DESIGN AND ANALYSIS OF ALGORITHM

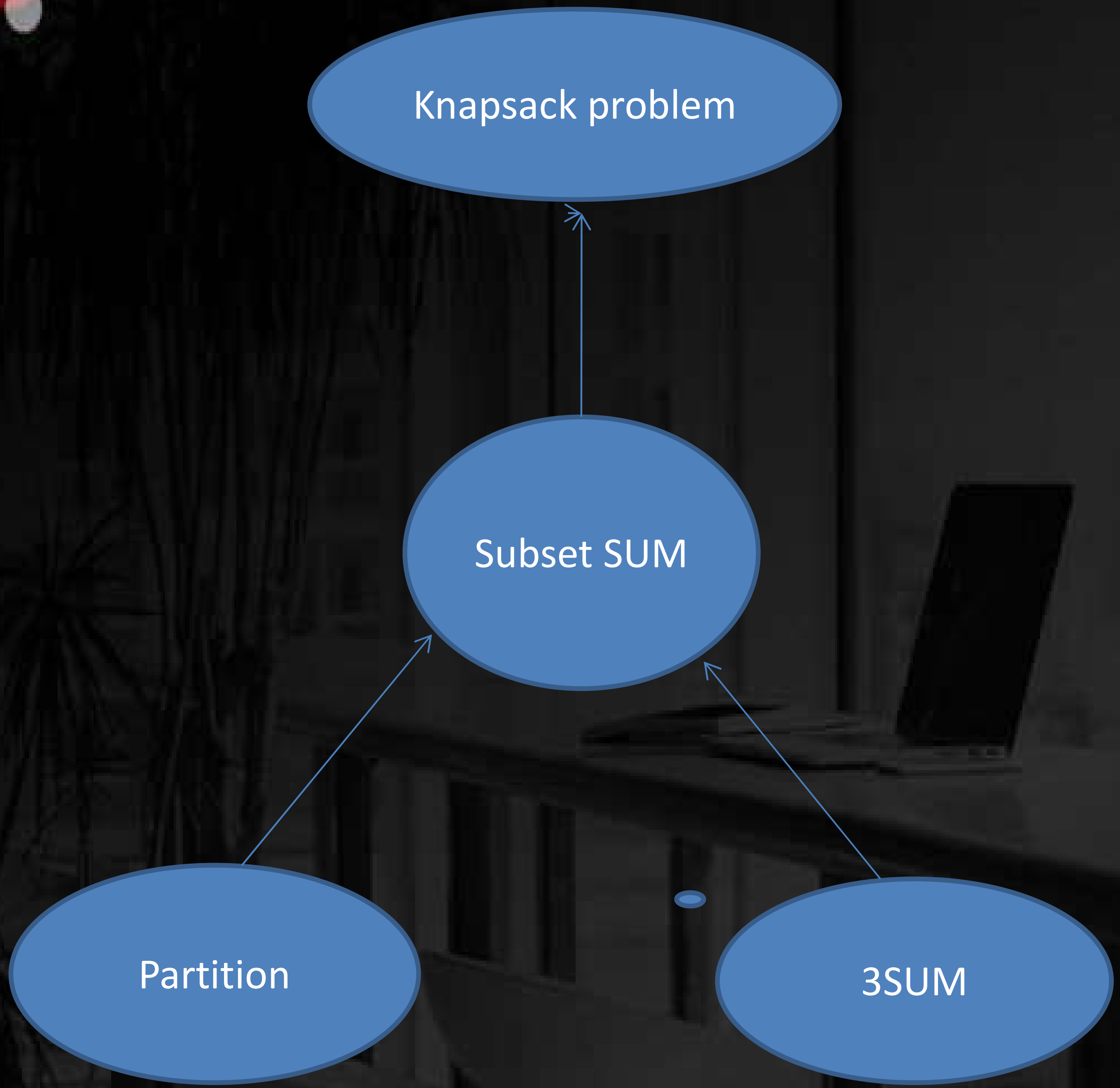
Subset Sum problem

K.Venkatesh –AP21110010933
Chandan Shah-AP21110010934
P. Dheeraj – AP21110010935
Damini Tiwari – AP21110010936

ABSTRACT:

Is the Subset-Sum Problem a sub-problem of this problem?

Yes. From the definition of 0-1 Knapsack Problem, we can know that if we let $p_i = w_i$ this problem is a Subset-Sum problem. Then for any instance of Subset-Sum problem,
it is also an instance of 0-1 Knapsack Problem. Once we get a subset that maximizes the Subset-Sum problem, it must also be maximized as a 0-1 Knapsack Problem.
So the Subset-Sum problem is a sub-problem of 0-1 Knapsack Problem.



Knapsack problem
Subset SUM
Partition
3SUM

Exhaustive Algorithm

The exhaustive search algorithm checks all possible multisets $T \subseteq S$ until finding a T (if any) with the appropriate sum. It should return True (it found such a multiset T) or False (it didn't). It will be extremely slow. For maximum credit, come up with pruning techniques to shorten the search.

There are three types:

1. Bit wise
2. Recursion
3. Evaluation

Here we am using the recursion method because we keen of it :

Approach: For the recursive approach we will consider two cases.

1. Inclusion
2. Exclusion

1.Consider the element and now the next **tempsum = tempsum of present + value of the element** and **index = index+1**

2.Leave the element and now the next **temp sum = tempsum of present** and **index = index+1**

ALGORITHM :

Using vector header file and bool :

helper(a,sum,tempsum,i)-----///here a = array ; sum = target value ;tempsum== depends on inclusion and exclusion ;i= index value in array

helper(a,sum,tempSum+a[i],i+1) || helper(a,sum,tempSum,i+1); [inclusion or exclusion]
// base case:(using bool)

if (sum==tempSum) then return true
if (sum=a.size()) then return false

ppt background images - Go | (2) WhatsApp | Untitled document | Dynamic Programming - Sub | Online C++ Compiler - onlin

onlinegdb.com/online_c++_compiler

OnlineGDB beta

online compiler and debugger for c/c++

code. compile. run. debug. share.

- IDE
- My Projects
- Classroom new
- Learn Programming
- Programming Questions
- Sign Up
- Login

82K

About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy

Run | Debug | Stop | Share | Save | {} Beautify

Language C++

main.cpp

```cpp

// Recursion
// Given an array of non-negative integers, and a value sum, determine if there is a subset of the
//given set with sum equal to given sum.
#include<bits/stdc++.h>
using namespace std;
class Solution{
public:
bool helper (vector <int> a, int sum,int tempSum, int i){
//Base Cases
if (sum==tempSum){
return true;
}
if (sum<tempSum){
return false;
}
if (i>=a.size()){
return false;
}
//Recursive Calls
int recCall1 = helper(a,sum,tempSum+a[i],i+1);
int recCall2 = helper(a,sum,tempSum,i+1);
//Small Calculation
```

input

```
enter the no ofelements in the array:
```

Waiting for www.google.com...

Type here to search

ENG 21:45 25-11-2022

ppt background images - Go | (2) WhatsApp | Untitled document | Dynamic Programming - Sub | Online C++ Compiler - onlin

onlinegdb.com/online_c++_compiler

OnlineGDB beta

online compiler and debugger for c/c++

code. compile. run. debug. share.

IDE

My Projects

Classroom new

Learn Programming

Programming Questions

Sign Up

Login

Run | Debug | Stop | Share | Save | { } Beautify

Language C++

main.cpp

```cpp
//Small Calculation
return recCall1 || recCall2;
}
bool isSubsetSum(vector<int>arr, int sum){
return helper(arr,sum,0,0);
}
};
int main()
{
int N, sum;
cout<<"enter the no ofelements in the array:"<<endl;
cin >> N;
vector<int> arr(N);
cout<<"enter the elements in the array"<<endl;
for(int i = 0; i < N; i++){
cin >> arr[i];
}
cout<<"enter the target sum:"<<endl;
cin >> sum;
Solution ob;
if(ob.isSubsetSum(arr, sum) == true){
cout<<"subset found"<<endl;
}
```

input

```
enter the no ofelements in the array:
```


About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy

Waiting for www.google.com...

Type here to search

ENG 21:45 25-11-2022

main.cpp

```cpp
int main()
{
int N, sum;
cout<<"enter the no ofelements in the array:"<<endl;
cin >> N;
vector<int> arr(N);
cout<<"enter the elements in the array"<<endl;
for(int i = 0; i < N; i++){
cin >> arr[i];
}
cout<<"enter the target sum:"<<endl;
cin >> sum;
Solution ob;
if(ob.isSubsetSum(arr, sum) == true){
cout<<"subset found"<<endl;
}
else{
cout<<"subset not found"<<endl;
}
return 0;
}
```

input

```
enter the elements in the array
1 2 3
enter the target sum:
5
subset found


...Program finished with exit code 0
Press ENTER to exit console.
```

Dynamic Programming

So we will create a 2D array of size (arr.size() + 1) * (target + 1) of type boolean. The state DP[i][j] will be true if there exists a subset of elements from A[0....i] with sum value = ‘j’. The approach for the problem is:

```
if (A[i-1] > j)
    DP[i][j] = DP[i-1][j]
else
      DP[i][j] = DP[i-1][j] OR DP[i-1][j-A[i-1]]
```

1. This means that if current element has value greater than ‘current sum value’ we will copy the answer for previous cases
2. And if the current sum value is greater than the ‘ith’ element we will see if any of previous states have already experienced the sum=’j’ OR any previous states experienced a value ‘j – A[i]’ which will solve our purpose.

The below simulation will clarify the above approach:

set[]={3, 4, 5, 2}
target=6

	0	1	2	3	4	5	6
0	T	F	F	F	F	F	F
3	T	F	F	T	F	F	F
4	T	F	F	T	T	F	F
5	T	F	F	T	T	T	F
2	T	F	T	T	T	T	T

main.cpp

```cpp
// A Dynamic Programming solution
// for subset sum problem
#include <iostream>
using namespace std;
// Returns true if there is a subset of set[]
// with sum equal to given sum
bool isSubsetSum(int set[], int n, int sum)
{
// The value of subset[i][j] will be true if
// there is a subset of set[0..j-1] with sum
// equal to i
bool subset[n + 1][sum + 1];
// If sum is 0, then answer is true
for (int i = 0; i <= n; i++)
subset[i][0] = true;
// If sum is not 0 and set is empty,
// then answer is false
for (int i = 1; i <= sum; i++)
subset[0][i] = false;
// Fill the subset table in bottom up manner
for (int i = 1; i <= n; i++) {
for (int j = 1; j <= sum; j++) {
if (j < set[i - 1])
```

input

```
enter the elments in the array:
3 4 5 2
enter the target sum:
6
Found a subset with given sum

...Program finished with exit code 0
console.
```

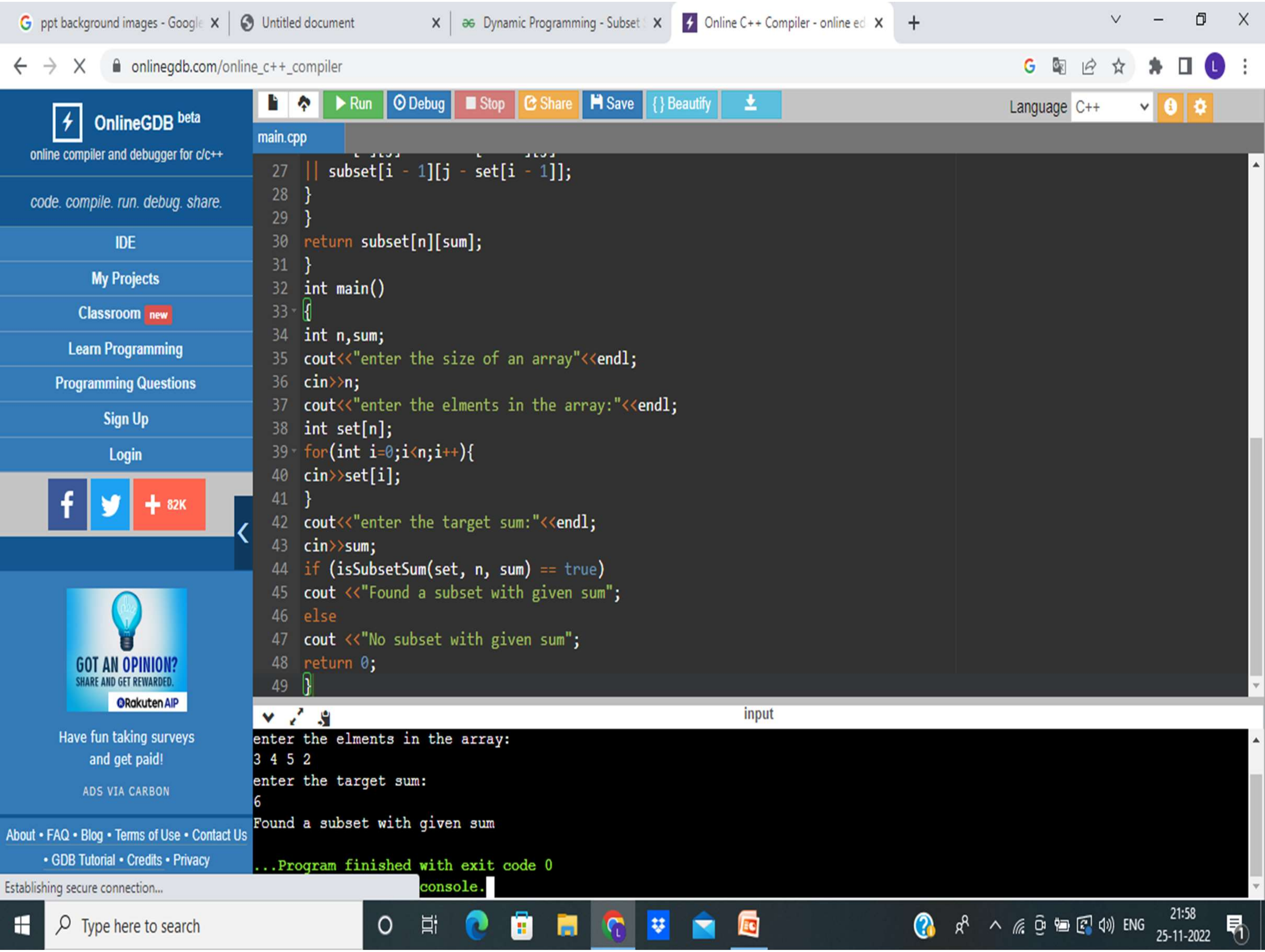
main.cpp

```
subset[i][j] = subset[i - 1][j];
if (j >= set[i - 1])
subset[i][j] = subset[i - 1][j]
|| subset[i - 1][j - set[i - 1]];
}
}
return subset[n][sum];
}
int main()
{
int n,sum;
cout<<"enter the size of an array"<<endl;
cin>>n;
cout<<"enter the elments in the array:"<<endl;
int set[n];
for(int i=0;i<n;i++){
cin>>set[i];
}
cout<<"enter the target sum:"<<endl;
cin>>sum;
if (isSubsetSum(set, n, sum) == true)
cout <<"Found a subset with given sum";
else
cout <<"No subset with given sum";
```

input

```
enter the elments in the array:
3 4 5 2
enter the target sum:
6
Found a subset with given sum

...Program finished with exit code 0
console.
```


ppt background images - Google
Untitled document
Dynamic Programming - Subset
Online C++ Compiler - online ed
onlinegdb.com/online_c++_compiler
Run
Debug
Stop
Share
Save
{} Beautify
Language C++
OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
82K
GOT AN OPINION?
SHARE AND GET REWARDED.
Rakuten AIP
Have fun taking surveys and get paid!
ADS VIA CARBON
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
Establishing secure connection...
main.cpp
|| subset[i - 1][j - set[i - 1]];
}
}
return subset[n][sum];
}
int main()
{
int n,sum;
cout<<"enter the size of an array"<<endl;
cin>>n;
cout<<"enter the elments in the array:"<<endl;
int set[n];
for(int i=0;i<n;i++){
cin>>set[i];
}
cout<<"enter the target sum:"<<endl;
cin>>sum;
if (isSubsetSum(set, n, sum) == true)
cout <<"Found a subset with given sum";
else
cout <<"No subset with given sum";
return 0;
}
input
enter the elments in the array:
3 4 5 2
enter the target sum:
6
Found a subset with given sum
...Program finished with exit code 0
console.
Type here to search
ENG
21:58
25-11-2022

FOR RECURSION COMPLEXITY IS :

The time complexity will be exponential because it will check for all the subset\

FOR DYNAMIC PROGRAMMING :

Time Complexity: O(sum*n), where sum is the ‘target sum’ and ‘n’ is the size of array.

Auxiliary Space: O(sum*n), as the size of 2-D array is sum*n. + O(n) for recursive stack space

THANK YOU