ORACLE - DSA Top Interview Questions & Solutions

Solution by: Prashant Kumar

1. https://practice.geeksforgeeks.org/problems/kadanes-algorithm-1587115620/1

```
    class Solution {

        long maxSubarraySum(int arr[], int n) {

                // Your code here
                int maxEndingHere = 0;
                int maxSoFar = Integer.MIN_VALUE;

                for (int i = 0; i<n; i++) {
                        maxEndingHere += arr[i];

                        if (maxEndingHere > maxSoFar) {
                                maxSoFar = maxEndingHere;
                        }

                        if (maxEndingHere<0) {
                                maxEndingHere = 0;
                        }
                }
                return maxSoFar;
        }
}
```

2. https://practice.geeksforgeeks.org/problems/remove-loop-in-linked-list/1

```
class Solution {
        public static void removeLoop(Node head) {
                // code here
                Node slow = head;
                Node fast = head;
                while (slow != null && fast.next != null) {
                        slow = slow.next;
                        fast = fast.next.next;
                        if (fast == null) {
                                break;
                        }
                        if (slow == fast) {
                                break;
                        }
                }
                Node newp = head;
                Node pointer = fast;
                if (slow == fast) {
                        while (slow != newp) {
                                slow = slow.next;
                                newp = newp.next;
                        }
                        while (pointer.next != slow) {
                                pointer = pointer.next;
                        }
                        pointer.next = null;
                }
        }
}
```

3. https://practice.geeksforgeeks.org/problems/detect-cycle-in-a-directed-graph/1

```
class Solution {
	// Function to detect cycle in a directed graph.
	public boolean isCyclic(int V, ArrayList<ArrayList<Integer>> adj) {
		// code here
		boolean[] vis = new boolean[V];
		boolean[] rec = new boolean[V];

		for (int i = 0; i<V; i++) {
			if (!vis[i]) {
				if (dfs(i, adj, vis, rec))
					return true;
			}
		}
		return false;
	}

	private boolean dfs(int v, ArrayList<ArrayList<Integer>> adj, boolean[] vis, boolean[]
rec) {
		vis[v] = true;
		rec[v] = true;

		for (Integer neighbor: adj.get(v)) {
			if (!vis[neighbor]) {
				if (dfs(neighbor, adj, vis, rec))
					return true;
			} else if (rec[neighbor] == true)
				return true;
		}

		rec[v] = false;
		return false;
	}
}
```

4. https://practice.geeksforgeeks.org/problems/binary-search-1587115620/1

```
class Solution {
    int binarysearch(int arr[], int n, int k){
        // code here
        int start = 0 ;
        int end = arr.length - 1;

        while (start <= end) {
            int mid = (start + end)/ 2;

            if (arr[mid] == k) {
                return mid;
            } else if (k > arr[mid]) {
                start = mid +1 ;
            } else if (k < arr[mid]) {
                end = mid -1 ;
            }
        }

       return -1;
    }
}
```

5. https://practice.geeksforgeeks.org/problems/parenthesis-checker2744/1

```
class Solution {
	//Function to check if brackets are balanced or not.
	static boolean ispar(String x) {
		// add your code here
		Stack<Character> s = new Stack<>();
		if (x.length() == 1 || x.charAt(0) == ']' || x.charAt(0) == ')' || x.charAt(0) == '}') {
			return false;
		}
		for (int i = 0; i<x.length(); i++) {
			if ((s.empty() == false) && ((x.charAt(i) == ']' && s.peek() == '[') ||
(x.charAt(i) == ')' && s.peek() == '(') || (x.charAt(i) == '}' && s.peek() == '{'))) {
				s.pop();
			} else {
				s.push(x.charAt(i));
			}

		}
		if (s.empty()) {
			return true;
		} else {
			return false;
		}
	}
}
```

6. **https://practice.geeksforgeeks.org/problems/remove-duplicate-element-from-sorted-linked-list/1**

```
class GfG {
	//Function to remove duplicates from sorted linked list.
	Node removeDuplicates(Node head) {
		// Your code here
		Node temp = head;
		Node prev = temp;
		while (temp != null) {
			temp = temp.next;
			while (temp != null && temp.data == prev.data) {
				prev.next = temp.next;
				temp = prev.next;
			}
			if (temp != null) {
				prev = temp;
			}

		}
		return head;
	}
}
```

7. https://practice.geeksforgeeks.org/problems/0-1-knapsack-problem0945/1

```
class Solution {
        //Function to return max value that can be put in knapsack of capacity W.
        static int solve(int capacity, int wt[], int val[], int index) {

                if (index == 0) {
                        if (wt[0]<= capacity) {
                                return val[0];
                        } else {
                                return 0;
                        }
                }

                int include = 0;
                if (wt[index]<= capacity) {
                        include = val[index] + solve(capacity - wt[index], wt, val, index - 1);
                }

                int exclude = 0 + solve(capacity, wt, val, index - 1);

                return Math.max(include, exclude);

        }

        //recursion + memoization
        static int solve1(int capacity, int wt[], int val[], int index, int dp[][]) {

                if (index == 0) {
                        if (wt[0]<= capacity) {
                                return val[0];
                        } else {
                                return 0;
                        }
                }

                if (dp[index][capacity] != -1) {
                        return dp[index][capacity];
                }

                int include = 0;
                if (wt[index]<= capacity) {
```

```
                include = val[index] + solve1(capacity - wt[index], wt, val, index - 1, dp);
            }

            int exclude = 0 + solve1(capacity, wt, val, index - 1, dp);

            dp[index][capacity] = Math.max(include, exclude);

            return dp[index][capacity];

        }

        //Using Tabulation - bottom up
        static int solve2(int capacity, int wt[], int val[], int n) {
            int dp[][] = new int[n][capacity + 1];
            for (int i = 0; i<n; i++) {
                Arrays.fill(dp[i], 0);
            }

            for (int w = wt[0]; w<= capacity; w++) {
                dp[0][w] = val[0];
            }

            for (int index = 1; index<n; index++) {
                for (int w = 0; w<= capacity; w++) {
                    int include = 0;
                    if (wt[index]<= w) {
                        include = val[index] + dp[index - 1][w - wt[index]];
                    }
                    int exclude = 0 + dp[index - 1][w];
                    dp[index][w] = Math.max(include, exclude);
                }
            }

            return dp[n - 1][capacity];

        }
        static int knapSack(int W, int wt[], int val[], int n) {
            // your code here

            int index = n - 1;
            return solve2(W, wt, val, n);
        }
}
```

8. **https://practice.geeksforgeeks.org/problems/spirally-traversing-a-matrix-1587115621/1**

```
class Solution
{
    //Function to return a list of integers denoting spiral traversal of matrix.
    static ArrayList<Integer> spirallyTraverse(int matrix[][], int r, int c)
    {
        // code here
        ArrayList<Integer> list = new ArrayList<>();

        int top= 0, bottom=r-1,left=0,right=c-1;
        int dir=1;

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

            if(dir==1){
                for(int i=left;i<=right;i++){
                    list.add(matrix[top][i]);
                }
                top++;
            }

            else if(dir==2){
                for(int i=top;i<=bottom;i++){
                    list.add(matrix[i][right]);
                }
                right--;
            }

            else if(dir==3){
                for(int i=right;i>=left;i--){
                    list.add(matrix[bottom][i]);
                }
                bottom--;
            }
            else if(dir==0){
                for(int i=bottom;i>=top;i--){

                    list.add(matrix[i][left]);
                }
                left++;
            }
            dir=(dir+1)%4;
        }

        return list;
    }
}
```

9. https://practice.geeksforgeeks.org/problems/merge-two-sorted-linked-lists/1

```
class LinkedList
{
   //Function to merge two sorted linked list.
   Node sortedMerge(Node head1, Node head2) {

    // This is a "method-only" submission.
    // You only need to complete this method
    Node sentinel = new Node(-1), sorted = sentinel;

   while(head1 != null && head2 != null){
          if(head1.data <= head2.data){
             sorted.next = head1;
             head1 = head1.next;
          }else{
             sorted.next = head2;
             head2 = head2.next;
          }
          sorted = sorted.next;
      }

      if(head1 != null) sorted.next = head1;
      else sorted.next = head2;

      return sentinel.next;
  }
}
```

10. https://practice.geeksforgeeks.org/problems/queue-using-two-stacks/1

```
class StackQueue {
	Stack<Integer> s1 = new Stack<Integer> ();
	Stack<Integer> s2 = new Stack<Integer> ();

	//Function to push an element in queue by using 2 stacks.
	void Push(int x) {
		// Your code here
		s1.push(x);
	}

	//Function to pop an element from queue by using 2 stacks.
	int Pop() {
		// Your code here
		int x;
		if (s1.isEmpty() && s2.isEmpty()) {
			return -1;
		}
		if (s2.isEmpty()) {
			while (!s1.isEmpty()) {
				x = s1.pop();
				s2.push(x);
			}
		}
		x = s2.pop();
		return x;
	}
}
```

11. https://practice.geeksforgeeks.org/problems/stack-using-two-queues/1

```
class Queues {
        Queue<Integer> q1 = new LinkedList<Integer> ();
        Queue<Integer> q2 = new LinkedList<Integer> ();

        //Function to push an element into stack using two queues.
        void push(int a) {
                // Your code here
                q1.add(a);
        }

        //Function to pop an element from stack using two queues.
        int pop() {
                // Your code here
                int pop = -1;
                if (!q1.isEmpty()) {
                        while (q1.size() != 1) {
                                q2.add(q1.poll());
                        }
                        pop = q1.poll();
                        while (!q2.isEmpty()) {
                                q1.add(q2.poll());
                        }
                }
                return pop;
        }

}
```

12. https://practice.geeksforgeeks.org/problems/first-repeating-element4018/1

```
class Solution {
    // Function to return the position of the first repeating element.
    public static int firstRepeated(int[] arr, int n) {
        // Your code here
        int count = 0;
        Map<Integer, Integer> countMap = new HashMap<>();
        for (int i = 0; i<n; i++) {
            if (countMap.containsKey(arr[i])) {
                countMap.put(arr[i], countMap.get(arr[i]) + 1);
            } else {
                countMap.put(arr[i], 1);
            }
        }
        for (int j = 0; j<n; j++) {
            if (countMap.get(arr[j]) > 1) {
                return j + 1;
            }
        }

        return -1;
    }
}
```

13. https://practice.geeksforgeeks.org/problems/merge-sort/1?page=1

```
class Solution {

        void merge(int arr[], int l, int m, int r) {

                // Your code here

                int size1 = m - l + 1;
                int size2 = r - m;

                int[] temp1 = new int[size1];
                int[] temp2 = new int[size2];

                for (int i = 0; i<size1; i++) {
                        temp1[i] = arr[l + i];;
                }

                for (int j = 0; j<size2; j++) {
                        temp2[j] = arr[m + j + 1];;
                }

                int first = 0;
                int second = 0;
                int k = l;
                while (first<size1 && second<size2) {
                        if (temp1[first]<= temp2[second]) {
                                arr[k++] = temp1[first++];
                        } else {
                                arr[k++] = temp2[second++];
                        }
                }

                while (first<size1) {
                    arr[k++] = temp1[first++];
                }






        while (second<size2) {
                    arr[k++] = temp2[second++];
                }
        }

        void mergeSort(int arr[], int l, int r) {
                //code here
                if (l<r) {
                    int m = l + (r - l) / 2;

                    mergeSort(arr, l, m);
                    mergeSort(arr, m + 1, r);

                    merge(arr, l, m, r);

                }
        }
}
```

14. https://practice.geeksforgeeks.org/problems/stock-buy-and-sell-1587115621/1

```
class Solution {
	//Function to find the days of buying and selling stock for max profit.
	ArrayList<ArrayList<Integer> > stockBuySell(int A[], int n) {
		// code here

		ArrayList<ArrayList<Integer>> aList = new ArrayList<ArrayList<Integer>> ();

		for (int i = 0; i<n; i++) {
			if (i<n - 1 && A[i]<A[i + 1]) {
				ArrayList<Integer> newArr = new ArrayList<Integer> (2);
				newArr.add(i);
				newArr.add(i + 1);
				aList.add(newArr);
			}
		}
		return aList;
	}
}
```

15. https://practice.geeksforgeeks.org/problems/connect-nodes-at-same-level/1

```
class Solution {
	//Function to connect nodes at same level.
	public void connect(Node root) {
		// Your code goes here.
		Queue<Node> q = new LinkedList<>();
		q.offer(root);
		int levelCount = 1;
		while (!q.isEmpty()) {
			int tmpCount = 0;
			Node next = q.poll();
			tmpCount = tmpCount + add(q, next);
			for (int i = 0; i<levelCount - 1; i++) {
				Node node = q.poll();
				tmpCount = tmpCount + add(q, node);
				next.nextRight = node;
				next = node;
			}

			levelCount = tmpCount;
		}
	}

	private int add(Queue<Node> q, Node node) {
		int result = 0;
		if (node.left != null) {
			q.add(node.left);
			result++;
		}
		if (node.right != null) {
			q.add(node.right);
			result++;
		}
		return result;
	}
}
```

16. https://practice.geeksforgeeks.org/problems/root-to-leaf-path-sum/1

```
class Solution {
    /*you are required to complete this function */

    boolean hasPathSum(Node root, int S) {
        // Your code here
        if (root == null)
            return false;

        if (root.left == root.right && S - root.data == 0)
            return true;

        if (S<0)
            return false;

        return hasPathSum(root.left, S - root.data) || hasPathSum(root.right, S -
root.data);
    }
}
```

17. https://practice.geeksforgeeks.org/problems/heap-sort/1

```
class Solution {
        //Function to build a Heap from array.


        void buildHeap(int arr[], int n) {
                // Your code here
                for (int i = n / 2 - 1; i >= 0; i--) {
                        heapify(arr, n, i);
                }
        }

        //Heapify function to maintain heap property.
        void heapify(int arr[], int n, int i) {
                // Your code here

                int largest = i;
                int l = 2 * i + 1;
                int r = 2 * i + 2;

                if (l<n && arr[l] > arr[largest]) {
                        largest = l;
                }

                if (r<n && arr[r] > arr[largest]) {
                        largest = r;
                }
                if (largest != i) {

                        swap(arr, i, largest);
                        heapify(arr, n, largest);
                }

        }
        //Function to sort an array using Heap Sort.

        public void heapSort(int arr[], int n) {
                //code here
                buildHeap(arr, n);
                for (int i = n - 1; i > 0; i--) {
                        swap(arr, 0, i);
                        heapify(arr, i, 0);
                }
          }

        void swap(int arr[], int i, int j) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
          }
}
```

18. https://practice.geeksforgeeks.org/problems/number-of-coins1824/1

```
class Solution {

    public int minCoins(int coins[], int M, int V) {
        // Your code goes here
        int arr[] = new int[V + 1];
        for (int i = 1; i<= V; i++) {
            arr[i] = -1;
        }
        for (int i = 1; i<= V; i++) {
            int temp = Integer.MAX_VALUE;
            for (int j: coins) {
                if (i >= j && arr[i - j] != -1) {
                    temp = Math.min(temp, arr[i - j] + 1);
                }
            }
            if (temp != Integer.MAX_VALUE)
                arr[i] = temp;
        }
        return arr[V];
    }
}
```

19. https://practice.geeksforgeeks.org/problems/egg-dropping-puzzle-1587115620/1

```
class Solution {
    //Function to find minimum number of attempts needed in
    //order to find the critical floor.
    static int eggDrop(int n, int k) {
        // Your code here
        int[][] dp = new int[n + 1][k + 1];

        return getMin(n, k, dp, k);

    }

    private static int getMin(int n, int k, int[][] dp, int min) {

        if (n == 1) {
            return k;
        }

        if (k == 1) {
            return 1;
        }

        if (dp[n][k] > 0) {
            return dp[n][k];
        }

        for (int i = 1; i<= k; i++) {
            int down = getMin(n - 1, i - 1, dp, min);
            int up = getMin(n, k - i, dp, min);

            min = Math.min(min, Math.max(down, up) + 1);
            dp[n][k] = min;
        }
        return min;
    }
}
```

20. https://practice.geeksforgeeks.org/problems/check-if-strings-are-rotations-of-each-other-or-not-1587115620/1

```
class Solution
{
    //Function to check if two strings are rotations of each other or not.

    public static boolean areRotations(String s1, String s2 )
    {
        // Your code here
        if(s1.length()!=s2.length()){
            return false;
        }

        return (s1+s1).contains(s2);
    }

}
```

21. https://practice.geeksforgeeks.org/problems/search-in-a-matrix-1587115621/1

```
class Solution {
    //Function to search a given number in row-column sorted matrix.
    static boolean search(int matrix[][], int n, int m, int x) {
        // code here
        int i = 0;
        int j = m - 1;

        while (i<n && j >= 0) {
            if (matrix[i][j] == x) return true;
            if (x > matrix[i][j]) {
                i++;

            } else {
                j--;
            }

        }

        return false;
    }
}
```

22. https://practice.geeksforgeeks.org/problems/search-in-a-matrix17201720/1

```
class Sol {
	public static int matSearch(int mat[][], int N, int M, int X) {
		// your code here
		int i = 0, j = M - 1;
		while (i<N && j >= 0) {
			if (mat[i][j] == X)
				return 1;
			else if (mat[i][j] > X)
				j--;
			else
				i++;
		}
		return 0;
	}
}
```

23. https://practice.geeksforgeeks.org/problems/search-in-a-matrix-1587115621/1

```
class Solution {
    //Function to search a given number in row-column sorted matrix.
    static boolean search(int matrix[][], int n, int m, int x) {

        int i = 0;
        int j = m - 1;

        while (i<n && j >= 0) {
            if (matrix[i][j] == x) return true;
            if (x > matrix[i][j]) {
                i++;

            } else {
                j--;
            }

        }

        return false;
    }
}
```

24. https://practice.geeksforgeeks.org/problems/stock-buy-and-sell2615/1

```
class Solution {
        public void stockBuySell(int[] price, int n) {
                // code here
                int profit = 0;
                int i = 0;
                int j = 0;
                while (i<n) {
                        while (j<n - 1 && price[j + 1] > price[j]) {
                                j++;
                        }
                        if (i != j) System.out.print("(" + i + " " + j + ")" + " ");
                        profit += price[j] - price[i];
                        i = ++j;
                        if (j >= n - 1) break;

                }

                if (profit == 0) System.out.print("No Profit");
                System.out.println();
        }
}
```

25. https://practice.geeksforgeeks.org/problems/sorting-elements-of-an-array-by-frequency-1587115621/1

```
class Solution {
	//Function to sort the array according to frequency of elements.
	static ArrayList<Integer> sortByFreq(int arr[], int n) {
		// add your code here
		HashMap<Integer, Integer> hm = new HashMap<>();
		for (int i = 0; i<n; i++) {
			if (hm.containsKey(arr[i])) {
				hm.put(arr[i], hm.get(arr[i]) + 1);
			} else {
				hm.put(arr[i], 1);
			}
		}
		ArrayList<Integer> ans = new ArrayList<>();
		Set<Entry<Integer, Integer>> entrySet = hm.entrySet();
		List<Entry<Integer, Integer>> list = new ArrayList<>(entrySet);
		Collections.sort(list, (a, b) -> (a.getValue() == b.getValue()) ? a.getKey() -
b.getKey() : b.getValue() - a.getValue());

		for (Entry<Integer, Integer> a1: list) {
			int freq = a1.getValue();
			int k = a1.getKey();
			while (freq > 0) {
				ans.add(k);
				freq--;
			}
		}
		return ans;
	}
}
```

26. https://practice.geeksforgeeks.org/problems/solve-the-sudoku-1587115621/1

```
class Solution {
        //Function to find a solved Sudoku.
        static boolean SolveSudoku(int grid[][]) {
                // add your code here
                for (int i = 0; i<9; i++) {
                        for (int j = 0; j<9; j++) {
                                if (grid[i][j] == 0) {
                                        for (int c = 1; c<= 9; c++) {
                                                if (isValid(grid, i, j, c)) {
                                                        grid[i][j] = c;
                                                        if (SolveSudoku(grid)) {
                                                                return true;
                                                        } else {
                                                                grid[i][j] = 0;
                                                        }
                                                }

                                        }
                                        return false;
                                }
                        }
                }
                return true;
        }
        static boolean isValid(int[][] grid, int row, int col, int c) {

                    for (int i = 0; i<9; i++) {
                        if (grid[row][i] == c) {
                                return false;
                        }
                        if (grid[i][col] == c) {
                                return false;
                        }
                        if (grid[3 * (row / 3) + i / 3][3 * (col / 3) + i % 3] == c) {
                                return false;
                        }
                }
                return true;
        }

        //Function to print grids of the Sudoku.
        static void printGrid(int grid[][]) {
                // add your code here
                for (int i = 0; i<9; i++) {
                        for (int j = 0; j<9; j++) {
                                System.out.print(grid[i][j] + " ");
                        }
                }
        }
}
```

27. https://practice.geeksforgeeks.org/problems/find-median-in-a-stream-1587115620/1

```
class Solution {
        //Function to insert heap.
        public static PriorityQueue<Integer> maxheap = new PriorityQueue<Integer>
(Collections.reverseOrder());
        public static PriorityQueue<Integer> minheap = new PriorityQueue<Integer> ();
        public static void insertHeap(int x) {
                if (maxheap.isEmpty() && minheap.isEmpty()) {
                        maxheap.add(x);
                } else {
                        if (maxheap.peek()<x) {
                                minheap.add(x);
                        } else {
                                maxheap.add(x);
                        }
                }
                balanceHeaps();
        }

        //Function to balance heaps.
        public static void balanceHeaps() {
                // add your code here
                int n = maxheap.size();
                int m = minheap.size();
                if (n - m == 2 || n - m == -2) {
                        if (n<m) {
                                int ele = minheap.peek();
                                minheap.poll();
                                maxheap.add(ele);
                        } else {
                                int ele = maxheap.peek();
                                maxheap.poll();
                                minheap.add(ele);
                        }
                }
                getMedian();

        }

        //Function to return Median.
        public static double getMedian() {
                // add your code here
                double res = 0;
                int n = maxheap.size();
                int m = minheap.size();
                if ((n + m) % 2 == 0) {
                        return ((double) maxheap.peek() + (double) minheap.peek()) / 2;
                }
                if (n > m)
                        return (double) maxheap.peek();

                return (double) minheap.peek();

        }

}
```

28. https://practice.geeksforgeeks.org/problems/armstrong-numbers2727/1

```
class Solution {
	static String armstrongNumber(int n) {
		// code here
		int sum = 0;
		int r = 0;
		int temp = n;
		while (n != 0) {
			r = n % 10;
			sum += Math.pow(r, 3);
			n = n / 10;
		}
		if (sum == temp) {
			return "Yes";
		} else {
			return "No";
		}
	}
}
```

29. https://practice.geeksforgeeks.org/problems/palindrome0746/1

```
class Solution {
        public String is_palindrome(int n) {
                // Code here
                int rev = 0;
                int org = n;
                while (n > 0) {
                        int d = n % 10;
                        rev = rev * 10 + d;
                        n /= 10;
                }
                return rev == org ? "Yes" : "No";
        }
}
```

30. https://practice.geeksforgeeks.org/problems/finding-profession3834/1

```
class Solution {
	static char profession(int level, int pos) {
		// code here
		if (pos == 1) {
			return 'e';
		}

		char ch = profession(level - 1, (pos + 1) / 2);
		if (pos % 2 != 0) {
			return ch;
		} else {
			if (ch == 'e') {
				return 'd';
			} else {
				return 'e';
			}
		}
	}
}
```

31. https://practice.geeksforgeeks.org/problems/avl-tree-insertion/1

```
class Solution {
	int max(int a, int b) {
		return (a > b) ? a : b;
	}
	int height(Node N) {
		if (N == null)
			return 0;

		return N.height;
	}
	int getdif(Node N) {
		if (N == null)
			return 0;

		return height(N.left) - height(N.right);
	}

	Node leftRotate(Node x) {
		Node y = x.right;
		Node T = y.left;

		y.left = x;
		x.right = T;

		x.height = max(height(x.left), height(x.right)) + 1;
		y.height = max(height(y.left), height(y.right)) + 1;
		return y;

	}
	Node rightRotate(Node y) {

		Node x = y.left;
		Node T = x.right;

		x.right = y;
		y.left = T;
```

```
                y.height = max(height(y.left), height(y.right)) + 1;
                x.height = max(height(x.left), height(x.right)) + 1;

                return x;

        }

        public Node insertToAVL(Node node, int data) {
                if (node == null) {
                        return (new Node(data));
                } else if (node.data<data) {
                        node.right = insertToAVL(node.right, data);
                } else if (node.data > data) {
                        node.left = insertToAVL(node.left, data);
                } else {
                        return node;
                }

                node.height = 1 + max(height(node.left), height(node.right));

                int balance = getdif(node);

                if (balance > 1 && data > node.left.data) {
                        node.left = leftRotate(node.left);
                        return rightRotate(node);
                }
                if (balance<-1 && data<node.right.data) {
                        node.right = rightRotate(node.right);
                        return leftRotate(node);
                }
                if (balance > 1 && data<node.left.data)
                        return rightRotate(node);

                if (balance<-1 && data > node.right.data)
                        return leftRotate(node);

                return node;
        }
}
```

32. https://practice.geeksforgeeks.org/problems/avl-tree-deletion/1

```
class Sol {
        public static Node deleteNode(Node root, int key) {
                // code here.
                if (root == null) return null;
                if (root.data == key) {
                        if (root.right == null) return root.left;
                        if (root.left == null) return root.right;
                        int val = getLeftMostVal(root.right);
                        root.data = val;
                        root.right = deleteNode(root.right, val);
                } else {
                        if (root.data<key) root.right = deleteNode(root.right, key);
                        else root.left = deleteNode(root.left, key);
                }
                root = balance(root);
                return root;
        }

        private static Node balance(Node root) {
                if (root == null) return root;
                int l = getHeight(root.left);
                int r = getHeight(root.right);
                if (Math.abs(l - r)<2) {
                        root.height = Math.max(l, r) + 1;
                        return root;
                }
                if (r > l) {
                        int rl = getHeight(root.right.left);
                        int rr = getHeight(root.right.right);
                        if (rl > rr) root.right = rotateRight(root.right);
                        root = rotateLeft(root);
                } else {
                        int ll = getHeight(root.left.left);
                        int lr = getHeight(root.left.right);
                        if (ll<lr) root.left = rotateLeft(root.left);
                        root = rotateRight(root);
                }
                return root;
        }
```

```
    private static Node rotateLeft(Node root) {
        Node temp = root.right;
        root.right = temp.left;
        root.height = Math.max(getHeight(root.left), getHeight(root.right)) + 1;
        temp.left = root;
        temp.height = Math.max(getHeight(temp.left), getHeight(temp.right)) + 1;
        return temp;
    }

    private static Node rotateRight(Node root) {

        Node temp = root.left;
        root.left = temp.right;
        root.height = Math.max(getHeight(root.left), getHeight(root.right)) + 1;
        temp.right = root;
        temp.height = Math.max(getHeight(temp.left), getHeight(temp.right)) + 1;
        return temp;
    }

    private static int getHeight(Node root) {
        if (root == null) return 0;
        return root.height;
    }

    private static int getLeftMostVal(Node root) {
        if (root.left == null) return root.data;
        return getLeftMostVal(root.left);
    }
}
```

33. https://practice.geeksforgeeks.org/problems/number-to-words0335/1

```
class Solution {
        static String one[] = {
                "", "one ", "two ", "three ", "four ", "five ", "six ", "seven ", "eight ",
                "nine ", "ten ", "eleven ", "twelve ", "thirteen ", "fourteen ", "fifteen ",
                "sixteen ", "seventeen ", "eighteen ", "nineteen "
        };

        static String ten[] = {
                "", "", "twenty ", "thirty ", "forty ","fifty ", "sixty ", "seventy ", "eighty ",
                "ninety "
        };

        String convertToWords(long n) {
                // code here
                String s = "";
                s += convert((int)(n / 10000000), "crore ");
                s += convert((int)((n / 100000) % 100), "lakh ");
                s += convert((int)((n / 1000) % 100), "thousand ");
                s += convert((int)((n / 100) % 10), "hundred ");

                if (n > 100 && n % 100 > 0) {
                        s += "and ";
                }

                s += convert((int)(n % 100), "");
                return s;
        }
        static String convert(int n, String s) {
                String str = "";
                if (n > 19) {
                        str += ten[n / 10] + one[n % 10];
                } else
                        str += one[n];

                if (n != 0)
                        str += s;

                return str;
        }
}
```

34. https://practice.geeksforgeeks.org/problems/remaining-string3515/1

```
class Solution {
    String printString(String S, char ch, int count) {
        // code here
        if (count == 0) {
            return S;
        }

        int c = 0;
        for (int i = 0; i<S.length(); i++) {
            char ch1 = S.charAt(i);
            if (ch1 == ch) {
                c++;
                if (c == count) {
                    String a = S.substring(i + 1);
                    if (a.length() == 0) {
                        return "Empty string";
                    }
                    return S.substring(i + 1);
                }
            }
        }
        return "Empty string";
    }
}
```

35. https://practice.geeksforgeeks.org/problems/partition-a-number-into-two-divisible-parts3605/1

```
class Solution {
	static String stringPartition(String S, int a, int b) {
		// code here
		for (int i = 1; i<S.length(); i++) {
			String sub1 = S.substring(0, i);
			String sub2 = S.substring(i, S.length());
			if ((Integer.parseInt(sub1) % a == 0) && (Integer.parseInt(sub2) % b ==
0))
				return sub1 + " " + sub2;
		}
		return "-1";
	}
}
```

36. https://practice.geeksforgeeks.org/problems/leaf-under-budget/1

```
class Solution{
    public int getCount(Node node, int bud)
    {
        //code here
        Queue<Node> q= new LinkedList<>();
         PriorityQueue<Integer> pQueue = new PriorityQueue<Integer>();
        q.add(node);
        int level=1;
        while(!q.isEmpty())
        {
            int length=q.size();
            for(int i=0;i<length;i++)
            {
                Node curr=q.remove();
                if(curr.left==null&&curr.right==null)
                {
                    pQueue.add(level);
                }
                else
                {
                    if(curr.left!=null)
                    {
                        q.add(curr.left);
                    }
                    if(curr.right!=null)
                    {
                        q.add(curr.right);
                    }
                }
            }
            level++;
        }
        return nodevisit(pQueue,bud);
    }
int nodevisit(PriorityQueue<Integer> pQueue,int bud)
   {
      int visit=0;
      while(!pQueue.isEmpty())
      {
         bud=bud-pQueue.poll();
         if(bud>=0)
            visit++;
         else
            break;
      }
      return visit;
   }
}
```

37. https://practice.geeksforgeeks.org/problems/find-the-minimum-time0253/1

```
class Solution {
	static int minTime(int S1, int S2, int N) {
		// code here
		int low = 0;
		int high = N;
		int ans = Integer.MAX_VALUE;
		while (low<= high) {
			int mid = (low + high) / 2;
			int temp = Math.max(S1 * mid, S2 * (N - mid));
			ans = Math.min(ans, temp);
			if ((S1 * mid) > (S2 * (N - mid)))
				high = mid - 1;
			else
				low = mid + 1;
		}
		return ans;
	}
}
```

38. https://practice.geeksforgeeks.org/problems/pattern-searching4145/1

```
class Solution {
        int search(String text, String pat) {
                // code here
                int m = text.length();
                int n = pat.length();
                for (int i = 0; i<= m - n; i++) {
                        boolean isBool = true;
                        for (int j = 0; j<n; j++) {
                                if (pat.charAt(j) != text.charAt(j + i)) {
                                        isBool = false;
                                        break;
                                }
                        }
                        if (isBool) {
                                return 1;
                        }
                }
                return 0;
        }
}
```

39. https://practice.geeksforgeeks.org/problems/maximum-number-of-zeroes4048/1

```
class Solution {
        String MaxZero(String arr[], int N) {
                int maxZero = 0;
                String res = "-1";
                for (int i = 0; i<N; i++) {
                        String str = arr[i];
                        int count = 0;
                        for (int j = 0; j<str.length(); j++) {
                                char ch = str.charAt(j);
                                if (ch == '0') {
                                        count++;
                                }
                        }
                        if (maxZero<count) {
                                maxZero = count;
                                res = str;
                        } else if (maxZero == count && count != 0) {
                                if (str.length() == res.length())
                                        res = (res.compareTo(str)<0) ? str : res;
                                else
                                        res = (res.length() > str.length()) ? res : str;
                        }
                }
                return res;
        }
}
```

40. **https://practice.geeksforgeeks.org/problems/sorting-employees5907/1**

```
class Solution {
	void sortRecords(node arr[], int n) {
		// Your code goes here
		Arrays.sort(arr, new Comparator<node> () {
			public int compare(node e1, node e2) {
				if (e1.salary > e2.salary) return 1;
				else if (e1.salary == e2.salary) {
					return e1.name.compareTo(e2.name);
				} else return -1;
			}
		});
	}

}
```

41. https://practice.geeksforgeeks.org/problems/shortest-un-ordered-subarray3634/1

```
class Solution {

	public long shortestUnorderedSubarray(long arr[], long n) {
		if (n<= 2) {
			return 0;
		}

		for (int i = 0; i<n - 2; i++) {
			if ((arr[i] > arr[i + 1] && arr[i + 1]<arr[i + 2]) || (arr[i]<arr[i + 1] && arr[i
+ 1] > arr[i + 2])) {
				return 3;
			}

		}
		return 0;
	}

}
```

42. https://practice.geeksforgeeks.org/problems/minimum-sum-partition3317/1

```
class Solution {

    public int minDifference(int arr[], int n) {
        // Your code goes here
        int sum = 0;
        for (int i: arr)
            sum += i;
        int sm = sum / 2;
        boolean t[][] = new boolean[n + 1][sm + 1];
        for (int i = 0; i<= n; i++)
            t[i][0] = true;
        for (int i = 1; i<= n; i++) {
            for (int j = 1; j<= sm; j++) {
                if (arr[i - 1]<= j)
                    t[i][j] = t[i - 1][j - arr[i - 1]] || t[i - 1][j];
                else
                    t[i][j] = t[i - 1][j];
            }
        }
        for (int i = sm; i >= 0; i--) {
            if (t[n][i] == true) {
                sm = i;
                break;
            }
        }
        return Math.abs(sm - sum + sm);
    }
}
```

43. https://practice.geeksforgeeks.org/problems/boolean-parenthesization5610/1

```
class Solution {
        static int countWays(int N, String S) {
                // code here
                StringBuilder s1 = new StringBuilder("");
                StringBuilder s2 = new StringBuilder("");
                for (int i = 0; i<N; i++) {
                        if (i % 2 == 0) {
                                s1.append(S.charAt(i));
                        } else {
                                s2.append(S.charAt(i));
                        }
                }
                String str1 = s1.toString();
                String str2 = s2.toString();
                int[][] truec = new int[str1.length()][str1.length()];
                int[][] falsec = new int[str1.length()][str1.length()];
                for (int d = 0; d<truec.length; d++) {
                        int j = d;
                        int i = 0;
                        while (j<truec.length) {
                                if (d == 0) {
                                        if (str1.charAt(i) == 'T') {
                                                truec[i][j] = 1;
                                                falsec[i][j] = 0;
                                        } else {
                                                truec[i][j] = 0;
                                                falsec[i][j] = 1;
                                        }
                                } else if (d == 1) {
                                        char o = str2.charAt(j - 1);
                                        if (o == '&') {
                                                if (truec[i][i] == 1 && truec[j][j] == 1) {
                                                        truec[i][j] = 1;
                                                        falsec[i][j] = 0;
                                                } else {
                                                        truec[i][j] = 0;
                                                        falsec[i][j] = 1;
                                                }
                                        } else if (o == '|') {

                                                if (truec[i][i] == 1 || truec[j][j] == 1) {
                                                        truec[i][j] = 1;
                                                        falsec[i][j] = 0;
                                                } else {
                                                        truec[i][j] = 0;
                                                        falsec[i][j] = 1;
                                                }
                                        } else {
                                                if ((truec[i][i] == 1 && truec[j][j] == 0) ||
truec[i][i] == 0 && truec[j][j] == 1) {
                                                        truec[i][j] = 1;
                                                        falsec[i][j] = 0;
                                                } else {
                                                        truec[i][j] = 0;
                                                        falsec[i][j] = 1;
                                                }
                                        }
                                } else {
                                        for (int k = i; k<j; k++) {
                                                char o = str2.charAt(k);
                                                if (o == '&') {
                                                        truec[i][j] += (truec[i][k] * truec[k + 1][j])
% 1003;
                                                        falsec[i][j] += ((falsec[i][k] * truec[k +
1][j]) % 1003 + (falsec[i][k] * falsec[k + 1][j]) % 1003 + (truec[i][k] * falsec[k + 1][j]) % 1003) %
1003;
                                                } else if (o == '|') {
                                                        falsec[i][j] += (falsec[i][k] * falsec[k + 1][j])
% 1003;
                                                        truec[i][j] += ((truec[i][k] * truec[k + 1][j])
% 1003 + (falsec[i][k] * truec[k + 1][j]) % 1003 + (truec[i][k] * falsec[k + 1][j]) % 1003) % 1003;
                                                } else {
                                                        falsec[i][j] += ((falsec[i][k] * falsec[k +
1][j]) % 1003 + (truec[i][k] * truec[k + 1][j]) % 1003) % 1003;
                                                        truec[i][j] += ((falsec[i][k] * truec[k + 1][j])
% 1003 + (truec[i][k] * falsec[k + 1][j]) % 1003) % 1003; }
                                        }
                                }
                                i++;
                                j++;
                        }
                }
                return truec[0][truec.length - 1] % 1003;
        }
}
```

44. https://practice.geeksforgeeks.org/problems/matrix-chain-multiplication0303/1

```
class Solution {
	static int matrixMultiplication(int N, int arr[]) {
		// code here
		int[][] dp = new int[1001][1001];

		for (int[] rows: dp) {
			Arrays.fill(rows, -1);
		}
		return solve(dp, arr, 1, arr.length - 1);
	}

	static int solve(int[][] dp, int[] arr, int i, int j) {
		if (i >= j) return dp[i][j] = 0;

		if (dp[i][j] != -1) return dp[i][j];
		int ans = Integer.MAX_VALUE;
		for (int k = i; k<= j - 1; k++) {
			int temp = solve(dp, arr, i, k) + solve(dp, arr, k + 1, j) + arr[k] * arr[i - 1]
* arr[j];
			ans = Math.min(temp, ans);
		}

		return dp[i][j] = ans;
	}
}
```