Algorithms @tutorialhorizon

ADOBE / AMAZON QUESTIONS / APACHE / EXPERT / GOOGLE INTERVIEW / MICROSOFT INTERVIEW / RECURSION / TOP COMPANIES

1

FOLLOW:

Dynamic Programming — Subset Sum Problem

BY SJ · MAY 10, 2015

Objective: Given a set of positive integers, and a value *sum S*, find out if there exist a subset in array whose sum is equal to given *sum S.*

Example:

```
int[] A = { 3, 2, 7, 1}, S = 6

Output: True, subset is (3, 2, 1}
```

We will first discuss the recursive approach and then we will improve it using Dynamic Programming.

Recursive Approach:

For every element in the array has two options, either we will include that element in subset or we don't include it.

- So if we take example as int[] A = { 3, 2, 7, 1}, S = 6
- If we consider another int

RECENT POSTS

Dynamic Programming — Longest Palindromic Subsequence

Dynamic Programming — Maximum Product Cutting Problem.

Dynamic Programming — Minimum Numbers are Required Whose Square Sum is Equal To a Given Number

Dynamic Programming — Longest Common Substring

Dynamic Programming — Longest Common Subsequence

Dynamic Programming — Rod Cutting Problem

Dynamic Programming — Coin Change Problem

array with the same size as A.

- If we include the element in subset we will put 1 in that particular index else put 0.
- So we need to make every possible subsets and check if any of the subset makes the sum as S.
- *If we think carefully this problem is quite similar to " Generate All Strings of n bits"*
- *See the code for better explanation.*

```
1  public static void find(ir
2                          ir
3                  if (currSu
4                          Sy
5                          fo
6
7
8
9                          }
10
11                 } else if
12                         re
13                 } else {
14                         sc
15                         cu
16                         fi
17
18                         sc
19                         cu
20                         fi
21                 }
22                 return;
23         }
```

SubSetSumRecursion.java view raw
hosted with ❤ by GitHub

Time Complexity: O(2^n).

Approach: Dynamic Programming (Bottom-Up)

Base Cases:

- If no elements in the set then we can't make any subset except for 0.
- If sum needed is 0 then by returning the empty subset we can make the subset with sum 0.

Given — Set = **arrA[]**, Size = **n**, sum = **S**

- Now for every element in he set we have 2 options, either we include it or exclude it.
- for any i^{th} element–
- If include it => S = S-arrA[i], n=n-1
- If exclude it => S, n=n-1.

Recursive Equation:

```
Base Cases:
SubsetSum(arrA, n, S)= false, if
```


ARCHIVES

```
sum > 0 and
n == 0
SubsetSum(a
rrA, n, S)=
true, if
sum == 0
(return
empty set)
```

Rest Cases

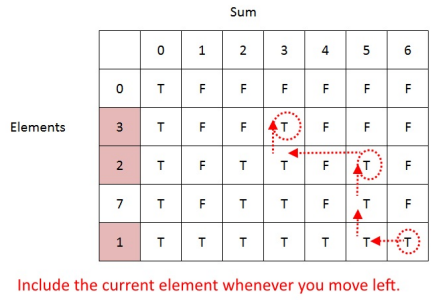
```
SubsetSum(a
rrA, n, S)
=
SubsetSum(a
rrA, n-1,
S)||
SubsetSum(a
rrA, n-1,
S-arrA[n-
1])
```

	Sum						
	0	1	2	3	4	5	6
0	T	F	F	F	F	F	F
Elements 3	T	F	F	T	F	F	F
2	T	F	T	T	F	T	F
7	T	F	T	T	F	T	F
1	T	T	T	T	T	T	T

Subset Sum Problem

How to track the elements.

- Start from the bottom-right corner and back-track and check from the True is coming.
- If value in the cell above if false that means current cell has become true after including the current element. So include the current element and check for the sum = sum — current element.



Subset Sum Problem — Track Solution

Complete Code:

```
1  public class SubSetSum {
2
3          public static bool
4                  boolean[][
5                  // if sum
6                  for(int i=
7                          sc
8                  }
9                  // if sum
10
11
12
13                 //
14                 for(int i=
15                         fc
```

```
16
17
18
19
20
21
22                          }
23                  }
24                  return sol
25          }
26
27          public static voic
28                  int[] A =
29                  System.out
30          }
31
32  }
```

SubsetSumDP.java hosted **view raw** with ❤ by **GitHub**

```
Output: From DP:
true
```

Related Posts:

- Dynamic Programming — Coin Change Problem
- Minimum number that cannot be formed by any subset of an array
- Print All Combinations of subset of size K from Given Array
- Dynamic Programming — Maximum size square sub-matrix with all 1s
- Print All N Length Strings from Given Number K

- Print All the Subsets of a Given Set (Power Set)
- Dynamic Programming — Longest Common Subsequence
- Dynamic Programming — Minimum Cost Path Problem
- Check if Array Contains All Elements Of Some Given Range
- Search an Element in a Rotated Sorted Array

Related

Print All Combin… of subset of size K from Given Array
In "Adobe"

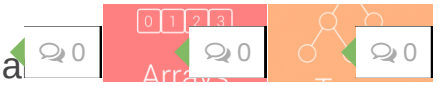
Print All the Subsets of a Given Set (Power Set)
In "Adobe"

Count All Paths from Top left to bottom right in Two Dimensi… Array including Diagonal Paths
In "Amazon Questions"

Tags: 2D Arrays

Dynamic Programming Recursion

YOU MAY ALSO LIKE...



Dynamic Programming — Minimum Numbers are Required Whose Square Sum is Equal To a Given Number

Check if Array Contains All Elements Of Some Given Range

Get the Height of a Node in a Binary Tree