Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Hrany DT vo vybraných planárnych kresbách

Bakalárska práca

2019 František Michal Sebestyén

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

Hrany DT vo vybraných planárnych kresbách

Bakalárska práca

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra algebry a geometrie
Školiteľ:	doc. RNDr. Andrej Ferko, PhD.

Bratislava, 2019 František Michal Sebestyén

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: František Michal Sebestyén
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský
Sekundárny jazyk: anglický

Názov: Hrany DT vo vybraných planárnych kresbách
Delaunay Edges in Selected Planar Drawings

Anotácia: Prehľad problematiky kreslenia planárnych grafov. Výber reálnych súborov dát z kultúrneho dedičstva. Implementácia nástroja na hľadanie hrán Delaunayovej triangulácie. Experimenty.

Cieľ: Cieľom práce je vybrať zo známych planárnych grafov reprezentatívne súbory a charakterizovať ich vlastnosti. Špecifikácia projektu, implementácia vo vhodnom prostredí. Výpočet a vyhodnotenie výsledkov experimentov.

Literatúra: Nishizeki, T. - Rahman, S. 2004. Planar Graph Drawing. World Scientific.
Di Giacomo,E. et al. Graph Drawing. In Goodman, J. E. et al. eds. 2017. Handbook of Discrete and Computational Geometry. Third Edition. CRC Press. [online]http://www.csun.edu/~ctoth/Handbook/chap55.pdf
Ferko, A. - Moravčík, J. - Kolingerová, I. 2016. Souhvězdí jako podgrafy triangulací. Pokroky matematiky, fyziky a astronomie 61 (1), 14-20. [online] https://dml.cz/bitstream/handle/10338.dmlcz/144898/PokrokyMFA_61-2016-1_3.pdf

Kľúčové slová: kreslenie grafov, planárne grafy

Vedúci: doc. RNDr. Andrej Ferko, PhD.
Katedra: FMFI.KAG - Katedra algebry a geometrie
Vedúci katedry: doc. RNDr. Pavel Chalmovianský, PhD.

Spôsob sprístupnenia elektronickej verzie práce:
prípustná pre vlastnú VŠ

Dátum zadania: 26.10.2018

Dátum schválenia: 27.10.2018 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Poďakovanie: Týmto by som sa chcel poďakovať vedúcemu bakalárskej práce Doc. RNDr. Andrejovi Ferkovi, PhD. za pomoc pri jej písaní a jeho odborný dohľad, Emanuelovi Tesařovi za jeho cenné rady pri tvorbe aplikácie, rodine za podporu, špeciálne Sarah Belejovej, ktorá sa počas písania tejto práce, oficiálne stala mojou polovičkou.

Abstrakt

Cieľom práce je hľadanie hrán Delaunayovej triangulácie vo vybraných dátach, ktoré patria do nehmotného kultúrneho dedičstva. Navrhujeme spôsob ich reprezentácie pomocou planárnych kresieb grafu priamymi hranami a následne overujeme, či hrany kresby patria do Delaunayovej triangulácie. V práci predstavujeme nami vyvinutú aplikáciu, ktorá slúži na spracovanie dát do ich reprezentácie kresbou grafu a overovanie hľadanej vlastnosti. Uvádzame výsledky experimentov na Babylonských čísliciach a runách Starého Futharka.

Kľúčové slová: kreslenie grafov, planárne grafy

Abstract

The aim of this thesis is to search for Delaunay edges in selected data belonging to the cultural heritage. We suggest a way of representing them by planar graph drawings with straight lines. Afterwards, we verify that the edges of the drawing belong to Delaunay triangulation. We introduce software application, that is used to process data into their representation by a graph drawing and verifying the searched property. We present the results of experiments on Babylonian numerals and runes of Old Futhark.

Keywords: graph drawing, planar graphs

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam Algoritmov

Zoznam skratiek a označení

$\mathbb{R}$ Reálne čísla.

E^2 Dvojrozmerný euklidovský priestor

DT Delaunayova triangulácia

CCW Proti smeru chodu hodinových ručičiek

CW V smere chodu hodinových ručičiek

ES6, ECMAScript2015 ECMAScript 6

JS JavaScript

HTML Hypertext Markup Language

CSS Cascading Style Sheets

JSON JavaScript Object Notation

Úvod

Cieľom tejto práce je vytvoriť nástroj, ktorý pomôže pri charakterizácii geometrických vlastností vybraných dát svetového kultúrneho dedičstva.

Dáta budeme reprezentovať kresbami grafov, ktoré budeme vytvárať pomocou nami vyvinutého grafického editora. Myšlienku, že človek podvedome preferuje hrany s určitými geometrickými vlastnosťami, sme prebrali z diplomovej práce Jerguša Moravčíka [16]. V tejto práci sa budeme zaoberať hľadaním hrán Delaunayovej triangulácie v kresbách grafu. Na výpočet DT sme vytvorili samostatnú knižnicu.

Na začiatku práce definujeme základné pojmy z téorie kreslenia grafov a DT. V kapitole 2 opíšeme algoritmus na hľadanie DT, ktorý publikovali Der-Tsai Lee a Bruce Jay Schachter [15]. V kapitole 3 až 7 sa venujeme implementácií a práci s vytvoreným nástrojom. Experimenty, ich výber, spracovanie a výsledky sa nachádzajú v kapitole 8. V závere zhrnieme výsledky našej práce, navrhneme budúce experimenty a budúce funkcie nástroja.

Kapitola 1

Definície a základné pojmy

V tejto kapitole zadefinujeme základné pojmy a definície [11, 17], ktoré používame v nasledujúcich kapitolách.

1.1 Graf a vlastnosti grafu

Neorientovaný graf je dvojica $G = (V, E)$, kde V je neprázdna konečná množina a E množina neusporiadaných dvojíc z V.

Orientovaný graf je dvojica $G = (V, E)$ kde V je neprázdna konečná množina a E je množina usporiadaných dvojíc z V.

Prvky množiny V nazývame **vrcholy** a spravidla ich označujeme v.

Prvky množiny E nazývame **hrany** a spravidla ich označujeme e. V istých prípadoch ich budeme označovať $\{v_i, v_j\}$ (v neorientovanom grafe) resp. (v_i, v_j) (v orientovanom grafe), keď daná hrana spája v_i a v_j.

Graf $G* = (V*, E*)$ nazveme **podgrafom** grafu $G = (V, E)$ práve vtedy, keď $V* \subseteq V$ a $E* \subseteq E$.

Vrcholy v_i a v_j v neorientovanom grafe nazývame **susedné**, ak existuje hrana, ktorá ich spája. Teda ak existuje taká hrana $e_i \in E$ a $v_i, v_j \in e$.

Ohodnotenie hrán grafu je funkcia $h : E \mapsto \mathbb{R}$, kde $\mathbb{R}$ je množina reálnych čísel.

Váhu grafu definujeme ako: $\sum_{e \in E} h(e)$.

Cesta z v_0 do v_k je postupnosť vrcholov $(v_0, v_1, \ldots, v_{k-1}, v_k)$ taká, že každé dva vrcholy v_i a v_{i+1} sú susedné a ľubovoľné dva vrcholy v_i a v_j rôzne. **Dĺžka cesty je** k.

Komponent súvislosti neorientovaného grafu je taký podgraf H grafu G, že existuje cesta medzi ľubovoľnými dvoma vrcholmi podgrafu H a zároveň neexistuje hrana, ktorá nepatrí do H a spája vrchol z podgrafu H a vrchol z grafu G.

Cyklus je postupnosť vrcholov $(v_0, v_1, \ldots, v_{k-1}, v_k, v_0)$ taká, že každé dva vrcholy v_i a v_{i+1} sú susedné a ľubovoľné dva vrcholy v_i a v_j sú rôzne. Ak graf neobsahuje cyklus, vravíme, že graf je **acyklický**.

Graf je **súvislý**, ak pre každé dva ľubovoľné vrcholy v_i a v_j existuje cesta z v_i do v_j.

1.2 Kresba grafu (drawing of graph)

Euklidovský priestor označujeme E^2. Nech S je podmnožina E^2. Bod X je **izolovaný bod** podmnožiny S, ak existuje také okolie bodu X, ktoré neobsahuje žiadny iný bod z podmnožiny S.

Kresba (drawing) grafu je geometrická reprezentácia grafu, vrcholy sú reprezentované izolovanými bodmi a hrany krivkami medzi vrcholmi.

Vnútorná oblasť kresby grafu je neprázdna časť roviny ohraničená hranami grafu, ktorá neobsahuje žiadne body patriace hranám, ktoré ju neohraničujú. **Vonkajšia oblasť** je časť roviny, ktorá vznikne odčítaním všetkých vnútorných oblastí od roviny.

Kresba grafu priamymi hranami (straight line drawing) je taká kresba, ktorej krivky tvoriace hrany grafu sú úsečky. Oblasti takéhoto grafu sú zjavne polygóny.

Kresba grafu je **planárna**, ak prienikom ľubovoľných dvoch kriviek je prázdna množina. Graf je **planárny**, ak existuje aspoň jedna jeho kresba, ktorá je planárna.

1.3 Vybrané grafy

Strom je acyklický súvislý graf.

Nech V je konečná množina izolovaných bodov z dvojrozmerného euklidovského priestoru E^2. **Voronoiov diagram** je také rozdelenie E^2, ktoré priradí každému bodu $X \in E^2$ vrchol $v \in V$, práve vtedy, keď vzdialenosť X od v je menšia alebo rovná, než vzdialenosť X od ľubovoľného bodu z V.

Body, ktoré sú rovnako vzdialené od aspoň dvoch bodov nazývame **hraničné body**.

Úsečky tvorené hraničnými bodmi sa nazývajú **hrany diagramu**. A body, ktoré vzniknú prienikom týchto úsečiek sa nazývajú **vrcholy diagramu**.

Ak žiadne štyri vrcholy diagramu neležia na jednej kružnici, potom hovoríme, že Voronoiov diagram je **nedegenerovaný** a vrcholy diagramu patria práve trom hranám.

Duálny graf planárneho grafu G obsahuje vrchol pre každú oblasť grafu G a hranu, ktorá spája dve oblasti oddelené hranou.

Planárnu kresbu grafu priamymi hranami nazveme **triangulációu**, ak každá oblasť kresby grafu, okrem vonkajšej, tvorí trojuholník.

Delaunayova triangulácia (DT) je taká triangulácia grafu, že žiadne štyri vrcholy nepatria vnútru opísanej kružnice ľubovoľného trojuholníka. Delaunayova triangulácia tvorí duálny graf k nedegenerovanému Voronoiovmu diagramu.

Kapitola 2

Hranová algebra a DT

V tejto kapitole opíšeme algoritmus, ktorý vytvorili Der-Tsai Lee a Bruce Jay Schachter na výpočet Delaunayovej triangulácie [15] a upravenú verziu dátovej štruktúry, ktorú publikovali v roku 1985 Leonidas Guibas a Jorge Stolfi, pomocou ktorej tento algoritmus implementovali [12]. Tento algoritmus nájde Delaunayovu trianguláciu s časovou zložitosťou $O(n \log n)$.

2.1 Quad-edge

Dátová štruktúra quad-edge slúži na reprezentovanie kresieb planárnych grafov priamymi čiarami. Algoritmus implementujeme v jazyku TypeScript a preto uvádzame jej objektovo-orientovanú verziu.

2.1.1 Triedy, ich atribúty a metódy

Trieda **vertex** (vrchol) obsahuje informácie o hodnote súradníc vrchola.

Neorientované hrany spájajú práve dva vrcholy a práve dve polygónové oblasti. Určené sú dvomi orientovanými hranami a dvomi duálnymi orientovanými hranami. Pôvodná dátová štruktúra určuje hranu ďalšími štyrmi orientovanými hranami, ktoré reprezentujú zrkadlový obraz vyššie spomenutých orientovaných hrán. Na výpočet DT nie sú potrebné a preto ich neimplementujeme.

Trieda **edge** (hrana) reprezentuje tieto orientované hrany a obsahuje nasledujúce atribúty:

org Vrchol, z ktorého hrana vychádza.

oNext Hrana, ktorá nasleduje proti smeru hodinových ručičiek okolo začiatočného bodu a jej začiatočný bod je zhodný so začiatočným bodom hrany.

rot Duálna hrana, ktorá vychádza z pravej oblasti a vchádza do ľavej.

Trieda implementuje nasledujúce metódy, ktoré vznikajú viacnásobným použitím predchádzajúcich atribútov:

Metódy vracajúce jednotlivé reprezentácie hrany

sym Hrana, ktorá vychádza z koncového vrcholu hrany a vchádza do začiatočného vrcholu.

symRot Duálna hrana, ktorá vychádza z ľavej oblasti a vchádza do pravej.

Metódy vracajúce hrany nasledujúce v CCW poradí

rNext Hrana, ktorá nasleduje okolo pravej oblasti a jej pravá oblasť je zhodná s pravou oblasťou hrany.

lNext Hrana, ktorá nasleduje okolo ľavej oblasti a jej ľavá oblasť je zhodná s ľavou oblasťou hrany.

dNext Hrana, ktorá nasleduje okolo koncového bodu a jej koncový bod je zhodný s koncovým bodom hrany.

Metódy vracajúce hrany nasledujúce v CW poradí

rPrev Hrana, ktorá nasleduje okolo pravej oblasti a jej pravá oblasť je zhodná s pravou oblasťou hrany.

lPrev Hrana, ktorá nasleduje okolo ľavej oblasti a jej ľavá oblasť je zhodná s ľavou oblasťou hrany

oPrev Hrana, ktorá nasleduj okolo začiatočného bodu a jej začiatočný bod je zhodný s koncovým bodom hrany.

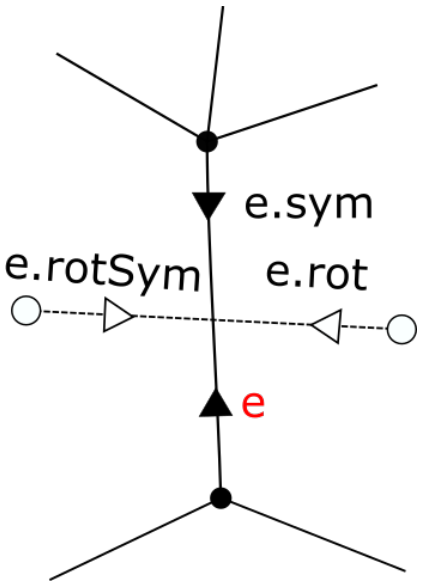
dPrev Hrana, ktorá nasleduje okolo koncového bodu a jej koncový bod je zhodný so začiatočným bodom hrany.

2.2 Základné funkcie na grafe

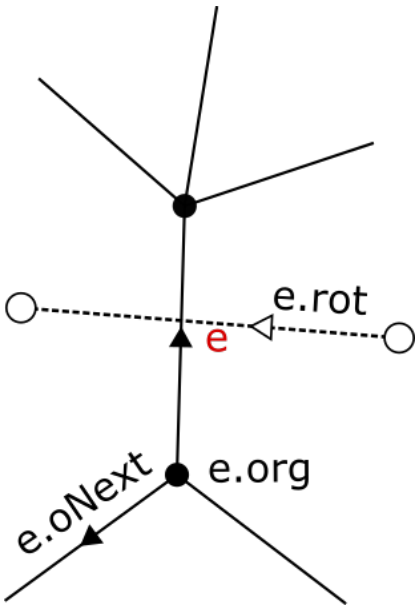
2.2.1 makeEdge()

Funkcia *makeEdge*() nedostáva na vstup žiadne parametre a jej funkciou je vytvoriť a vrátiť novú orientovanú hranu e, ktorá je v konzistentnom stave.

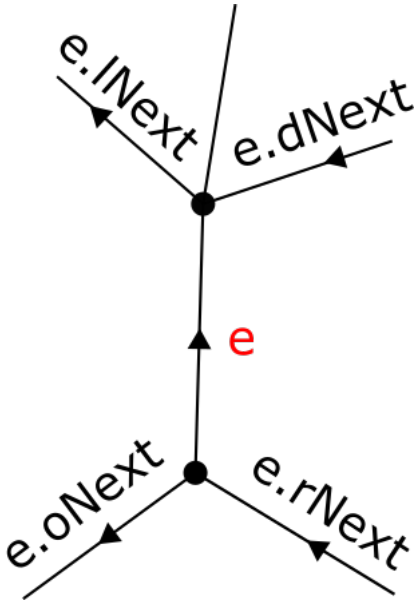
Funkcia vytvorí hranu e; hranu e_2, ktorá je symetrická k hrane e a dve duálne hrany e_1 a e_3, ktoré vychádzajú z vonkajšej oblasti.



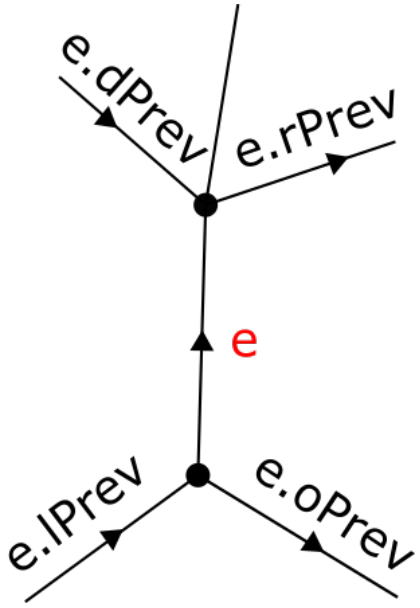
(a) Orientované hrany



(b) Vlastnosti triedy



(c) Metódy vracajúce v ccw poradí



(d) Metódy vracajúce hrany v cw poradí

Obr. 2.1: Quad-Edge

Kedže e a e_2 nie sú pripojené ku zvyšku grafu a $e.sym = e_2$, potom platí, že $e.oNext = e$ a $e_2.oNext = e_2$. Hrany e_1 a e_3 vychádzajú z rovnakej oblasti a nie sú súčasťou zvyšku dátovej štruktúry, a teda platí: $e_1.oNext = e_3$ a $e_3 = e_1.oNext$.

Algorithm 1 Funkcia *makeEdge*

procedure MAKEEDGE
  Vytvor nové hrany e, e_1, e_2, e_3.
  Nastav hranám hodnoty $oNext$
  Nastav hranám hodnoty rot
  return e
end procedure

2.2.2 splice(a, b)

Funkcia $splice(a, b)$ dostáva na vstup dve hrany a nevracia nič, namiesto toho upravuje parametre hrán a a b. Je určená na rozdeľovanie, spájanie a iné úpravy štruktúry.

Postupnosť hrán, vychádzajúcich z jedného vrcholu o alebo postupnosť duálnych hrán vychádzajúcich z jednej oblasti o nazveme **kruh** okolo o.

Ak hrany patria rovnakému kruhu, kruh je rozdelený na dve časti. Ak hrany patria rozdielnym kruhom, funkcia ich spojí do jedného.

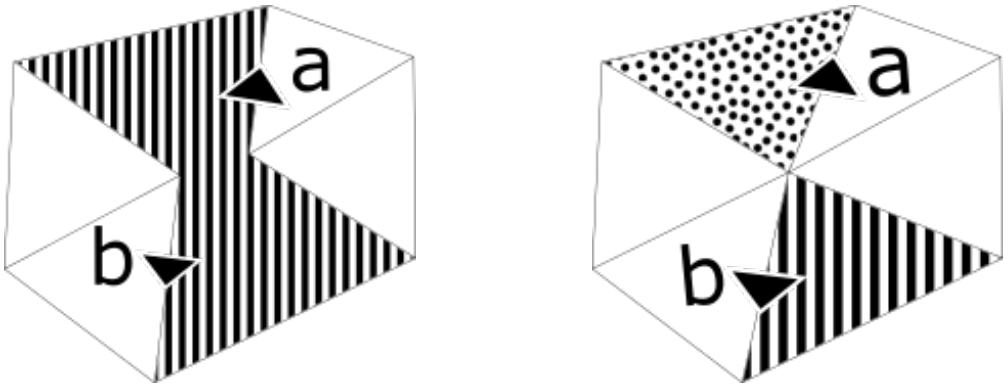
Hrany a a b určujú, kde sa má táto operácia vykonať. Pre kruhy okolo $a.org$ a $b.org$ nastane delenie alebo spájanie ihneď po a a b proti smeru hodinových ručičiek. Pre kruhy $a.oNext.rot$ a $b.oNext.rot$ sa funkcia vykoná po $a.oNext.rot$ a $b.oNext.rot$.

Funkciu vieme zadefinovať ako jednoduché úpravy hodnôt $oNext$ a $oNext.rot.oNext$. Funkcia $splice(a, b)$ je inverzná sama k sebe.

Najjednoduchší prípad nastane, ak hrany majú rovnaký počiatok a rôzne ľavé oblasti, ktorý ilustrujeme na obrázku 2.2. Funkcia rozdelí kruh okolo počiatku a spojí kruhy ich ľavých oblastí. Ak hrany majú rovnakú ľavú oblasť a rôzne počiatky, potom je výsledkom funkcie opak: funkcia rozdelí kruhy okolo ľavej oblasti a spojí kruhy ich počiatkov.

Algorithm 2 Funkcia *splice*

procedure SPLICE(a, b)
  $\alpha \leftarrow a.oNext.rot$
  $\beta \leftarrow b.oNext.rot$
  SWAP(a.oNext, b.oNext)
  SWAP($\alpha.oNext, \beta.oNext$)
end procedure



(a) $a.Left = b.Left$
$a.org \neq b.org$

(b) $a.Left \neq b.Left$
$a.org = b.org$

Obr. 2.2: Ilustrácia funkcie $splice(a, b)$

2.2.3 connect(a,b)

Funkcia $connect(a, b)$ dostane na vstup dve hrany, vytvorí novú hranu, ktorá spája $a.dest$ s $b.org$, aby platilo: $e.symRot.org = a.symRot.org = b.symRot.org$. Funkcia vytvorenú hranu vráti.

Algorithm 3 Funkcia *connect*

```
procedure CONNECT(a, b)
    e ← MAKEEDGE()
    e.org ← a.dest
    e.dest ← a.org
    SPLICE(e, a.lNext)
    SPLICE(e.sym, b)
    return e
end procedure
```

2.2.4 deleteEdge(e)

Funkcia $deleteEdge(e)$ oddelí hranu e od zvyšku štruktúry a môže rozdeliť graf na dva komponenty. Volanie $deleteEdge(connect(a, b))$ pridá hranu, hranu odstráni a graf vráti do pôvodného stavu, a teda funkcia $deleteEdge(e)$ je opakom $connect(a, b)$.

Algorithm 4 Funkcia *deleteEdge*

```
procedure DELETEEDGE(e)
    SPLICE(e, e.oPrev)
    SPLICE(e.sym, e.sym.oPrev)
end procedure
```

2.3 Pomocné funkcie na výpočet DT

2.3.1 Testovanie poradia

Pri výpočte Delaunayovej triangulácie nám vo viacerých prípadoch záleží na poradí vrcholov. Nasledujúca funkcia vráti *true*, ak budú vrcholy A, B a C usporiadané proti smeru hodinových ručičiek.

$$ccw(A, B, C) = \det \begin{vmatrix} A_x & A_y & 1 \\ B_x & B_y & 1 \\ C_x & C_y & 1 \end{vmatrix} > 0$$

Pomocou $ccw(A, B, C)$ implementujeme nasledujúce funkcie, ktoré využijeme na výpočet Delaunayovej triangulácie:

leftOf(A, e) Vráti *true*, ak sa bod A nachádza naľavo od hrany e.

rightOf(A, e) Vráti *true*, ak sa bod A nachádza napravo od hrany e.

valid(e, basel) Vráti *true*, ak sa $e.dest$ nachádza napravo od hrany *basel*.

2.4 Testovanie bodu vo vnútri opísanej kružnice

Funkcia $inCircleTest(A, B, C, D)$ vracia *true*, práve vtedy, keď bod D leží vo vnútri opísanej kružnice trojuholníka ABC a nachádza sa naľavo od trojuholníka. K výsledku funkcie sa dopracujeme nižšie uvedeným spôsobom [12].

$$inCircleTest(A, B, C, D) = \det \begin{vmatrix} A_x & A_y & A_x^2 + A_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ C_x & C_y & C_x^2 + C_y^2 & 1 \\ D_x & D_y & D_x^2 + D_y^2 & 1 \end{vmatrix} > 0$$

2.5 Výpočet DT

Na výpočet Delaunayovej triangulácie využijeme metódu **rozdeľuj a panuj**. Na vstupe máme množinu súradníc vrcholov, ktorá obsahuje aspoň štyri vrcholy a pred-

pokladáme, že žiadne štyri neležia na jednej kružnici.

Na začiatku usporiadame vrcholy vzhľadom na ich x-ové súradnice.

Usporiadanú množinu vrcholov **delíme na polovicu** vzhľadom na x-ové súradnice vrcholov, kým žiadna z nich neobsahuje menej ako tri vrcholy. Nájsť Delaunayovu trianguláciu v týchto podmnožinách je triviálne, kedže existuje len jediná možnosť, ako z týchto vrcholov vytvoriť súvislý graf.

Algorithm 5 Delaunayova triangulácia - časť 1

```
procedure DELAUNAY(S)
    if S = 2 then
        Vytvor hranu a
        return a, a.sym
    else if S = 3 then
        if CCW(ABC) then
            Vytvor trojuholník ABC
            return [a, b.sym]
        else if CCW(ACB) then
            Vytvor trojuholník ABC
            return [c.sym, c]
        else
            Vytvor hranu a
            Vytvor hranu b
            return [a, b.sym]
        end if
    else
        [ldo, ldi] ← DELAUNAY(S.splice(0, S.length/2))
        [rdi, rdo] ← DELAUNAY(S.splice(length/2))
```

V ďalších krokoch **spájame** podgrafy nasledovným spôsobom: Nech L sú hrany z ľavého podgrafu, P sú hrany z pravého podgrafu a S sú hrany, ktoré vzniknú spájaním podgrafov. Ako prvú vytvoríme hranu medzi vrcholom s najmenšou ypsilónovou súradnicou z ľavej a z pravej strany, ktorá nepretína žiadnu doteraz nájdenú hranu. Túto hranu nazveme základná hrana (**basel**), jej ľavý vrchol A a pravý vrchol B. Následne postupujeme zdola nahor, postupne pridávajúc hrany.

Kedže oblasti triangulácie tvoria trojuholníky, ďalšia hrana musí obsahovať vrchol buď z L alebo P a druhý vrchol zo základnej hrany. Nájdime vhodného potenciálneho kandidáta K z pravej strany. Ako prvý testujeme vrchol, ktorý je susedný s B a uhol $\angle ABK$ je v smere chodu hodinových ručičiek menší, než uhol $\angle ABX$, kde X je ľubovoľný vrchol susedný s B. Overme nasledovné podmienky:

1. $\angle ABK < 180$

2. Kružnica opísaná trojuholníku AKB neobsahuje žiadny iný vrchol z P.

Ak testovaný vrchol nespĺňa prvú podmienku, ukončíme hľadanie potenciálneho kandidáta z pravej strany. Ak nespĺňa druhú podmienku, vymažeme hranu, ktorá spája B a K, vyberieme nový vrchol a pokračujeme v hľadaní. Ľavého kandidáta nájdeme obdobne.

Algorithm 6 Delaunayova triangulácia - časť 2

```
        Vytvor základnú hranu - basel
        while true do
            Nájdi ľavého kandidáta - lcand
            Nájdi pravého kandidáta - rcand
            if !VALID(lcand) AND !VALID(rcand) then
                EXIT;
            else if !VALID(lcand) OR
                (VALID(rcand) AND
                INCIRCLE(lcand.dest, lcand.org,rcand.org, rcand.dest) ) then
                basel ← CONNECT(rcand, basel.Sym)
            else
                basel ← CONNECT(basel.sym, lcand.Sym)
            end if
        end while
    end if
end procedure
```

Ak neexistuje žiaden kandidát, potom je spájanie ukončené. Ak je vybraný jeden kandidát, potom je daná hrana pridaná do S. Nech sú teraz vybraní dvaja kandidáti, ak pravý kandidát leží vo vnútri opísaného trojuholníka určeného ľavým kandidátom a základnou hranou, potom pridáme hranu určenú pravým kandidátom a ľavým vrcholom základnej hrany, a naopak. Práve pridanú hranu označíme ako základnú a pokračujeme v pridávaní hrán.

Kapitola 3

Návrh

Cieľom bakalárskej práce je zistiť, či vybrané dáta tvoria podgrafy Delaunayovej triangulácie. Na spracovanie dát sme sa rozhodli vytvoriť jednoduchý grafický editor, ktorého úlohou bude umožniť používateľovi pohodlne a rýchlo vytvárať kresby grafov priamymi hranami. Grafický vstup sme zvolili, pretože budeme spracovávať rastrové obrázky a kvôli jednoduchosti projektu je potrebné zisťovať vrcholy ručne. Grafický editor nám umožní zistiť polohu vrcholu a vytvoriť jeho reprezentáciu jedným kliknutím.

3.1 Výber programovacieho jazyka

Súčasné desktopové operačné systémy obsahujú predinštalovaný webový prehliadač. Kvôli snahe požadovať od používateľa minimálne nároky sa implementovanie programu v jazykoch na tvorbu webových aplikácií zdá najprijateľnejšia cesta. Zároveň nám tento prístup umožní používať aplikáciu nezávisle na operačnom systéme počítača. Projekty, ktoré opisujeme v tejto kapitole, umožňujú vytvárať rozsiahle aplikácie, bežiace na strane klienta.

Vďaka technologickému pokroku sa rýchlosť internetových služieb výrazne zrýchlila, a preto nie je problém zverejniť aplikáciu ako súčasť webovej stránky. Pretože aplikácia beží jedine na strane klienta, aplikáciu je možné uložiť priamo do pamäte počítača a používať ju bez prístupu k internetovému pripojeniu. Webovú aplikáciu programujeme v jazyku TypeScript a modulový systém píšeme v štandarde ECMAScript 6, ktoré opisujeme v nasledujúcej časti.

3.2 Použité technológie

3.2.1 HTML

Na tvorbu štruktúry a obsahu webových stránok a aplikácií sa primárne používa značkovací jazyk HTML. Najnovší štandard jazyka HTML sa nazýva HTML5 [6].

Jedným z jeho hlavých cieľov je pridať priamu podporu multimédií. Do jazyka pridáva nové značky ako $< video >$ alebo $< canvas >$. **Plátno (canvas)** umožnuje dynamické a programovateľné vykresľovanie základných objektov (napríklad krivky alebo oblúky) a obrázkov.

3.2.2 CSS

Na opis vzhľadu HTML dokumentov slúži jazyk CSS [19]. Jeho cieľom je oddelenie obsahu od vzhľadu a vďaka tomu dosiahnuť čistý a prehľadný kód. Napríklad umožňuje zdieľať jeden vzhľad viacerými dokumentmi, alebo naopak nastaviť viacero vzhľadov pre rôzne typy zariadení.

3.2.3 ECMAScript

ECMAScript je špecifikácia vysoko-úrovňového programovacieho jazyka štandardizovaného organizáciou Ecma International [3]. Vývoj ECMAScriptu začal v roku 1996 a Brendan Eich publikoval prvú edíciu v roku 1997. ECMAScript bol pôvodne určený na tvorbu skriptov slúžiacich na oživenie stránok vo webových prehliadačoch. Stále viac sa však používa ako univerzálny jazyk a kvôli tomu sa rozšírili aj funkcie a možnosti jazyka.

V našej bakalárskej práci používame šiestu edíciu - ECMAScript 2015. Jedná sa zatiaľ o najrozsiahlejšiu aktualizáciu. Jej cieľom je pridať lepšiu podporu pre veľké aplikácie, tvorbu knižníc a použitie ECMAScriptu ako cieľ kompilácie ostatných jazykov. Aktualizácia pridáva novú syntax určenú na tvorbu komplexných aplikácií. Napríklad deklaráciu tried, moduly, iterátory alebo deklaráciu konštánt.

3.2.4 JavaScript

Najznámejšou implementáciou ECMAScriptu je JavaScript. Jedná sa o interpretovaný, objektovo-orientovaný, dynamický programovací jazyk. Spoločne s HTML a CSS patrí k základným webovým technológiám. Jeho hlavnou úlohou je vytvárať interaktívne webové stránky. Webové prehliadače obsahujú JS enginy, ktoré kód interpretujú.

3.2.5 TypeScript

TypeScript je open-source programovací jazyk spravovaný firmou Microsoft. Jedná sa o syntaktickú nadmnožinu JavaScriptu, ktorá pridáva voliteľné statické typovanie [7]. Taktiež podporuje šiestu edíciu ECMAScriptu a celú jej funkčnosť. Jednou z najväčších výhod je umožnenie vývojovému prostrediu odhaľovať chyby už počas písania kódu. Okrem základných typov TypeScript podporuje aj union, intersection, generické programovanie a hybridné rozhrania.

TypeScript kompilátor transkompiluje kód do JavaScriptu, ktorý je potom podporovaný webovými prehliadačmi. Kedže sa jedná o nadmnožinu JavaScriptu, existujúci JavaScriptový kód je taktiež validný TypeScript kód. TypeScript podporuje definičné súbory (definition files), ktoré obsahujú informácie o existujúcich JavaScriptových knižniciach. To umožňuje používať knižnice rovnako, ako keby boli staticky typované v TypeScripte.

3.2.6 Document Object Model

DOM je rozhranie pre programovanie aplikácií, ktoré reprezentuje súbory značkovacích jazykov v stromovej štruktúre [5]. V každom liste sa nachádza objekt, reprezentujúci časť dokumentu. Pri načítavaní stránky vytvorí webový prehliadač jej objektovo-orientovanú verziu, ktorá slúži ako rozhranie medzi JavaScriptom a HTML dokumentom. HTML objektom umožňuje pridávať event handlery, a tak vytvárať dynamické webové stránky.

3.2.7 npm

Nástroj npm je správca balíčkov pre programovací jazyk JavaScript [2]. Obsahuje online databázu balíčkov, ku ktorým klient pristupuje cez príkazový riadok. Zoznam balíčkov je verejne dostupný na webovej stránke projektu. Balíčky využívajú CommonJS formát modulov, súbor so základnými údajmi je vo formáte JSON a .npmignore súbor určuje, ktoré súbory nie sú súčasťou balíčka, ako napríklad súbory vývojového prostredia alebo TypeScript súbory.

3.2.8 CommonJS

Projekt CommonJS je iniciatíva, ktorá si kladie za cieľ vytvárať konvencie pre programovanie v JavaScripte mimo webového prehliadača [14] . Rastúca kolekcia obsahuje napríklad štandardy pre moduly alebo kódovanie znakov.

3.2.9 Browserify

Takmer žiadne webové prehliadače v súčasnosti nepodporujú ECMAScript alebo CommonJS moduly. Browserify je open-source JavaScriptový nástroj, ktorého úlohou je prekladať CommonJS moduly do kódu, ktorý je spustiteľný vo vnútri webového prehliadača [13]. Program rekurzívne prehľadáva jednotlivé moduly, ktoré zozbiera do jediného súboru a na použitie kódu v HTML súbore postačí jediné volanie funkcie $< script >$.

3.2.10 JSON

JSON je syntax určená na ukladanie a výmenu dát, ktoré sú čitateľné človekom [4]. Pôvodne slúžil na asynchrónny prenos informácií medzi serverom a webovým prehliadačom. Každý JavaScript-ový objekt je možné konvertovať do formátu JSON. Jedná sa však o dátový formát nezávislý na programovacom jazyku.

Má mnoho ďalších využití. V tejto bakalárskej práci využívame nasledovné JSON súbory. Súbor $tsconfig.json$, ktorý signalizuje, že daný priečinok je koreňom TypeScript projektu. Určuje súbory, ktoré majú byť kompilované a nastavuje kompilátor. Súbor $package.json$ obsahuje informácie o npm balíku, ako napríklad názov a verzia projektu.

3.3 Knižnica

Výpočet DT má mnohé praktické aplikácie. Používa sa napríklad na výpočet Euklidovskej minimálnej kostry. Preto sme sa rozhodli oddeliť kód grafického editora od implementácie algoritmu vytvorením knižnice.

Dôvodom implementovania novej knižnice bol nedostatok voľne dostupných knižníc počítajúcich DT v jazyku JavaScript. Pri hľadaní knižnice sme narážali na nasledujúce problémy: knižnice implementovali jednoduchý algoritmus, ktorý počíta DT s horšou časovou zložitosťou; algoritmus bol súčasťou veľkých geometrických knižníc; knižnice neobsahovali definičné súbory, a teda primárne nepodporovali TypeScript alebo neboli rozdelené do modulov, čo malo za následok neprehľadnosť kódu.

Kapitola 4

Implementácia knižnice

V tejto časti opíšeme knižnicu slúžiacu na vypočet Delaunayovej triangulácie.

4.1 Moduly

Knižnica je rozdelená do nasledujúcich modulov:

vertex.ts Modul obsahuje triedu $Vertex$, ktorá reprezentuje bod v dvojrozmernom priestore.

edge.ts Modul obsahuje triedu $Edge$, ktorá implementuje jednu orientovanú hranu z dátovej štruktúry $quad - edge$ a prístupové (*getter* a *setter*) funkcie.

edgeFunctions.ts Modul obsahuje funkcie, ktoré slúžia na úpravy dátovej štruktúry. Implementuje funkcie, ktoré sme opísali v kapitole 3.

triangle.ts Modul obsahuje triedu $Triangle$, ktorá predstavuje trojuholník. Implementovaná je pomocou troch vrcholov.

edgeError.ts Modul slúži na vyhodenie chyby, ak niekto žiada prístup k nedefinovanej hrane. Obsahuje triedu $EdgeError$ a funkciu $checkUndefined$, ktorá vyhodí $EdgeError$, ak je vstupný parameter nedefinovaný, inak ho vráti.

delaunay.ts Modul obsahuje funkcie potrebné na výpoćet Delaunayovej triangulácie. Implementuje funkcie, ktoré sme opísali v X. Funkcie $delaunayEdges$, ktorá vypočíta trianguláciu pomocou dátovej štruktúry $quad - edge$ a funkciu $delaunay$, ktorá prehľadáva dátovú štruktúru vytvorenú funkciou $delaunayEdges$ a vracia pole trojuholníkov tvoriacich trianguláciu.

index.ts Jedná sa o štandardný súbor určujúci vstupný bod TS projektu. Exportuje časti, ktoré sú dostupné používateľovi. V našej knižnici exportuje triedu $Triangle$ a funkciu $delaunay$.

4.2 Publikovanie a inštalácia knižnice

Knižnica bola publikovaná do správcu balíkov npm pod licenciou ISC, ktorá sa nachádza v elektronickej prílohe. Názov knižnice znie: **delaunayguibasstolfi**. Odporúčaný spôsob inštalácie knižnice je používať npm. Knižnicu nainštalujeme nasledovným volaním príkazového riadku, v koreni projektu:

npm install delaunayguibasstolfi

Príkaz vytvorí priečinok **node_modules** (ak ešte neexistuje) a stiahne balíček do novovzniknutého priečinku.

Na využívanie knižnice v TS súbore je potrebné ju importovať do súboru, v ktorom ju plánujeme použiť, príkazom:

import { Vertex, delaunay, Triangle} from "delaunayguibasstolfi"

Kapitola 5

Implementácia aplikácie

V tejto kapitole opíšeme implementáciu grafického editoru, ktorý sme vyvinuli a použili na kreslenie reprezentovaných dát.

5.1 Zoznam susedov

Na reprezentáciu grafu používame zoznam susedov, jedná sa o dátovú štruktúru, ktorá reprezentuje neorientovaný graf. Implementujeme ju pomocou objektu **Map**, ktorý musí byť implementovaný JS enginom ako hašovacia tabuľka alebo pomocou mechanizmov, ktoré poskytujú čas prístupu k prvku so sublineárnou časovou zložitosťou [3]. Predpokladajme, že používame JS engine, prekladajúci objekt *Map* do hešovacej tabuľky.

Údajová štruktúra pozostáva z dvojíc kľúčov a hodnôt, kde kľúče tvoria vrcholy a hodnoty sú vrcholy susedné ku kľúčom. Najväčšou výhodou je pridávanie vrcholov a hrán v priemernom konštantom čase. Na odstránenie vrcholu v potrebujeme prejsť každý vrchol a ak záznam obsahuje v, potom ho zo záznamu odstránime, preto túto operáciu vykonáme s priemernou časovou zložitosťou $O(|E|)$, kde $|E|$ je počet hrán v grafe. Na odstránenie hrany $\{v1,v2\}$ musíme prejsť hodnotami vrchola $v1$ a odstrániť z nich vrchol $v2$ a prejsť hodnotami $v2$ a odstrániť z nich vrchol $v1$, preto je priemerná časová zložitosť mazania vrcholu $O(|V|)$, kde $|V|$ je počet vrcholov.

Ak predpokladáme, že užívateľ bude vrcholy a hrany pridávať často — na rozdiel od ich mazania, potom je táto štruktúra najvhodnejšia pre naše potreby.

5.2 Formát grafu

Grafy ukladáme a načítavame pomocou nasledujúcej textovej reprezentácie: Formát začína zoznamom vrcholov. Každý vrchol je reprezentovaný jedným riadkom, ktorý začína identifikačným číslom (id) vrchola, oddeleného od jeho súradníc dvojbodkou.

Súradnice od seba oddeľujeme čiarkou. Na ďalšom riadku sa nachádza symbol # a zvyšné riadky tvoria zoznam hrán v grafe. V každom riadku sa nachádza dvojica id, ktorá určuje vrcholy spojené hranou.

5.3 Moduly

Aplikácia je rozdelená do nasledujúcich modulov:

index.ts Modul obsahuje event handlery interaktívnych prvkov na stránke.

graph.ts Modul obsahuje triedu **Graph**, ktorá implementuje dátovú štruktúru zoznam susedov.

point.ts Modul obsahuje triedu **Point**, ktorá reprezentuje vrchol grafu.

5.4 Publikovanie aplikácie

Aplikácia je dostupná na webovej stránke `http://www.st.fmph.uniba.sk/~sebestyen1/GraphDrawing/`. Taktiež je možné ju nainštalovať kompiláciou TS súborov, pomocou volania príkazového riadku:

tsc && browserify index.js -o graphBrowser.js

Skript skompiluje TS kód do JS súborov a ES6 moduly preloží do CommonJS modulov. Nástroj Browserify následne zozbiera CommonJS moduly do jedného JS súboru, ktorý následne importujeme do html súboru.

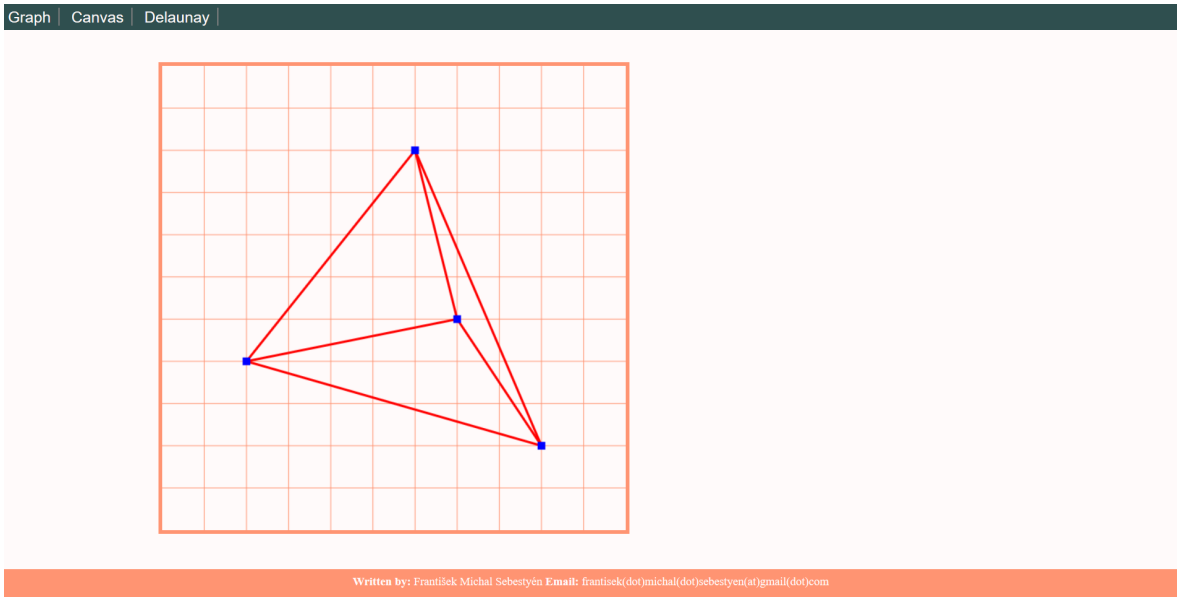
Obrázok 5.1 zobrazuje logo aplikácie. Jedná sa o kombináciu mena a symbolu. Názov aplikácie znie **Graph Drawing (kreslenie grafov)**. Symbol ilustruje Delaunayovu trianguláciu na množine štyroch bodov. Farba symbolu je červená — rovnaká, ako sa používa v aplikácii, na kreslenie hrán. Názov je napísaný oranžovou farbou a fontom IBM Plex Sans [1].

Obr. 5.1: Logo aplikácie

Kapitola 6

Práca s grafickým editorom

Grafický editor je rozdelený na plátno na ľavej strane a menu v hornej časti aplikácie.



Obr. 6.1: Snímka obrazovky pri práci s aplikáciou

6.1 Práca s plátnom

Plátno slúži na vykresľovanie kresby grafu priamymi hranami a pridávanie hrán a vrcholov. Na plátno dokážeme vykresliť mriežku, ktorá nám pomôže napríklad pri kreslení vertikálnych a horizontálnych čiar. Plátno ovládame nasledovne:

- Kliknutím ľavého tlačidla na plátno sa vytvorí nový vrchol na súradniciach určených pozíciou kurzora.
- Kliknutím ľavého tlačidla na vrchol označíme daný vrchol.

- Držaním klávesy control a kliknutím pravého tlačidla na vrchol sa daný vrchol vymaže.
- Držaním klávesy shift a kliknutím ľavého tlačidla na vrchol sa vytvorí hrana medzi daným a označeným vrcholom. Ak nie je označený žiaden vrchol, potom sa daný vrchol označí.
- Držaním kláves shift a control a kliknutím ľavého tlačidla na vrchol sa vymaže hrana medzi daným a označeným vrcholom.

6.2 Menu

Menu obsahujúce tri tlačidlá, ktoré ho rozdeľujú na skupiny podľa funkčnosti.

6.2.1 Graph

Táto skupina funkcií slúži na prácu používateľa s grafom.

Download Pomocou tlačidla stiahneme textovú reprezentáciu grafu.

Delete Tlačidlo slúži na vymazanie grafu.

Load graph Formulár slúži na načítanie grafu z lokálneho súboru.

6.2.2 Canvas

Pomocou tejto skupiny funkcií nastavujeme vlastnosti plátna.

Set grid Formulár slúži na nastavenie mriežky.

Set background Pomocou formulára sa načíta obrázok, ktorému sa proporcionálne zmení výška a šírka na veľkosť plátna a nastaví sa ako pozadie plátna.

Delete background Voľbou tlačidla sa pozadie vymaže.

6.2.3 Delaunay

Na prácu s knižnicou používame samostatnú sadu nasledujúcich tlačidiel:

Create DT Voľbou tlačidla sa z množiny vrcholov nakreslených na plátne vytvorí Delaunayova triangulácia, ktorá nahradí kresbu grafu na plátne.

Compare DT Pomocou tlačidla sa vyznačia hrany grafu, ktoré nie sú súčasťou DT na množine vrcholov.

Download comparison Stlačením tlačidla sa vytvorí súbor s textovou reprezentáciou grafu, Delaunayovej triangulácie; informáciami o počte ich hrán a ich prieniku, ktorý sa stiahne na disk.

Kapitola 7

Experimenty

V tejto kapitole opíšeme vybrané súbory dát, spôsob ich získavania, testy na dátach a výsledky experimentov.

Ako dáta sme volili grafémy používané rôznymi kultúrami, ktoré boli kreslené rovnými čiarami. Hypotézu, že staré kultúry preferovali hrany DT, preberáme z diplomovej práce Jerguša Moravčíka [16]. Naším cieľom bolo overiť, či nasledujúce súbory dát tvoria podgrafy Delaunayovej triangulácie. Preto sme z vybraných súborov vytvorili planárne kresby grafov. Kresby sme vytvárali nasledujúcim postupom: Krajné body úsečiek sme vyhlásili za vrcholy. Miesto, kde sa dve úsečky pretínajú, sme vyhlásili za vrchol. Nakoniec sme hranami spojili body ležiace na jednotlivých úsečkách.

7.1 Babylonská číselná sústava, kreslenie číslic

Klinové písmo vynašla Sumerská civilizácia a jedná sa o jeden z najstarších systémov písania. Znaky sú tvorené kombináciou rovných čiar a trojuholníkov. Jedným z príkladov klinového písma je Babylonská číselná sústava. Okolo roku 2000 p.n.l bola prevzatá od Sumerskej civilizácie, ktorá používala sexagesimálnu nepozičnú číselnú sústavu, ktorá sa v obmedzenej miere používa dodnes na meranie času alebo uhlov.

Babylonci pokračovali v používaní tejto číselnej sústavy, avšak zmenili ju na pozičnú, čo znamenalo veľký pokrok v matematike [8]. Neobsahovala žiaden symbol pre nulu a na jej reprezentáciu slúžilo vynechanie miesta o veľkosti číslice. Avšak napríklad číslo 60 bolo reprezentované rovnakou číslicou, ako číslo 1. To malo za následok, že význam číslice bolo nutné určiť z kontextu, v ktorom sa nachádzala. V neskorších babylonských textoch sa namiesto nuly používa špeciálny znak, reprezentujúci nulu, avšak iba na pravých okrajoch čísla.

Číslice sú tvorené dvomi základnými znakmi, 1 a 10. Čísla od 1 do 9 sú tvorené kombináciou znaku 1. Na reprezentáciu čísla bolo potrebné znak opakovať toľkokrát, koľko bola hodnota čísla. Najviac tri-krát v jednom riadku a zvyšok znakov začať

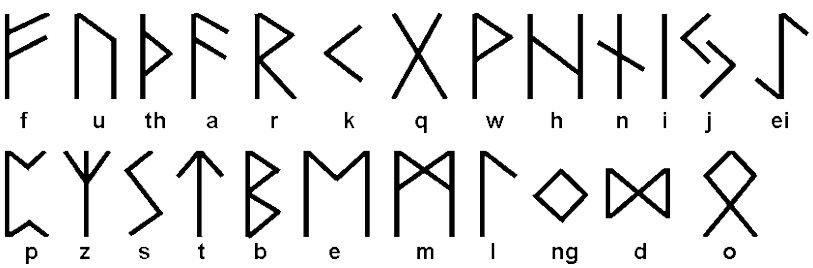
písať do nového riadku. Na reprezentáciu násobkov čísla 10 od 10 do 50 sa používalo opakovanie znaku 10. Znaky sa opakovali vertikálne, najviac tri-krát v jednom riadku, potom sa znaky začali zapisovať diagonálne do dvoch riadkov. Na zápis čísla, ktorý sa rovná súčtu násobku 10 a čísla od 1 do 9 zapíšeme číslicu reprezentujúcu násobok naľavo a číslicu reprezentujúcu číslo od 1 do 9 napravo [18].

Obr. 7.1: Na ľavej strane sú zapísane 4 znaky reprezentujúce násobky 10 v dvoch riadkoch. Na pravej strane štyri znaky reprezentujúce 1 zapísané v troch riadkoch

7.2 Starší Futhark, ťahy písma

Germánský Futhark prostý je najstarší spôsob zapisovanie runových znakov. Slúžil na písanie textu v Praseverskom jazyku, používaného severnými germánskymi kmeňmi. Runy boli nájdené na šperkoch, zbraniach alebo na runových kameňoch, ktoré slúžili napríklad ako pomníky. Písmo má pôvod v Archaickej latinke a predpokladá sa, že bolo vynájdené okolo prvého storočia, človekom alebo skupinov osôb, ktoré prišli do kontaktu s Rímskou kultúrou.

Názov písma je odvodený z pomenovanie prvých šiestich rún. Abeceda obsahuje 24 rún, ktoré sa často delia do troch skupín. Názvy jednotlivých rún sú odvodené zo slov z bežného života alebo majú pôvod v mytologii alebo v prírode. Pomenovania všetkých 24 rún a ďalších 5 anglosaských rún sú uchované v básni Old English rune poem, ktorá sa datuje do 8. storočia. Báseň nepomenováva runy priamo, namiesto toho každá strofa tvorí hádanku a odpoveďou na ňu je názov runy[9].



Obr. 7.2: Runy Starého Futharka s ich tradičnou transliteráciou

Číslica	Počet hrán grafu	Počet hrán DT	DT hrany v grafe	% DT hrán v grafe
1	4	6	4	100
2	8	18	8	100
3	12	32	12	100
4	16	48	15	93.75
5	20	62	18	90
6	24	76	24	100
7	28	90	27	96.43
8	32	104	30	93.75
9	36	124	36	100
10	5	10	5	100
20	10	26	10	100
30	15	44	15	100
40	20	68	18	90
50	25	84	23	92

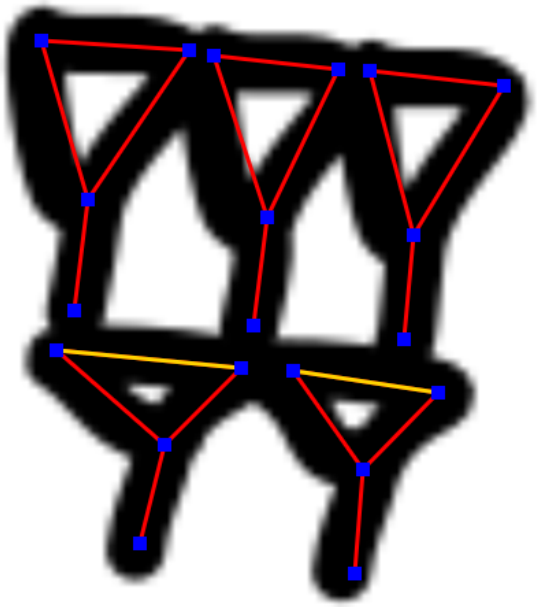
Tabuľka 7.1: Výsledky experimentov na dátach babylonských číslic

Názov runy	Hrany grafu	Hrany DT	DT hrany v grafe	% DT hrán v grafe
Fehu	5	9	4	80
Ur	2	3	2	100
Thurisaz	5	7	5	100
Ansuz	5	9	5	100
Raido	4	8	4	100
Kaunan	2	3	2	100
Gyfu	4	8	4	100
Wynn	5	6	5	100
Haglaz	5	11	5	100
Naudiz	4	8	4	100
Isaz	2	0	0	0
Jeran	4	12	4	100
Eihwaz	3	5	2	66.66
Pertho	5	12	5	100
Algiz	4	8	4	100
Sowilo	4	7	4	100
Tiwaz	3	6	2	66.66
Berkanan	6	9	6	100
Ehwaz	4	8	4	100
Mannaz	8	13	8	100
Laukaz	2	3	2	100
Ingwaz	4	6	4	100
Othalan	6	10	6	100
Dagaz	6	8	6	100

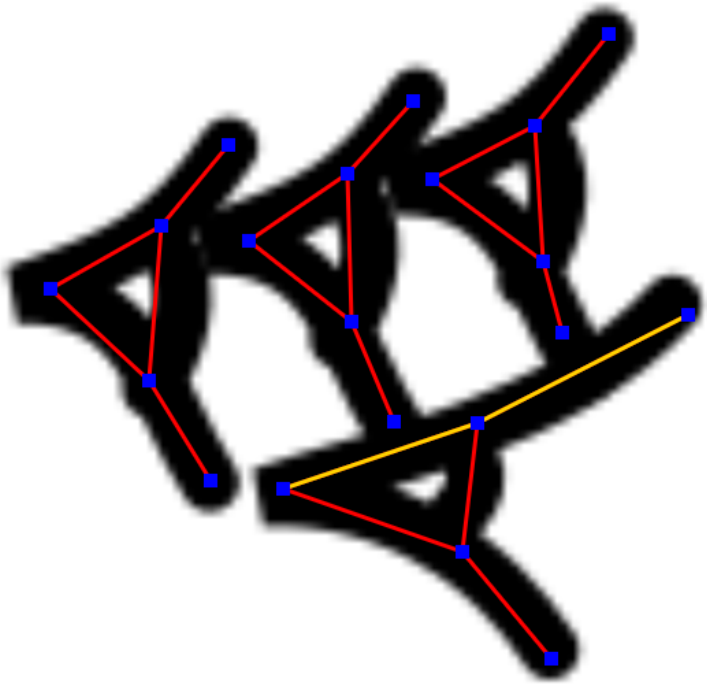
Tabuľka 7.2: Výsledky experimentov na dátach rún Starého Futharka

7.3 Výsledky experimentov

Jednotlivé runy a číslice nájde čitateľ v elektronickej prílohe. V tabuľke 7.1 uvádzame výsledky experimentov na datasete Babylonských číslic [20]. Hrany, ktoré nepatrili Delaunayovej triangulácií sa objavili v prípadoch, kedy sa znak 1 nachádzal vo viacerých riadkoch, ale počet znakov v poslednom riadku bol menší, než tri; a v prípade, kedy sa znak 10 nachádzal v dvoch riadkoch. Na obrázku 7.3 sa nachádzajú príklady číslic obsahujúce hrany, ktoré nepatria Delaunayovej triangulácii. Červenou farbou sú nakreslené hrany, ktoré patria DT a žltou hrany, ktoré nepatria DT. V tabuľke 7.2 sa nachádzajú výsledky experimentov na dátach rún [21].



(a) Číslica reprezentujúca číslo 5



(b) Číslica reprezentujúca číslo 40

Obr. 7.3: Príklady hrán, ktoré nepatria DT

Záver

V našej práci sme skúmali geometrické vlastnosti súborov dát z kultúrneho dedičstva starých civilizácií. Opísali sme algoritmus na výpočet Delaunayovej triangulácie, ktorý sme implementovali v programovacom jazyku TypeScript. Vytvorili sme webovú aplikáciu, slúžiacu na tvorbu kresieb grafov z vybraných dát a počítanie DT. Vybrali sme súbory dát: Babylonské číslice a Starý Futhrak. Navrhli sme spôsob ich reprezentovania pomocou planárnych kresieb grafu. Overili sme geometrické vlastnosti ich hrán. Na prácu je možné nadviazať rozšírením skúmania ďalších dát, akým je napríklad Kórejské písmo, Rozšíriť nástroj o funkcie, ktoré počítajú ďalšie geometrické algoritmy, ako napríklad Euklidovská minimálna kostra alebo o funkcie, ktoré vylepšujú prácu s grafickým editorom.

Literatúra

[1] IBM Plex. https://www.ibm.com/plex/.

[2] npm Documentation. Retrieved May 9, 2019, from `https://docs.npmjs.com/`.

[3] ECMAScript 2015 Language Specification. Standard, Ecma International, Geneva, CH, May 2015. Retrieved May 9, 2019, from `http://www.ecma-international.org/ecma-262/6.0/ECMA-262.pdf`.

[4] The JSON data interchange syntax. Standard, Ecma International, Geneva, CH, December 2017. Retrieved May 9, 2019, from `https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf`.

[5] DOM. Standard, WHATWG, April 2019. Retrieved May 9, 2019, from `https://dom.spec.whatwg.org`.

[6] HTML. Standard, WHATWG, May 2019. Retrieved May 9, 2019, from `https://html.spec.whatwg.org`.

[7] Peter Bright. Microsoft TypeScript: the JavaScript we need, or a solution looking for a problem?, March 2012. Retrieved May 9, 2019, from `https://arstechnica.com/information-technology/2012/10/microsoft-typescript-the-javascript-we-need-or-a-solution-looking-for-a-problem/`.

[8] Stephen Chrisomalis. *Numerical notation: A comparative history*. Cambridge University Press, 2010.

[9] Bruce Dickins. *Runic and Heroic Poems of the Old Teutonic Peoples*. Cambridge University Press, 1915.

[10] Andrej Ferko, Jerguš Moravčík, and Ivana Kolingerová. Souhvězdí jako podgrafy triangulací. *Pokroky matematiky, fyziky a astronomie*, 61(1):14–20, 2016.

[11] Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Tóth, editors. *Handbook of Discrete and Computational Geometry*. CRC Press, third edition, 2017.

[12] Leonidas Guibas and Jorge Stolfi. Primitives for the Manipulation of General Subdivisions and the Computation of Voronoi Diagram. *ACM Transactions on Graphics*, 4(3):74–123, 1985.

[13] James Halliday. Browserify Handbook, 2015. Retrieved May 9, 2019, from `https://github.com/browserify/browserify-handbook`.

[14] Kris Kowal. CommonJS effort sets JavaScript on path for world domination. 2009. Retrieved May 9, 2019, from `https://arstechnica.com/information-technology/2009/12/commonjs-effort-sets-javascript-on-path-for-world-domination/`.

[15] Der-Tsai Lee and Bruce J Schachter. Two algorithms for constructing a delaunay triangulation. *International Journal of Computer & Information Sciences*, 9(3):219–242, 1980.

[16] Jerguš Moravčík. O charakterizácii niektorých rovinných grafov. Diplomová práca, Univerzita Komenského v Bratislave, 1998.

[17] Takao Nishizeki and Md. Saidur Rahman. *Planar Graph Drawing*. World Scientific, 2004.

[18] Neeraj Anant Pande. Numeral systems of great ancient human civilizations. *Journal of Science and Arts*, 2(13):209, 2010.

[19] W3Schools. *CSS tutorial*. Retrieved May 9, 2019, from `https://www.w3schools.com/css/default.asp`.

[20] Wikipedia contributors. Babylonian numerals — Wikipedia, the free encyclopedia, 2019. [Online; accessed 19-May-2019].

[21] Wikipedia contributors. Elder futhark — Wikipedia, the free encyclopedia, 2019. [Online; accessed 19-May-2019].