Lecture 3: Reductions from 3-partition

* 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: $\frac{\Sigma A}{2} = \Sigma A_1 = \Sigma A_2 = t$

[Karp 1972]

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

- 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

3-Partition along with 3-SAT might be the most important problems to prove NP-hardness.

* 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)

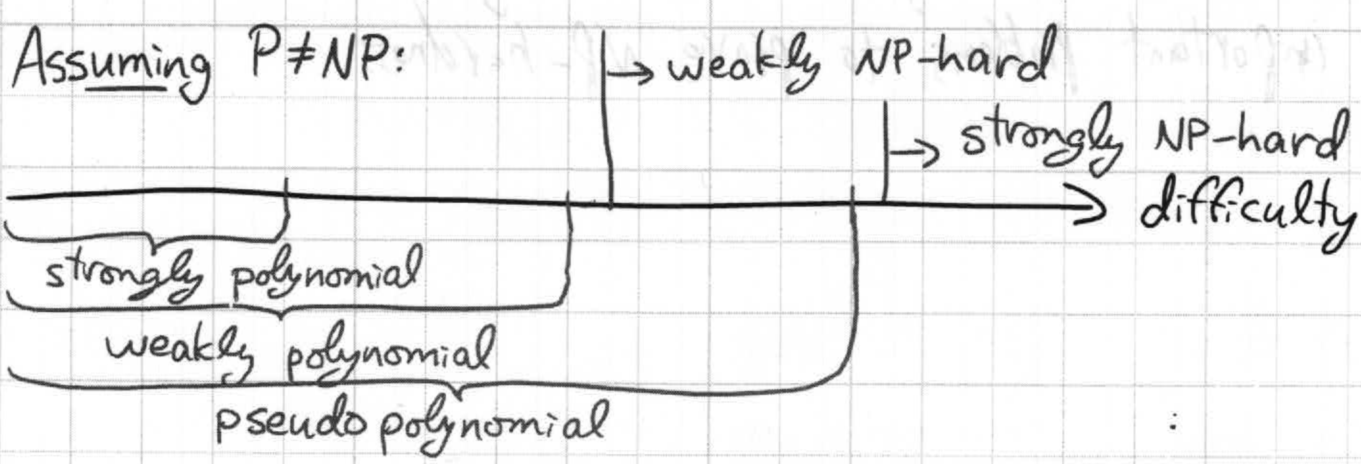
this happens in other places as well, e.g., approximation algorithms

* 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
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)

(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 & target rectangle ($\nearrow A$, $\nearrow B$)
- can you pack former into latter?
  - ↳ rotate & translate to fit without overlap
- special case: exact packing — no gaps
  - ↳ hardness result is stronger theorem

Reduction from Partition: $A =$ a_1 a_2 $\cdots$ a_n ε

$B =$ $2\varepsilon \ll 1$, $t = \Sigma a_i/2$

too avoid rotation

Reduction from 3-Partition: $B =$ $\frac{n}{3}\varepsilon \ll 1$, $t = \Sigma a_i/(n/3)$

Scaling trick to make all dimensions integral:

$A = \{$ $n a_i$ 1 $\}$, $B =$ nt $\frac{n}{3}$ (for 3-Partition)

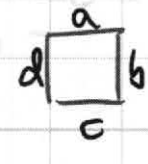
Here, just adding $n/3$ to each a_i suffices:

$A = \{$ $n/3 + a_i$ 1 $\}$, $B =$ $t+n$ $\frac{n}{3}$

[Demaine & Demaine — G&C 2007]

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

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

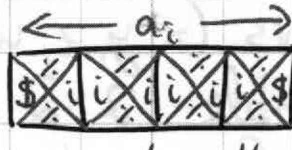


- given target rectangle
- goal: put tiles in target such that tiles sharing an edge have matching colors

No numbers ⇒ can't use Partition!

Reduction from 3-Partition: (like rect. packing)

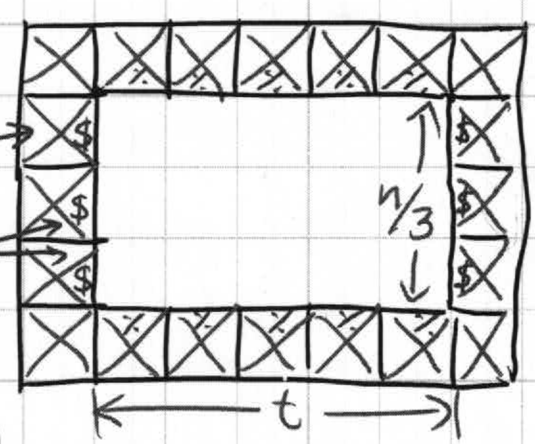
- a_i gadget:



← effectively unary encoding!
← prevents rotation

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

unique
unique pair



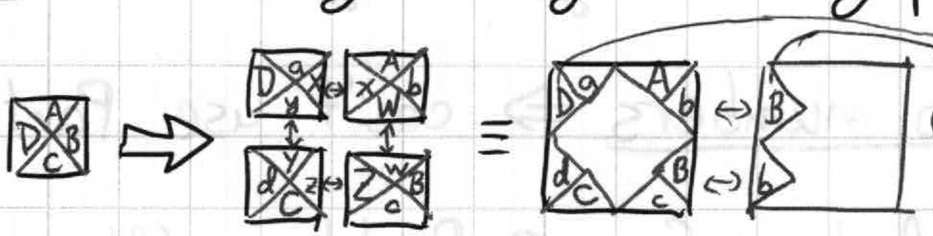
add more rows to make target square (instead of triangle)

- 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:



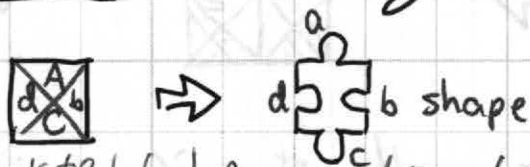
Note that Capitals are consistent by this construction

- 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

(e.g. for a map, a face, a picture)

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

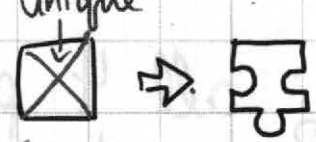
Reduction from signed edge-matching puzzles:



lower case → pocket
upper case → tab

Note that pocket & tab shapes are different for different colors.

- for rectangular boundary:
  ↳ even square

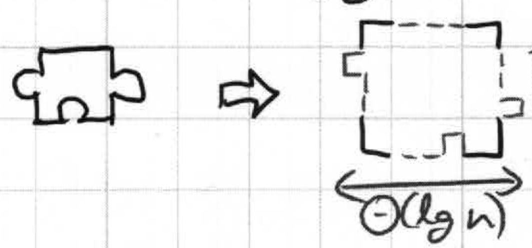


OR alternatively they (pocket & tab) can have matching unique colors.

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: (in term of cycle of reductions) [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

3-partition → unsigned edge matching
unsigned edge matching → signed edge matching
signed edge matching ↔ jigsaw
jigsaw → polyomino packing
polyomino packing → unsigned edge matching

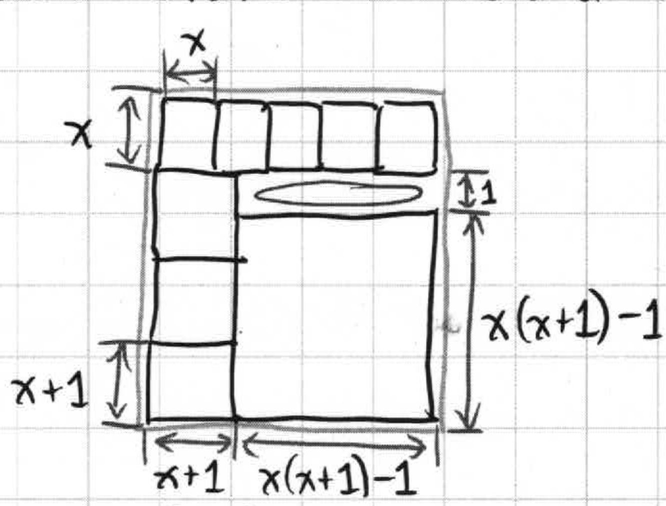
Demaine & Demaine G&C 2007 =
http://erikdemaine.org/papers/Jigsaw_GC/

Packing squares into a square: strongly NP-complete

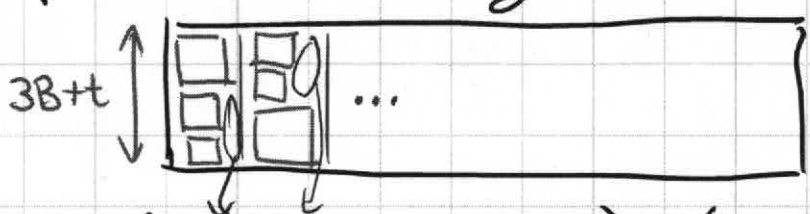
[Leung, Tam, Wong, Young, Chin – JPDC 1990]

- motivation: scheduling square jobs on grid supercomputer

- infrastructure to build rectangular space



- squares of dimension $a_i + B$ ← huge $\Rightarrow \approx B$
- pack into rectangle of height $\approx 3B$:



- total slop $\le \max(A) \cdot (3B+t)$
  $< B^2 <$ one square
  if $B > 4\max(A)$

$\Rightarrow$ "doesn't help"