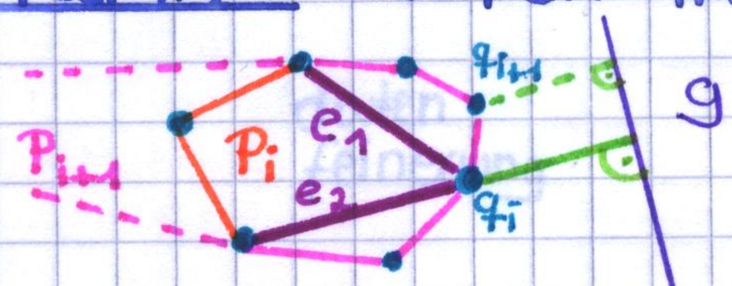
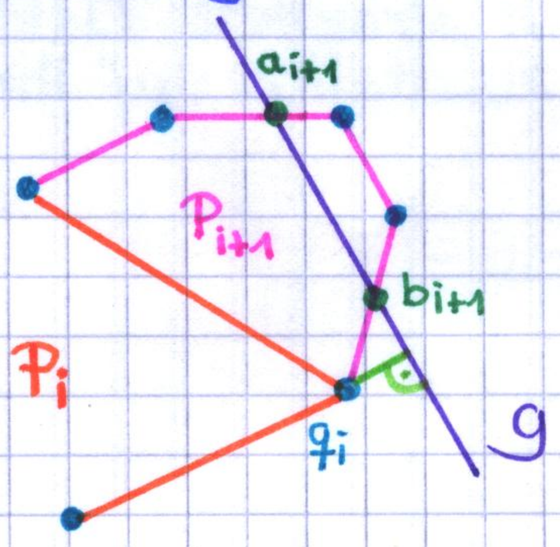
Fall 1.1: nein, dh $P_{i+1} \cap g = \emptyset$



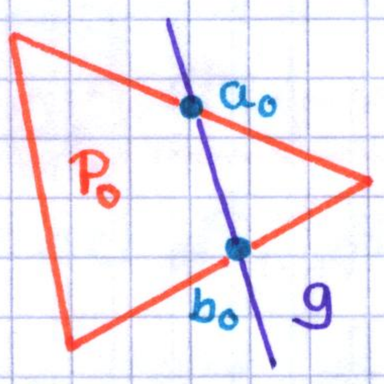
- Finde $g_{i+1} \in P_{i+1}$ mit minimalem Abstand zu g (Minimumssuche auf konst # von Kandidaten)
- goto ①

Fall 1.2: ja, dh.: $P_{i+1} \cap g \neq \emptyset$ $P_{i+1} \cap g = (a_{i+1}, b_{i+1})$



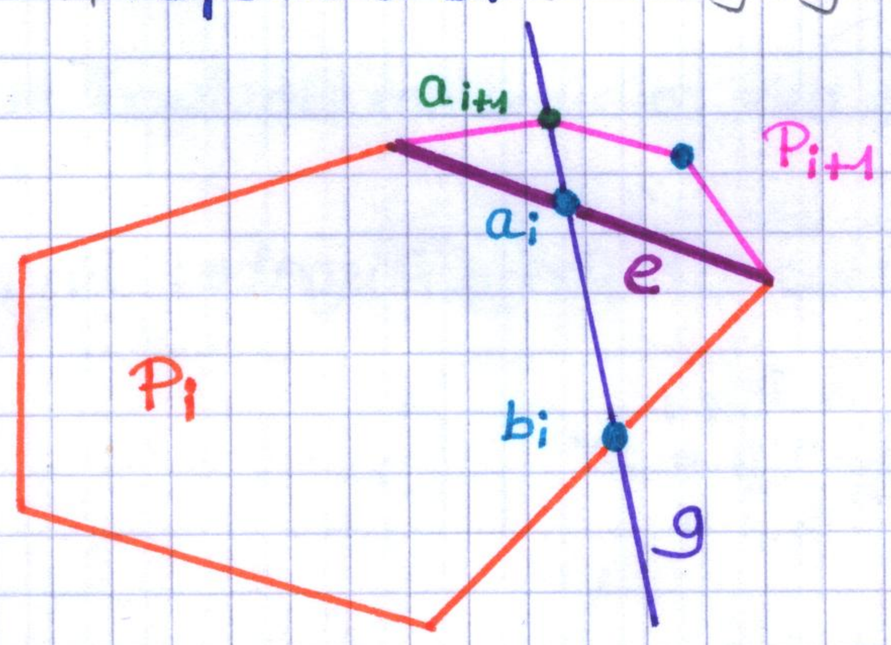
- $P_i \leftarrow P_{i+1}$ dh wir benennen P_{i+1} jetzt P_i (bedeutet keinen "Rückgang" der Verfeinerung !!)
- goto ②

Fall 2: $P_0 \cap g \neq \emptyset$ $P_0 \cap g = (a_0, b_0)$



- i=0

② • if i==k then (dh if $P_i == P$)
  Algorithmus ist fertig. Ausgabe: $P \cap g = (a_k, b_k)$
  end if
- if a_i== Ecke von P_i then
  $a_k = a_i$
  else
  (*) verfeinere a_i dh Übergang $a_i \rightarrow a_{i+1}$



(*) In diesen beiden Fällen muss a_{i+1} (b_{i+1}) Schnittpkt mit einer Verfeinerungskante von e sein wobei $a_i = e \cap g$ ($b_i = e \cap g$) oder Eckpkt von 2 Verfeinerungskanten von e
⇒ a_{i+1} (b_{i+1}) kann in Zeit $O(1)$ aus a_i (b_i) berechnet werden. Da es $O(\log n)$ Stufen gibt, benötigt diese Berechnung von a_i (b_i) Zeit $O(\log n)$

  end if
- if b_i== Ecke von P_i then
  $b_k = b_i$
  else
  (*) verfeinere b_i dh Übergang $b_i \rightarrow b_{i+1}$
  end if

if (a_k==a_i und b_k=b_i) then → • $P_i \leftarrow P_{i+1}$ dh wir benennen P_{i+1} jetzt P_i (bedeutet keinen "Rückgang" der Verfeinerung !!)
  Ausgabe $P \cap g = (a_k, b_k)$ • goto ②
endif