Dual Voronojevega diagrama v Hilbertovi metriki, definirani z večkotnikom K in mesti S, je *Delaunayeva triangulacija*, označena kot $\mathrm{DT}(S)$, v kateri sta mesti p in q povezani, če sta njuni pripadajoči Voronojevi celici sosednji.

Povezanost dveh mest lahko preverimo s pomočjo Hilbertovih krogel. Dve mesti sta povezani, če in samo če obstaja Hilbertova krogla, katere rob vsebuje obe mesti, medtem ko njena notranjost ne vsebuje nobenega. Podobno, tri točke določajo trikotnik v Delaunayejevi triangulaciji, ko obstaja Hilbertova krogla, ki vsebuje vse tri točke na svojem robu in ne vsebuje nobene druge točke. Krožnico, ki omejuje to kroglo, imenujemo *Hilbertova očrtana krožnica*, ki je enolično določena, če obstaja.

0.1 Vpeljava pomožnih struktur

V nadaljevanju bomo predstavili algoritem za konstrukcijo $\mathrm{DT}(S)$. Pri tem bomo uporabili simetrale in njihova krajišča, pri čemer po pomemben vrstni red zapisa. Krajišče na (a, b)-simetrali naj bo tisto, ki leži na levi strani vektorja $\vec{ab}$. Drugemu krajišču bomo rekli krajišče (b, a)-simetrale.

Kot je razvidno iz slike **??**, triangulacija ne pokrije celotnega večkotnika, zato bomo definirali nekaj pomožnih elementov, s katerimi bomo zagotovili, da bo K popolnoma pokrit.

Trikotnik, katerega vsa ogljišča so mesta, se imenuje *standardni trikotnik*. Za vsako povezavo pq, ki meji na zunanjo stran triangualizacije, obstaja eno krajišče (p, q)-simetrale, ki leži na robu $\delta(K)$. Naj x označuje to krajišče. Potem trikotniku Δpqx pravimo *zob*. Vse dele večkotnika K, ki niso standardni trikotniki ali zobje, bomo imenovali *praznine*. Slednje niso nujno trikotniki, ampak so enolično določene s tremi točkami. Zato bomo uporabljali notacijo Δpxy za praznino definirano z mestom p in robnima točkama x in y.

0.2 Naključnostni postopni algoritem

Algoritem se začne z naključno permutacijo množice S in inicializacijo triangulacije. Za slednjo izberemo poljubni mesti a in b in poiščemo krajišči (a, b)-simetrale. Nato dodamo povezavo ab in ustvarimo dva zoba tako, da povežemo a in b s krajišči simetrale. Postopoma dodajamo mesta in posodabljamo triangulacijo.

Vsako naslednjo točko lahko dodamo v standardni trikotnik, zob ali praznino. Če jo dodamo v standardni trikotnik, jo povežemo z ogljišči trikotnika. Če točko dodamo v zob, izbrišemo ogljišče zoba, ki je na robu, povežemo novo točko z najbližjima mestoma in dodamo dva nova zoba glede na novonastali simetrali. Če točko dodamo v praznino, jo povežemo z edinim mestom v praznini in dodamo dva nova zoba.

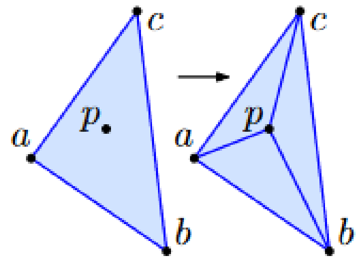
Ob dodajanju novega vozlišča se lahko zgodi, da nova triangulacija ne ustreza pogoju praznih Hilbertovih očrtanih krogov. V nekaterih primerih pa lahko pride do izgube geometrijske pravilnosti, na primer, ko se dva zoba prekrivata. Morebitne kršitve popravimo s procedurama Obrni povezavo in Popravi zob.

Algoritem 1 Konstrukcija Delaunayeve triangulacije množice S.

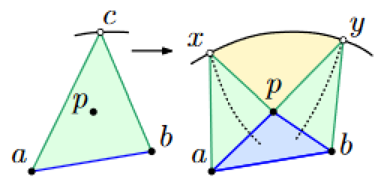
Procedura DELAUNAYEVA TRIANGULACIJA(S)
  $\tau \leftarrow$ prazna triangulacija
  Naključno permutiraj S
  $a, b \leftarrow$ poljubni točki iz S
  $x, y \leftarrow$ krajišči (a, b)-simetrale
  Dodaj povezave ab, ax, ay, bx in by v τ
  for all $p \in S \setminus \{a, b\}$ **do**
    DODAJ(p, τ)
  end for
  return τ
end Procedura

Algoritem 2 Dodajanje točke p iz S v triangulacijo τ.

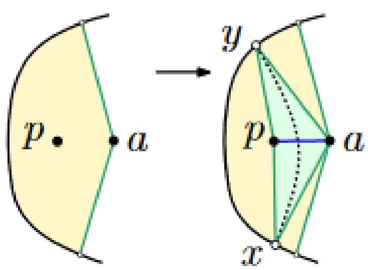
Procedura DODAJ(p, τ)
  $\Delta abc \leftarrow$ trikotnik v τ, ki vsebuje p
  if Δabc je standardni trikotnik **then**
    Dodaj povezave ap, bp in cp v τ
    OBRNI POVEZAVO(ab, p, τ), OBRNI POVEZAVO(bc, p, τ),
    OBRNI POVEZAVO(ca, p, τ)
  else if Δabc je zob **then**
    Naj bosta a in b mesti in c krajišče (a, b)-simetrale
    $x, y \leftarrow$ krajišči (a, p)- in (p, b)-simetral
    Iz τ odstrani vozlišče c ter povezavi ac in bc
    Dodaj povezave ap, bp, ax, px, by in py v τ
    OBRNI POVEZAVO(ab, p, τ)
    POPRAVI ZOB($\Delta apx, p, \tau$), POPRAVI ZOB($\Delta pby, p, \tau$)
  else
    Naj bo a mesto in naj bosta b in c na robu
    $x, y \leftarrow$ krajišči (a, p)- in (p, a)-simetral
    Dodaj povezave ap, ax, px, ay in py v τ
    POPRAVI ZOB($\Delta apx, p, \tau$), POPRAVI ZOB($\Delta pay, p, \tau$)
  end if
end Procedura



Slika 1: Dodajanje v standardni trikotnik.



Slika 2: Dodajanje v zob.



Slika 3: Dodajanje v praznino.

Obrni povezavo sprejme povezavo ab in točko, ki smo jo dodali, p. Nato poišče trikotnik Δabc, ki leži na desni strani povezave ab, in preveri ali p leži znotraj očrtanega kroga določenega s točkami a, b in c. Če leži, odstrani povezavo ab in preveri ali je Δabc standardni trikotnik ali zob. V kolikor je standardni trikotnik, doda povezavo pc in preveri ali morata biti tudi povezavi ac in cb obrnjeni. Če je Δabc zob, odstranimo vozlišče c na robu, poiščemo krajišči x in y (p, a)- in (b, p)-simetral in v τ dodamo nova zoba Δpax in Δbpy.

Procedura Popravi zob razreši probleme, ko se dva zoba prekrivata. Procedura sprejme trikotnik Δabx, kjer sta a in b mesti, x pa krajišče na robu. Novo dodano mesto je ena izmed točk a ali b. Naj bo Δbcy sosednji zob zoba Δabx v smeri urinega kazalca. Če se prekrivata, ju zamenjamo s standardnim trikotnikom Δabc in zobom Δacz, kjer je z krajišče (a, c)-simetrale. Če je novododana točka a, moramo preveriti ali je potrebno obrniti povezavo bc in popraviti zob

Algoritem 3 Obračanje povezav, v primeru kršenja pogoja praznih hilbertovih krogov.

Procedura OBRNI POVEZAVO(ab, p, τ)
  $c \leftarrow$ vozlišče trikotnika na desni strani povezave ab v τ
  if p leži v Hilbertovem očrtanem krogu določenem z a, b, c **then**
    Odstrani povezavo ab iz τ
    if Δabc je standardni trikotnik **then**
      Dodaj povezavo pc v τ
      OBRNI POVEZAVO(ac, p, τ), OBRNI POVEZAVO(cb, p, τ)
    else
      $x, y \leftarrow$ krajišči (p, a)- in (b, p)-simetral
      Odstrani vozlišče c in povezavi ac ter bc it τ
      Dodaj povezave ax, px, by in py v τ
    end if
  end if
end Procedura

Δacz.

Algoritem 4 Poprabljanje zob, ki se prekrivajo.

Procedura POPRABI ZOB($\Delta abx, p, \tau$)
  $\Delta bcy \leftarrow$ sosednji zob zobu Δabx v smeri urinega kazalca
  if Δabx in Δbcy se prekrivata **then**
    $z \leftarrow$ krajišče (a, c)-simetrale
    Odstrani vozlišči x in y ter povezave ax, bx, by in cy iz τ
    Dodaj povezave ac, az in cz v τ
    if $p = a$ **then**
      OBRNI POVEZAVO(bc, p, τ)
      POPRAVI ZOB($\Delta acz, p, \tau$)
    end if
  else
    Ponovi za zob, ki je sosednji Δabx v nasprotni smeri urinega kazalca
  end if
end Procedura

Na koncu izpeljimo pričakovano časovno zahtevnost danega algoritma. Podobno kot pri Delaunayevi triangulaciji v Evklidskem prostoru, lahko poiščemo trikotnik, v katerem je vsebovana novododana točka v pričakovanem času $\mathcal{O}(\log n)$ z definiranjem podatkovnih strukture za določanje položaja točke, podobno kot to storijo v članku .... Glavna razlika v časovni zahtevnosti nastane zaradi izračuna Hilbertovih očrtanih krogov. Omenimo brez dokaza, da so le-ti izračunani v pričakovanem času $\mathcal{O}(\log^3 m)$. Dokaz bralec najde v članku ... . Faktorja $\mathcal{O}(\log n)$ in $\mathcal{O}(\log^3 m)$ se izvedeta končno število krat za vsako od n dodanih točk. Tako smo dokazali naslednjo trditev.

Trditev 1 *Naključna postopna konstrukcija Delaunayeve triangulacije množice n točk znotraj konveksnega m-kotnika ima pričakovano časovno zahtevnost $\mathcal{O}(n(\log^3 m + \log n))$.*