

# Problem iskanja najdaljšega drevesa v ravnini

Ema Kozin in Filip Škrlj

december in januar 2020

# Kazalo

<b>1</b>	<b>Prestavitev problema</b>	<b>2</b>
<b>2</b>	<b>Algoritem</b>	<b>2</b>
2.1	Opis . . . . .	2
2.2	Koda . . . . .	3
<b>3</b>	<b>Sklep</b>	<b>4</b>

# 1 Prestavitev problema

Pogledala sva si, kako bi na dani množici  $P$ , ki vsebuje  $n$  točk, našla graf z največjo dolžino, za katerega velja, da se njegove povezave ne križajo ter je drevo. Problem bova prikazala na nekaj primerih in ugotovila, v kolikšen času, je ta problem rešljiv glede na velikost množice  $P$ .

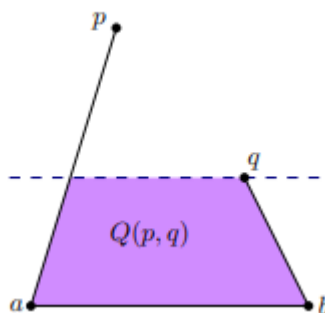
## 2 Algoritem

### 2.1 Opis

Pri reševanju problema se bova osredotočila na iskanje najdaljše *dvozvezde*, torej drevesa s takima točkama  $a$  in  $b$ , da v tem drevesu obstaja povezava med njima in da je vsaka druga točka povezana bodisi z  $a$  bodisi z  $b$ . Sedaj lahko uporabimo dinamično programiranje.

Brez škode za splošno lahko predpostavimo, da  $a$  leži levo od  $b$ , ti točki naj ležita na vodoravni črti, nad katero so še vse preostale točke.

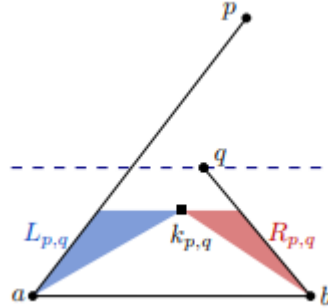
Naprej si ogledamo podproblem, označen s parom  $(p, q)$ , kjer sta  $p$  in  $q$  iz množice  $P$  in se povezavi  $ap$  in  $bq$  ne sekata. Za vsak tak par  $(p, q)$  opazimo, da povezave  $ap, pq, qb$  in  $ba$  tvorijo štirikotnik. Naj bo sedaj  $Q(p, q)$  označen del tega štirikotnika, ki je pod črto, vzporedno stranici  $ab$ , ki gre skozi nižje ležečo točko izmed  $p$  in  $q$ , torej pod  $y = \min\{y(p), y(q)\}$ . Glej spodnjo sliko.



Dolžino najdaljše dvozvezde s korenoma  $a$  in  $b$  na točkah znotraj  $Q(p, q)$  označimo z  $Z(p, q)$ , pri tem pa ne upoštevamo dolžine povezave  $ab$ . Če  $Q(p, q)$  ne vsebuje nobene točke iz  $P$ , je  $Z(p, q) = 0$ .

V primeru, da vsebuje vsaj eno točko iz  $P$ , pa označimo s  $k_{p,q}$  najvišje ležečo točko znotraj  $Q(p, q)$ . To točko povežemo z točko  $a$  ali točko  $b$ . Če jo povežemo z  $a$ , bomo tudi točke, ki so znotraj trikotnika  $L_{p,q}$ , omejenega z povezavama  $ap$  in  $ak_{p,q}$  in vodoravno črto  $y = y(k_{p,q})$ , povezali z  $a$ . Podobno,

če  $k_{p,q}$  povežemo z  $b$ , dobimo trikotnik na desni strani, označen z  $R(p, q)$ , omejen z  $bq, bk_{p,q}$  in  $y = y(k_{p,q})$ .



V prvem primeru dobimo namesto para  $(p, q)$  nov par  $(k_{p,q}, q)$ , v drugem primeru pa par  $(p, k_{p,q})$ . Za vsak veljaven par  $(p, q)$  dobimo naslednje:

$$Z(p, q) = \begin{cases} 0 & \text{če v } Q(p, q) \text{ ni točke iz } P; \\ \max \left\{ \begin{array}{l} Z(k_{p,q}, q) + \|ak_{p,q}\| + \sum_{l \in L_{p,q}} \|al\| \\ Z(p, k_{p,q}) + \|bk_{p,q}\| + \sum_{r \in R_{p,q}} \|br\| \end{array} \right\} & \text{sicer} \end{cases}$$

Zgornjo zvezo bova uporabila na vsakem veljavnem paru  $(p, q)$  in tako našla najdaljšo dvoezvezdo s korenoma  $a$  in  $b$ .

## 2.2 Koda

Najino kodo za reševanje problema iskanja najdaljšega drevesa sva uporabila programski jezik *Python*. Koda z komentarji je vsebovana v datoteki *ogrodje\_in\_algoritem.py* na najinem repozitoriju. V tem poročilo pa bova opisala njene glavne komponente, ključne za rešitev problema. Glavni del kode sva razdelila na tri dele, funkcije *pomožna*, *pomožna\_B* in *algo*.

Funkcija *pomožna* sprejme naslednje argumente: naš graf ter točke  $a, b, p$  in  $q$  (v opisu algoritma sva že razložila njihovo uporabo in pomen). Funkcija najprej preveri, ali ima par  $(p, q)$  že določeno vrednost  $Z(p, q)$ , ki predstavlja dolžino najdaljše dvoezvezde s korenoma v  $a$  in  $b$ . Potem določimo štirikotnik  $Q(p, q)$  tako, da primerjamo razdalji med daljico  $ab$  in točkama  $p$  in  $q$ . Nato najdemo še točko znotraj tega štirikotnika, ki je najdlje od daljice  $ab$  in jo označimo z  $k$ . Definiramo še levi trikotnik (če  $k$  povežemo z  $a$ ) in desni trikotnik (če  $k$  povežemo z  $b$ ). Če je znotraj teh trikotnikov še kakšna druga

točka našega grafa, jo povežemo z ustrezno točko izmed  $a$  in  $b$ . V primeru levega trikotnika nadaljujemo naš postopek z novim parom  $(k, q)$ , v primeru desnega pa z parom  $(p, k)$ . Na koncu vse združimo in z uporabo naše formule iz opisa algoritma izračunamo  $Z(p, q)$  za štirico točk  $(a, b, p, q)$ .

Naslednja pomembna funkcija je *pomožna\_B*, ki funkcijo *pomožna* uporabi na vseh parih  $(p, q)$  našega grafa. Najprej preveri, koliko je točk v grafu. Če sta točki le dve, torej ravno  $a$  in  $b$ , je naša dvozvezda dolžine 0. Če so točki tri, pa je dolžina najdaljše dvozvezde enaka maksimalni dolžini med točko  $a$  oziroma točko  $b$  in tretjo točko. Ko pa je točk več, funkcija preveri s pomočjo funkcije *alisesekata*, ali je par točk, ki bi ju radi povezali z  $a$  in  $b$  primeren in nato v tem primeru uporabi funkcijo *pomožna*.

Nazadnje funkcija *algo* steče po vseh točkah grafa, jih uporabi za korena dvozvede  $a$  in  $b$  ter naprej uporabi funkcijo *pomožna\_B* na ta kandidata korenov. Za vsak tak par korenov preveri, če je njuna dvozvedza najdaljša in na koncu to najdaljšo dolžino izpiše.

### 3 Sklep

Z uporabo najine kode sva ugotovila, da je problem res rešljiv v času  $O(n^2)$  preko dinamičnega programiranja kot je sugerirala najina literatura. Pri primerih uporabe kode na kvadratu, ozkem pravokotniku in kolobarju sva to potrdila.

## Literatura

- [1] Sergio Cabello, Michael Hoffmann, Katharina Klost, Wolfgang Mulzer, Josef Tkadlec. *Long plane trees (work in progress)*
- [2] Ahmad Biniaz, Prosenjit Bose, Kimberly Crosbie, Jean-Lou De Carufel, David Eppstein, Anil Maheshwari, Michiel Smid. *maximum plane trees in multipartite geometric graphs. Algorithmica*
- [3] Noga Alon, Sridhar Rajagopalan, Subhash Suri. *Long non-crossing configurations in the plane. Fundam.*