INF 4130, 17. november 2011

kap. 8.6 ”Computational Geometry”

Stein Krogdahl

Hovedkapittelet (kap. 8) dreier seg generelt om

“devide-and-conquer” eller “splitt og hersk”:

- Splitt problemet opp i mindre problemer.
- Løs hvert av de mindre problemene (kanskje rekursivt ved ytterligere oppdelinger?)
- Finn løsningen på det store problemet ved å bruke løsningene vi fant på de mindre problemene

- Vi skal se på det å finne den konvekse innhylling av en punktmengde, 8.6.2

- Også triangulering kan løses greit v.h.a. splitt og hersk

Geometriske algoritmer
"Computational Geometry"

- Feltet dreier seg om algoritmer som løser problemer i det vanlige 2- eller 3-dimensjonale (eller n-dimensjonale) rom
- Vi skal her bare ha 2 dimensjoner. Bruksområder:
  - Mønstergjenkjening
  - Datagrafikk
  - Grafiske informasjonssystemer GIS
  - CAD/CAM (DAK/DAP)
  - Robotikk
  - VLSI-design
  - Og ikke minst: Triangulering
- *Eksempler vi skal se på:*

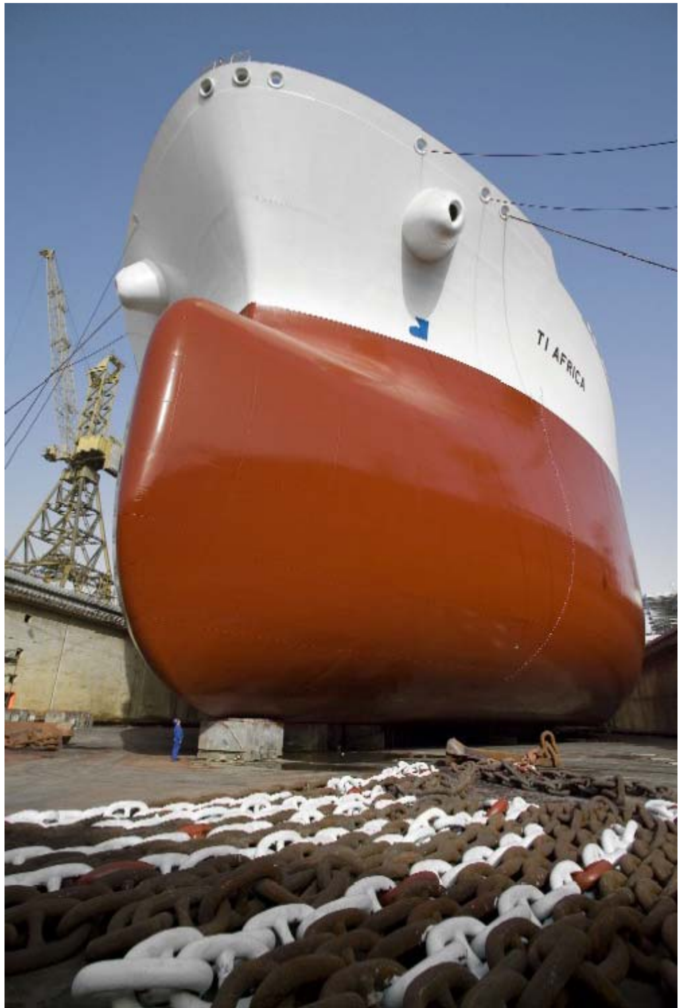
  Gitt en mengde punkter i planet:
  - Finn convex hull = Konveks innhylling (se neste foil)
  - Finn det par av punkter som har minst avstand

"Convex hull"

Kunne oversettes til
"det konvekse hull"
av en punktmengde.

Men det er ikke et *sånt* hull!

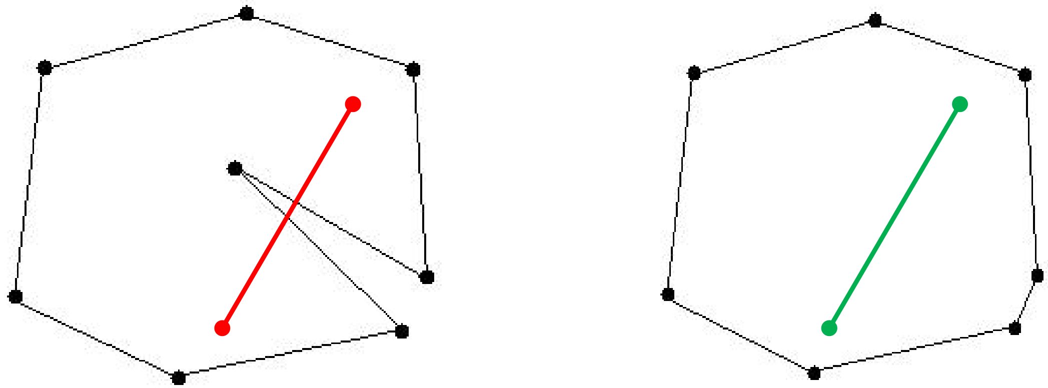
Det er et *sånt* hull = skrog
Vi kaller det derfor her *”den konvekse innhylling”*



Konveks innhylling

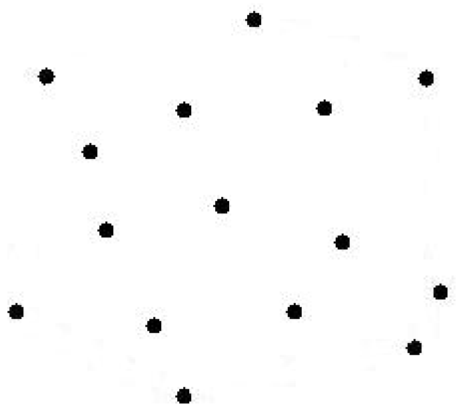
La P være en mengde punkter i et k-dimensjonalt rom, $P \in \mathbf{R}^k$.
(Vi skal for enkelthets skyld bare se på k = 2.)

Definisjon En mengde $Q \in \mathbf{R}^k$ er *konveks* dersom:
for alle punkter $q_1, q_2 \in Q$ linjesegmentet $\overline{q_1 q_2}$ ligger i Q.

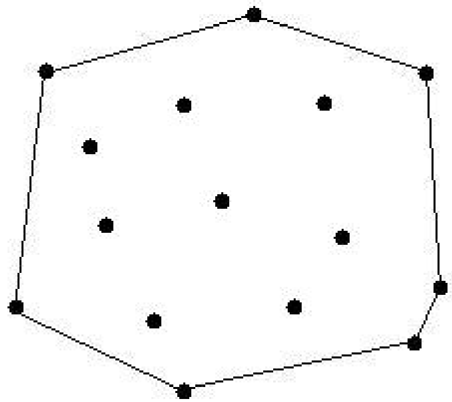


Definisjon Den *konvekse innhyllingen* til en mengde punkter $P \in \mathbf{R}^k$ er den minste konvekse mengde Q som inneholder punktene i P.

Utgangspunkt: En punktmengde i planet

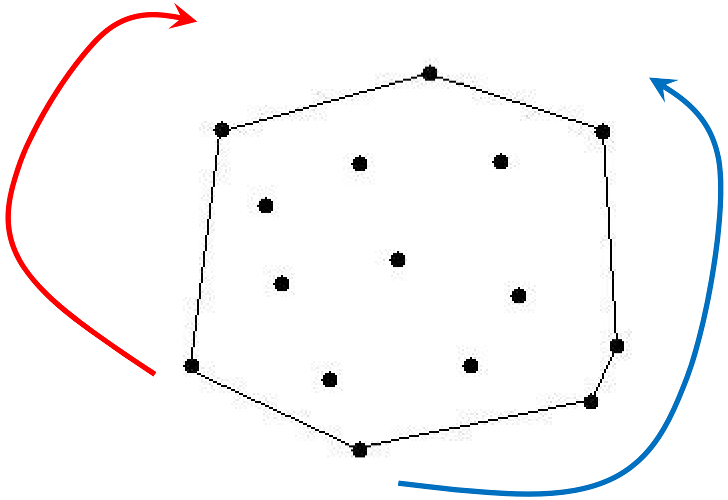


Konveks innhylling av en punktmengden



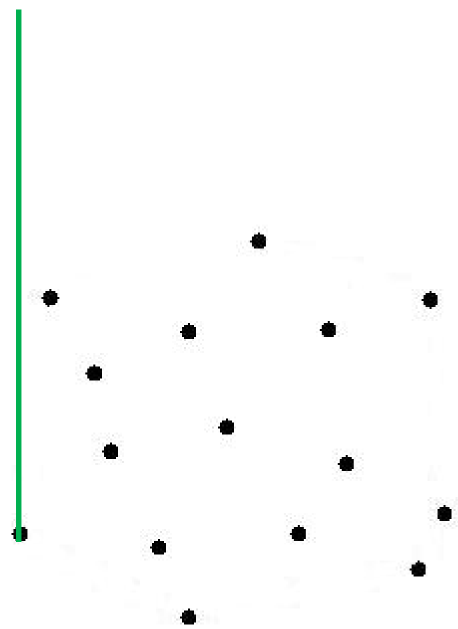
Bokas intuisjon:
Punktene er spiker, og vi legger en strikk rundt

Svaret skal komme som punkter (x,y) i *rekkefølge* rundt polygonet,



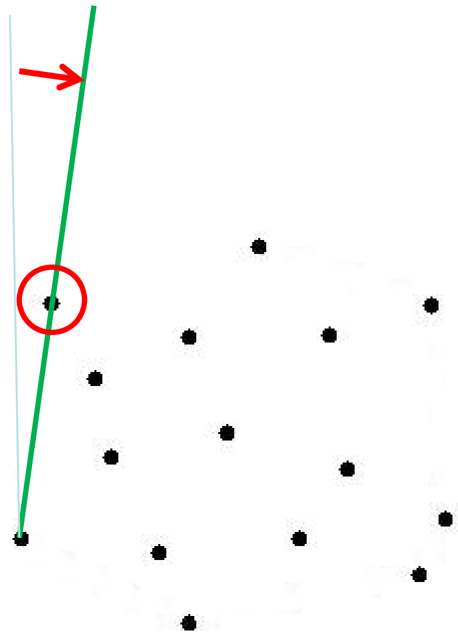
Startpunkt og retning kan variere

Én måte å lage den konveks innhyllingen på:
Kalles “Jarvis’ March”



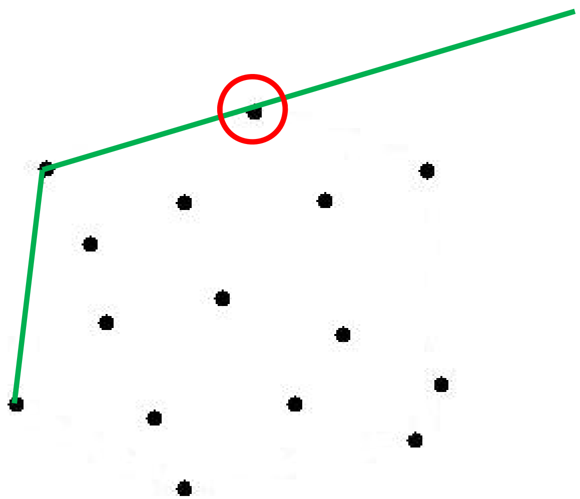
- Idé: “Snurr” en hyssing stegvis rundt punktene
- Start f.eks. med punktet med minst x-verdi (er alltid på den konvekse innhyllingen!?).

Første skritt



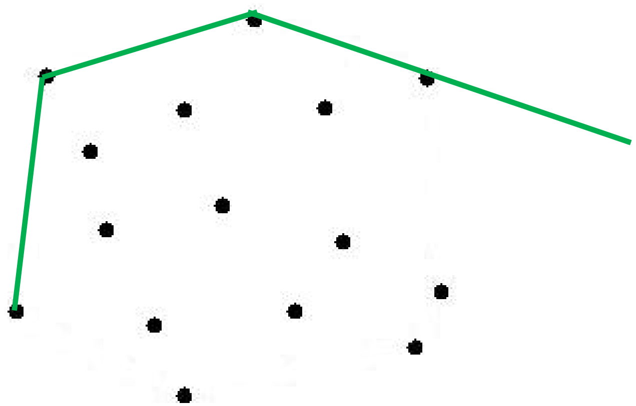
- Sving ”tråden” mot høyre, og stopp ved første berøring med et punkt
- Dette er neste punkt på innhyllingen

Videre:

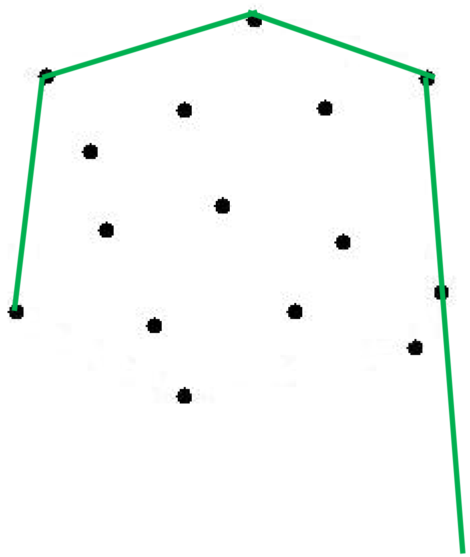


- Snurr så videre, alltid rundt siste punkt man fant, til man berører et nytt punkt.
- Må i hvert steg finne det punktet som det er minst *svingvinkel* til.
- Tar tid O(n) med rett fram algoritme

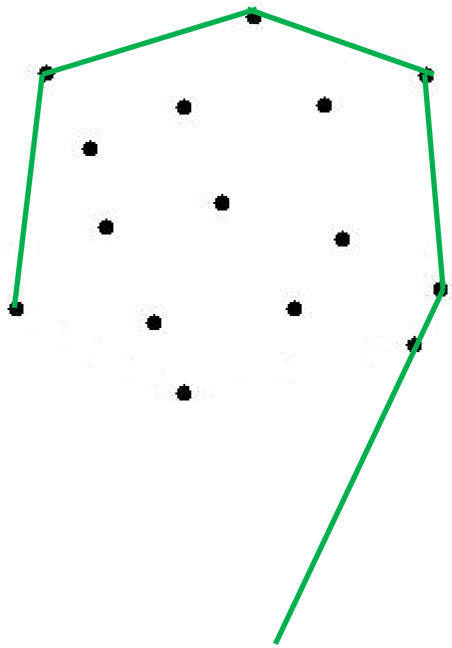
Neste steg:



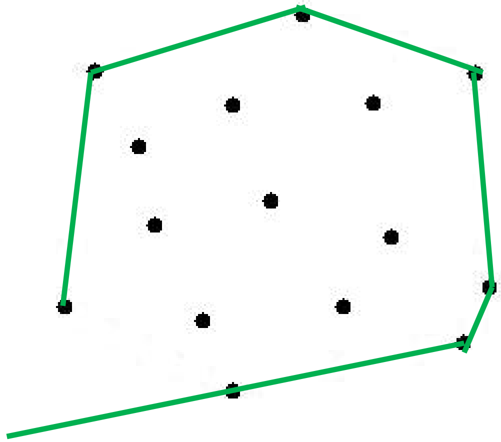
Og neste:



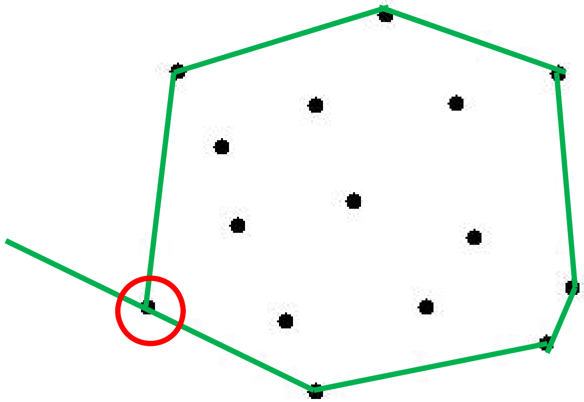
Og neste:



Og neste:

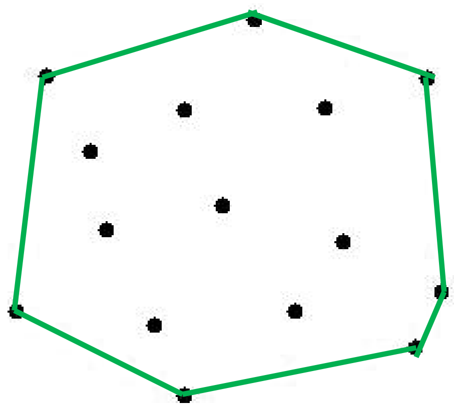


Avslutning:



Når man finner startpunktet som neste punkt

Og altså: Den konvekse innhylling:

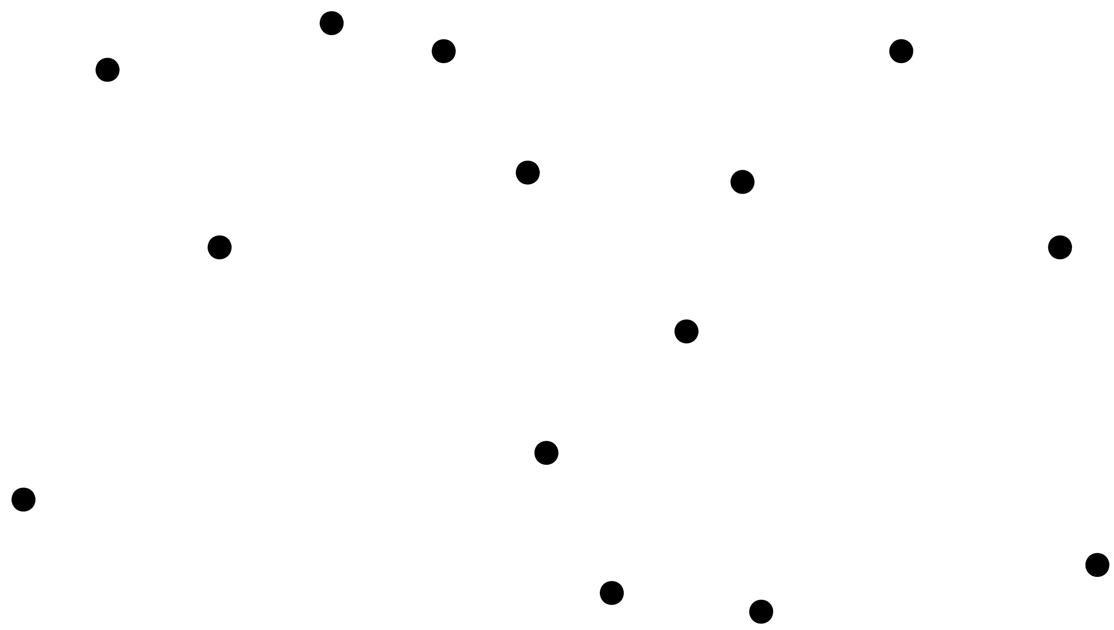


Og Jarvis' march" har marsjert ferdig

- Har i to dimensjoner worst case tid: $O(n^2)$
- I *d* dimensjoner er w-c-tiden: $O(n^{\text{floor}(d/2)+1})$

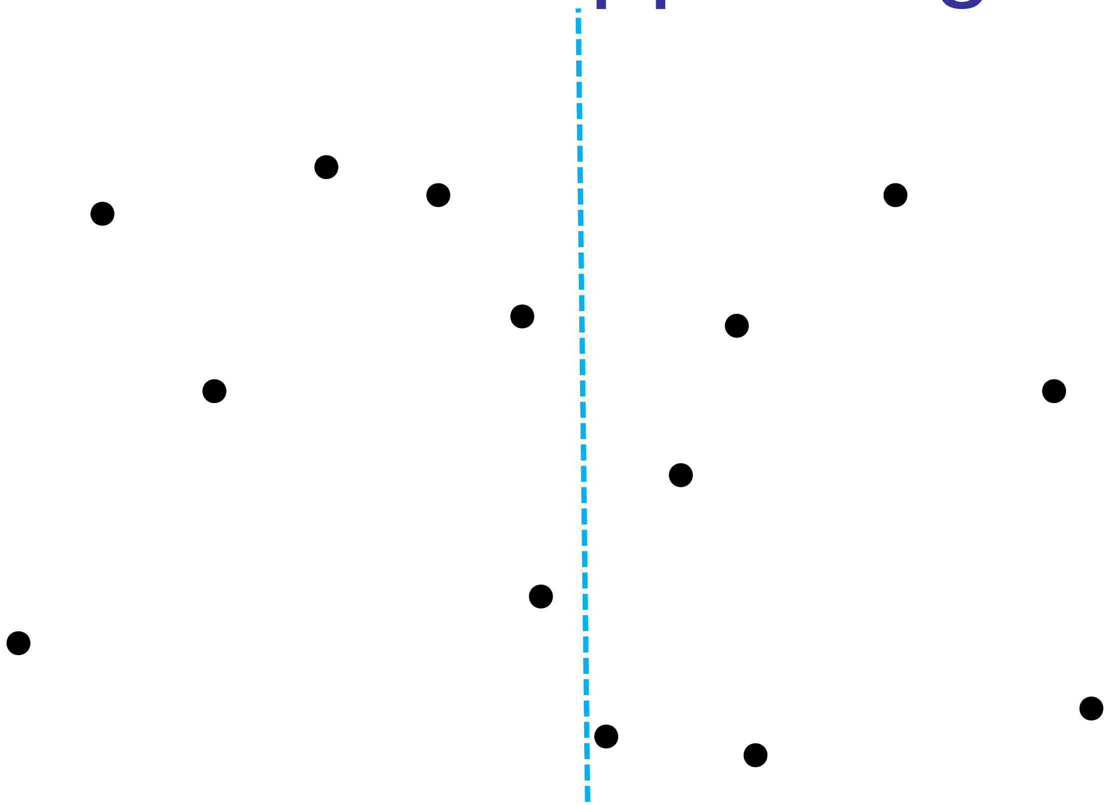
En annen metode for å finne konveks innhylling.

Denne bruker ”splitt og hersk”-teknikken



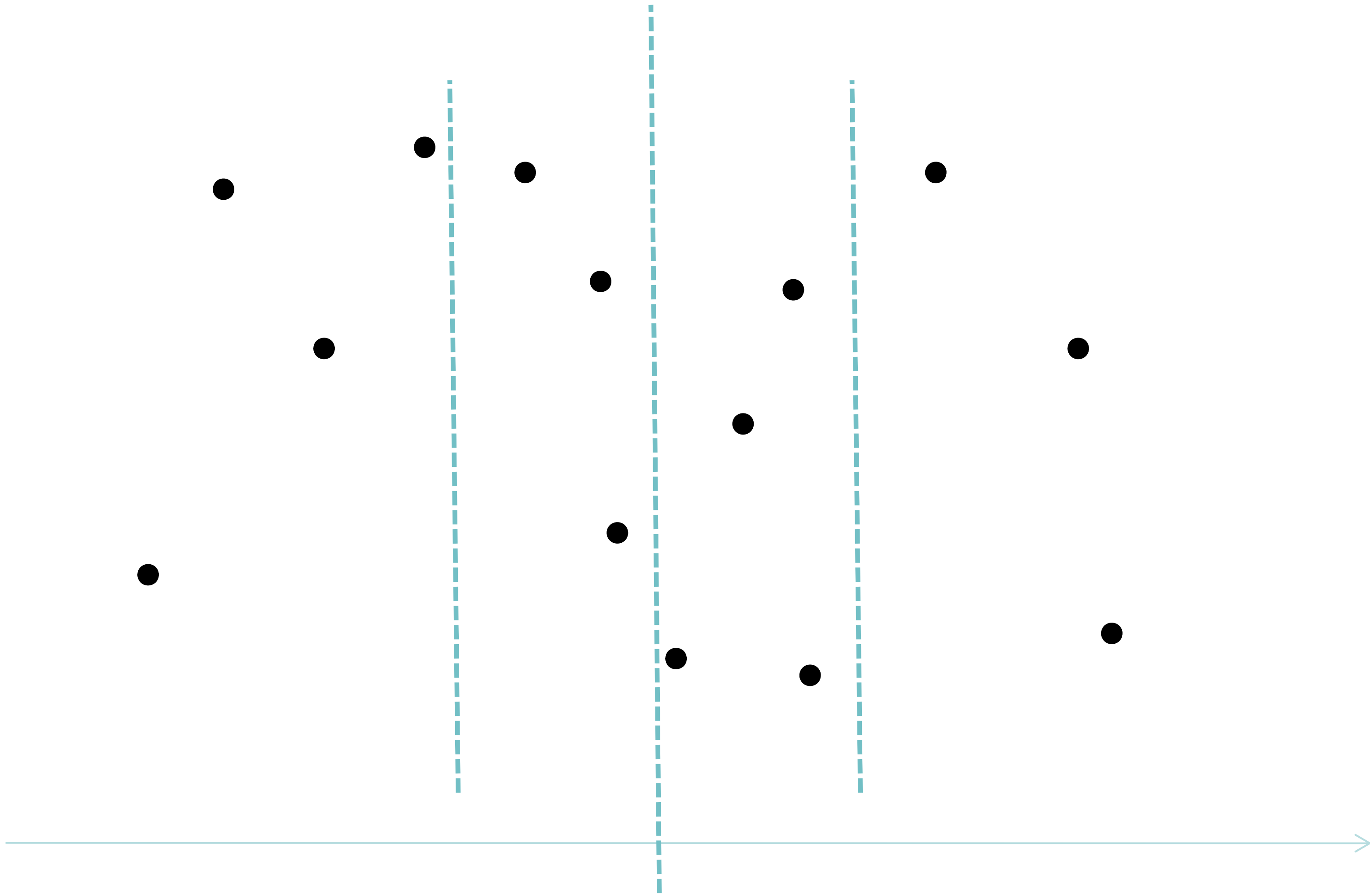
Deler først punktene opp i to ”like store” (modulo odde tall) mengder vha. vertikale linjer

Første oppdeling



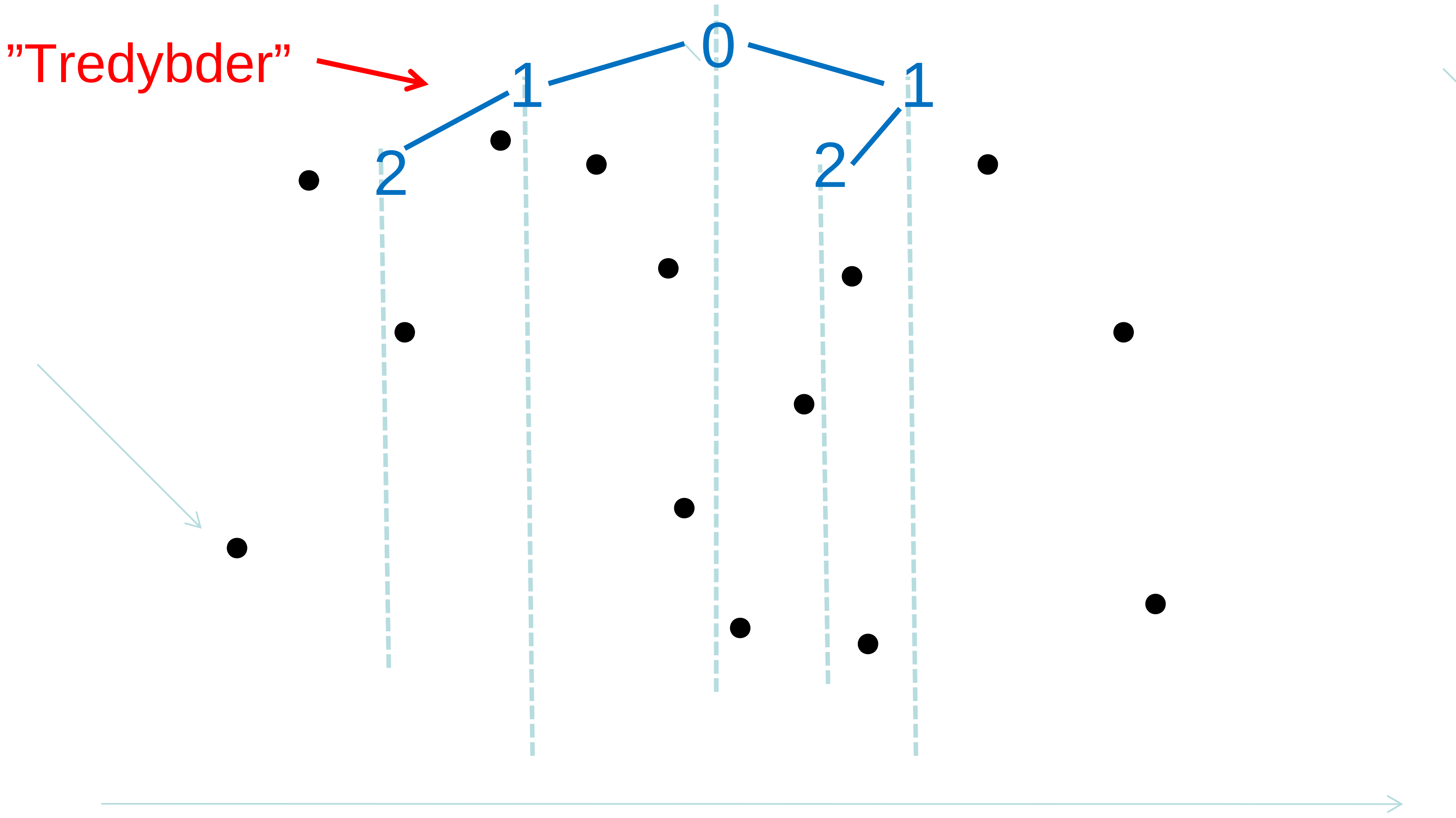
- Deler mengden i to ved *x*-medianen (kan finnes i lineær tid, men ikke viktig)
- Gjentar dette for hver mengde helt til vi får 1, 2 eller 3 punkter i hver av mengdene.

Konveks innhylling



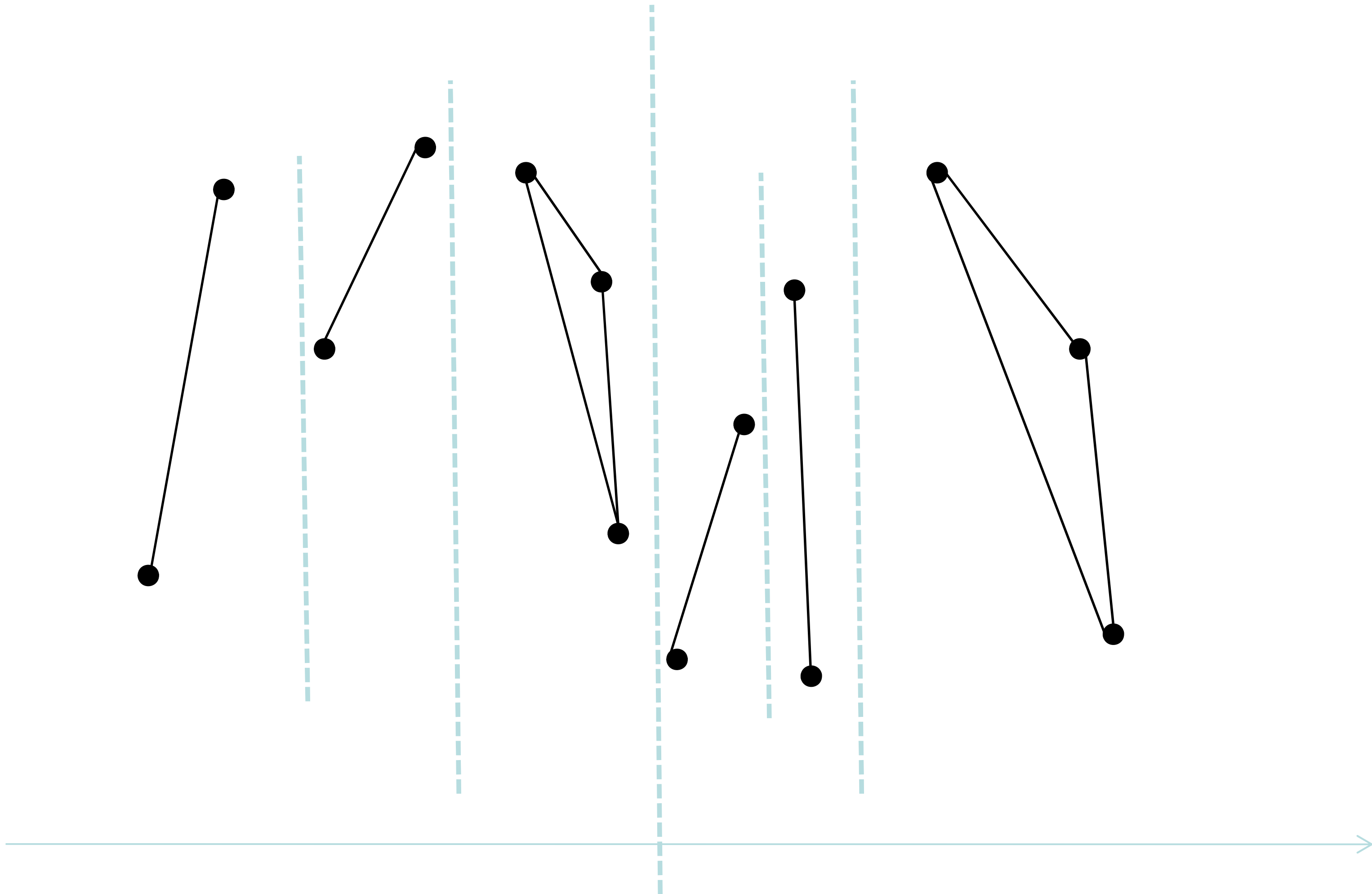
Deler disse mengdene igjen i to ved *x*-median,
helt til vi får 2 eller 3 punkter i mengdene.

Ferdig oppdeling



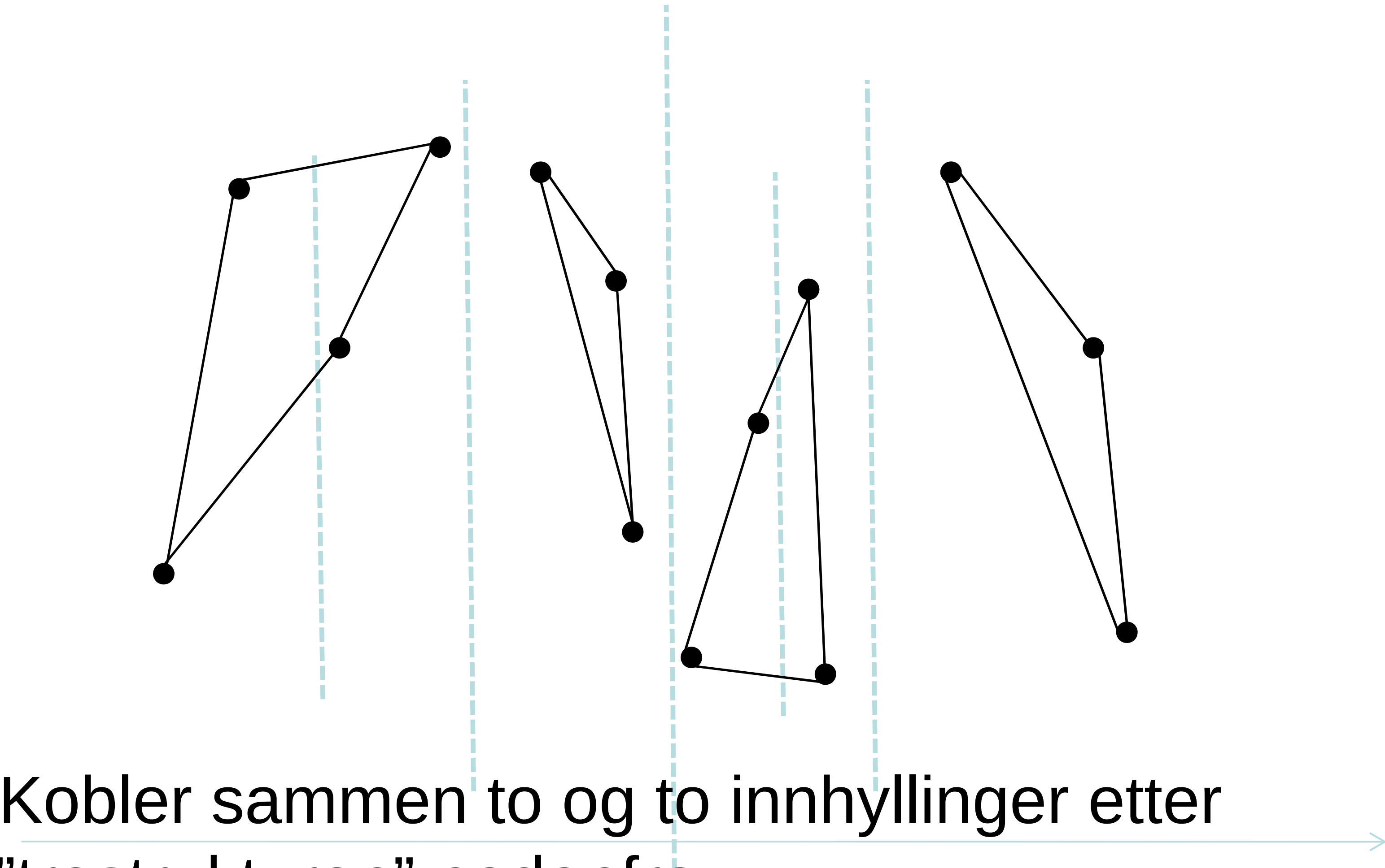
”Dybden” i oppdelingen blir ikke større enn $\log_2 n$

Løser det dypeste nivået



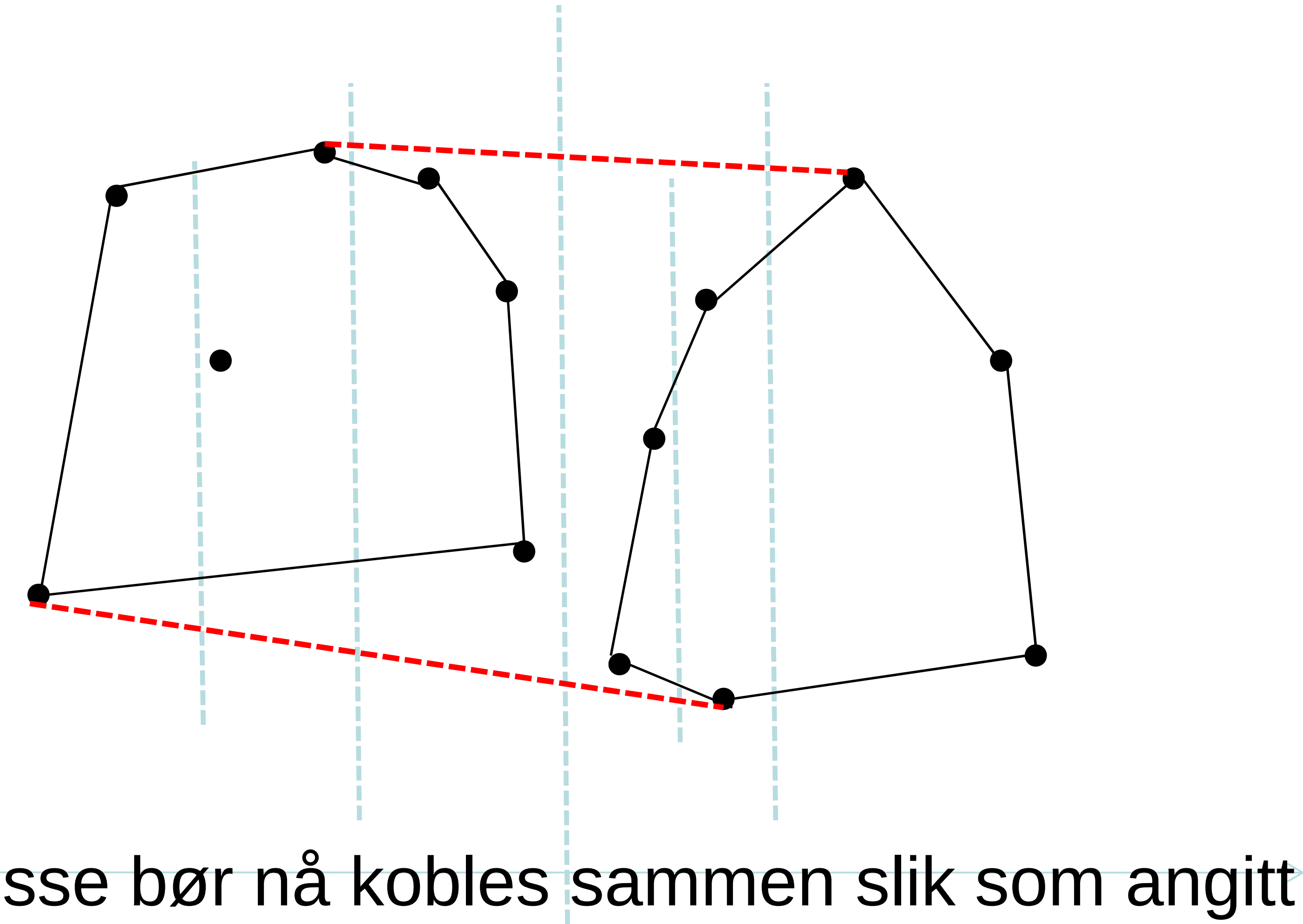
Lett: Finner innhyllinger for 2 eller 3 punkter

Kobler sammen igjen



- Kobler sammen to og to innhyllinger etter ”trestrukturen” nedenfra
- Metoden for dette kommer om et øyeblikk!

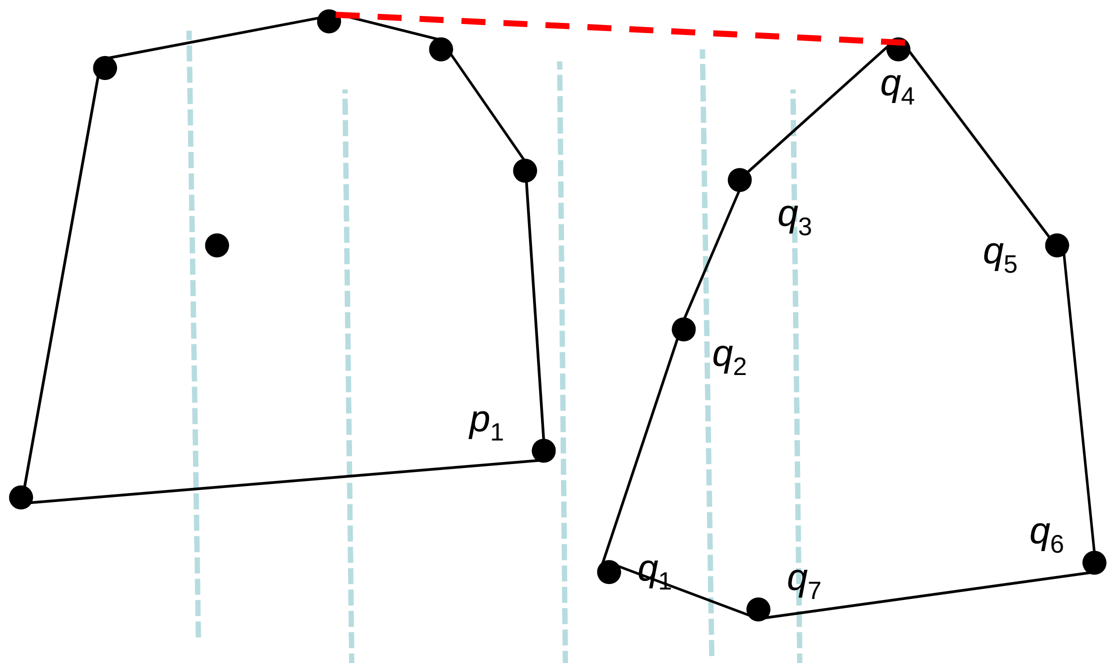
Fram til siste sammenkobling:



- Disse bør nå kobles sammen slik som angitt
- Hvordan finne disse ”broene”?

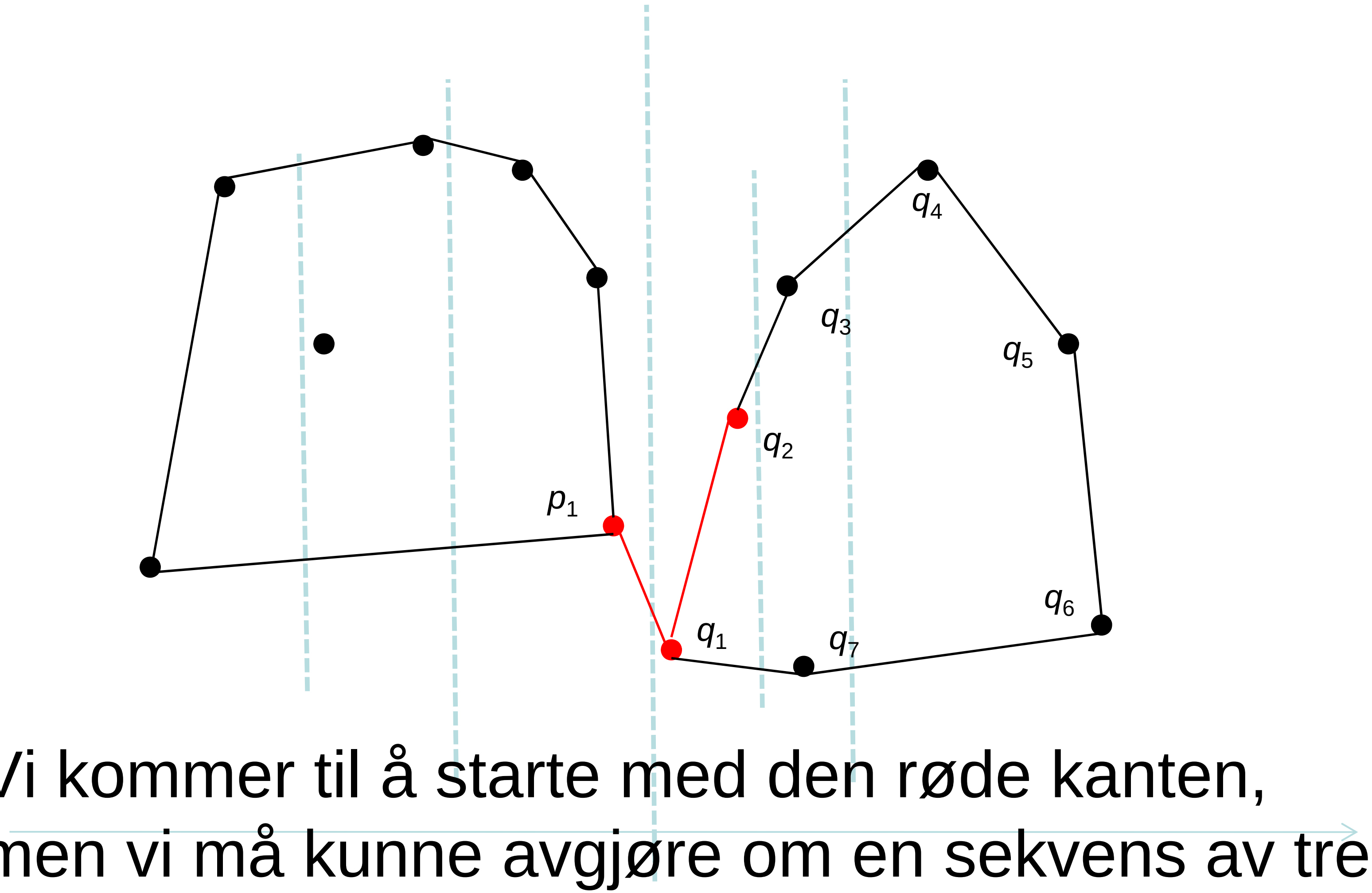
Sammenslåing av to konvekse innhyllinger:

Hvordan finne ”øvre bro”?



- Vet at alle x_i i den ene er mindre enn alle x_j i den andre.
- Antar at venstre innhylling er nummerert *mot* klokka og at den til høyre er nummerert *med* klokka
- La p1 være lengst til høyre (langs x-aksen) i sin innhylling, og q1 lengst til venstre i sin.

Hvordan starte sammensetningen av to?

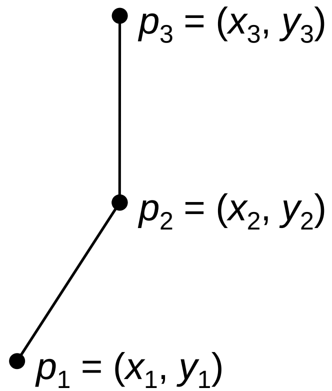


- Vi kommer til å starte med den røde kanten,
- men vi må kunne avgjøre om en sekvens av tre punkter svinger mot høyre eller mot venstre?

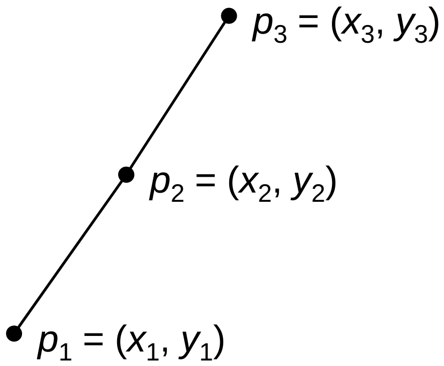
Lite mellomspill:

Hvordan kan vi finne retningsforandringen i punktet $P_2 = (x_2, y_2)$ for punktsekvensen:

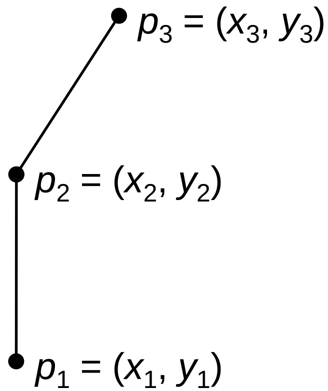
$P_1=(x_1, y_1)$ -> $P_2=(x_2, y_2)$ -> $P_3=(x_3, y_3)$



Venstresving



Rett fram



Høyresving

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} > 0 \qquad \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0 \qquad \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} < 0$$

- Dette er såkalte determinanter
- Bergningen fremgår av neste foil

Hvordan beregne determinanten for 3x3-matriser?

Gitt matrisen *A*:

$$A = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

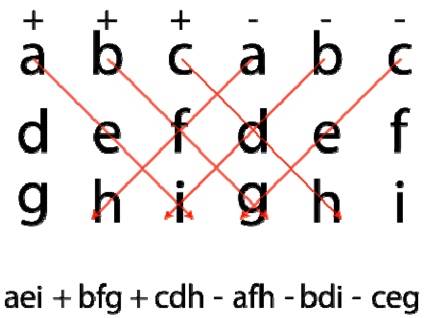
$$\det(A) = a\begin{vmatrix} e & f \\ h & i \end{vmatrix} - b\begin{vmatrix} d & f \\ g & i \end{vmatrix} + c\begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$= aei - afh - bdi + bfg + cdh - ceg$$

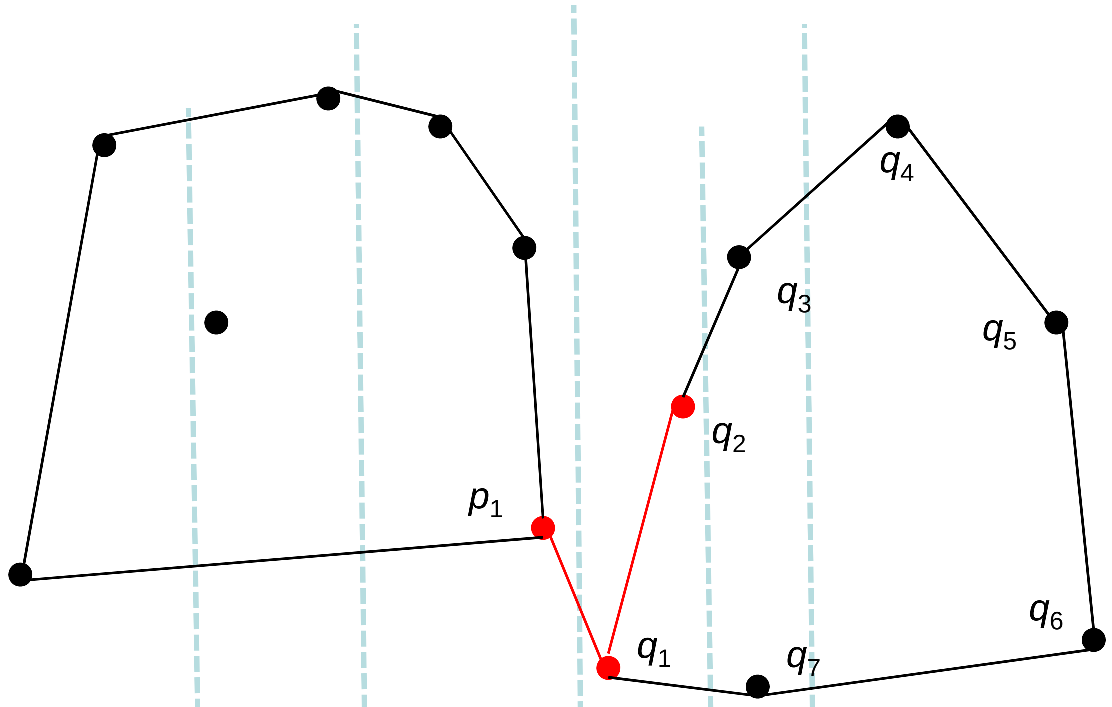
Siden *c, f* og *i* er lik 1 i vårt tilfelle blir dette:

$= ae – ah – bd + bg + dh – eg$

Et morsomt skjema for å beregne 3x3-determinanter:

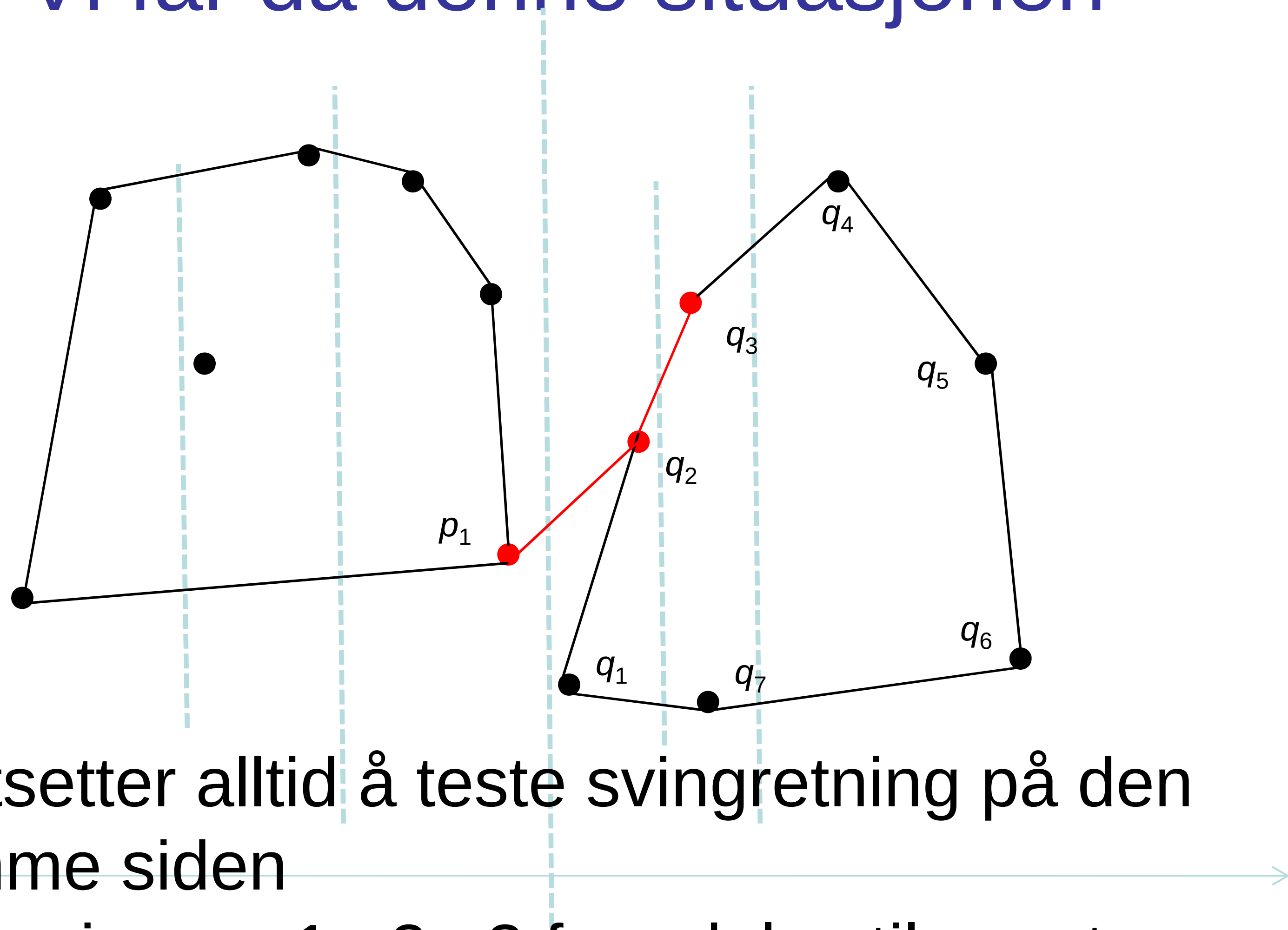


Hvordan starte sammensetningen av to?



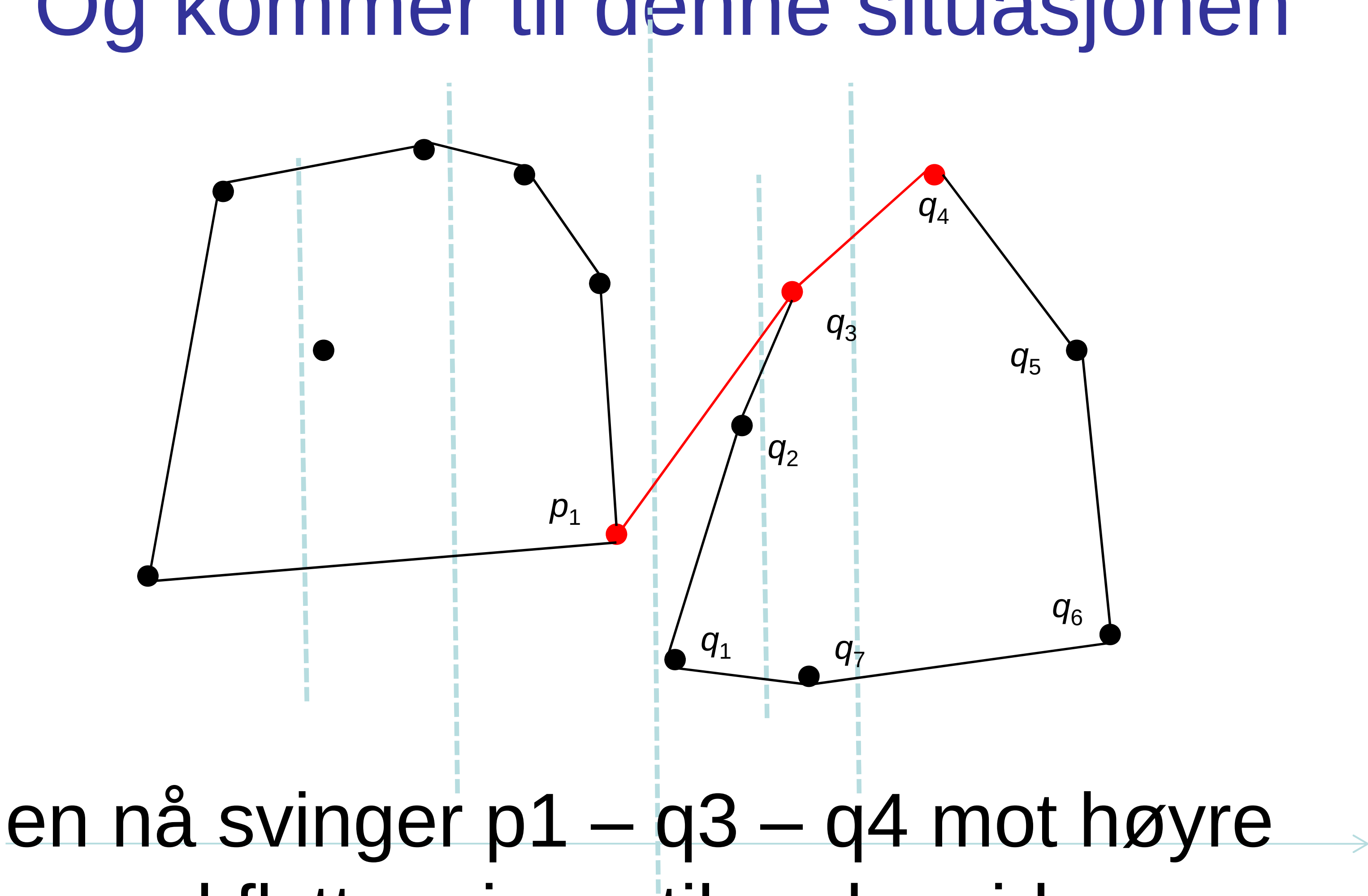
- Altså p_1 er lengst til høyre i sin innhylling, og q_1 lengst til venstre i sin, og vi bruker linjen (p_1, q_1) som utgangspunkt
- Når vi skal finne øvre bro går vi *med* nummereringen i begge polygoner
- Starter med å ”flytte” på en side der p_1 - q_1 - q_2 svinger til venstre eller q_1 - p_1 - p_2 svinger til høyre. Gjelder her begge!

Vi får da denne situasjonen



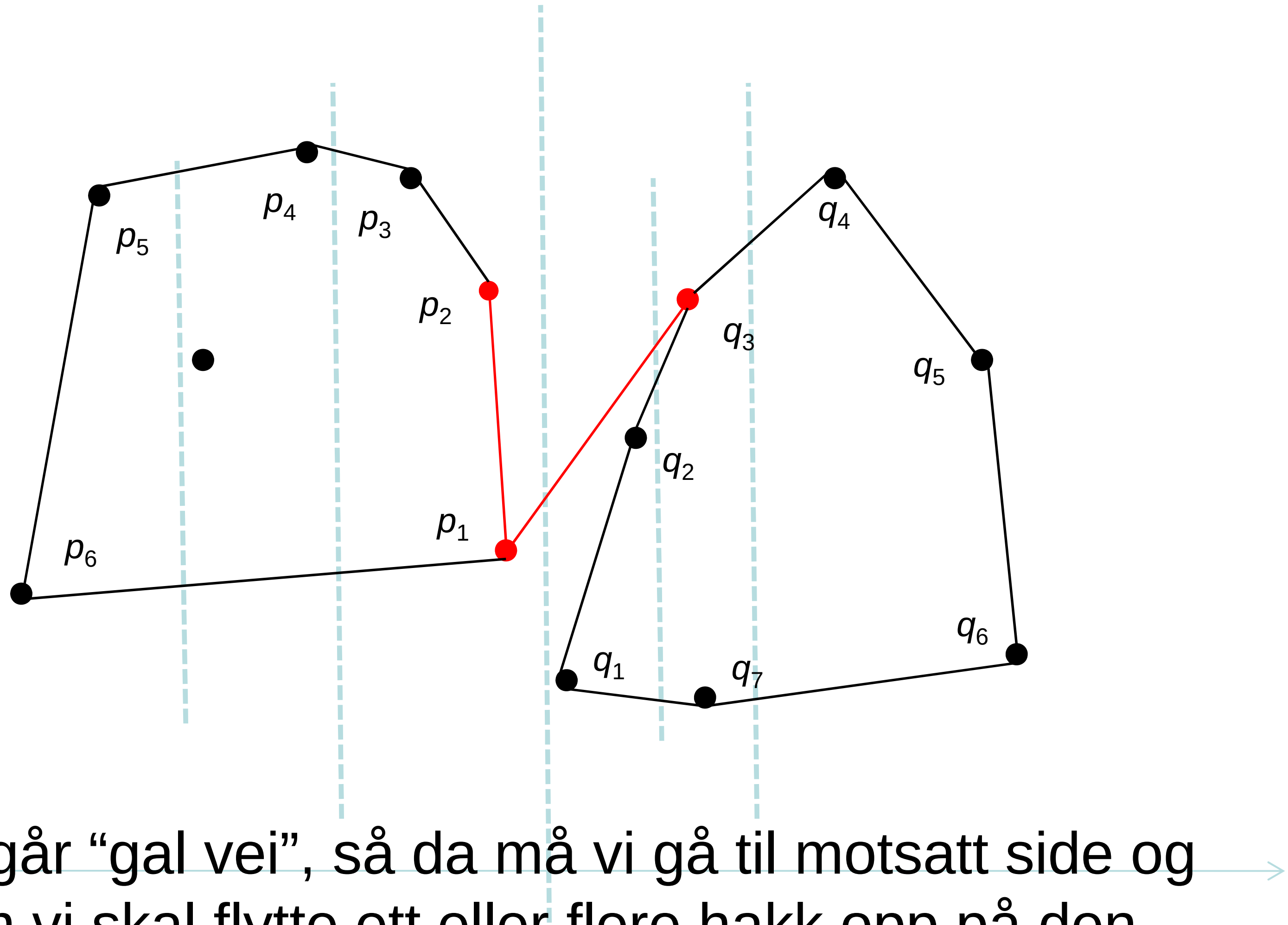
- Fortsetter alltid å teste svingretning på den samme siden
- Her svinger p1 q2 q3 fremdeles til venstre
- Derfor, vi går ett hakk til på den samme siden

Og kommer til denne situasjonen



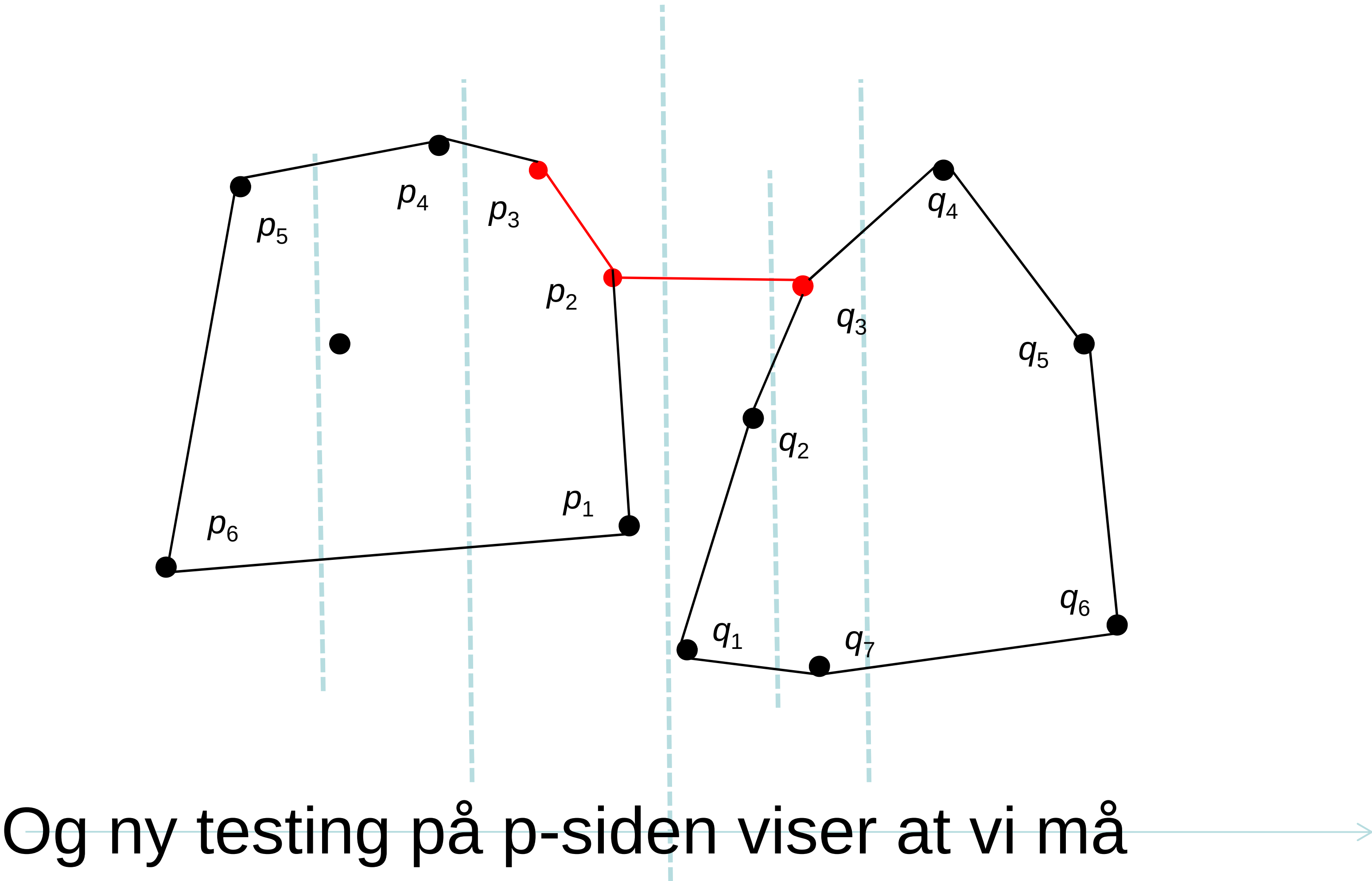
- Men nå svinger p1 – q3 – q4 mot høyre
- Dermed flytter vi oss til andre siden

Tester svingretning på q3 - p1- p2



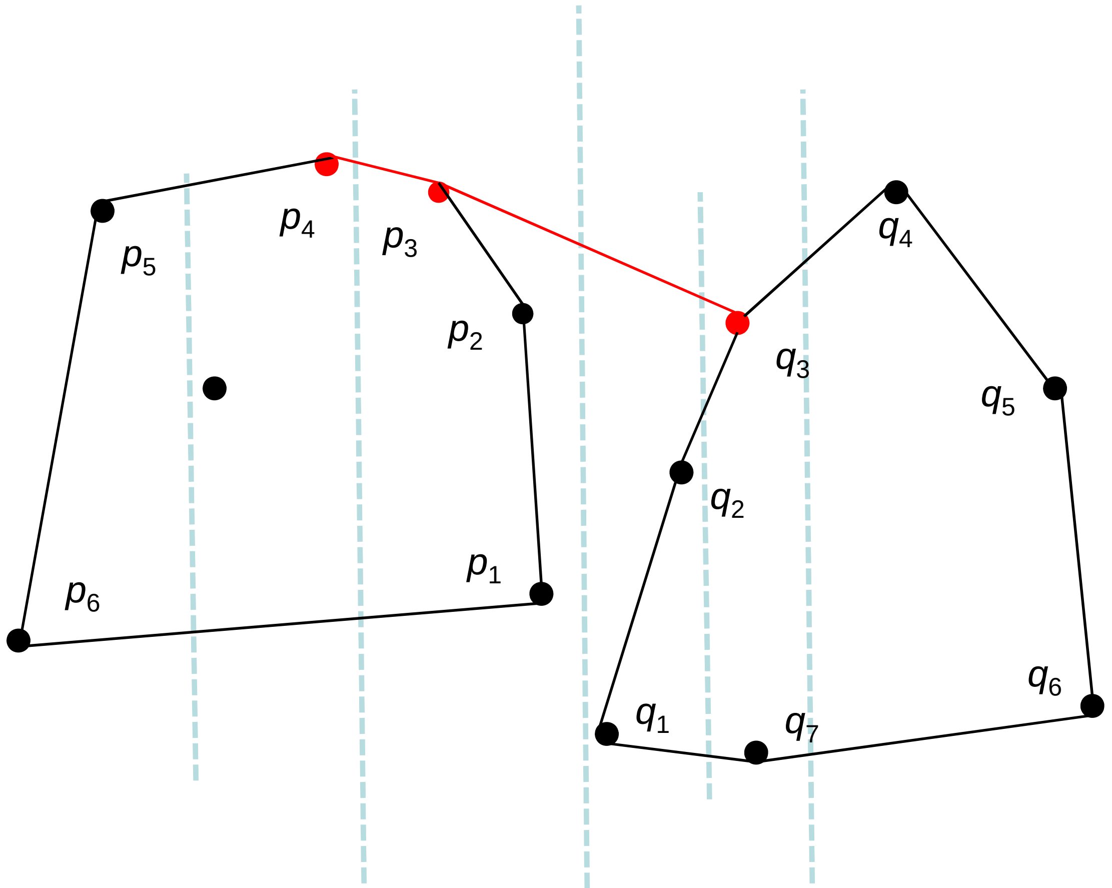
Denne går "gal vei", så da må vi gå til motsatt side og se om vi skal flytte ett eller flere hakk opp på den siden (her *p*-siden)

Så vi får denne situasjonen



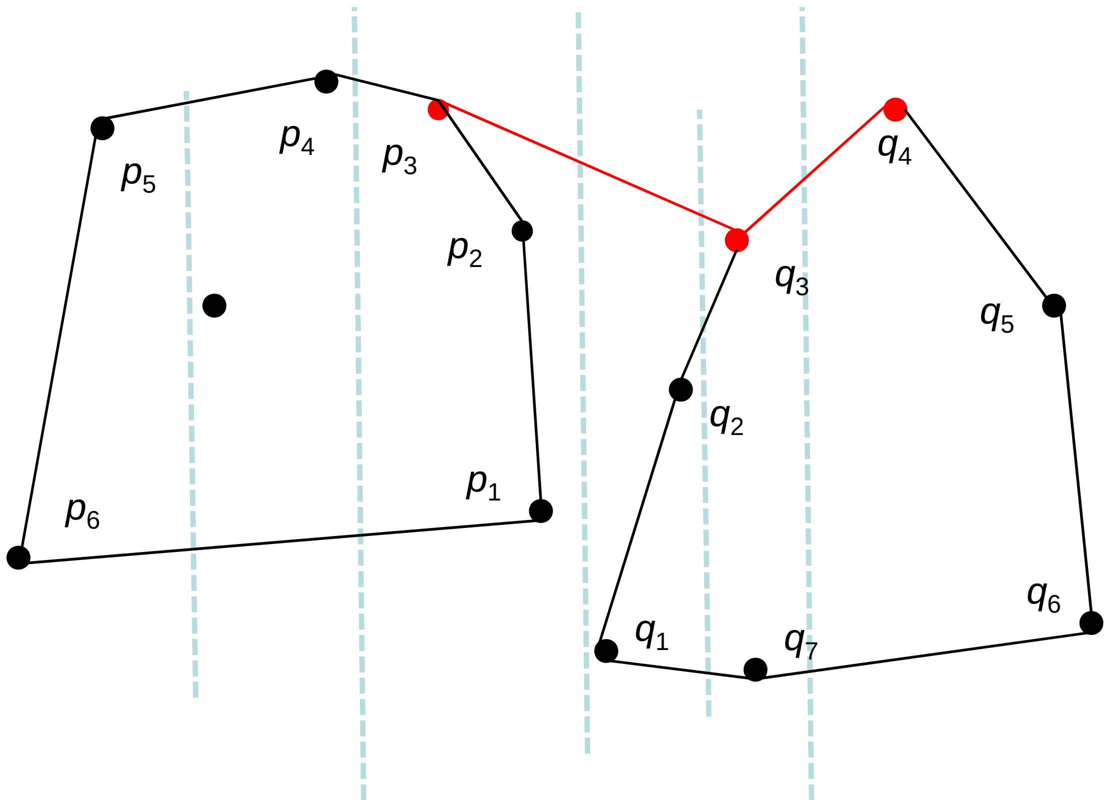
Og ny testing på p-siden viser at vi må fortsette her

Til denne situasjonen



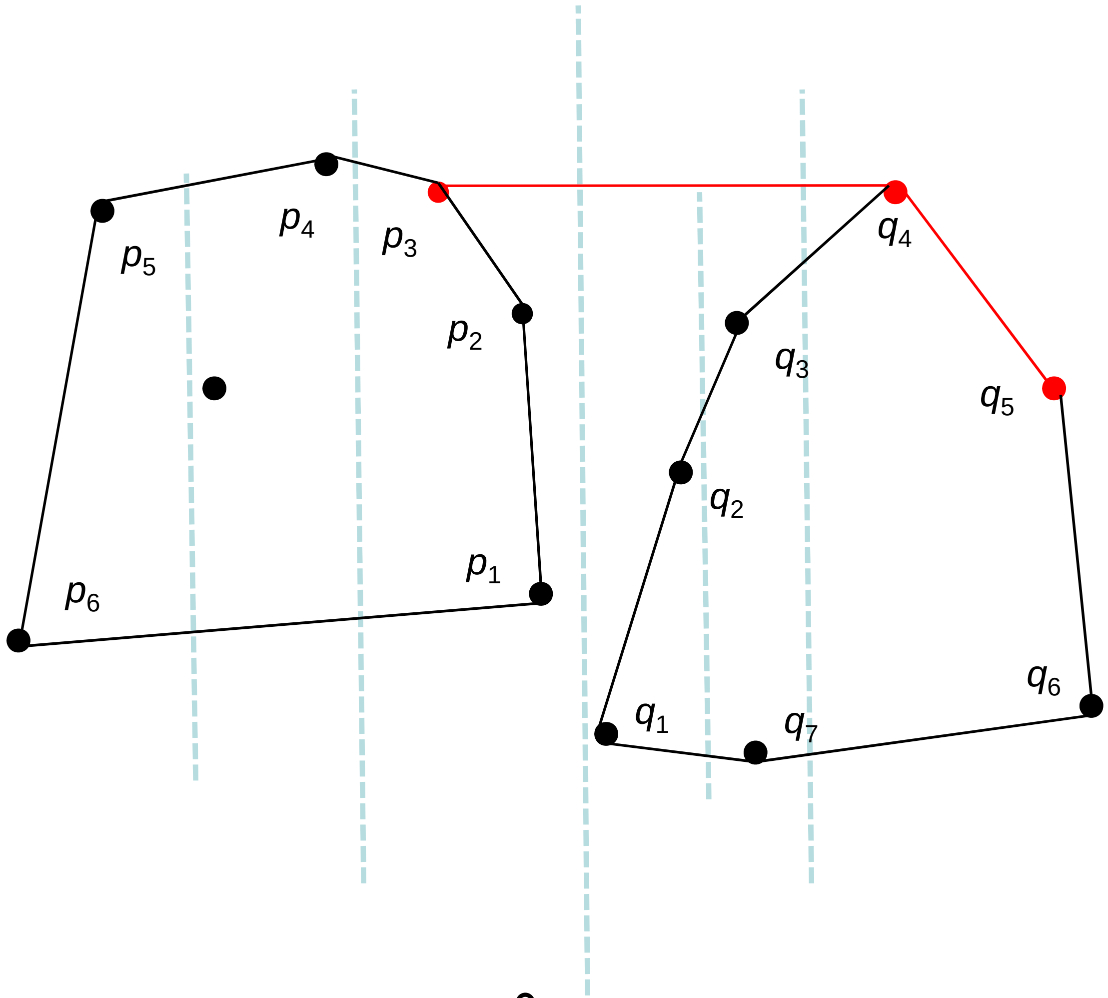
- Men *nå* er svingretningen riktig på p-siden
- Vi skifter tilbake til q-siden

Må da sjekke vinklen på høyre side



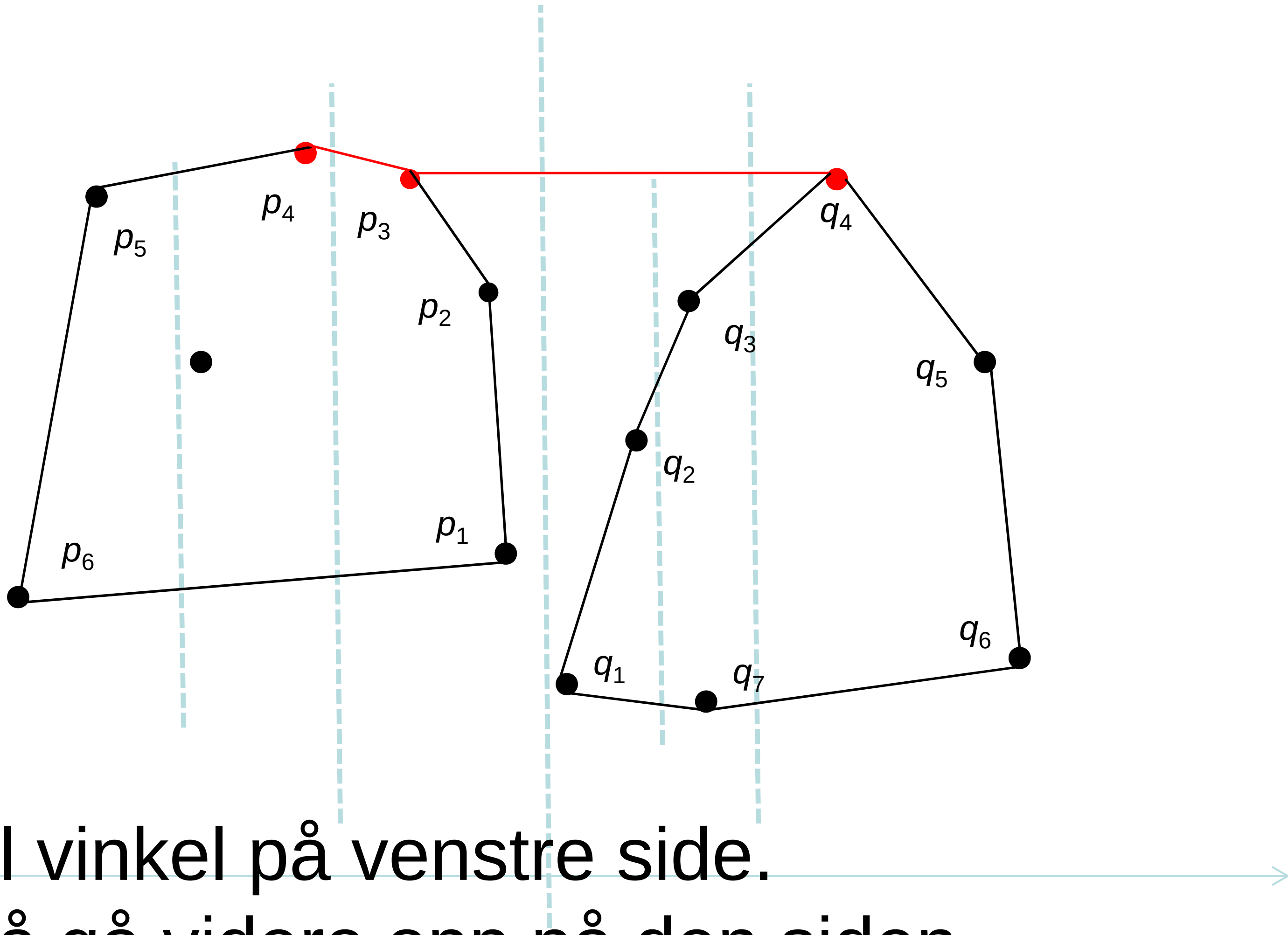
Denne er "gal", så vi går ett hakk videre her

Konveks innhylling



Riktig vinkel på høyre side, MEN …

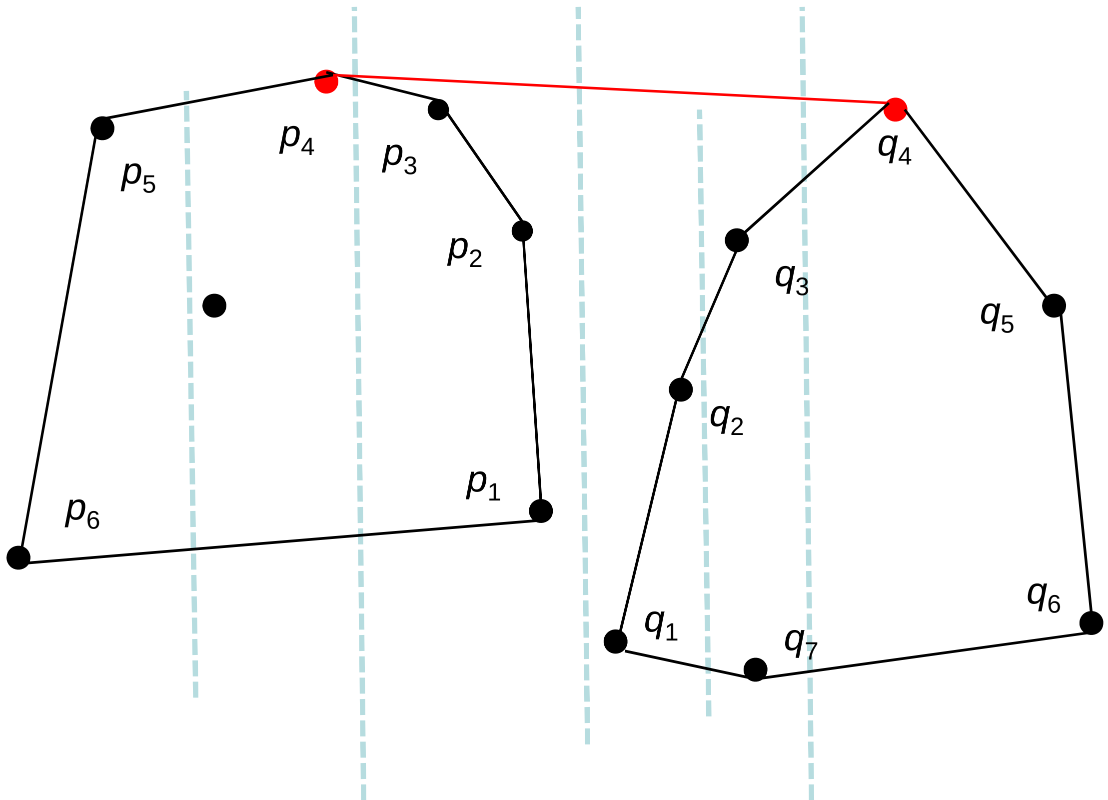
Konveks innhylling



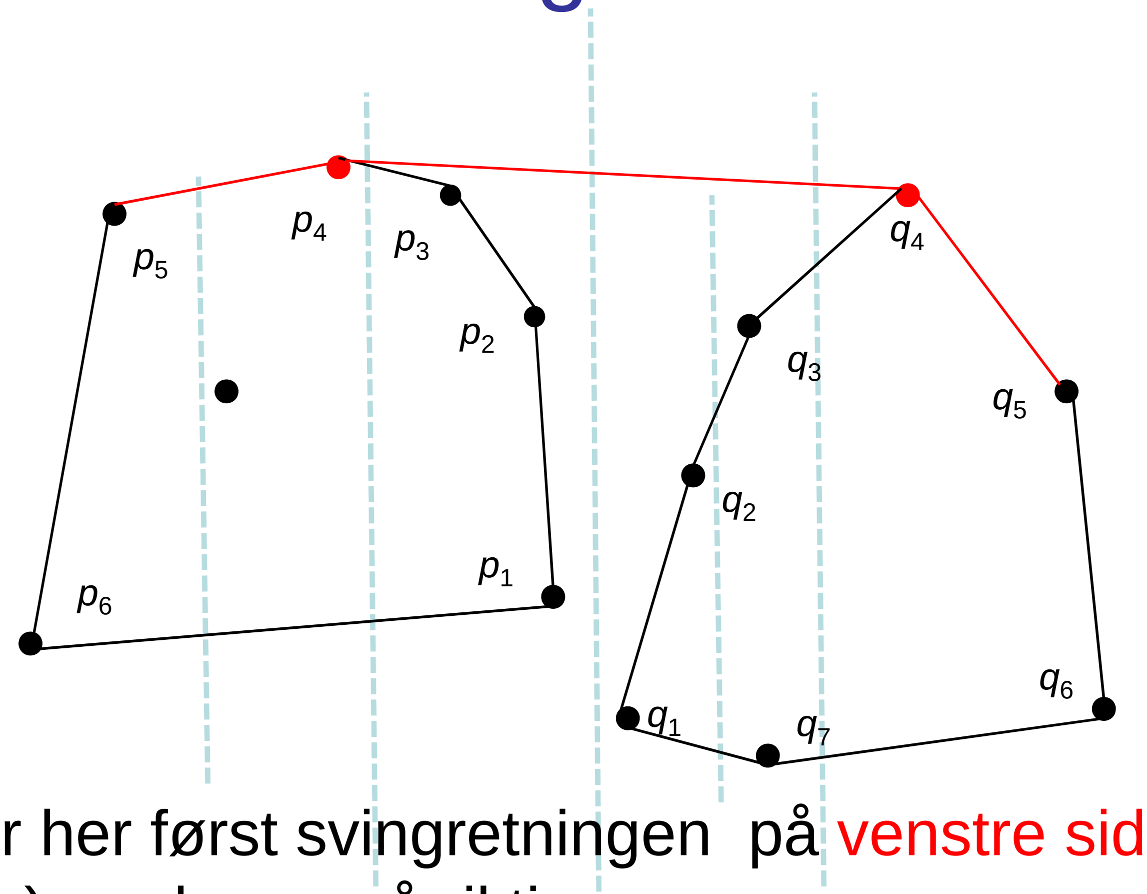
...feil vinkel på venstre side.

- Må gå videre opp på den siden

Får situasjonen

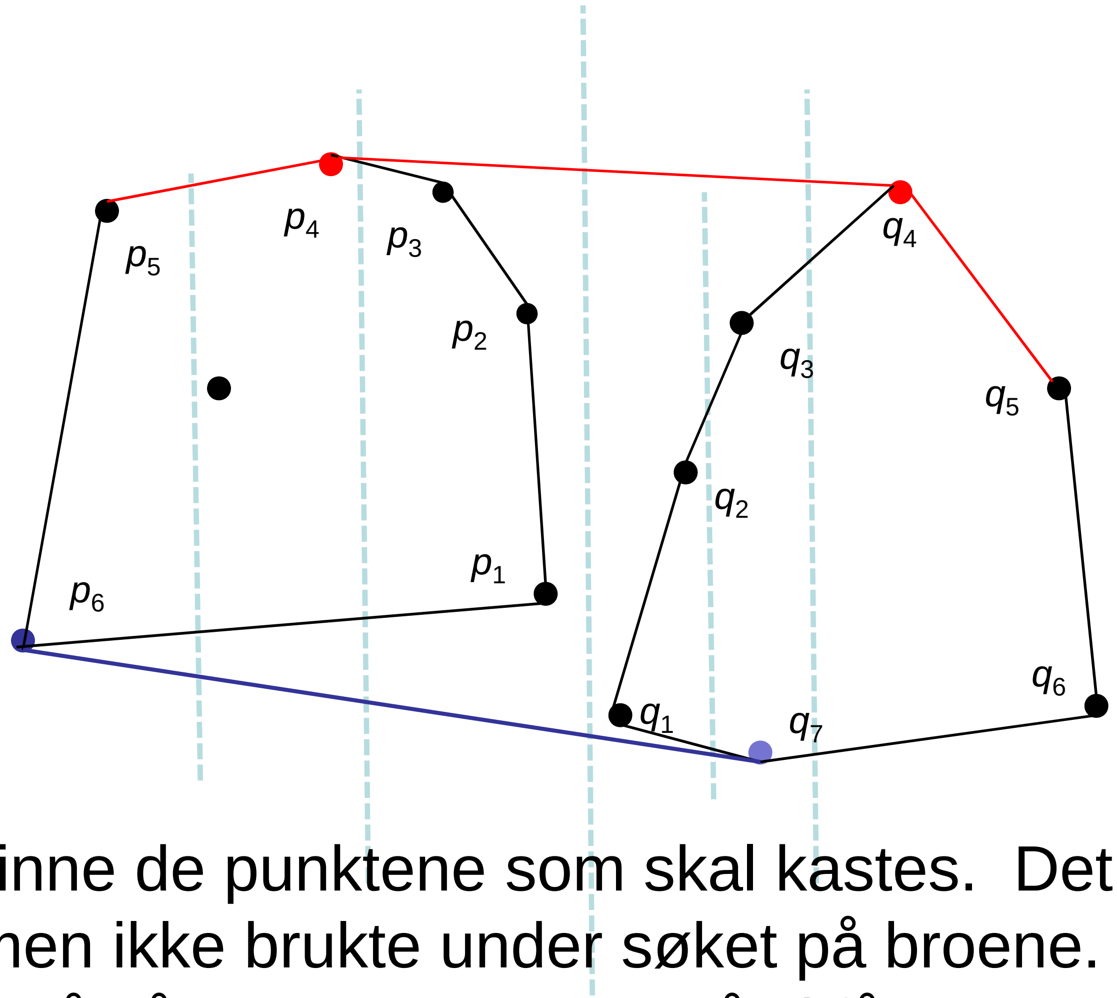


Avslutning av øvre bro



- Vi sjekker her først svingretningen på venstre side (der vi var) og den er nå riktig.
- Må så sjekke om det er mer igjen på høyre side, men også her er svingretningen nå riktig
- Når nå *begge* er riktige har vi funnet øvre bro!

Finner nedre bro på tilsvarende måte:



- Må nå finne de punktene som skal kastes. Det er de vi var innom men ikke brukte under søket på broene.
- Resten må så om-nummereres så vi får en sammenhengende nummerering rundt det nye polygonet.

Tidsforbruk:
Algoritme for konveks innhylling

- Vi sorterer først nodene etter x-koordinat. Dette tar tid $O(n \log n)$, der n er totalt antall punkter

- Hver gang vi spleiser to innhyllinger må vi potensielt flytte endepunktene på broen(e) $O(m)$ ganger, der m er antall noder til sammen i de to som skal spleises. Så hver spleis tar tid $O(m)$.

- All spleising på ett dybdenivå tar derfor tid:

  $O(m_1) + O(m_2) + O(m_3) + \ldots + O(m_k)$ som blir lik

  $O(n)$ siden $m_1 + m_2 + m_3 + \ldots + m_k = n$

- Det er maks $O(\log n)$ nivåer, og total kjøretid blir derfor:

  $O(n \log n)$