

# Re: Summing combinations to a known total problem, for cash-allocation / open-item accounting

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2005-04/msg00331.html>

---

- *From:* [stephen@xxxxxxxxxx](mailto:stephen@xxxxxxxxxx)
  - *Date:* 1 Apr 2005 19:02:53 GMT
- 

Dave Lomax <billy\_boy\_clinton@xxxxxxxxxxxx> wrote:  
: Hi Randy,

: Thanks for your help. Unless I'm missing something, this method is  
: basically a "decision tree"? The trouble with this is that it tries  
: both combinations of £70 + £30, AND £30 + £70, which are obviously the  
: same. And, as suggested in a previous posting, we can trim the  
: "solution test set" at the beginning and end, if we use a binary table  
: method, and COUNT through the solutions.

: Is there any more to DYNAMIC PROGRAMMING, than just using a recursive  
: function? Am I missing something here?

: All help much appreciated :)

: Cheers,  
: Dave

One of the main ideas behind dynamic programming is avoiding solving  
the same sub-problem more than once. You do this by either  
computing the solution in a bottom up fashion using a table  
of some sort, or by using memoization.

Your problem looks like the change-making problem. The dynamic  
programming solution is an  $O(mn)$  algorithm where  $m$  is the total  
you are trying to reach, and  $n$  is the number of values.

Stephen

: "Randy Poe" <poespam-trap@xxxxxxxxxx> wrote in message  
news:<1112025282.758926.52500@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx>...

:> Dynamic programming has been suggested. Let me get a  
:> little more specific.

Re: Summing combinations to a known total problem, for cash-allocation / open-item accounting

:>  
:> Recursion works well on this problem.  
:>  
:> 1=2E Sort the n amounts in decreasing order: 75, 70, 30, 25.  
:>  
:> 2=2E Write a recursive routine (let's call it "subdivide")  
:> which takes a list of amounts in decreasing order, and  
:> a total to subdivide, and returns a division.  
:>  
:> div =3D subdivide(total, amounts)  
:>  
:> 3=2E subdivide does the following:  
:> a. If amount has only one entry see if you can  
:> divide the total into evenly spaced chunks  
:> of that size. (so a total of 50 with an amount  
:> of 25 would work, but a total of 45 would not).  
:> Return either the solution or a code that says  
:> "not possible".  
:>  
:> b. If amount has more than one entry, then take  
:> the largest entry. Try using 0 to the largest  
:> possible amount of that entry, and call subdivide  
:> with the remainder.  
:>  
:> Ex: subdivide(125,[75 70 30 25])  
:> try 0 75's, call subdivide(125, [70 30 25])  
:> recurse: try 0 70's, call subdivide(125,[30 25])  
:> try 1 70, call subdivide(55,[30 25])  
:> try 1 75, call subdivide(50,[70 30 25])  
:> recurse: try 0 70's, subdivide(50,[30 25])  
:>  
:> I hope that gives you the general scheme.  
:>  
:> - Randy  
.

---

• **Follow-Ups:**

◆ ***Re: Summing combinations to a known total problem, for cash-allocation / open-item accounting***

◇ From: Randy Poe

• Prev by Date: ***Re: JSH: Critique means slow, and thorough***

• Next by Date: ***Re: natural log of a summation***

• Previous by thread: ***Re: Distinct linear orderings on Z***

• Next by thread: ***Re: Summing combinations to a known total problem, for cash-allocation / open-item accounting***

• Index(es):

◆ ***Date***

◆ ***Thread***