Assuming that we have a radius r, a height h and array A where a row denotes one cylinder with center x,y,z. Also assume no cylinder can be. A valid config is:

* 1. there is at least one cylinder with z axis equal to 0 on the “floor”.
  2. other cylinders have an x and y that is either less than or equal to r (e.g. cylinders can be stacked off-center but still touching and valid until greater than r.)
  3. other cylinders have a z value that when added is equal to the height h multiplied by n, the number of cylinders. (using the absolute value of the number, negatives included)

1. for i to n:

for j to n:

for k to n:

check if the last value of the matrix is 0

check to see if x,y <=r

check to see if sum of all z values = height \* n



**(Matrix-Matrix Multiplication on mesh of n­2 processors)**

**Input:**

1. Processor in mesh, P(i, j) for an integer i, j.
2. n tuples of from matricies A and B, A(i, k) and B(k, j), that hold the factors of the product C(i, j); some integer k.

**Output:** Process Pi, j will compute C(i, j) when receiving A(i, k) and B(k, j). All processorswill have a value C(i, j) that is part of the product of A and B.

1. Compute C(i, j) += A(i, k) + B(k, j). (+= is like x+=2 instead of x = x + 2)
2. Send A(i, k) to processor P(i, j+1) if exists.
3. Send B(k, j) to processor P(i+1, j) if exists.
4. Each value of processor P(i, j) is C(i, j), a submatrix of the product matrix C.

Notes: Algorithm runs synchronously instead of asynchronously. The diameter of the mesh, , ensures that every Step 1, which takes n2 operations takes now n operations across all the processors; Step 2 and 3 take constant time, excluding communication time.