BINARY SEARCH
(1) BINARY SEARCH

```
def binary_search(arr, target):
    left, right = 0, len(arr) - 1

    while left <= right:
        # Avoiding overflow in the calculation of mid
        mid = left + (right - left) // 2

        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            left = mid + 1
        else:
            right = mid - 1

    # Return -1 if the target is not found
    return -1

# Example usage
arr = [1, 2, 4, 5, 6, 7, 8, 9]
target = 5
result = binary_search(arr, target)
print(result)
```

(2) FIND MIN IN ROTATED SORTED ARRAY

```
class Solution:
    def findMin(self, nums: List[int]) -> int:
      l, r = 0, len(nums) - 1
      while l < r:
          m = (l + r) // 2
          if nums[m] > nums[r]:
              l = m + 1
          else:
            r = m
      return nums[l]
```

(3) SEARCH IN ROTATED SORTED ARRAY

```
class Solution:
    def search(self, nums: List[int], target: int) -> int:
        l, r = 0, len(nums) - 1

        while l <= r:
            m = (l + r) // 2
            if nums[m] == target:
                return m
            if nums[l] <= nums[m]:

                if nums[l] <= target <= nums[m]:
                    r = m - 1
                else:
                    l = m + 1
            else:
                if nums[m] < target <= nums[r]:
                    l = m + 1
                else:
                    r = m - 1
        return -1
```

(4) KOKO EATING BANANAS

```
class Solution:
    def minEatingSpeed(self, piles: List[int], h: int) -> int:
        l, r = 1, max(piles)
        while l < r:
            m = (l + r) // 2
            hours = 0
            for pile in piles:
                hours += math.ceil(pile / m)

            if hours <= h:
                r = m
            else:
                l = m + 1

        return l
```

(5)SEARCH IN A 2D MATRIX

```
class Solution:
    def searchMatrix(self, matrix: List[List[int]], target: int) -> bool:
        r, c = len(matrix), len(matrix[0])
        t, b = 0, r - 1
        while t <= b:
            row = (t + b) // 2
            if target < matrix[row][0]:
                b = row - 1
            elif target > matrix[row][-1]:
                t = row + 1
            else:
                break

        if not (t <= b):
            return False
        row = (t + b) // 2
        l, r = 0,c - 1
        while l <= r:
            m = (l + r) // 2
            if target < matrix[row][m]:
                r = m - 1
            elif target > matrix[row][m]:
                l = m + 1
            else:
                return True

        return False
```

(6) SET MATRIX TO ZEROES NOT A BINARY SEARCH

```
class Solution:
    def setZeroes(self, matrix: List[List[int]]) -> None:
        """
        Do not return anything, modify matrix in-place instead.
        """
        R, C = len(matrix), len(matrix[0])
        rows, cols = set(), set()
        for i in range(R):
            for j in range(C):
                if matrix[i][j] == 0:
                    rows.add(i)
                    cols.add(j)

        for i in range(R):
            for j in range(C):
                if i in rows or j in cols:
                    matrix[i][j] = 0
```