Break Chocolate Bar into 1x1 pieces

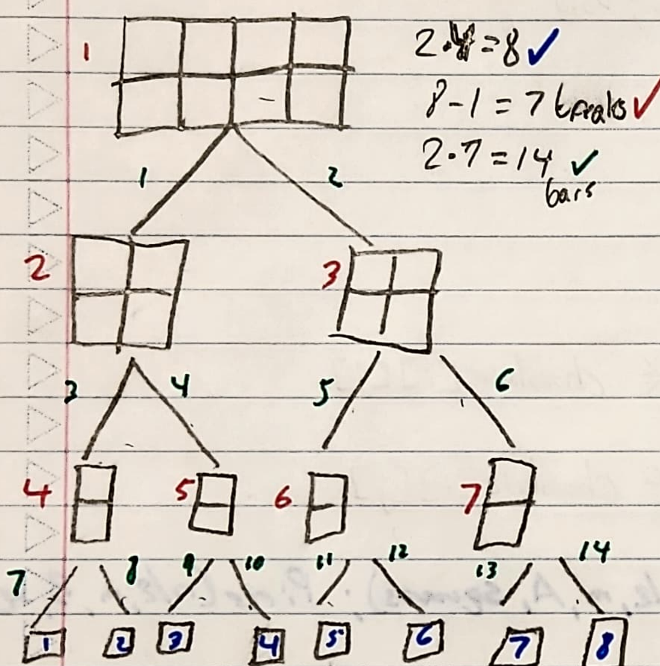
Piece

Input: n, m, chocolate [n][m], squares [nm] (ref/global)

Output: nm pieces in squares container; void

```
if n>1
    A[⌊n/2⌋][m]
    B[⌈n/2⌉][m]
    for i=0 ... n-1
        for j=0 ... m-1
            if i<n/2
                A[i][j] ← chocolate[i][j]
            else
                B[i][j] ← chocolate[i][j]
        exit for
    exit for; Piece(n/2, m, A, squares); Piece(n/2, m, B, squares);
else if m>1
    A[n][⌊m/2⌋]
    B[n][⌈m⌉]
    for j=0 ... m-1
        for i=0 ... m-1
            if j < m/2
                A[i][j] ← chocolate[i][j]
            else
                B[i][j] ← chocolate[i][j]
        exit for
    exit for; Piece(n, m/2, A, squares); Piece(n, m/2, B, squares);
else if n=1 & m=1
    push chocolate[0][0] onto squares
    return
```

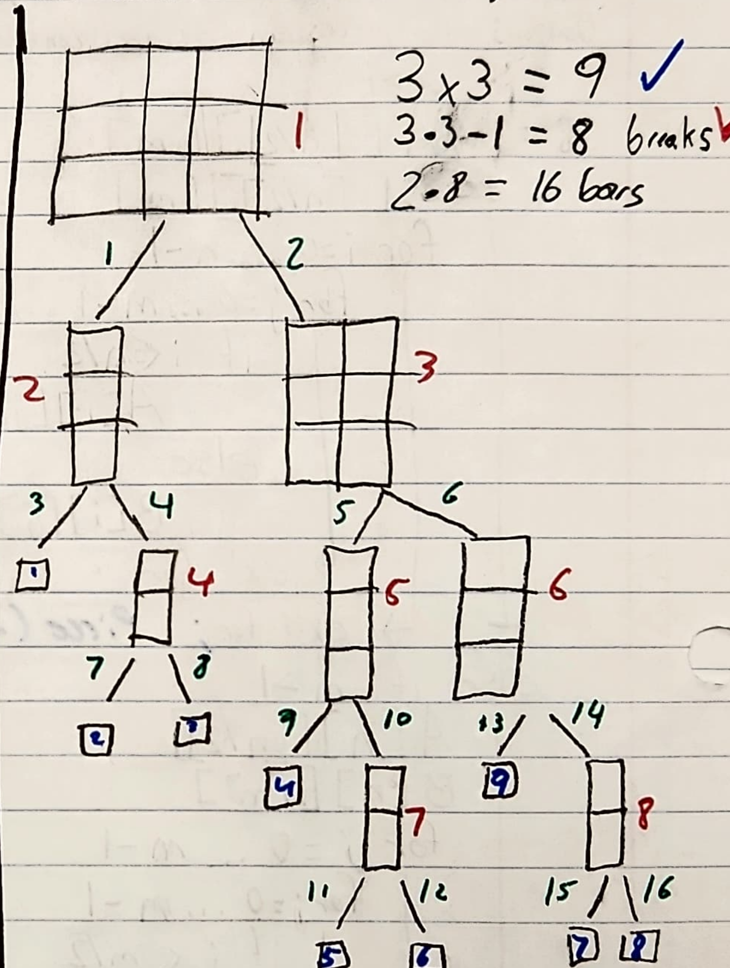
Because every cut yields two pieces, this has the structure of a binary tree / full binary tree



$2 \cdot 4 = 8$ ✓

$8 - 1 = 7$ breaks ✓

$2 \cdot 7 = 14$ ✓ bars



$3 \times 3 = 9$ ✓

$3 \cdot 3 - 1 = 8$ breaks ✓

$2 \cdot 8 = 16$ bars

This behavior means # of bars is the same as half the number of edges in a FBT.

edges = $2 \cdot$ nodes with 2 children

bars = $2 \cdot$ breaks

# of leaves/squares is always = #nodes with 2 children $+1$ (breaks)

$nm = b + 1$

$nm - 1 = b$

$$\begin{array}{r} 2 \cdot 4 = 8 \\ -1 \\ \hline 7 \text{ breaks} \end{array}$$