**Experiment 17**

**Write a program to implement the Backtracking algorithm to solve the Subset-sum problem.**

**Program:-**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

void printSubset(int A[], int size) {

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

printf("%d ", A[i]);

}

printf("\n");

}

void subsetSum(int S[], int T[], int S\_size, int T\_size, int sum, int ite, int const target\_sum) {

if(target\_sum == sum) {

printSubset(T, T\_size);

if(ite + 1 < S\_size && sum - S[ite] + S[ite+1] <= target\_sum) {

subsetSum(S, T, S\_size, T\_size-1, sum - S[ite], ite + 1, target\_sum);

}

return;

}

else {

if(ite < S\_size && sum + S[ite] <= target\_sum) {

for(int i = ite; i < S\_size; i++) {

T[T\_size] = S[i];

if(sum + S[i] <= target\_sum) {

subsetSum(S, T, S\_size, T\_size + 1, sum + S[i], i + 1, target\_sum);

}

}

}

}

}

void generateSubsets(int S[], int size, int target\_sum) {

int \*tuplet\_vector = (int \*)malloc(size \* sizeof(int));

subsetSum(S, tuplet\_vector, size, 0, 0, 0, target\_sum);

free(tuplet\_vector);

}

int main() {

int weights[1000];

int i, n, sum;

double time;

clock\_t start, end;

printf("Enter number of items:");

scanf("%d", &n);

printf("Enter target sum:");

scanf("%d", &sum);

start = clock();

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

weights[i] = rand() % 1000;

printf("Item :%d \n", weights[i]);

}

printf("Subsets with sum %d are:\n", sum);

generateSubsets(weights, n, sum);

end = clock();

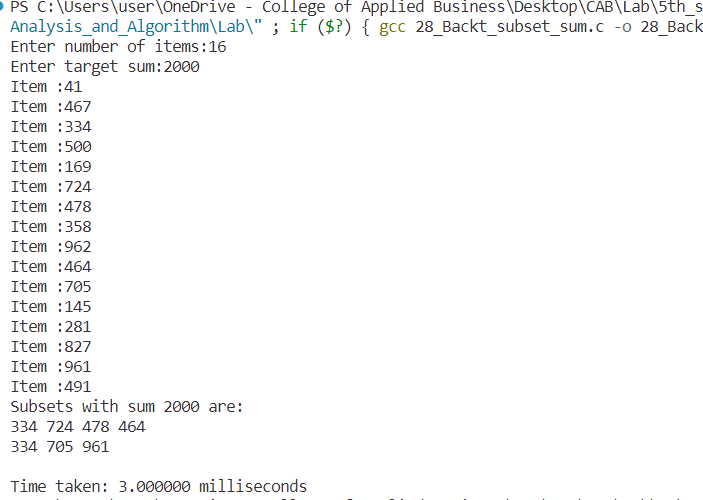
time = ((double)(end - start) \* 1000) / CLOCKS\_PER\_SEC;

printf("\nTime taken: %lf milliseconds\n", time);

return 0;

}

**Output:**



**Conclusion:**

This experiment had been conducted in a 64-bit system with 16 GB RAM and Processor 12th Gen Intel(R) Core (TM) i5-12500H 3.10 GHz. The algorithm was implemented in C programming language in Visual Studio Code 1.85.1 Code Editor. The time taken by this algorithm for 16 number of input size is 3 milliseconds.. The running time is analyzed as O(2n).