**CHANDIGARH UNIVERSITY**

**UNIVERSITY INSTITUTE OF ENGINEERING**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**



|  |  |
| --- | --- |
| **Submitted By: Kanishk Soni Submitted To: Er. Tanu Dhiman** | |
| **Subject Name** | **Design and Analysis of Algorithm Lab** |
| **Subject Code** | **20CSP-312** |
| **Branch** | **BE-CSE** |
| **Semester** | **5th** |

**Worksheet - 6**

**Student Name:** Kanishk Soni **UID:** 20BCS9398

**Branch:** BE CSE **Section/Group:** 20BCS\_WM\_707-B

**Semester:** 5 **Date of Performance:** 02/10/2022

**Subject Name:** Design & Analysis of Algorithm **Subject Code:** 20CSP-312

1. **Aim/Overview of the practical:** To implement subset-sum problem using Dynamic Programming.
2. **Task to be done:** Find whether or not there exists any subset of the given set.
3. **Algorithms:**
4. We create a boolean subset[][] and fill it in bottom up manner.
5. The value of subset[i][j] will be true if there is a subset of set[0..j-1] with sum equal to i., otherwise false.
6. subset[i][j] = true if there is a subset with:
7. the i-th element as the last element \* sum equal to j
8. subset[i][0] = true as sum of {} = 0
9. subset[0][j] = false as with no elements we can get no sum
10. subset[i][j] = subset[i-1][j-E1]; where E1 = array[i-1]
11. Finally, we return subset[n][sum]
12. **Code:**

#include <iostream>

using namespace std;

bool subsetsum\_DP(int a[], int n, int sum) {

bool dp[n + 1][sum + 1];

int i, j;

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

dp[i][0] = true;

}

for (j = 1; j <= sum; j++) {

dp[0][j] = false;

}

for (i = 1; i <= n; i++) {

for (j = 1; j <= sum; j++) {

if (dp[i - 1][j] == true) {

dp[i][j] = true;

} else {

if (a[i - 1] > j) {

dp[i][j] = false;

} else {

dp[i][j] = dp[i - 1][j - a[i - 1]];

}

}

}

}

return dp[n][sum];

}

int main() {

int set[] = {3, 34, 4, 12, 5, 2};

int sum = 9;

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

if (subsetsum\_DP(set, n, sum) == true) {

cout<<"Found a subset with given sum";

} else {

cout<<"No subset with given sum";

}

cout<<endl;

return 0;

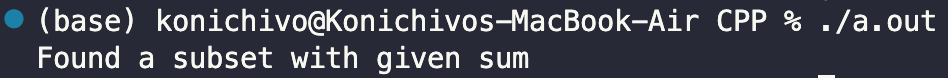
}

1. **Complexity:**

• Worst case time complexity: Θ(n\*sum)

• Space complexity: Θ(sum)

1. **Result/Output:**



**Learning outcomes (What I have learnt):**

1. Create a program keeping in mind the time complexity
2. Create a program keeping in mind the space complexity
3. Steps to make optimal algorithm
4. Learnt about how to implement subset sum problem using dynamic programming