You are given an unsorted list of length $n$ whose items are the numbers $0..n+1$ with one number missing. Write a program to find the missing number.

Let the given list be $L$ (a constant), and its length is $n$ (a constant). Then

$L(0..n): 0..n+1 \land \forall i, j: 0..n \implies i \neq j \implies L_i \neq L_j$

Let $m: \text{nat}$ be a variable whose final value will be the missing number. The problem can be stated

$m': 0..n+1 \land \neg m': L(0..n)$

One way to solve the problem is with an extra list variable $M: [(n+1)*\text{bin}]$ to record which numbers are present in $L$. Here's an easier way: the problem is

$m' = (\Sigma[0..n+1]) - (\Sigma L)$

and its solution is

$m' = (\Sigma[0..n+1]) - (\Sigma L) \iff m := n \times (n+1)/2$. A$0 \implies A'n$

$A0 \implies A'n \iff \text{for } i := 0..n \text{ do } i: 0..n \land Ai \implies A'(i+1) \text{ od }$

$i: 0..n \land Ai \implies A'(i+1) \iff m := m - Li$

where $Ai \equiv m = (\Sigma[0..n+1]) - (\Sigma L[0..i])$.

Similarly the problem can be stated as

$m' = n + \Sigma i: 0..n \cdot i - Li$

and solved as

$m' = n + \Sigma i: 0..n \cdot i - Li \iff m := n$. B$0 \implies B'n$

$B0 \implies B'n \iff \text{for } j := 0..n \text{ do } j: 0..n \land Bj \implies B'(j+1) \text{ od }$

$j: 0..n \land Bj \implies B'(j+1) \iff m := m + j - Lj$

where $Bj \equiv m = n + \Sigma i: 0..j \cdot i - Li$. For either solution, recursive time is $t' = t + n$. 
