

**Szegedi Tudományegyetem**  
**Informatikai Tanszékcsoport**

**Minőségkinyerés borkóstolási adatokból, web és  
android alkalmazás fejlesztés**

Szakdolgozat

*Készítette:*  
**Varga Tamás**  
Programtervező Informatikus  
hallgató

*Témavezető:*  
**Dr. Csendes Tibor**  
tanszékvezető egyetemi tanár

Szeged  
2015

# Tartalomjegyzék

Feladatkiírás . . . . .	4
Tartalmi összefoglaló . . . . .	5
Bevezetés . . . . .	6
<b>1. Borkóstoló algoritmusok</b>	<b>7</b>
1.1. CoHITS . . . . .	7
1.2. Hamming . . . . .	9
1.3. Koszinusz . . . . .	9
1.4. Precedencia . . . . .	9
1.5. Összefüggőségi . . . . .	9
<b>2. A weboldal</b>	<b>11</b>
2.1. Iterációk . . . . .	12
2.1.1. Első mérföldkő . . . . .	12
2.1.2. Második mérföldkő . . . . .	12
2.1.3. Harmadik mérföldkő . . . . .	13
2.1.4. Negyedik mérföldkő . . . . .	14
2.1.5. Ötödik mérföldkő . . . . .	14
2.1.6. Hatodik mérföldkő . . . . .	14
2.1.7. Hetedik mérföldkő . . . . .	15
2.2. Adatbázis . . . . .	15
2.3. PHP . . . . .	15
2.4. JavaScript . . . . .	15
2.5. JQuery . . . . .	15
2.6. Grafikonok . . . . .	15
2.6.1. Google Charts . . . . .	15
2.7. Statikus tartalom . . . . .	15
2.7.1. index.php - Főoldal . . . . .	15
2.7.2. borkostolasEredmenyek.php - Eredmények . . . . .	15
2.7.3. modszerek.php - Módszerek . . . . .	15
2.7.4. kapcsolat.php - Kapcsolat . . . . .	15
2.8. Dinamikus tartalom . . . . .	15
2.8.1. Regisztráció . . . . .	15
2.8.2. Bejelentkezés . . . . .	15
2.8.3. demo.php - Demó . . . . .	15

<b>3. A mobil alkalmazás</b>	<b>16</b>
3.1. Android . . . . .	16
<b>4. A weboldal és a mobil alkalmazás összefűzése</b>	<b>17</b>
<b>5. Tesztelés</b>	<b>18</b>
5.1. Regisztrációs és bejelentkeztető rendszer . . . . .	18
5.2. Demo adatkezelésének ellenőrzése . . . . .	18
5.3. Algoritmusok ellenőrzése kis adatokon . . . . .	18
5.4. Algoritmusok ellenőrzése ismert eredményekkel . . . . .	18
<b>6. Összefoglalás</b>	<b>19</b>
<b>7. Egyebek</b>	<b>20</b>
7.1. Verziókezelés . . . . .	20
7.1.1. Git . . . . .	20
7.1.2. GitHub . . . . .	20
7.1.3. A választás . . . . .	21
7.2. Környezetek . . . . .	21
<b>8. Függelék</b>	<b>23</b>
8.1. A program forráskódja . . . . .	23
Nyilatkozat . . . . .	24
Irodalomjegyzék . . . . .	25

# Feladatkiírás

A hallgató feladata egy olyan web és hozzá tartozó android alkalmazás készítése, amely képes borkóstolás során gyűjtött bor értékelések alapján több különböző algoritmus használatával a kóstolók rangsorolására. A web és android alkalmazásoknak képeseknek kell lenniük egymás közötti szinkronizációra, több felhasználó kezelésére valamint a bevitt adatok és az alkalmazás állapotaink tárolására. Az android alkalmazásnak offline módban is használható kell hogy legyen.

# Tartalmi összefoglaló

**A téma megnevezése** ♦ Minőségkinyerés borkóstolási adatokból, web és android alkalmazás fejlesztés

**A feladat megfogalmazása** ♦ Web és android alkalmazás fejlesztése, amely képes bor értékelések alapján már létező algoritmusok által rangsorolni a borkóstolókat. A mobil alkalmazás felhasználó barát megvalósítása, mely bármilyen körülmény mellett valós idejű adat bevitelt is garantál.

**A megoldási mód** ♦ A web alkalmazás a következő címekről érhető el:

– `http://bor.tvarga.hu`

Az android alkalmazás letölthető bármelyik fent említett URL -ről a jobb oldali floating menün keresztül, vagy az alábbi közvetlen linken: `borkostolas.apk`

**Alkalmazott eszközök, módszerek** ♦

– Web alkalmazás:

Programozási nyelvek: PHP, JavaScript, MySQL

Fejlesztői környezetek: SublimeText, PhpStorm

– Android alkalmazás:

Programozási nyelvek: Java

Fejlesztői környezetek: Android Studio

– Verziókezelés: GitHub

**Kulcsszavak** ♦ Android, PHP, JavaScript, Borkóstolás

# Bevezetés

Borkóstolók rangsorolásával illetve minőségkinyeréssel valamint borversenyek lebonyolításával már előttem is többen foglalkoztak. Bár még van kutatni való ezen a területen, jól működő algoritmus már több is ismert ilyen jellegű feladatok megoldására. Ezért ennek a szakdolgozatnak nem is újabb algoritmusok kifejlesztése vagy már meglévők továbbfejlesztése a fő célja. Mivel található több olyan web alkalmazás is, mely egy specifikus egyszerű algoritmust vagy talán egy sokat tesztelt komplexebbet használ, mégsem létezik egy közismert, modern és felhasználóbarát alkalmazás, amely egy igényes felhasználói felületen keresztül bemutatná ezen algoritmusok működését lehetővé téve hogy nyomon kövessük akár saját laikus eredményeinket neves szakértőkhöz viszonyítva.

Mivel a mai világban a mobil eszközök egyre nagyobb teret hódítanak, az is célt volt hogy egy olyan mobil (android) alkalmazás társuljon a web alkalmazáshoz mely egy könnyen kezelhető intuitív felületen keresztül lehetővé tegye a rendszer használatát, bizonyos funkciókat még offline üzemmódban is elérhetővé téve, kielégítve napjaink átlagos felhasználóinak igényeit.

A dolgozat elején röviden bemutatom az alkalmazás által használt algoritmusokat. Majd rátérek a weboldal és a mögötte álló rendszerek felépítésére, a hozzájuk társuló fejlesztői környezetek és eszközök rövid ismertetésére. Ezt majd a mobil (android) alkalmazás bemutatás követ, melynél szintén kitérek a rendszer felépítésére illetve a megvalósítást lehetővé tevő eszközökre, erőforrásokra. Logikusan ezt a két alkalmazási felület (web és mobil) összefűzésének megvalósítása követi. Az összefoglalás előtt néhány teszten keresztül bemutatom a rendszer megbízhatóságát. Legvégül pedig kitérek néhány olyan dologra mely segítette a munkámat, mint például verziókezelő rendszerek használata.

# 1. fejezet

## Borkóstoló algoritmusok

Ezen szakdolgozat keretében a CoHITS algoritmus került implementálásra, mely mellé csatlakozott néhány Borkóstolás projekt [1] keretében használt már Szilárd Iván által implementált algoritmus, mint a Hamming, Koszinusz, Precedencia, Összefüggősségi. Ezeknél az algoritmusoknál a PageRank egy hasonlósági mátrixon propagál, ennek elemeit a borkóstolók páronkénti értékeléseinek inverz távolságából kapjuk a hozzájuk illő távolság mértékek alkalmazásával (Hamming távolság, koszinusz távolság, precedencia távolság, összefüggősségi távolság) [1].

### 1.1. CoHITS

A CoHITS algoritmus a PageRank és a HITS algoritmuson egy kiterjesztett változata [5]. A PageRank algoritmust Sergey Brin és Larry Page fejlesztette ki a Google kereső használatához [2] keresési találatok fontossági rangsorolásnak meghatározása céljából. Tőlük függetlenül Jon Kleinberg egy hasonló koncepcióval állt elő mely ugyan ilyen jellegű feladatot látott el [3].

Legyen  $X$  és  $Y$  a kóstolók és borok. Ugyan abból a  $p^0$  értékből indulunk ki minden  $x_i \in X$  kóstolónál. Ekkor legyen  $w'(\overrightarrow{x_i y_j})$  az az értékelés amit  $y_j$  bor kapott az  $x_i$  kóstolótól és legyen  $w(\overrightarrow{x_i y_j}) = w'(\overrightarrow{x_i y_j}) / \sum_{j \in Y} w'(\overrightarrow{x_i y_j})$  a normalizáltja. Továbbá legyen  $q_j^0$  érték (az  $y_j$  borra) az értékelések átlaga. Majd definiáljuk a  $w(\overleftarrow{x_i y_j})$  az alábbi képen: tegyük fel hogy  $y_j$  bort  $l$  különböző kóstoló kóstolta és legyen

$$D = \sum_{i \in X} |q_j^0 - w'(\overrightarrow{x_i y_j})|, \quad (1.1)$$

az összegek különbsége az átlagtól az  $y_j$  borra tekintve. Végül, legyen

$$w(\overleftarrow{x_i y_j}) = \frac{|\sum_{i \in X} |q_j^0 - w'(\overrightarrow{x_i y_j})||}{(l-1) D} \quad (1.2)$$

Ekkor  $\sum_{i \in X} w(\overleftarrow{x_i y_j}) = 1$  tehát minden  $w(\overleftarrow{x_i y_j})$  súlyra lehet úgy tekinteni mint egy átmeneti valószínűség  $y_j$ -ből  $x_i$ -be. Az 1.1 ábra a súlyok kiszámítására mutat egy konkrét példát.

A két kóstoló közti súly  $x_i$  és  $x_j$  úgy definiálható mint egy rejtett átmeneti valószínűség 1 által definiálva. Ekkor a  $p = W_{xp}$  HITS egyenlet  $p = (p_1, p_2, \dots, p_m)$  megoldás megadja a kóstolók rangsorát.



1.1. ábra. A gráf súlyai amikor a Kóstoló 1 a 20, 30 és 70 értékeléseket adott a Bor 1, Bor 2 és Bor 3 borokra, valamint a Bor 1 a 20, 30 és 70 értékeléseket kapott a Kóstoló 1, Kóstoló 2 és Kóstoló 3 aktól.

Tehát látható, hogy az így elő állított algoritmust alkalmazni lehet egy borkóstolási gráfra kóstolók rangsorolásának céljából az ő képességeik és szakmai hozzáértésük alapján. Általánosságban az figyelhető meg, hogy az így előálló rangsor jobban teljesít mint más egyszerű statisztikai algoritmusok. Ki tudja szűrni a hozzá nem értő kóstolókat [5].



## 1.2. Hamming

A Hamming-távolságot (1.3) többnyire vektorok, karaktersorozatok valamint bitminták közötti eltérések kimutatására használják. A Hamming-távolság csak azt mondja meg, hogy hány helyen nem illeszkedik az egyik minta a másikra. Legyen  $S_1$  és  $S_2$  két azonos hosszú minta.

$$D_{Hamming}(S_1, S_2) = \sum_{p=0}^{|S_1|} S_1(p) \neq S_2(p) \quad (1.3)$$

Ez a távolság nem lesz megfelelő mérték vizsgálathoz mivel nem túl informatív.

## 1.3. Koszinusz

Legyen  $\pi^1$  és  $\pi^2$  két permutáció-vektor. Ekkor ezek koszinusz távolságát az (1.4) képlet szerint számíthatjuk ki, ahol  $\|\cdot\|$  az euklideszi normát jelöli.

$$D_{cos}(\pi^1, \pi^2) = 1 - \frac{\pi^1 \cdot \pi^2}{\|\pi^1\| \cdot \|\pi^2\|} \quad (1.4)$$

## 1.4. Precedencia

A precedencia távolság meghatározásához definiálnunk kell először a precedencia mátrix fogalmát. Egy  $\pi$ ,  $n$  hosszúságú permutáció precedencia mátrixa egy olyan bináris elemekből álló  $n \times n$ -es  $P$  mátrix, melyben  $P_{\pi_i \pi_j} = 1, i < j$  egyébként  $P_{ij} = 0$ , azaz a permutáció elemeinek sorrendjét kódoljuk egy bináris mátrixszal. A precedencia távolságot két  $\pi^1$  és  $\pi^2$  permutáció között ( $P_1$  és  $P_2$  a nekik megfelelő precedencia mátrix) a precedencia mátrixuk egyező nem nulla elemeinek számából számíthatjuk ki (1.5) szerint.

$$D_{prec}(\pi^1, \pi^2) = \frac{n(n-1)}{2} - \sum_{i=1}^n \sum_{j=1}^n P_{ij}^1 P_{ij}^2 \quad (1.5)$$

## 1.5. Összefüggősségi

Az összefüggősségi távolság a szomszédos elemek azonos előfordulásából számítható. Legyen a két permutációnk  $\pi^1$  és  $\pi^2$ ,  $n$  hosszúságúak, és a szomszédos elemek azonos előfordulásának száma  $n_{adj}$ .

Szemléletesen: Legyen  $N_\pi$  a  $\pi$  permutáció összefüggősségi mátrixa. Kezdetben minden eleme legyen 0, majd a  $\pi$  permutáció minden  $i$  elemére az  $N_{\pi i}(\pi_i, \pi_i + 1) = 1$ . A  $\pi^1$

és  $\pi^2$  permutációra kiszámoljuk az  $N_{\pi^1}$  és  $N_{\pi^2}$  mátrixokat.  $n_{adj}$  az  $N_{\pi^1}$  és  $N_{\pi^2}$  mátrixban azonos cellában lévő 1-esek száma.

Ekkor az összefüggősségi távolság (1.6) szerint számítandó.

$$D_{adj}(\pi^1, \pi^2) = n - n_{adj} - 1 \quad (1.6)$$

[1]

## 2. fejezet

### A weboldal

A web oldal kialakításánál számomra fontos szempont volt, hogy az egy letisztult könnyen kezelhető felületet prezentáljon a felhasználók felé. Ennek érdekében már a kezdetektől folyton szem előtt tartottam hogy a stilisztikai elemek hogyan viszonyulnak egymáshoz, illetve külön figyelmet fordítottam arra, hogy a CSS (Cascading Style Sheets) konzisztens megjelenést biztosítson a web alkalmazás minden oldalán. Ez azért is jó döntés volt, mert így sokkal könnyebb volt kisebb finomításokat végezni, mint ha a stilisztikai definíciók közvetlen a php fájlokban található HTML tagokban szerepeltek volna.

A web alkalmazás alapvetően három részből épül fel:

- Statikus oldalak
- Dinamikus oldalak
- Az android alkalmazást ellátó szolgáltatások

Státikus oldalnak akkor nevezünk egy weblapot, ha annak a tartalma nem változik attól függően, hogy a látogató kicsoda, hanyadszor jelentkezik be vagy hogy honnan jelentkezik be stb..., tehát a tartalom konzisztens minden egyes látogatás alkalmával. Ez a megközelítés főképp akkor előnyös ha valami olyan tartalmat akarunk nyújtani a felhasználóknak amin egyáltalán nem áll szándékunkban változtatni, vagy csak nagy ritkán.

Dinamikus oldalakról akkor beszélünk amikor valamilyen feltétel vagy beavatkozás közvetlenül befolyásolja azt hogy a felhasználó által megtekintett tartalom pontosan micsoda is. Legtöbbször akkor használatos, ha egy adatbázis tartalmát akarjuk megjeleníteni, vagy olyan felületet hozunk létre amely interakciót tesz lehetővé a felhasználó és a web alkalmazás között. Rengeteg programozás nyelv létezik ennek a célnak az ellátására. Én a php-t és a JavaScript-et választottam.

Az android alkalmazást szolgáltatásokkal ellátó rész fő feladata hogy a szerveren található adatbázishoz elérést biztosítson, mivel android alkalmazások nem tudnak közvetlenül

csatlakozni egy távoli adatbázishoz, ezért kell ez a közbülső réteg amely kiszolgálja a mobil eszköz kéréseit, illetve olyan tartalmakat elérését teszi lehetővé amelyek közvetlenül valamilyen limitációs okokból nem lennének elérhetőek. Sokszor ezeket egy böngészőből megnyitva nem látunk semmit, ez teljesen természetes, mivel itt a hangsúly nem azon van hogy emberek számára értelmezhető/olvasható tartalmat biztosítsunk, hanem fő célunk a web és a mobil alkalmazás közötti kommunikációs csatorna megteremtése, tehát egy olyan környezet ahol csak gépek beszélnek egymással.

## **2.1. Iterációk**

Az oldal megalkotásához egy agilis iterációs dizájn megközelítést alkalmaztam, melyen során egy rövid kezdeti tervezés után elkezdtem a tényleges munkát ami abból állt, hogy implementáltam az adott feladatot, majd az így elkészült munkát tesztelve prezentáltam a megrendelőnek (konzulensnek) A következő lépés a tesztek során előkerülő hibák megbeszélése és azok javításának lehetséges orvosolása valamint az alkalmazás új funkciókkal való bővítése volt. Ez az utolsó három lépés, tehát az implementálás, tesztelés és hibajavítás valamint további funkciókkal való bővítés képezte az iterációs dizájn megközelitésem ciklikus magját. Az iterációk során felhasznált eszközökre és technológiák ismertetésére majd csak az iterációs mérföldkövek ismertetése után fogok kitérni.

### **2.1.1. Első mérföldkő**

Az első mérföldkő folyamán először megbeszéltem a megrendelővel hogy milyen elvárásokkal rendelkezek illetve vázoltam az általam tervezett munkamenetet. Továbbá kiválasztottam a fejlesztéshez szükséges fejlesztői környezeteket, valamint ezek és a használatukhoz elengedhetetlen eszközöket előkészítettem a projekt munkában való felhasználáshoz. Ekkor történt meg még a felhasználói felület megjelenési terveinek elkészítése. Webszerver, domain és adatbázis létrehozás illetve telepítés. A kész oldalszerkezeti tervekkel 2.1 neki kezdtem a grafikus elemek vizualizációjának valamint implementáltam az oldalszerkezet statikus elemeit mint: Menü rendszer, fejléc, alsó információs sáv, kapcsolatok és egyéb hír csatornák megjelenítésért felelős elemeket.

### **2.1.2. Második mérföldkő**

A második mérföldkő során került sor a megrendelőtől kapott dokumentumok és források feldolgozására majd azok feltöltése a projektbe főként statikus HTML elemekként. Továbbá végrehajtása a megrendelő által kért grafikus és szerkezeti változtatásokat mint:



2.1. ábra. Első oldalszerkezeti terv

Alsó információs sáv átmozgatása és re-faktorizációja egy oldalsó mozgó elemmé, menüpon-  
tok átnevezése, fejléc logó frissítése, A *borkostolasEredmenyek* oldal átdolgozása az  
alábbi módon: Táblázatok kettéválasztása. Vízszintes görgető sáv hozzáadása a borok  
táblázat felső és alsó részéhez, melyek együtt mozognak egymással így megoldva a táblázat  
nagyságából fakadó kezelhetőségi problémát. Az előre láthatólag bekövetkező grafikai  
felépítés esetleges módosítása könnyítésének érdekében több stílus elemet is újabb osztá-  
lyokba rendeztem, tehát egy átfogóbb re faktorizáció történt mely mint később kiderült a  
további stilisztikai elemek hozzáadását és stilizálását is megkönnyítette.

### 2.1.3. Harmadik mérföldkő

A harmadik mérföldkő fő feladata az azt megelőző konzultáció során átbeszélt *demo* oldal  
elkészítése volt. Tehát itt került sor a CoHITS algoritmus megismerése a hozzá tartozó  
szakirodalom áttanulmányozására majd az algoritmus implementálásának megkezdésére  
a projektbe. Ezt követte a korreláció és átlagtól való átlagos eltérés számolásának imple-  
mentálása, de mint később kiderült ezek az algoritmusok nem tűrték jól a hiányos be-  
meneti adatokat ezért végül az általuk generált kimenet nem került felhasználásra. A mér-  
földkő végére a CoHITS algoritmus implementálásának befejezése és tesztelése maradt.  
A Google grafikonszerkesztő API megismerése, majd felhasználása a CoHITS algoritmus,  
korreláció és átlagtól való átlagos eltérés adatsorok reprezentációjához.

#### 2.1.4. Negyedik mérföldkő

A negyedik mérföldkő során egy másik projektből[1] származó algoritmusok és azok már kész implementálásának feldolgozása és megismerése volt a fő feladat. Át kellett tekinteni a kapott kódbázist abból a célból, hogy ezeket is mint majd további opciókat integráljam a rendszerbe. Az említett algoritmusok: Hamming, Koszinusz, Precedencia, Összefüggősségi. De még mielőtt ebbe bele kezdtem, kísérleteztem egy kicsit egy új grafikonrajzoló API (Charts.js) használata mely könnyebben, stílusosabban és személyre szabhatóbban reprezentálja az adatokat. Miután megkaptam a fent említett már implementált algoritmusokat, elkezdtem azokat integrálni az én projektem kódbázisába. Sajnos ez nem ment teljesen gördülékenyen, szükség volt a kapott kódbázisból előkerülő implementációs és működési problémákkal kapcsolatos kérdések átgondolására valamint konzultálásra ezekkel kapcsolatban, majd a hibák javítása. A sikeres hibaelhárítás és javítások után jött az algoritmusok és demo oldal működésének tesztelése.

#### 2.1.5. Ötödik mérföldkő

Az ötödik mérföldkő bebizonyította, hogy érdemes volt jól megtervezni és szétválasztani az egyes stilisztikai elemeket, mivel a megrendelő ismét a menü rendszer valamint a fejléc kép ismételt átrendezését és szerkesztését kérte. Továbbá új teszt adatok használatát javasolta a *demo* menüpontban, amelyek már a friss adatokat tükrözik. Implementálásra került egy visszaigazolást kérő felhasználói felület mely fő célja a beviteli mátrix véletlen felülírásának meggátolása volt. Később a bevitel metódus megváltoztatása miatt ennek a funkcionalitása elévült. A korrelációt számoló kód működéséből fakadó hiba ki lett javítva, így már nem áll meg a számolás, ha a kapott hiányos adatsoron nem lehet egyszerűen korrelációt számolni. Valamint megtörtént a tesztelések során előkerült további hibák kijavítása is.

#### 2.1.6. Hatodik mérföldkő

A hatodik mérföldkő vissza hozta a régi grafikonrajzoló APIt (Google charts API), valamint megtörtént a grafikonokat megjelenítő elemek stilizálásnak egységesítése. Fő ok a visszaváltásra az volt, hogy az értékeket mint címkéket megjeleníthetővé tegyük az oszlopok mellett. A borszakértők személyiségi jogainak védelme érdekében a nevük anonimizálásra került. A web alkalmazás új funkciókkal bővült: Regisztráció, belépés. Ezt követte a demo oldal teljes átdolgozása így már az hozzá kapcsolódik a felhasználói rendszerhez. Ebből fakadóan a demo oldal már csak regisztráció illetve bejelentkezés után használható. így a felhasználók a neves borszakértők által kóstolt borokat saját maguk is tudják értékelni.

### **2.1.7. Hetedik mérföldkő**

A hetedik és egyben utolsó mérföldkő nagy részét a mobil android alkalmazás elkészítése és tesztelése tette ki, de itt valósult meg az őt kiszolgáló web szolgáltatások implementálása valamint egy alap adminisztrációs felület létrehozása. Legvégül pedig elvégeztem némely kód csinosítást és re-faktorizációt hogy a kódom könnyebben olvasható és általában véve jobb tetszést keltsen az emberben.

## **2.2. Adatbázis**

## **2.3. PHP**

## **2.4. JavaScript**

## **2.5. JQuery**

## **2.6. Grafikonok**

### **2.6.1. Google Charts**

## **2.7. Statikus tartalom**

### **2.7.1. index.php - Főoldal**

### **2.7.2. borkostolasEredmenyek.php - Eredmények**

### **2.7.3. modszerek.php - Módszerek**

### **2.7.4. kapcsolat.php - Kapcsolat**

## **2.8. Dinamikus tartalom**

### **2.8.1. Regisztráció**

### **2.8.2. Bejelentkezés**

### **2.8.3. demo.php - Demó**

## 3. fejezet

# A mobil alkalmazás

Android alkalmazás fejlesztés

### 3.1. Android

Android VC ndroid register, passwordReset alk about threading UI/new thread mi futhat  
hol és mi nem ávoli adatbázis elérés ocal SQLLite adatbázis okális változok kezelése  
SQL/properties vagy mifene

M

a

t

t

l

l



## **4. fejezet**

# **A weboldal és a mobil alkalmazás összefűzése**

A weboldal és a mobil alkalmazás összefűzése

## 5. fejezet

### Tesztelés

#### 5.1. Regisztrációs és bejelentkeztető rendszer

#### 5.2. Demo adatkezelésének ellenőrzése

#### 5.3. Algoritmusok ellenőrzése kis adatokon

#### 5.4. Algoritmusok ellenőrzése ismert eredményekkel

ellékelni az excel file-t és arra hivatkozni, valamint az oldal kezdetleges verziójáról egy  
kép

m

## **6. fejezet**

### **Összefoglalás**

## 7. fejezet

# Egyebek

### 7.1. Verziókezelés

#### Verziókezelés röviden

Ez alatt több verzióval rendelkező adatok kezelését értjük. Leggyakrabban a szoftverfejlesztésben használnak verziókezelő rendszereket fejlesztés alatt álló dokumentumok, tervek, forráskódok és egyéb olyan adatok verzióinak kezelésére, amelyeken több ember dolgozik egyidejűleg vagy amelyen több fizikai helyről dolgoznak.

#### 7.1.1. Git

A **Git** egy nyílt forráskódú, elosztott verziókezelő szoftver, amely a sebességre helyezi a hangsúlyt melyet eredetileg Linus Torvalds fejlesztette ki a Linux kernel fejlesztéséhez. Az elosztottság abban valósul meg, hogy a Git minden fejlesztő helyi munkaváltozatában rendelkezésre bocsátja a teljes addigi fejlesztési történetet, és a változtatások másolása mindig két repository között történik. Ezeket a változtatásokat mint külön ágakat importálják és összefésülhetők, hasonlóan a helyben létrehozott fejlesztési ágakhoz. Ez azért jó, mert így minden munkamásolat egy teljes értékű repository teljes verziótörténettel és teljes revíziókövetési lehetőséggel, amely nem függ a hálózat elérésétől vagy központi szervertől.

#### 7.1.2. GitHub

A GitHubot eredetileg Tom Preston-Werner, Chris Wanstrath és PJ Hyett hozta létre a kódmegosztási procedura szimplifikálásának érdekében, mára a világ legnagyobb kód távoli kód repository szolgáltatójává nőtt [4].

### 7.1.3. A választás

Már a kezdetektől nyilvánvaló volt számomra, hogy valamilyen verziókezelő rendszer használata elengedhetetlen lesz, mivel gyakran nem csak egy specifikus munkaállomásról szoktam dolgozni. Ez nem csak azért jó, mert nagyobb flexibilitást nyújt térben és időben, hanem azért is mert valamilyen szinten szimulálja azt amikor valaki egy projekt munkában egy közös kódbázison dolgozik. Korábbi munkáim és tanulmányaim során már néhány ilyen rendszerrel is megismerkedtem, többek között SVN, Git, Mercurial. A szakdolgozathoz tartozó alkalmazások készítéséhet végül a Git -et választottam mint verziókezelő rendszer, melyhez távoli repository szolgáltatónak a GitHub társult. A választás fő támpontjai között talán a felhasználó barátságot és könnyű automatizálhatóságot említeném meg.

## 7.2. Környezetek

SublimeText - linterek

PhpStorm

Android Studio

Külön fájlban elkészített grafika beillesztését a

## **8. fejezet**

### **Függelék**

#### **8.1. A program forráskódja**

# Nyilatkozat

Alulírott Varga Tamás programtervező informatikus BSc szakos hallgató, kijelentem, hogy a dolgozatomat a Szegedi Tudományegyetem, Informatikai Tanszékcsoport Számítógépes Optimalizálás Tanszékén készítettem, programtervező informatikus BSc diploma megszerzése érdekében.

Kijelentem, hogy a dolgozatot más szakon korábban nem védtem meg, saját munkám eredménye, és csak a hivatkozott forrásokat (szakirodalom, eszközök, stb.) használtam fel.

Tudomásul veszem, hogy szakdolgozatomat a Szegedi Tudományegyetem Informatikai Tanszékcsoport könyvtárában, a helyben olvasható könyvek között helyezik el.

Szeged, 2015. április 28.

.....

aláírás



# Irodalomjegyzék

- [1] Borkóstolás projekt - <http://www.inf.u-szeged.hu/tmeth/osa/>
- [2] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks and ISDN Systems*, vol. 30, no 1, 107-117, 1998
- [3] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, vol. 46, no. 5, 604-632, 1999
- [4] GitHub - <https://github.com/about>
- [5] London András és Csentes Tibor. HITS based network algorithm for evaluating the professional skills of wine tasters