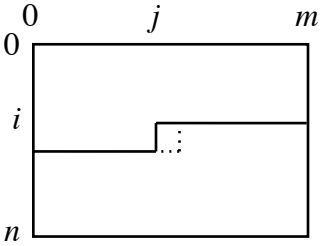
283 (largest true square) Write a program to find, within a two-dimensional binary array, a largest square subarray consisting entirely of items with value `T`.

After trying the question, scroll down to the solution.

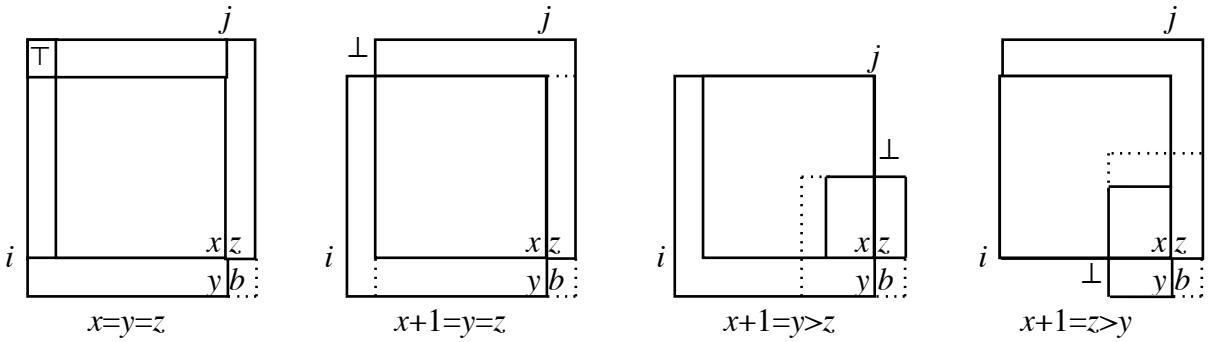
§ Let the array be A: [n*[m*bin]] . Let's try for a solution of the form

for i:= 0;..n
do for j:= 0;..m
do something **od od**

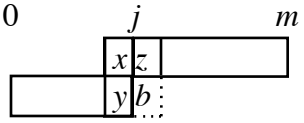
with execution time $n \times m$. (The **for**-loops (Chapter 5) are never necessary.) Part way through execution, we know the size of the largest true square in the upper region of the following picture, which is an informally written condition.



To increase the upper region by one item we need to know the size of the largest true square whose bottom right corner is the one new item. Imagine a new array B: [n*[m*nat]] so that $B\ i\ j$ is the length of a side of the largest true square with bottom right corner at $i\ j$. Clearly, $\neg\ A\ i\ j\ \Rightarrow\ B\ i\ j = 0$. Now suppose $A\ i\ j$. There are four cases to consider.



In all four cases, $b = x{\downarrow}y{\downarrow}z + 1$. We need only one row of the B array, plus three variables x , y , and z .



For each row, x and y start at 0 so that $x{\downarrow}y{\downarrow}z + 1 = 1$.

s:= 0.
for i:= 0;..n
do x:= 0. y:= 0.
for j:= 0;..m
do z:= $B\ j$.
B:= $j \rightarrow$ **if** $A\ i\ j$ **then** $x{\downarrow}y{\downarrow}z + 1$ **else** 0 **fi** | B.
x:= z. y:= $B\ j$ **od od**