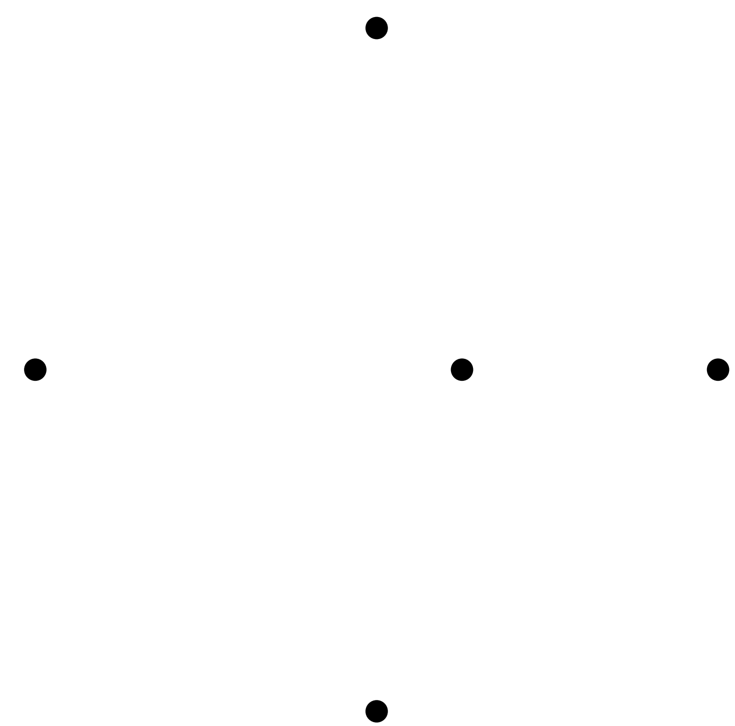
Delaunay Triangulation

CS60064: Computational Geometry (2021-22 Spring)

March 2022

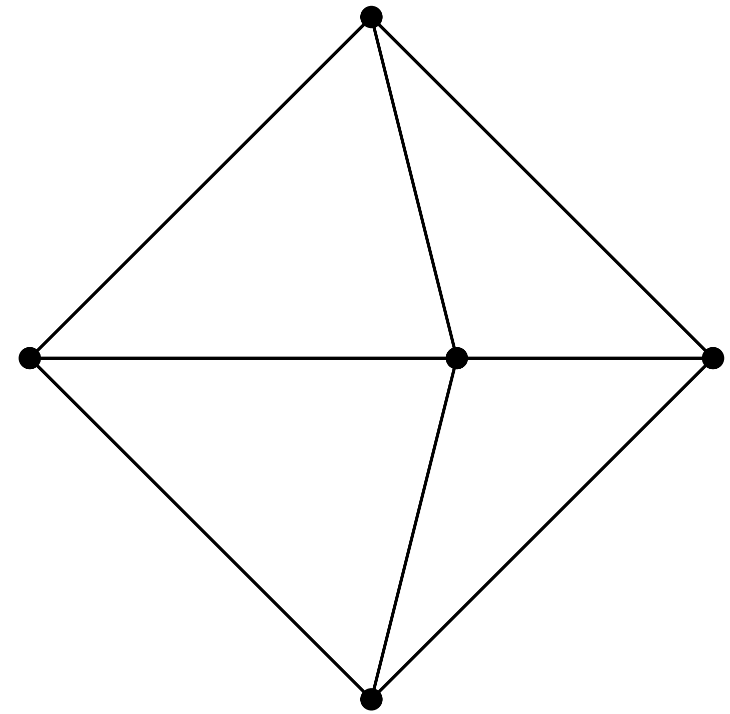
Partha Bhowmick (CSE, IIT Kharagpur)



Input $P = \{5 \text{ sites}\}$

Task:

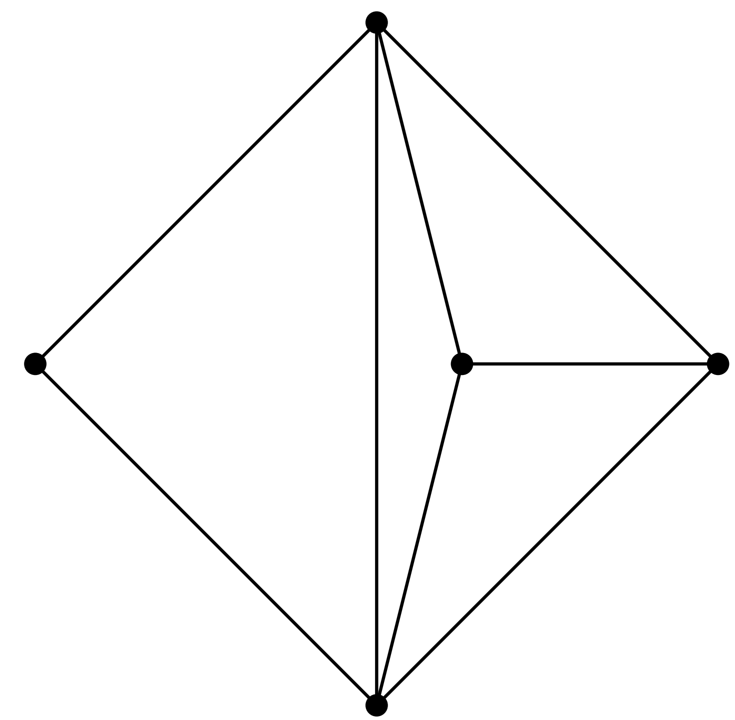
1. Join the sites of P by non-intersecting straight line segments so as to get a triangulation of P.



Output $\mathcal{T} = \{4 \text{ triangles}\}$
$A = \alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{12}$

Task:

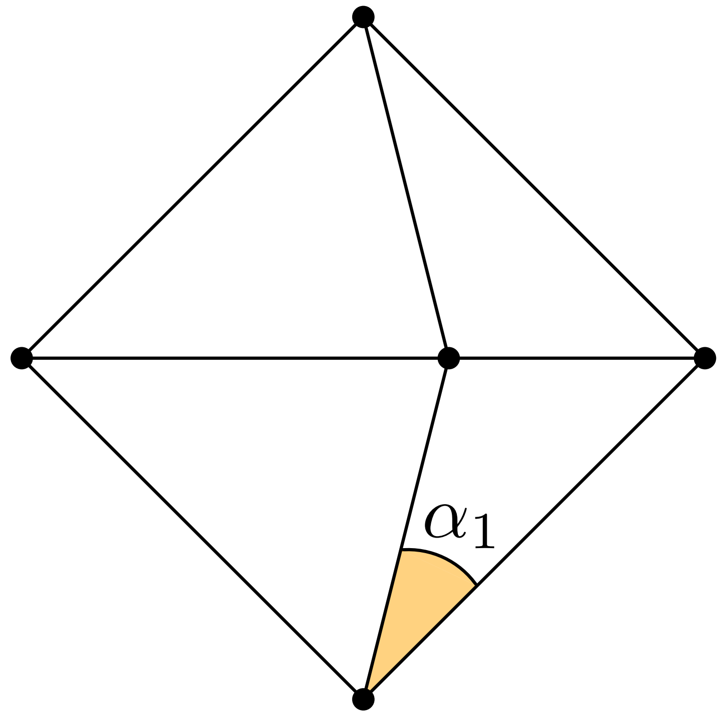
1. Join the sites of P by non-intersecting straight line segments so as to get a triangulation of P.



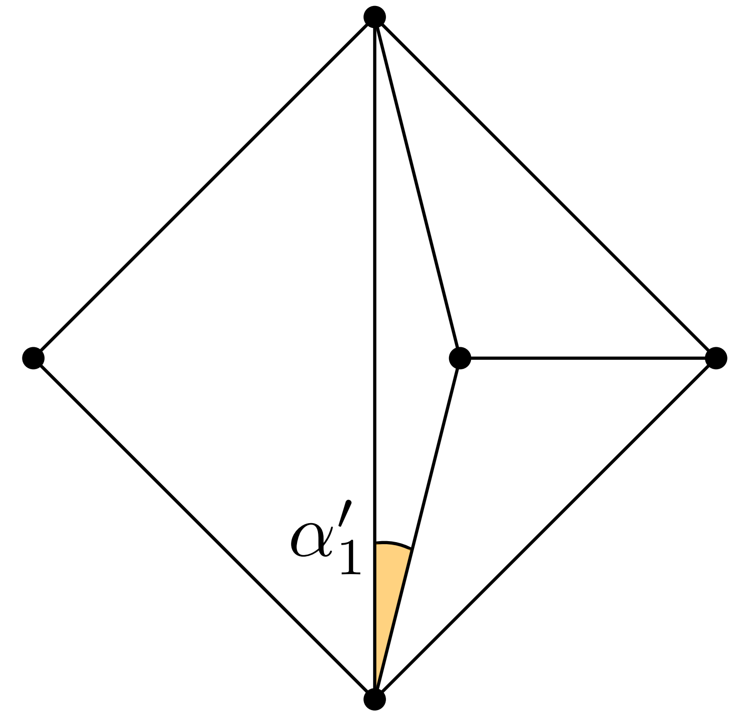
Output $\mathcal{T}' = \{\text{4 triangles}\}$
$A' = \alpha'_1 \leq \alpha'_2 \leq \cdots \leq \alpha'_{12}$

Task:

1. Join the sites of P by non-intersecting straight line segments so as to get a triangulation of P.



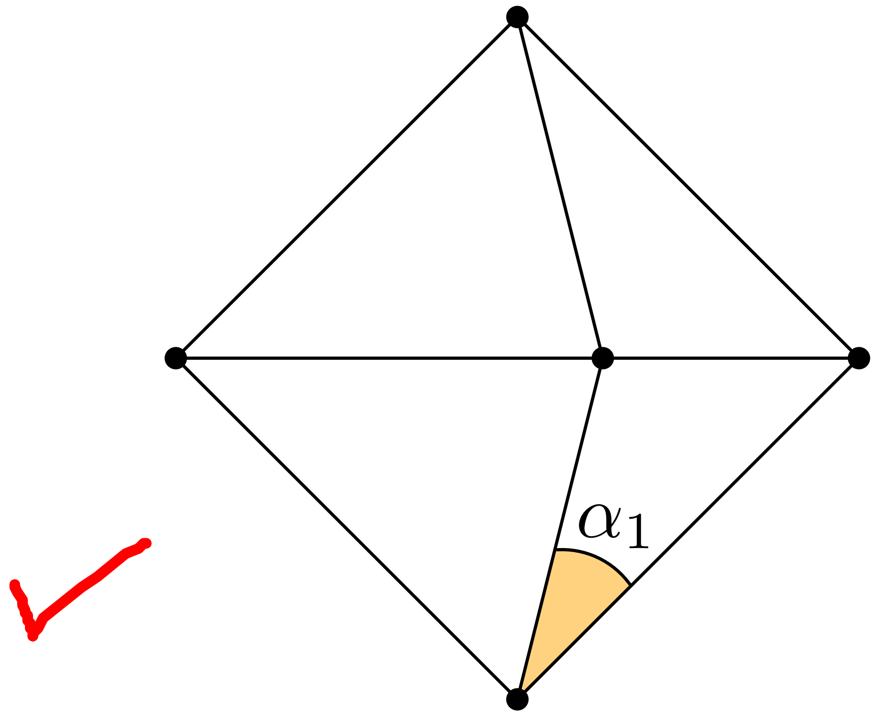
Output $\mathcal{T} = \{$4 triangles$\}$
$A = \alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{12}$



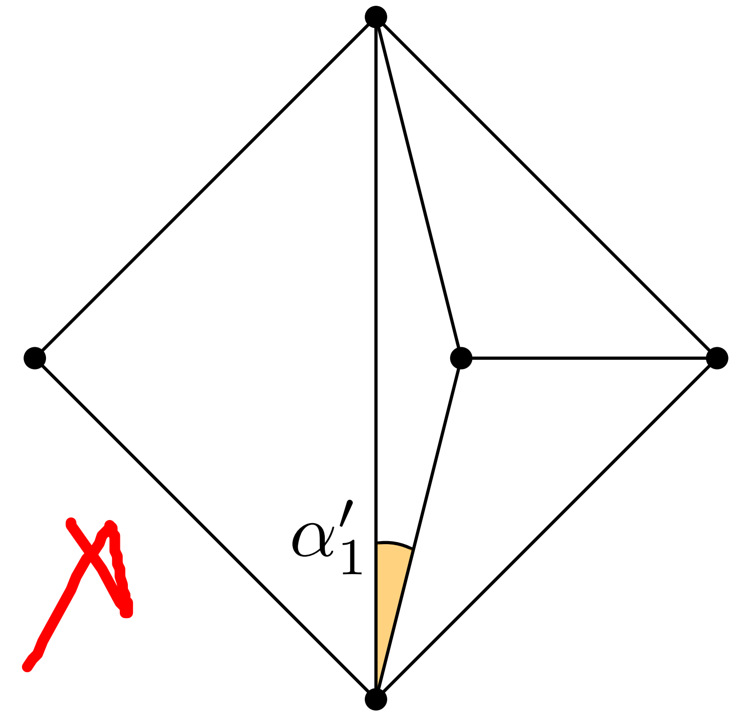
Output $\mathcal{T}' = \{$4 triangles$\}$
$A' = \alpha'_1 \leq \alpha'_2 \leq \cdots \leq \alpha'_{12}$

Task:

1. Join the sites of P by non-intersecting straight line segments so as to get a triangulation of P.



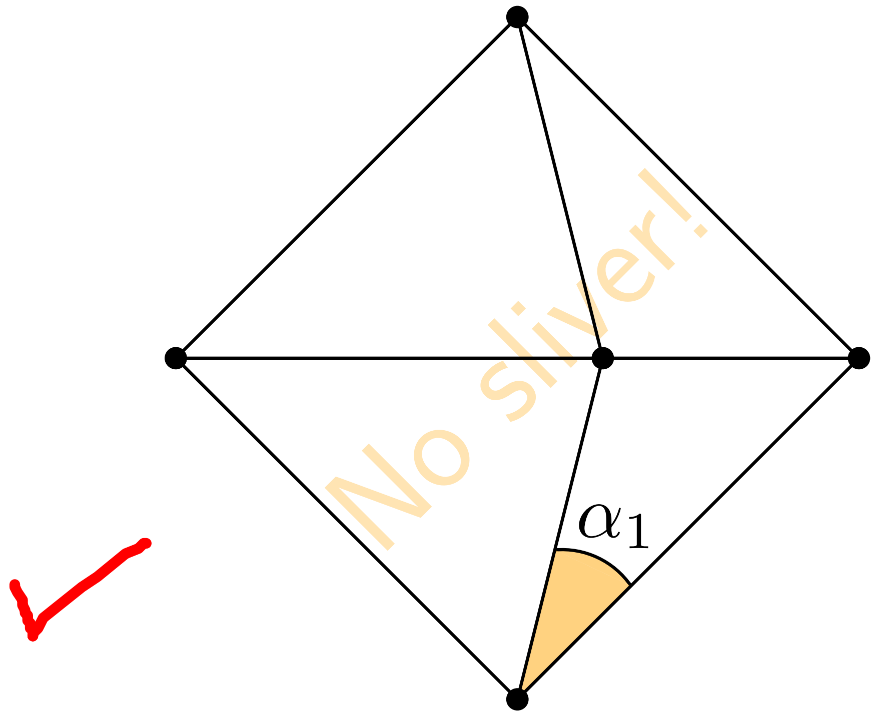
Output $\mathcal{T} = \{4 \text{ triangles}\}$
$A = \alpha_1 \leq \alpha_2 \leq \cdots \leq \alpha_{12}$

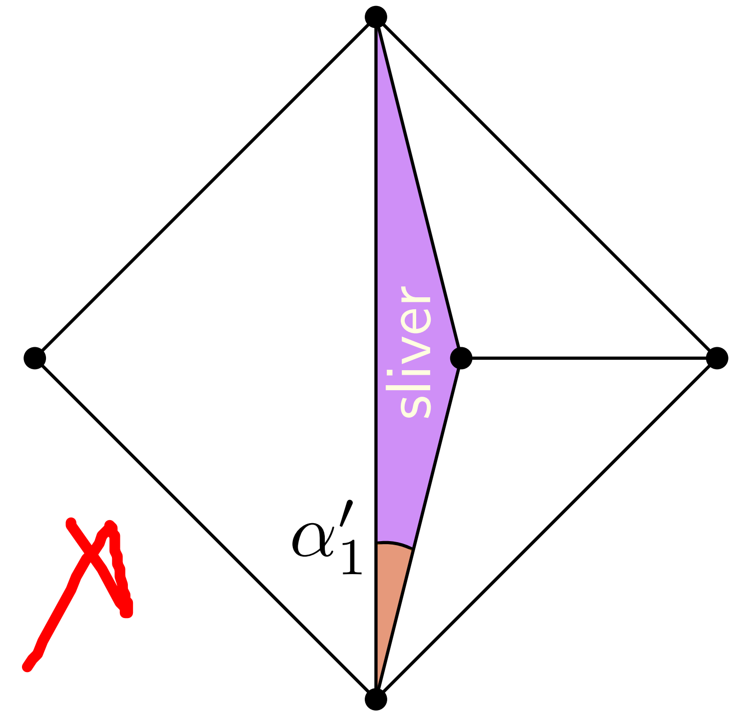


Output $\mathcal{T}' = \{4 \text{ triangles}\}$
$A' = \alpha'_1 \leq \alpha'_2 \leq \cdots \leq \alpha'_{12}$

Task:

1. Join the sites of P by non-intersecting straight line segments so as to get a triangulation of P.
2. Minimum angle is maximized.



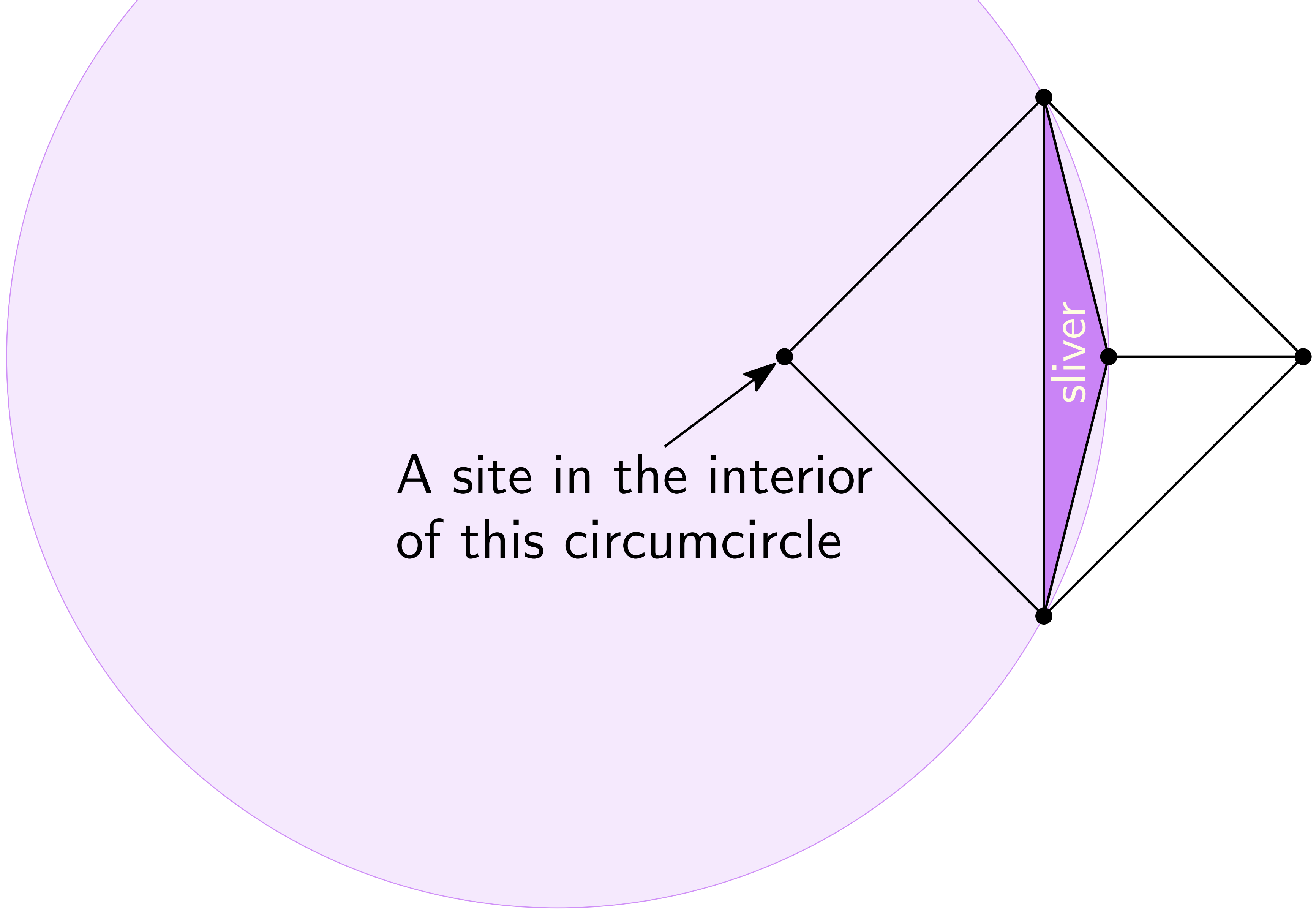


Output $\mathcal{T}$ = {4 triangles}
$A = \alpha_1 \le \alpha_2 \le \cdots \le \alpha_{12}$

Output $\mathcal{T}'$ = {4 triangles}
$A' = \alpha'_1 \le \alpha'_2 \le \cdots \le \alpha'_{12}$

Task:

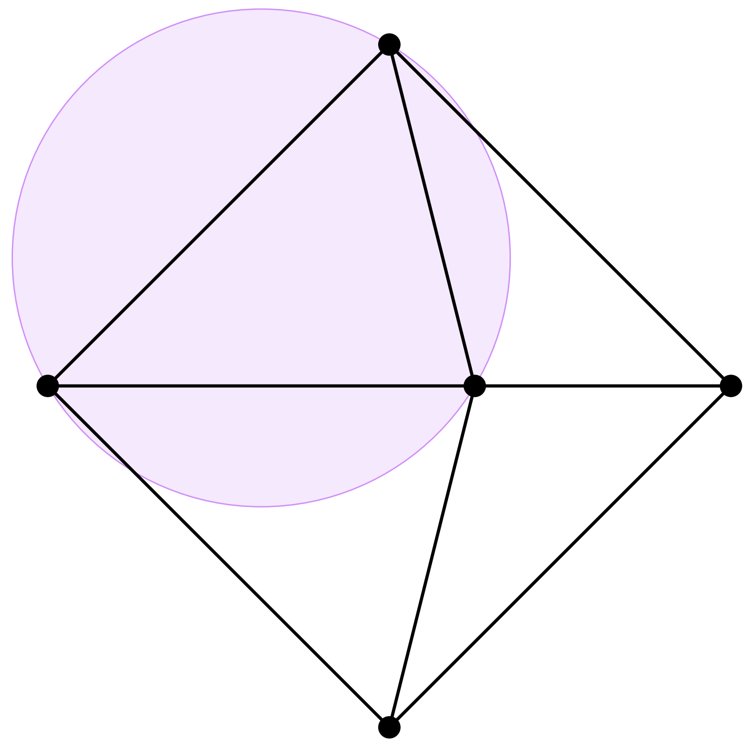
1. Join the sites of P by non-intersecting straight line segments so as to get a triangulation of P.
2. Minimum angle is maximized. $\Longrightarrow$ (possibly) no sliver!
   (more precisely, all circumcircles are empty — will see later) $\Longrightarrow$ **Delaunay**

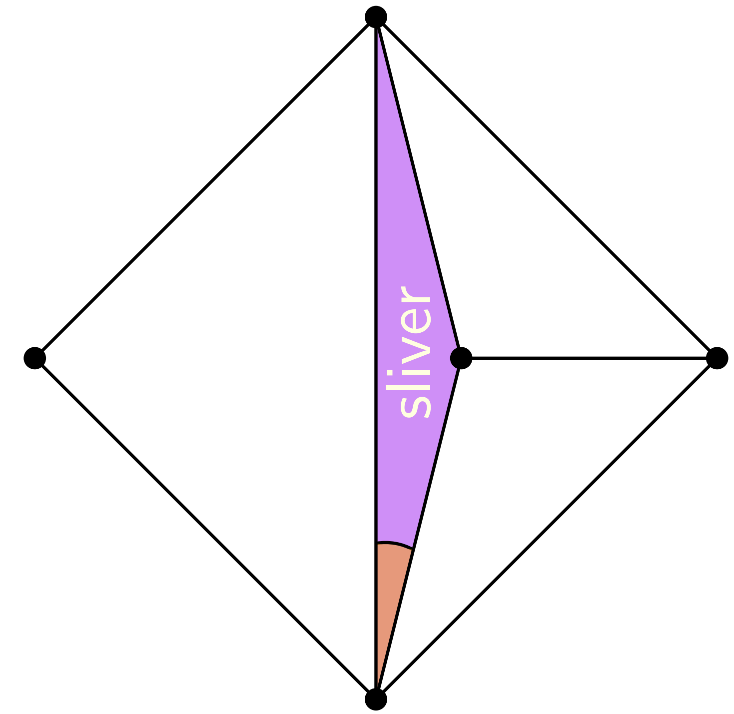


Circumcircle of sliver is not empty

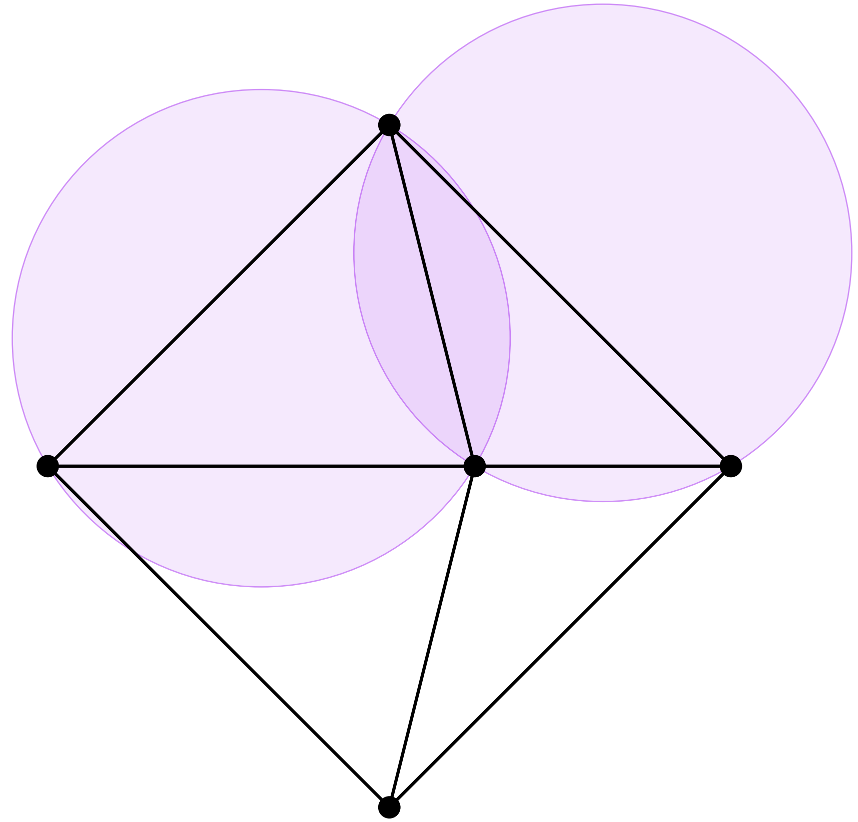
Task:

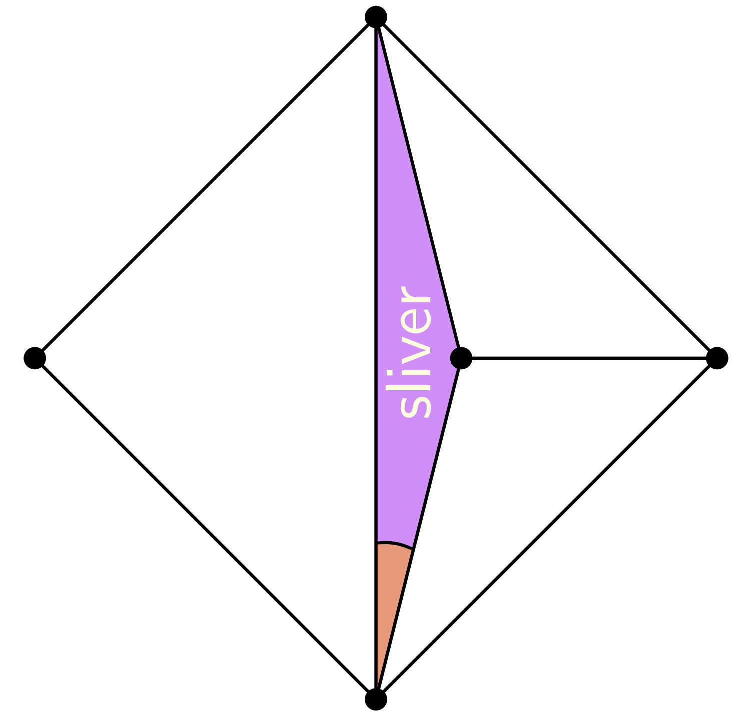
1. Join the sites of P by non-intersecting straight line segments so as to get a triangulation of P.
2. Minimum angle is maximized. $\Longrightarrow$ No sliver!



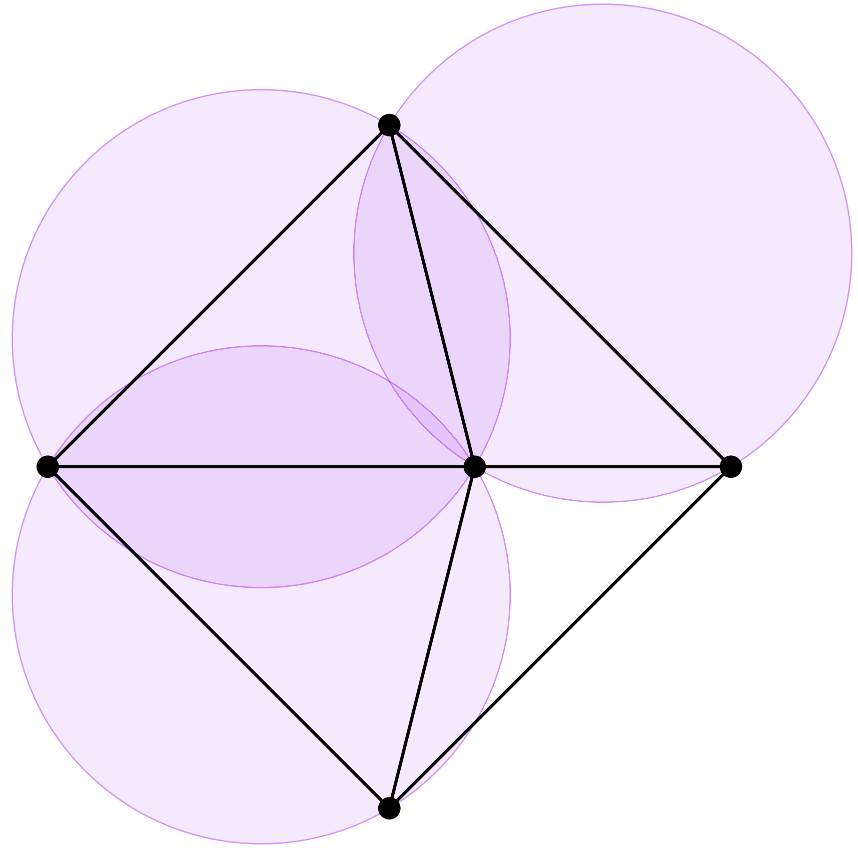


1st circumcircle is empty

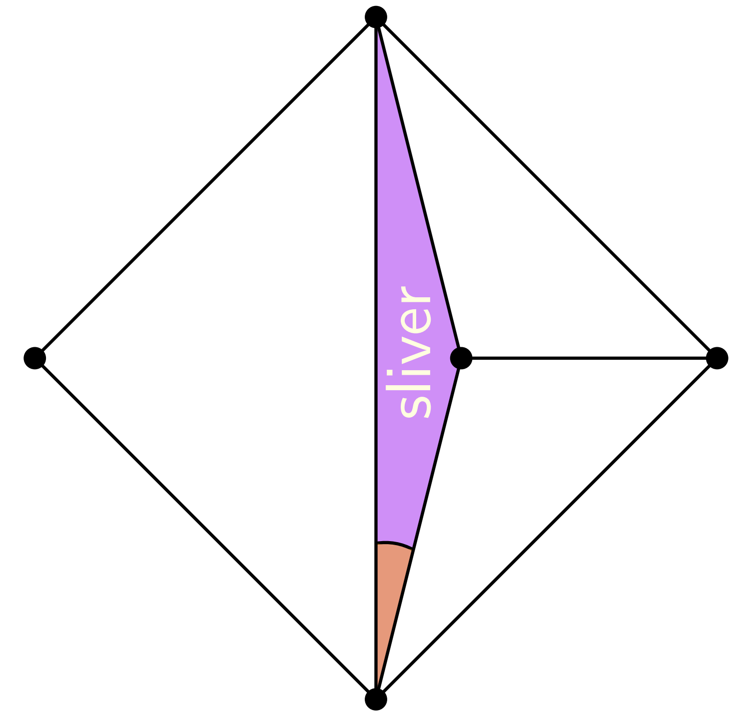


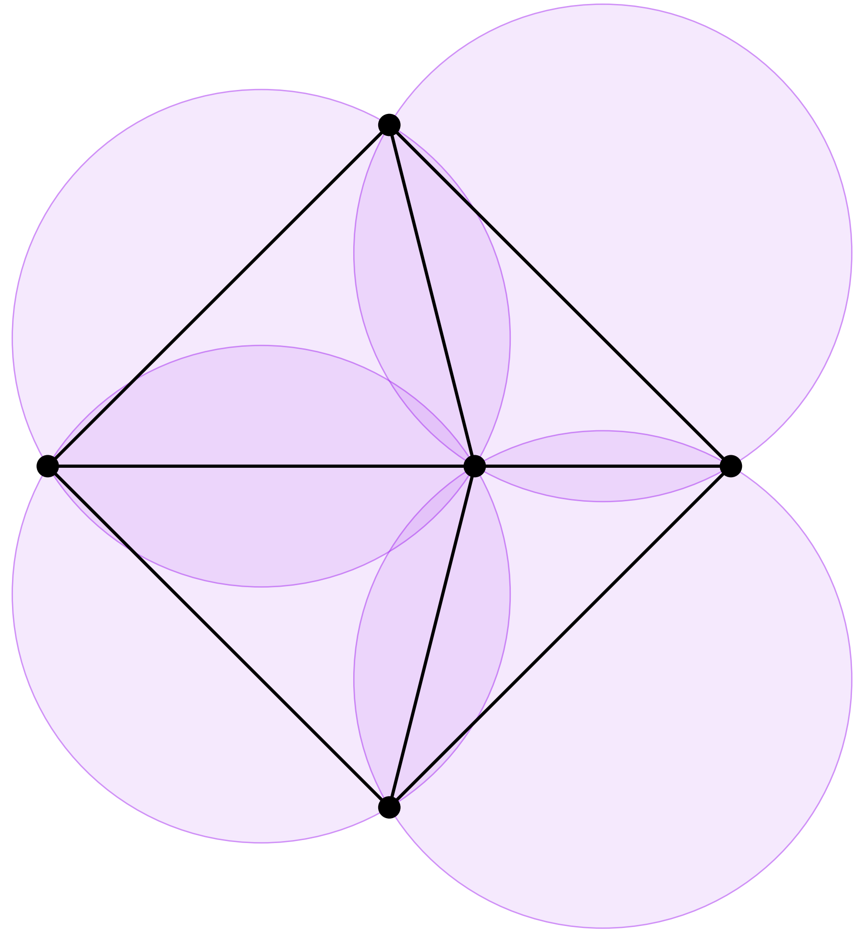


2nd circumcircle is empty

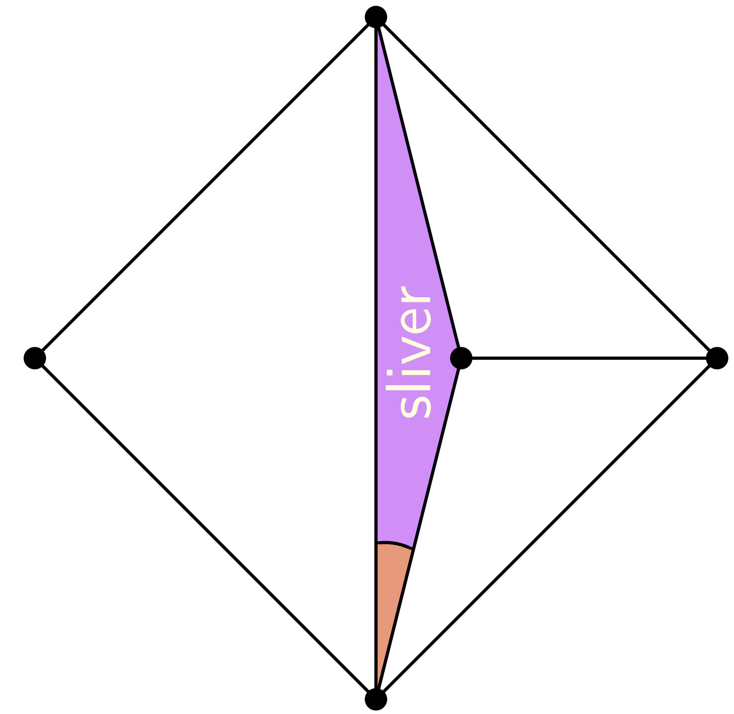


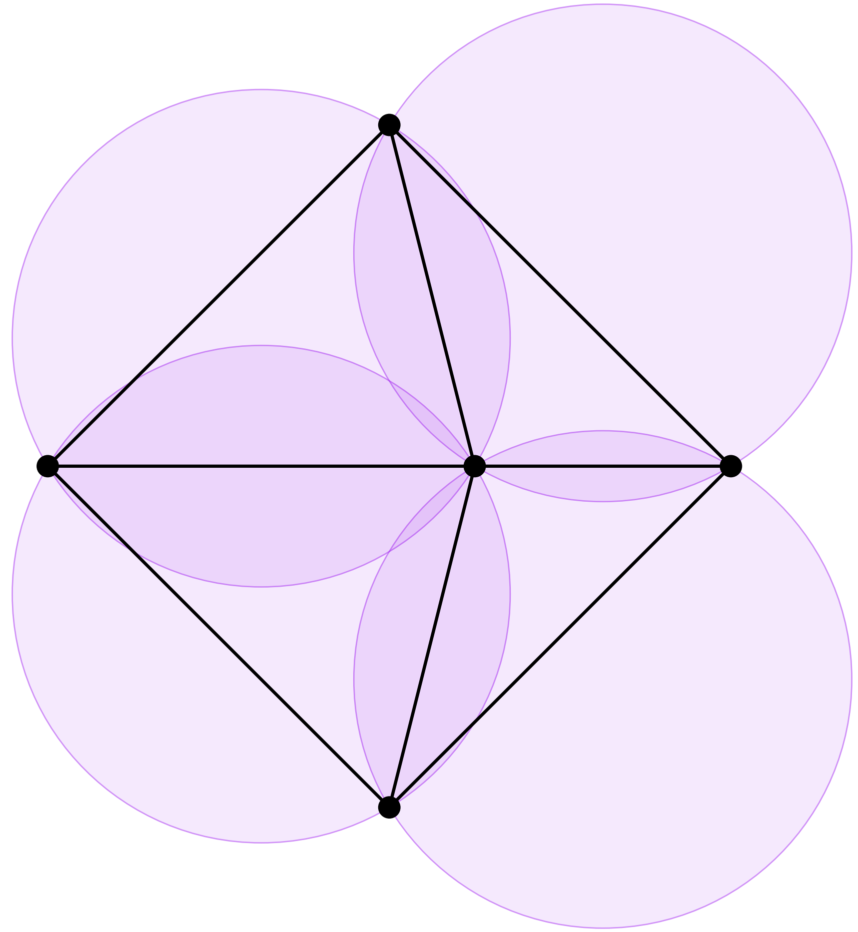
3rd circumcircle is empty



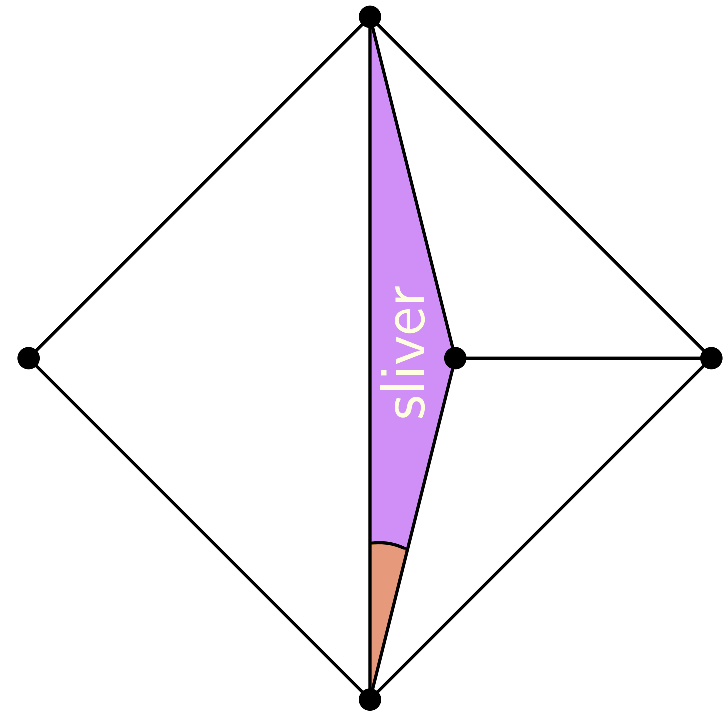


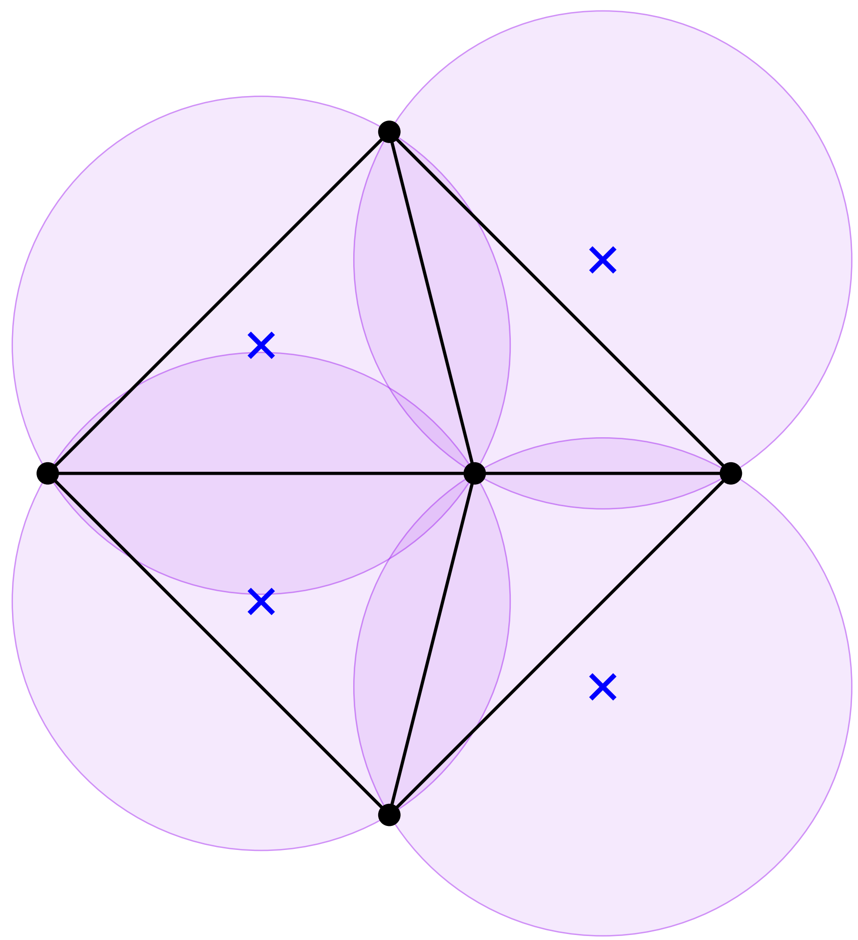
4th circumcircle is empty



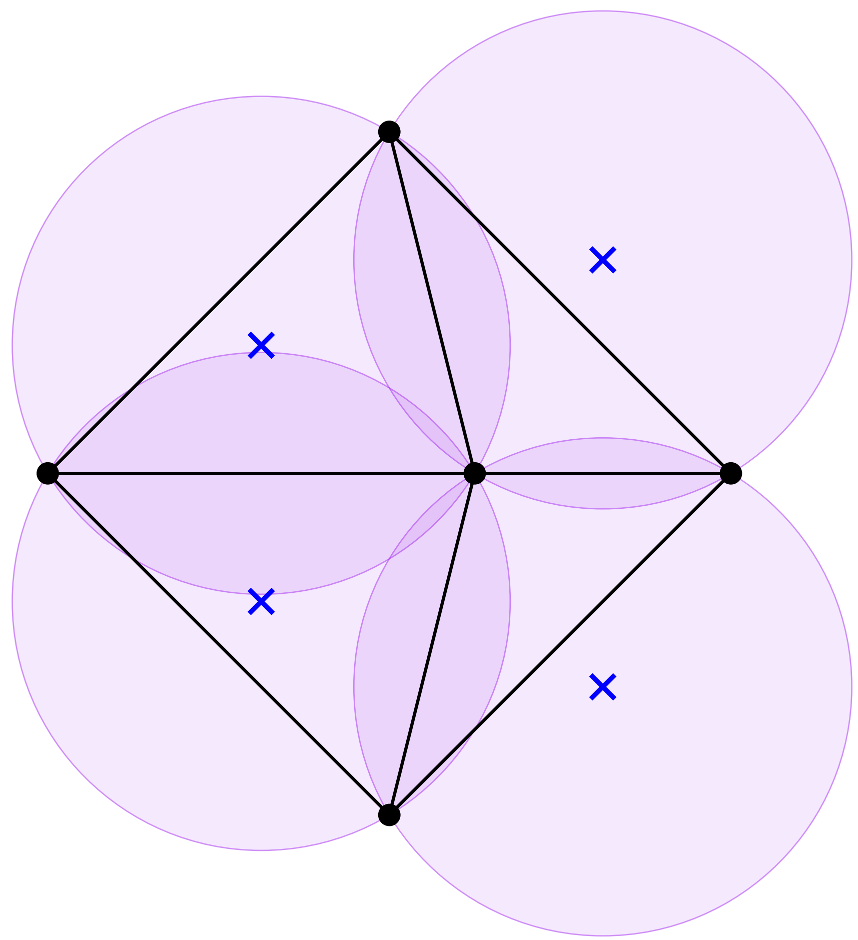


So, all circumcircles are empty

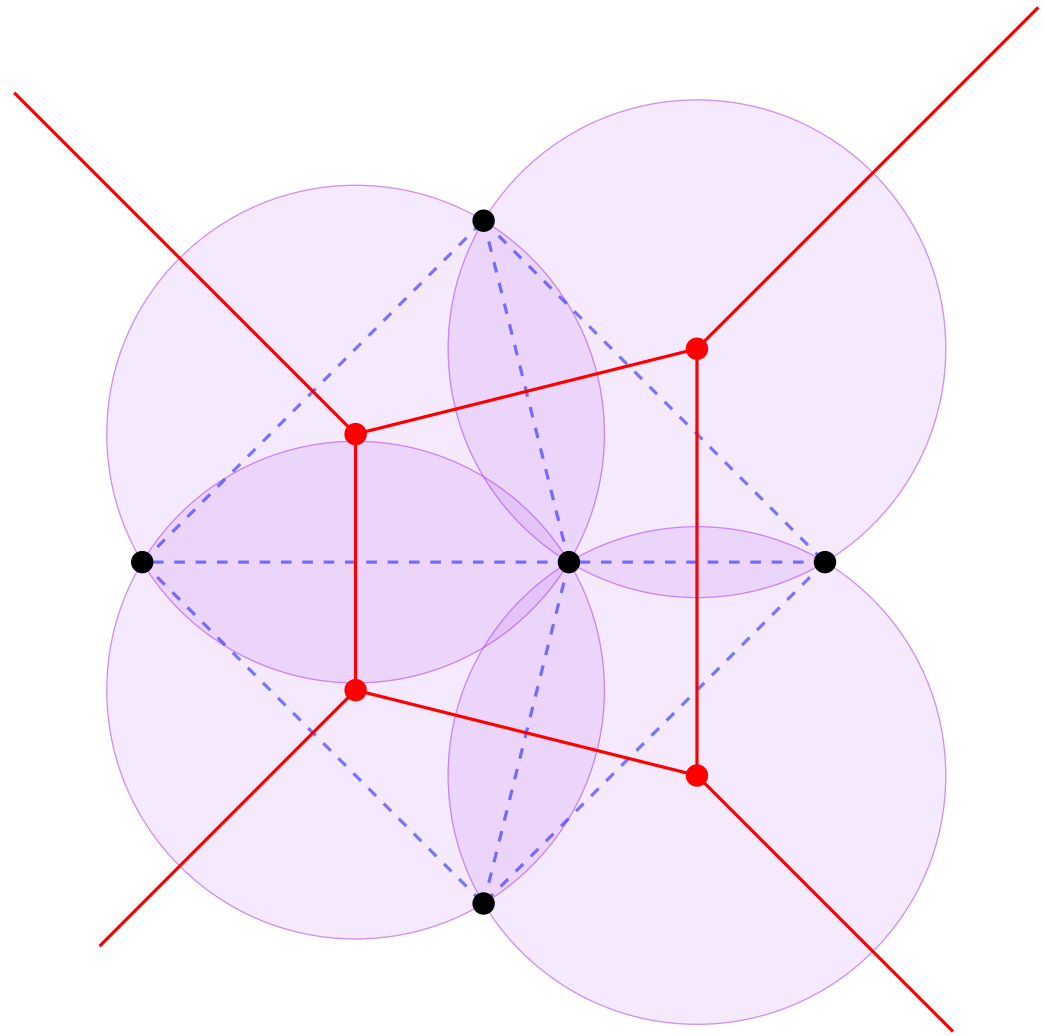




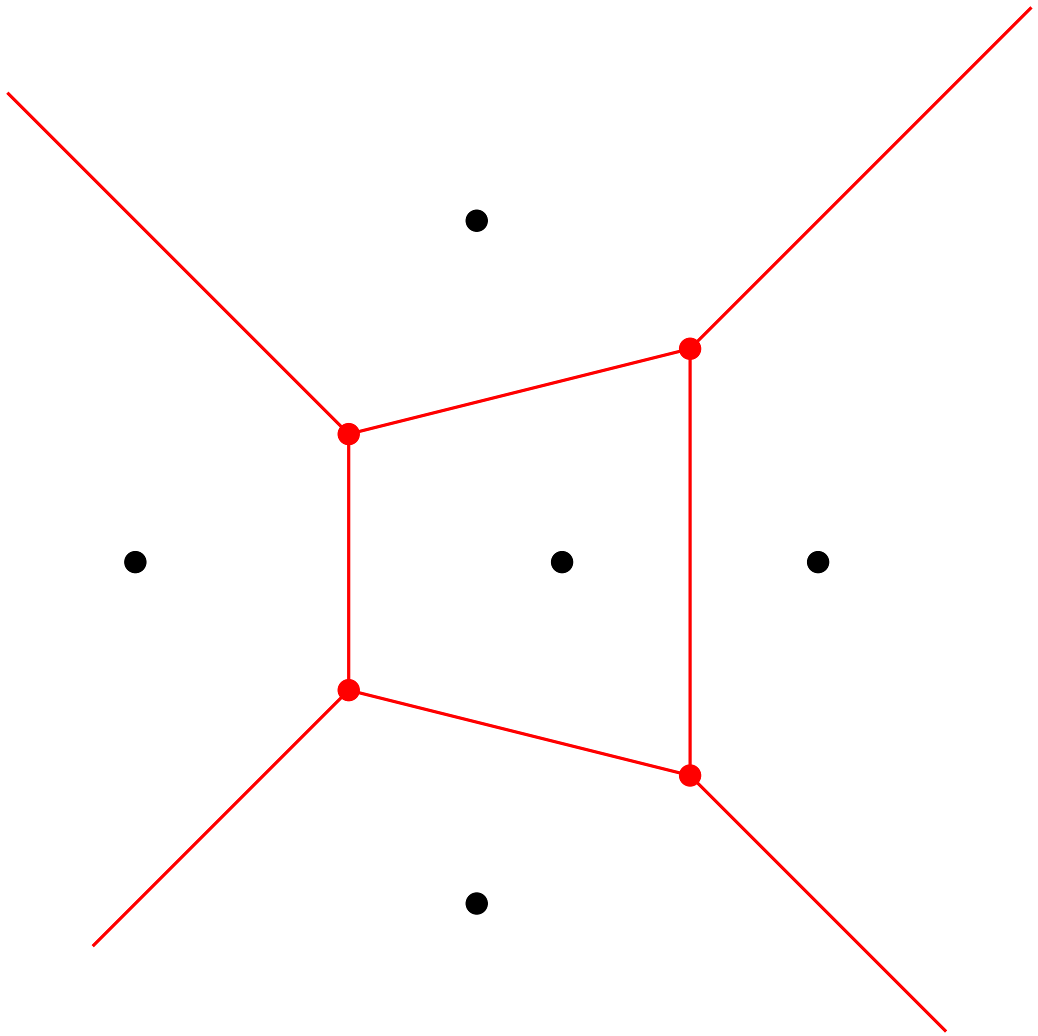
1. Mark the centers of the empty circumcircles



1. Notice the centers of the empty circumcircles.
2. What are these centers?

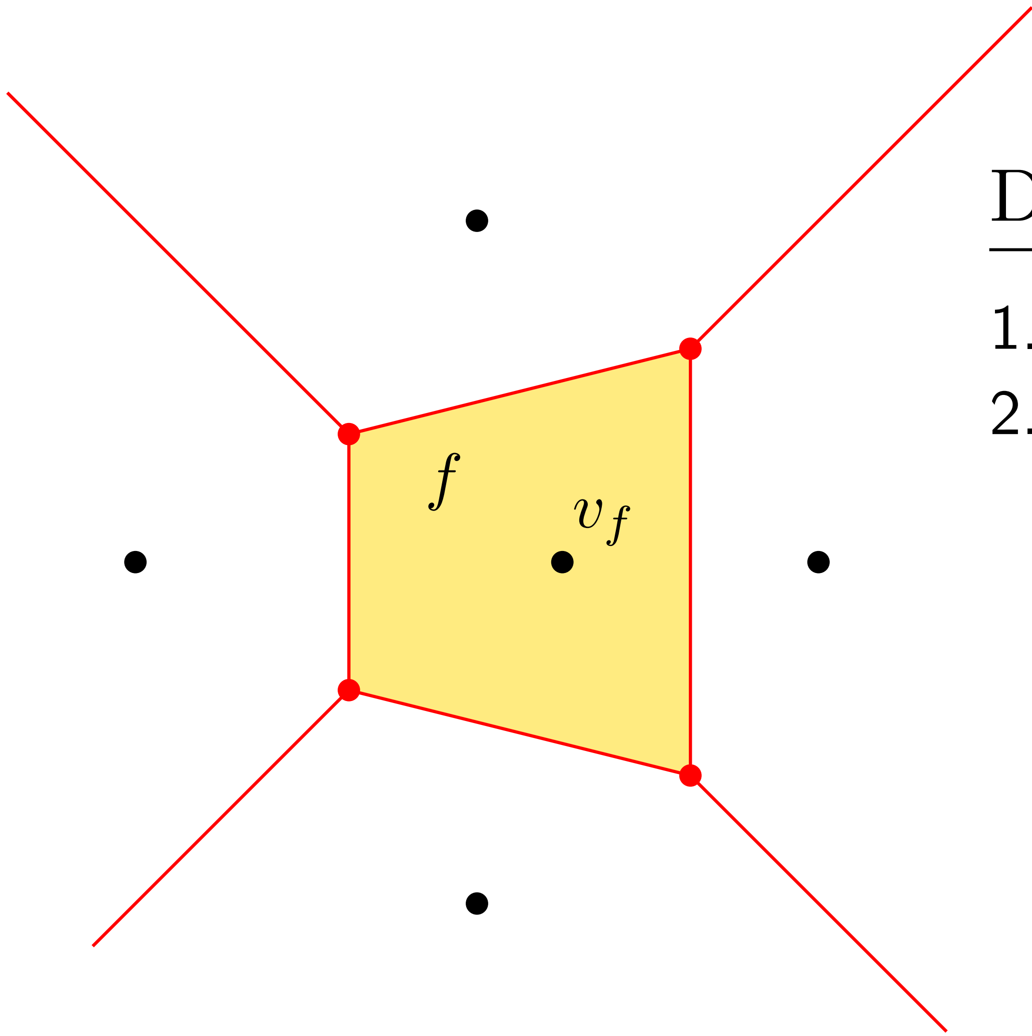


1. Notice the centers of the empty circumcircles.
2. What are these centers? They are the vertices in $\mathrm{VD}(P)$!



So, a reverse thinking gives $\mathrm{DT}(P)$:

1. Construct $\mathrm{VD}(P)$.



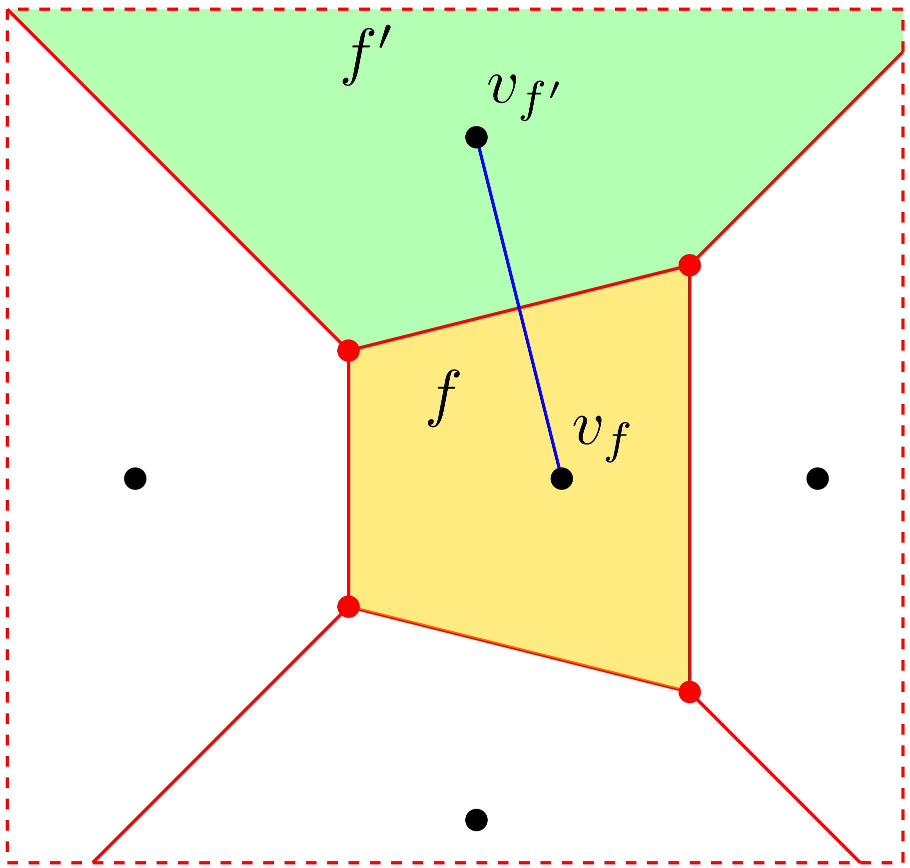
$\mathrm{DT}(P) =$ **dual of** $\mathrm{VD}(P)$

1. Imagine both $\mathrm{VD}(P)$ and $\mathrm{DT}(P)$ as planar graphs.
2. **for** every face $f \in \mathrm{VD}(P)$

   add a unique vertex v_f in $\mathrm{DT}(P)$

So, a reverse thinking gives $\mathrm{DT}(P)$:

1. Construct $\mathrm{VD}(P)$.
2. Construct the dual of $\mathrm{VD}(P)$ to get $\mathrm{DT}(P)$.

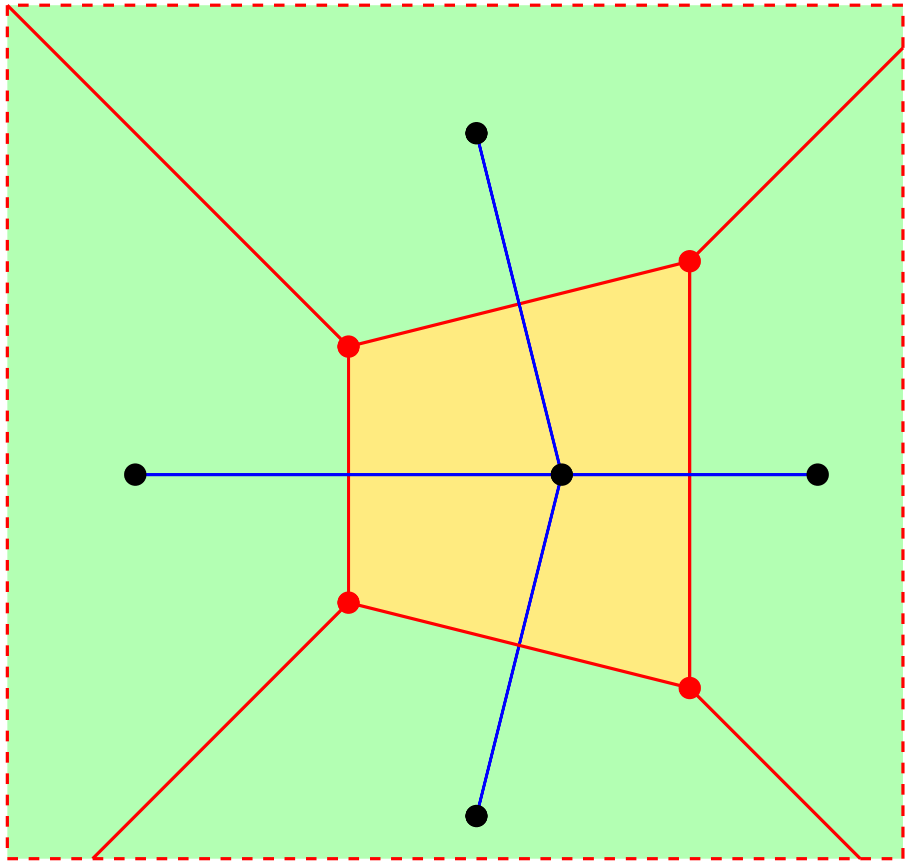


$\mathrm{DT}(P)$ = **dual of** $\mathrm{VD}(P)$

1. Imagine both $\mathrm{VD}(P)$ and $\mathrm{DT}(P)$ as planar graphs.
2. **for** every face $f \in \mathrm{VD}(P)$
   add a unique vertex v_f in $\mathrm{DT}(P)$
3. **for** every face $f \in \mathrm{VD}(P)$
   for every face f' adjacent to f
   add the edge $(v_f, v_{f'})$ in $\mathrm{DT}(P)$

So, a reverse thinking gives $\mathrm{DT}(P)$:

1. Construct $\mathrm{VD}(P)$.
2. Construct the dual of $\mathrm{VD}(P)$ to get $\mathrm{DT}(P)$.

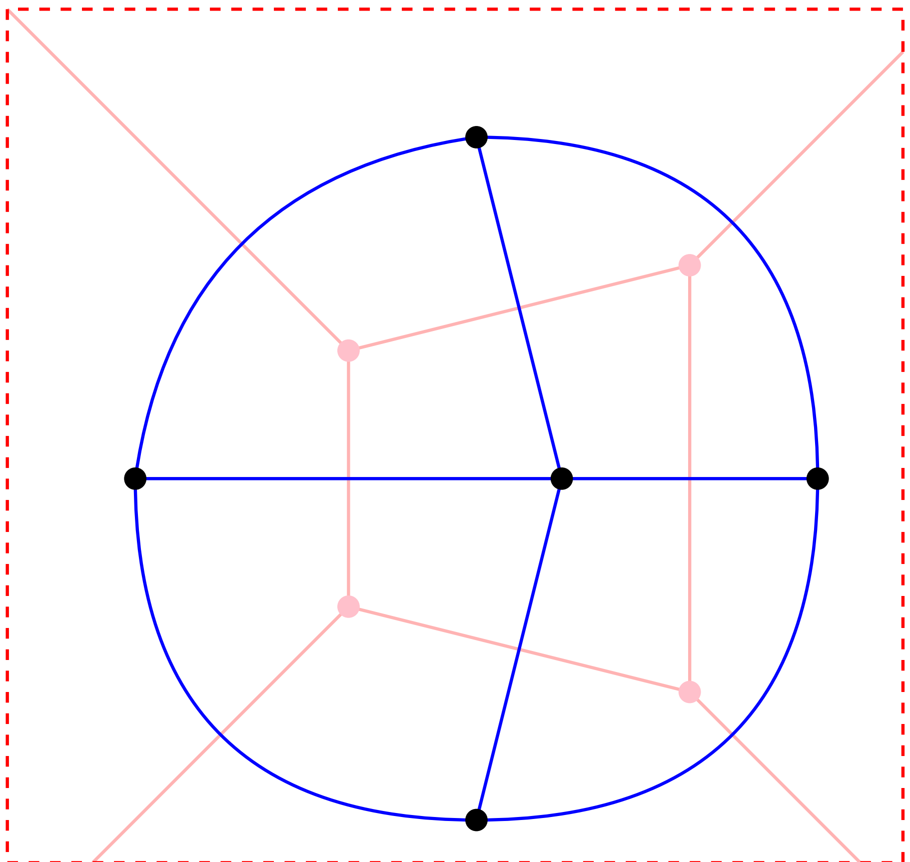


$\mathrm{DT}(P)$ = **dual of** $\mathrm{VD}(P)$

1. Imagine both $\mathrm{VD}(P)$ and $\mathrm{DT}(P)$ as planar graphs.
2. **for** every face $f \in \mathrm{VD}(P)$
   add a unique vertex v_f in $\mathrm{DT}(P)$
3. **for** every face $f \in \mathrm{VD}(P)$
   for every face f' adjacent to f
      add the edge $(v_f, v_{f'})$ in $\mathrm{DT}(P)$

So, a reverse thinking gives $\mathrm{DT}(P)$:

1. Construct $\mathrm{VD}(P)$.
2. Construct the dual of $\mathrm{VD}(P)$ to get $\mathrm{DT}(P)$.

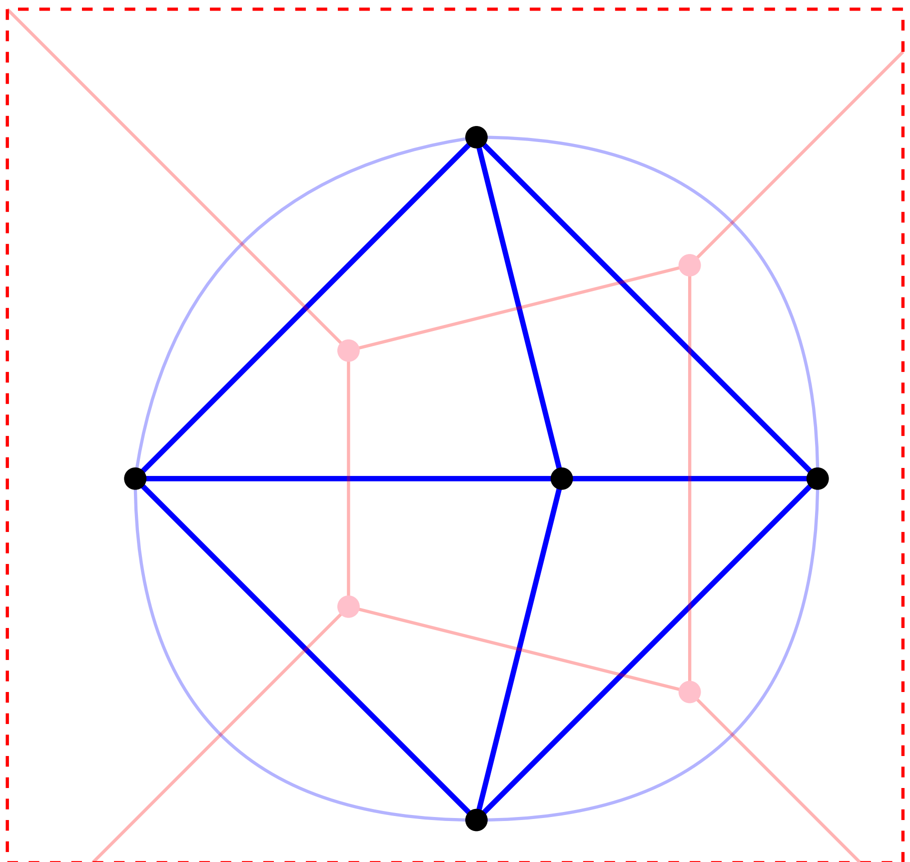


$\underline{\mathrm{DT}(P) = \textbf{dual of } \mathrm{VD}(P)}$

1. Imagine both $\mathrm{VD}(P)$ and $\mathrm{DT}(P)$ as planar graphs.
2. **for** every face $f \in \mathrm{VD}(P)$
   add a unique vertex v_f in $\mathrm{DT}(P)$
3. **for** every face $f \in \mathrm{VD}(P)$
   for every face f' adjacent to f
   add the edge $(v_f, v_{f'})$ in $\mathrm{DT}(P)$

So, a reverse thinking gives $\mathrm{DT}(P)$:

1. Construct $\mathrm{VD}(P)$.
2. Construct the dual of $\mathrm{VD}(P)$ to get $\mathrm{DT}(P)$.

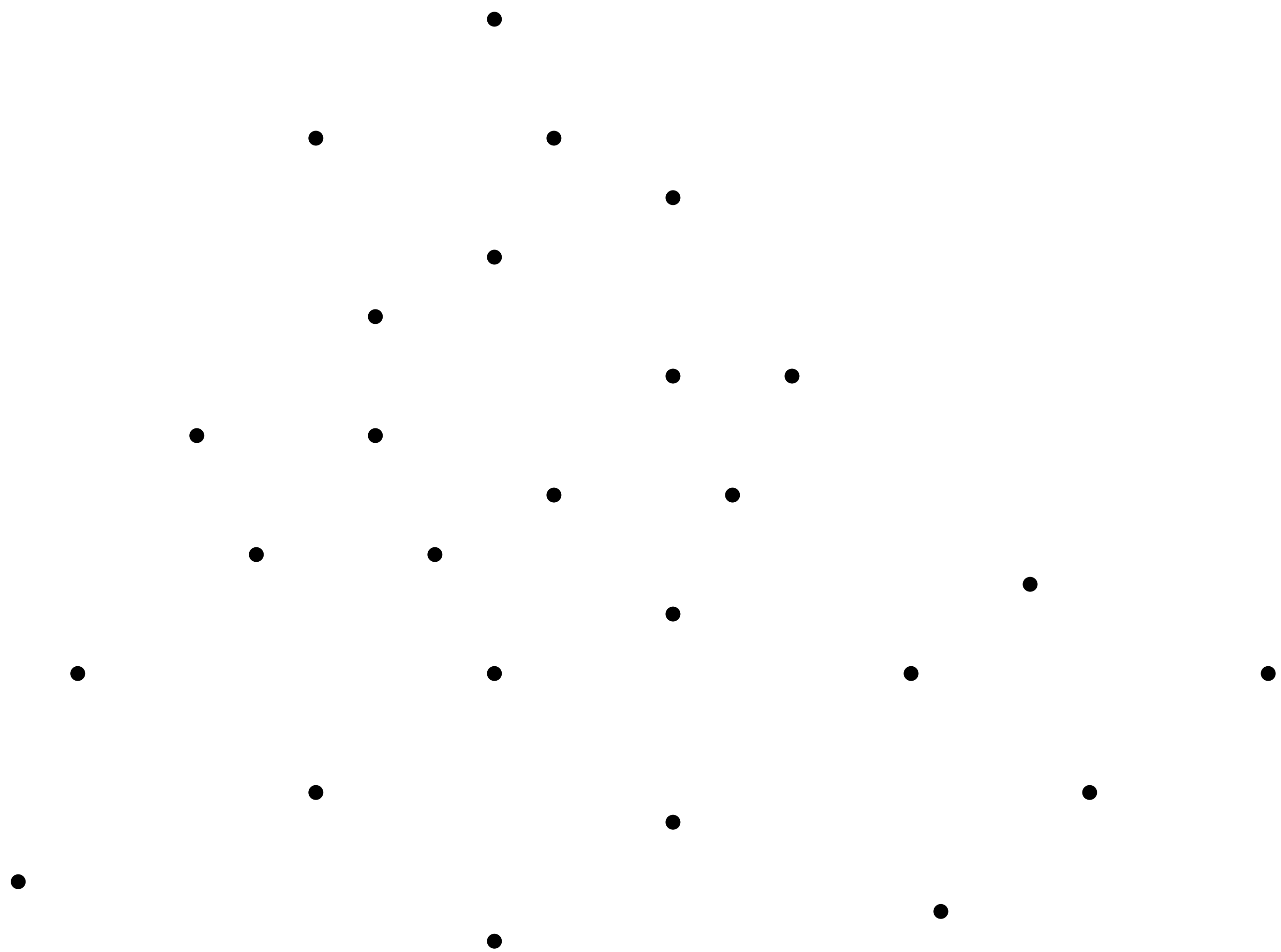


$\underline{\mathrm{DT}(P) = \textbf{dual of } \mathrm{VD}(P)}$

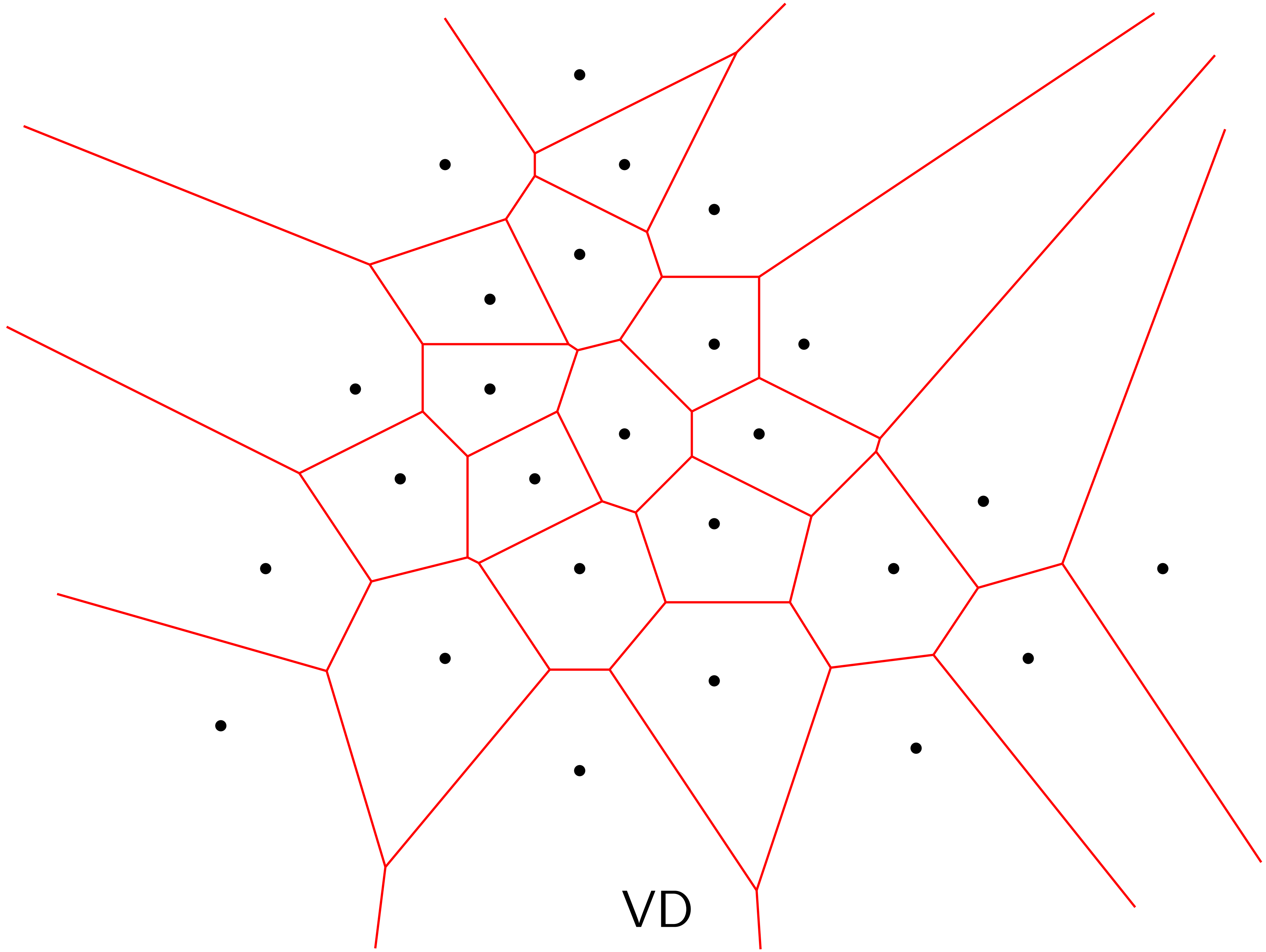
1. Imagine both $\mathrm{VD}(P)$ and $\mathrm{DT}(P)$ as planar graphs.
2. **for** every face $f \in \mathrm{VD}(P)$
   add a unique vertex v_f in $\mathrm{DT}(P)$
3. **for** every face $f \in \mathrm{VD}(P)$
   for every face f' adjacent to f
   add the edge $(v_f, v_{f'})$ in $\mathrm{DT}(P)$
4. Straighten the edges to get the triangulation

So, a reverse thinking gives $\mathrm{DT}(P)$:

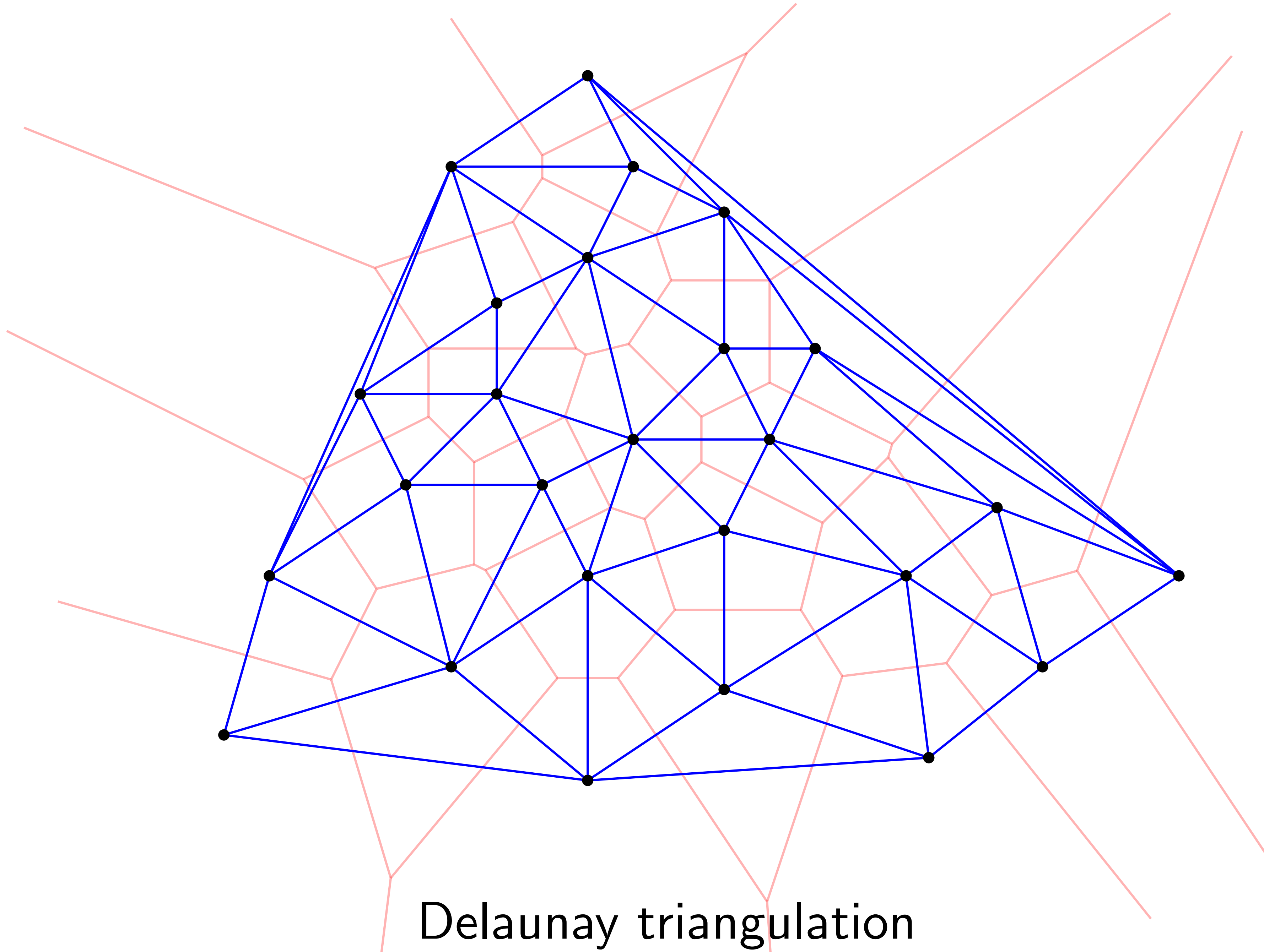
1. Construct $\mathrm{VD}(P)$.
2. Construct the dual of $\mathrm{VD}(P)$ to get $\mathrm{DT}(P)$.



Sites

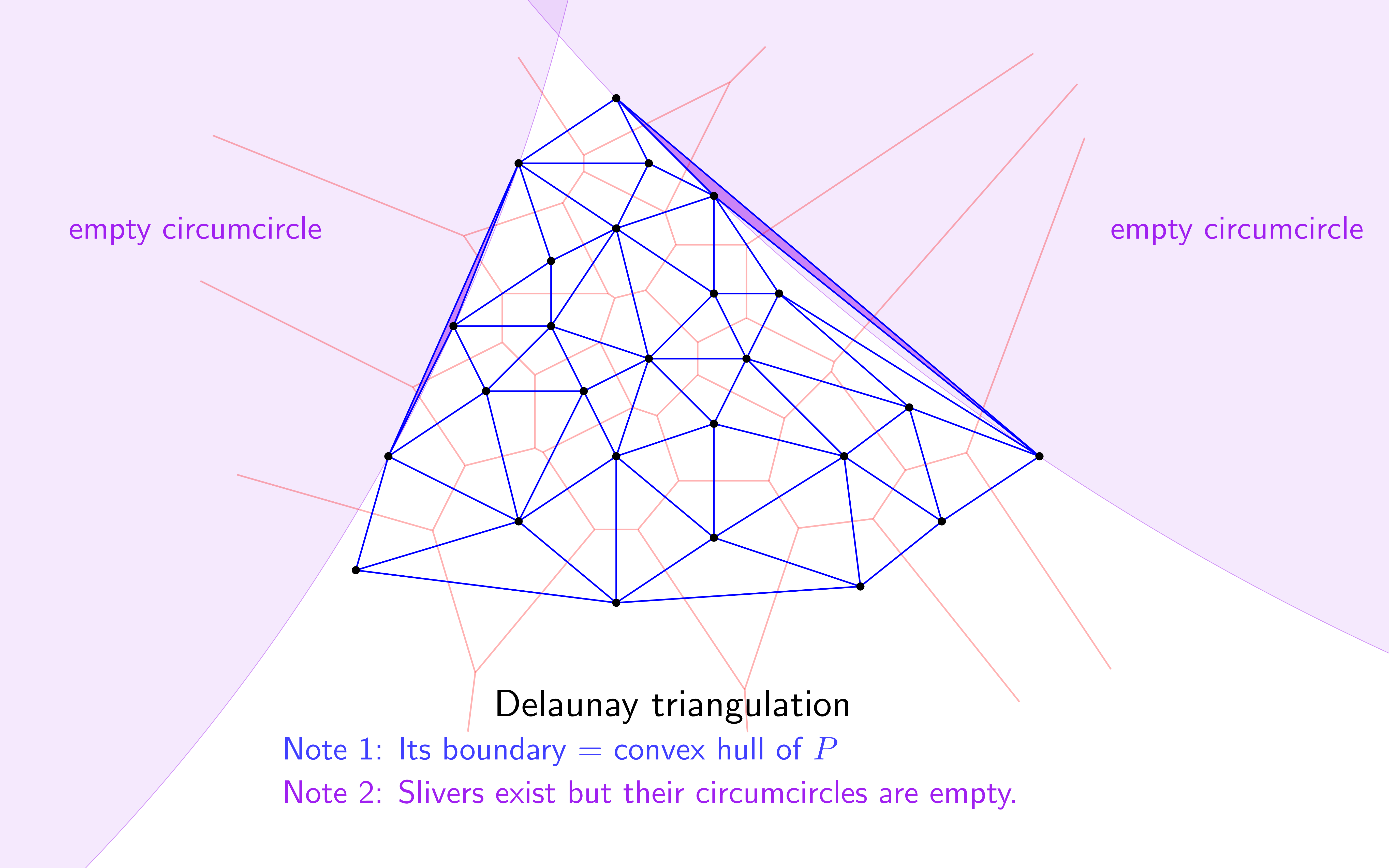


VD



Delaunay triangulation

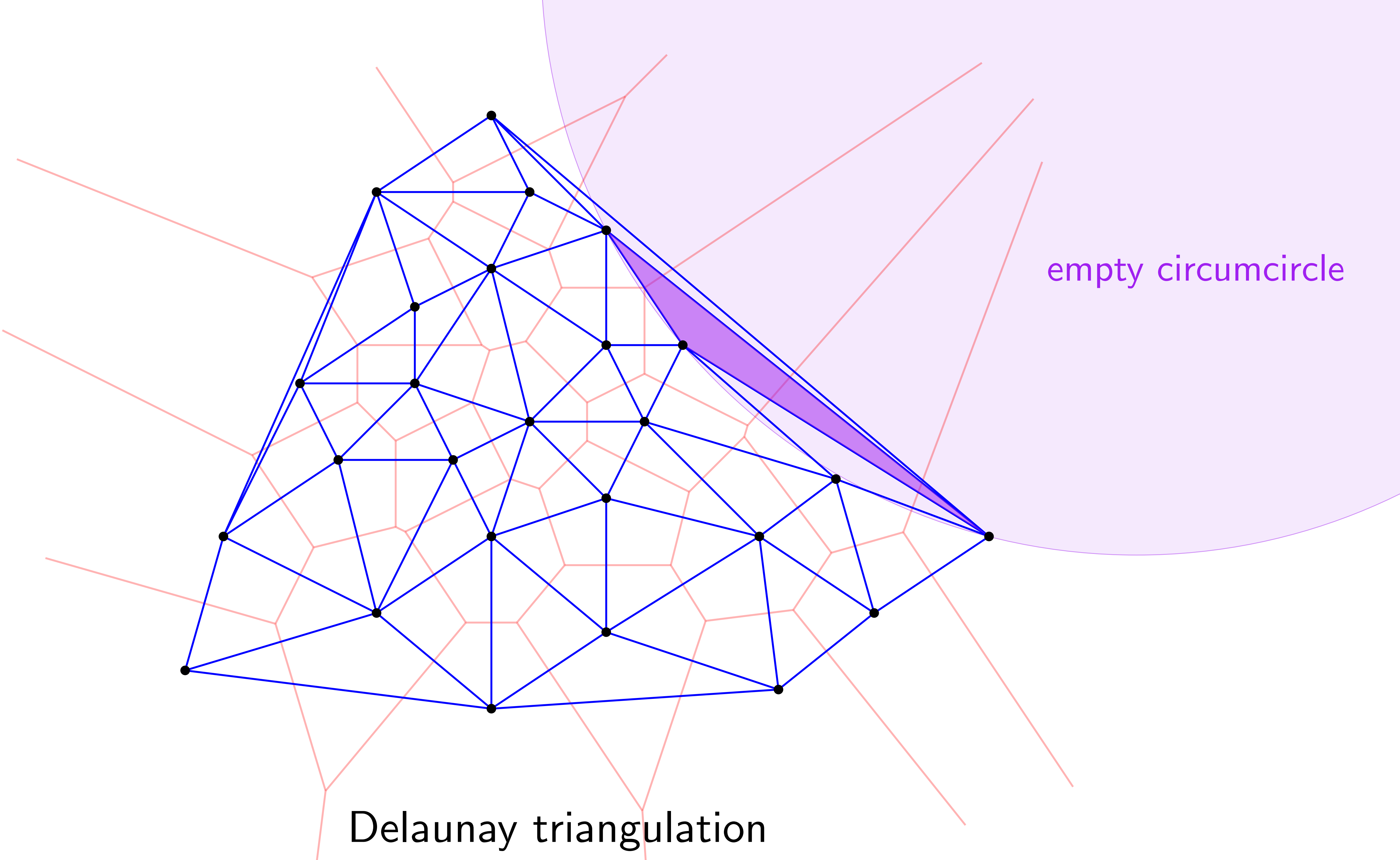
Note 1: Its boundary = convex hull of P



Delaunay triangulation

Note 1: Its boundary = convex hull of P

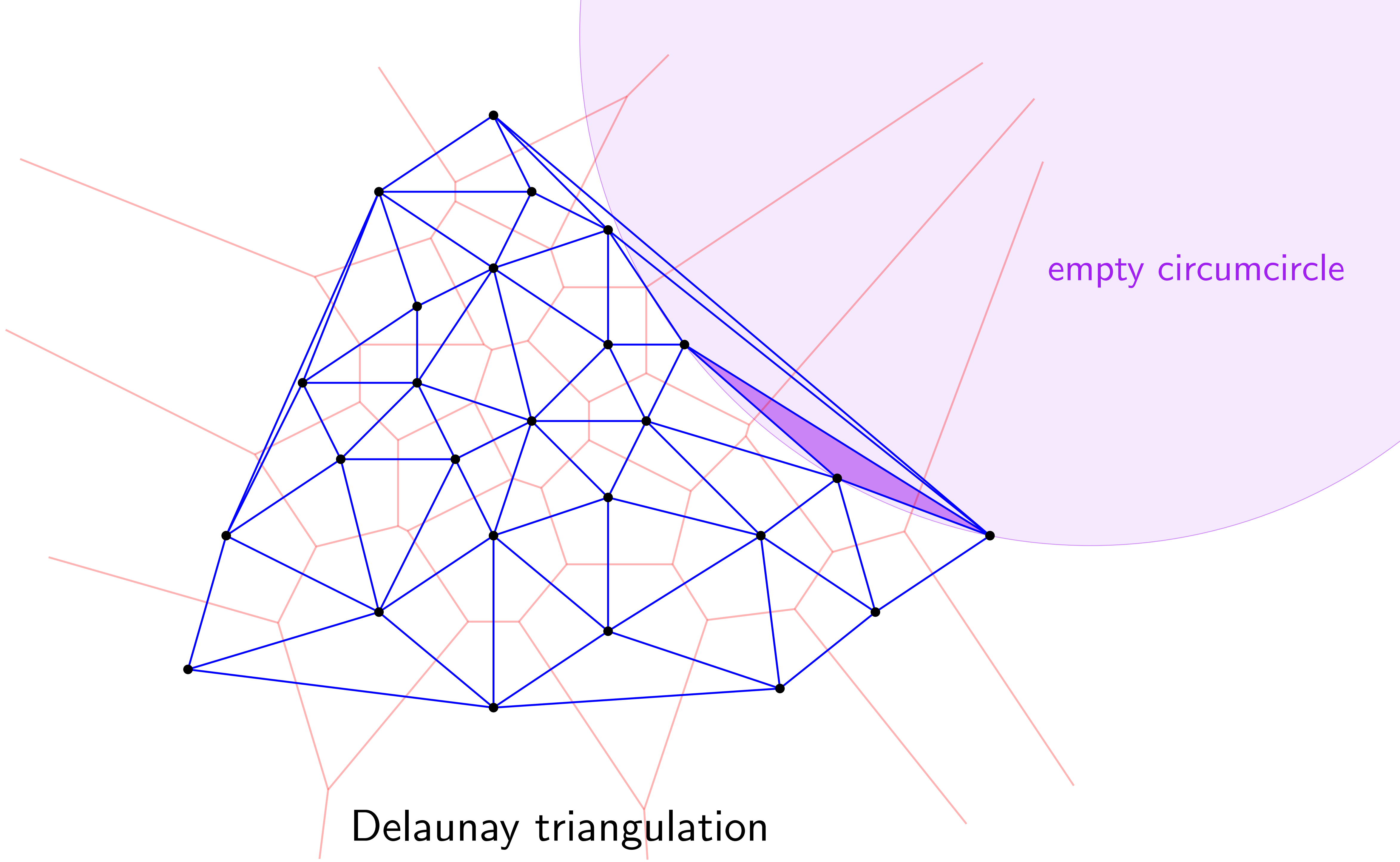
Note 2: Slivers exist but their circumcircles are empty.



Delaunay triangulation

Note 1: Its boundary = convex hull of P

Note 2: Slivers exist but their circumcircles are empty.



Delaunay triangulation

Note 1: Its boundary = convex hull of P

Note 2: Slivers exist but their circumcircles are empty.

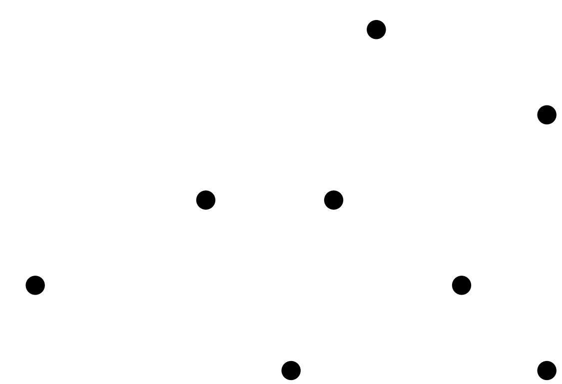
Algorithms for DT

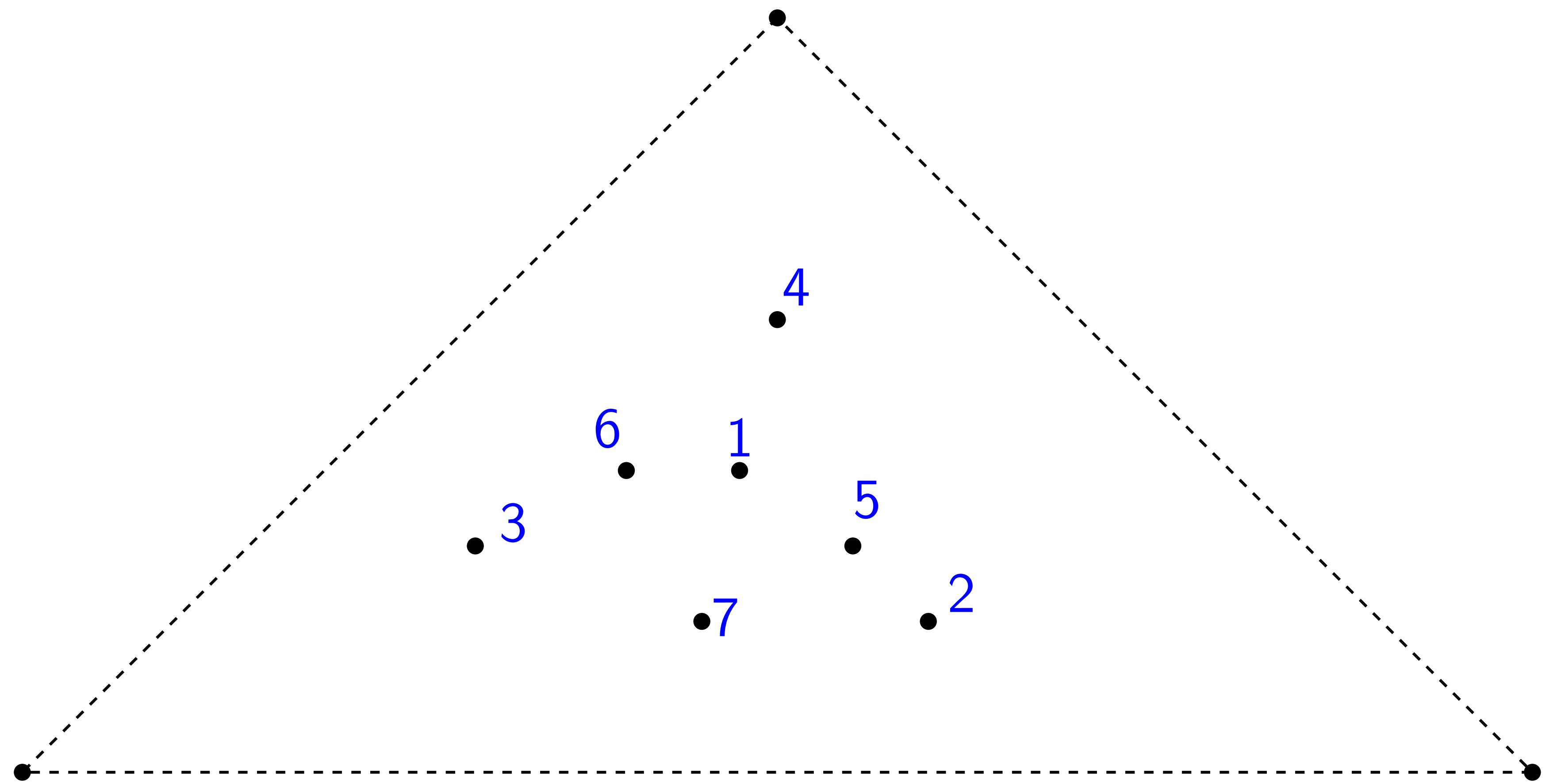
1. Using VT [**last slide**] — $O(n \log n)$ time

2. Independent (simple) randomized algorithm using search tree and edge-flip operations — $O(n \log n)$ expected time

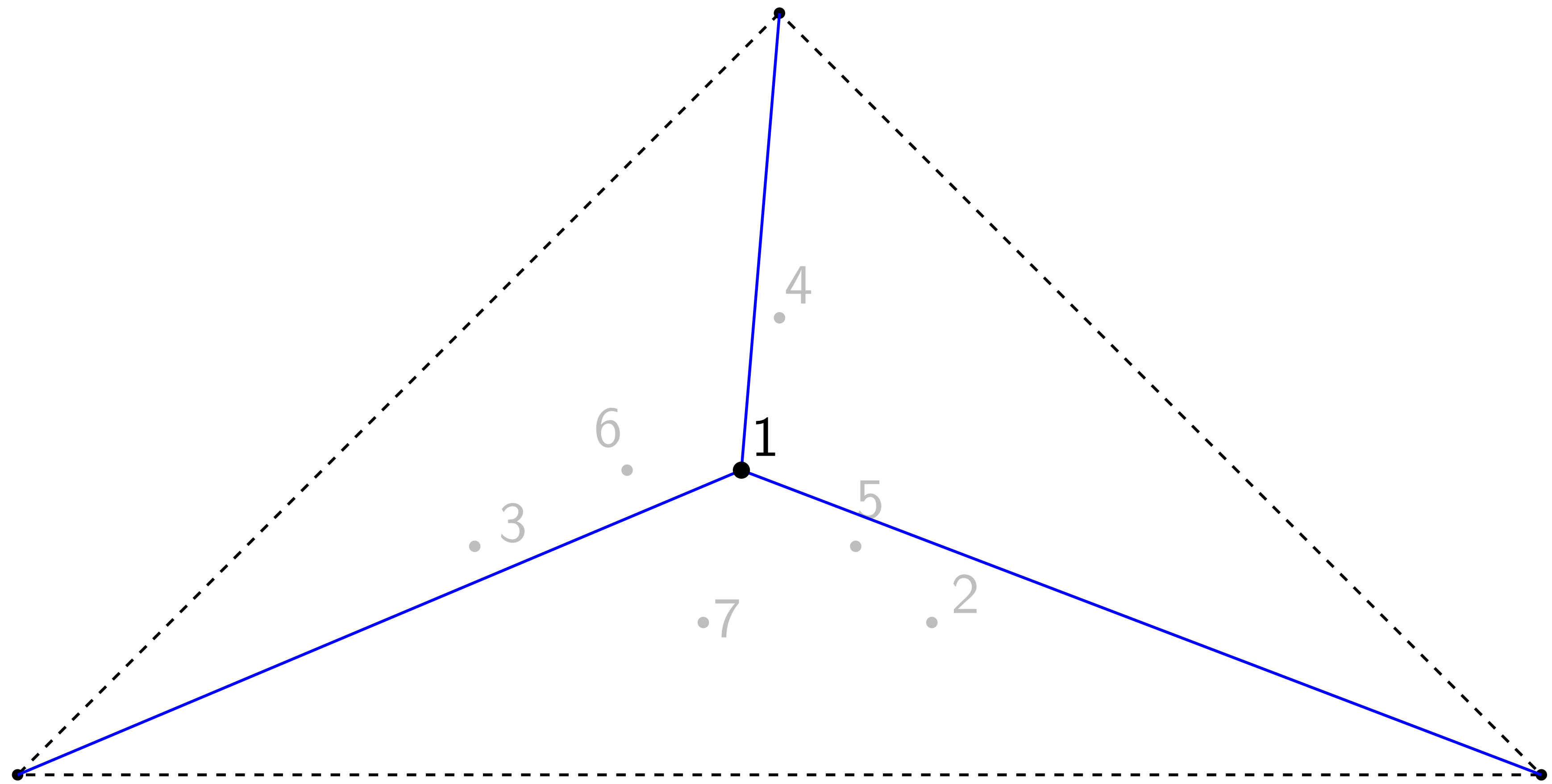
3. Using only edge-flips iteratively (greedy) — $O(n^2)$ time, convergence in question

2. Randomized algorithm using search tree

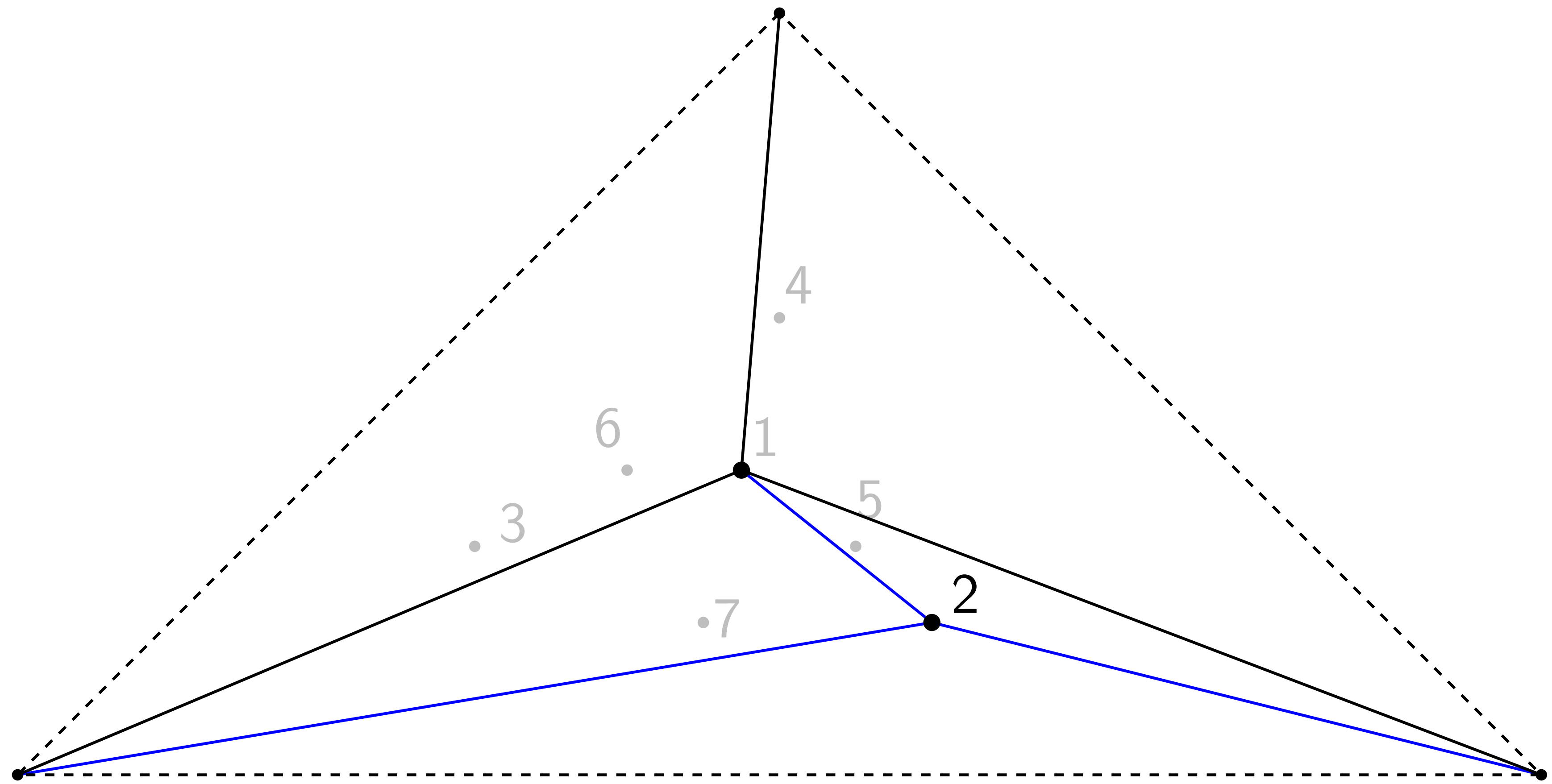




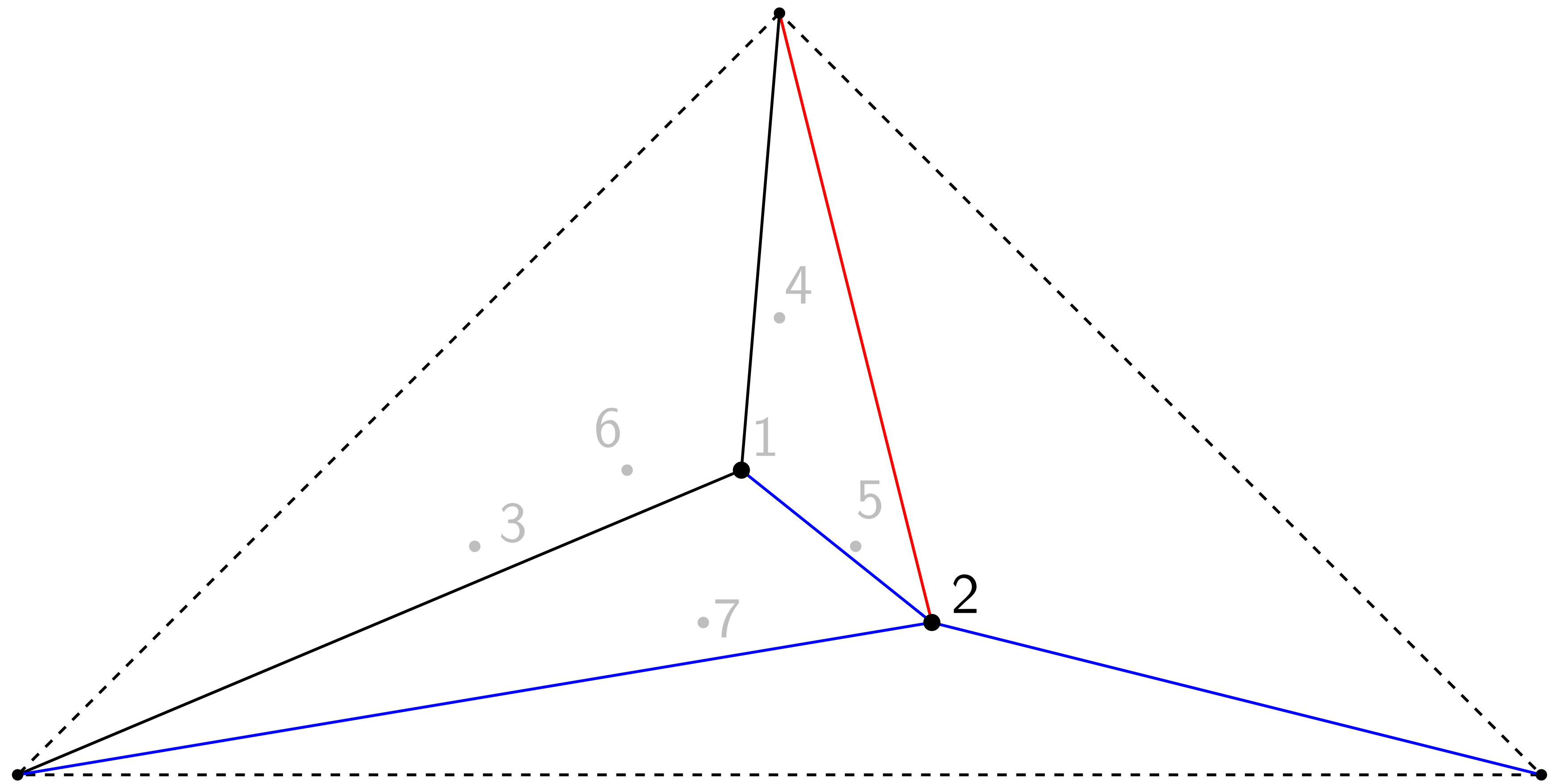
Take a huge triangle (with 3 extra sites as vertices),
and label the sites randomly.



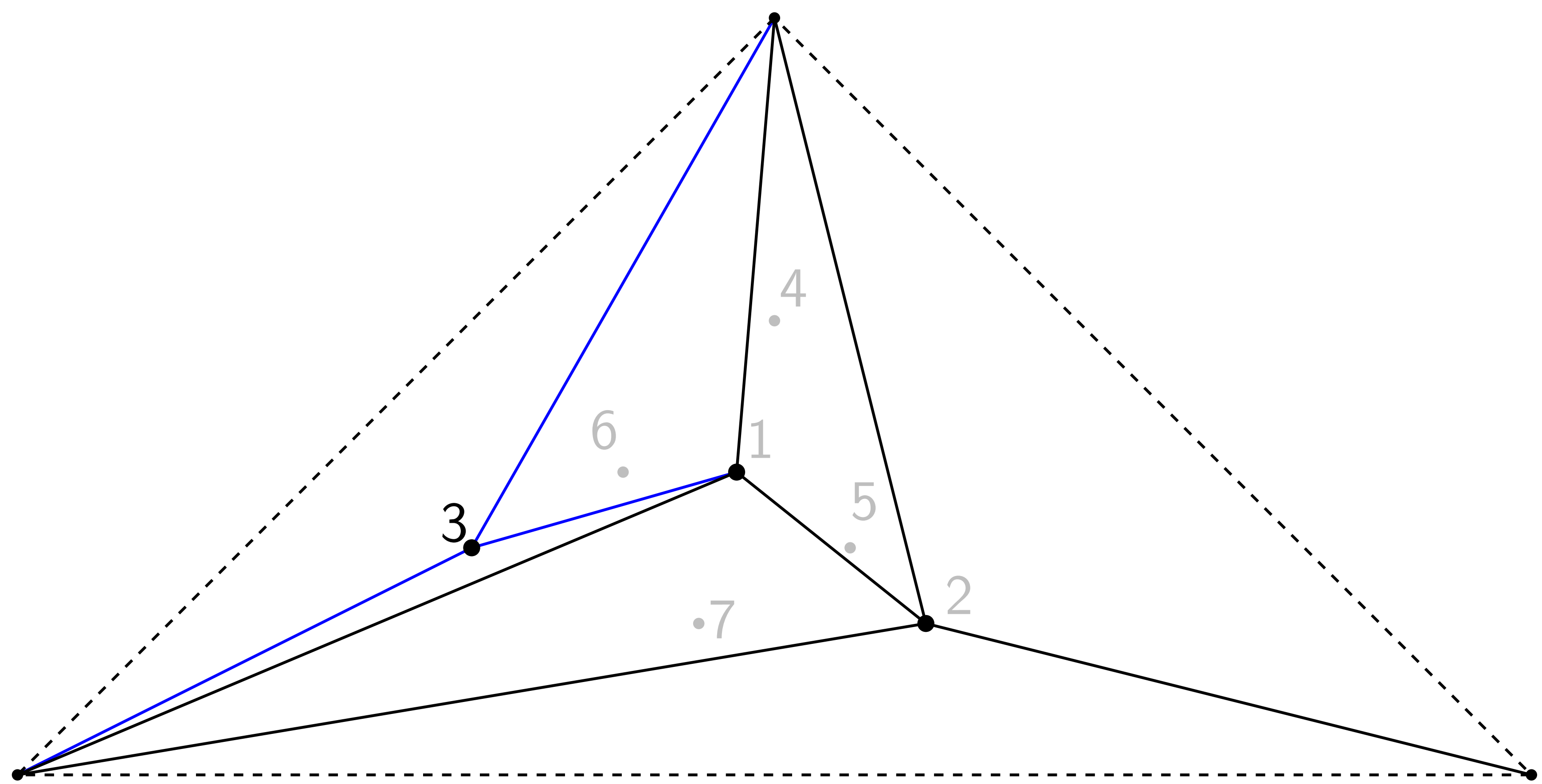
Triangulation for p_1



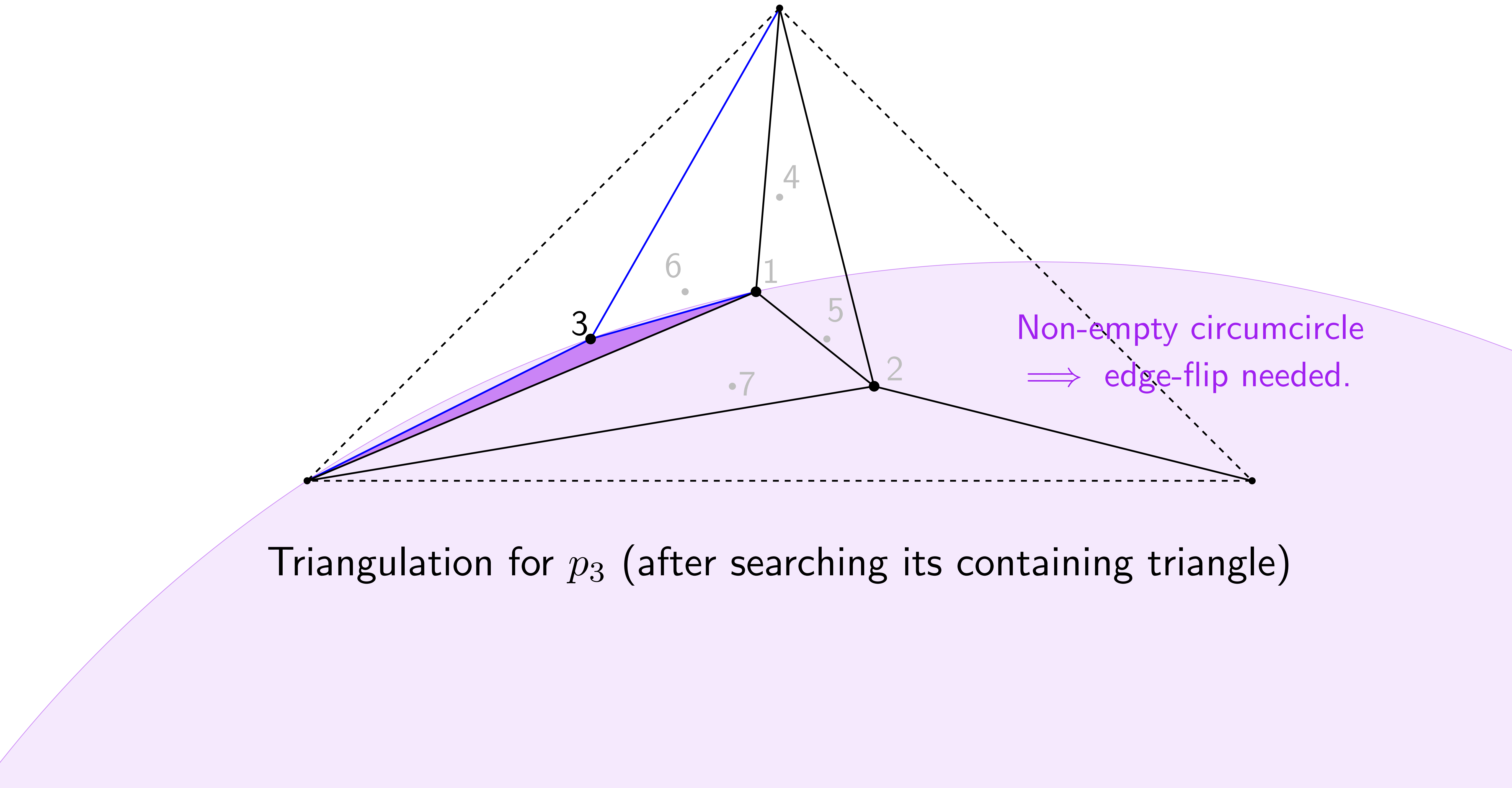
Triangulation for p_2 (after searching its containing triangle)



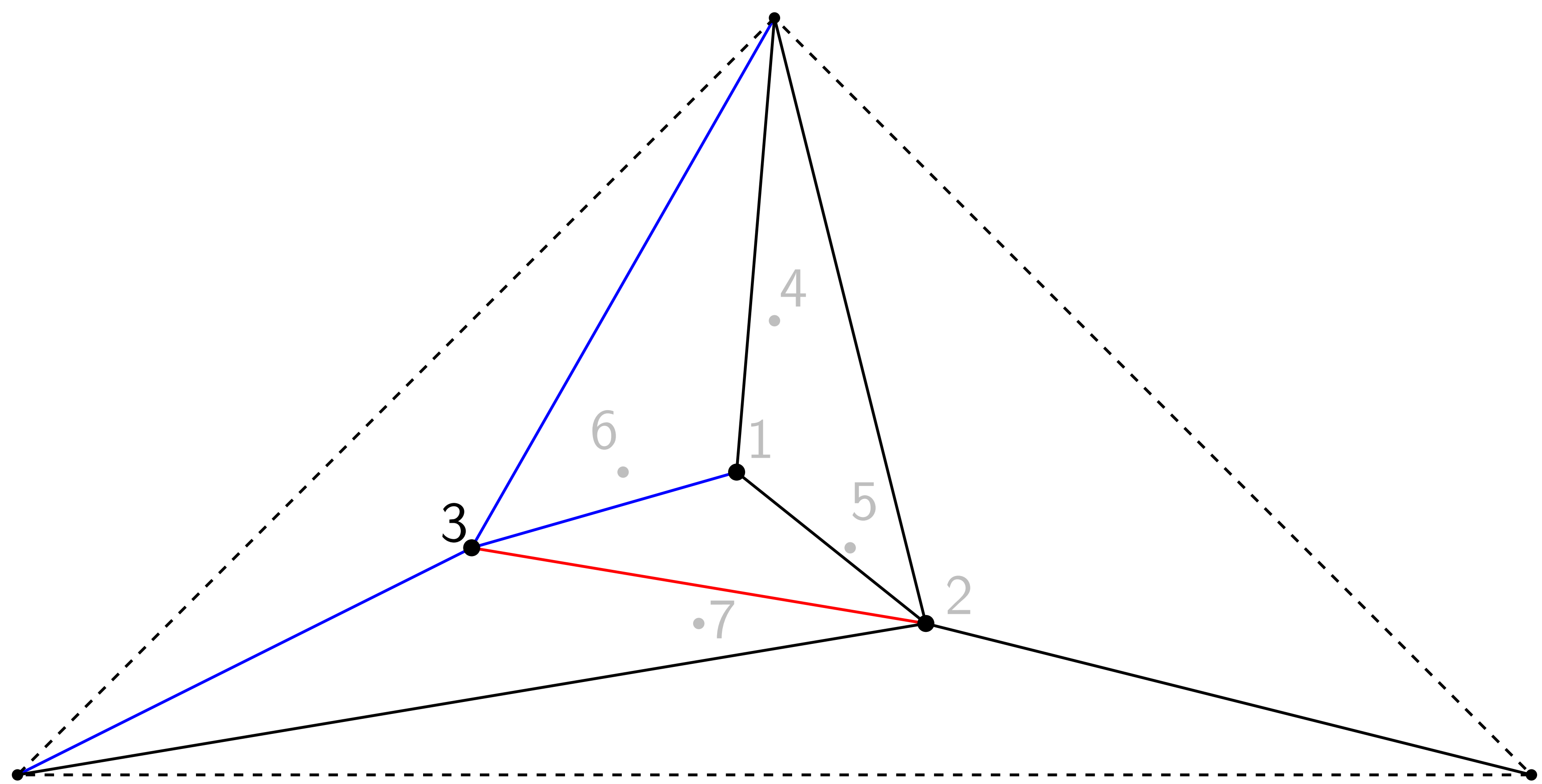
Edge-flip after triangulation for p_2



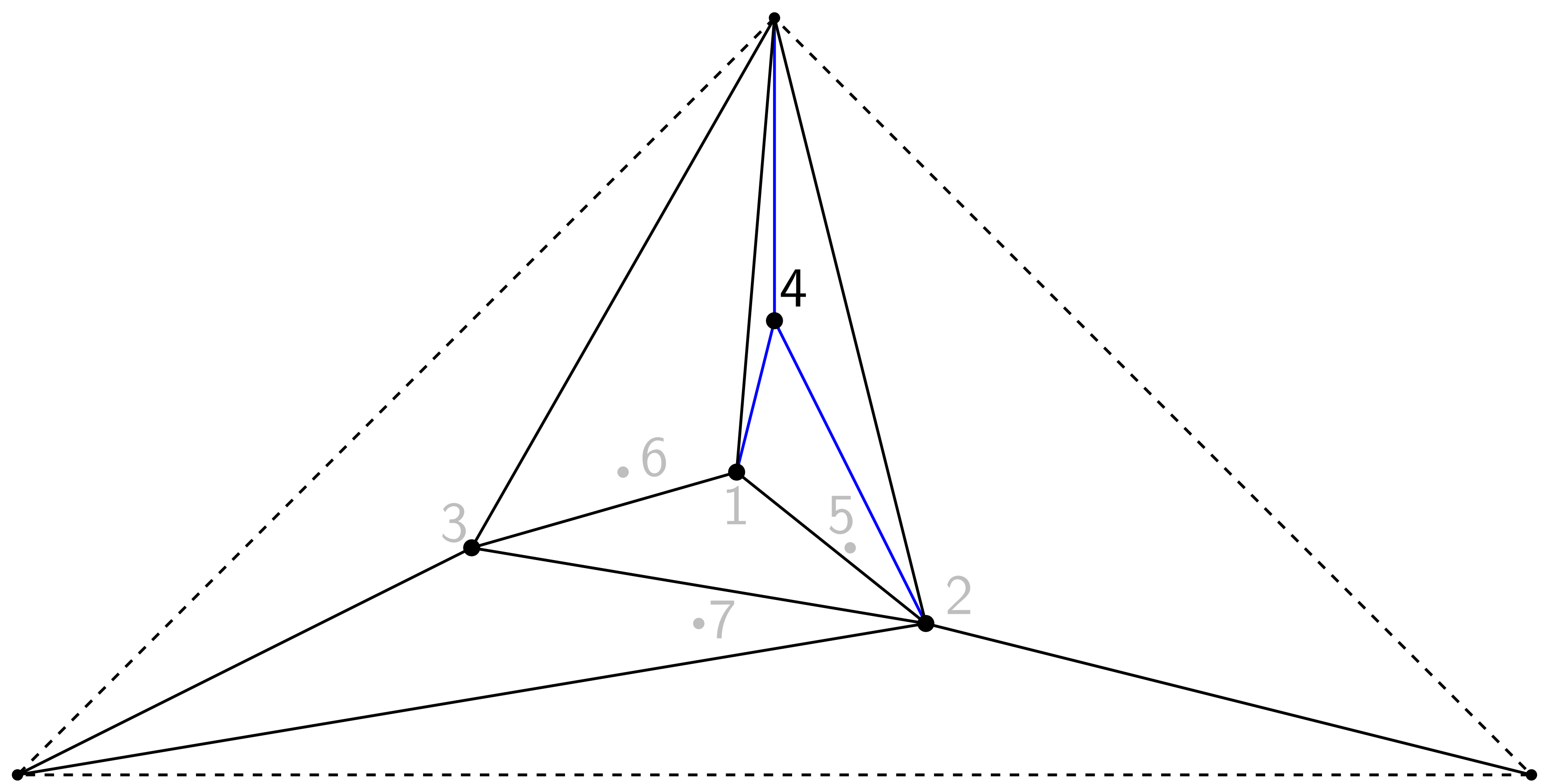
Triangulation for p_3 (after searching its containing triangle)



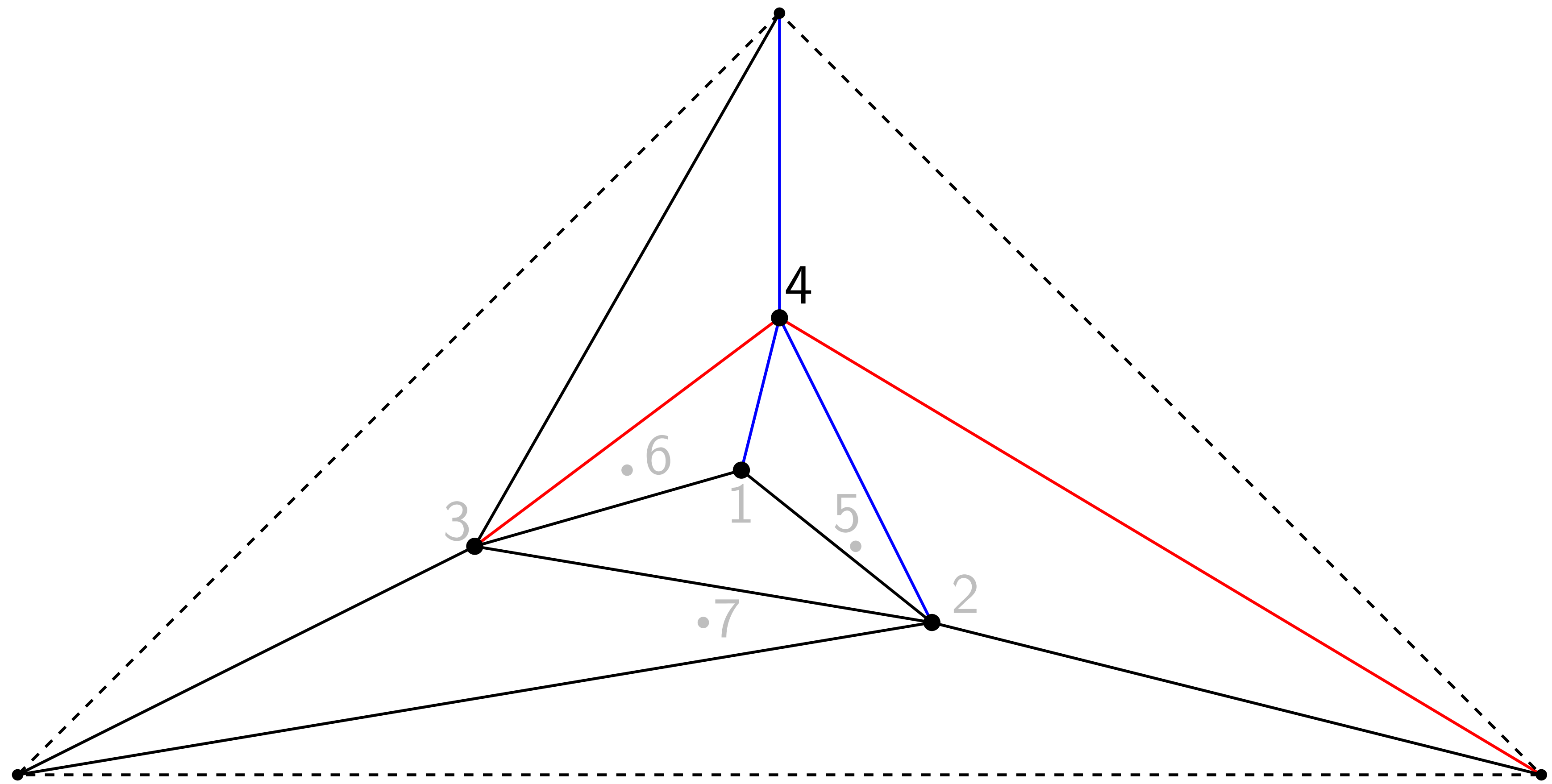
Triangulation for p_3 (after searching its containing triangle)



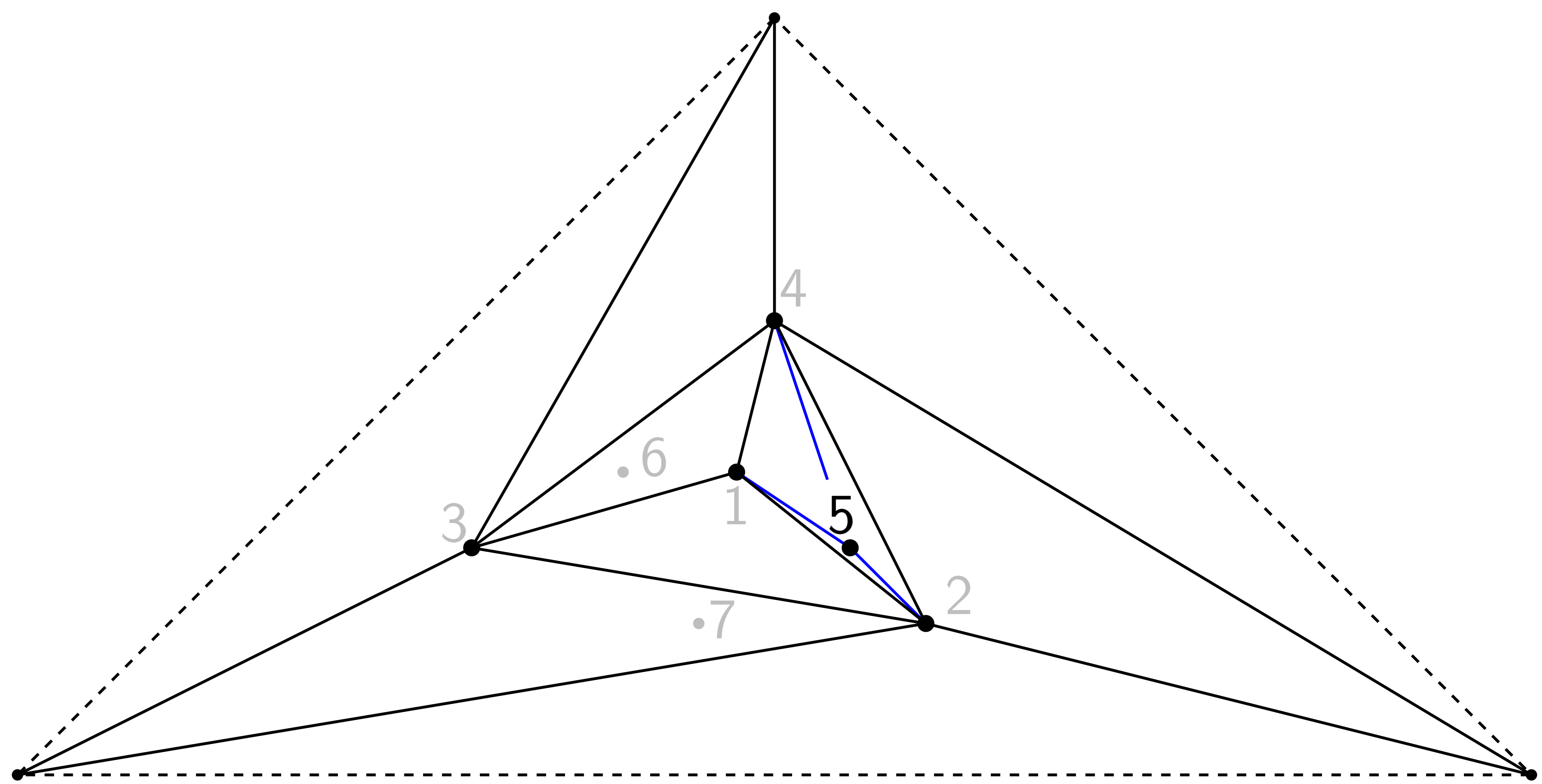
Edge-flip after triangulation for p_3



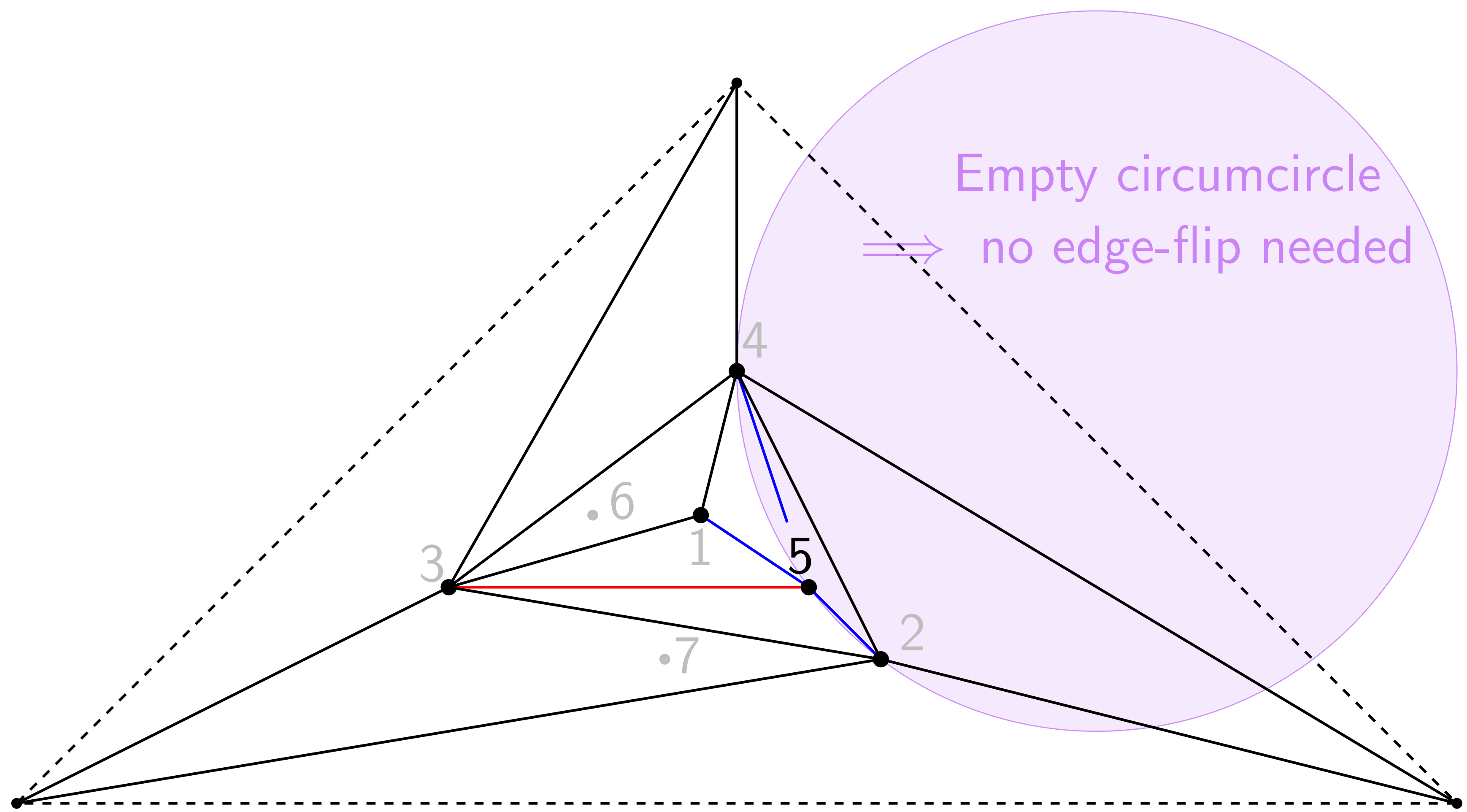
Triangulation for p_4 (after searching its containing triangle)



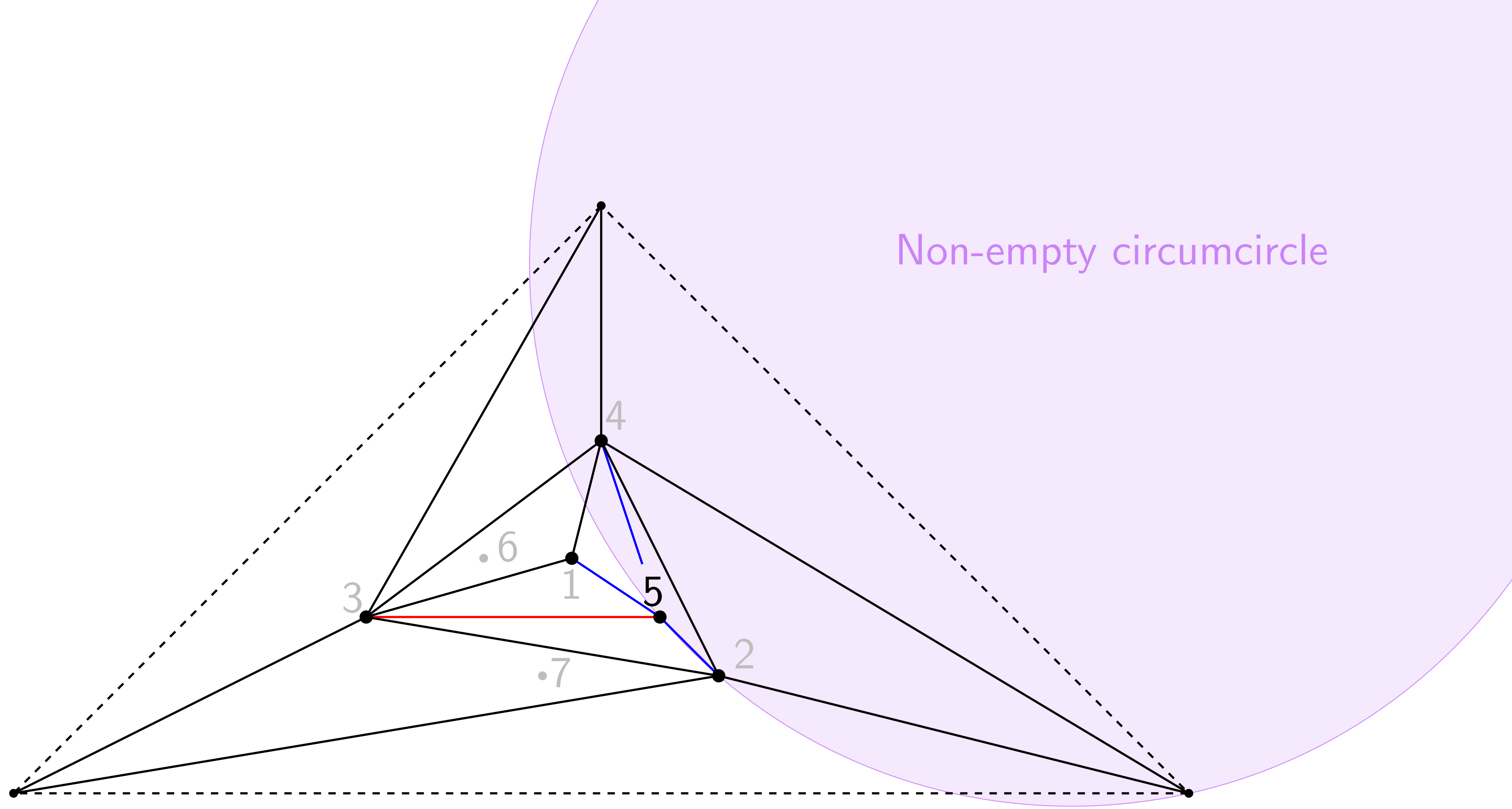
Edge-flips after triangulation for p_4



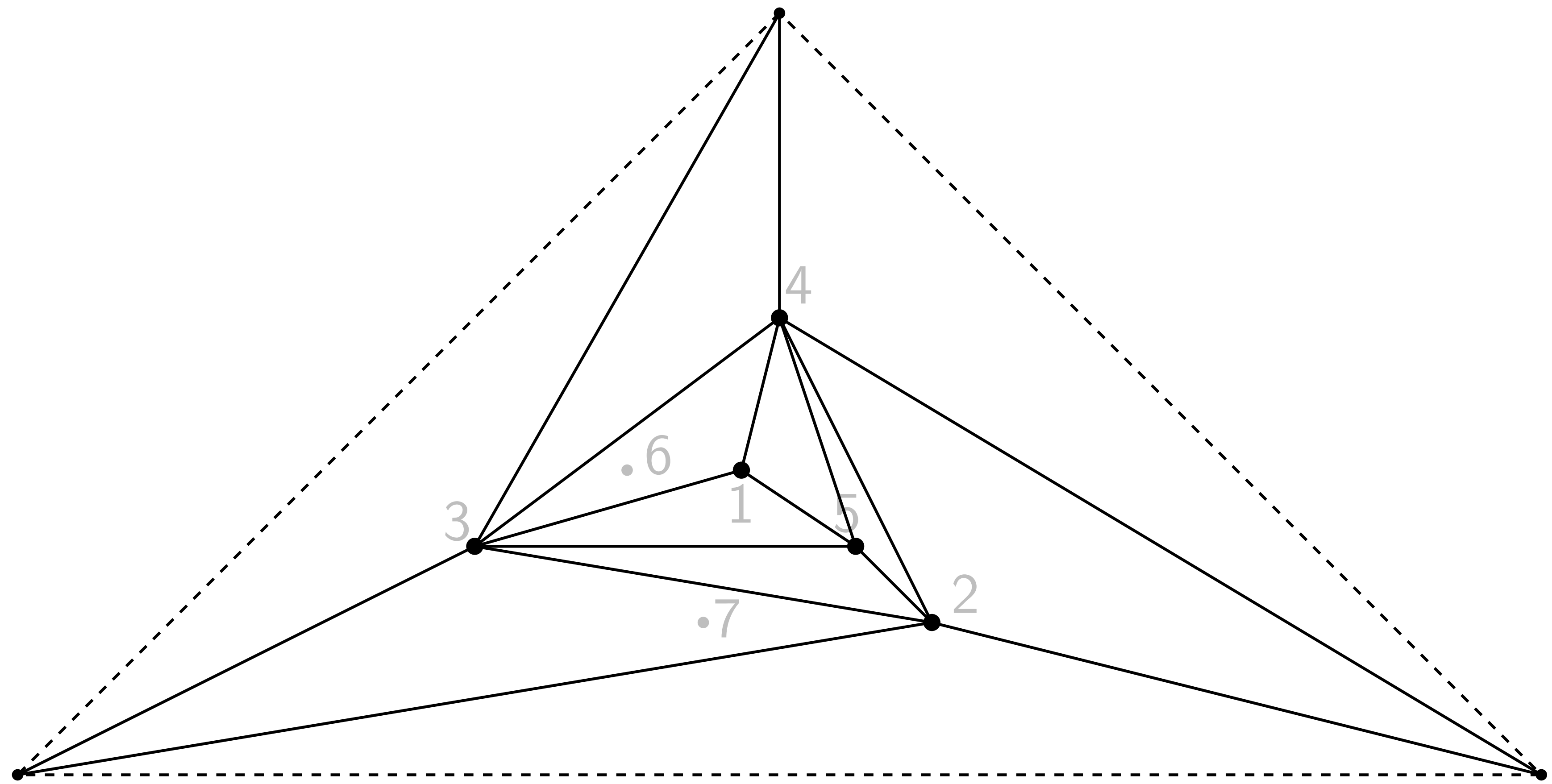
Triangulation for p_5



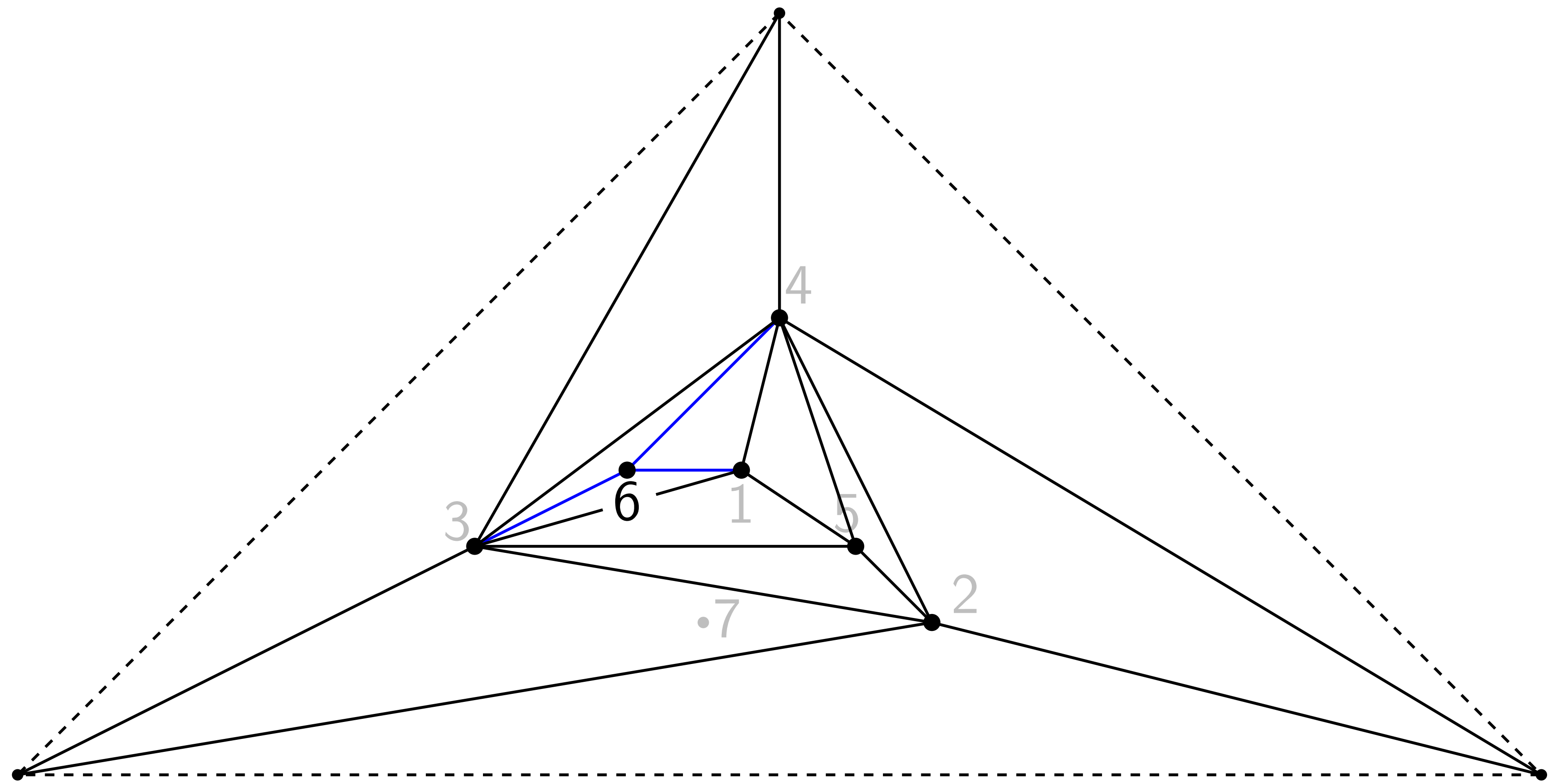
Edge-flips after triangulation for p_5



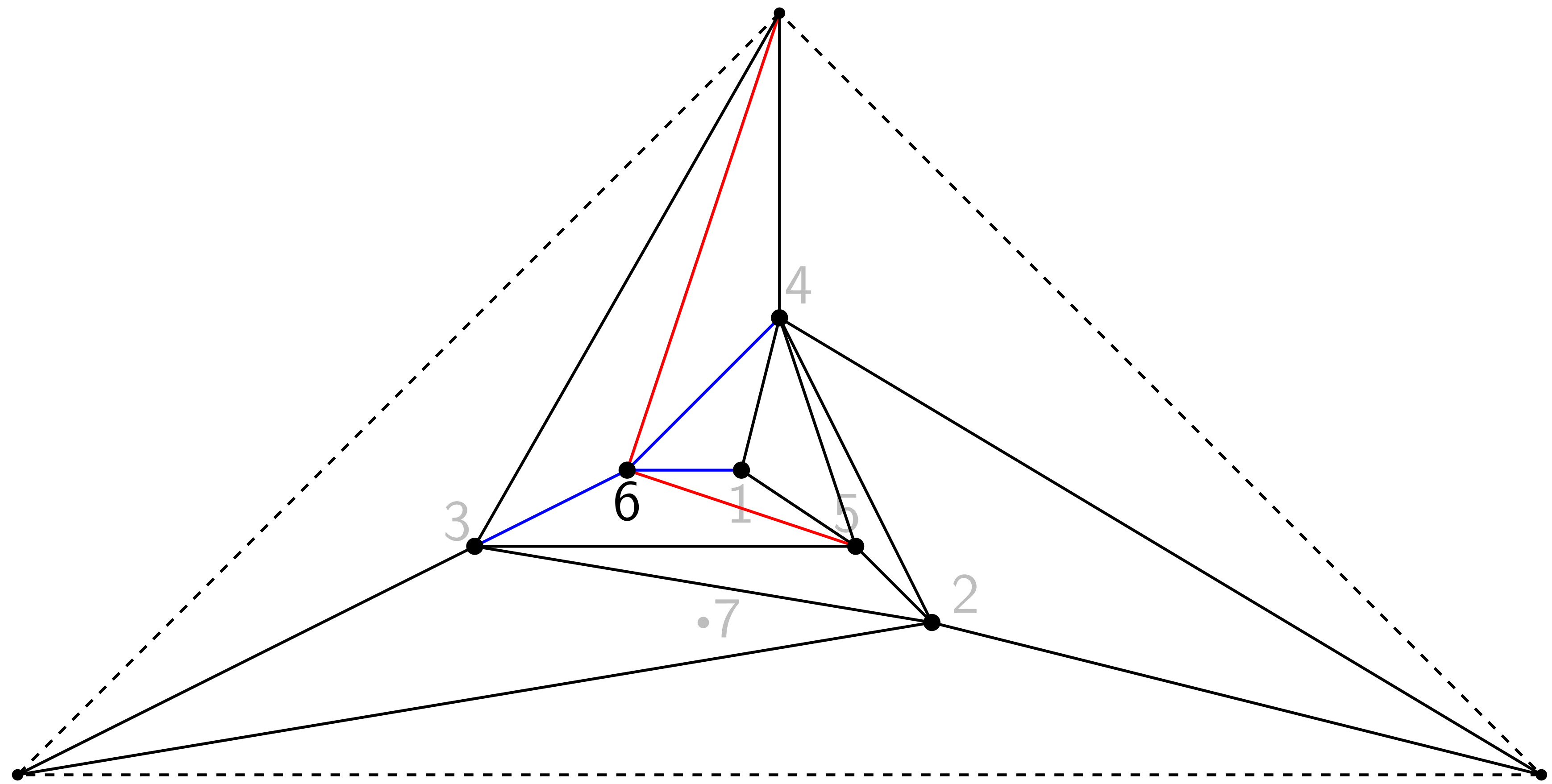
Edge-flips after triangulation for p_5



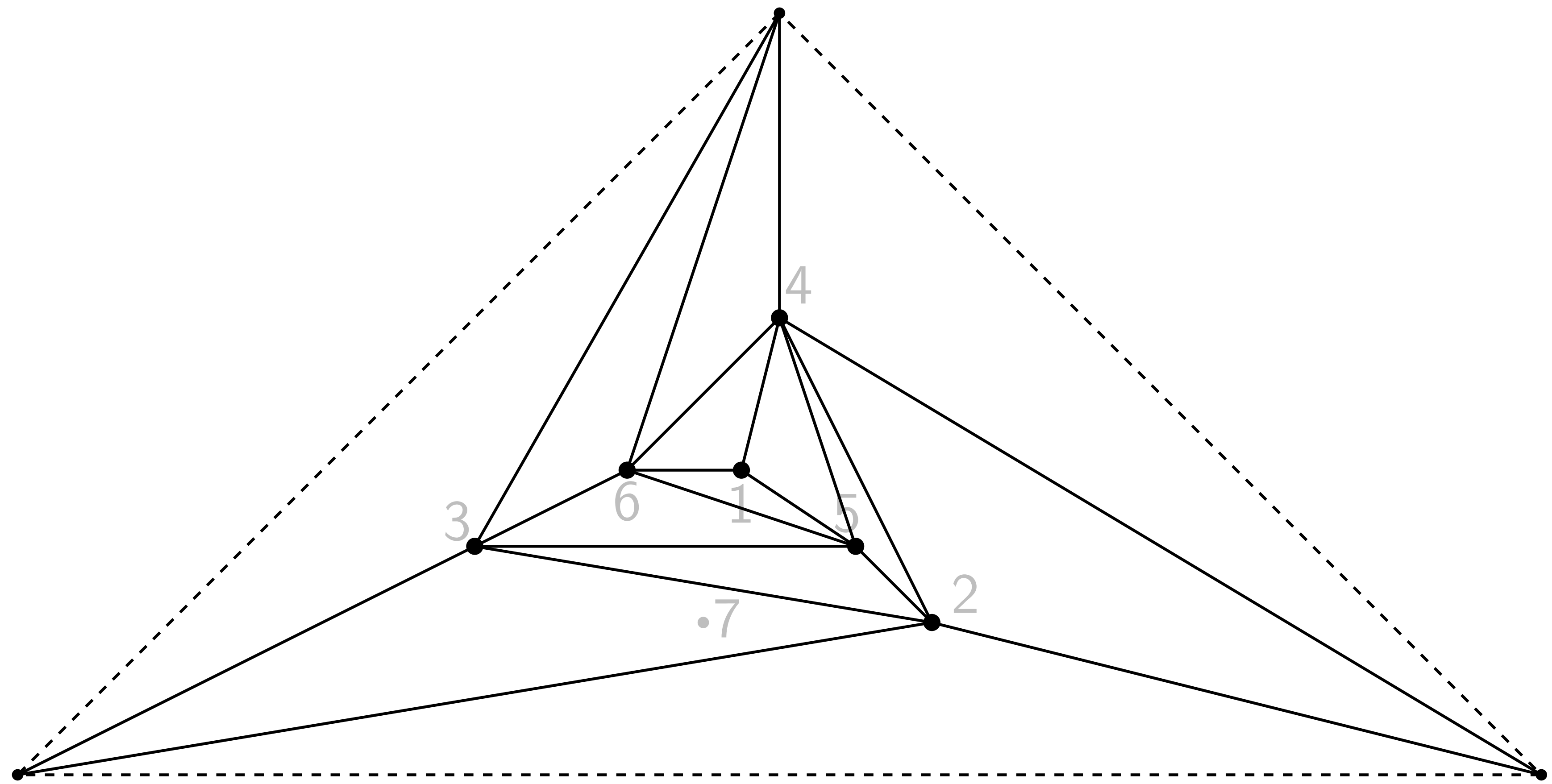
Triangulation for $p_1, \ldots, p_5$



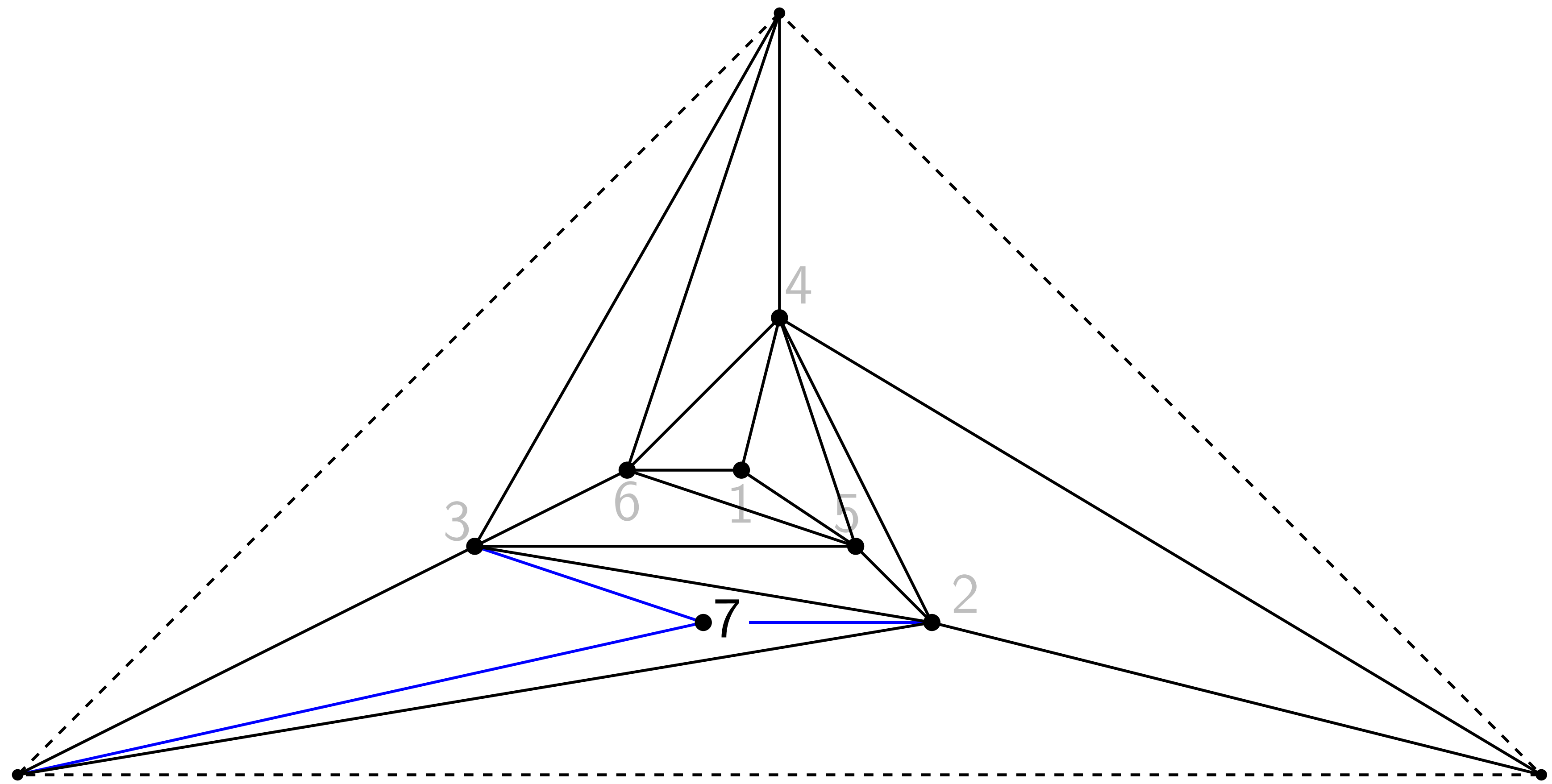
Triangulation for p_6



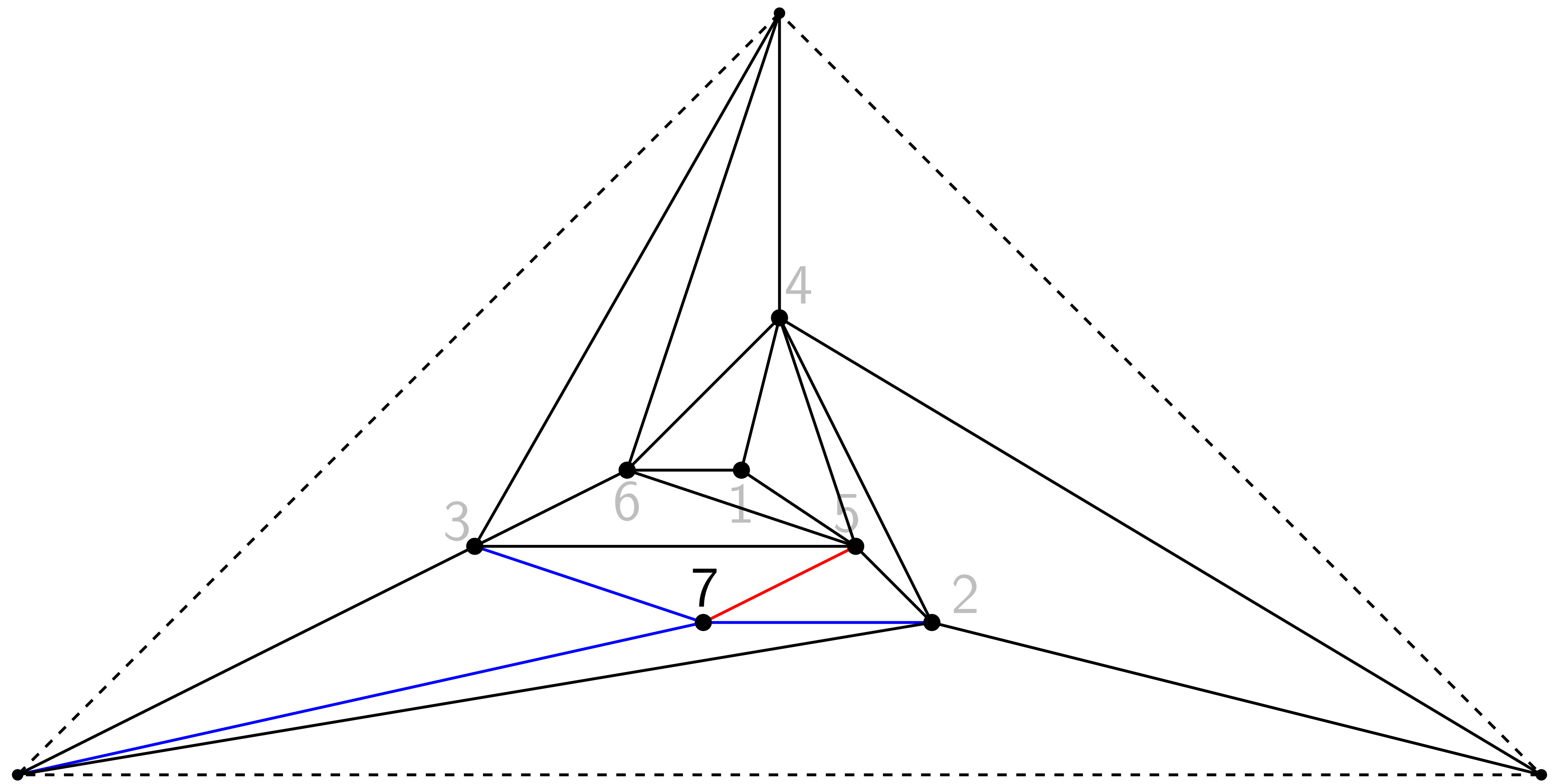
Edge-flips after triangulation for p_6



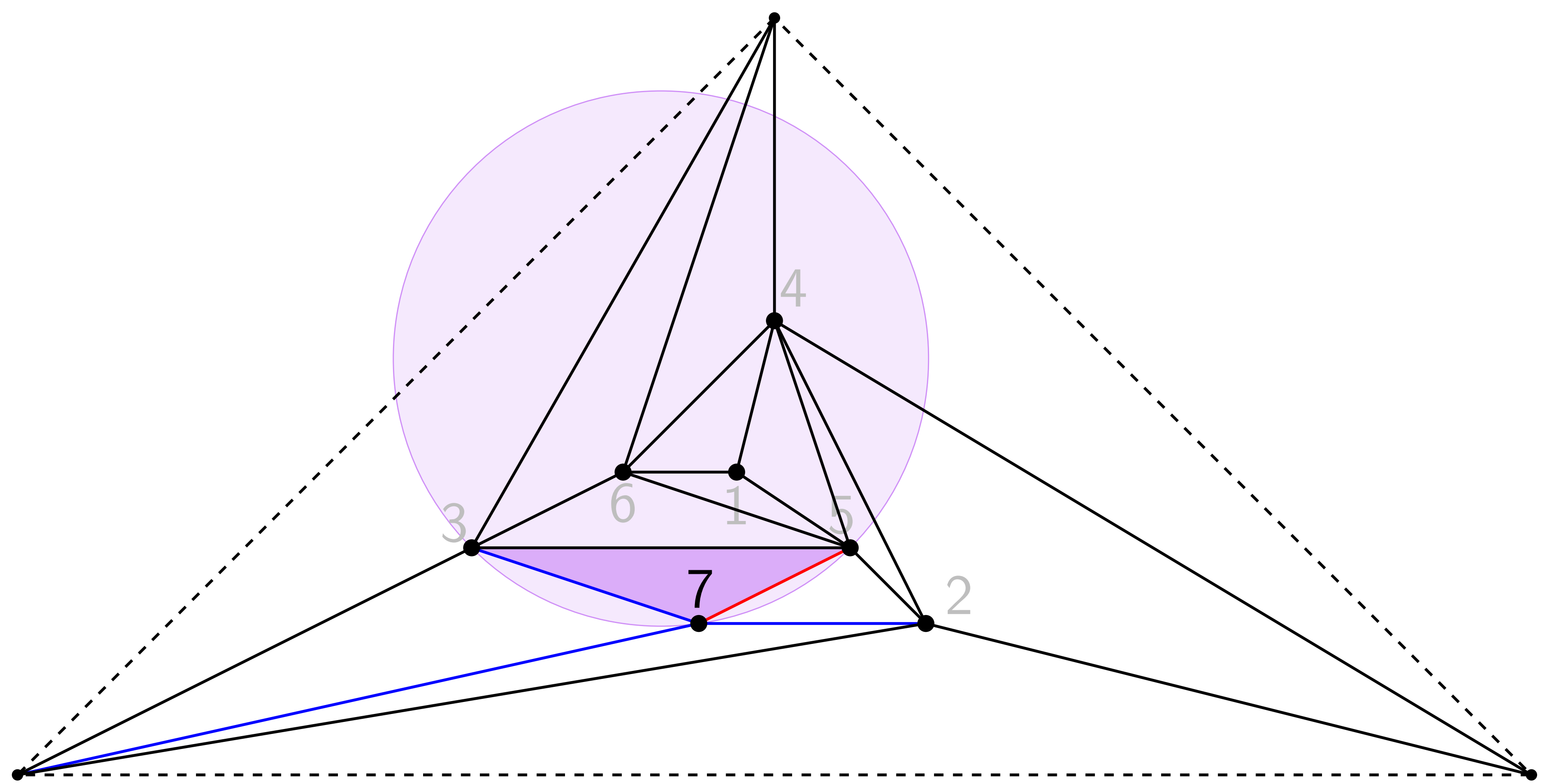
Triangulation for $p_1, \dots, p_6$



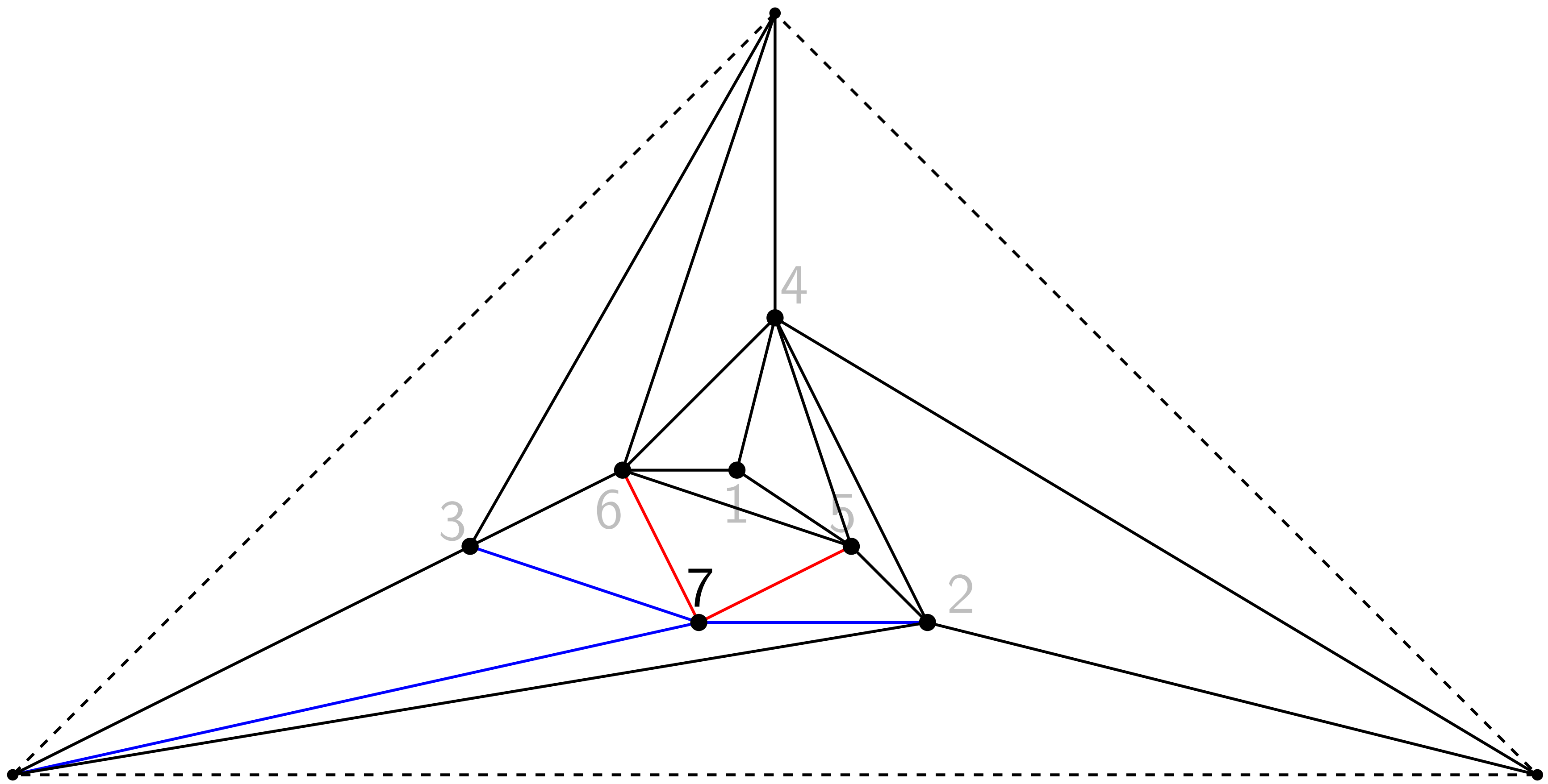
Triangulation for p_7



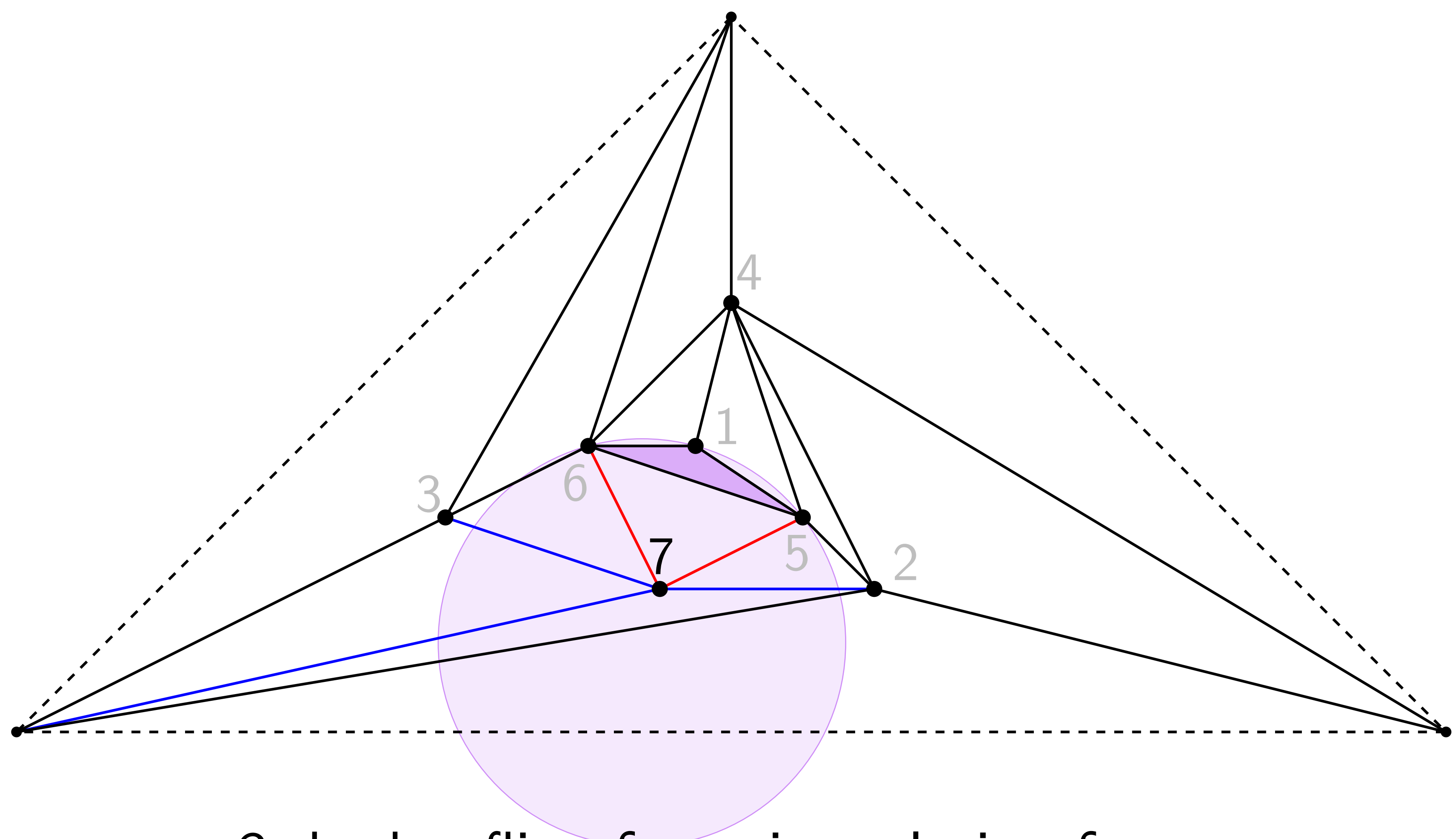
1st edge-flip after triangulation for p_7



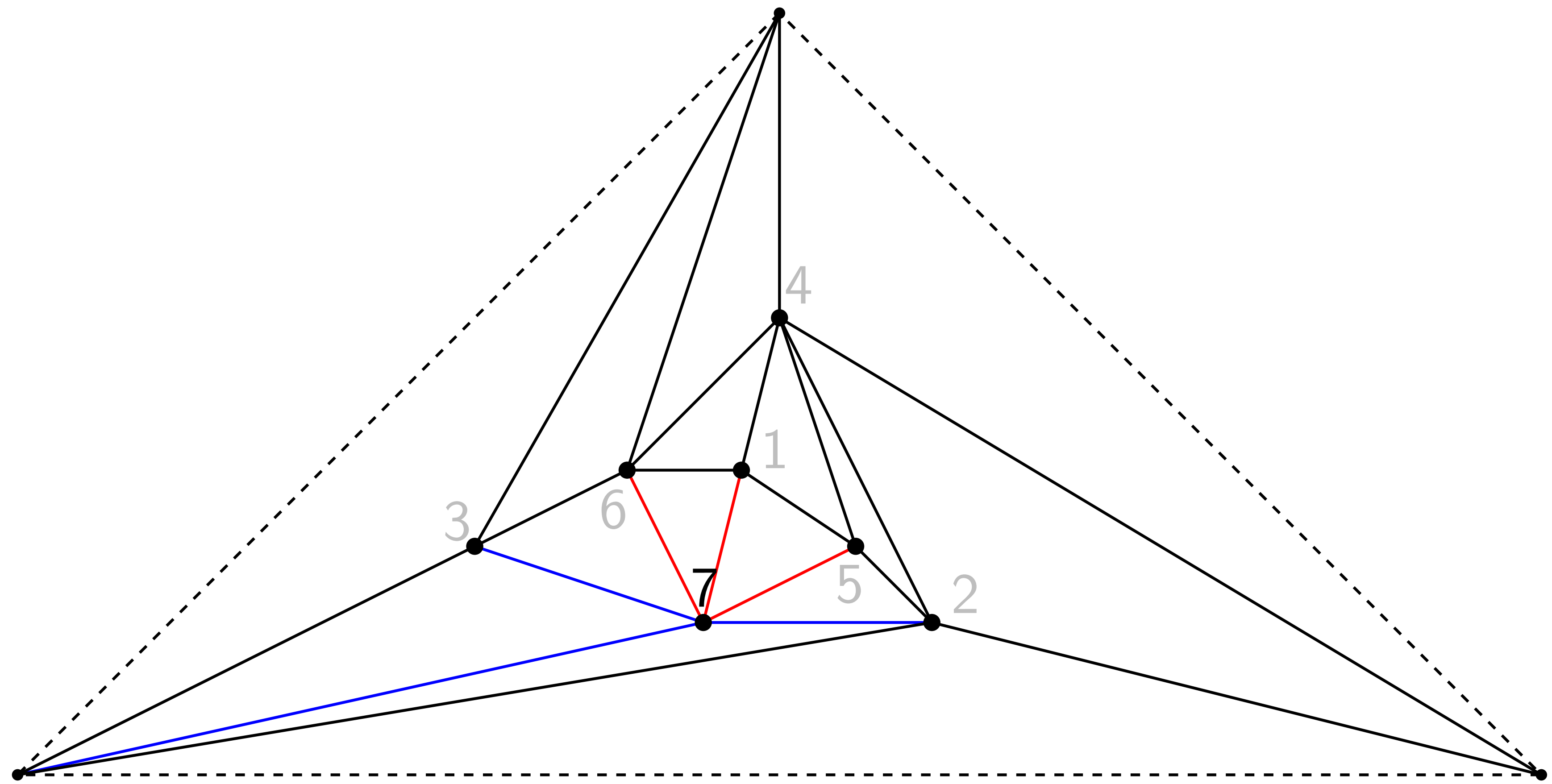
2nd edge-flip after triangulation for p_7



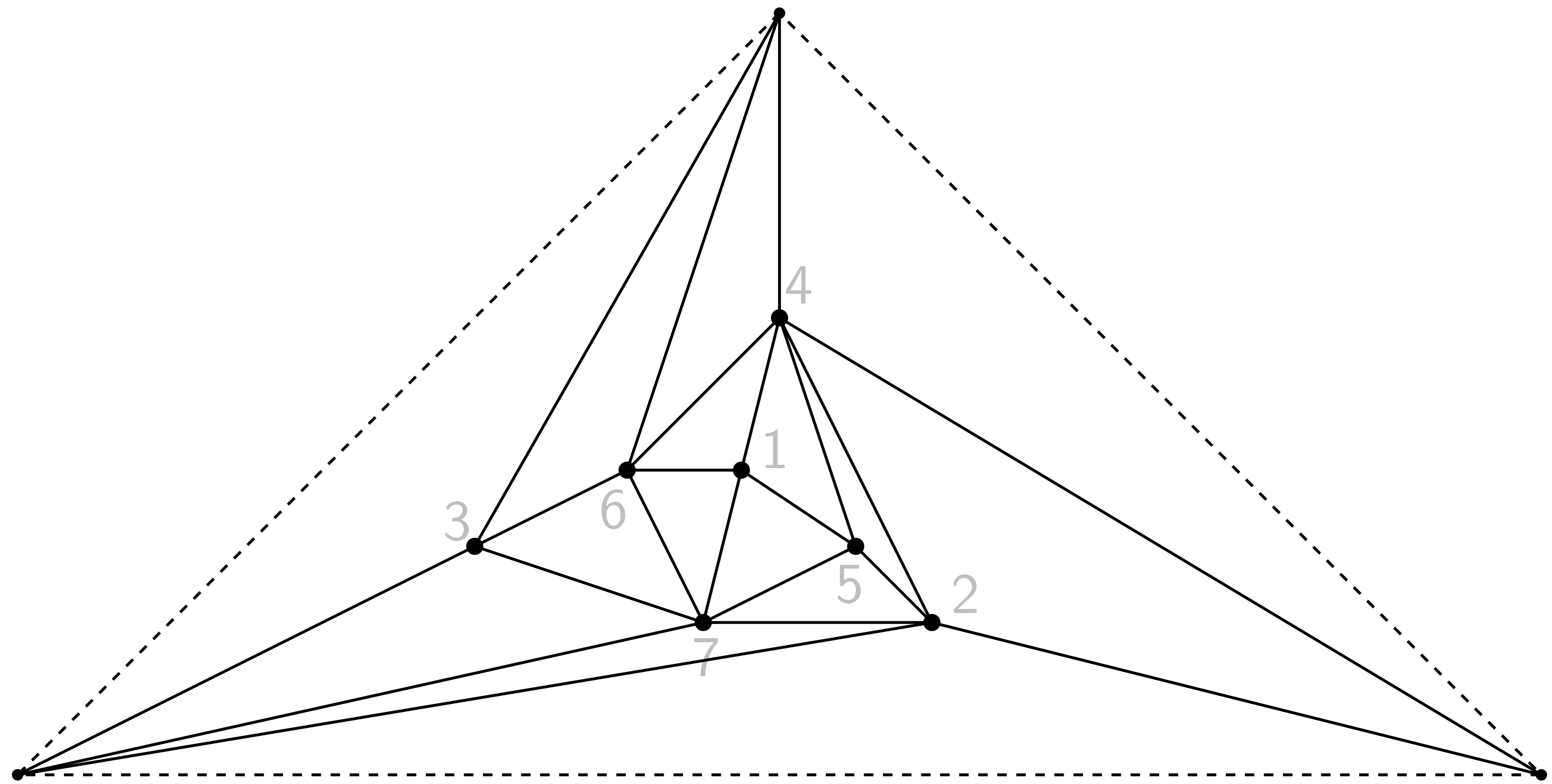
2nd edge-flip after triangulation for p_7



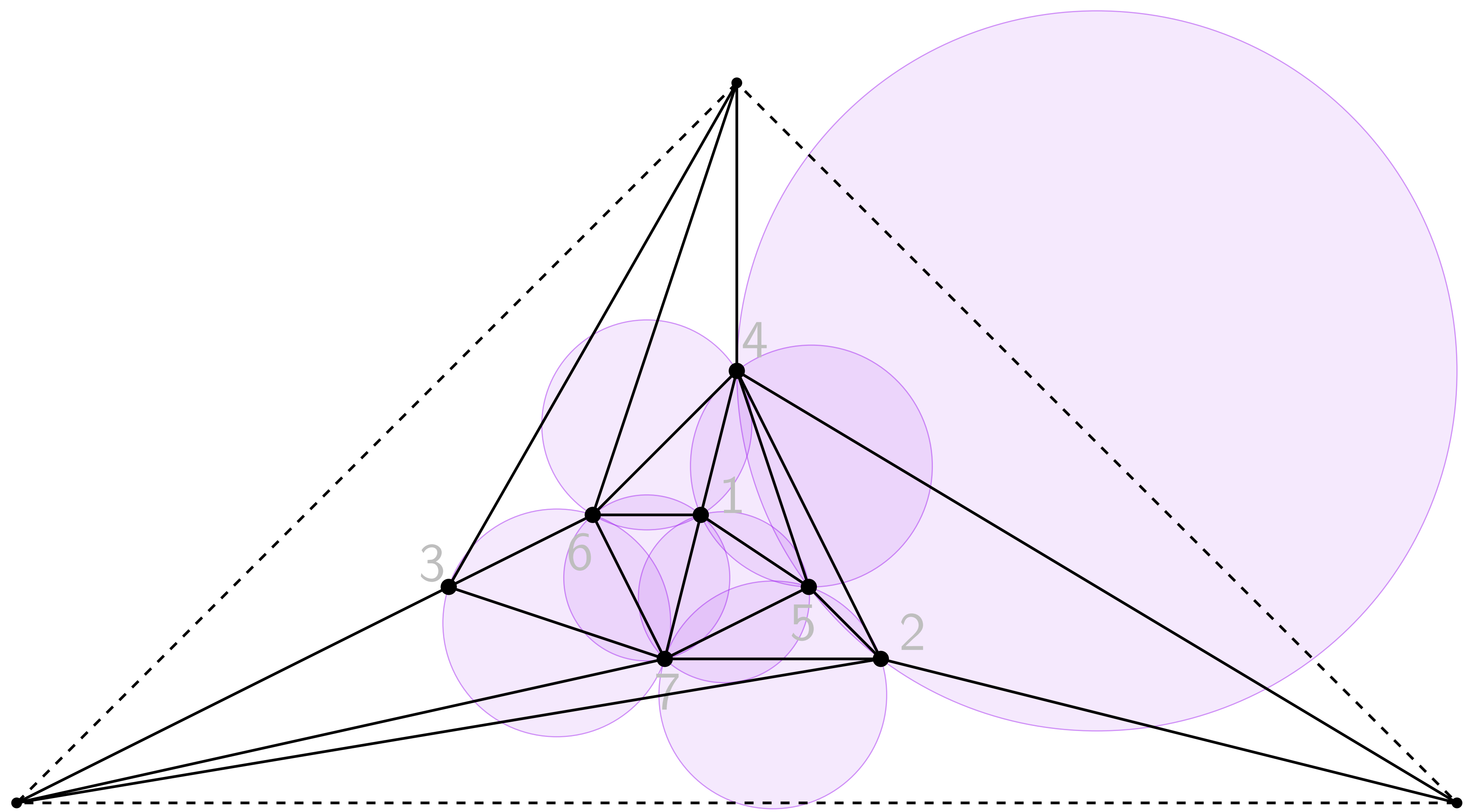
3rd edge-flip after triangulation for p_7



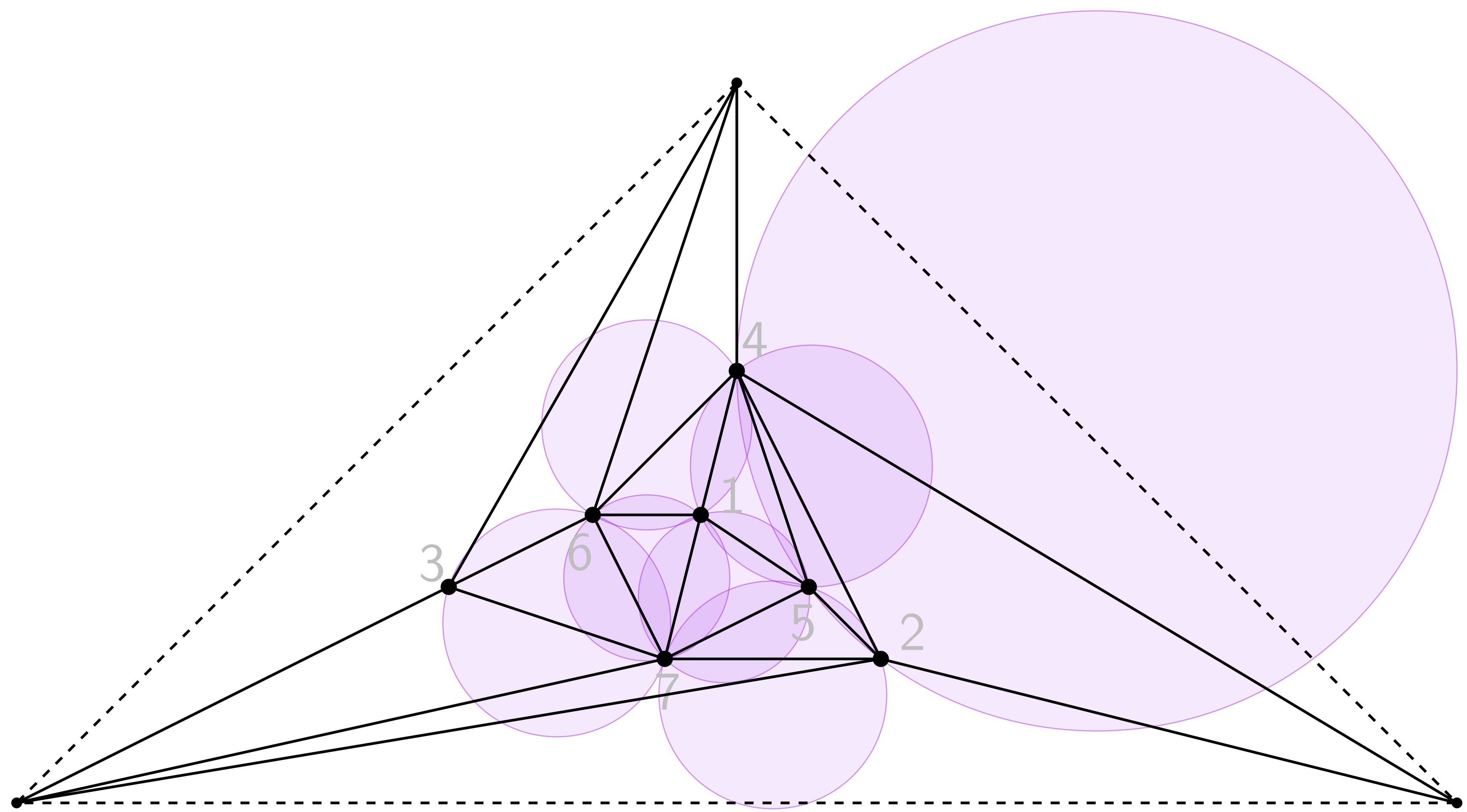
3rd edge-flip after triangulation for p_7



Triangulation for $p_1, \ldots, p_7$

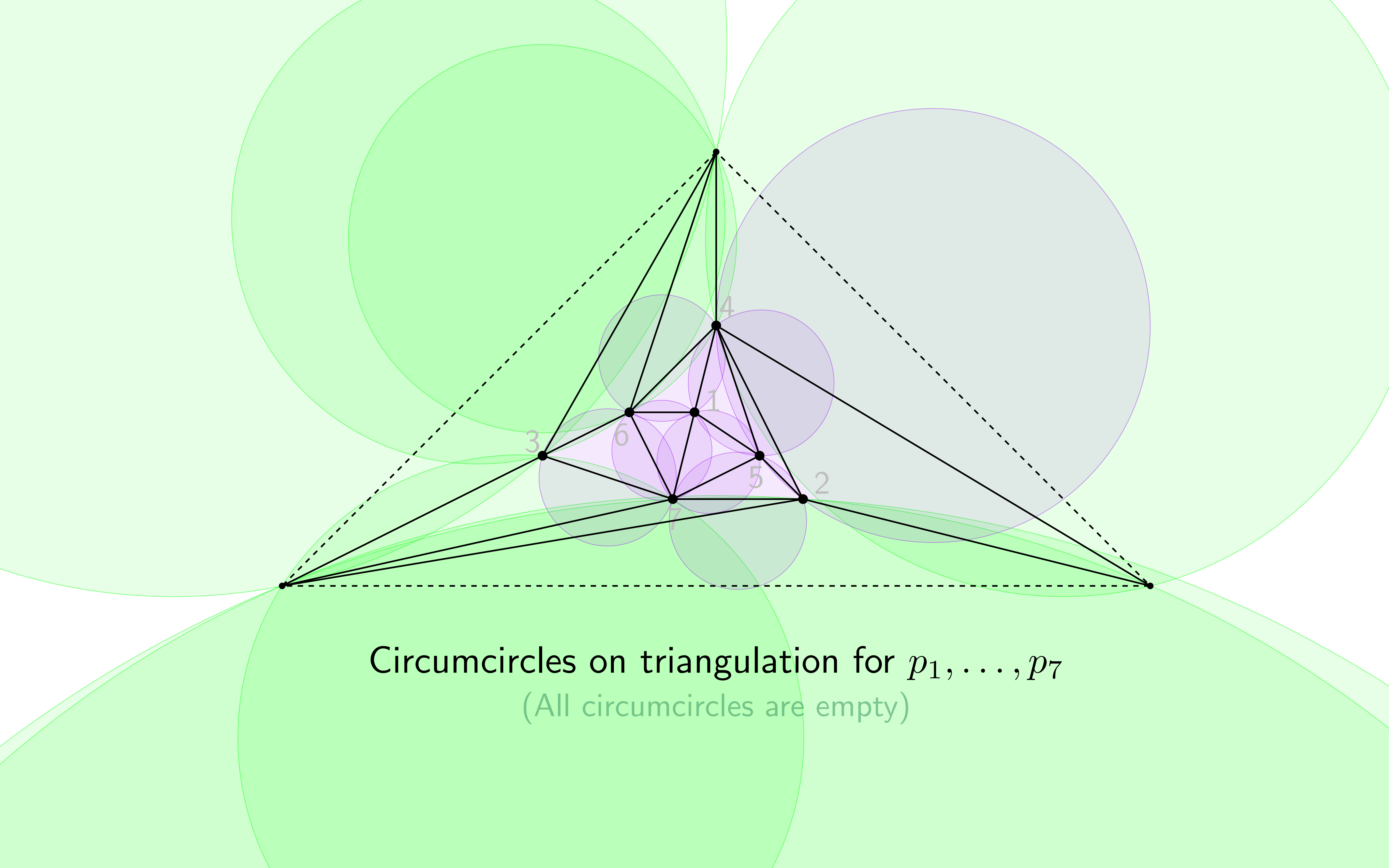


Circumcircles on triangulation for $p_1, \ldots, p_7$



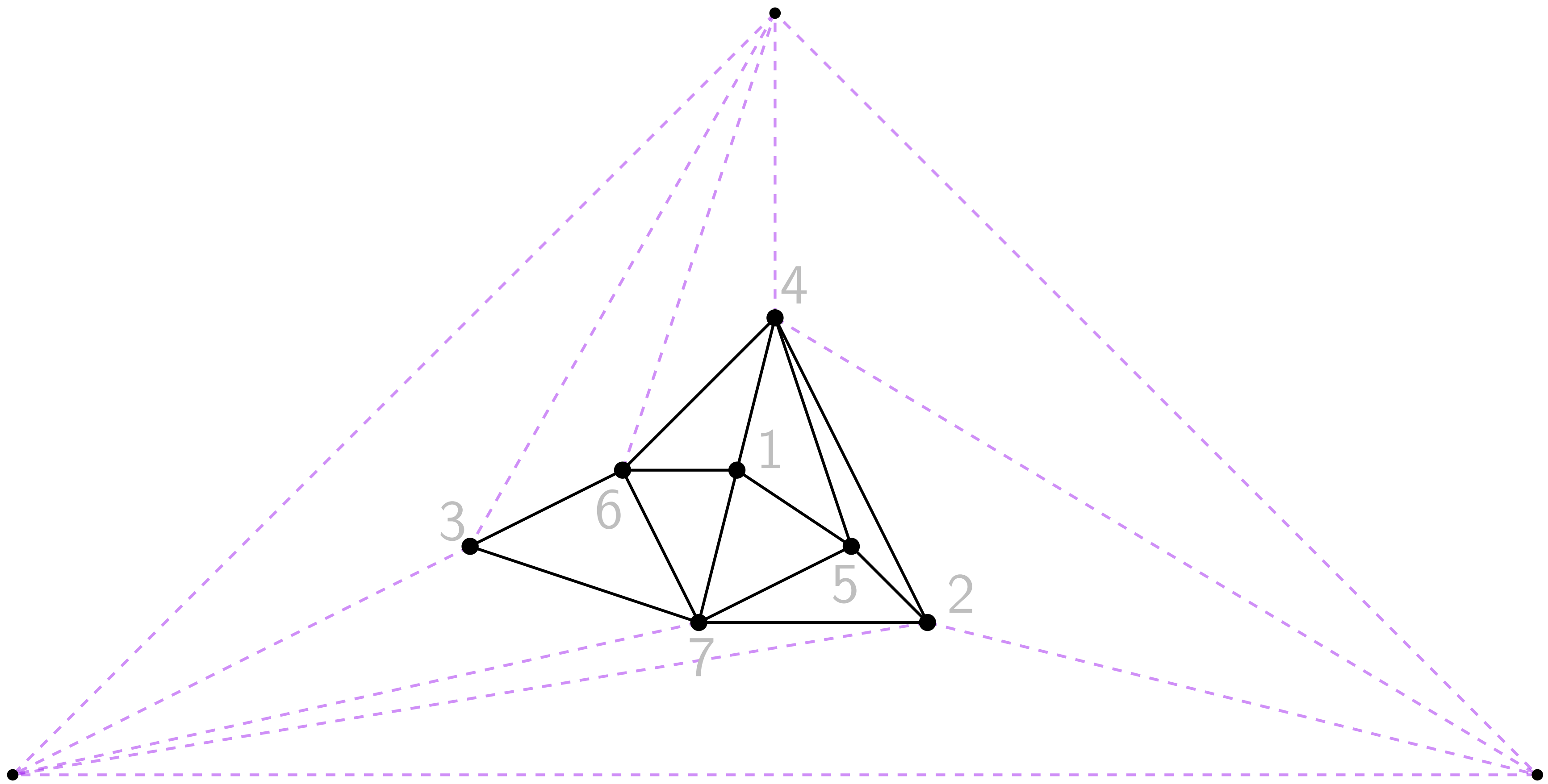
Circumcircles on triangulation for $p_1, \ldots, p_7$

(All circumcircles are empty)

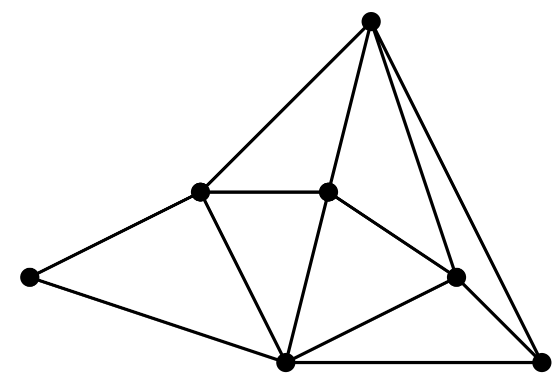


Circumcircles on triangulation for $p_1, \ldots, p_7$

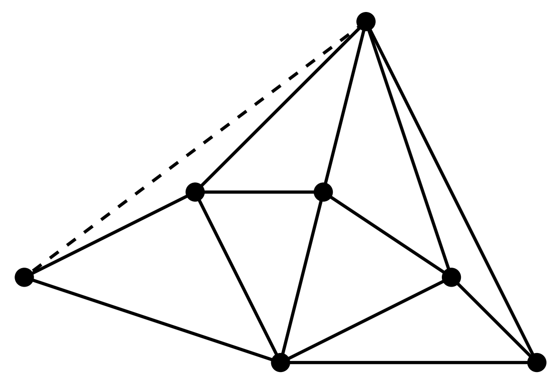
(All circumcircles are empty)



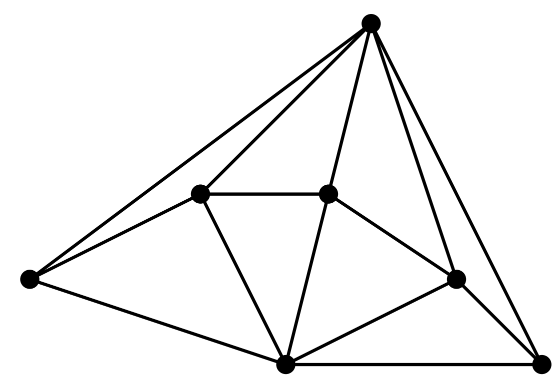
Almost final triangulation (with extra edges & 3 extra sites)



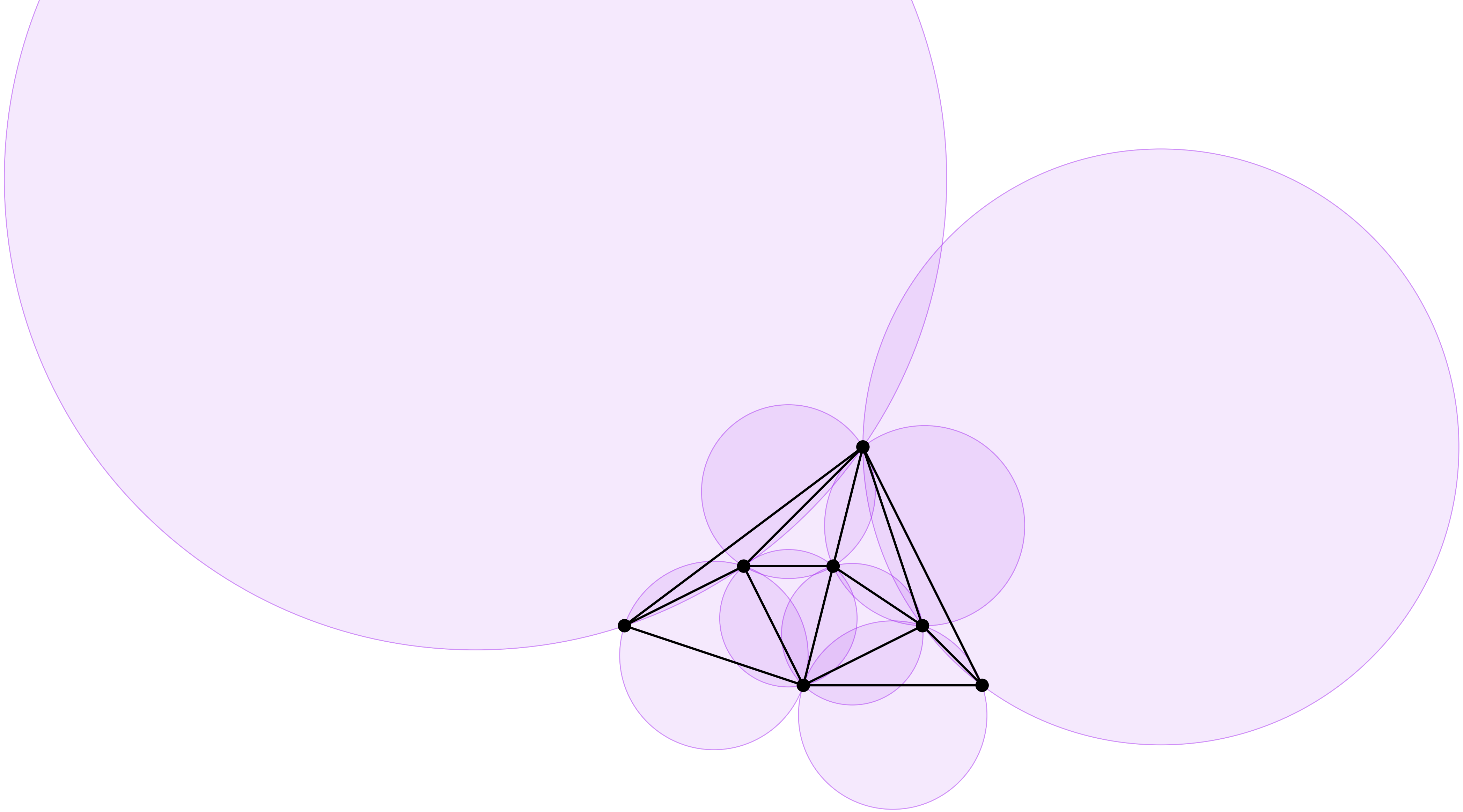
After removing extra edges



After adding convex-hull edges



Final triangulation

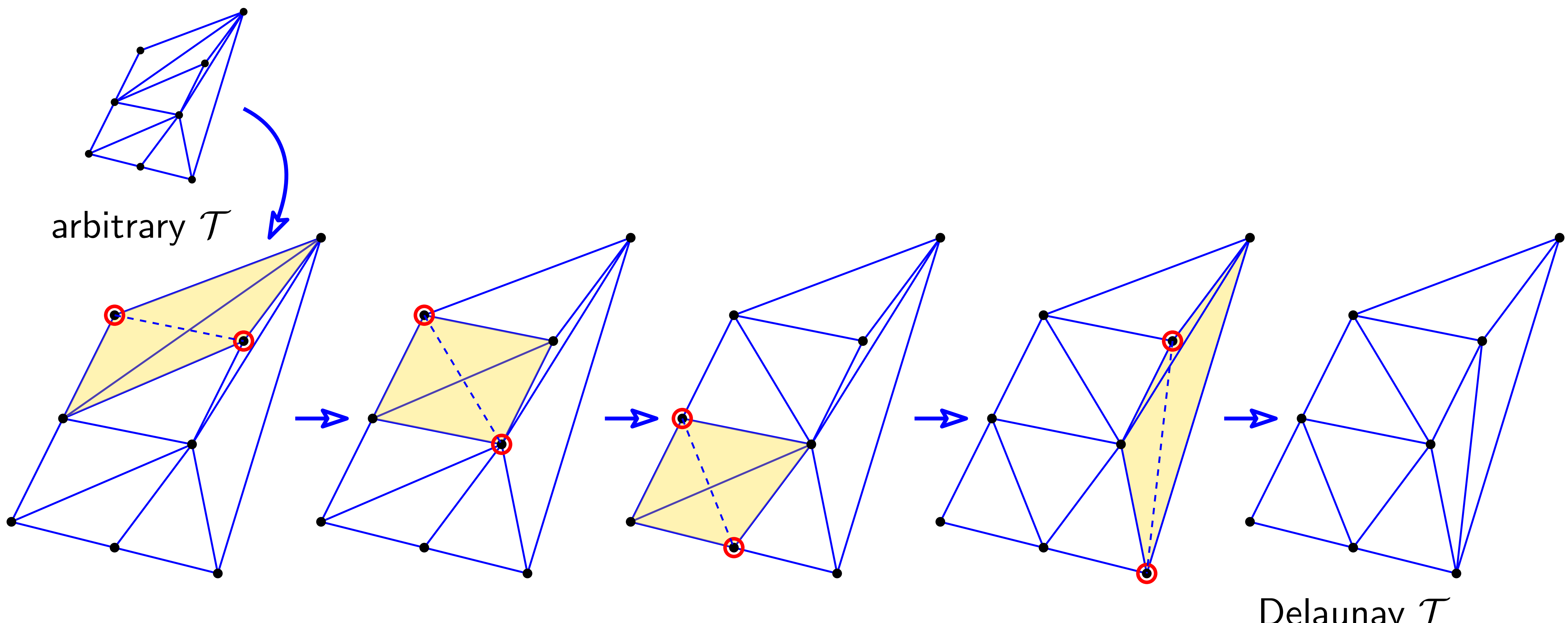


Final triangulation

(All circumcircles are empty)

Algorithms for DT

3. Using only edge-flips iteratively (greedy)

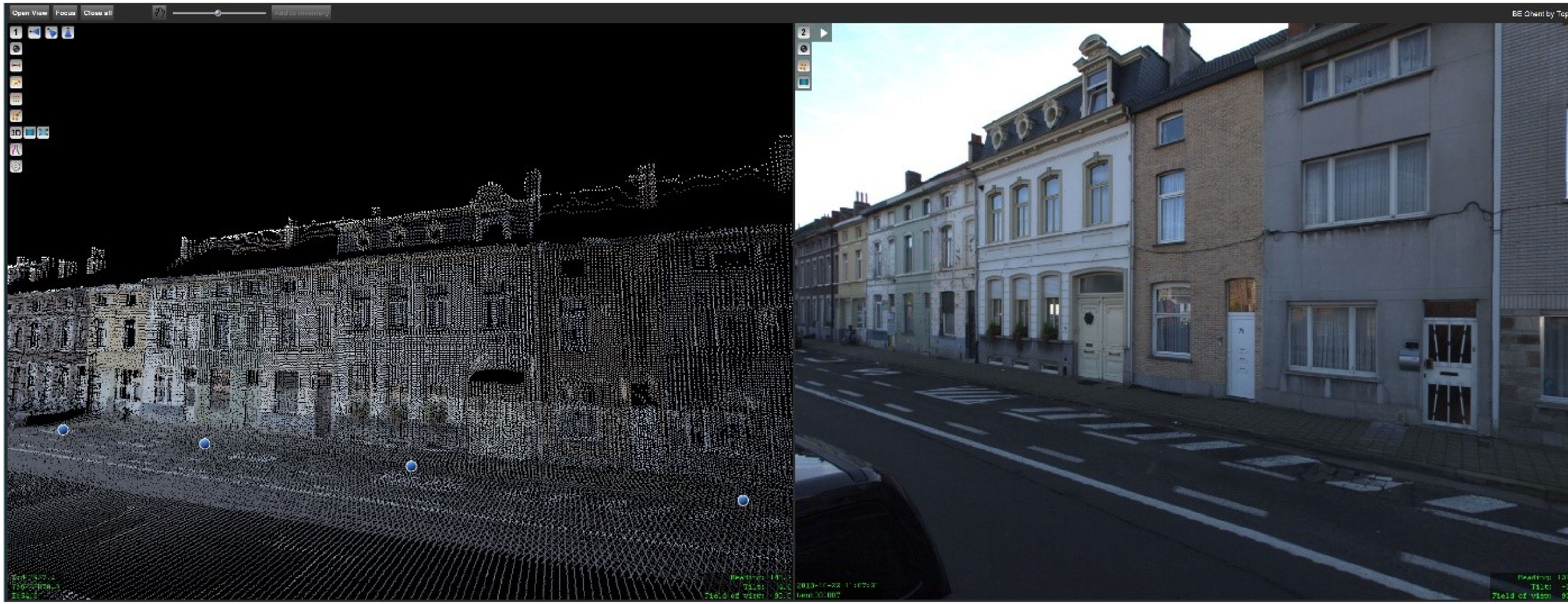


Applications for DT

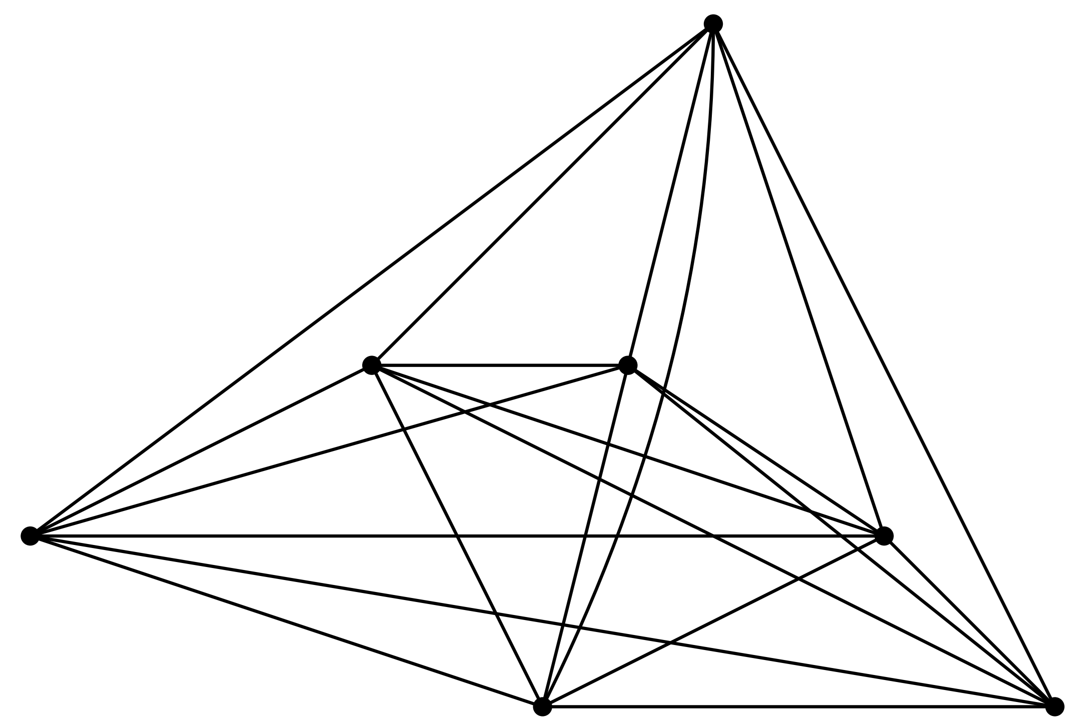
1. 3D map / 3D terrain modeling from discrete points
2. 3D surface reconstruction
3. Euclidean minimum spanning tree of P
4. And many such...

3D map modeling from discrete points

3D surface reconstruction



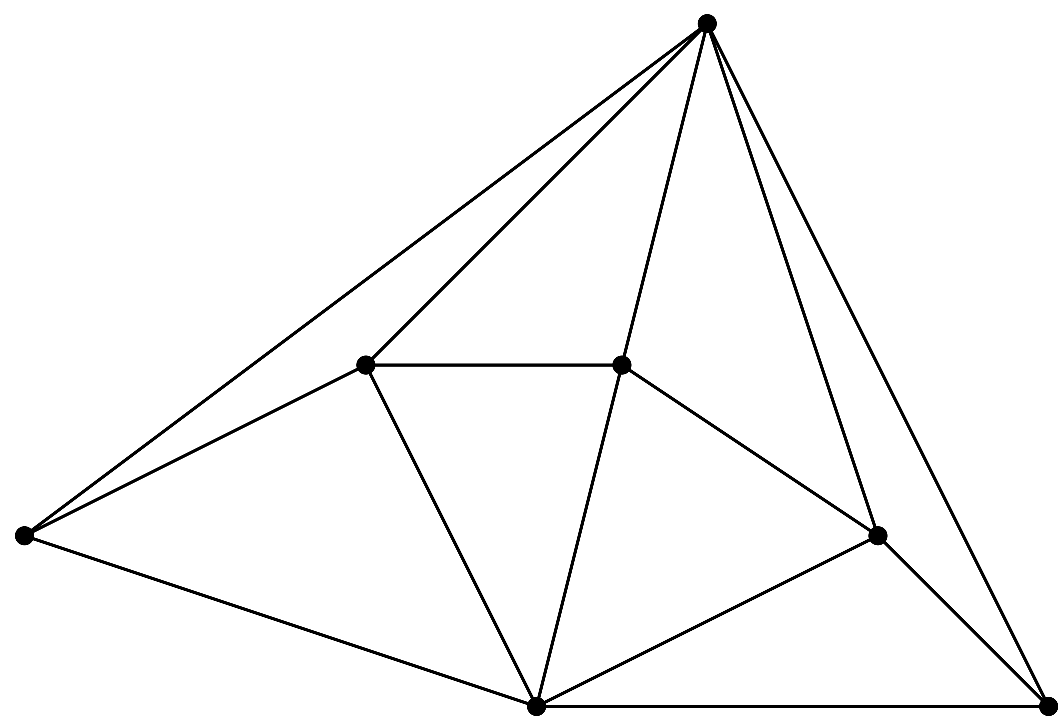
Euclidean MST



Given: n sites, $\binom{n}{2}$ pairwise distances

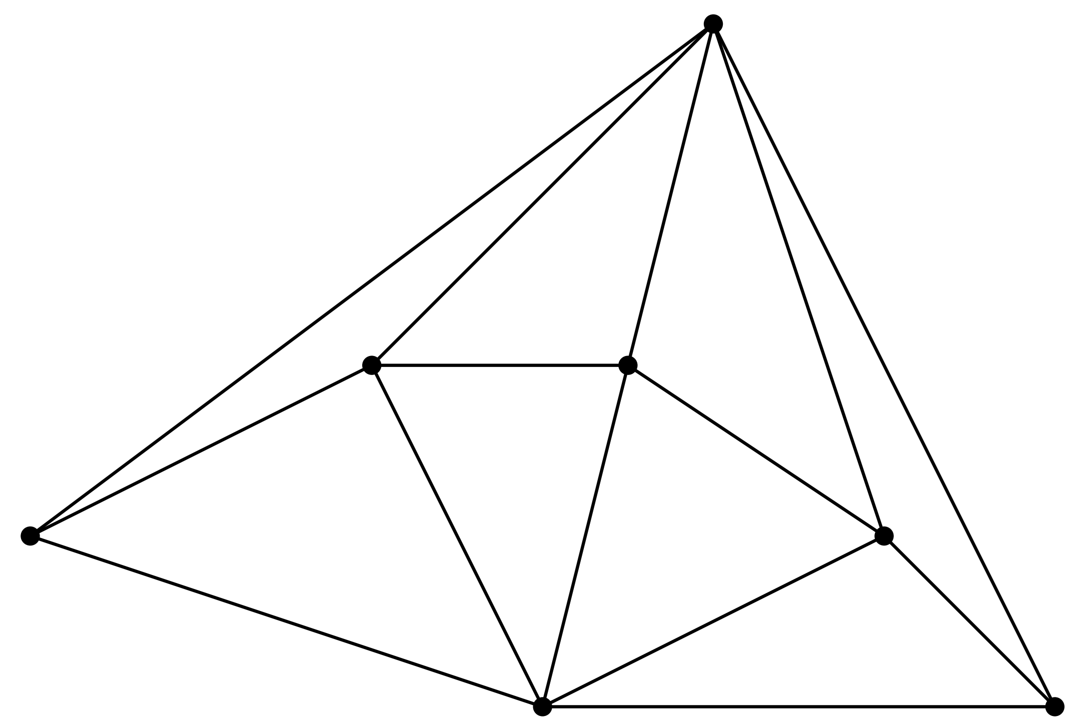
To find: MST connecting these sites

Euclidean MST



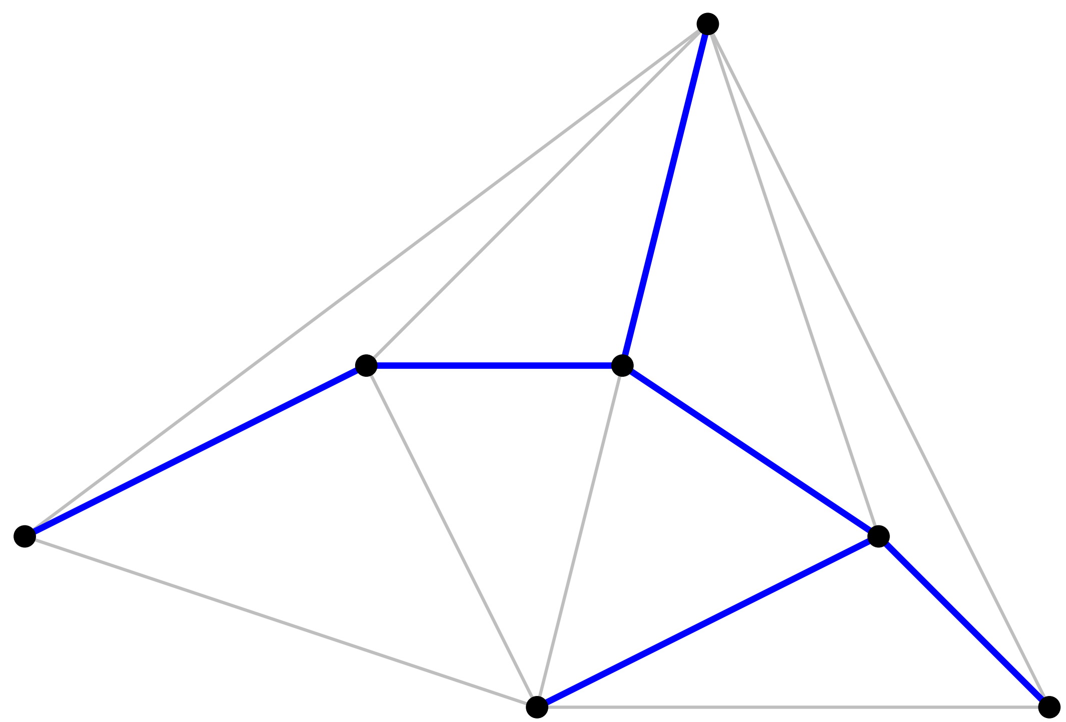
1. Compute DT

Euclidean MST



1. Compute DT
2. Find MST with edges of DT as edges of the graph — $O(n)$ time

Euclidean MST



1. Compute DT
2. Find MST with edges of DT as edges of the graph — $O(n)$ time