Find All Sum Combinations

Problem Statement

Given a positive integer, `target`, print all possible combinations of positive integers that sum up to the `target` number.

For example, if we are given input ‘5’, these are the possible sum combinations.

```
1, 4
2, 3
1, 1, 3
1, 2, 2
1, 1, 1, 2
1, 1, 1, 1, 1
```

The output will be in the form a list of lists or an array of arrays. Each element in the list will be another list containing a possible sum combination.

Hint

- Recursion
- Two lists

Try it yourself

C++ Java Python JavaScript Ruby

```cpp
vector<vector<int>> print_all_sum(int target){
    vector<vector<int>> output;
    //Write - Your - Code
    return output;
}
```

Test

Solution

C++ Java Python JavaScript Ruby

```cpp
void print(vector<vector<int>> output){
  for(int i = 0; i < output.size(); i++){
    cout << "[ ";
    for(int j = 0; j < output[i].size(); j++){
      cout << output[i][j] << ", ";
    }
    cout << "]" << endl;
  }
}

void print_all_sum_rec(
    int target,
    int current_sum,
```

```
14      int start, vector<vector<int>>& output,
15      vector<int>& result) {
16
17    if (target == current_sum) {
18      output.push_back(result);
19    }
20
21    for (int i = start; i < target; ++i) {
22      int temp_sum = current_sum + i;
23      if (temp_sum <= target) {
24        result.push_back(i);
25        print_all_sum_rec(target, temp_sum, i, output, result);
26        result.pop_back();
27
28      } else {
29        return;
30      }
31    }
```

Run

Solution Explanation

Runtime Complexity

Exponential.

Memory Complexity

Linear, O(n).

Solution Breakdown

Here we will recursively go through all possible sum combinations. Whenever the running sum equals the target, we will print that combination.

The algorithm will recursively check all the numbers which can sum up to the `target`. In each recursive call, there is a `for` loop which runs from `start` to `target`. `start` is initially `1`. The `current_sum` is the variable that is incremented in every recursive call.

Here is the logic of the code; every time a value is added to the `current_sum`, it is also added to the `result` list which is the sum combination for that particular call. Whenever `current_sum` becomes equal to `target`, we can be sure that the `result` list contains a possible combination for `target`. This list is appended to the final `output` list.

```
Base condition of recursion:

if current_sum equals target
  print the output contents
```

Before each recursive call, an element is added to `result`. However, after each call, this element is also removed from the list in order to reset the list.

Let's run this algorithm step-by-step for an example where we have to find all possible sum combinations of `4`.

```
current_sum: 0, start: 1, result: [ ]
current_sum: 1, start: 1, result: [ 1 ]
current_sum: 2, start: 1, result: [ 1,1 ]
current_sum: 3, start: 1, result: [ 1,1,1 ]
current_sum: 4, start: 1, result: [ 1,1,1,1 ]

Add to output: 1, 1, 1, 1

current_sum: 3, start: 1, result: [ 1,1,1 ]
current_sum: 4, start: 2, result: [ 1,1,2 ]

Add to output: 1, 1, 2


current_sum: 3, start: 2, result: [ 1,2 ]
current_sum: 4, start: 3, result: [ 1,3 ]

Add to output: 1, 3

current_sum: 2, start: 2, result: [ 2 ]
current_sum: 4, start: 2, result: [ 2,2 ]

Add to output: 2, 2

current_sum: 3, start: 3, result: [ ]
```

LEGAL

Privacy Policy

(/privacy)
Cookie Settings

Terms of Service

(/terms)
Business Terms of Service

(/enterprise-terms)

Contributor

(/m/write-on-edpresso)

ABOUT US

Our Team

(/team)
Careers **Hiring**

(//jobs.lever.co/educative)

(/b2c-trial)
CodingInterview.com

(//codinginterview.com)
Press

(/press)
Contact Us

(/contactUs)

book.com/educativeinc) (//linkedin.com/company/educative-inc/) (//twitter.com/educativeinc) (//www.youtube.com/channel/UCT_8FqzTIr2Q1BOtvX_DPPw/?sub_confirmation=1) (//educativesessions.podb