Problem Solving 19/11/2024

1. Next Permutation
   Time complexity: O(n)
   Space complexity: O(1)

```java
package program19thNov;
import java.util.Scanner;
public class NextPermutation {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = scanner.nextInt();
        int[] nums = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        Solution solution = new Solution();
        solution.nextPermutation(nums);
        for (int num : nums) {
            System.out.print(num + " ");
        }
        scanner.close();
    }
}
class Solution {
    public void nextPermutation(int[] nums) {
        int ind1 = -1;
        int ind2 = -1;
        for (int i = nums.length - 2; i >= 0; i--) {
            if (nums[i] < nums[i + 1]) {
                ind1 = i;
                break;
            }
        }
        if (ind1 == -1) {
            reverse(nums, 0);
        } else {
            for (int i = nums.length - 1; i >= 0; i--) {
                if (nums[i] > nums[ind1]) {
                    ind2 = i;
                    break;
                }
            }
            swap(nums, ind1, ind2);
            reverse(nums, ind1 + 1);
        }
    }
    void swap(int[] nums, int i, int j) {
        int temp = nums[i];
        nums[i] = nums[j];
        nums[j] = temp;
    }
    void reverse(int[] nums, int start) {
        int i = start;
        int j = nums.length - 1;
        while (i < j) {
            swap(nums, i, j);
            i++;
            j--;
        }
```

```
<terminated> NextPermutation [Java Application
Enter the size of the array: 5
Enter the elements of the array:
2 5 6 4 1
2 6 1 4 5
```

2. Longest Substring Without Repeating Characters
   Time complexity: O(n)
   Space complexity: O(min(n, m))

```java
package program19thNov;
import java.util.*;
public class LongestSubstringWithoutRepeating {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String input = scanner.nextLine();
        LongestSubstringWithoutRepeating solution = new LongestSubstringWithoutRepeating();
        int maxLength = solution.lengthOfLongestSubstring(input);
        System.out.println("Longest substring without repeating characters: " + maxLength);
        scanner.close();
    }
    public int lengthOfLongestSubstring(String s) {
        int left = 0;
        int maxLength = 0;
        HashSet<Character> charSet = new HashSet<>();
        for (int right = 0; right < s.length(); right++) {
            while (charSet.contains(s.charAt(right))) {
                charSet.remove(s.charAt(left));
                left++;
            }
            charSet.add(s.charAt(right));
            maxLength = Math.max(maxLength, right - left + 1);
        }
        return maxLength;
    }
}

```

```
<terminated> LongestSubstringWithoutRepeating [Java Application] C:\Program Files\Ja
Enter a string: askjdfhaksljdh
Longest substring without repeating characters: 8
```

3. Remove Elements from Linked List

   Time complexity: O(n)

   Space complexity: O(1)

```java
package program19thNov;
import java.util.Scanner;
public class RemoveElementsFromLinkedList {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements ");
        int n = scanner.nextInt();
        System.out.println("Enter the elements of the linked list:");
        ListNode head = null;
        ListNode current = null;
        for (int i = 0; i < n; i++) {
            int value = scanner.nextInt();
            if (head == null) {
                head = new ListNode(value);
                current = head;
            } else {
                current.next = new ListNode(value);
                current = current.next;
            }
        }
        System.out.print("Enter the value to be removed: ");
        int val = scanner.nextInt();
        RemoveElementsFromLinkedList solution = new RemoveElementsFromLinkedList();
        ListNode updatedHead = solution.removeElements(head, val);
        solution.printList(updatedHead);
        scanner.close();
    }
    public ListNode removeElements(ListNode head, int val) {
        ListNode ans = new ListNode(0, head);
        ListNode dummy = ans;
        while (dummy != null) {
            while (dummy.next != null && dummy.next.val == val) {
                dummy.next = dummy.next.next;
            }
            dummy = dummy.next;
        }
        return ans.next;
    }
    private void printList(ListNode head) {
        ListNode current = head;
        while (current != null) {
            System.out.print(current.val + " ");
            current = current.next;
        }
        System.out.println();
    }
}
class ListNode {
    int val;
    ListNode next;
    ListNode(int val) {
        this.val = val;
        this.next = null;
    }
    ListNode(int val, ListNode next) {
        this.val = val;
        this.next = next;
    }
}
```

```
<terminated> RemoveElementsFromLinkedList [Java Application] C:\Prog
Enter the number of elements 8
Enter the elements of the linked list:
1 2 4 5 5 6 7 7
Enter the value to be removed: 5
1 2 4 6 7 7
```

4. Palindrome Linked List

   Time complexity: O(n)

   Space complexity: O(n)

```java
package program19thNov;
import java.util.*;
public class PalindromeLinkedList {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of elements in the linked list: ");
        int n = scanner.nextInt();
        System.out.println("Enter the elements :");
        ListNode head = null;
        ListNode current = null;
        for (int i = 0; i < n; i++) {
            int value = scanner.nextInt();
            if (head == null) {
                head = new ListNode(value);
                current = head;
            } else {
                current.next = new ListNode(value);
                current = current.next;
            }
        }
        PalindromeLinkedList solution = new PalindromeLinkedList();
        boolean isPalindrome = solution.isPalindrome(head);
        System.out.println(isPalindrome);
        scanner.close();
    }
    public boolean isPalindrome(ListNode head) {
        List<Integer> list = new ArrayList<>();
        while (head != null) {
            list.add(head.val);
            head = head.next;
        }
        int left = 0;
        int right = list.size() - 1;
        while (left < right && list.get(left).equals(list.get(right))) {
            left++;
            right--;
        }
        return left >= right;}}

```

```
<terminated> PalindromeLinkedList (1) [Java Application] C:\Program Files\Java\bin\javaw.exe (19-
Enter the number of elements in the linked list: 8
Enter the elements :
2 1 4 5 6 7 9 3
false
```

5. Spiral Matix

Time complexity: O(n * m)

Space complexity: O(n * m)

```java
package program19thNov;
import java.util.*;
public class SpiralMatrix {
        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);
            System.out.println("Enter the number of rows:");
            int rows = scanner.nextInt();
            System.out.println("Enter the number of columns:");
            int cols = scanner.nextInt();
            int[][] mat = new int[rows][cols];
            System.out.println("Enter the matrix values:");
            for (int i = 0; i < rows; i++) {
                for (int j = 0; j < cols; j++) {
                    mat[i][j] = scanner.nextInt();
                }
            }
            List<Integer> ans = printSpiral(mat);
            for (int num : ans) {
                System.out.print(num + " ");
            }
            System.out.println();
            scanner.close();
        }
        public static List<Integer> printSpiral(int[][] mat) {
            List<Integer> ans = new ArrayList<>();
            int n = mat.length;
            int m = mat[0].length;
            int top = 0, left = 0, bottom = n - 1, right = m - 1;

            while (top <= bottom && left <= right) {
                for (int i = left; i <= right; i++)
                    ans.add(mat[top][i]);
                top++;

                for (int i = top; i <= bottom; i++)
                    ans.add(mat[i][right]);
                right--;

                if (top <= bottom) {
                    for (int i = right; i >= left; i--)
                        ans.add(mat[bottom][i]);
                    bottom--;
                }

                if (left <= right) {
                    for (int i = bottom; i >= top; i--)
                        ans.add(mat[i][left]);
                    left++;
                }
            }
            return ans;
        }
```

```
<terminated> SpiralMatrix [Java Application] C:\Program Files
Enter the number of rows:
3
Enter the number of columns:
3
Enter the matrix values:
1 2 3 4 5 6 7 8 9
1 2 3 6 9 8 7 4 5
```

6. Minimum Path Sum

Time complexity: O(n + m)

Space complexity: O(n + m)

```java
package program19thNov;
import java.util.Scanner;
public class MinimumPathSum {
    public static int minPathSum(int[][] grid) {
        int rows = grid.length;
        int cols = grid[0].length;
        int[][] res = new int[rows + 1][cols + 1];
        for (int r = 0; r <= rows; r++) {
            for (int c = 0; c <= cols; c++) {
                res[r][c] = Integer.MAX_VALUE;
            }
        }
        res[rows - 1][cols] = 0;
        for (int r = rows - 1; r >= 0; r--) {
            for (int c = cols - 1; c >= 0; c--) {
                res[r][c] = grid[r][c] + Math.min(res[r + 1][c], res[r][c + 1]);
            }
        }
        return res[0][0];
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of rows: ");
        int rows = scanner.nextInt();
        System.out.print("Enter the number of columns: ");
        int cols = scanner.nextInt();
        int[][] grid = new int[rows][cols];
        System.out.println("Enter the grid values row by row:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                grid[i][j] = scanner.nextInt();
            }
        }
        int result = minPathSum(grid);
        System.out.println("Minimum Path Sum: " + result);
    }
}
```

```
<terminated> MinimumPathSum [Java Application] C:\Progra
Enter the number of rows: 4
Enter the number of columns: 4
Enter the grid values row by row:
1 2 4 3
5 3 2 4
2 4 1 3
1 2 4 1
Minimum Path Sum: 13
```

7. Valid Binary Tree

Time complexity: O(n)

Space complexity: O(n)

```java
package program19thNov;
import java.util.Scanner;
public class ValidBinaryTree {
    static class TreeNode {
        int val;
        TreeNode left;
        TreeNode right;
        TreeNode(int x) {
            val = x;
        }
    }
    public boolean isValidBST(TreeNode root) {
        return valid(root, Long.MIN_VALUE, Long.MAX_VALUE);
    }
    private boolean valid(TreeNode node, long left, long right) {
        if (node == null) {
            return true;
        }
        if (node.val <= left || node.val >= right) {
            return false;
        }
        return valid(node.left, left, node.val) && valid(node.right, node.val, right);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of nodes: ");
        int n = scanner.nextInt();
        int[] values = new int[n];
        System.out.println("Enter values for nodes (In-order): ");
        for (int i = 0; i < n; i++) {
            values[i] = scanner.nextInt();
        }
        TreeNode root = buildTree(values, 0);
        ValidBinaryTree solution = new ValidBinaryTree();
        boolean result = solution.isValidBST(root);
        if (result) {
            System.out.println("The binary tree is a valid BST.");
        } else {
            System.out.println("The binary tree is not a valid BST.");
        }
        scanner.close();
    }
    public static TreeNode buildTree(int[] values, int index) {
        if (index >= values.length) {
            return null;
        }
        TreeNode node = new TreeNode(values[index]);
        node.left = buildTree(values, 2 * index + 1);
        node.right = buildTree(values, 2 * index + 2);
        return node;
    }
}
```

```
<terminated> ValidBinaryTree [Java Application] C:\Program Files\Java\
Enter the number of nodes: 3
Enter values for nodes (In-order):
5
4
3
The binary tree is not a valid BST.
```

8. Word Ladder

Time complexity: O(n * m)

Space complexity: O(n)

```java
package program19thNov;
import java.util.*;
public class WordLadder {
    public int ladderLength(String beginWord, String endWord, List<String> wordList) {
        Set<String> wordSet = new HashSet<>(wordList);
        if (beginWord.equals(endWord)) {
            return 1;        }
        Queue<String> queue = new LinkedList<>();
        queue.add(beginWord);
        int level = 1;
        while (!queue.isEmpty()) {
            int size = queue.size();
            level++;
            for (int i = 0; i < size; i++) {
                String currentWord = queue.poll();
                for (int j = 0; j < currentWord.length(); j++) {
                    char[] temp = currentWord.toCharArray();
                    for (char c = 'a'; c <= 'z'; c++) {
                        temp[j] = c;
                        String newWord = new String(temp);
                        if (newWord.equals(endWord)) {
                            return level;}
                        if (wordSet.contains(newWord)) {
                            queue.add(newWord);
                            wordSet.remove(newWord);
                        }}}}}return 0;}
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the begin word: ");
        String beginWord = scanner.nextLine();
        System.out.print("Enter the end word: ");
        String endWord = scanner.nextLine();
        System.out.print("Enter the number of words in the word list: ");
        int n = scanner.nextInt();
        List<String> wordList = new ArrayList<>();
        System.out.println("Enter words in the word list:");
        for (int i = 0; i < n; i++) {
            wordList.add(scanner.nextLine());
        }
        WordLadder solution = new WordLadder();
        int result = solution.ladderLength(beginWord, endWord, wordList);
        System.out.println(result);
        scanner.close();
    }
```

```
<terminated> WordLadder [Java Application] C:\Program Files\Java\bin\javaw.ex
Enter the begin word: hit
Enter the end word: cog
Enter the number of words in the word list: 5
Enter words in the word list:
hot
dot
dog
lot
5
```

9. Word Ladder 2

Time complexity: O(n * m * m)

Space complexity: O(n * m)

```java
package program19thNov;
import java.util.*;
public class WordLadder2 {
    public List<List<String>> findLadders(String beginWord, String endWord, List<String> wordList) {
        Map<String, Integer> hm = new HashMap<>();
        List<List<String>> res = new ArrayList<>();
        Queue<String> q = new LinkedList<>();
        q.add(beginWord);
        hm.put(beginWord, 1);
        HashSet<String> hs = new HashSet<>(wordList);
        hs.remove(beginWord);
        while (!q.isEmpty()) {
            String word = q.poll();
            if (word.equals(endWord)) {
                break;
            }
            for (int i = 0; i < word.length(); i++) {
                int level = hm.get(word);
                for (char ch = 'a'; ch <= 'z'; ch++) {
                    char[] replaceChars = word.toCharArray();
                    replaceChars[i] = ch;
                    String replaceString = new String(replaceChars);
                    if (hs.contains(replaceString)) {
                        q.add(replaceString);
                        hm.put(replaceString, level + 1);
                        hs.remove(replaceString);
                    }
                }
            }
        }
        if (hm.containsKey(endWord)) {
            List<String> seq = new ArrayList<>();
            seq.add(endWord);
            dfs(endWord, seq, res, beginWord, hm);
        }
        return res;
    }
    public void dfs(String word, List<String> seq, List<List<String>> res, String beginWord, Map<String, Intege
        if (word.equals(beginWord)) {
            List<String> ref = new ArrayList<>(seq);
            Collections.reverse(ref);
            res.add(ref);
            return;
        }
```

```
        }
        int level = hm.get(word);
        for (int i = 0; i < word.length(); i++) {
            for (char ch = 'a'; ch <= 'z'; ch++) {
                char[] replaceChars = word.toCharArray();
                replaceChars[i] = ch;
                String replaceStr = new String(replaceChars);
                if (hm.containsKey(replaceStr) && hm.get(replaceStr) == level - 1) {
                    seq.add(replaceStr);
                    dfs(replaceStr, seq, res, beginWord, hm);
                    seq.remove(seq.size() - 1);
                }
            }
        }
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the begin word: ");
        String beginWord = scanner.nextLine();
        System.out.print("Enter the end word: ");
        String endWord = scanner.nextLine();
        System.out.print("Enter the number of words in the word list: ");
        int n = scanner.nextInt();
        scanner.nextLine();
        List<String> wordList = new ArrayList<>();
        System.out.println("Enter words in the word list:");
        for (int i = 0; i < n; i++) {
            wordList.add(scanner.nextLine());
        }
        WordLadder2 solution = new WordLadder2();
        List<List<String>> result = solution.findLadders(beginWord, endWord, wordList);
        System.out.println(result);
        scanner.close();
    }
}
```

```
<terminated> WordLadder2 [Java Application] C:\Program Files\Java\bin\javaw.ex
Enter the begin word: hit
Enter the end word: cog
Enter the number of words in the word list: 5
Enter words in the word list:
hot
dot
dog
lot
cog
[[hit, hot, dot, dog, cog]]
```

10. Course Schedule

Time complexity: O(n * m)

Space complexity: O(n * m)

```java
package program19thNov;
import java.util.*;
public class CourseSchedule {
    public boolean canFinish(int numCourses, int[][] prerequisites) {
        Map<Integer, List<Integer>> preMap = new HashMap<>();
        for (int[] pre : prerequisites) {
            preMap.putIfAbsent(pre[0], new ArrayList<>());
            preMap.get(pre[0]).add(pre[1]);
        }
        Set<Integer> visitSet = new HashSet<>();
        for (int crs = 0; crs < numCourses; crs++) {
            if (!dfs(crs, preMap, visitSet)) return false;
        }
        return true;
    }
    public boolean dfs(int crs, Map<Integer, List<Integer>> preMap, Set<Integer> visitSet) {
        if (visitSet.contains(crs)) return false;
        if (!preMap.containsKey(crs) || preMap.get(crs).isEmpty()) return true;
        visitSet.add(crs);
        for (int pre : preMap.get(crs)) {
            if (!dfs(pre, preMap, visitSet)) return false;
        }
        visitSet.remove(crs);
        preMap.put(crs, new ArrayList<>());
        return true;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of courses: ");
        int numCourses = scanner.nextInt();
        System.out.print("Enter the number of prerequisite pairs: ");
        int n = scanner.nextInt();
        int[][] prerequisites = new int[n][2];
        System.out.println("Enter the prerequisite pairs:");
        for (int i = 0; i < n; i++) {
            prerequisites[i][0] = scanner.nextInt();
            prerequisites[i][1] = scanner.nextInt();
        }
        CourseSchedule solution = new CourseSchedule();
        boolean result = solution.canFinish(numCourses, prerequisites);
        System.out.println(result);
        scanner.close();
    }
}
```

```
<terminated> CourseSchedule [Java Application] C:\Program Files\Java\bin\
Enter the number of courses: 5
Enter the number of prerequisite pairs: 4
Enter the prerequisite pairs:
0 1
1 2
2 3
0 3
true
```