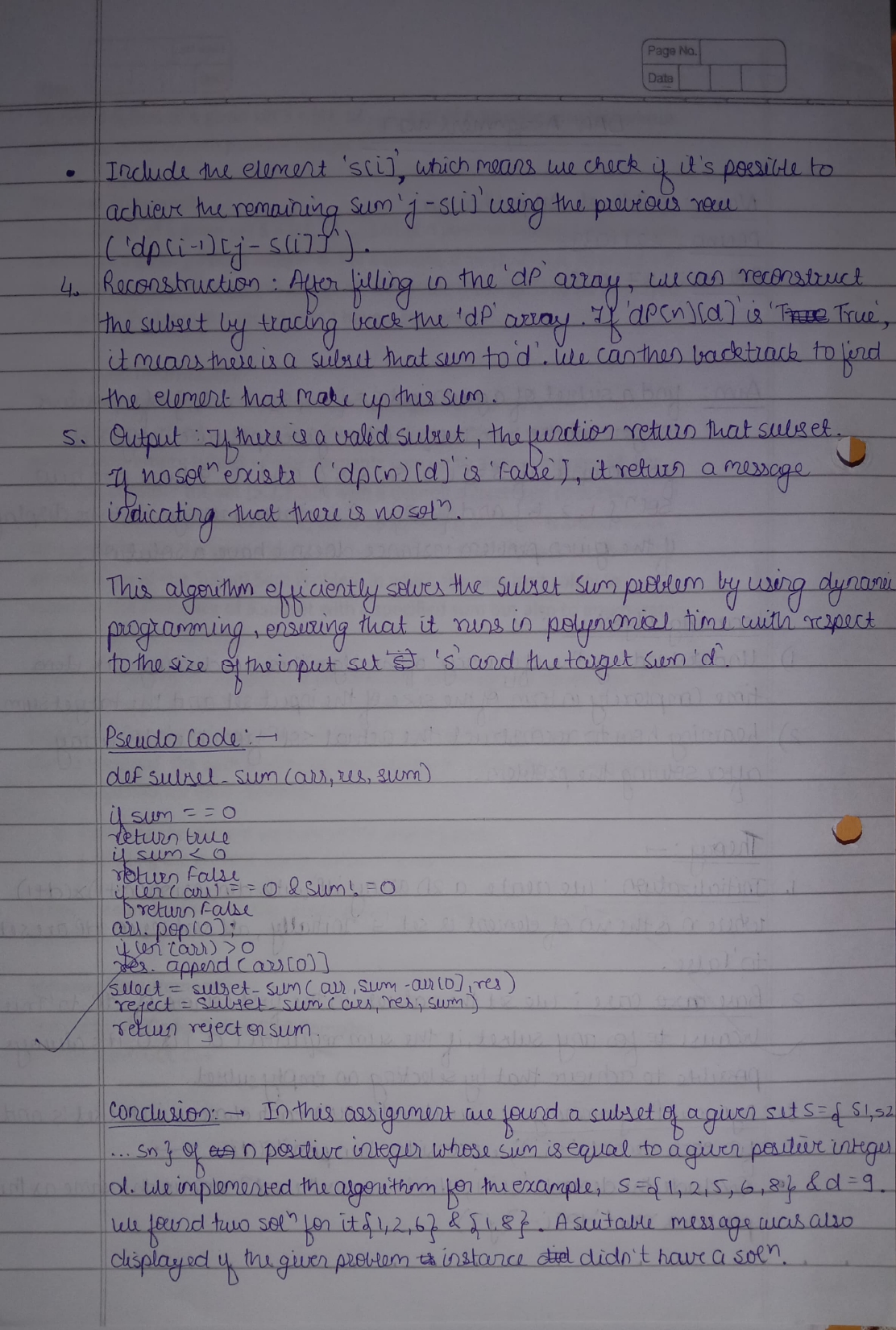
|  |  |  |  |
| --- | --- | --- | --- |
|  | Bansilal Ramnath Agarwal Charitable Trust's  Vishwakarma Institute of Information Technology  **Department of**  **Artificial Intelligence and Data Science** | | |
| Name: Siddhesh Dilip Khairnar | | | |
| Class: TY | Division: B | | Roll No: 372028 |
| Semester: V | | Academic Year: 2023-24 | |
| Subject Name & Code: Design and Analysis of Algorithm: ADUA31202 | | | |
| Title of Assignment: Find a subset of a given set S = {s1, s2 ..., s n} of n positive integers whose sum is equal to a given positive integer d. | | | |

**ASSIGNMENT NO. 7**

A piece of paper with writing on it

Description automatically generated



**Aim:**

To find a subset of a given set S = {s1, s2 ......, s n} of n positive integers whose sum is equal to a given positive integer d.

**Problem Statement:**

Find a subset of a given set S = {s1, s2 ..., s n} of n positive integers whose sum is equal to a given positive integer d. For example, if S= {1, 2, 5, 6, 8} and d = 9 there are two solutions {1,2,6} and {1,8}. A suitable message is to be displayed if the given Problem instance doesn't have a solution.

**Background Information:**

The problem at hand is a fundamental challenge in complexity theory. Given a set of integers *a1​, a2​,…, an​* up to *n* integers, the question is whether there exists a non-empty subset whose elements sum up to a given integer *M*. For instance, consider the set [5,2,1,3,9] with a desired subset sum of 9; the answer is YES as the subset [5,3,1] sums up to 9. This problem is known to be NP-complete and is a specialized case derived from the knapsack problem. It's an extension of the Subset Sum Problem in which the task isn't solely to ascertain the existence of a subset with a specified sum but also to enumerate and print all such subsets.

To tackle this, a 2D array *dp[i][j]* is constructed, where *dp[i][j]* stores true if the sum *j* is achievable using array elements from 0 to *i*. Once this array is populated, a recursive traversal is conducted starting from *dp[n−1] [sum]*. During traversal, the path leading to the current cell is recorded, considering two possibilities for each element:

1. Including the current element in the ongoing path.
2. Excluding the current element from the ongoing path.

When the sum becomes 0, the recursive calls are halted, and the current path is printed.

**Software Requirements:**

Text Editor: VSCode, Online GDB Compiler

Environment: GCC C++

**Program Code:**

#include <iostream>

#include <vector>

using namespace std;

bool \*\*dp;

void display(const vector<int> &v)

{

    for (int i = 0; i < v.size(); ++i)

        cout << v[i] << " ";

    cout << endl;

}

void printSubsetsRec(int arr[], int i, int sum, vector<int> &p)

{

    if (i == 0 && sum != 0 && dp[0][sum])

    {

        p.push\_back(arr[i]);

        display(p);

        return;

    }

    if (i == 0 && sum == 0)

    {

        display(p);

        return;

    }

    if (dp[i - 1][sum])

    {

        vector<int> b = p;

        printSubsetsRec(arr, i - 1, sum, b);

    }

    if (sum >= arr[i] && dp[i - 1][sum - arr[i]])

    {

        p.push\_back(arr[i]);

        printSubsetsRec(arr, i - 1, sum - arr[i], p);

    }

}

void printAllSubsets(int arr[], int n, int sum)

{

    if (n == 0 || sum < 0)

        return;

    dp = new bool \*[n];

    for (int i = 0; i < n; ++i)

    {

        dp[i] = new bool[sum + 1];

        dp[i][0] = true;

    }

    if (arr[0] <= sum)

        dp[0][arr[0]] = true;

    for (int i = 1; i < n; ++i)

        for (int j = 0; j < sum + 1; ++j)

            dp[i][j] = (arr[i] <= j) ? dp[i - 1][j] || dp[i - 1][j - arr[i]] : dp[i - 1][j];

    if (dp[n - 1][sum] == false)

    {

        cout << "There are no subsets with sum " << sum << endl;

        return;

    }

    vector<int> p;

    printSubsetsRec(arr, n - 1, sum, p);

}

int main()

{

    int arr[] = {1, 2, 5, 6, 8};

    int n = sizeof(arr) / sizeof(arr[0]);

    int sum = 9;

    printAllSubsets(arr, n, sum);

    // Free allocated memory

    for (int i = 0; i < n; ++i)

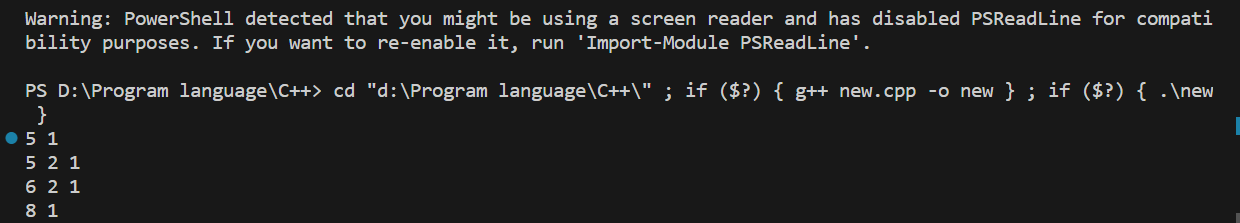
        delete[] dp[i];

    delete[] dp;

    return 0;

}

**Output:**



**Conclusion:**

In this assignment we found a subset of a given set S = {s1, s2 ...s n} of n positive integers whose sum is equal to a given positive integer d. We implemented the algorithm for the example, S= {1, 2, 5, 6, 8} and d = 9. We found two solutions for it, {1,2,6} and {1,8}. A suitable message was also displayed if the given Problem instance didn’t have a solution.