

BABEŞ–BOLYAI UNIVERSITY OF CLUJ-NAPOCA  
FACULTY OF MATHEMATICS AND INFORMATICS  
SPECIALIZATION: COMPUTER SCIENCE

## **Diploma Thesis**

# **GRoutes - A practical application of the travelling salesman problem**

### **Abstract**

The present paper describes a practical application of the travelling salesman problem (TSP). In the first chapters I describe the theoretical background of the work, the definition of graphs, notions in graph theory, graph problems, algorithmic approaches, TSP algorithms, reduction of the asymmetric TSP to a symmetric one. The emphasis is on selecting an algorithm that matches, and serves our use case practically.

To exemplify the travelling salesman problem, I have developed an android application, with a Firestore database, Firebase Authentication and Maps API. The functionalities of the application include searching for locations, determining the best route (visiting every node), choosing the travelling mode (by car, on foot), automatically adapting the search algorithm, saving the searches, managing routes and places as favourites.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

2018

LORENZOVICI ZSOMBOR

ADVISOR:  
ASSIST PROF. DR. GASKÓ NOÉMI

BABEȘ-BOLYAI UNIVERSITY OF CLUJ-NAPOCA  
FACULTY OF MATHEMATICS AND INFORMATICS  
SPECIALIZATION: COMPUTER SCIENCE

**Diploma Thesis**

# **GRoutes - A practical application of the travelling salesman problem**



ADVISOR:

ASSIST PROF. DR. GASKÓ NOÉMI

STUDENT:

LORENZOVICI ZSOMBOR

2018

UNIVERSITATEA BABEȘ-BOLYAI, CLUJ-NAPOCA  
FACULTATEA DE MATEMATICĂ ȘI INFORMATICĂ  
SPECIALIZAREA INFORMATICĂ

Lucrare de licență

# **GRoutes - Aplicarea practică a problemei comis-voiajorului**



CONDUCĂTOR ȘTIINȚIFIC:  
LECTOR DR. GASKÓ NOÉMI

ABSOLVENT:  
LORENZOVICI ZSOMBOR

2018

BABEŞ–BOLYAI TUDOMÁNYEGYETEM KOLOZSVÁR  
MATEMATIKA ÉS INFORMATIKA KAR  
INFORMATIKA SZAK

**Szakdolgozat**

**GRoutes - Az utazó ügynök  
problémájának gyakorlati  
alkalmazása**



TÉMAVEZETŐ:

DR. GASKÓ NOÉMI,  
EGYETEMI ADJUNKTUS

SZERZŐ:

LORENZOVICI ZSOMBOR

2018

# Tartalomjegyzék

<b>1. Bevezetés</b>	<b>4</b>
<b>2. Gráfok - alapfogalmak</b>	<b>6</b>
<b>3. Gráf problémák</b>	<b>7</b>
3.1. Gráfok leszámmlálása . . . . .	7
3.2. Algráf izomorfizmus probléma . . . . .	7
3.3. Gráfok színezése . . . . .	7
3.4. Útproblémák . . . . .	7
3.4.1. Hamilton út probléma . . . . .	7
3.4.2. Minimális feszítőfa . . . . .	7
3.4.3. Kínaipostás-probléma . . . . .	7
3.4.4. Königsbergi hidak problémája . . . . .	7
3.4.5. Legrövidebb út probléma . . . . .	7
3.4.6. Steiner-fa probléma . . . . .	7
3.4.7. Három ház-három kút-probléma . . . . .	7
<b>4. Az utazó ügynök problémája (TSP)</b>	<b>8</b>
4.1. A Hamilton körrel való kapcsolat . . . . .	8
4.2. A probléma komplexitása . . . . .	8
4.3. Egzakt megoldások . . . . .	8
4.4. Approximációk . . . . .	8
4.5. Aszimmetrikus TSP . . . . .	8
<b>5. TSP algoritmusok</b>	<b>9</b>
5.1. Egzakt algoritmusok . . . . .	9
5.1.1. Brute force . . . . .	9
5.1.2. Held-Karp algoritmus . . . . .	9
5.1.3. Concorde algoritmus . . . . .	9
5.2. Heurisztikus algoritmusok . . . . .	9
5.2.1. Greedy - legközelebbi szomszéd . . . . .	9
5.2.2. Lin-Kernighan heurisztika . . . . .	9
<b>6. Technológiák</b>	<b>10</b>
6.1. Android . . . . .	10
6.2. Gradle . . . . .	11
6.3. Firebase . . . . .	11
6.4. Maps API . . . . .	11
<b>7. GRoutes - alkalmazás bemutatása</b>	<b>12</b>

## TARTALOMJEGYZÉK

7.1. Funkcionalitások . . . . .	12
7.2. Adatbázis . . . . .	12
7.3. Osztályok . . . . .	12
7.4. Felmerülő problémák . . . . .	12
<b>8. Következtetések</b>	<b>13</b>

## 1. fejezet

# Bevezetés

Dolgozatom témája az "utazó ügynök" problémájának (TSP)<sup>1</sup> a gyakorlatban történő alkalmazása. Az utazó ügynök problémája egy komputacionális optimalizálási probléma, amely az 1930-as évek óta nagyon intenzíven foglalkoztatja a tudományos világot. Egyszerűen megfogalmazva: adott valamennyi város, valamint az köztük levő távolságok. Melyik a legrövidebb lehetséges útvonal, ami egyszer érinti az összes várost, majd visszaérkezik a kiinduló pontba? Jelenleg nem létezik olyan polinomiális komplexitással rendelkező algoritmus, amely erre a problémára egzakt megoldást nyújtana.

Az utazó ügynök problémájának példázásának céljából egy android alkalmazást készítettem, amelyben különböző helyekre (csomópontokra) rákeresve eredményként egy olyan útvonal rajzolódik ki, amely az összes helyet érinti, valamint a legrövidebb is egyúttal. A felhasználó két utazási mód közül választhat: gyaloglás, autó. Az alkalmazás felhasználói célterülete a turistaútvonalak tervezői, a csomagkihordó szolgáltatást biztosító cégek, valamint a különböző eladási szakterülettel foglalkozó vállalkozások, ahol oda kell utazni az ügyfélhez.

Az első fejezetben a gráfelméleti alapfogalmakat fogom definiálni, részletezni. Ezek a későbbiekben fontosak lesznek, hogy egzakt módon taglalhassuk a témát. A következő fejezetben a különböző gráfelméleti problémákat, feladatokat fogom taglalni, majd ezt követően egy külön fejezetet szentelek az általam kiválasztott problémára, az "utazó ügynök" problémájára. Ebben a fejezetben meg kell magyaráznom a különböző komplexitási alapfogalmakat, hogy rávilágítsak a téma komputacionális nehézségeire. Arról is itt fog szó esni, hogy milyen módszerekkel közelítjük meg a problémát. A későbbiekben nagyító alá helyezem a különböző algoritmusokat az előbbieken említett megoldási stratégiák szerint osztályozva. Bizonyos algoritmusok csak szimmetrikus TSP esetén adnak helyes eredményt, így az asszimmetrikus TSP szimmetrikusra való visszavezetését is tárgyalni fogom.

A dolgozat gyakorlati része a projekt során felhasznált technológiák bemutatásával fog kezdődni. Ez fontos lehet azok számára, akik alapjaiban jobban meg szeretnék érteni az applikációt, vagy egy hasonló alkalmazás elkészítéséhez szeretnének hasznos ismereteket elsajátítani. A technológiák bemutatása után maga az alkalmazás nagyító alá helyezése következik. Ebben a részben fogok írni az applikáció különböző komponenseiről, arról, hogy ezek milyen kapcsolatban vannak egymással. Ezt osztálydiagrammal is fogom illusztrálni. Itt fogok részletesebben írni az adatbázisról, az autentikációról, a Maps API-ről<sup>2</sup>, az alkalmazás funkcionalitásairól, hogy milyen esetben melyik algoritmust alkalmazom és miért kell egyáltalán különböző esetekben más-más algoritmust használni, valamint a nehézségekről,

---

1. Travelling salesman problem

2. Application Programming Interface

## 1. FEJEZET: BEVEZETÉS

melyekbe munkám során ütköztem, és ezek megoldásairól.

Végül levonom a következtetéseket a munkám során szerzett tapasztalatokból, valamint javaslatokat teszek az alkalmazás jövőbeli továbbfejlesztési lehetőségeire.



## 2. fejezet

# Gráfok - alapfogalmak

**Összefoglaló:** Ebben a fejezetben a későbbiekben használt fogalmakat fogom definiálni.

Egy gráf  $G = (V(G), E(G))$  vagy  $G = (V, E)$  két véges halmazból áll.  $V(G)$  vagy  $V$ , egy nem üres halmaz, melynek az elemeit csúcsoknak nevezzük, ezek alkotják a gráf csúcsait.  $E(G)$  vagy  $E$ , egy halmaz, melynek elemeit éleknek nevezzük, ezek alkotják a gráf éleit, úgy, hogy minden  $e \in E$  élet meghatároz egy csúcs-pár  $(u, v)$ , melyeket  $e$  csúcsainak nevezünk.

**rend** – definíció szerint a  $G$  gráf rendje  $|V| = n$

**méret** – definíció szerint a  $G$  gráf mérete  $|E| = m$

## **3. fejezet**

# **Gráf problémák**

*Összefoglaló: A fejezetek elején egy rövid összefoglalót teszünk. Ez a rész opcionális.*

### **3.1. Gráfok leszámlálása**

### **3.2. Algráf izomorfizmus probléma**

### **3.3. Gráfok színezése**

### **3.4. Útproblémák**

#### **3.4.1. Hamilton út probléma**

#### **3.4.2. Minimális feszítőfa**

#### **3.4.3. Kínaipostás-probléma**

#### **3.4.4. Königsbergi hidak problémája**

#### **3.4.5. Legrövidebb út probléma**

#### **3.4.6. Steiner-fa probléma**

#### **3.4.7. Három ház-három kút-probléma**

## **4. fejezet**

# **Az utazó ügynök problémája (TSP)**

*Összefoglaló: A fejezetek elején egy rövid összefoglalót teszünk. Ez a rész opcionális.*

### **4.1. A Hamilton körrel való kapcsolat**

### **4.2. A probléma komplexitása**

### **4.3. Egzakt megoldások**

### **4.4. Approximációk**

### **4.5. Aszimmetrikus TSP**

## 5. fejezet

# TSP algoritmusok

*Összefoglaló: A fejezetek elején egy rövid összefoglalót teszünk. Ez a rész opcionális.*

## 5.1. Egzakt algoritmusok

### 5.1.1. Brute force

### 5.1.2. Held-Karp algoritmus

### 5.1.3. Concorde algoritmus

## 5.2. Heurisztikus algoritmusok

### 5.2.1. Greedy - legközelebbi szomszéd

### 5.2.2. Lin-Kernighan heurisztika

## 6. fejezet

# Technológiák

***Összefoglaló:** Ebben a fejezetben, a projekt elkészítése során felhasznált technológiákat ismertetem.*

### 6.1. Android

Az Android egy nyílt forráskódú, Linux kernel alapú többfelhasználós operációs rendszer, ahol minden applikáció egy külön felhasználó. Hivatalos nyelvei a Java és a Kotlin. Alapértelmezetten, a rendszer minden applikációnak kioszt egy egyedi Linux felhasználói ID-t (az ID-t csak a rendszer használja, és ismeretlen az applikáció számára). Az operációs rendszer úgy osztja ki a hozzáférési jogokat az applikáció állományai számára, hogy csak az a felhasználói ID férjen hozzájuk, amivel az adott applikáció rendelkezik. Minden folyamat (process) rendelkezik a saját virtuális gépével, tehát minden applikáció kódjá egymástól teljesen izoláltan fut. Alapértelmezetten minden applikáció a saját Linux folyamatában fut. Az Android operációs rendszer elindítja a folyamatot, amikor az applikáció valamelyik komponensének szüksége van rá, majd leállítja azt, mikor többé már nincs rá szükség, vagy ha a rendszernek memóriát kell lefoglalnia, más applikációk számára. Az Android operációs rendszer “a legkevesebb kiváltság elvét” (“the principle of least privilege”) alkalmazza, tehát alapértelmezetten, minden applikációnak csak azokhoz a komponensekhez van hozzáférése, amik szükségesek a feladatának elvégzéséhez, és semmi egyébhez.

Egy android applikációnak négy fajta komponense lehet: Activity, Service, Broadcast receiver, Content provider. Ezek közül az Activity szolgál a felhasználóval történő interakció eszközeként, ez ugyanis egy felhasználói felülettel rendelkező képernyő. Activity-k, Service-k és Broadcast receiver-ek egy aszinkron üzenet által aktiválódnak, amit Intent-nek (szándék) nevezünk.

Az Activity egy osztály (class), amely a logikát tartalmazza, a felhasználói felületért azonban egy ehhez tartozó XML állomány felel, ami a különböző UI elemeket (gombok, szövegdobozok, konténerek) tartalmazza.

Az AndroidManifest.xml egy konfigurációs állomány, ami a projekt gyökeri könyvtárában található. Itt vannak deklarálva az applikáció komponensei, valamint a szükséges hozzáférési jogokat is itt kell feltüntetni.

## 6.2. Gradle

A Gradle egy nyílt forráskódú projektépítő eszköz (build automation tool). A Groovy, vagy Kotlin nyelvű scriptekbe, megadhatjuk a projektünk külső függőségeit (például API-k, library-k), melyeket letölti, majd lekompillálja és lefordítja a forráskódot.

## 6.3. Firebase

A felhasználók adatainak, beállításainak tárolására, kezelésére a Firestore nevű no-sql (.json) alapú adatbázist használtam, mely a CRUD<sup>1</sup> műveletekhez is biztosít metódusokat. Logikai építőelemei a kollekciók (collection) és a dokumentumok (document). Az előbbi tartalmazhat dokumentumokat, míg az utóbbi alkollekciókat, vagy magukat az adatokat. Az adatok kulcs, érték párok, az értékek lehetnek számok, karakterláncok, tömbök, geopontok vagy akár sajátos osztályok. Amennyiben sajátos osztály akarunk használni, annak rendelkeznie kell egy publikus konstruktorral, melynek nincsenek paraméterei, valamint az attribútumokhoz kell tartozzon egy-egy publikus „getter”<sup>2</sup> metódus. Az adatbázisban található adatokhoz való hozzáférést egy szabállyal kell megadni, amelyet a szerver leellenőriz a CRUD műveletek végrehajtása esetén.

A felhasználók menedzselésére, mint a regisztráció, bejelentkezés, e-mail cím aktiválása, elfelejtett jelszó visszaállítása, a Firebase Authentication szolgáltatást használtam.

## 6.4. Maps API

A csomópontok közti távolságok mátrixának lekérdezésére a Distance Matrix API-t, valamint két csomópont közötti útvonal meghatározására a Directions API-t használtam.

---

1. create, read, update, delete

2. egy paraméter nélküli metódus, mely egy attribútumot térít vissza - példa: String getName()

## **7. fejezet**

# **GRoutes - alkalmazás bemutatása**

*Összefoglaló: A fejezetek elején egy rövid összefoglalót teszünk. Ez a rész opcionális.*

### **7.1. Funkcionalitások**

### **7.2. Adatbázis**

### **7.3. Osztályok**

### **7.4. Felmerülő problémák**

## 8. fejezet

# Következtetések

*Összefoglaló: A fejezetek elején egy rövid összefoglalót teszünk. Ez a rész opcionális.*



## Irodalomjegyzék