1. **public** **class** Salary {
2. **public** **double** employeeSalary(**double** hours) {
3. // Space complexity is O(1)
4. **double** salary  = 0;
5. **double** rate = 15;
6. **int** firstHours = 36;
7. **int** nextHours = 5;
8. **int** maxHours = 48;
10. // Time complexity is O(1)
11. **if** (hours <= firstHours) {
12. salary = rate \* hours;
13. } **else** **if** (hours <= firstHours + nextHours) {
14. salary = rate \* firstHours + rate \*1.5\*(hours - firstHours);
15. } **else** **if** (hours <= maxHours) {
16. salary = rate \* firstHours + rate\*1.5\*nextHours + rate\*2 \*(hours -firstHours-nextHours ) ;
17. } **else** {
18. System.out.println("The hour is beyond the maximum");
19. }
20. **return** salary;
21. }
22. **public** **static** **void** main(String[] args) {
24. System.out.println("Enter worked hours, the maximum is 48: ");
25. Scanner scanner = **new** Scanner(System.in);
26. String hours = scanner.nextLine();
28. System.out.println("The salary is " + **new** Salary().employeeSalary(Double.parseDouble(hours)));
29. }
30. }

Total time complexity is O(1) , space complexity is O(1)

1. **public** **class** ThreeSum {
2. **public** List<List<Integer>> threeSum(**int**[] nums) {
3. // Space complexity is O(n)
4. List<List<Integer>> res = **new** ArrayList<>();
5. // Time complexity is O(n\*logn)
6. Arrays.sort(nums);
8. // Time complexity is O(n\*n)
9. **for** (**int** i = 0; i < nums.length - 2; i++) {
10. **if** (i > 0 && nums[i] == nums[i-1]) **continue**;
11. **int** val = - nums[i];
12. **int** low = i+1, high = nums.length - 1;
13. **while** ( low < high) {
14. **if** (nums[low] + nums[high] > val) {
15. high--;
16. } **else** **if** (nums[low] + nums[high] < val) {
17. low++;
18. } **else** {
19. res.add(Arrays.asList(nums[i],nums[low],nums[high]));
20. **while** (low < high && nums[low] == nums[low+1]) low++;
21. **while** (low < high && nums[high] == nums[high-1]) high--;
22. low++;
23. high--;
24. }
25. }
26. }
27. **return** res;
28. }
29. }

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

1. **public** **class** LongestSubstring {
2. **public** **int** lengthOfLongestSubstring(String s) {
3. **if** (s == **null** || s.length() == 0) **return** 0;
4. **int** left = 0;
5. **int** right = 0;
6. **int** res = 0;
7. // Space complexity is O(n)
8. Set<Character> set = **new** HashSet<>();
10. //Time complexity is O(n)
11. **while** (right < s.length()) {
12. **if** (!set.contains(s.charAt(right))) {
13. set.add(s.charAt(right));
14. right++;
15. res = Math.max(res, set.size());
16. } **else** {
17. set.remove(s.charAt(left));
18. left++;
19. }
20. }
21. **return** res;
22. }
23. }

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