GeeksforGeeks

A computer science portal for geeks

GeeksQuiz

- Home
- Algorithms
- DS
- GATE
- Interview Corner
- Q&A
- C
- C++
- Java
- Books
- Contribute
- Ask a Q
- About

Array
Bit Magic
C/C++
Articles
GFacts
Linked List
MCQ
Misc
Output
String
Tree
Graph

Dynamic Programming | Set 25 (Subset Sum Problem)

Given a set of non-negative integers, and a value *sum*, determine if there is a subset of the given set with sum equal to given *sum*.

```
Examples: set[] = {3, 34, 4, 12, 5, 2}, sum = 9
Output:  True  //There is a subset (4, 5) with sum 9.
```

Let isSubSetSum(int set[], int n, int sum) be the function to find whether there is a subset of set[] with sum equal to *sum*. n is the number of elements in set[].

The isSubsetSum problem can be divided into two subproblems
…a) Include the last element, recur for n = n-1, sum = sum – set[n-1]
…b) Exclude the last element, recur for n = n-1.

If any of the above the above subproblems return true, then return true.

Following is the recursive formula for isSubsetSum() problem.

```
isSubsetSum(set, n, sum) = isSubsetSum(set, n-1, sum) ||
                           isSubsetSum(arr, n-1, sum-set[n-1])
Base Cases:
isSubsetSum(set, n, sum) = false, if sum > 0 and n == 0
isSubsetSum(set, n, sum) = true, if sum == 0
```

Following is naive recursive implementation that simply follows the recursive structure mentioned above.

```
// A recursive solution for subset sum problem
#include <stdio.h>

// Returns true if there is a subset of set[] with sun equal to given sum
bool isSubsetSum(int set[], int n, int sum)
{
   // Base Cases
   if (sum == 0)
     return true;
   if (n == 0 && sum != 0)
     return false;

   // If last element is greater than sum, then ignore it
   if (set[n-1] > sum)
     return isSubsetSum(set, n-1, sum);

   /* else, check if sum can be obtained by any of the following
      (a) including the last element
      (b) excluding the last element   */
   return isSubsetSum(set, n-1, sum) || isSubsetSum(set, n-1, sum-set[n-1]);
}

// Driver program to test above function
int main()
{
  int set[] = {3, 34, 4, 12, 5, 2};
  int sum = 9;
  int n = sizeof(set)/sizeof(set[0]);
  if (isSubsetSum(set, n, sum) == true)
     printf("Found a subset with given sum");
  else
     printf("No subset with given sum");
  return 0;
}
```

Output:

```
 Found a subset with given sum
```

The above solution may try all subsets of given set in worst case. Therefore time complexity of the above solution is exponential. The problem is in-fact NP-Complete (There is no known polynomial time solution for this problem).

We can solve the problem in Pseudo-polynomial time using Dynamic programming. We create a boolean 2D table subset[][] and fill it in bottom up manner. 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. Finally, we return subset[sum][n]

```
// A Dynamic Programming solution for subset sum problem
#include <stdio.h>

// Returns true if there is a subset of set[] with sun 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..
    //  with sum equal to i
    bool subset[sum+1][n+1];

    // If sum is 0, then answer is true
    for (int i = 0; i <= n; i++)
      subset[0][i] = true;

    // If sum is not 0 and set is empty, then answer is false
    for (int i = 1; i <= sum; i++)
      subset[i][0] = false;

     // Fill the subset table in botton up manner
     for (int i = 1; i <= sum; i++)
     {
       for (int j = 1; j <= n; j++)
       {
         subset[i][j] = subset[i][j-1];
         if (i >= set[j-1])
           subset[i][j] = subset[i][j] || subset[i - set[j-1]][j-1];
       }
     }

    /* // uncomment this code to print table
     for (int i = 0; i <= sum; i++)
     {
       for (int j = 0; j <= n; j++)
          printf ("%4d", subset[i][j]);
       printf("\n");
     } */

     return subset[sum][n];
}

// Driver program to test above function
int main()
{
  int set[] = {3, 34, 4, 12, 5, 2};
  int sum = 9;
  int n = sizeof(set)/sizeof(set[0]);
  if (isSubsetSum(set, n, sum) == true)
     printf("Found a subset with given sum");
  else
     printf("No subset with given sum");
  return 0;
}
```

Output:

```
Found a subset with given sum
```

Time complexity of the above solution is O(sum*n).

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A’s using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Dynamic Programming

Tweet 4 +1 1

Writing code in comment? Please use **ideone.com** and share the link here.

100 Comments GeeksforGeeks 1 Login

Recommend 2 Share Sort by Newest

Join the discussion…

Guest • 2 days ago

Can't it be solved using same solution as 0-1 knapsack with both weight and benefit array as set array and total capacity of knapsack equal to desired sum?? Correct me if i am wrong...

• Reply • Share ›

Mission Peace • 11 days ago

https://www.youtube.com/watch?...

Watch this video on above question

• Reply • Share ›

Modius • a month ago

OMG, O(sum*n)? This code has so many problems I can't begin to list them all. First, I pass you {1,2} for the array and 20000000000 for the sum and you take O(4000000000) time! nice.

2nd, I pass you {1,2} and -15 for the sum and allocate an array with -15 elements and throw an exception.

With using O(anything * sum) your dependent on the sum for your speed. This means that you had better know that the sum is a very small number or this entire method goes up in flames.

I just had to do this for a job interview - in C#, and I expect they were looking for the 'cool dynamic programming' solution that everyone is teaching, but instead I did it in O(n * log2(n)) time (pretty sure I got that right)- which always worked, even for: {-2,2} sum: 0 - something that would blow chunks here. And it doesn't depend on the sum, just the number of elements in the list. Furthermore, it doesn't try to allocate all of the memory on the computer and crash in the event of a huge sum.

• Reply • Share ›

Ted ➦ Modius • 17 days ago

P=NP guys, we can all go home.

Modius, go claim your million dollar prize.

• Reply • Share ›

plasma ➦ Modius • 25 days ago

Are you saying you found a solution that always solves the subset sum problem in polynomial time?

• Reply • Share ›

satty ➦ Modius • 25 days ago

can you explain or show your code about how you did it?

• Reply • Share ›

Aashish Karki • a month ago

Java Impl: https://ideone.com/jZQdnm

• Reply • Share ›

alex • 2 months ago

I try to change the implementation to return an array in C that will contain these that adds up to the sum . // Returns true if there is a subset of set[] with sun equal to given sum

int** isSubsetSum(int set[], int n, int sum)

```
int   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

int subset[sum+1][n+1];

int i, j;

// If sum is 0, then answer is true

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

subset[0][i] = 0;
```

see more

⌃ | ⌄ • Reply • Share ›

ajr • 4 months ago

How can this be modified if I have 3 arrays which must simultaneously add up to 3 given values?

1 ⌃ | ⌄ • Reply • Share ›

typed... ➦ ajr • 4 months ago

fb hacker rank problem, eh ?

1 ⌃ | ⌄ • Reply • Share ›

Sri ➦ ajr • 4 months ago

Hi, ajr
Call this method for every array and its associated sum,
and store the each result in different bool variable.At the end, you can
decide what you want by putting them in a "if" statement. Like, if(v1
&& v2 && v3).

⌃ | ⌄ • Reply • Share ›

np ➦ Sri • 3 months ago

But index should be same for all the three arrays how will you take care of that????

⌃ | ⌄ • Reply • Share ›

ravi • 5 months ago

here we are creation a 2-d array of size [sum+1][n+1].
what if sum is of the range 10^10. we can;t array of that size . Right?

• Reply • Share ›

typing.. ravi • 4 months ago

That's why it is mentioned here "pseudo polynomial time solution", if sum is too high, we have no other option than going recursively.

• Reply • Share ›

Doakes • 6 months ago

how to count no. of ways?

• Reply • Share ›

Ajay Gaur Doakes • 4 months ago

Then, it will be same as coin change problem - "Finding the no of ways we can get a change of a money from the set of coins."

and the solution for that will be :
SubsetSum(set, n-1, sum)+SubsetSum(set, n-1, sum-set[n-1])

These are the number of ways by including a coin plus the number of ways excluding that coin.

I guess, I'm not wrong.

• Reply • Share ›

piacentero • 6 months ago

I need a dynamic programming algorithm for this approach:
Find the longest subset of the given set with sum equal to given sum.
Thanksssss!!!!!!!!!!

• Reply • Share ›

antipiacentero piacentero • 2 months ago

sets have no inherent ordering, so im guessing you mean array

• Reply • Share ›

helper • 6 months ago

i really appreciate this approach to start with 0 and reach sum

• Reply • Share ›

helper helper • 6 months ago

and yeah, it reminded me of the named coin change problem.

1 • Reply • Share ›

Rajesh M D • 7 months ago

Anyone help me how to implement this if input array is having repeated elements?

^ | ⌄ • Reply • Share ›

richa • 8 months ago

can anyone plz tell me how to find the exact elements which sum upto given sum using dp solution rather than just verifying whether the given sum exists or not

^ | ⌄ • Reply • Share ›

Karstin Petersen ➦ richa • 6 months ago

I had this same question, ut if you look carefully, there is a way to reverse engineer the solution elements. The key here is the line:
subset[i - set[j-1]][j-1];

This is what propagates the true values forward, while the rest mostly just maintain the list. If you put in the coordinates for the values from which you want to find the elements (ie. if you want to find which elements ad up to 9, which will be at i=9 and some j that has a "true" value) into the above mentioned line of code, you'll find that it leads you to i = 4, with 5 being the added value. Do that again for i=4 and j=[some true field] until you reach the values on line 0. This is how I implemented it. I hope it's clearer than my words :P

```
int[] resultSet = new int[n]; // This could also be an arraylist.
int index = 0;
int i = sum; // start at target value.
int j = subset[0].length-1; // start at last position of row.
do{
while(subset[i][j] == true) // I try to keep to leftmost "true" value.
```

see more

2 ^ | ⌄ • Reply • Share ›

kaushik Lele • 8 months ago

Code for approach 2 of Dynamic programming is too compact to understand easily. So I expanded the steps and added few comments so that it is easier to grasp.
Giving below is only main nested loop part.
Once you have understood expanded code correctly try to combining/shortening the commands; you will reach original code.

```
// Fill the subset table in botton up manner
for (int i = 1; i <= sum; i++)   {
 for (int j = 1; j <= n; j++) {
   if(subset[i][j-1] == true){
   // it is possible to generate sum "i" from smaller subset itself.
   // So obviously it can be generated by bigger one. So no need to
   //    think more
   subset[i][j] = true;
```

```
    subset[i][j] = true;
    }else if (i == set[j-1]) {
          // Required sum is equal to current number. So mark it true
          subset[i][j] = true;
    }else if (i >= set[j-1]) {
```

see more

27 ^ | ⌄ • Reply • Share ›

Pankaj Joshi ➦ kaushik Lele • 4 months ago

awesome explain :D

^ | ⌄ • Reply • Share ›

Guest • 8 months ago

Can someone elaborate approach 2 of Dynamic programming ?

^ | ⌄ • Reply • Share ›

Guest ➦ Guest • 8 months ago

Ok I got it. Above code is very compact for initial comprehension.

I expanded the steps and added few comments so that it is easier to grasp.

Giving below only only nested loop part.

```
// Fill the subset table in botton up manner
for (int i = 1; i <= sum; i++)   {
 for (int j = 1; j <= n; j++) {
    if(subset[i][j-1] == true){
    // it is possible to generate sum "i" from smaller subset itself.
    // So obviously it can be generated by bigger one. So no need to
    //    think more
    subset[i][j] = true;
    }else if (i == set[j-1]) {
          // Required sum is equal to current number. So mark it true
          subset[i][j] = true;
    }else if (i >= set[j-1]) {
      //  sum "i" is bigger than current number set[j-1]
```

see more

1 ^ | ⌄ • Reply • Share ›

Guest ➦ Guest • 8 months ago

If you have understood expanded code correctly try to combining/shortening the commands; you will reach original code.

^ | ⌄ • Reply • Share ›

John • 8 months ago

Does my solution work for all cases?
http://ideone.com/ig6DkT

⌃ | ⌄ • Reply • Share ›

ALEX • 9 months ago

here is very simple solution using backtracking....Check it out.

i'm sure u'll like it...

http://ideone.com/G9kPAv

2 ⌃ | ⌄ • Reply • Share ›

ALEX ➦ ALEX • 9 months ago

If u have any doubt..let me 9... :)

⌃ | ⌄ • Reply • Share ›

Ankita Sahu ➦ ALEX • 5 months ago

why i<n? as="" we="" know="" size="" of="" int="" will="" be="" 4.="">

⌃ | ⌄ • Reply • Share ›

Priyal Rathi • 9 months ago

Another recursive approach:
from the start of array:
check if we can include this number to form the subset sub (recurse for rest of array with sum=sum-this num)
check rest for the array with the original sum (if we dont include this number in the subset sum)

Exponantial time

Link: http://ideone.com/BnIR8b

⌃ | ⌄ • Reply • Share ›

Ekta Goel • 9 months ago

What if some elements are negative?

1 ⌃ | ⌄ • Reply • Share ›

vidit • 9 months ago

can anybody please provide a description of how the above two solutions are going to work

⌃ | ⌄ • Reply • Share ›

krishna • 10 months ago

here is my implementation with O(n2)
http://ideone.com/OQ6zH4

⌃ | ⌄ • Reply • Share ›

Gaurav Gupta • 10 months ago

Is O(n^2) a brute force solution to this problem?

⌃ | ⌄ • Reply • Share ›

Goutham ➦ Gaurav Gupta • 10 months ago

In brute force solution , we are checking for each element in the array whether it will fit into the subset or not(2 options for each element, there are n elements). So the complexity will be O(2^n) => exponential complexity

1 ⌃ | ⌄ • Reply • Share ›

GeeksforGeeks Mod ➦ Gaurav Gupta • 10 months ago

Gaurav, the naive solution for this problem is exponential.

⌃ | ⌄ • Reply • Share ›

Pluth • 10 months ago

Here's explained in detail how algorithm works (boolean and improved 1-d DP table version): http://stringarray.net/?p=172

1 ⌃ | ⌄ • Reply • Share ›

Joey • 10 months ago

Are you sure your recursive solution works? Try the same set {3, 34, 4, 12, 5, 2} with sum = 0;

You program with return true which is wrong.

⌃ | ⌄ • Reply • Share ›

Ashu ➦ Joey • 10 months ago

Empty set is by default a subset of any set, so it is correct..

1 ⌃ | ⌄ • Reply • Share ›

vvkk • a year ago

botton up manner

```
for (int i = 1; i <= sum; i++)
{
for (int j = 1; j <= n; j++)
{
subset[i][j] = subset[i][j-1];
if (i >= set[j-1])
subset[i][j] = subset[i][j] || subset[i - set[j-1]][j-1];
}
}
```

set should be renamed to subset at various places

^ | ⌄ • Reply • Share ›

Guest • a year ago

```
bool subset(int set[], int size, int sum)

{

for(int i = 0; i<size; i++)="" {="" if="" (set[i]="" <="sum)" sum="" -="set[i];" }="" if(sum="=" 0)=""
return="" true;="" else="" return="" false;="" }="">
```

^ | ⌄ • Reply • Share ›

Guest • a year ago

The first problem, if replaced by:

```
bool subset(int set[], int size, int sum)

{

for(int i = 0; i<size; i++)="" {="" if="" (set[i]="" <="sum)" sum="" -="set[i];" }="" if(sum="=" 0)=""
return="" true;="" else="" return="" false;="" }="" ...works="" well="" in="" all="" the="" cases=""
i've="" tried="" so="" far.="" what="" is="" it="" i'm="" missing="" here?="">
```

^ | ⌄ • Reply • Share ›

vinnu • a year ago

Can we try this sum - ar[i] = key search the key value using binary search the only problem is if sum = 8 and array has 2 fours. Can anyone suggest me is it works with tweaking of binary search?

^ | ⌄ • Reply • Share ›

vrg • a year ago

Isn't the statement

```
if (set[n-1] > sum)
return isSubsetSum(set, n-1, sum);
```

in recursive solution redundant?
We are anyway handling both cases
(a) including the last element
(b) excluding the last element
in the statement
return isSubsetSum(set, n-1, sum) || isSubsetSum(set, n-1, sum-set[n-1]);

Can somebody explain why is it used?

^ | ⌄ • Reply • Share ›

guest11 ➦ vrg • a year ago

guest11 vrg • a year ago

it will avoid the last stmt where sum will become -ve ....that is not handled in the base case.......

or you can make an extra base case ..... if(sum <0) return 0;

^ | ⌄ • Reply • Share ›

guest11 vrg • a year ago

dont include the element which is already greater than sum

^ | ⌄ • Reply • Share ›

Vinay Singh • a year ago

```
//this is my solution
//O(nlogn)

#include<stdio.h>

int main()

{

int array[6]={3,6,4,1,5,2};

int n=6;

int i,j,t;

int f=0,u=5,k;

int sum=5;

int no;
```

see more

^ | ⌄ • Reply • Share ›

Load more comments

✉ Subscribe Add Disqus to your site Privacy

-
-
-
-
    - Interview Experiences
    - Advanced Data Structures
    - Dynamic Programming
    - Greedy Algorithms
    - Backtracking
    - Pattern Searching
    - Divide & Conquer
    - Mathematical Algorithms
    - Recursion
    - Geometric Algorithms
-
- ## Popular Posts
    - All permutations of a given string
    - Memory Layout of C Programs
    - Understanding “extern” keyword in C
    - Median of two sorted arrays
    - Tree traversal without recursion and without stack!
    - Structure Member Alignment, Padding and Data Packing
    - Intersection point of two Linked Lists
    - Lowest Common Ancestor in a BST.
    - Check if a binary tree is BST or not
    - Sorted Linked List to Balanced BST
- Follow @GeeksforGeeks
- ## Recent Comments
    - lt_k

i need help for coding this function in java...

Java Programming Language · 1 hour ago

- Piyush

  What is the purpose of else if (recStack[*i])...

  Detect Cycle in a Directed Graph · 1 hour ago

- Andy Toh

  My compile-time solution, which agrees with the...

  Dynamic Programming | Set 16 (Floyd Warshall Algorithm) · 1 hour ago

- lucy

  because we first fill zero in first col and...

  Dynamic Programming | Set 29 (Longest Common Substring) · 2 hours ago

- lucy

  @GeeksforGeeks i don't n know what is this long...

  Dynamic Programming | Set 28 (Minimum insertions to form a palindrome) · 3 hours ago

- manish

  Because TAN is not a subsequence of RANT. ANT...

  Given two strings, find if first string is a subsequence of second · 3 hours ago

-