6.890	Lecture 2	Sept. 9, 2014

* Two NP-complete problems useful for reducing to arithmetic (summing) problems:

(2-)Partition: given integers $A = \{a_1, a_2, \ldots, a_n\}$, partition A into two sets $A = A_1 \dot\cup A_2$ of equal sum: $\frac{\sum A}{2} = \sum A_1 = \sum A_2 = t$

[Karp 1972]

Generalization: Subset Sum

$t = \frac{\sum A}{2}$

given integers $A = \{a_1, a_2, \ldots, a_n\}$, and a target integer t, find a subset $S \subseteq A$ of sum $\sum S = t$

3-Partition: given integers $A = \{a_1, a_2, \ldots, a_n\}$, partition A into $n/3$ sets A_i of equal sum, $\sum A / (n/3) = \sum A_i = t$

- can assume each $a_i \in (t/4, t/2)$

$\Rightarrow$ each set A_i contains exactly 3 items

[Garey & Johnson — SICOMP 1975]

$\Rightarrow$ can make each a_i close to $t/3$: add huge number $(n^{100} \cdot \max A)$ to each a_i

Garey & Johnson [book] reduce

3SAT $\to$ 3DM $\to$ 4-partition $\to$ 3-partition

$\to$ numerical 3DM

Variation: Numerical 3-dimensional matching

given integers $A = \{a_1, a_2, \dots, a_n\}$,
$B = \{b_1, b_2, \dots, b_n\}$,
$C = \{c_1, c_2, \dots, b_n\}$

partition into n triples $S_i \in A \times B \times C$
of equal sum $t = \Sigma(A \cup B \cup C) / n$

[Garey & Johnson – SICOMP 1975]

Reduction to 3-partition: (so it's simpler)

- add $\varepsilon \ll 1$ to each a_i e.g. $\varepsilon = 1/4$
- add $\delta \ll \varepsilon$ to each b_i $\delta = 1/16$
- subtract $\varepsilon + \delta$ from each c_i
- scale back to integers $\times 16$
- in sum of 3, δ never becomes ε & ε never becomes 1

$\Rightarrow$ ε & δs must cancel algebraicly

cf. (2D) matching

Generalization: 3-dimensional matching (3DM)

given a tripartite hypergraph with vertices $A \dot\cup B \dot\cup C$, $|A| = |B| = |C| = n$,
& hyperedges $E \subseteq A \times B \times C$,
find n disjoint edges $S \subseteq E$
(which must partition the vertices)

[Karp 1972]

Generalization: Exact Cover by 3-sets (X3C)

given 3-uniform hypergraph (V, E),
$\forall e \in E : |e| = 3$
find $|V|/3$ disjoint edges ($\Rightarrow$ partition V)

* Two types of NP-hardness for number problems:
  involving integers, not just combinatorics

Weakly NP-hard = NP-hard
- allow numbers to have value exponential in n
- encoding length $= \log(2^{n^c}) = n^c$ still polynomial

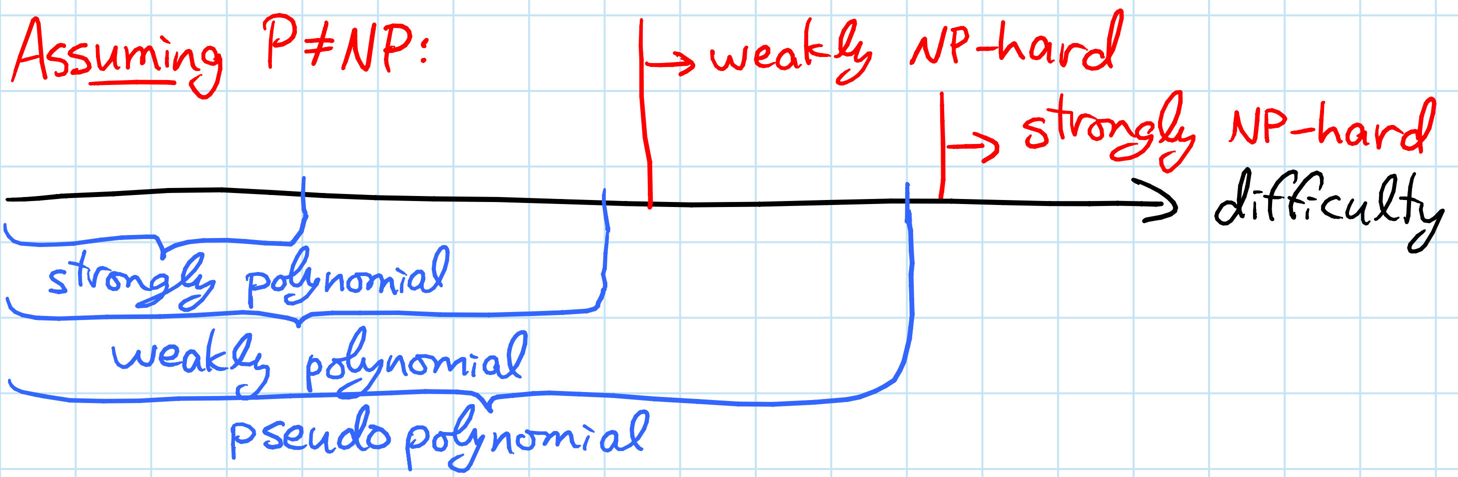
Strongly NP-hard = NP-hard even when restricted to numbers with value polynomial in n
(i.e. even if numbers encoded in unary)

* Corresponding algorithmic notions:

Pseudopolynomial = polynomial in n & largest number (unary encoding)
Weakly polynomial = polynomial = polynomial in n & log(largest number)
Strongly polynomial = polynomial in n
↳ # numbers

Weak NP-hardness precludes polynomial algorithm (assuming $P \neq NP$) but leaves possible pseudopolynomial

Assuming $P \neq NP$:



Multiprocessor scheduling: [Garey & Johnson – SICOMP 1975]

- given n jobs with processing times $a_1, a_2, \ldots, a_n$
- given p processors (each sequential & identical)
- assign jobs to processors to minimize maximum completion time (makespan)
- **decision version:** can all processors finish by $\leq t$?
- **NP certificate:** job → processor mapping

(a_i as is)

Reduction from Partition: $p = 2$ ⇒ weakly NP-hard

Reduction from 3-Partition: $p = n/3$ ⇒ strongly NP-hard

(This was Garey & Johnson's motivation for introducing 3-partition in 1975.)

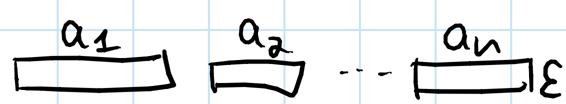
Claim: jobs finishable in makespan t → target sum

⇔ (3-)Partition instance has a solution

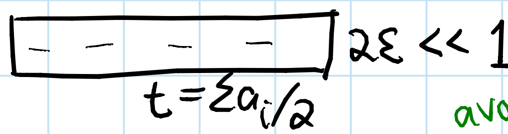
Rectangle packing:

- given n rectangles $\nearrow A$ & target rectangle $\nearrow B$
- can you pack former into latter?
  ↳ rotate & translate to fit without overlap
- $\boxed{\text{OPEN}}$: $\in NP$?
- special case: exact packing — no gaps
  ↳ hardness result is stronger theorem
  - rotation $\in \{0, 90°, 180°, 270°\}$, translation integral
    (proof by induction: consider corner, repeat)
  - NP certificate: translations & rotations

Reduction from Partition: $A =$

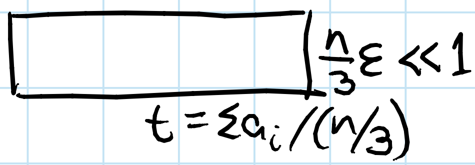


$B =$



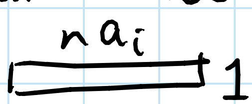
avoid rotation

Reduction from 3-Partition: $B =$

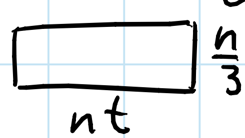


Scaling trick to make all dimensions integral:

$A = \{$

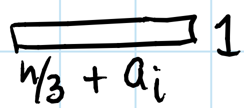


$\}$, $B =$

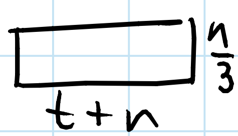


Here, just adding $n/3$ to each a_i suffices:

$A = \{$



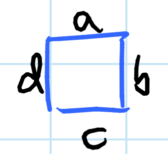
$\}$, $B =$



[Demaine & Demaine – G&C 2007]

Edge-matching puzzles: [Demaine & Demaine – G&C 2007]

- given unit square tiles,
  each side labeled with a "color"

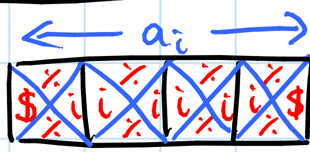


- given target rectangle
- <u>goal</u>: put tiles in target such that
  tiles sharing an edge have matching colors

<u>No numbers</u> ⇒ can't use Partition!

<u>Reduction from 3-Partition</u>: (like rect. packing)

- a_i gadget:



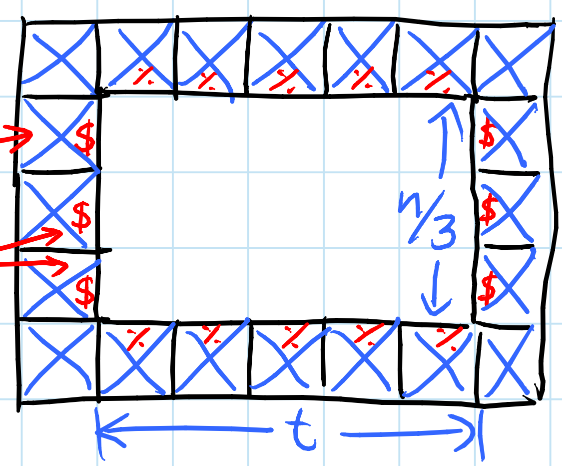
← effectively unary encoding!

← prevents rotation

  - if i colors go together, forced to make this
  - but some could go on boundary...
- frame gadget:
  ("<u>infrastructure</u>")

unique

unique pair



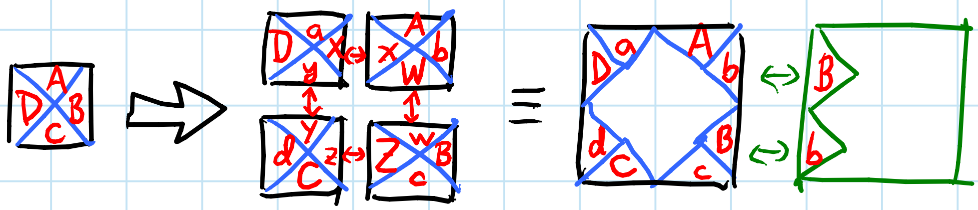
add more rows to make target square

  - unique colors forced on boundary
    ⇒ frame construction forced
- target shape: $(n/3 + 2) \times (t+2)$
  ⇒ a_i construction forced (no boundary left)
  ⇒ effectively rectangle packing

Signed edge-matching puzzles: (lizards etc.)

- colors come in matching pairs:
  a & A, b & B, etc.
- color does not match itself ~ only its mate

Reduction from unsigned edge-matching puzzles:



- interior colors (x, y, z, w) are unique pairs
$\Rightarrow$ must assemble 2×2
(assuming frame to prevent boundary use)
$\Rightarrow$ acts like unsigned tile

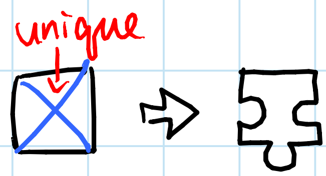
Jigsaw puzzles: [Demaine & Demaine – G&C 2007]
- no guiding picture
- ambiguous mates (fitting $\not\Rightarrow$ correct)

Reduction from signed edge-matching puzzles:

shape

lower case → pocket
upper case → tab

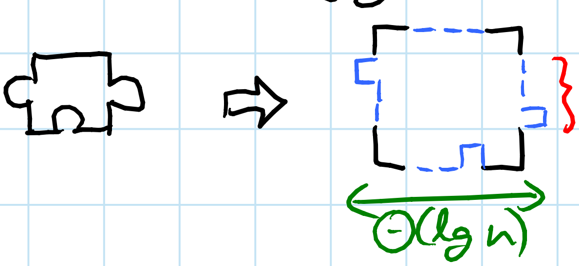
- for rectangular boundary:
  ↳ even square



Polyomino packing: [Demaine & Demaine – G&C 2007]
- given polyominoes = edge-to-edge joinings of unit squares (like Tetris)
- given target rectangle
- goal: exact pack former into latter

- rectangle packing is a special case ⇒ done
- but piece areas are $> n$
- what if areas are polylog?
- $\boxed{\text{OPEN}}$: logarithmic area

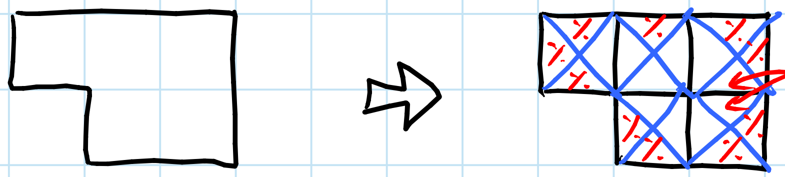
Reduction from jigsaw puzzles:



binary encoding of color
– can get equal areas

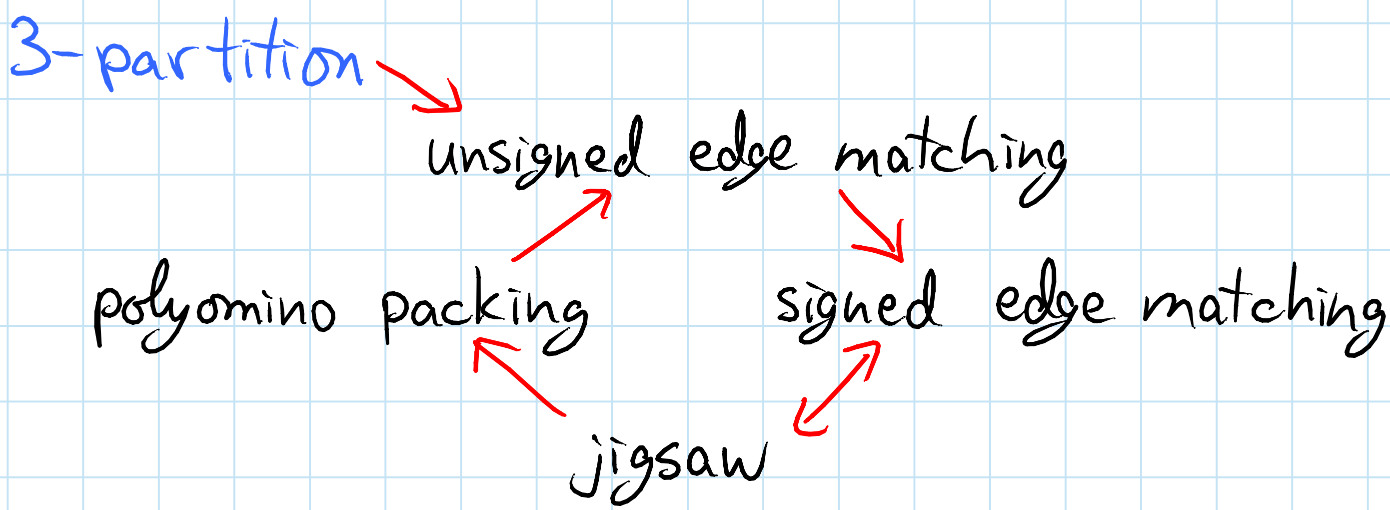
Closing the loop: [Demaine & Demaine – G&C 2007]

reduction from polyomino packing
to unsigned edge-matching puzzles



– use frame, but with \$ = %

So: all 4 puzzle types are NP-complete
& constant-factor equivalent: can convert one to the other with O(1) factor blowup



(exact)

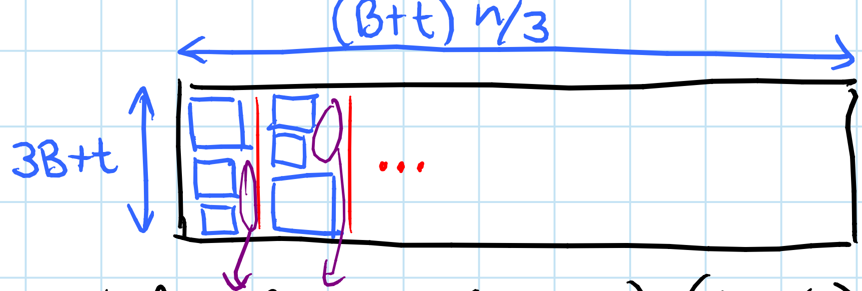
Packing squares into a square: strongly NP-complete

[Leung, Tam, Wong, Young, Chin – JPDC 1990]

- motivation: scheduling square jobs on grid supercomputer

Rectangle target:

- squares of dimension $a_i + B$ ← huge ⇒ ≈ B
- pack into rectangle of height ≈ 3B:

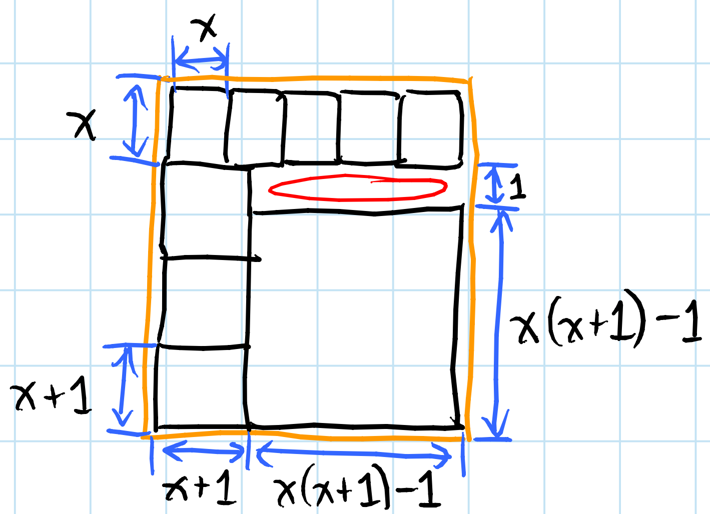


- total slop $\le (3B+t)\cdot(t\,n/3)$
  $< B^2 <$ one square
  if $B > tn$
  ⇒ "doesn't help"

Exact packing: add 1×1 □s to fill extra area

Square target:

- infrastructure to build rectangular space



- scale by 3B+t
- set x large enough to get enough width
- pad excess with B×B squares