CHAIN SIMPLIFICATION : "ITERATIVE ENDPOINTS FIT"

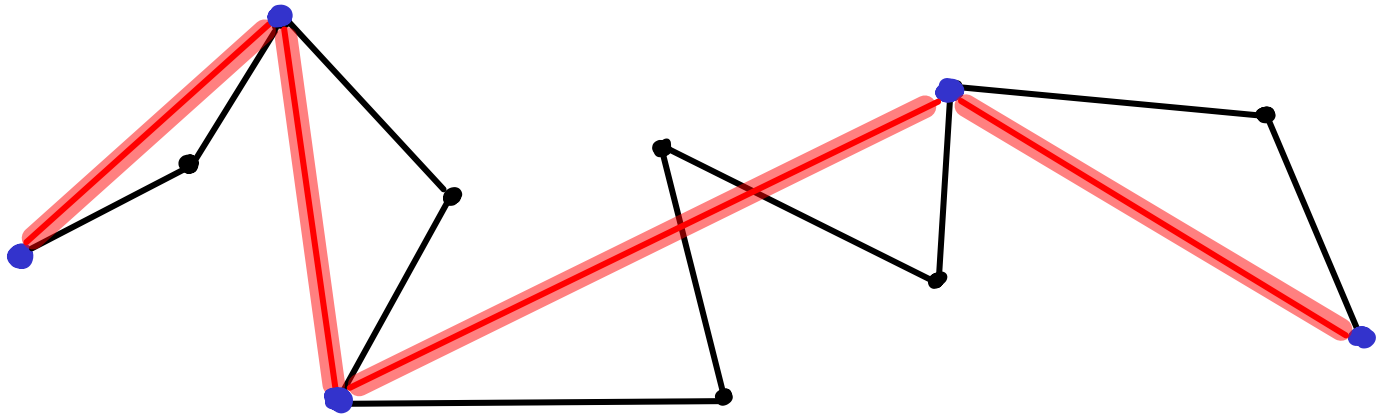
input:

① a chain

② an error tolerance

③ a distance definition

(between points & segments)



Algorithm:

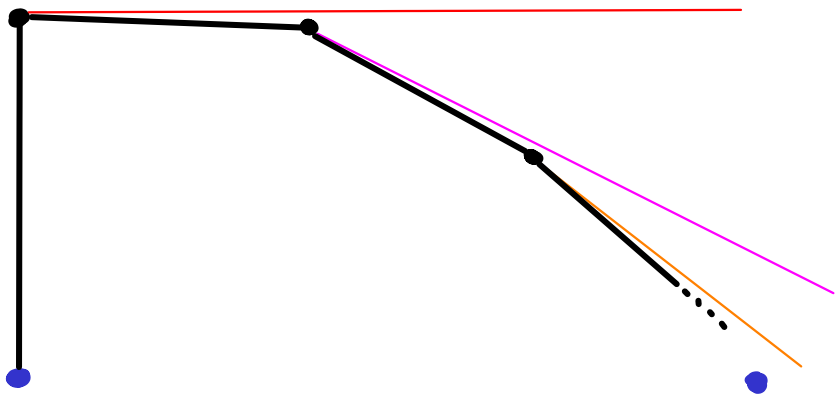
```
simplify(a,b)   // chain from a to b
  if ab doesn't approximate chain(a,b) "well"
    find v ∈ chain(a,b) w/ MAX dist(v,ab)
    simplify(a,v)
    simplify(v,b)
  else output ab
```

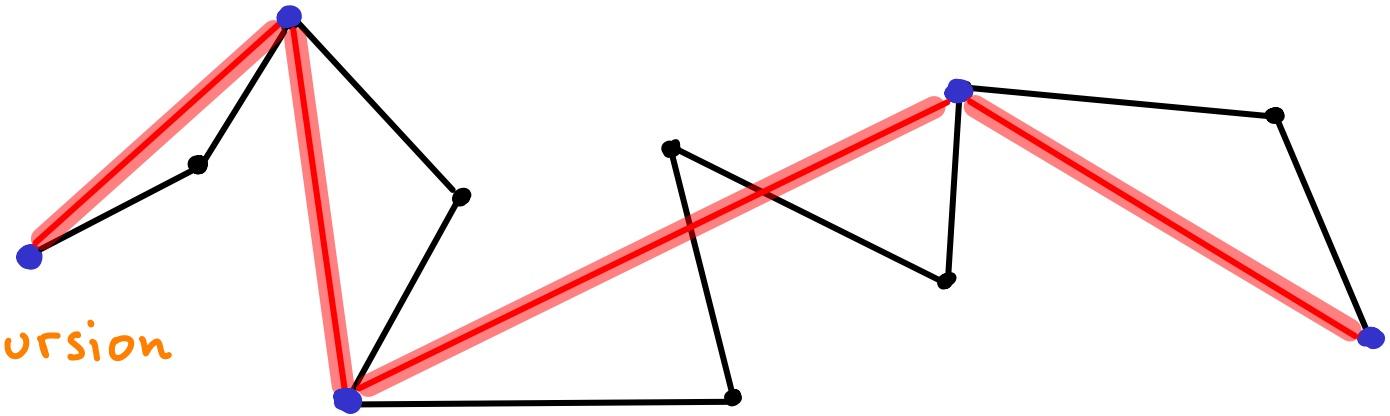
CHAIN SIMPLIFICATION : "ITERATIVE ENDPOINTS FIT"

Time complexity

- testing $\overline{ab}$: linear time (size of chain)
- worst case : unbalanced recursion

↳ $O(n^2)$

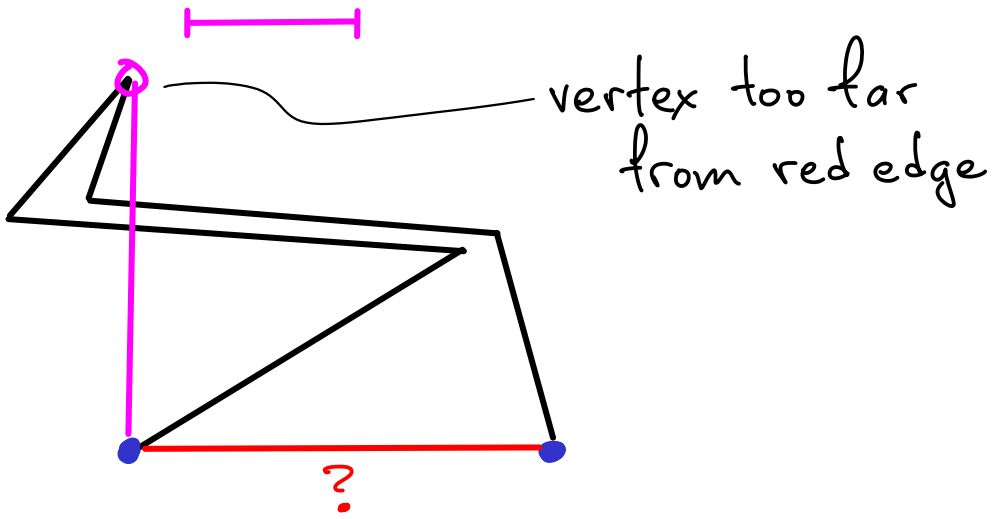




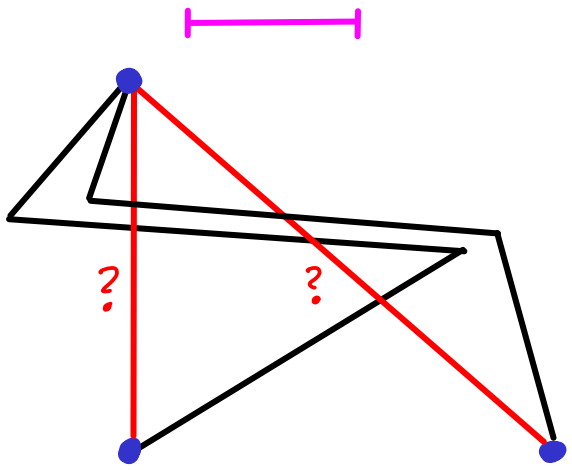
Algorithm:

```
simplify(a,b)   // chain from a to b
  if ab doesn't approximate chain(a,b) "well"
    | find v ∈ chain(a,b) w/ MAX dist(v, ab)
    | simplify(a,v)
    | simplify(v,b)
  else output ab
```

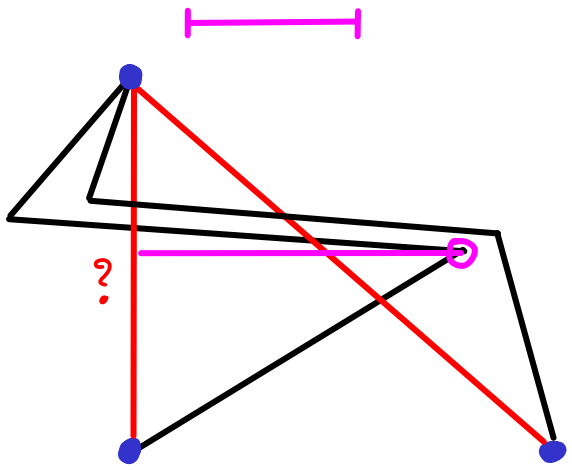
do we always get a simple (non-crossing) chain?



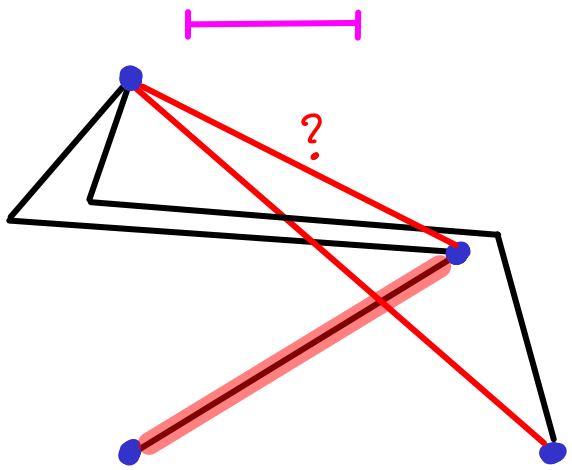
do we always get a simple (non-crossing) chain?



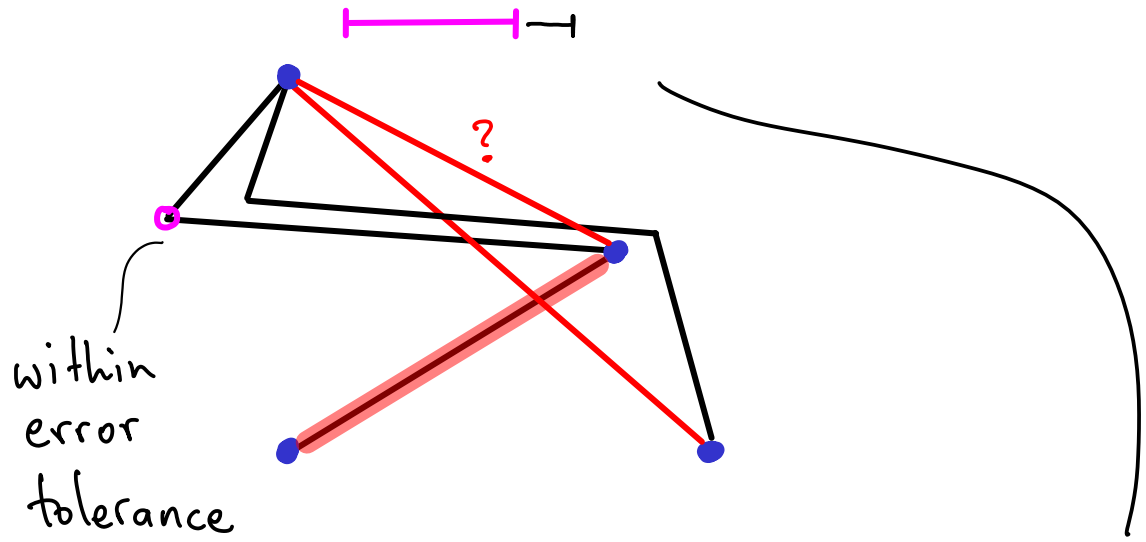
do we always get a simple (non-crossing) chain?



do we always get a simple (non-crossing) chain?

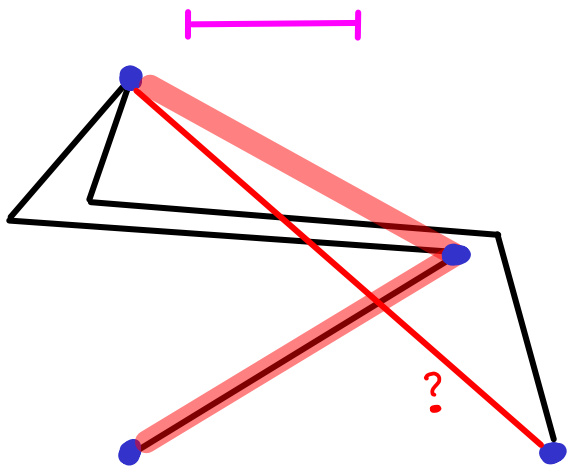


do we always get a simple (non-crossing) chain?

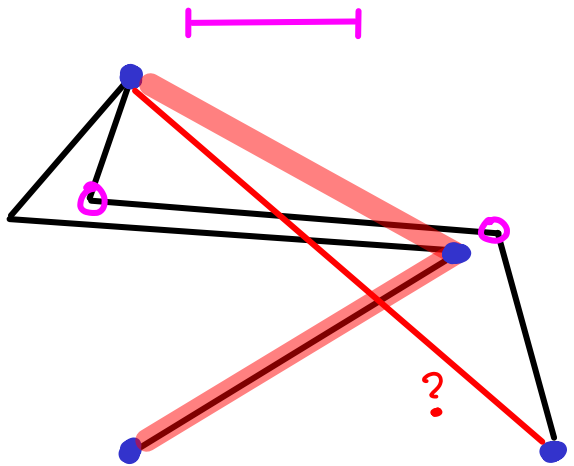


(ok maybe the threshold should be a bit greater)

do we always get a simple (non-crossing) chain?

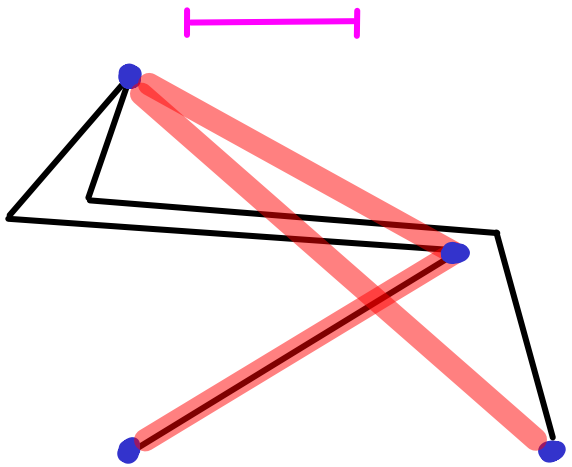


do we always get a simple (non-crossing) chain?



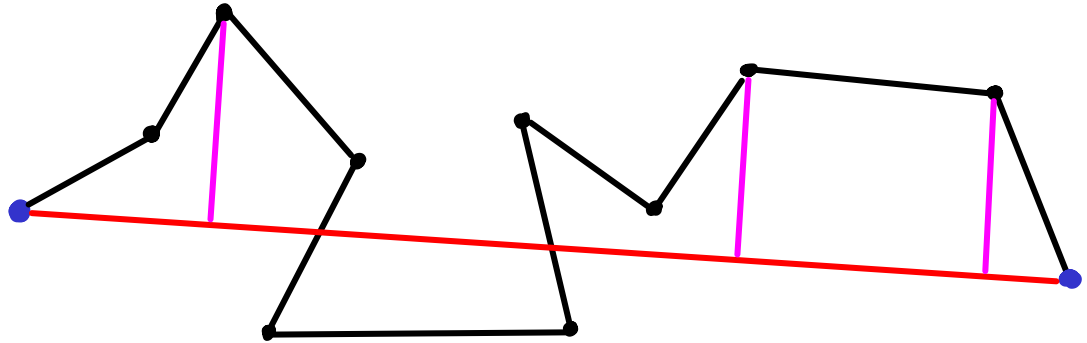
both vertices within tolerance

do we always get a simple (non-crossing) chain? NO

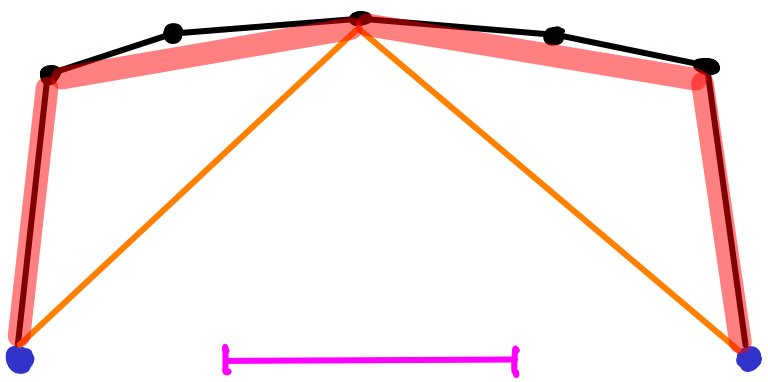


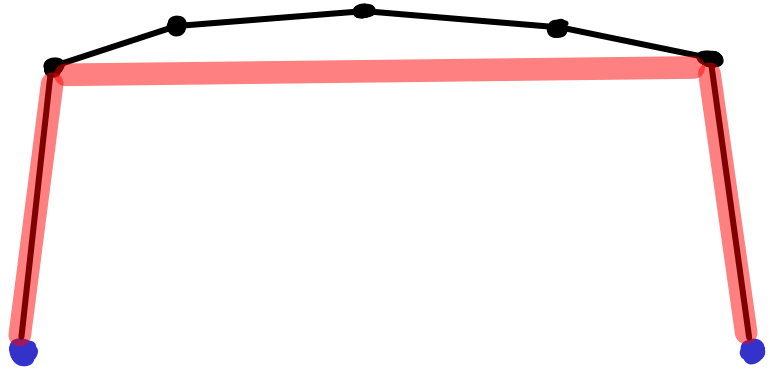
variants (possibly unexplored)

- require non-crossing output
- deal with multiple error violations
  - ↳ use median "offender"?
  - ↳ ⩾2 recursive calls?



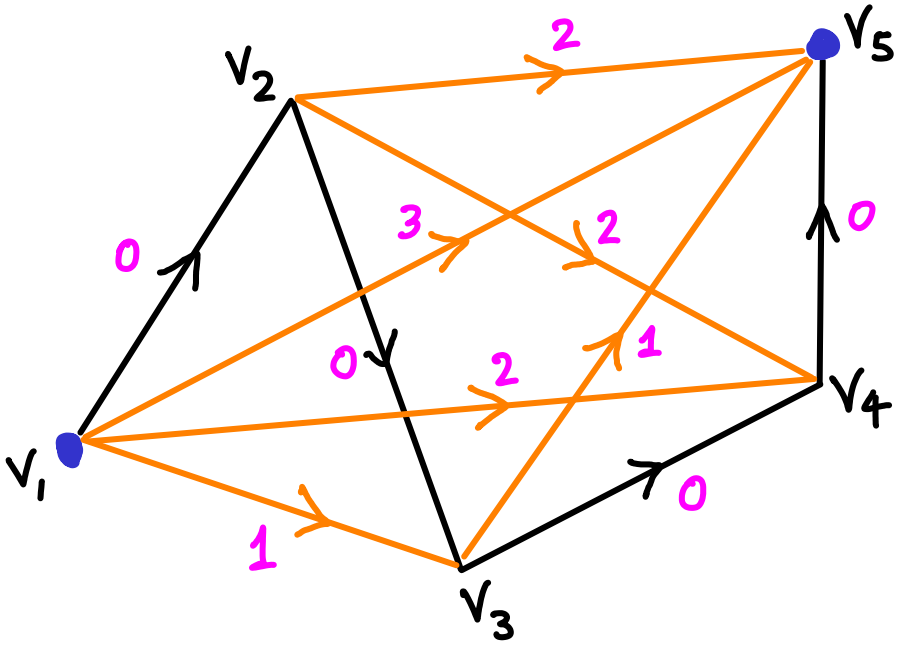
Does the algorithm minimize the number of segments used ?



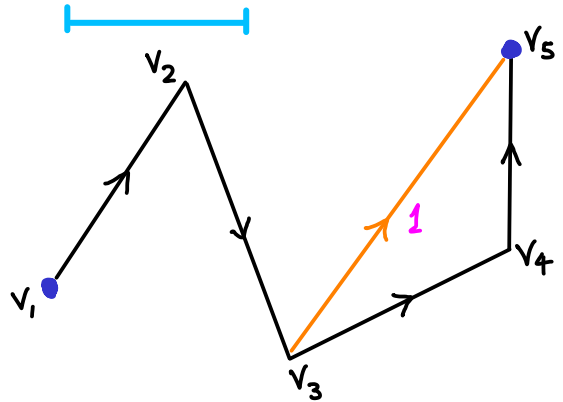


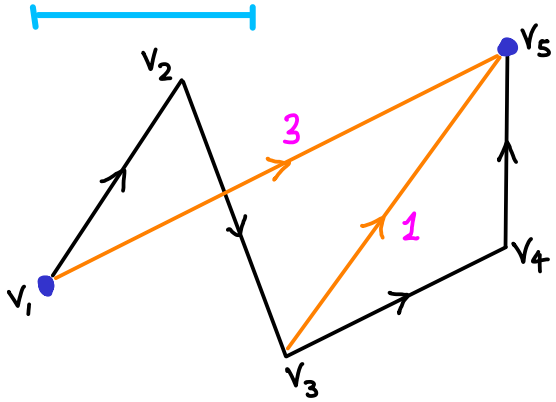
No. can be made arbitrarily worse

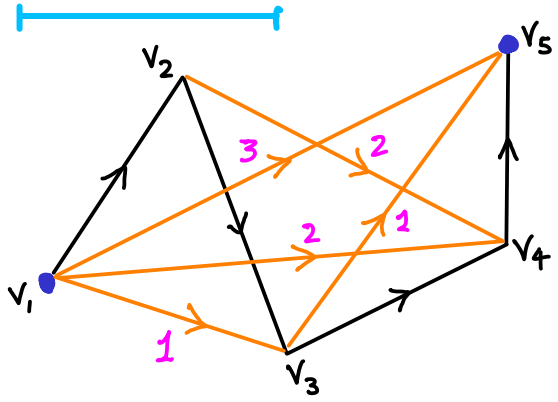
IRI-IMAI algorithm to minimize segments (given path & tolerance)



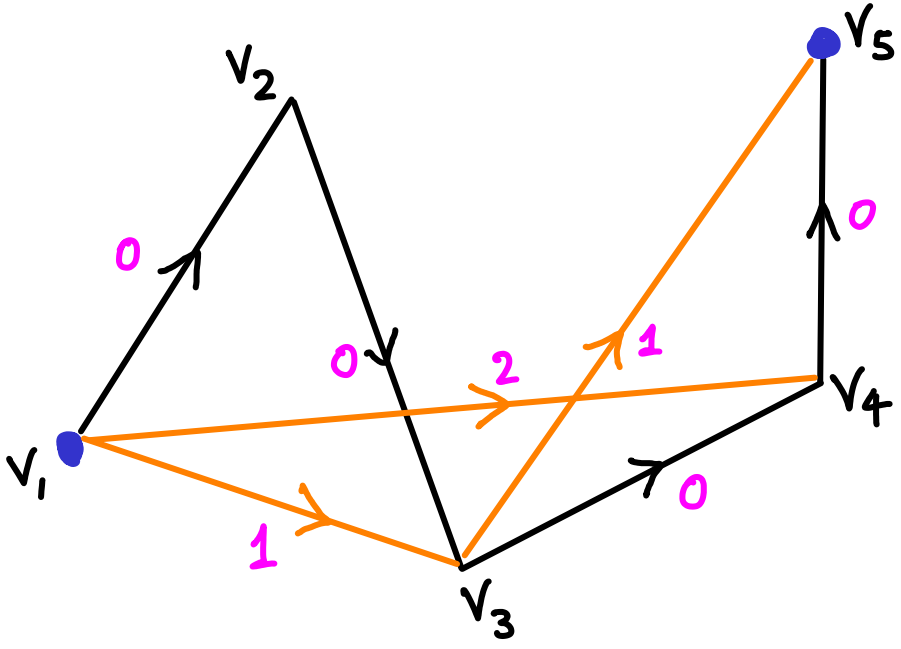
- form directed graph:
  - ↳ make $\overrightarrow{v_i v_j}$ iff $i<j$
- give a score to each edge:
  - ↳ # path edges skipped
- keep only edges satisfying tolerance







IRI-IMAI algorithm to minimize segments (given path & tolerance)



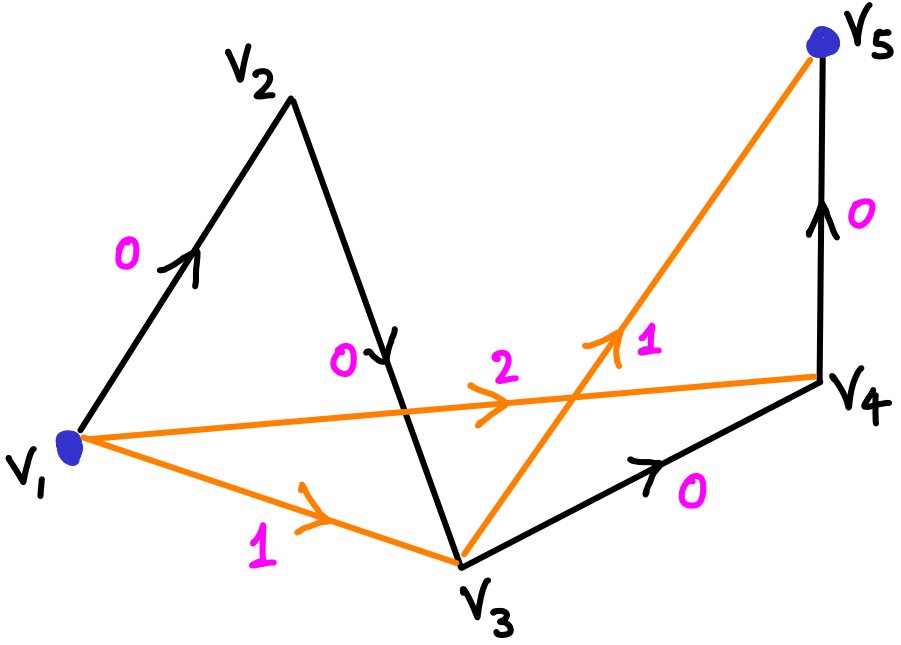
1 - form directed graph:
↳ make $\overrightarrow{v_i v_j}$ iff $i<j$

2 - give a score to each edge:
↳ # path edges skipped

3 - keep only edges satisfying tolerance

$\Theta(v^2)$ edges $\begin{cases} 1 & \Theta(v^2) \\ 2 & >> \\ 3 & \Theta(v^2)\cdot O(v) \end{cases}$

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

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

OR

- simple (unweighted) BFS

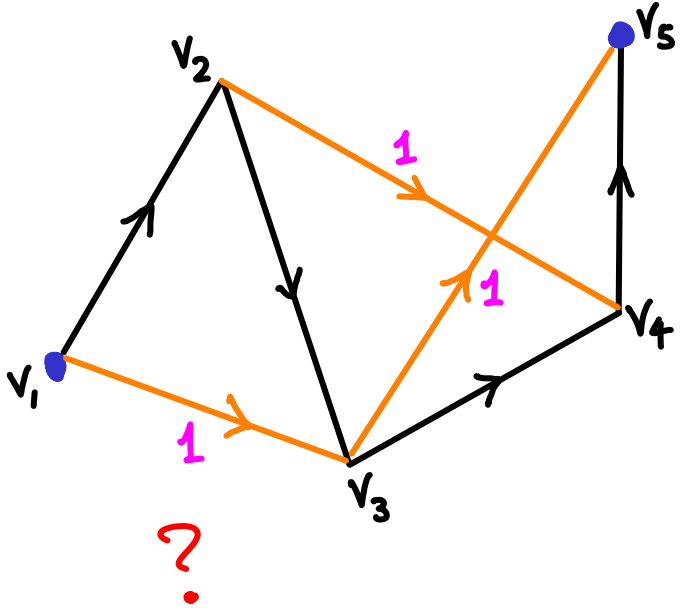
} $O(E)$

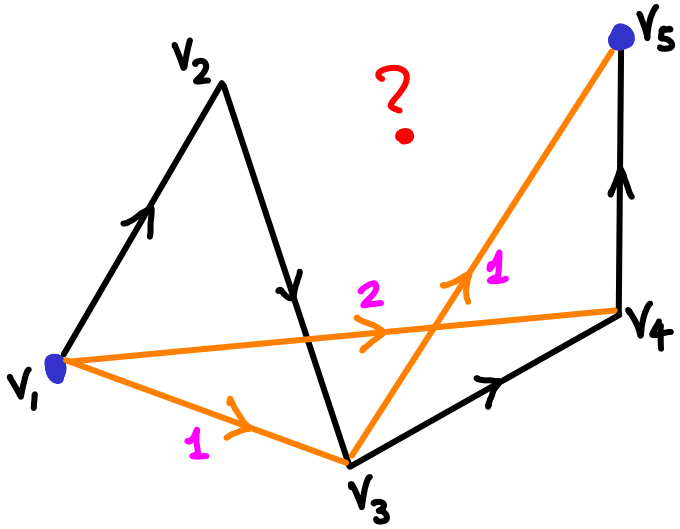
$\Theta(V^2)$ edges $\Theta(V^2)$ >> $\Theta(V^2) \cdot O(V)$

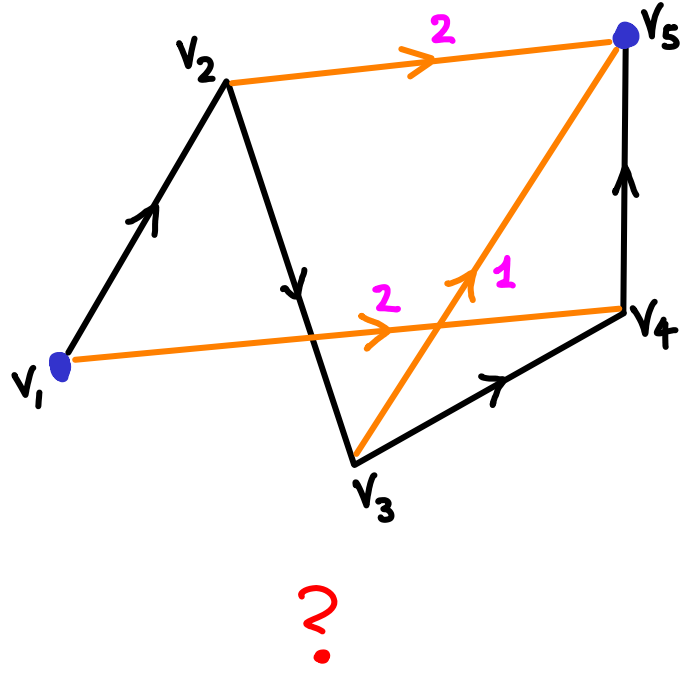
IRI-IMAI algorithm to minimize segments

Suppose we are willing to use up to k edges.

↳ then the goal is to minimize error







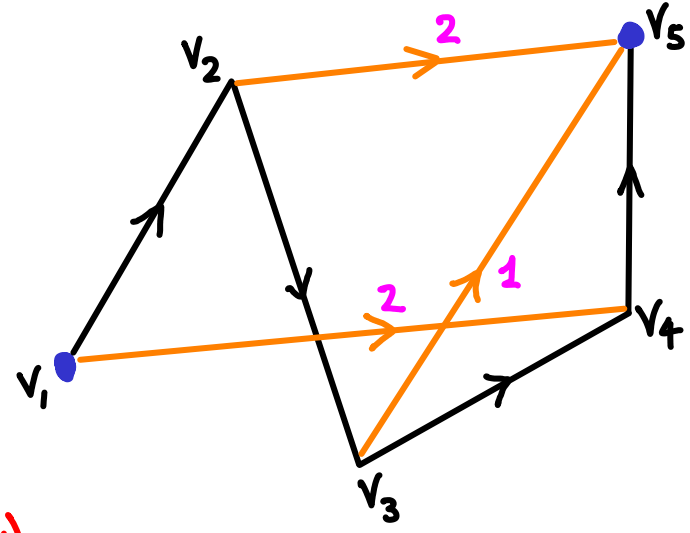
IRI-IMAI algorithm to minimize segments

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 = $O(V^3 \cdot \log V)$

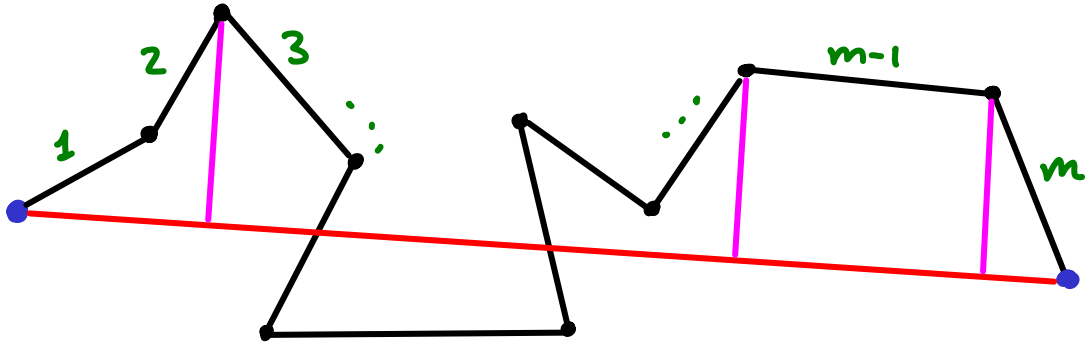


0 ← error → ∞

n ← #edges → 1

+ a dyn. prog. 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

↳ brute force : $\Theta(m)$ time

↳ under certain conditions (e.g. parallel strip distance)
& w/ preprocessing, we can do better

(and thus beat $O(V^3)$ overall)

PROJECT