CHAIN SIMPLIFICATION: "ITERATIVE ENDPOINTS FIT"

input:
① a chain
② an error tolerance
③ a distance definition (between points & segments)

Algorithm:
\[\text{simplify}(a,b) \quad \text{// chain from } a \text{ to } b\]
\[\text{if } \overline{ab} \text{ doesn't approximate chain}(a,b) \text{ "well" }\]
\[\text{find } v \in \text{chain}(a,b) \text{ w/ MAX dist}(v,\overline{ab})\]
\[\text{simplify}(a,v)\]
\[\text{simplify}(v,b)\]
\[\text{else output } \overline{ab}\]
CHAIN SIMPLIFICATION: "ITERATIVE ENDPOINTS FIT"

- Time complexity
  - Testing $\overline{ab}$: linear time (size of chain)
  - Worst case: unbalanced recursion

$\mathcal{O}(n^2)$

Algorithm:

simplify$(a, b)$ // chain from $a$ to $b$
if $\overline{ab}$ doesn't approximate chain$(a, b)$ "well"
  find $v \in$ chain$(a, b)$ with $\text{MAX dist}(v, \overline{ab})$
  simplify$(a, v)$
  simplify$(v, b)$
else output $\overline{ab}$
do we always get a simple (non-crossing) chain?

vertex too far from red edge
do we always get a simple (non-crossing) chain?
do we always get a simple (non-crossing) chain?
do we always get a simple (non-crossing) chain?
do we always get a simple (non-crossing) chain?

within error tolerance

(ok maybe the threshold should be a bit greater)
do we always get a simple (non-crossing) chain?
do we always get a simple (non-crossing) chain?

both vertices within tolerance
do we always get a simple (non-crossing) chain?  \[\text{No}\]

variants (possibly unexplored)

- require non-crossing output
- deal with multiple error violations

\[\text{use median "offender"?}\]

\[\gg 2\text{ recursive calls?}\]
Does the algorithm minimize the number of segments used?

No. can be made arbitrarily worse.
IRI-IMAI algorithm to minimize segments  (given path & tolerance)

- form directed graph:
  \[ \overrightarrow{v_i v_j} \iff i < j \]

- give a score to each edge:
  \[ \#	ext{path edges skipped} \]

- keep only edges satisfying tolerance
IRI-IMAI algorithm to minimize segments (given path & tolerance)

1. Form directed graph: 
   - make $\overrightarrow{v_i v_j}$ iff $i < j$

2. Give a score to each edge: 
   - #path edges skipped

3. Keep only edges satisfying tolerance

$$\Theta(V^2) \text{ edges } \underbrace{1}_{2} \underbrace{\Theta(V^2)}_{3} \underbrace{\Theta(V^2) \cdot O(v)}_{\text{time complexity}}$$
IRI-IMAI algorithm to minimize segments  (given path & tolerance)

1. Form directed graph:
   - make $\overrightarrow{v_i v_j}$ iff $i < j$

2. Give a score to each edge:
   - # path edges skipped

3. Keep only edges satisfying tolerance

$\Theta(V^2)$ edges $\leq \frac{1}{2} \Theta(V^2)$ $\gg \frac{3}{3} \Theta(V^2) \cdot O(V)$

- max-weight path in a DAG
  - min-weight w/ scores inverted
  (no neg. cycles: ok)
  $O(E)$

- Simple (unweighted) BFS
IRI-IMAI algorithm to minimize segments

Suppose we are willing to use up to \( k \) edges. Then the goal is to minimize error.
IRI-IMAI algorithm to minimize segments

Suppose we are willing to use up to \( k \) edges. Then the goal is to minimize error.

- Build the full DAG \( \Theta(V^2) \)
- Sort all edges by error \( \Theta(V^2 \log V) \)
- Binary search on error \( \log(V^2) \)

For each error value:
- Trim graph \( \Theta(V^3) \)
- Test \( O(E) \)

Total time \( \mathcal{O}(V^3 \cdot \log V) \) + a dynamic programming approach to be written up.
IRI-IMAI algorithm to minimize segments (given path & tolerance)

Testing an edge that connects endpoints of a path w/ m segments

\[ \Rightarrow \text{brute force: } \Theta(m) \text{ time} \]

\[ \Rightarrow \text{under certain conditions (e.g. parallel strip distance) } \]
\[ \& w/ \text{preprocessing, we can do better} \]
(\text{and thus beat } O(V^3) \text{ overall})