Time complexity is a function describing the amount of time an algorithm takes in terms of the amount of input to the algorithm.

Space complexity is a function describing the amount of memory or space an algorithm takes in terms of the amount of input to the algorithm.

2.

**public** **class** **Solution** {

**public** **int**[] **twoSum**(**int**[] nums, **int** target) {

// **Space complexity is O(1)**

**int**[] ans = **new** **int**[**2**];

**//Space complexity is O(N)**

Map<Integer,Integer> map = **new** HashMap<Integer,Integer>();

**// Time complexity is O(N)**

**for** (**int** i=**0**;i<nums.length;i++) {

**if** (map.containsKey(nums[i])) {

ans[**0**] = map.get(nums[i]);

ans[**1**] = i;

**break**;

} **else** {

map.put(target-nums[i],i);

}

}

**return** ans;

}

}

Total time complexity is O(N), space complexity is O(N)

**class** **Solution** {

**public** **int** **numIslands**(**char**[][] grid) {

**if** (grid == **null** || grid.length == **0** || grid[**0**].length == **0**) {

**return** **0**;

}

**int** row = grid.length;

**int** col = grid[**0**].length;

**int** ans = **0**;

**// Time complexity is O(M\*N), space complexity is O(M\*N)**

**for** (**int** i =**0**; i < row; i++) {

**for** (**int** j=**0**;j < col; j++) {

**if** (grid[i][j] == '1') {

DFS(grid, i, j);

ans++;

}

}

}

**return** ans;

}

**public** **void** **DFS**(**char**[][] grid, **int** x, **int** y) {

**if** ( x<**0** || y<**0** || x > grid.length-**1** || y > grid[**0**].length-**1** || grid[x][y] == '0') {

**return**;

}

grid[x][y] = '0';

DFS(grid,x-**1**,y);

DFS(grid,x+**1**,y);

DFS(grid,x,y-**1**);

DFS(grid,x,y+**1**);

}

}

Total time complexity is O(M\*N), space complextiy is O(M\*N) which is stack space and recalled M\*N times