1. https://leetcode.com/problems/sqrtx/

```
public class MySqrt {

    public int mySqrt(int x) {
        if (x < 0) {
            return -1;
        }

        int low = 0;
        int high = x;
        int mid;

        while (low <= high) {
            mid = (low + high) / 2;

            if (mid * mid <= x && (mid + 1) * (mid + 1) > x)
{
                return mid;
            } else if (mid * mid < x) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
```

```
        }

        return -1;
    }

    public static void main(String[] args) {
        System.out.println(new MySqrt().mySqrt(4)); //
2
        System.out.println(new MySqrt().mySqrt(8)); //
2
        System.out.println(new MySqrt().mySqrt(16));
// 4
        System.out.println(new MySqrt().mySqrt(25));
// 5
        System.out.println(new MySqrt().mySqrt(36));
// 6
    }
}
```

2.https://leetcode.com/problems/search-in-rotated-sorted-array/

```
public class SearchInRotatedSortedArray {
```

```
    public int search(int[] nums, int target) {
        if (nums.length == 0) {
            return -1;
        }

        int low = 0;
        int high = nums.length - 1;
        int mid;

        while (low <= high) {
            mid = (low + high) / 2;

            if (nums[mid] == target) {
                return mid;
            }

            // Check if the left half is sorted.
            if (nums[mid] >= nums[low]) {
                if (target < nums[mid] && target >=
nums[low]) {
                    high = mid - 1;
                } else {
                    low = mid + 1;
                }
```

```
            }

            // Check if the right half is sorted.
            else {
                if (target > nums[mid] && target <=
nums[high]) {
                    low = mid + 1;
                } else {
                    high = mid - 1;
                }
            }
        }

        return -1;
    }

    public static void main(String[] args) {
        int[] nums = {4, 5, 1, 2, 3};
        int target = 1;
        System.out.println(new
SearchInRotatedSortedArray().search(nums,
target)); // 2
    }
}
```

3. https://leetcode.com/problems/minimum-size-subarray-sum/

```
public class MinimumSizeSubarraySum {

    public int minSubArraySum(int target, int[] nums) {
        if (nums.length == 0) {
            return -1;
        }

        int minLength = Integer.MAX_VALUE;
        int left = 0;
        int sum = 0;

        for (int right = 0; right < nums.length; right++) {
            sum += nums[right];

            while (sum >= target) {
                minLength = Math.min(minLength, right - left + 1);
```

```
                sum -= nums[left];
                left++;
            }
        }

        return minLength == Integer.MAX_VALUE ? -1 :
minLength;
    }

    public static void main(String[] args) {
        int target = 7;
        int[] nums = {2, 3, 1, 2, 4, 3};
        System.out.println(new
MinimumSizeSubarraySum().minSubArraySum(targe
t, nums)); // 2
    }
}
```

4. https://leetcode.com/problems/nth-magical-number/

```
public class NthMagicalNumber {

    private static final int MOD = 1000000007;

    public int nthMagicalNumber(int n, int a, int b) {
        int low = 1;
        int high = Integer.MAX_VALUE;
        int mid;

        while (low <= high) {
            mid = (low + high) / 2;

            if (countMagicalNumbers(mid, a, b) >= n) {
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }

        return low % MOD;
    }

    private static int countMagicalNumbers(int x, int
a, int b) {
```

```
        int count = 0;

        for (int i = x; i <= x * b; i += b) {
            if (i % a == 0) {
                count++;
            }
        }

        return count;
    }

    public static void main(String[] args) {
        int n = 10;
        int a = 2;
        int b = 3;
        System.out.println(new
NthMagicalNumber().nthMagicalNumber(n, a, b)); //
28
    }
}
```

5.https://www.interviewbit.com/problems/matrix-median/

```
public class MatrixMedian {

    private static final int INVALID_VALUE = -1;

    public int findMedian(int[][] matrix) {
        int n = matrix.length;
        int m = matrix[0].length;

        int low = 0;
        int high = n * m - 1;
        int mid;

        while (low <= high) {
            mid = (low + high) / 2;

            int row = mid / m;
            int col = mid % m;

            int value = matrix[row][col];

            if (value == INVALID_VALUE) {
                continue;
            }
```

```
            if (mid % 2 == 0) {
                // If mid is even, then the median is the
average of the two middle elements.
                int next = mid + 1;

                if (next / m == row && next % m == col) {
                    // If the next element is also invalid, then
the median is the current element.
                    return value;
                } else {
                    // If the next element is valid, then the
median is the average of the current and next
elements.
                    return (value + matrix[row][next % m]) / 2;
                }
            } else {
                // If mid is odd, then the median is the
middle element.
                return value;
            }
        }

        return -1;
    }
```

```
    public static void main(String[] args) {
        int[][] matrix = {{1, 3, 5}, {2, 6, 9}, {3, 6, 9}};
        System.out.println(new
MatrixMedian().findMedian(matrix)); // 5
    }
}
```

6.https://www.interviewbit.com/problems/search-in-bitonic-array/

```
public class SearchInBitonicArray {

    public int search(int[] arr, int x) {
        int low = 0;
        int high = arr.length - 1;

        while (low <= high) {
            int mid = (low + high) / 2;

            if (arr[mid] == x) {
                return mid;
```

```
            }

            // Check if the current element is less than
the previous element.
            if (mid > 0 && arr[mid - 1] > arr[mid]) {
                if (x < arr[mid]) {
                    high = mid - 1;
                } else {
                    low = mid + 1;
                }
            } else {
                if (x < arr[mid]) {
                    low = mid + 1;
                } else {
                    high = mid - 1;
                }
            }
        }

        return -1;
    }

    public static void main(String[] args) {
        int[] arr = {3, 9, 10, 20, 17, 5, 1};
        int x = 20;
```

```
        System.out.println(new
SearchInBitonicArray().search(arr, x)); // 3
    }
}
```

7. https://leetcode.com/discuss/general-discussion/1302335/aggressive-cows-spoj-fully-explained-c

```
public class AggressiveCows {

    public int solve(int[] stalls, int cows) {
        int low = 1;
        int high = stalls[stalls.length - 1] - stalls[0];
        int mid;
        int maxDist = -1;

        while (low <= high) {
            mid = (low + high) / 2;
```

```
        // Check if mid is a valid distance.
        boolean valid = true;
        for (int i = 1; i < cows; i++) {
            if (stalls[i] - stalls[i - 1] < mid) {
                valid = false;
                break;
            }
        }

        if (valid) {
            maxDist = mid;
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }

    return maxDist;
}

public static void main(String[] args) {
    int[] stalls = {1, 4, 7, 11, 15};
    int cows = 3;
    System.out.println(new
```

```
AggressiveCows().solve(stalls, cows)); // 3
    }
}
```

8. https://www.interviewbit.com/problems/painters-partition-problem/

```
public class PaintersPartitionProblem {

    private static final int MOD = 1000000007;

    public int paint(int A, int B, int[] C) {
        long sum = 0;

        for (int i = 0; i < C.length; i++) {
            sum += C[i];
        }

        long low = 1;
        long high = sum;
```

```
        long mid;
        long maxTime = -1;

        while (low <= high) {
            mid = (low + high) / 2;

            if (canPaint(mid, A, C)) {
                maxTime = mid;
                high = mid - 1;
            } else {
                low = mid + 1;
            }
        }

        return (int)(maxTime % MOD);
    }

    private static boolean canPaint(long time, int A,
int[] C) {
        long painters = 1;
        long sum = 0;

        for (int i = 0; i < C.length; i++) {
            if (sum + C[i] > time) {
                painters++;
```

```
            sum = 0;
        }

        sum += C[i];
    }

    return painters <= A;
  }

  public static void main(String[] args) {
    int A = 2;
    int B = 3;
    int[] C = {10, 20, 30};
    System.out.println(new
PaintersPartitionProblem().paint(A, B, C)); // 60
  }
}
```

9. https://www.interviewbit.com/problems/allocate-books/

```
public class AllocateBooks {
```

```
public int allocateBooks(int[] A, int B) {
    int sum = 0;

    for (int i = 0; i < A.length; i++) {
        sum += A[i];
    }

    int low = 1;
    int high = sum;
    int mid;
    int maxPages = -1;

    while (low <= high) {
        mid = (low + high) / 2;

        if (canAllocate(mid, A, B)) {
            maxPages = mid;
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }

    return maxPages;
```

```
    }

    private static boolean canAllocate(int pages, int[]
A, int B) {
        int students = 1;
        int currentPages = 0;

        for (int i = 0; i < A.length; i++) {
            if (currentPages + A[i] > pages) {
                students++;
                currentPages = 0;
            }

            currentPages += A[i];
        }

        return students <= B;
    }

    public static void main(String[] args) {
        int[] A = {1, 2, 3, 4, 5};
        int B = 2;
        System.out.println(new
AllocateBooks().allocateBooks(A, B)); // 3
    }
```

```
}
```