6.849	Class 7	Sept. 27, 2012

- Box pleating history
  - Mooser's train [Raymond McLain, 1967]
  - Black Forest Cuckoo Clock [Lang 1987]

- OPEN: universal folding of e.g. polytetrahedra or polyoctahedra from triangular grid?

- Maze folding examples
  - our print designs

- Meaning of NP-hardness:
  - doesn't mean anything about specific instances
  - about scaling of running time as problem size n grows
  - e.g. 8×8 Chess is "trivial"
    $n\times n$ Chess is EXP-hard
    $\Rightarrow$ running time scales exponentially

- Simple fold hardness review:
  - convert Partition instance $(a_1, a_2, \ldots, a_n)$ into equivalent simple-fold instance (polygon + creases)
    solution for Partition exists
    $\Leftrightarrow$ solution for simple folds exists

($\Leftarrow$) vertical creases will bind otherwise

($\Rightarrow$) fold creases between a_i & a_{i+1} when in different halves
fold both vertical creases
fold rest

- Flat foldability hardness review:
  - convert NAE triples into crease pattern

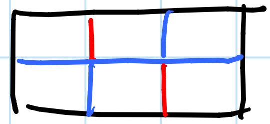
  (⇐) gadgets force NAE constraints
  read T/F assignment off M/V assignment

  (⇒) verify gadgets <u>do</u> fold as needed
  patch together (glue) foldings together

  OPEN: simpler proof? [Tom Hull]

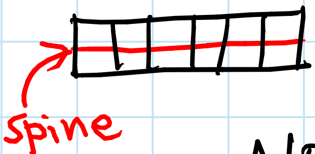
- NP-hardness even given M/V assignment:
  [Bern & Hayes 1996]

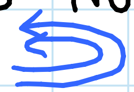
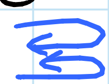
Map folding: (nonsimple folds, unlike L2)



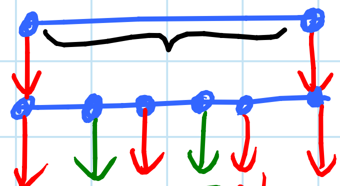
- horizontal & vertical creases in rectangular paper
- given M/V assignment, does it fold flat?
- OPEN: polynomial? NP-hard?
  [posed by Edmonds 1997]

2×n has polynomial-time algorithm
[Demaine, Liu, Morgan 2012]
(from 6.849 project in 2010)



- NEWS labeling: for each vertex, mark which eminating crease is different
- top edge view: top of folded map = N & S sides of unfolded map
  - nested pairings from map spine
  - N = left turn 
  - S = right turn 
  - E = "in" 
  - W = "out" 

- ray diagram: [Charlton & Zhou, 6.849, 2007]
  - follow map spine (merging N & S sides)
  - y coord. = "nesting depth"; x coord. flexible
  - E = down turn ↩/↪ – W = up turn ↩/↪
  - N & S shoot downward rays ↓N ↓S
  - rules: (equivalent to flat folding)
    - spine doesn't self-intersect
    - N rays must hit N rays or go to ∞
    - S rays ditto
    - constrained spine segment (with no view to infinity below it) have equal number of N & S vertices below it



- spaces between spine in ray diagram forms a tree structure
- "guess" this tree structure (effectively trying them all) using dynamic programming