PW SKILLS

Lesson Plan

DP - 3

PW SKILLS

Today's checklist

- CSES Removing Digits
- Minimizing Coins
- Dice Combinations
- Number of dice rolls with Target Sum
- MCOINS-Coin Game
- C-Vacation

CSES Removing Digits

Q. You are given an integer n. On each step, you may subtract one of the digits from the number. How many steps are required to make the number equal to 0?

Input
The only input line has an integer n.
Output
Print one integer: the minimum number of steps.

Example

Input: 27
Output: 5

Solution:

Code:

```
import java.util.Scanner;

public class MinSteps {

    static int minSteps(int n) {
        int[] dp = new int[n + 1];
        for (int i = 1; i ⩽ n; ++i) {
            int temp = i;
            dp[i] = Integer.MAX_VALUE;
            while (temp > 0) {
                int digit = temp % 10;
                temp ⊨ 10;
                dp[i] = Math.min(dp[i], dp[i - digit] + 1);
            }
        }
        return dp[n];
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a number: ");
        int n = scanner.nextInt();

        int result = minSteps(n);

        System.out.println("Minimum number of steps to reach 0: " +
result);
    }
}
```

Approach:

In this solution, the minSteps function uses dynamic programming to calculate the minimum number of steps to reach 0 from a given number n.
The dp vector is used to store the minimum number of steps for each number from 0 to n. The solution iterates through each number, calculates the number of steps needed to reach it by subtracting a digit, and updates the dp array accordingly.

The result is the minimum number of steps needed to reach 0 from the given input number.

Minimizing Coins

Q. Consider a money system consisting of n coins. Each coin has a positive integer value. Your task is to produce a sum of money xxx using the available coins in such a way that the number of coins is minimal.
For example, if the coins are {1,5,7} and the desired sum is 11, an optimal solution is 5+5+1 which requires 3 coins.

Input
The first input line has two integers n and x : the number of coins and the desired sum of money.
The second line has n distinct integers c1,c2,...,cn: the value of each coin.

Output
Print one integer: the minimum number of coins. If it is not possible to produce the desired sum, print -1.

Example

Input:
3 11
1 5 7

Output:
3

Solution:

Code:

```
import java.util.Arrays;

public class MinimizingCoins {

    static int minimizingCoins(int[] coins, int target) {
        int n = coins.length;

        // Create an array to store the minimum number of coins for
each target value
        int[] dp = new int[target + 1];
        Arrays.fill(dp, Integer.MAX_VALUE);
        dp[0] = 0;

        // Fill up the dp array in a bottom-up manner
        for (int i = 1; i ⩽ target; ++i) {
            for (int j = 0; j < n; ++j) {
                if (i - coins[j] ⩾ 0 && dp[i - coins[j]] ≠
Integer.MAX_VALUE) {
                    dp[i] = Math.min(dp[i], dp[i - coins[j]] + 1);
                }
            }
        }

        // Check if it's possible to make the target sum
        if (dp[target] == Integer.MAX_VALUE) {
            return -1; // It's not possible to make the target sum
        }
        return dp[target];
    }

    public static void main(String[] args) {
        int[] coins = {2, 5, 7};
        int target = 14;

        int result = minimizingCoins(coins, target);

        if (result == -1) {
            System.out.println("It's not possible to make the
target sum.");
        } else {
            System.out.println("Minimum number of coins needed: " +
result);
        }
    }
}
```

Dice Combinations

Q. Your task is to count the number of ways to construct sum n by throwing a dice one or more times. Each throw produces an outcome between 1 and 6.

For example, if n=3 there are 4 ways:

- 1+1+1
- 1+2
- 2+1
- 3

Input
The only input line has an integer n

Input:
3

Output:
4

Solution:

```
import java.util.Scanner;

public class DiceCombinations {
    static final int MOD = (int) 1e9 + 7;

    public static int diceCombinations(int target) {
        // Create an array to store the number of ways to get each
sum
        int[] dp = new int[target + 1];
        dp[0] = 1; // There is one way to get a sum of 0 (no dice
roll)

        // Fill up the dp array in a bottom-up manner
        for (int i = 1; i ⩽ target; ++i) {
            for (int j = 1; j ⩽ 6 && i - j ⩾ 0; ++j) {
                dp[i] = (dp[i] + dp[i - j]) % MOD;
            }
        }
        return dp[target];
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the target sum: ");
        int target = scanner.nextInt();
        int result = diceCombinations(target);
        System.out.println("Number of ways to get the target sum: "
+ result);
    }
}
```

Number of dice rolls with Target Sum | Leetcode 1155

Q. You have n dice, and each dice has k faces numbered from 1 to k.

Given three integers n, k, and target, return the number of possible ways (out of the kn total ways) to roll the dice, so the sum of the face-up numbers equals target. Since the answer may be too large, return it modulo 109 + 7.

Example 1:

Input: n = 1, k = 6, target = 3
Output: 1

Explanation: You throw one die with 6 faces.
There is only one way to get a sum of 3.

Example 2:

Input: n = 2, k = 6, target = 7
Output: 6

Explanation: You throw two dice, each with 6 faces.
There are 6 ways to get a sum of 7: 1+6, 2+5, 3+4, 4+3, 5+2, 6+1.

Example 3:

Input: n = 30, k = 30, target = 500
Output: 222616187

Explanation: The answer must be returned modulo 109 + 7.

Constraints:

1 <= n, k <= 30
1 <= target <= 1000

Solution:

If 'n' becomes 0 and the target becomes 0 (target == 0 && n == 0), there is exactly one way to achieve this - return 1.
If 'n' becomes 0 or the target reaches 0 (n == 0 || target <= 0), no further dice rolls are possible, so return 0.
Calculation:

If the result for the current state is already calculated (dp[n][target] != -1), return the stored result.
Otherwise, for each possible face value of the die (from 1 to 'k'), calculate the number of ways to achieve the target sum by recursively calling the function for 'n-1' dice rolls and subtracting the current face value from the target sum.
Accumulate these ways in the ways variable.
Memoize Result:

Store the computed result in the dp array to avoid recalculating the same state.
Return: Return the final count of ways modulo 10^9 + 7 to handle large values and provide the result.

Code:

```
import java.util.Arrays;

class Solution {
    final int mod = (int) Math.pow(10, 9) + 7;

    public int numRollsToTarget(int n, int k, int target) {
        int[][] dp = new int[n + 1][target + 1];
        for (int[] d : dp) {
            Arrays.fill(d, -1);
        }
        return recursion(dp, n, k, target);
    }

    private int recursion(int[][] dp, int n, int k, int target) {
        if (target == 0 && n == 0) return 1;
        if (n == 0 || target <= 0) return 0;

        if (dp[n][target] != -1) return (int) (dp[n][target] %
mod);
        int ways = 0;
        for (int i = 1; i <= k; i++) {
            ways = (ways + recursion(dp, n - 1, k, target - i)) %
mod;
        }
        dp[n][target] = ways % mod;
        return dp[n][target];
    }
}
```

MCOINS-Coin Game

Q. Asen and Boyan are playing the following game. They choose two different positive integers K and L, and start the game with a tower of N coins. Asen always plays first, Boyan – second, after that – Asen again, then Boyan, and so on. The boy in turn can take 1, K or L coins from the tower. The winner is the boy, who takes the last coin (or coins). After a long, long playing, Asen realizes that there are cases in which he could win, no matter how Boyan plays. And in all other cases Boyan being careful can win, no matter how Asen plays.
So, before the start of the game Asen is eager to know what game case they have. Write a program coins which help Asen to predict the game result for given K, L and N.

INPUT
The input describes m games.

The first line of the standard input contains the integers K, L and m, 1 < K < L < 10, 3 < m < 50. The second line contains m integers N1, N2, ..., Nm, 1 ≤ Ni ≤ 1 000 000, i = 1, 2, ..., m, representing the number of coins in each of the m towers

SAMPLE INPUT
2 3 5
3 12 113 25714 88888

OUTPUT

The standard output contains a string of length m composed of letters A and B. If Asen wins the ith game (no matter how the opponent plays), the ith letter of the string has to be A. When Boyan wins the ith game (no matter how Asen plays), the ith letter of the string has to be B.

SAMPLE OUTPUT

ABAAB

Solution:

Code:

```
import java.util.Scanner;

public class GameResults {
    static final int MAX_N = (int) 1e6 + 5;
    static final int MAX_K = 105;

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int k = scanner.nextInt();
        int l = scanner.nextInt();
        int m = scanner.nextInt();

        // Calculate the maximum value to consider
        int maxValue = Math.max(Math.max(k, l), m);

        // Create an array to store the game results
        char[] dp = new char[MAX_N];

        // Base cases
        dp[1] = dp[k] = dp[l] = 'A';

        // Fill up the dp array in a bottom-up manner
        for (int i = 2; i <= maxValue; ++i) {
            if (dp[i] == 0) dp[i] = 'A';
            if (i >= k && dp[i - k] == 'L') dp[i] = 'A';
            if (i >= l && dp[i - l] == 'L') dp[i] = 'A';
        }

        // Process the game results
        for (int i = 0; i < m; ++i) {
            int n = scanner.nextInt();
            System.out.print(dp[n]);
        }

        System.out.println();
    }
}
```

C-Vacation

Q. Taro's summer vacation starts tomorrow, and he has decided to make plans for it now.

The vacation consists of N days. For each i (1≤i≤N), Taro will choose one of the following activities and do it on the i-th day:

A: Swim in the sea. Gain ai points of happiness.
B: Catch bugs in the mountains. Gain bi points of happiness.
C: Do homework at home. Gain ci points of happiness.

As Taro gets bored easily, he cannot do the same activities for two or more consecutive days.

Find the maximum possible total points of happiness that Taro gains.

Output
Print the maximum possible total points of happiness that Taro gains.

Sample Input 1
3
10 40 70
20 50 80
30 60 90

Sample Output 1
210

If Taro does activities in the order C, B, C, he will gain 70+50+90=210 points of happiness.

Solution:

```
import java.util.Scanner;

public class MaximumScore {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = scanner.nextInt();

        int[][] dp = new int[n + 1][3];

        for (int i = 1; i ⩽ n; ++i) {
            int a = scanner.nextInt();
            int b = scanner.nextInt();
            int c = scanner.nextInt();

            dp[i][0] = a + Math.max(dp[i - 1][1], dp[i - 1][2]);
            dp[i][1] = b + Math.max(dp[i - 1][0], dp[i - 1][2]);
            dp[i][2] = c + Math.max(dp[i - 1][0], dp[i - 1][1]);
        }

        int result = Math.max(Math.max(dp[n][0], dp[n][1]), dp[n]
[2]);
        System.out.println(result);
    }
}
```

PW SKILLS

THANK YOU !