Fragments

1. Solve the equation

Central to this problem, is the solving of the problem:

$a_1 + a_2 + a_3 + ...a_f = N_0$; $a_1,...,a_f > 1$

where f is the number of fragments under consideration, and N_0 is the size of the bitstring that is being framented (in our application, it is 16.) The solutions to this are generated via Richard Bellman's notion of *dynamic programming*.

We may understand this approach very simply by solving it for four fragments: $f= 4$.

We first note that the solutions to:

$a + b = n$; $a, b > 1$

are: $a= 1, 2, ..., n - 1$, and $b= n - a$. Thus we can split the problem of finding the solutions to:

$a + b + c + d = N$

by finding the solutions to the sub-problems. We thus divide the problem and solve it in *levels*:

$(a + (b + (c + d))) = N$

This is similar to the drawing of a *tree*. This translates to a `for` loop, which generates all the solutions:

```
/*Generate_all_solutions_in_levels.*/
for( a= 1; a<=N-3; a++ )
{
/*b+c+d= N-a.*/
for( b= 1; b<=N-a-2; b++ )
{
/*c+d= N-a-b.*/
for( c= 1; c<=N-a-b-1; c++ )
{

printf( "\n(%d,%d,%d,%d)", a, b, c, N-a-b-c
);

}}}
```

This solution method is also used to compute the *number of solutions* to the equation, with pen and paper, and this is actually closely related to the *linear programming* example which Richard Bellman himself used to present his method in the original papers. It can be used to compute the *number of solutions* to this by hand.
Also, taking the idea from E.W. Dijkstra, who used the ALGOL notation to solve mathematical problems in some papers in his *Collected Writings*, we can write down the function to find the number of solutions. Let a variable `count` hold the number of solutions to a, b, c and d. We can find it by performing the computation:

```
/*Find_the_number_of_solutions.*/
for( a= 1; a<=N-3; a++ )
{
/*b+c+d= N-a.*/
for( b= 1; b<=N-a-2; b++ )
{
/*c+d= N-a-b.*/
for( c= 1; c<=N-a-b-1; c++ )
{

count++;

}}}
```

Now we can trace the value of `count` throught the `for` loop, and we get the expression:
$\sum_{1 \le a \le N\text{-}3} \sum_{1 \le b \le N\text{-}a\text{-}2} \sum_{1 \le c \le N\text{-}a\text{-}b\text{-}1} 1$
When we substitute N=16, as in our problem, we get 455, which is reasonable. This is also obtained by running the `for` loop above in a program:

```
#include//stdio
int main()
{
```

```
int N, a, b, c;
printf( "Enter the size of the bitstring:"
);
scanf( "%d", &N );

%/*Generate_all_solutions_in_levels.*/

int count=0;
%/*Find_the_number_of_solutions.*/
printf( "\n\nNumber of solutions: %d", count
);

return 0;
}
```

This should solve the problem for f=4. Next we may tackle the larger problem.

2.<u>The larger problem</u>
The actual problem is to find the *best possible* fragmentation scheme. This means that the value of f must be varied until we find the best scheme. This, however, is not so difficult, or time-inefficient, since we may use the method of an *updating threshold*, a common technique in *searching* and in various problems in electrical engineering. This should save time by a large amount, and, at the same time, use some of the best methods of the early days of artificial intelligence, without any difficulties.
...To be completed.