DSA Sheet Solutions

Top Questions of: DSA SHEET by FRAZ

Solution by: Prashant Kumar

---

1. https://leetcode.com/problems/valid-parentheses/

```
class Solution {
    public boolean isValid(String s) {
        Stack<Character> stack = new Stack<Character>();

        for (int i = 0 ; i < s.length() ; i++) {
            char ch = s.charAt(i);
            if (ch == '[' || ch == '{' || ch == '(') {
                stack.push(ch);
            } else if (!stack.empty() && ch == ')' && stack.peek() == '(') {
                stack.pop();
            } else if (!stack.empty() && ch == '}' && stack.peek() == '{') {
                stack.pop();
            } else if (!stack.empty() && ch == ']' && stack.peek() == '[') {
                stack.pop();
            } else {
                return false;
            }
        }
        return stack.empty();
    }
}
```

2. https://leetcode.com/problems/maximum-subarray/

```
class Solution {
   public int maxSubArray(int[] nums) {
      int maxSum = Integer.MIN_VALUE;
      int maxCurrent = 0;

      for (int i = 0 ; i < nums.length ; i++) {
         maxCurrent = maxCurrent + nums[i];
         if (maxSum < maxCurrent) {
            maxSum = maxCurrent;
         }

         if (maxCurrent < 0) {
            maxCurrent = 0;
         }
      }
      return maxSum;
   }
}
```

3. https://leetcode.com/problems/search-a-2d-matrix/

```
class Solution {
   public boolean searchMatrix(int[][] matrix, int target) {
      int i = 0;
      int j = matrix[0].length - 1;

      while (i < matrix.length && j >= 0) {
         if (target == matrix[i][j]) {

            return true;
         } else if (target < matrix[i][j]) {
            j--;
         } else {
            i++;
         }
      }
      return false;
   }
}
```

4. https://leetcode.com/problems/permutations/

```
class Solution {
   public List < List < Integer >> permute(int[] nums) {

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


      getCombinations(nums, output, result);
      return result;
   }

   private void getCombinations(int[] nums, List < Integer > output, List < List < Integer >> result) {

      if (output.size() == nums.length) {
         result.add(new ArrayList < Integer > (output));
      } else {
         for (int i = 0; i < nums.length; i++) {
            if (output.contains(nums[i])) continue;
            output.add(nums[i]);
            getCombinations(nums, output, result);

            output.remove(output.size() - 1);

         }
      }

   }

}
```

5. https://leetcode.com/problems/search-in-rotated-sorted-array/

```
class Solution {
	public int search(int[] nums, int target) {

		int pivot = nums.length > 1 ? find(nums) : 0;

		int left = binarySearcH(nums, 0, pivot, target);
		int right = binarySearcH(nums, pivot + 1, nums.length - 1, target);

		return Math.max(left, right);
	}
	public int find(int[] a) {
		int n = a.length;
		int l = 0, hi = n - 1;
		int ans = 0;
		while (l<= hi) {
			int mid = (l + hi) / 2;
			if (mid > 0 && mid<n - 1 && a[mid - 1]<a[mid] && a[mid] > a[mid + 1]) {
				ans = mid;
				break;
			}
			if (mid == 0 && mid + 1<n && a[mid] > a[mid + 1] && a[mid] > a[n - 1]) {
				ans = mid;
				break;
			}
			if (mid == n - 1 && mid - 1 >= 0 && a[mid] > a[mid - 1] && a[mid] > 0) {
				ans = mid;
				break;
			}
			if (a[n - 1]<a[mid]) l = mid + 1;
			else hi = mid - 1;
		}
		return ans;
	}
	public int binarySearcH(int[] a, int l, int hi, int target) {
		while (l<= hi) {
			int mid = (l + hi) / 2;
			if (a[mid] == target) return mid;

			if (a[mid] > target) hi = mid - 1;
			else l = mid + 1;
		}
		return -1;
	}
}
```

6. https://leetcode.com/problems/reverse-linked-list/

```
class Solution {
   public ListNode reverseList(ListNode head) {
      ListNode current = head;
      ListNode prev = null;
      while (current != null) {

         ListNode temp = current.next;
         current.next = prev;
         prev = current;
         current = temp;

      }
     return prev;
   }
}
```

7. https://leetcode.com/problems/top-k-frequent-elements/

```
class Solution {
	public int[] topKFrequent(int[] nums, int k) {
		HashMap<Integer, Integer> hm = new HashMap<>();
		for (int e: nums)
			hm.put(e, hm.getOrDefault(e, 0) + 1);
		PriorityQueue<Value> pq = new PriorityQueue();
		for (Integer key: hm.keySet()) {
			if (pq.size()<k) pq.add(new Value(key, hm.get(key)));
			else {
				while (pq.size() >= k && pq.peek().freq<hm.get(key)) {
					pq.poll();
				}
				if (pq.size()<k) pq.add(new Value(key, hm.get(key)));
			}
		}
		int ans[] = new int[k];
		int i = 0;
		for (Value v: pq) {
			ans[i++] = v.val;
		}
		return ans;
	}
}

class Value implements Comparable<Value> {
	int val, freq;

	public Value(int a, int b) {
		val = a;
		freq = b;
	}

	@Override
	public int compareTo(Value v2) {
		if (this.freq > v2.freq) return 1;
		else if (this.freq<v2.freq) return -1;
		else return this.freq - v2.freq;
	}

	@Override
	public String toString() {
		return "{val: " + this.val + ", freq: " + this.freq + "}";
	}
}
```

8. https://leetcode.com/problems/decode-string/

```
class Solution {
   public String decodeString(String s) {

      int n = s.length();
      Stack<Integer> numStack = new Stack<>();
      Stack<Integer> indexStack = new Stack<>();
      Stack<StringBuilder> strStack = new Stack<>();

      StringBuilder res = new StringBuilder();
      for (int i = 0; i < n; i++) {
         int h = 0;
         while (i < n && s.charAt(i) >= '0' && s.charAt(i) <= '9') {
            h = h * 10 + (s.charAt(i) - '0');
            i++;
         }
         if (h == 0) {
            res.append(s.charAt(i));
            continue;
         }
         i++;
         numStack.push(h);
         indexStack.push(i);
         strStack.push(new StringBuilder());

         while (!numStack.isEmpty()) {
            int index = indexStack.pop();
            int num = numStack.pop();
            StringBuilder str = strStack.pop();

            for (int j = index; j < n; j++) {
               char c = s.charAt(j);
               if (c == ']') {
                  if (numStack.isEmpty()) {
                     res.append(str.toString().repeat(num));
                     i = j;
                  }
                  else {
                     indexStack.pop();
                     StringBuilder rStr = strStack.pop();
                     indexStack.push(j + 1);
                     strStack.push(rStr.append(str.toString().repeat(num)));
                  }
                  break;
               }
               else if (c >= '0' && c <= '9') {
```

```
                numStack.push(num);
                indexStack.push(j);
                strStack.push(str);

                int l = 0;
                while (j < n
                    && s.charAt(j) >= '0'
                    && s.charAt(j) <= '9'
                ) {
                  l = l * 10 + (s.charAt(j) - '0');
                  j++;
                }
                j++;
                numStack.push(l);
                indexStack.push(j);
                strStack.push(new StringBuilder());
                break;
              }
              else {
                str.append(c);
              }
            }

          }
      }
      return res.toString();
    }
}
```

9. https://leetcode.com/problems/reorganize-string/

```
class Solution {
	public String reorganizeString(String s) {

		String result = "";

		int[] charFreq = new int[26];

		PriorityQueue<Pair> pq = new PriorityQueue<>();

		for (char c: s.toCharArray()) {
			charFreq[c - 'a']++;
		}

		Pair[] pair = new Pair[26];

		for (int i = 0; i<26; i++) {
			if (charFreq[i] != 0)
				pair[i] = new Pair((char)('a' + i), charFreq[i]);
		}

		for (int i = 0; i<26; i++) {
			if (pair[i] != null && pair[i].freq != 0) {
				pq.add(pair[i]);
			}
		}

		while (pq.size() >= 2) {

			Pair firstPair = pq.poll();
			Pair secondPair = pq.poll();

			result += firstPair.c;
			result += secondPair.c;

			firstPair.freq--;
			secondPair.freq--;

			if (firstPair.freq > 0) {
				pq.add(firstPair);
			}

			if (secondPair.freq > 0) {
				pq.add(secondPair);
			}
```

```
                }

                if (pq.size() >= 1) {
                        result += pq.poll().c;
                }

                if (result.length() != s.length())
                        return "";

                return result;
        }

        static class Pair implements Comparable<Pair> {

                char c;
                int freq;

                Pair(char c, int freq) {
                        this.c = c;
                        this.freq = freq;
                }

                public int compareTo(Pair p) {
                        return p.freq - this.freq;
                }
        }
}
```

10. https://leetcode.com/problems/construct-binary-tree-from-preorder-and-inorder-traversal/

```
class Solution {
	public TreeNode buildTree(int[] preorder, int[] inorder) {
		HashMap<Integer, Integer> inorder_map = populate_map(inorder);
		int[] pre_index = { 0 };
		return construct_tree(preorder, inorder_map, 0, inorder.length - 1, pre_index);

	}

	private HashMap<Integer, Integer> populate_map(int[] inorder) {
		HashMap<Integer, Integer> map = new HashMap<>();
		for (int i = 0; i<inorder.length; i++) {
			map.put(inorder[i], i);
		}
		return map;
	}

	private TreeNode construct_tree(int[] preorder, HashMap<Integer, Integer> inorder_map,
int start, int end, int[] pre_index) {
		if (start > end)
			return null;

		TreeNode root = new TreeNode(preorder[pre_index[0]]);
		int index = inorder_map.get(preorder[pre_index[0]]);
		pre_index[0]++;

		root.left = construct_tree(preorder, inorder_map, start, index - 1, pre_index);
		root.right = construct_tree(preorder, inorder_map, index + 1, end, pre_index);
		return root;
	}
}
```

11. https://leetcode.com/problems/lowest-common-ancestor-of-a-binary-tree/

```
class Solution {

	TreeNode resultNode = null;
	public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {
		helper(root, p, q);
		return this.resultNode;
	}

	public TreeNode helper(TreeNode node, TreeNode p, TreeNode q) {
		if (node == null) {
			return null;
		}

		TreeNode leftNode = helper(node.left, p, q);
		TreeNode rightNode = helper(node.right, p, q);
		if (((node == p || node == q) && (leftNode != null || rightNode != null)) || (leftNode
!= null && rightNode != null)) {
			this.resultNode = node;
			return node;
		} else if ((leftNode != null || rightNode != null) || (node == p || node == q)) {
			return node;
		} else {
			return null;
		}
	}
}
```

12. https://leetcode.com/problems/minimum-window-substring/

```
class Solution {
	public String minWindow(String s, String t) {
		if (t.length() > s.length()) return "";
		int m = t.length();
		int n = s.length();
		int[] freq = new int[58];
		int[] count = new int[58];
		String S = "";
		int window = n + 1;
		for (int i = 0; i<m; i++) {
			freq[t.charAt(i) - 'A']++;
		}
		int l = 0;
		for (int r = 0; r<n; r++) {
			count[s.charAt(r) - 'A']++;
			while (compare(count, freq)) {
				if (r - l + 1<window) {
					S = s.substring(l, r + 1);
					window = r - l + 1;
				}
				count[s.charAt(l) - 'A']--;
				l++;
			}

		}
		return S;
	}
	public boolean compare(int[] count, int[] freq) {
		for (int i = 0; i<freq.length; i++) {
			if (count[i]<freq[i]) return false;
		}
		return true;

	}
}
```

13. https://leetcode.com/problems/best-time-to-buy-and-sell-stock-iii/

```
class Solution {
	public int maxProfit(int[] prices) {
		int maxProfit = 0;
		int[] dp1 = dpIncreasing(prices);
		int[] dp2 = dpDecreasing(prices);
		for (int i = 1; i<prices.length; i++) {
			int l1 = dp1[i];
			int l2 = dp2[i];
			maxProfit = Math.max(maxProfit, l1 + l2);
		}
		return maxProfit;
	}

	public int[] dpIncreasing(int[] prices) {
		int[] dp = new int[prices.length];
		int minIndex = 0;
		int maxDiff = 0;
		for (int i = 0; i<prices.length; i++) {
			if (prices[i] - prices[minIndex] > maxDiff) {
				maxDiff = prices[i] - prices[minIndex];
			}
			if (prices[minIndex] > prices[i]) {
				minIndex = i;
			}
			dp[i] = maxDiff;
		}
		return dp;
	}

	public int[] dpDecreasing(int[] prices) {
		int[] dp = new int[prices.length];
		int maxIndex = prices.length - 1;
		int maxDiff = 0;
		for (int i = prices.length - 1; i >= 0; i--) {
			if (prices[maxIndex] - prices[i] > maxDiff) {
				maxDiff = prices[maxIndex] - prices[i];
			}
			if (prices[maxIndex]<prices[i]) {
				maxIndex = i;
			}
			dp[i] = maxDiff;
		}
		return dp;
	}
}
```

14. https://leetcode.com/problems/integer-to-english-words/submissions/

```
class Solution {
	private static final int[] INT_NUMBERS = {
		1 _000_000_000, 1 _000_000, 1000, 100, 90, 80, 70, 60, 50, 40, 30, 20, 19, 18, 17, 16,
15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
	};
	private static final String[] STRING_NUMBERS = {
		"Billion", "Million", "Thousand", "Hundred", "Ninety", "Eighty", "Seventy", "Sixty",
"Fifty", "Forty", "Thirty", "Twenty",
		"Nineteen", "Eighteen", "Seventeen", "Sixteen", "Fifteen", "Fourteen", "Thirteen",
"Twelve", "Eleven", "Ten",
		"Nine", "Eight", "Seven", "Six", "Five", "Four", "Three", "Two", "One"
	};

	public String numberToWords(int num) {
		if (num == 0) return "Zero";
		return numberToWordsHelper(num).toString();
	}

	private StringBuilder numberToWordsHelper(int num) {
		StringBuilder sb = new StringBuilder();
		if (num == 0) return sb;
		for (int i = 0; i<INT_NUMBERS.length; i++) {
			if (num >= INT_NUMBERS[i]) {
				if (num >= 100) {
					sb.append(numberToWordsHelper(num /
INT_NUMBERS[i]).append(" "));
				}

				sb.append(STRING_NUMBERS[i]).append("
").append(numberToWordsHelper(num % INT_NUMBERS[i]));
				break;
			}
		}
		return sb.charAt(sb.length() - 1) == ' ' ? sb.deleteCharAt(sb.length() - 1) : sb; // trim
	}
}
```

15. https://leetcode.com/problems/concatenated-words/

```
class Solution {
   public List<String> findAllConcatenatedWordsInADict(String[] words) {
      Set<String> allWords = new HashSet<>();
      for(String word: words) allWords.add(word);

      List<String> output = new ArrayList<>();
      for(String word: words){
         StringBuilder sb = new StringBuilder();
         Queue<Integer> startPoints = new LinkedList<>();
         Queue<Integer> seen = new LinkedList<>();
         startPoints.add(0);
         boolean isDone = false;
         while(!startPoints.isEmpty() && !isDone){
            int startPoint = startPoints.poll();
            if(!seen.contains(startPoint)){
               seen.add(startPoint);
               for(int index=startPoint+1; index<=word.length(); index++){
                  String subWord = word.substring(startPoint,index);
                  if(allWords.contains(subWord)){
                     if(index==word.length() && startPoint!=0){
                        output.add(word);
                        isDone = true;
                        break;
                     }
                     startPoints.add(index);
                  }
               }
            }
         }
      }
      return output;
   }
}
```

16. https://leetcode.com/problems/two-sum/

```
class Solution {
    public int[] twoSum(int[] nums, int target) {

        int[] res = new int[2];
        Map<Integer, Integer> map = new HashMap<>();

        for (int i = 0 ; i < nums.length ; i++) {
            map.put(nums[i], i);
        }

        for(int j = 0; j < nums.length; j++){
            if(map.containsKey(target-nums[j]) && j!=map.get(target-nums[j])){
                res[0] = j;
                res[1] = map.get(target-nums[j]);
                break;
            }
        }

        return res;
    }
}
```

17. https://leetcode.com/problems/majority-element/

```
class Solution {
   public int majorityElement(int[] nums) {
     Arrays.sort(nums);

                    int prev = -1;
                    int count = 0;
                    for (int i = 0; i < nums.length; i++) {
                              if (prev != nums[i]) {
                                        count = 1;
                                        prev = nums[i];
                              } else {
                                        count++;
                              }

                              if (count > nums.length / 2) {
                                        return prev;
                              }
                    }

                    return -1;
   }
}
```

18. https://leetcode.com/problems/spiral-matrix/

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

		int row = matrix.length;
		int col = matrix[0].length;

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

		int T = 0, B = row - 1, R = col - 1, L = 0, dir = 0;

		while (T<= B && L<= R) {

			if (dir == 0) {
				for (int i = L; i<= R; i++)
					result.add(matrix[T][i]);
				T++;
				dir = 1;

			} else if (dir == 1) {
				for (int i = T; i<= B; i++)
					result.add(matrix[i][R]);
				R--;
				dir = 2;

			} else if (dir == 2) {
				for (int i = R; i >= L; i--)
					result.add(matrix[B][i]);
				B--;
				dir = 3;

			} else if (dir == 3) {
				for (int i = B; i >= T; i--)
					result.add(matrix[i][L]);
				L++;
				dir = 0;

			}
		}

		System.out.println("result : " + result);
		return result;

	}
}
```

19. https://leetcode.com/problems/product-of-array-except-self/

```
class Solution {
   public int[] productExceptSelf(int[] nums) {
      int n = nums.length;
   int[] res = new int[n];
   res[0] = 1;
   for (int i = 1; i < n; i++) {
      res[i] = res[i - 1] * nums[i - 1];
   }
   int right = 1;
   for (int i = n - 1; i >= 0; i--) {
      res[i] *= right;
      right *= nums[i];
   }
   return res;
   }
}
```

20. https://leetcode.com/problems/word-search/

```
class Solution {
   public boolean exist(char[][] board, String word) {
      int []flag=new int[1];
      int [][]vis=new int[board.length][board[0].length];
      for(int i=0;i<board.length;i++){
         for(int j=0;j<board[0].length;j++){
            if(board[i][j]==word.charAt(0)){
               check(i,j,board,word,0,flag,vis);
            }
         }
      }
      if(flag[0]==1){
         return true;
      }else{
       return false;
      }
   }
   public void check(int sr,int sc,char[][] board,String word,int idx,int []flag,int [][]vis){
      if(idx==word.length()){
        flag[0]=1;
        return;
      }

if(sr<0||sc<0||sr>=board.length||sc>=board[0].length||vis[sr][sc]==1||board[sr][sc]!=word.charAt
(idx)){
         return;
      }
      vis[sr][sc]=1;
      check(sr+1,sc,board,word,idx+1,flag,vis);
      check(sr,sc+1,board,word,idx+1,flag,vis);
      check(sr-1,sc,board,word,idx+1,flag,vis);
      check(sr,sc-1,board,word,idx+1,flag,vis);
      vis[sr][sc]=0;
   }
}
```

21. https://leetcode.com/problems/find-the-duplicate-number/

```
class Solution {
	public int findDuplicate(int[] nums) {
		int slow = 0;
		int fast = 0;
		do {
			slow = nums[slow];
			fast = nums[nums[fast]];
		} while (slow != fast);

		slow = 0;
		while (slow != fast) {
			slow = nums[slow];
			fast = nums[fast];
		}

		return slow;
	}
}
```

22. https://leetcode.com/problems/k-diff-pairs-in-an-array/

```
class Solution {
	public int findDuplicate(int[] nums) {
		int slow = 0;
		int fast = 0;
		do {
			slow = nums[slow];
			fast = nums[nums[fast]];
		} while (slow != fast);

		slow = 0;
		while (slow != fast) {
			slow = nums[slow];
			fast = nums[fast];
		}

		return slow;
	}
}
```

23. https://leetcode.com/problems/subarray-sums-divisible-by-k/

```
class Solution {
	public int subarraysDivByK(int[] nums, int k) {
		HashMap<Integer, Integer> map = new HashMap<>();
		int sum = 0;
		int temp;
		int res = 0;
		map.put(0, 1);
		for (int i = 0; i<nums.length; i++) {
			sum += nums[i];
			temp = sum % k;
			if (temp<0) temp += k;
			if (map.containsKey(temp)) {
				res += map.get(temp);
			}
			map.put(temp, map.getOrDefault(temp, 0) + 1);

		}
		return res;
	}
}
```

24. https://leetcode.com/problems/first-missing-positive/

```
class Solution {
	public int firstMissingPositive(int[] nums) {
		int n = nums.length;

		if (n == 1 && nums[0] == 1) {
			return 2;
		} else if (n == 1 && nums[0]<0) {
			return 1;
		}

		HashSet<Integer> set = new HashSet<>();

		for (int elm: nums) {
			if (elm >= 0 && !set.contains(elm)) {
				set.add(elm);
			}
		}

		if (set.size() == 0) {
			return 1;
		}

		int missnum = 0;
		for (int i = 1; i<= nums.length + 1; i++) {
			if (!set.contains(i)) {
				missnum = i;
				break;
			}
		}
		return missnum;
	}
}
```

25. https://leetcode.com/problems/max-value-of-equation/

```
class Solution {
	public int findMaxValueOfEquation(int[][] points, int k) {
		int max = Integer.MIN_VALUE;
		int ans = 0;
		int mod = 0;
		int flag = 1;
		for (int i = 0; i<points.length - 1; i++) {
			if (flag<i + 1)
				flag = i + 1;
			for (int j = flag; j<points.length; j++) {
				mod = points[i][0] - points[j][0];
				if (mod<0)
					mod = -mod;
				if (mod > k)
					break; // x coordinate are sorted
				ans = points[i][1] + points[j][1] + mod;
				if (max<ans) {
					max = ans;
					flag = j - 1;
				}
			}
		}
		return max;
	}
}
```

26. https://leetcode.com/problems/word-break/

```
class Solution {
   public boolean wordBreak(String s, List<String> wordDict) {
      boolean[] f = new boolean[s.length() + 1];

      f[0] = true;
       for(int i=1; i <= s.length(); i++){
          for(int j=0; j < i; j++){
             if(f[j] && wordDict.contains(s.substring(j, i))){
                f[i] = true;
                break;
             }
          }
      }

      return f[s.length()];
   }
}
```

27. https://leetcode.com/problems/knight-dialer/

```
class Solution {
	public int knightDialer(int n) {
		var dp = new long[10];
		var tmp = new long[10];
		Arrays.fill(dp, 1);
		for (int i = 1; i<n; i++) {
			tmp[1] = dp[6] + dp[8];
			tmp[2] = dp[7] + dp[9];
			tmp[3] = dp[4] + dp[8];
			tmp[4] = dp[0] + dp[3] + dp[9];
			tmp[5] = 0;
			tmp[6] = dp[0] + dp[1] + dp[7];
			tmp[7] = dp[2] + dp[6];
			tmp[8] = dp[1] + dp[3];
			tmp[9] = dp[2] + dp[4];
			tmp[0] = dp[4] + dp[6];
			for (int j = 0; j<10; j++) tmp[j] = tmp[j] % 1000000007;
			var arr = dp;
			dp = tmp;
			tmp = arr;
		}
		long res = 0;
		for (int i = 0; i<10; i++) {
			res = (res + dp[i]) % 1000000007;
		}
		return (int) res;
	}
}
```

28. https://leetcode.com/problems/unique-paths/

```
class Solution {
   public int uniquePaths(int m, int n) {

      int[][] dp = new int[m][n];


      for (int i = 0 ; i < m ; i++) {
         for (int j = 0 ; j < n ; j++) {
            if (i ==0 || j == 0) {
               dp[i][j] = 1;
            } else {
               dp[i][j] = dp[i][j-1] + dp[i-1][j];
            }
         }
      }

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

   }
}
```

29. https://leetcode.com/problems/longest-arithmetic-subsequence/

```
class Solution {
	public int longestArithSeqLength(int[] nums) {
		int[][] dp = new int[nums.length][1001];
		for (int i = 0; i<nums.length; i++) Arrays.fill(dp[i], 1);
		int offset = 500;
		int maxAns = 0;
		for (int i = 1; i<nums.length; i++) {
			for (int j = 0; j<i; j++) {
				int diff = nums[j] - nums[i];
				int offseted = diff + offset;
				dp[i][offseted] = Math.max(dp[i][offseted], dp[j][offseted] + 1);
				maxAns = Math.max(maxAns, dp[i][offseted]);
			}
		}
		return maxAns;
	}
}
```

30. https://leetcode.com/problems/regular-expression-matching/

```
class Solution {
	public boolean isMatch(String s, String p) {
		Boolean[][] memo = new Boolean[s.length() + 1][p.length() + 1];
		return find(s, p, 0, 0, memo);
	}

	private boolean find(String s, String p, int i, int j, Boolean[][] memo) {

		if (memo[i][j] != null) return memo[i][j];

		if (i >= s.length() || j >= p.length()) {
			if (i<s.length()) return false;

			for (; j<p.length(); j++) {
				if (j<p.length() - 1 && p.charAt(j + 1) == '*') {
					continue;
				}
				if (p.charAt(j) != '*') {
					return false;
				}
			}
			return true;
		}

		boolean isMatch = s.charAt(i) == p.charAt(j) || p.charAt(j) == '.';

		if (j<p.length() - 1 && p.charAt(j + 1) == '*') {
			if (isMatch) {
				memo[i][j] = find(s, p, i + 1, j, memo) || find(s, p, i, j + 2, memo);
			} else {
				memo[i][j] = find(s, p, i, j + 2, memo);
			}
		} else {
			if (isMatch) {
				memo[i][j] = find(s, p, i + 1, j + 1, memo);
			} else {
				memo[i][j] = false;
			}
		}
		return memo[i][j];
	}
}
```

31. https://leetcode.com/problems/longest-valid-parentheses/

```
class Solution {
   public int longestValidParentheses(String s) {
       Stack<Integer> stack = new Stack<>();
      stack.push(-1);
      int res = 0;
      for(int i=0;i<s.length();i++){
         char c=s.charAt(i);
         if(c==')' && stack.peek()!=-1 && s.charAt(stack.peek())=='('){
            stack.pop();
            res = Math.max(res, i-stack.peek());
         }else{
            stack.push(i);
         }
      }
      return res;
   }
}
```

32. https://leetcode.com/problems/minimum-difficulty-of-a-job-schedule/

```
class Solution {
	public int minDifficulty(int[] jobDifficulty, int d) {
		if (jobDifficulty.length<d) return -1;
		int[][] dp = new int[jobDifficulty.length][d];
		int[][] rangeMax = getRangeMax(jobDifficulty);
		for (int i = 0; i<jobDifficulty.length; i++) dp[i][0] = rangeMax[0][i];
		for (int cut = 1; cut<d; cut++) {
			for (int i = cut; i<jobDifficulty.length; i++) {
				dp[i][cut] = Integer.MAX_VALUE;
				for (int j = cut - 1; j<i; j++) {
					dp[i][cut] = Math.min(dp[i][cut], dp[j][cut - 1] + rangeMax[j
+ 1][i]);
				}
			}
		}
		return dp[jobDifficulty.length - 1][d - 1];
	}

	private int[][] getRangeMax(int[] arr) {
		int[][] rangeMax = new int[arr.length][arr.length];
		for (int i = 0; i<arr.length; i++) {
			for (int j = i; j<arr.length; j++) {
				rangeMax[i][j] = i == j ? arr[j] : Math.max(rangeMax[i][j - 1], arr[j]);
			}
		}
		return rangeMax;
	}

}
```

33. https://leetcode.com/problems/minimum-cost-to-cut-a-stick/

```
class Solution {
	public int minCost(int n, int[] cuts) {
		int c = cuts.length;

		ArrayList<Integer> arr = new ArrayList<>();
		arr.add(0);
		for (int i = 0; i<cuts.length; i++)
			arr.add(cuts[i]);
		arr.add(n);
		Collections.sort(arr);
		int dp[][] = new int[c + 2][c + 2];
		for (int i = c; i >= 1; i--) {
			for (int j = 1; j<= c; j++) {
				if (i > j) continue;
				int min = Integer.MAX_VALUE;
				for (int k = i; k<= j; k++) {
					int cost = dp[i][k - 1] + dp[k + 1][j] +
						arr.get(j + 1) - arr.get(i - 1);
					min = Math.min(min, cost);
				}
				dp[i][j] = min;
			}
		}
		return dp[1][c];
	}
}
```

34. https://leetcode.com/problems/implement-strstr/

```
class Solution {
	public int strStr(String haystack, String needle) {
		haystack = haystack.replaceAll(needle, "-");
		for (int i = 0; i<haystack.length(); i++) {
			if (haystack.charAt(i) == '-')
				return i;
		}
		return -1;
	}
}
```

35. https://leetcode.com/problems/minimum-remove-to-make-valid-parentheses/

```
class Solution {
	public String minRemoveToMakeValid(String s) {
		StringBuilder sb = new StringBuilder(s);
		Stack<Integer> st = new Stack<>();
		for (int i = 0; i<sb.length(); ++i) {
			if (sb.charAt(i) == '(') st.add(i);
			if (sb.charAt(i) == ')') {
				if (!st.empty()) st.pop();
				else sb.setCharAt(i, '*');
			}
		}
		while (!st.empty())
			sb.setCharAt(st.pop(), '*');
		return sb.toString().replaceAll("\\*", "");
	}
}
```

36. https://leetcode.com/problems/basic-calculator-ii/

```
class Solution {
	public int calculate(String s) {
		if (s == null || s.length() == 0) return 0;
		int curr = 0;
		char op = '+';
		char[] ch = s.toCharArray();
		int sum = 0;
		int last = 0;
		for (int i = 0; i<ch.length; i++) {
			if (Character.isDigit(ch[i])) {
				curr = curr * 10 + ch[i] - '0';
			}
			if (!Character.isDigit(ch[i]) && ch[i] != ' ' || i == ch.length - 1) {
				if (op == '+') {
					sum += last;
					last = curr;
				} else if (op == '-') {
					sum += last;
					last = -curr;
				} else if (op == '*') {
					last = last * curr;
				} else if (op == '/') {
					last = last / curr;
				}
				op = ch[i];
				curr = 0;
			}
		}
		sum += last;
		return sum;
	}
}
```

37. https://leetcode.com/problems/power-of-two/

```
class Solution {
   public boolean isPowerOfTwo(int n) {
      return n > 0 && (n == 1 || (n%2 == 0 && isPowerOfTwo(n/2)));
   }
}
```

38. https://leetcode.com/problems/string-to-integer-atoi/

```
class Solution {
	public int myAtoi(String s) {
		if (s == null || s.isEmpty() || s.length() > 200) return 0;

		int length = s.length();
		int signDriver = 1;
		int currentIndex = 0;
		int result = 0;
		int maximumLimit = Integer.MAX_VALUE / 10;
		while (currentIndex<length && s.charAt(currentIndex) == ' ')
			currentIndex++;

		if (currentIndex<length) {
			if (s.charAt(currentIndex) == '-') {
				signDriver *= -1;
				currentIndex++;
			} else if (s.charAt(currentIndex) == '+') {
				currentIndex++;
			}
		}

		while (currentIndex<length && isCharacterADigit(s.charAt(currentIndex))) {
			int currentDigit = s.charAt(currentIndex) - '0';

			if (result > maximumLimit || (result == maximumLimit && currentDigit > 7))
				return signDriver == -1 ? Integer.MIN_VALUE : Integer.MAX_VALUE;

			result = result * 10 + currentDigit;
			currentIndex++;
		}
		return signDriver * result;
	}

	private boolean isCharacterADigit(char character) {
		int possibleDigit = character - '0';

		return possibleDigit >= 0 && possibleDigit<= 9 ? true : false;
	}
}
```

39. https://leetcode.com/problems/max-points-on-a-line/

```
class Solution {
    public int maxPoints(int[][] points) {
      int max = 0;
      for(int i = 0; i < points.length; ++i){
         int vertical = 0;
         int horizontal = 0;
         HashMap<Double,Integer> map = new HashMap();
         for(int j = 0; j<points.length; ++j){
            if(i != j){
               if(points[i][0] == points[j][0]){
                  vertical++;
                  max = Math.max(vertical,max);
               }
               else if(points[i][1] == points[j][1]){
                  horizontal++;
                  max = Math.max(horizontal,max);
               }
               else{
               double gradient = (points[i][1] - points[j][1])*1.00/(points[i][0] - points[j][0]);
               max = Math.max(max,map.getOrDefault(gradient,0)+1);
               map.put(gradient,map.getOrDefault(gradient,0)+1);}
            }
         }
      }

      return max + 1;
    }
}
```

40. https://leetcode.com/problems/remove-k-digits/

```
class Solution {
	public String removeKdigits(String num, int k) {
		Stack<Integer> stack = new Stack<>();

		for (char ch: num.toCharArray()) {
			int currNum = ch - '0';
			while (!stack.isEmpty() && stack.peek() > currNum && k > 0) {
				stack.pop();
				k--;
			}
			stack.push(currNum);
		}
		while (k--> 0) stack.pop();

		StringBuilder sbr = new StringBuilder();
		while (!stack.isEmpty()) sbr.append(stack.pop());
		sbr.reverse();
		while (sbr.length() > 1 && sbr.charAt(0) == '0') sbr.deleteCharAt(0);

		return sbr.toString().length() > 0 ? sbr.toString() : "0";
	}
}
```

41. https://leetcode.com/problems/clone-graph/

```
class Solution {

    public Node cloneGraph(Node node) {
        if (node == null) return null;
        Node copy = new Node(node.val);
        Node[] visited = new Node[101];
        Arrays.fill(visited, null);
        dfs(node, copy, visited);
        return copy;
    }

    public void dfs(Node node, Node copy, Node[] visited) {
        visited[copy.val] = copy;
        for (Node n: node.neighbors) {
            if (visited[n.val] == null) {
                Node newNode = new Node(n.val);
                copy.neighbors.add(newNode);
                dfs(n, newNode, visited);
            } else {
                copy.neighbors.add(visited[n.val]);
            }
        }

    }

}
```

42. https://leetcode.com/problems/house-robber-iii/

```
class Solution {
	public int rob(TreeNode root) {
		Map<TreeNode, Integer> map = new HashMap<>();
		return helper(root, map);
	}
	int helper(TreeNode root, Map<TreeNode, Integer> map) {
		if (root == null)
			return 0;
		if (map.get(root) != null)
			return map.get(root);
		int taken = root.val;
		if (root.left != null) {
			taken += helper(root.left.left, map);
			taken += helper(root.left.right, map);
		}
		if (root.right != null) {
			taken += helper(root.right.left, map);
			taken += helper(root.right.right, map);
		}
		int notTaken = 0;
		notTaken += helper(root.left, map);
		notTaken += helper(root.right, map);
		int ans = Math.max(taken, notTaken);
		map.put(root, ans);
		return ans;
	}
}
```

43. https://leetcode.com/problems/critical-connections-in-a-network/

```
class Solution {
	public void dfs(int node, ArrayList<ArrayList<Integer>> graph, int timer, int parent,
boolean[] vis, int[] tin, int[] low, List<List<Integer>> critical_connections) {
		vis[node] = true;
		tin[node] = low[node] = ++timer;
		Iterator<Integer> itr = graph.get(node).listIterator();
		while (itr.hasNext()) {
			int neigh = itr.next();
			if (neigh == parent) continue;
			if (!vis[neigh]) {
				dfs(neigh, graph, timer, node, vis, tin, low, critical_connections);
				low[node] = Math.min(low[node], low[neigh]);
				if (low[neigh] > tin[node]) {
					ArrayList<Integer> connection = new ArrayList<>();
					connection.add(node);
					connection.add(neigh);
					critical_connections.add(connection);
				}
			} else {
				low[node] = Math.min(low[node], tin[neigh]);
			}
		}
	}
	public List<List<Integer>> criticalConnections(int n, List<List<Integer>> connections) {
		ArrayList<ArrayList<Integer>> graph = new ArrayList<>();
		for (int i = 0; i<n; i++) graph.add(new ArrayList<Integer> ());
		for (List<Integer> edge: connections) {
			int u = edge.get(0), v = edge.get(1);
			graph.get(u).add(v);
			graph.get(v).add(u);
		}
		List<List<Integer>> critical_connections = new ArrayList<>();
		boolean[] vis = new boolean[n];
		int[] tin = new int[n];
		int[] low = new int[n];
		int parent = -1;
		int timer = 0;

		for (int i = 0; i<n; i++) {
			if (!vis[i]) {
				dfs(i, graph, timer, parent, vis, tin, low, critical_connections);
			}
		}
		return critical_connections;
	}
}
```

44. https://leetcode.com/problems/diameter-of-binary-tree/

```
class Solution {
	int max = 0;
	public int heightTree(TreeNode root) {
		if (root == null)
			return 0;
		int lh = (heightTree(root.left));
		int rh = (heightTree(root.right));
		max = Math.max(max, lh + rh);
		return 1 + Math.max(lh, rh);
	}
	public int diameterOfBinaryTree(TreeNode root) {
		heightTree(root);
		return max;

	}
}
```

45. https://leetcode.com/problems/redundant-connection/

```
class DSU {
	int[] parent;
	int[] rank;
	DSU(int n) {
		parent = new int[n];
		Arrays.fill(parent, -1);
		rank = new int[n];
		Arrays.fill(rank, 1);
	}

	public void union(int a, int b) {
		int pa = find(a);
		int pb = find(b);
		if (pa == pb) return;
		if (rank[pa] > rank[pb]) {
			parent[pb] = pa;
			rank[pa] += rank[pb];
		} else {
			parent[pa] = pb;
			rank[pb] += rank[pa];
		}
		return;
	}
	public int find(int a) {
		if (parent[a] == -1) return a;
		return parent[a] = find(parent[a]);
	}
}
class Solution {
	public int[] findRedundantConnection(int[][] edges) {
		int n = edges.length;
		int[] ans = new int[2];
		DSU sets = new DSU(n + 1);
		for (int[] edge: edges) {
			int a = edge[0], b = edge[1];
			if (sets.find(a) == sets.find(b)) {
				ans[0] = edge[0];
				ans[1] = edge[1];
				return ans;
			}
			sets.union(a, b);
		}
		return null;
	}
}
```

46. https://leetcode.com/problems/redundant-connection-ii/

```
class UnionFind {
	int[] parents;
	int[] ranks;
	UnionFind(int n) {
		parents = new int[n];
		ranks = new int[n];
		Arrays.fill(ranks, 1);
		for (int i = 0; i<n; i++)
			parents[i] = i;
	}
	public int find(int v) {
		if (parents[v] == v)
			return v;
		return find(parents[v]);
	}
	public boolean union(int v1, int v2) {
		int ar1 = find(v1);
		int ar2 = find(v2);
		if (ar1 == ar2)
			return false;
		if (ranks[ar1] > ranks[ar2])
			parents[ar2] = ar1;
		else if (ranks[ar2] > ranks[ar1])
			parents[ar1] = ar2;
		else {
			parents[ar1] = ar2;
			ranks[ar2]++;
		}
		return true;
	}
}
class Solution {
	public int[] findRedundantDirectedConnection(int[][] edges) {
		Integer nodeWith2Indegree = getIndegreeTwo(edges);
		UnionFind uf = new UnionFind(edges.length + 1);
		if (nodeWith2Indegree == null) {
			for (int[] edge: edges) {
				if (!uf.union(edge[0], edge[1]))
					return edge;
			}
		} else {
			int[][] twoEdges = new int[2][2];
			int top = -1;
```

```
                        for (int[] edge: edges) {
                                if (edge[1] == nodeWith2Indegree)
                                        twoEdges[++top] = edge;
                                if (top == 1)
                                        break;
                        }
                        if (hasCycle(uf, edges, twoEdges[1]))
                                return twoEdges[0];
                        else
                                return twoEdges[1];
                }
                return null;
        }
        private boolean hasCycle(UnionFind uf, int[][] edges, int[] skipEdge) {
                for (int[] edge: edges) {
                        if (edge != skipEdge)
                                if (!uf.union(edge[0], edge[1]))
                                        return true;
                }
                return false;
        }
        private Integer getIndegreeTwo(int[][] edges) {
                int[] map = new int[edges.length + 1];
                for (int[] edge: edges)
                        if (++map[edge[1]] == 2)
                                return edge[1];
                return null;
        }
}
```

47. https://leetcode.com/problems/capacity-to-ship-packages-within-d-days/

```
class Solution {
	public int shipWithinDays(int[] weights, int days) {
		int n = weights.length;
		int ans = 0;

		int max = 0;
		int sum = 0;

		for (Integer i: weights) {
			sum += i;
			max = Math.max(max, i);
		}

		if (n == days) {
			return max;
		}

		int l = max;
		int h = sum;

		while (l<= h) {
			int mid = l + (h - l) / 2;
			if (isPossible(weights, mid, days) == true) {
				ans = mid;
				h = mid - 1;
			} else {
				l = mid + 1;
			}
		}
		return ans;
	}

	boolean isPossible(int weights[], int mid, int days) {
		int d = 1;
		int sum = 0;
		for (int i = 0; i<weights.length; i++) {
			sum += weights[i];
			if (sum > mid) {
				d++;
				sum = weights[i];
			}
		}
		return d<= days;
	}
}
```

48. https://leetcode.com/problems/rotting-oranges/

```
class Solution {
   public int orangesRotting(int[][] grid) {
      if(grid == null || grid.length == 0) return 0;
      int rows = grid.length;
      int cols = grid[0].length;
      Queue<int[]> queue = new LinkedList<>();
      int count_fresh = 0;

      for(int i = 0 ; i < rows ; i++) {
         for(int j = 0 ; j < cols ; j++) {
            if(grid[i][j] == 2) {
               queue.offer(new int[]{i , j});
            }
            else if(grid[i][j] == 1) {
               count_fresh++;
            }
         }
      }
      if(count_fresh == 0) return 0;
      int count = 0;
      int[][] dirs = {{1,0},{-1,0},{0,1},{0,-1}};
      while(!queue.isEmpty()) {
         ++count;
         int size = queue.size();
         for(int i = 0 ; i < size ; i++) {
            int[] point = queue.poll();
            for(int dir[] : dirs) {
               int x = point[0] + dir[0];
               int y = point[1] + dir[1];
               if(x < 0 || y < 0 || x >= rows || y >= cols || grid[x][y] == 0 || grid[x][y] == 2) continue;
               grid[x][y] = 2;
               queue.offer(new int[]{x , y});
               count_fresh--;
            }
         }
      }
      return count_fresh == 0 ? count-1 : -1;
   }
}
```

49. https://leetcode.com/problems/longest-repeating-character-replacement/

```
class Solution {
	public int characterReplacement(String s, int k) {

		if (k == s.length() || k - 1 == s.length()) {
			return s.length();
		}

		HashMap<Character, Integer> map = new HashMap<>();

		int l = 0, size = 0, freq = 0;

		for (int i = 0; i<s.length(); i++) {
			if (!(map.containsKey(s.charAt(i)))) {
				map.put(s.charAt(i), 1);
			} else {
				map.put(s.charAt(i), map.get(s.charAt(i)) + 1);
			}
			freq = Math.max(freq, map.get(s.charAt(i)));
			if (i - l + 1 - freq > k) {
				map.put(s.charAt(l), map.get(s.charAt(l)) - 1);
				l++;
			}

			size = Math.max(size, i - l + 1);
		}
		return size;
	}
}
```

50. https://leetcode.com/problems/lru-cache/

```
class LRUCache {

	Node head = new Node(0, 0);
	Node tail = new Node(0, 0);
	int capacity;

	HashMap<Integer, Node> map = new HashMap<>();

	public LRUCache(int capacity) {
		this.capacity = capacity;
		head.next = tail;
		tail.prev = head;
	}

	public int get(int key) {

		if (map.containsKey(key)) {
			Node node = map.get(key);

			remove(node);
			insert(node);

			return node.value;
		} else {
			return -1;
		}

	}

	public void insert(Node node) {
		map.put(node.key, node);

		Node headNext = head.next;

		node.prev = head;
		head.next = node;
		node.next = headNext;
		headNext.prev = node;
	}

	public void remove(Node node) {
		map.remove(node.key);

		node.prev.next = node.next;
		node.next.prev = node.prev;
```

```
	}

	public void put(int key, int value) {

		if (map.containsKey(key)) {
			remove(map.get(key));
		}
		if (map.size() == capacity) {
			remove(tail.prev);
		}

		insert(new Node(key, value));

	}

	class Node {
		Node prev, next;
		int key, value;

		public Node(int key, int value) {
			this.key = key;
			this.value = value;
		}

	}
}
```