E) Write a program to solve a 0

-1 Knapsack problem using dynamic programming or branch and

bound strategy.

Aarti Ambekar

Rollno:20121056

public class KnapsakProblem {

    public static int knapsack(int[] weights, int[] values, int capacity) {

        int n = weights.length;

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

        for (int i = 0; i <= n; i++) {

            for (int w = 0; w <= capacity; w++) {

                if (i == 0 || w == 0) {

                    dp[i][w] = 0;

                } else if (weights[i - 1] <= w) {

                    dp[i][w] = Math.max(values[i - 1] + dp[i - 1][w - weights[i - 1]], dp[i - 1][w]);

                } else {

                    dp[i][w] = dp[i - 1][w];

                }

            }

        }

        return dp[n][capacity];

    }

    public static void main(String[] args) {

        int[] weights = {2, 3, 4, 5};

        int[] values = {3, 4, 5, 6};

        int capacity = 5;

        int maxProfit = knapsack(weights, values, capacity);

        System.out.println("Maximum profit: " + maxProfit);

    }

}

Output

PS D:\7 th semister\DAA> cd "d:\7 th semister\DAA\" ; if ($?) { javac KnapsakProblem.java } ; if ($?) { java KnapsakProblem }

Maximum profit: 7

PS D:\7 th semister\DAA>