

**Težji projekt pri Matematičnem programiranju**

Timen Bobnar

Ljubljana, 2024

# Povzetek

V ravnini so dane točke A1, A2, … An. Enostavna sklenjena lomljenka je krivulja brez samopresečišč, ki je sestavljena iz daljic A1A2, A2A3,…,AnA1 .

Zanimala nas bosta dva problema, in sicer:

Problem 1:

Zanimalo nas bo ali neko zaporedje točk res predstavlja enostavno sklenjeno lomljenko.

Problem 2:

Če imamo enostavno sklenjeno lomljenko in še neko poljubno točko, nas bo zanimalo, ali ja ta točka v notranjosti enostavne sklenjene lomljenke ali ne.

# Problem 1

## Navodilo:

Sestavi funkcijo, ki ugotovi, ali je zaporedje točk A1, A2, … An res dana sklenjena lomljenka brez samopresečišč.

## Primeri:

Primer, ki ima samo presečišče, torej ni enostavna sklenjena lomljenka.

Slika, ki vsebuje besede vrstica, grafični prikaz, diagram, posnetek zaslona

Opis je samodejno ustvarjen

Primer, ki nima samo presečišča, torej je enostavna sklenjena lomljenka.

Slika, ki vsebuje besede vrstica, diagram, grafični prikaz

Opis je samodejno ustvarjen

## Teoretična osnova:

Da rešimo naš problem moramo za vsako daljico preveriti ali se seka s kakšno od ostalih daljic naše sklenjene lomljenke.

Presečišče se lahko zgodi le na naslednje načine. Vzemimo 2 daljici prva daljica je definirana s točkama A1 in A2 . Druga daljica pa z B1 in B2.

* Dve premici se sekata natanko tedaj, ko veljata naslednja pogoja.

1. Če imamo daljico A1 A2  in pogledamo orientacijo točk B1 ter B2 glede na daljico A1 A2, sta orientaciji nasprotno predznačeni.
2. Če imamo daljico B1 B2  in pogledamo orientacijo točk A1 ter A2 glede na daljico B1 B2, sta orientaciji nasprotno predznačeni.

* Dve daljici se tudi lahko sekata natanko tedaj, ko eno izmed krajišč prve daljice leži na drugi daljici.

### Pogoj 1:

Prvo vprašanje, ki se pojavi je, kako pridobimo orientacijo točke glede na daljico.

Pri tem si lahko pomagamo z vektorskim produktom, ki ima formulo:

Torej če imamo daljico A1 A2  in poljubno točko C, potem definiramo vektor in vektor . Če stvari tako definiramo, potem bo vektorski produkt pozitiven, ko bo točka C na levi strani premice, negativen, ko je ne desni strani in 0, ko sta vektorja in vzporedna.

Sedaj si oglejmo kako deluje pogoj 1.

Imamo 3 možne situacije:

1. Situacija:

Slika, ki vsebuje besede vrstica, grafični prikaz, vzporedno, diagram

Opis je samodejno ustvarjen

To je primer ko velja le en pogoj. B1 je orientirana pozitivno glede na daljico A1 A2,točka B2 pa negativno. Hkrati sta toči A1 in A2 obe orientirani negativno glede na premico B1B2. Daljici se torej ne sekata.

1. Situacija

Slika, ki vsebuje besede vrstica, diagram, grafični prikaz, vzporedno

Opis je samodejno ustvarjen

To je primer ko veljata oba pogoja. B1 je orientirana pozitivno glede na daljico A1 A2, točka B2 pa negativno. Hkrati sta točki A1 in A2 nasprotno orientirani glede na premico B1B2. Torej se daljici sekata.

1. Situacija

Slika, ki vsebuje besede vrstica, diagram, grafični prikaz, posnetek zaslona

Opis je samodejno ustvarjen

Tu pridemo do situacije ko velja prvi pogoj. Ko preverimo drugi pogoj, pa pridemo do situacije, da je A1  negativno orientiran, ter da A2 laži na premici med B1 in B2. Ta problem je rešljiv že samo s pregledamo orientacij, vendar ga bomo zaradi enostavnejše kode združili z naslednjim problemom (ali točka leži na premici).

1. Situacija

Slika, ki vsebuje besede vrstica, grafični prikaz, diagram, številka

Opis je samodejno ustvarjen Slika, ki vsebuje besede vrstica, grafični prikaz, posnetek zaslona, diagram

Opis je samodejno ustvarjen

Tu pridemo do situacije, ko so vse orientacije 0. In si z orientacijami ne moremo pomagati da ocenimo ali se daljici sekata ali ne. Oba primera imata vse orientacije enake 0, vendar prvi ima presečišče drugi pa ne.

Zadnje dva problema bomo oba rešili tako, da preverimo ali katero od krajišč leži na drugi daljici. Torej pogoj 2.

### Pogoj 2:

Spomnimo se, da pogoj 2 izvedemo, če je ena od orientacij enaka 0.

Pri pogoju 2 si lahko pomagamo z preprosto idejo, ki jo izpeljemo iz algebraične definicije premice. Imejmo premico A1A2 ter točko C. Sedaj lahko definiramo:

Iz formule sledi, da v primeru, ko je , je točka C na premici A1A2.

Lambda je torej enaka:

Lambde bomo posebej izračunali za x in y koordinate tako, da bomo pisali .

V splošnem primeru sledi, da ko sta enaki ter iz intervala [0,1], da točka leži na daljici.

Vendar imamo nekaj problemov.

#### Problemi:

Problemi nastopijo ko so naše daljice vodoravne ali navpične. V tem primeru je ena od nedefinirana, saj bomo delili z 0.

Recimo da je nedefinirana. To pomeni, da je premica navpična, saj se *x* koordinata začetne ter končne točke ne spremeni. Iz tega sledi, da če hočemo, da naša točka leži na daljici mora izpolnjevati dva pogoja.

1. Kot smo omenili je premica navpična, torej imata končna in začetna točka isto x koordinato, zato jo mora imeti tudi točka C, če hočemo da leži na nje.
2. Ker je premica navpična, se lahko naša točka premika le med min(A1y , A2y) in max(A1y , A2y), če hočemo da leži na daljici.

Podobno velja, če je nedefinirana.

## Implementacija v Matlab-u

Nalogo smo implementirali, tako da smo definirali nekaj različnih funkcij.

* Prva funkcija je orientacija oziroma implementacija prvega pogoja. Ta nam pove, kako je točka orientirana glede na neko daljico.

Slika, ki vsebuje besede besedilo, pisava, posnetek zaslona

Opis je samodejno ustvarjenV funkciji *or* izračunamo preko vektorskega produkta.

* Druga funkcija je funkcija z imenom na premici oziroma implementacija drugega pogoja. Ta ugotovi ali se neka točka nahaja na daljici.

Slika, ki vsebuje besede besedilo, pisava, vrstica, posnetek zaslona

Opis je samodejno ustvarjen

Lamde izračunamo po definiciji premice. V prvem pogoju rešujemo tako imenovan Problem pri pogoju dva iz teoretične osnove. V drugem pogoju pa samo sledimo pogoju 2, ki pravi, da če sta lamdi enaki in iz intervala [0,1], potem točka leži na premici.

* Tretja funkcija se imenuje ali\_se\_sekajo. Ta funkcija prejme 4 točke oziroma 2 daljici in vrne true če se daljici sekata in false sicer. To naredi tako, da najprej preveri pogoje orientacije:

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, številka

Opis je samodejno ustvarjenV primeru označenim s splošno preverjamo pogoj:

Dve premici se sekata natanko tedaj ko veljata naslednja pogoja.

1. Če imamo daljico A1 A2  in pogledamo orientacijo točk B1 ter B2 glede na daljico A1 A2. Sta orientaciji nasprotno predznačeni.
2. Če imamo daljico B1 B2  in pogledamo orientacijo točk A1 ter A2 glede na daljico B1 B2. Sta orientaciji nasprotno predznačeni.

V primeru, ko pa je kakšna od orientacije enaka 0 uporabimo funkcijo na\_premici:

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, algebra

Opis je samodejno ustvarjen

Pri teh pogojih preverimo ali točka, ki je imela orientacijo 0 leži na daljici.

* Četrta funkcija se imenuje lomljenka. V tej funkciji se sprehodimo čez vse možne kombinacije daljic naše enostavne sklenjene lomljenke.

Slika, ki vsebuje besede besedilo, pisava, vrstica, posnetek zaslona

Opis je samodejno ustvarjen

Najprej preverimo ali je sploh možno sestaviti lomljenko. Če imamo manj kot 3 točke to ni možno, saj dve daljici ne moreta sestaviti lomljenke.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, algebra

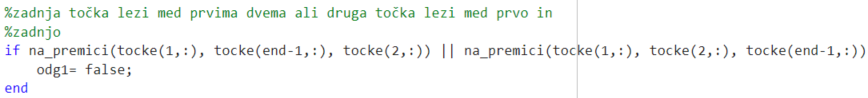
Opis je samodejno ustvarjen

V tem delu najprej preverimo ali je zadnja točka enaka prvi, če sta enaki to ni sklenjena lomljenka. Sicer pa dodamo na konec prvo točko za lažje risanje in obravnavo.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava, številka

Opis je samodejno ustvarjen

V tem delu najprej *n* zmanjšamo za ena, da ne pride do prekrivanj v zanki. Zunanja zanka nam bo torej opisala vse segmente lomljenke razen zadnjega. Druga zanka pa vse segmente od i+2-gega naprej. Opazimo, da pri preverjanju preskočimo en segment in sicer Ai+1Ai+2, ta segment ustvari samo presečišče natanko tedaj, ko je Ai+2 na daljici AiAi+1, kar pa že tako ali tako preverimo.



Ker smo zadnjo daljico izpustili, moramo preveriti še njo. Preverimo pogoj, zadnja točka leži med prvima dvema ali druga točka leži med prvo in zadnjo. Če je to res, potem to ni enostavna sklenjena lomljenka.

# Problem 2

## Navodilo:

Sestavi funkcijo, ki ugotovi, ali se dana točka T nahaja v notranjosti enostavne sklenjene lomljenke A1A2, A2A3,…,AnA1 ali ne.

## Primeri:

Primer, ko je točka v notranjosti enostavne sklenjene lomljenke:

Slika, ki vsebuje besede diagram, vrstica, grafični prikaz, vzporedno

Opis je samodejno ustvarjen

Primer ko je točka izven enostavne sklenjene lomljneke:

Slika, ki vsebuje besede vrstica, diagram, grafični prikaz, vzporedno

Opis je samodejno ustvarjen

## Teoretična osnova

Teoretična osnova za našo rešitev prihaja iz Ray casting aloritma. Ray casting algoritem sloni na matematičnem ozadju iz teorije grafov in topologije. Tu bomo samo uporabili idejo, ki pravi, če imamo poljubno točko ter enostavno sklenjeno lomljenko. Iz poljubne točke potegnemo premico (ni važno v katero smer). Naša premica sedaj seka enostavno sklenjeno lomljenko v sodo oziroma liho mnogo točkah. V primeru, da premica seka lomljenko sodo krat pomeni, da točka ni vsebovana. Če premica seka lomljenko liho krat pomeni, da je točka vsebovana v lomljenki.

Slika, ki vsebuje besede posnetek zaslona, besedilo, pravokotnik, vrstica

Opis je samodejno ustvarjen Slika, ki vsebuje besede vrstica, diagram, posnetek zaslona, grafični prikaz

Opis je samodejno ustvarjen

Opazimo, da je točka res zunaj in, če preštejemo presečišča jih je sodo.

Slika, ki vsebuje besede posnetek zaslona, besedilo, pravokotnik, vrstica

Opis je samodejno ustvarjenSlika, ki vsebuje besede vrstica, besedilo, posnetek zaslona, diagram

Opis je samodejno ustvarjen

Tu opazimo, da je točka v notranjosti, in kot pričakovano je število presečišč liho.

Uporabili bomo tudi vse kar je bilo povedano v teoretični osnovi za prvi problem.

## Implementacija v Matlab-u

Nalogo smo implemenirali z uporabo različnih funkcij. Večina je enakih kot v prejšnjem poglavju in jih bomo zato samo našteli:

1. Orientacija
2. na\_premici
3. ali\_se\_sekajo

Definirali smo še eno novo funkcijo, ki preveri ali točka leži v lomljenki ali ne. Podano imamo naše točke, ki predstavljajo lomljenko in točko *p* ki predstavlja našo poljubno točko.

Slika, ki vsebuje besede besedilo, pisava, posnetek zaslona, algebra

Opis je samodejno ustvarjen

Našo lomljenko zapremo tako, da dodamo še eno točko. In *n*(števec daljic) povečamo za ena, da bomo preverili vse daljice. Ustvarimo tudi točko extreme, ki jo potrebujemo, da ustvarimo daljico od točke *p* do neskončnosti v neko smer.

Slika, ki vsebuje besede besedilo, posnetek zaslona, pisava

Opis je samodejno ustvarjen

Za vsako daljico naše lomljenke preverimo ali se seka z našo umetno daljico od *p* do neskončno. Če se seka povečamo števec za 1. Preverimo pa tudi ali se točka *p* nahaja na daljici, to storimo le, ko je orientacija enaka 0, če se izkaže da je točka *p* res na neki daljici naše lomljenke, lahko takoj odgovorimo, da naša točka leži znotraj lomljenke.

Slika, ki vsebuje besede besedilo, pisava, posnetek zaslona

Opis je samodejno ustvarjen

Preverimo ali je število presečišč liho ali sodo in podamo odgovor.

# Viri

* Point in polygon, <https://en.wikipedia.org/wiki/Point_in_polygon> , 25.2.2024
* How to chech if a given point lies liside or outside a polygon?, <https://www.geeksforgeeks.org/how-to-check-if-a-given-point-lies-inside-a-polygon/> , 25.2.2024
* Nekatere ideje so bile pridobljene s pomočjo ChatGPT-ja
* Nekatere ideje so bile razvite skupaj s kolegi