?
A. Linear Search
B. Binary Search on pages
C. Greedy Backtracking
D. Sliding Window
Answer: B
Explanation: This is the classic "Book Allocation" problem where binary search is applied on the answer space of page counts.

Q27. [Difficult] Sneha is optimizing a warehouse where she has to find the minimum weight capacity of a truck that can ship n packages in d days. The weights are given in an array. Which algorithm can help her?
A. Greedy
B. Linear Scan
C. Binary Search on answer
D. Two Pointer
Answer: C
Explanation: Sneha must binary search the minimum feasible truck capacity over the weight range.

Q28. [Difficult] A scientist is measuring temperature data from sensors. He needs to find the first temperature that crossed 100°C in a sorted log. What is the best way to find this efficiently?
A. Linear Scan
B. Binary Search
C. Prefix Sum
D. Hashing
Answer: B
Explanation: Since the log is sorted, binary search efficiently finds the **first index** where temperature > 100.

Q29. [Moderate] In a video game, the player must find the highest level they can clear with their current power. The canClear(level) API returns true if they can clear the level. Levels are sorted by difficulty. What search method will find the answer in the fewest steps?
A. Ternary Search
B. Linear Search
C. Modified Binary Search
D. DFS
Answer: C
Explanation: Binary search on levels using canClear() reduces unnecessary checks and finds the max feasible level.

Q30. [Moderate] Aarav is developing a mobile app to match passengers and drivers. He wants to find the closest available driver within a sorted list of distances. What approach should he use?
A. Queue
B. Binary Search
C. Heap
D. Recursion
Answer: B
Explanation: Since distances are sorted, binary search helps find the closest or nearest value.

Q31. [Difficult] A startup wants to test how fast their server can respond under load. They know the response time increases with the number of users. They want to find the maximum number of users for which the response time stays under 200ms. Which approach helps best?
A. Linear Scan on response time
B. Hashing
C. Binary Search on user count
D. Graph Traversal
Answer: C
Explanation: Binary search on the user range efficiently finds the threshold limit.

Q32. [Difficult] A company has n metal rods of different lengths and wants to cut them into k equal pieces of maximum possible length. What is the best way to solve this?
A. Backtracking
B. Binary Search on rod lengths
C. Stack
D. Divide and Conquer
Answer: B
Explanation: This is a classic case of binary search on the answer space: the max length of each piece.

Q33. [Moderate] In a mystery game, a player is trying to unlock a safe by guessing a number between 1 and 1000. Each wrong guess gives a hint: "higher" or "lower". Which algorithm is he unknowingly using?
A. Depth First Search
B. Breadth First Search
C. Binary Search
D. Linear Search
Answer: C
Explanation: The feedback reduces the range by half each time — exactly how binary search works.

Q34. [Moderate] A robot is searching for a charging station in a line of stations, sorted by distance. The battery is limited. It wants to stop at the furthest reachable station without running out. What strategy will minimize power checks?
A. Brute Force
B. Binary Search
C. Greedy Scan
D. BFS
Answer: B
Explanation: Binary search on index of stations optimizes finding the farthest reachable point within battery limit.

Q35. [Difficult] A video streaming platform wants to determine the minimum bitrate required to stream a video without buffering. They can simulate playback at any bitrate. What algorithm should they use to minimize test runs?
A. Greedy
B. Binary Search on bitrate values
C. Linear Search
D. Sliding Window
Answer: B
Explanation: This is a classic case of binary search on continuous values (float/int bitrate range).
Q36. [Difficult] A delivery company needs to allocate parcels to k trucks such that no truck carries more than a certain max weight. The goal is to minimize this maximum weight. What algorithmic approach should be used?
A. Two Pointers
B. Binary Search on Weight
C. Greedy + Recursion
D. Hash Map
Answer: B
Explanation: Binary search on the weight capacity helps find the smallest feasible "maximum load" across trucks.

Q37. [Difficult] A team is optimizing video encoding. Given a function canCompress(size), they want to find the smallest video size that can be compressed within 2 seconds. Sizes are between 1MB and 10GB. Which technique is best?
A. Hash Table
B. Binary Search on size
C. Graph Traversal
D. Sliding Window
Answer: B
Explanation: Binary search over possible sizes efficiently finds the smallest value satisfying canCompress.

Q38. [Difficult] Riya is developing a security system. It records access times in a sorted log. She needs to quickly check if any access occurred at exactly 14:32:00. What should she use?
A. Brute Force
B. Binary Search on time log
C. Stack
D. DFS

Answer: B
Explanation: Binary search on a sorted array of timestamps gives O(log n) lookup for exact matches.

Q39. [Moderate] A fintech app needs to approve loans if a user's credit score exceeds a threshold. The list of scores is sorted. To quickly determine the first user eligible, what should the app use?
A. Two Pointers
B. Linear Search
C. Binary Search for lower bound
D. Recursion
Answer: C
Explanation: Use binary search to find the first index where score >= threshold (lower bound search).

Q40. [Moderate] A factory must produce n items using machines with different speeds. They want to know the minimum time required to finish production. What DSA approach helps?
A. Priority Queue
B. Binary Search on Time
C. Dynamic Programming
D. DFS
Answer: B
Explanation: Binary search on time helps simulate whether all machines can meet the target in a given time.

Q41. [Difficult] An e-commerce platform stores the number of daily users in a sorted array. The team wants to find the first day where the number of users exceeded 1 million. What algorithm should they apply?
A. Hash Map
B. Linear Search
C. Binary Search
D. BFS
Answer: C
Explanation: Binary search is efficient to find the earliest index where a condition is satisfied.

Q42. [Moderate] In a social media app, the backend logs post likes in increasing order of time. Given a target number of likes, they want to find when the post reached that number for the first time. Best approach?
A. Hash Set
B. Binary Search on likes[]
C. Stack
D. Sorting
Answer: B
Explanation: Since data is sorted over time, binary search helps find the point when likes cross a threshold.

Q43. [Difficult] A video game boss has n health points. You can deal variable damage per second and want to finish the boss in minimum time. You can test if a speed works using canDefeat(dps). How to find minimum DPS needed?
A. Brute Force
B. Greedy
C. Binary Search on DPS
D. Sliding Window

Answer: C
Explanation: Binary search on numeric values (damage/speed) efficiently finds the minimal value that satisfies canDefeat.

Q44. [Difficult] A company is analyzing customer churn. They want to find the minimum number of ads a user needs to see before purchasing a product. Each user's behavior is binary searchable. Best algorithm?
A. Graph Search
B. Binary Search over Ad Views
C. Queue
D. Hashing
Answer: B
Explanation: Searching over the number of ads viewed and applying buysAfter(k) function can be done with binary search.

Q45. [Difficult] In a coding competition, scores are sorted. The team wants to identify the last participant who qualified (score ≥ 100). What's the ideal approach?
A. Sort and pick
B. Binary Search for upper bound
C. Linear Scan from end
D. Heap
Answer: B
Explanation: Use binary search to find the **last** index with score ≥ 100 (upper bound logic).
Q46. [Definition] What does "lower bound" mean in binary search context?
A. First element greater than key
B. Smallest index where value ≥ key
C. Largest index where value ≤ key
D. Index of minimum element
Answer: B
Explanation: Lower bound is the first position where the element is ≥ key.

Q47. [Definition] What does "upper bound" mean in binary search?
A. Last element equal to key
B. Index of highest value
C. First index where element > key
D. Index of key if present
Answer: C
Explanation: Upper bound returns the first index with value > key.

Q48. [Definition] Binary Search is best applied when:
A. Array is unsorted
B. Array contains duplicates
C. Array is sorted
D. Array is reversed
Answer: C
Explanation: Binary search requires sorted input for correctness.

Q49. [Definition] What is the worst-case time complexity of binary search?
A. O(n)
B. O(log n)
C. O(n log n)
D. O(1)
Answer: B
Explanation: Binary search halves the range each time.

Q50. [Definition] Which data structure is commonly used with lower/upper bounds?
A. Stack
B. Queue
C. Sorted Array
D. Graph
Answer: C
Explanation: Sorted arrays allow efficient binary searching for bounds.

◆ Part 2: Moderate – Code Understanding

Q51. [Moderate] Given arr = [1, 3, 3, 5, 7], what is the lower bound of 3?
A. 1
B. 2
C. 3
D. 4
Answer: A
Explanation: First index where arr[i] ≥ 3 is index 1.

Q52. [Moderate] For the same array, what is the upper bound of 3?
A. 1
B. 2
C. 3
D. 4
Answer: C
Explanation: Upper bound is index of first element > 3 → index 3 (element 5).

Q56 [Moderate] What does this code return for target = 6?
```
int lowerBound(int[] arr, int target) {
    int low = 0, high = arr.length;
    while (low < high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] < target) low = mid + 1;
        else high = mid;
    }
    return low;
}
```
A. Index of first element ≥ 6
B. Index of first element == 6
C. Index of last element < 6

D. Not found
Answer: A
Explanation: Lower bound returns index of first element ≥ target.

Q57. [Moderate] What's the base condition used to stop binary search loop in lower bound?
A. low <= high
B. low != high
C. low < high
D. low == mid
Answer: C
Explanation: Standard lower bound loop runs while low < high.

Q58. [Moderate] What's the output for this input: arr = [1, 2, 2, 2, 3], target = 2, using lowerBound?
A. 1
B. 2
C. 3
D. 0
Answer: A
Explanation: First 2 appears at index 1.

Q59. [Moderate] What's the output for the same array with upperBound for 2?
A. 3
B. 4
C. 5
D. 2
Answer: B
Explanation: First element > 2 is at index 4.

◆ Part 3: First/Last Occurrence

Q60. [Moderate] Which condition helps to find the first occurrence of a target in a sorted array?
A. Move left on match
B. Move right on match
C. Return mid
D. Use upper bound
Answer: A
Explanation: Keep searching left after a match to find first occurrence.

Q61. [Moderate] What condition helps find the last occurrence?
A. Stop on first match
B. Move left on match
C. Move right on match
D. Sort the array
Answer: C
Explanation: You continue to right even after a match.

Q62. [Difficult] What is the time complexity of finding both first and last occurrence of an element using binary search?
A. O(n)
B. O(log n)
C. O(n log n)
D. O(1)
Answer: B
Explanation: Each call (first and last) takes O(log n) independently.

Q63. [Difficult] What is returned by this binary search variation for first occurrence of 4 in [1,2,4,4,4,5,6]?
if (arr[mid] >= target) high = mid;
else low = mid + 1;
A. 2
B. 3
C. 4
D. 5
Answer: A
Explanation: First 4 appears at index 2.

Q64. [Difficult] How do you modify binary search to return last occurrence of an element?
A. Use arr[mid] <= target
B. Use arr[mid] >= target
C. Use arr[mid] == target only
D. Reverse the array
Answer: A
Explanation: Continue moving right to find last match.

Q67. [Difficult] For arr = [1,2,3,3,3,3,4,5], what are first and last occurrences of 3?
A. [2, 4]
B. [2, 5]
C. [3, 6]
D. [1, 4]
Answer: B
Explanation: First 3 at index 2, last 3 at index 5.

Q68. [Difficult] What will upperBound(arr, 5) return for arr = [1,2,4,5,5,5,6,7]?
A. 4
B. 6
C. 5
D. 7
Answer: B
Explanation: First index > 5 is index 6 (element = 6).

Q69. [Difficult] In a sorted array, when target is not found, what does lower bound give?
A. -1
B. The last index
C. Index where it can be inserted

D. Null
Answer: C
Explanation: It returns the index to insert the element while maintaining order.

Q70. [Difficult] What is the difference in implementation between lower and upper bound?
A. Loop condition
B. Mid calculation
C. Comparison operator (< vs <=)
D. No difference
Answer: C
Explanation: Lower uses <, upper uses <= to adjust the range.

◆ **Part 4: Scenario-Based**

Q71. [Difficult] In a sorted array with duplicates, which function returns the count of occurrences of a target?
A. upper_bound - lower_bound
B. mid value
C. max - min
D. binarySearch(target)
Answer: A
Explanation: Difference between upper and lower bound gives count.

Q72. [Difficult] What will be the count of 7 in arr = [5,6,7,7,7,8,9] using bounds?
A. 2
B. 3
C. 4
D. 0
Answer: B
Explanation: lowerBound = 2, upperBound = 5 → 5 - 2 = 3

Q73. [Difficult] Why is lower_bound preferred over indexOf() in sorted arrays with duplicates?
A. Faster
B. Works for non-existing keys
C. Supports insertion
D. All of the above
Answer: D
Explanation: Lower bound is binary-search based, handles duplicates and insertions efficiently.

Q74. [Difficult] What is the space complexity of standard lower bound implementation?
A. O(log n)
B. O(n)
C. O(1)
D. O(n log n)
Answer: C
Explanation: Binary search is in-place, only uses constant extra space.

Q75. [Difficult] In Java, which class provides binarySearch() for primitive arrays?
A. Collections
B. Arrays
C. HashMap
D. TreeSet
Answer: B
Explanation: Arrays.binarySearch() in java.util.Arrays class supports primitive arrays.

Q76. [Definition] What is the main challenge in binary search on a rotated sorted array?
A. Array is unsorted
B. Midpoint does not guarantee sorted halves
C. No duplicates allowed
D. Elements are in reverse
Answer: B
Explanation: After rotation, only one half remains sorted; identifying it is key.

Q77. [Moderate] What is the worst-case time complexity of searching in a rotated sorted array (no duplicates)?
A. O(n)
B. O(log n)
C. O(n log n)
D. O(1)
Answer: B
Explanation: Even with rotation, modified binary search runs in O(log n) time.

Q78. [Moderate] How do you decide which side is sorted during binary search in a rotated array?
A. Compare arr[mid] with arr[low] and arr[high]
B. Check if array is rotated
C. Always move left
D. Check if arr[mid] equals the target
Answer: A
Explanation: One half is guaranteed to be sorted; compare arr[mid] with arr[low] and arr[high].

Q79. [Moderate] Given arr = [4,5,6,7,0,1,2], what will binary search return for target = 0?
A. 3
B. 4
C. 5
D. -1
Answer: B
Explanation: Element 0 is at index 4.

Q80. [Difficult] What is the output of this function on input arr = [6,7,1,2,3,4,5], target = 3?
```
int search(int[] arr, int target) {
    int low = 0, high = arr.length - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (arr[mid] == target) return mid;
        if (arr[low] <= arr[mid]) {
```

```
        if (arr[low] <= target && target < arr[mid])
            high = mid - 1;
        else low = mid + 1;
      } else {
        if (arr[mid] < target && target <= arr[high])
            low = mid + 1;
        else high = mid - 1;
      }
    }
    return -1;
}
```
A. 2
B. 3
C. 4
D. 5
Answer: C
Explanation: Element 3 is at index 4.

Q81. [Difficult] What is the time complexity of searching in a rotated sorted array with duplicates?
A. O(log n)
B. O(n)
C. O(n log n)
D. O(1)
Answer: B
Explanation: In worst-case (e.g., all duplicates), binary search may degrade to linear search.

Q82. [Difficult] Why might standard binary search fail on rotated arrays with duplicates?
A. It assumes unique values for decision making
B. Duplicates are not sorted
C. mid value can't be computed
D. The array becomes unsorted
Answer: A
Explanation: If arr[low] == arr[mid] == arr[high], binary search loses its ability to decide direction.