## Subset Sum

Another problem that we will encounter again is the Subset Sum problem (SSP).  Given a set S = {s0, s1, …, sn-1} of n numbers (which may include duplicates), and a target number t, we want to find out whether some subset of these numbers sums to t.  Note that SSP is a Boolean function, it returns either true (if there is such a subset that sums to t) or false (if there is not).  For now, we don’t worry about how to find out which elements are in the subset.

Note that there are C0n + C1n + C2n + … + Cnn = 2n-1 possible subsets of any set of n things (including the null set).

## ****Question 1****(1 point)

*Saved*

Ow long does it take to do a brute force calculation to determine if there is a subset that sums to the target value t for a set of size n?

Question 1 options:

|  |  |
| --- | --- |
|  | O(n) |
|  | O(n2) |
|  | O(2n) |
|  | O(n!) |

## ****Question 2****(1 point)

*Saved*

If IMG_256 means the OR of k clauses, then one recurrence for the subset sum problem is SSP(S,t) = IMG_257 , since there must be some number in the solution, if there is one.

Here is an idea to speed up the computation.

Instead of recursively trying every subset minus each element as in the equation above, work backwards from the end.  Either the last element is in the sum, or it is not.  So instead of trying to solve *n* problems of size *n-1*, you are trying to solve 2 problems of size *n-1*.

Could you work forward from the beginning as well as working backwards from the end?

Question 2 options:

|  |  |
| --- | --- |
|  | Yes |
|  | No |

## ****Question 3****(1 point)

*Saved*

You know the value of every element si.  Calculate the value of every pair of numbers, then every triple of numbers, etc., in a bottom up fashion, until you either find the target sum or every calculation gives you a sum greater than the target sum.

In a sentence or two, what would you do to avoid recalculations for overlapping subproblems?