We want to find the smallest number in \( 0,..n \) with property \( p \). Linear search solves the problem. But evaluating \( p \) is expensive; let us say it takes time \( 1 \), and all else is free. The fastest solution is to evaluate \( p \) on all \( n \) numbers concurrently, and then find the smallest number that has the property. Write a program without concurrency for which the sequential to parallel transformation gives the desired computation.

§ We introduce array \( A: [n*bin] \). We define the desired result \( R \), condition \( I_i \), and helper specification \( P \) as follows.

\[
R = \neg (\exists j: 0,..h' \cdot pj) \land (ph' \lor h'=n) \\
I_i = \forall j: 0,..i \cdot Aj=pj \\
P = In \land \neg (\exists j: 0,..h' \cdot pj) \Rightarrow R
\]

Now the program is

\[
R \leftarrow I_0 \Rightarrow I'n. \ h:= 0. \ P \\
I_0 \Rightarrow I'n \leftarrow \text{for } i:= 0,..n \text{ do } I_i \Rightarrow I'(i+1) \text{ od} \\
I_i \Rightarrow I'(i+1) \leftarrow \text{Ai:= pi} \\
P \leftarrow \text{if } h=n \text{ then ok else if } Ah \text{ then ok else } h:= h+1. \ P \text{ fi fi}
\]

The \( n \) iterations of the \( \text{for} \)-loop can be executed in parallel.

We can express the result of the sequential to parallel transformation at source as follows.

\[
R \leftarrow I_0 \Rightarrow I'n. \ h:= 0. \ P \\
I_0 \Rightarrow I'n \leftarrow i:= 0. \ I_i \Rightarrow I'n \\
I_i \Rightarrow I'n \leftarrow \text{if } i=n \text{ then ok else } Ai:= pi \parallel (i:= i+1. \ I_i \Rightarrow I'n) \text{ fi} \\
P \leftarrow \text{if } h=n \text{ then ok else if } Ah \text{ then ok else } h:= h+1. \ P \text{ fi fi}
\]

To understand the execution, it might help to unroll the recursion a little: in the refinement of \( I_i \Rightarrow I'n \), replace the recursive call \( I_i \Rightarrow I'n \) by what's called \( \text{if } i=n \text{ then ok else } Ai:= pi \parallel (i:= i+1. \ I_i \Rightarrow I'n) \text{ fi} \). And maybe do the same once more.