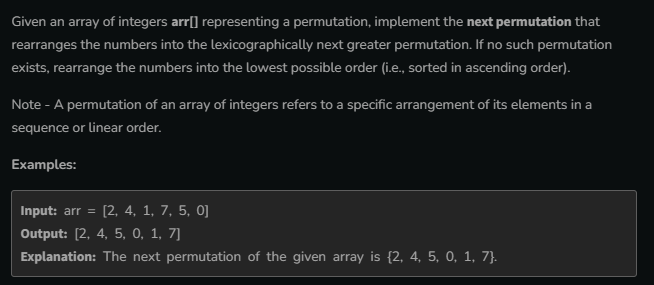
DSA\_PRACTISE\_7(19/11/2024)

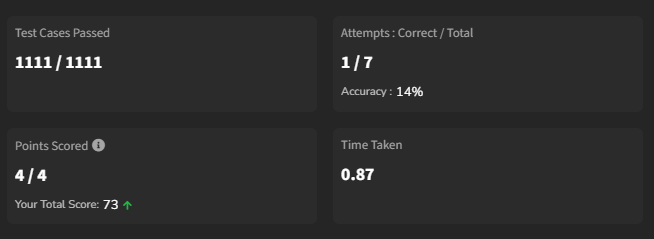
1.



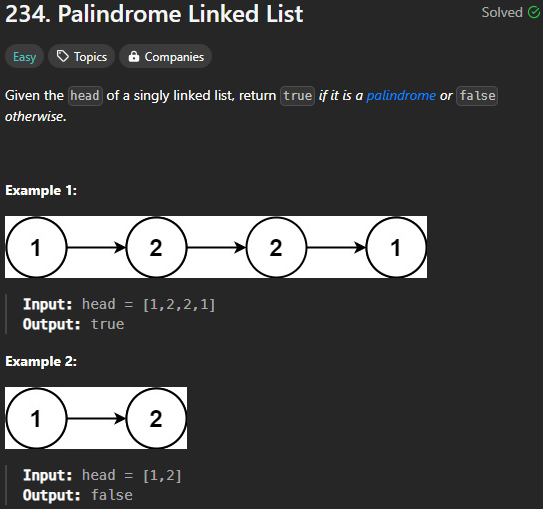
Code:



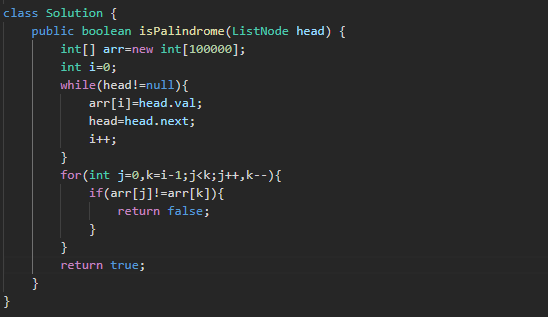
Output:



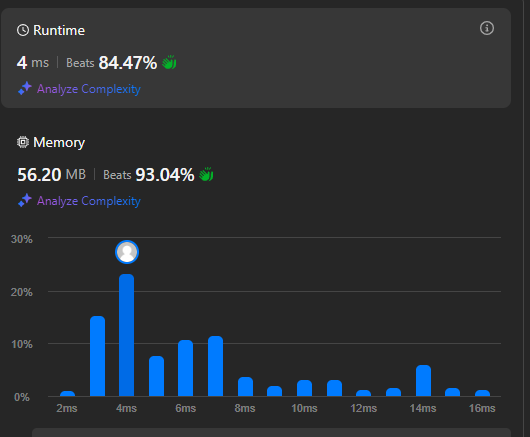
2.



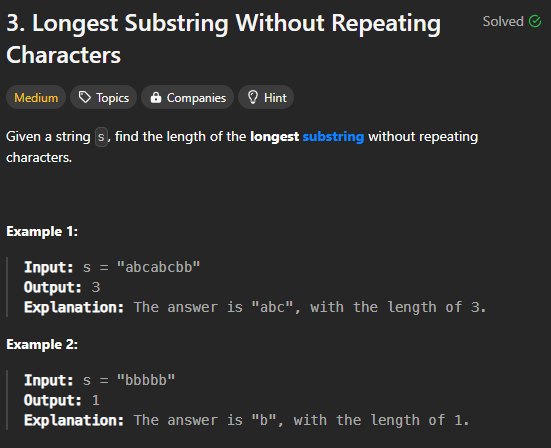
Code:



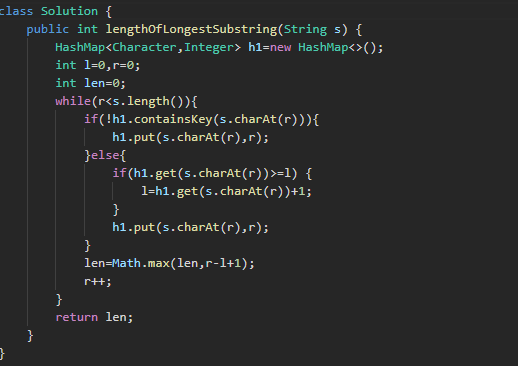
Ouput:



3.

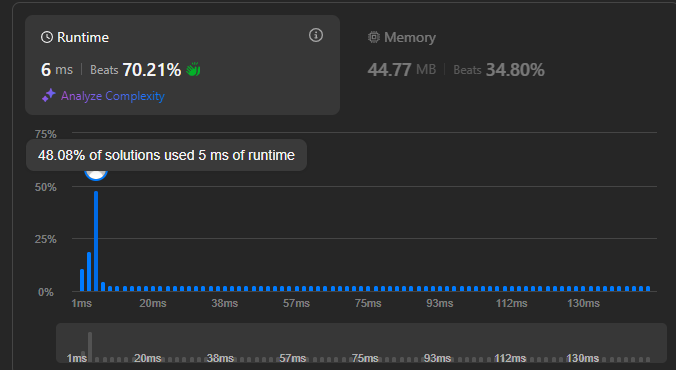


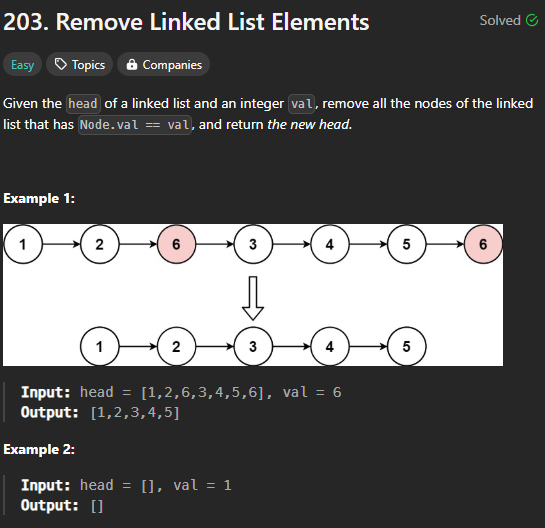
Code:



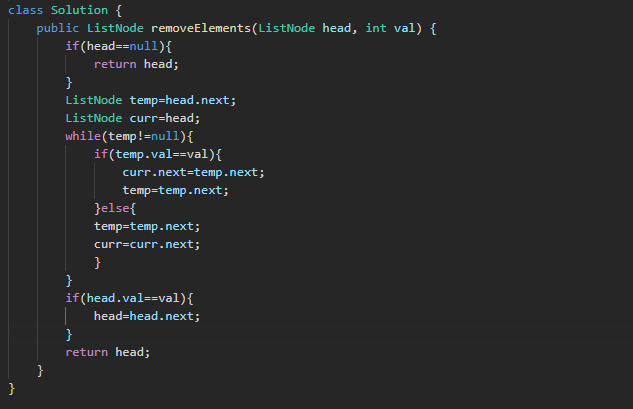
Output:

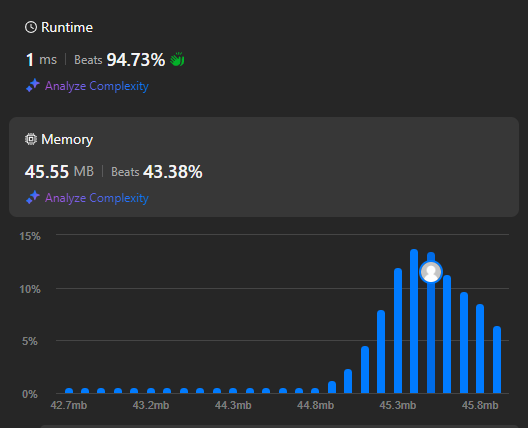
Output:

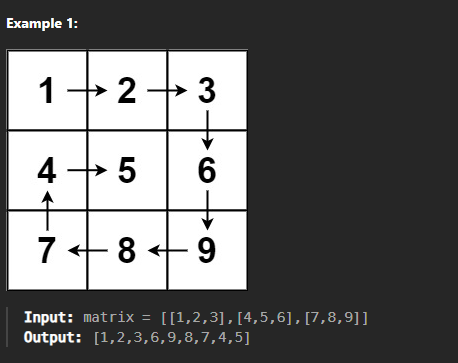


4. 

Code:

output:



5. 

Code:

class Solution {

public List<Integer> spiralOrder(int[][] matrix) {

List<Integer> result = new ArrayList<>();

if (matrix == null || matrix.length == 0) {

return result;

}

int rows = matrix.length, cols = matrix[0].length;

int left = 0, right = cols-1, top = 0, bottom = rows-1;

while (left <= right && top <= bottom) {

for (int i = left; i <= right; i++) {

result.add(matrix[top][i]);

}

top++;

for (int i = top; i <= bottom; i++) {

result.add(matrix[i][right]);

}

right--;

if (top <= bottom) {

for (int i = right; i >= left; i--) {

result.add(matrix[bottom][i]);

}

bottom--;

}

if (left <= right) {

for (int i = bottom; i >= top; i--) {

result.add(matrix[i][left]);

}

left++;

}

}

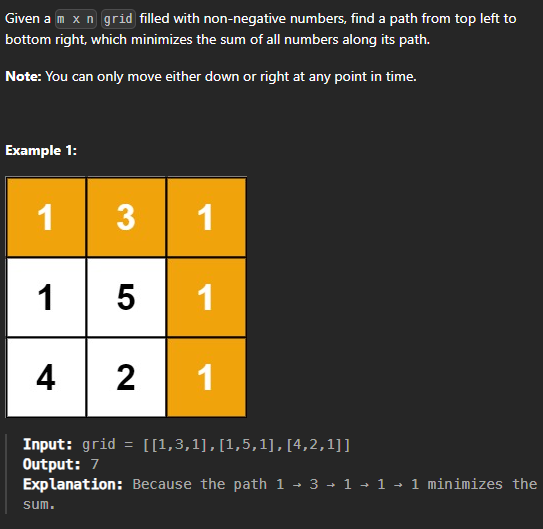
return result;

}

}

Output:



6. 

Code:

class Solution {

public int minPathSum(int[][] grid) {

int m = grid.length;

int n = grid[0].length;

for (int i = 1; i < m; i++) {

grid[i][0] += grid[i-1][0];

}

for (int j = 1; j < n; j++) {

grid[0][j] += grid[0][j-1];

}

for (int i = 1; i < m; i++) {

for (int j = 1; j < n; j++) {

grid[i][j] += Math.min(grid[i-1][j], grid[i][j-1]);

}

}

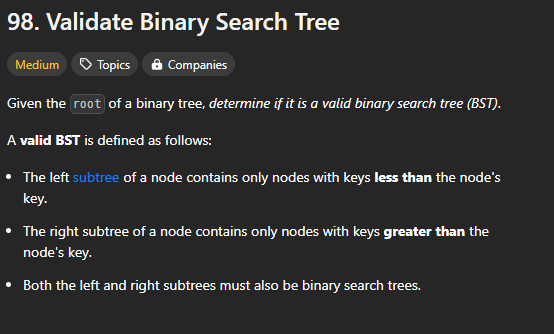
return grid[m-1][n-1];

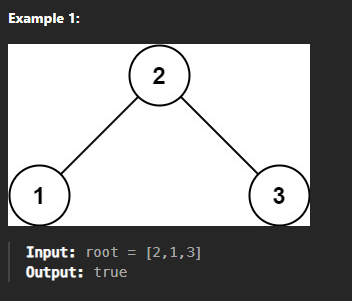
}

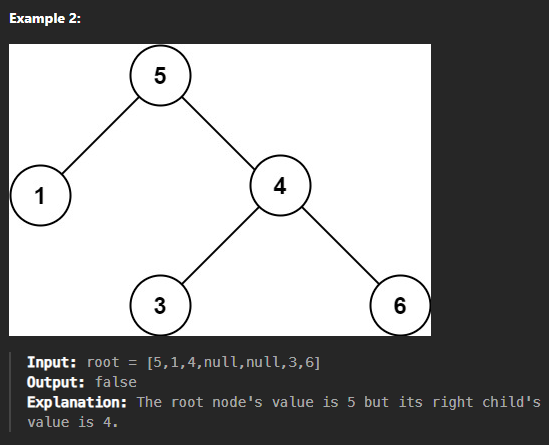
}

Output: 

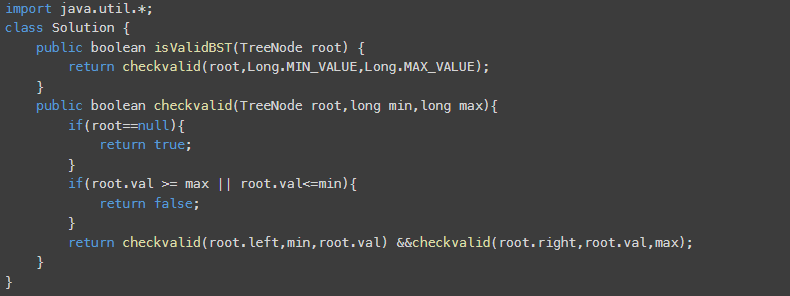
7.







Code:

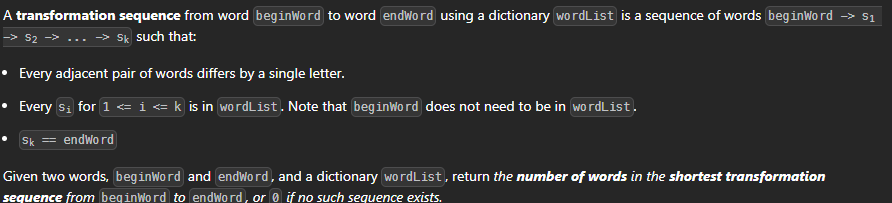


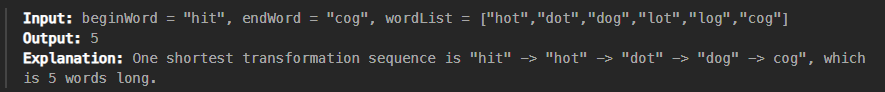
Output:



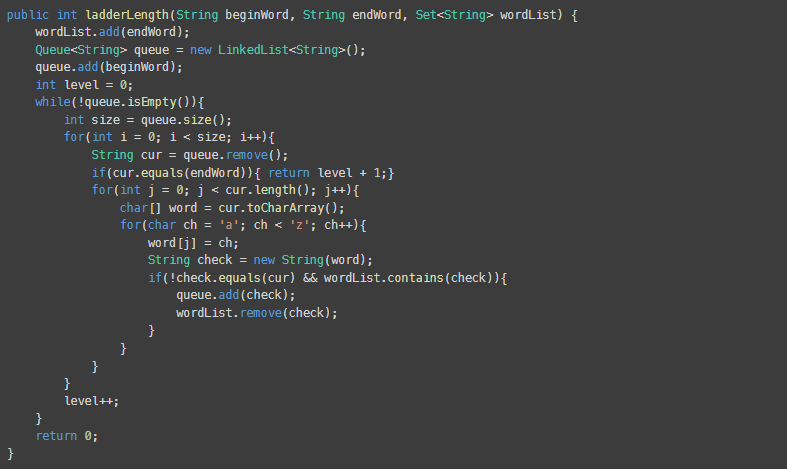
Word ladder 1:

8.





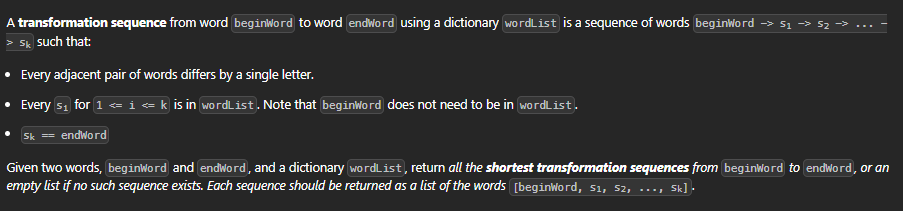
Code:

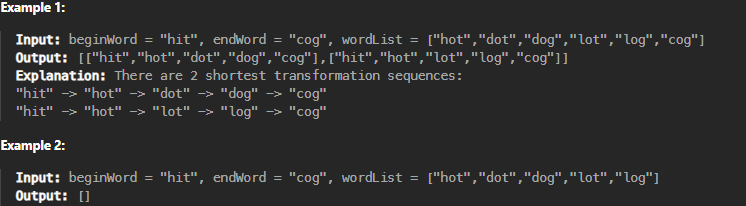


Output:



9.

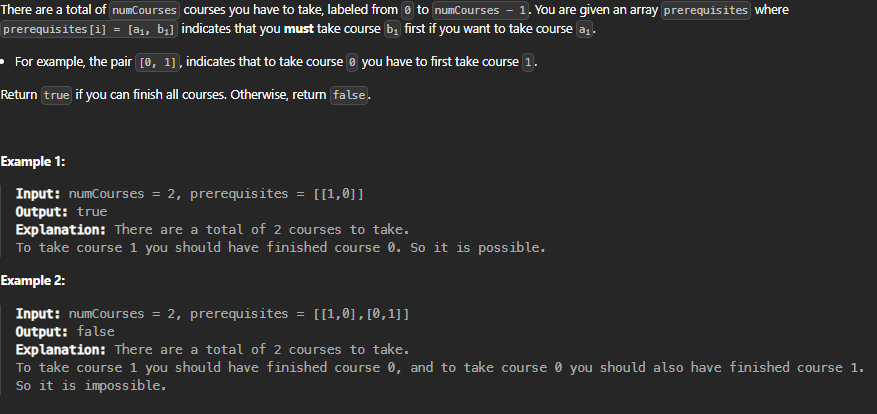


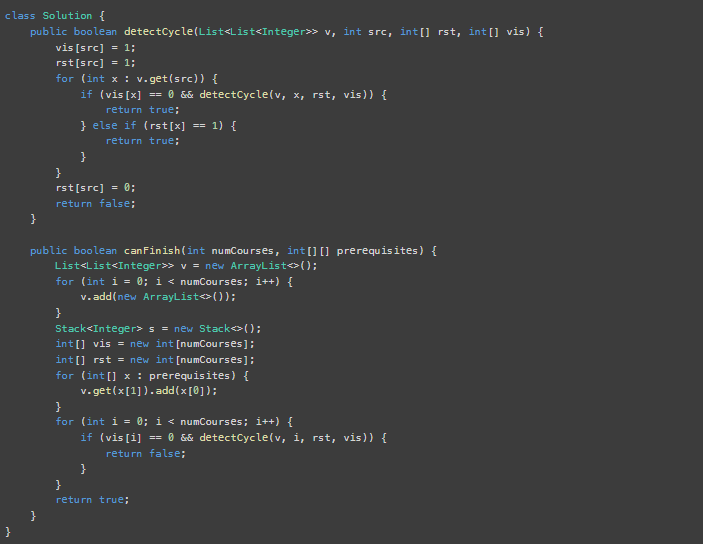


Output:



9.

output:



Output:

