Pebble algorithm: [Jacobs & Hendrickson 1997]

1. test 2k property: every k vertices induce ≤2k edges
   - each vertex has 2 attached pebbles
   - each pebble can cover 1 incident edge
   - free if not used to cover
   - goal: cover every edge

Claim: 2k property $\iff$ pebble cover

Proof:

$(\iff)$ edges induced by k vertices must be covered by 2k pebbles of those vertices
$\implies$ ≤2k induced edges

$(\Rightarrow)$ by correctness of algorithm below:
  - no pebble cover
  - algorithm below will fail
  - find vertex set violating 2k property $\square$
Algorithm:
- add edges one at a time
- view covered edge as directed from pebble
- for each added edge \( vw \):
  - search for directed path from \( v \) or \( w \) to free pebble
  - if found: shift pebbles (reverse edge)
- else: nodes reachable from \( v \) & \( w \) violate 2k property

Proof: no outgoing edges
\Rightarrow pebbles cover induced edges except \( vw \)
\Rightarrow > 2k edges among \( k \) vertices

Running time: \( O(V + E) \) per search
  \* \( O(V) \) searches
  = \( O(V^3 + VE) \)

\( \Leftrightarrow \) just check whether \( E > 2V \) at start
(\( \Rightarrow \) return NO)
Claim: $G$ has $2k-3$ property

$\iff G+3e$ has $2k$ property

add $3$ copies of $e$ for every edge $e$ in $G$

Proof: consider $k$ vertices,

$(\Rightarrow)$ $\leq 2k-3$ induced edges

if $e$ among them:

$G+3e$ induces $\leq 2k$ edges

else: still $\leq 2k-3 < 2k$ edges

$(\Leftarrow)$ if no induced edges: done

else: add $3$ copies of induced edge results in $\leq 2k$ induced edges

remove $3$ extra copies

$\Rightarrow \leq 2k-3$ induced edges $\square$

$O(V^3)$ algorithm: call previous on $G+3e\forall e$

$O(V^2)$ algorithm: incremental as above

- for each added edge $e$:

  - add $4$ copies of $e$ as above

  - if succeed: remove $3$ copies of $e$
    (freeing $3$ pebbles)

  - if fail: remove all $4$ copies of $e$
    mark edge as redundant

  - gen. rigid $\iff 2n-3$ nonredundant edges
Implementation [Audrey Lee]

Generalization to a \( k-b \) property
[Lee & Streinu - Discr. Math. 2008]

Rigid component decomposition:
[above paper + Lee, Streinu, Theran - CCCG 2005]
roughly, component = what you can reach, including backward edges if reachable
component on other side has no free pebbles

Body & bar frameworks:
- generically rigid in \( d \)-D
\[ \iff \text{graph has } a_k-a \text{ property} \]
where \( a = \frac{d(d+1)}{2} = 6 \) in 3D

[Tay 1984 + Nash-Williams/Tutte (indep.)]
- can also support hinges (3D):
  equivalent to 5 bars

Angular rigidity: [Lee-St. John & Streinu - CCCG 2009]
- lines/planes & angles: angles min. gen. rigid
  \[ \iff \text{constraint graph is Laman in } 3D \]
- bodies & angles: angles gen. rigid in 3D
  \[ \iff \text{constraint graph has } 3k-3 \text{ property} \]
5-connected double bananas: [Mantler & Snoeyink - 2004]

- in fact, any graph can be made 5-connected while preserving Laman & generic flexibility
  - just add spiders: