HF04 - Clustering

Vajna Levente

Október 8, 2024

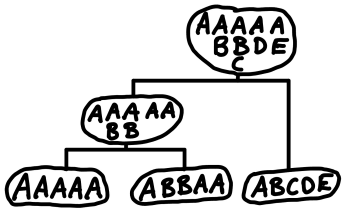
Leadási Határidő: 2024.10.16. 23:59

Ismertető

A feladat egy cluster-elési algoritmus, valamint egy példa egyed megírása lesz. Adott egy egyedhalmaz, azonos típusokkal. Egy hasonlósági metrikával meg kell határozni a két leghasonlóbb egyedet, majd azokat összekötni úgy, hogy ezek lesznek a gyerekei azon szülőnek, mely a kettő egyed egyesítéséből jön létre. Innentől már csak az egyesített szülővel szükséges számolni. Ez az algoritmus egy bottom-up, azaz alulról felfelé felépített fát fog létrehozni.

Példa: Fix hosszú String

Ebben a példában az egyedek 5 karakter hosszú, angol nagybetűkből álló karakterláncok lesznek: *AAAAA*, *ABBAA*, *ABCDE*. A leghasonlóbb itt *AAAAA* és *ABBAA* lesz, hiszen 3 közös betűjük van. Ezek összekapcsolásából lesz $A\{A,B\}\{A,B\}AA$. A következő kapcsolat már csak *ABCDE* egyeddel lehet, tőle a távolság 2, szülőjük pedig $A\{A,B\}\{A,B,C\}\{A,D\}\{A,E\}$ lesz, ami pedig jelen esetben a gyökérelem is. Ennek vázlatos rajza látható az 1. ábrán.



1. ábra. Példa string fa

1. rész

Első rész a `String10` class implementálása, és `String10.hpp` néven való feltöltése.

Elvárások

- `String10` osztály implementálása, melyben az egyed egy 10 hosszú, angol nagybetűkből álló karakterláncot tárol. (Ehhez skeleton kód a `String10.hpp`, ezen függvények implementálása szükséges.)
- `String10(string line)` konstruktor implementálása, mely egy stringet fogad paraméterül, ami az input file egy sora, és ebből építi fel az egyedet.
- Legyen egy `String10(const String10 & s1, const String10 & s2)` konstruktora is, mely a korábban mutatott módon előállítja a két egyedből a szülőjüket.
- Tartalmaznia kell egy `friend double operator*(const String10 & s1, const String10 & s2)` függvényt [1] [2], mely visszatér azzal, hogy hány pozícióban egyezik meg a karakter (”joker karakterek” esetén is működjön).
- `bool operator==(const String10 & s) const` egyenlőségvizsgálat operátor implementálása.

Tudnivalók

- Az input file minden sorában csak 10 angol nagybetűs karakter van, ez alkot egy egyedet.
- Egyedül a ”köztes” egyedeknek vannak fent mutatott ”joker karakterei”.

2. rész

Második részként meg kell írni a `ClassTree` implementációját.

Elvárások

- `ClassTree` implementálása.
- Az adathalmazt a `ClassTree(string file)` konstruktor paraméterként átvett nevű file-ból olvassuk ki, melynek minden sorában egy egyedhez tartozó adatok találhatóak.
- Kezelje a program, ha két ugyanolyan egyed lenne az inputon. Ne szerepeljen többször, csupán az első előfordulás maradjon meg!
- Adatok fába helyezése során beolvasás szerinti korábbi legyen a balgyerek, későbbi a jobbgyerek. Ezt követően a végére kerül a szülő, többire többé nincs szükség.
- A feladat során elvárás a `template` használata, hogy bármely összehasonlítható adattal működjön a fa.
- Az így felépített fában egy `int distance(const T & t1, const T & t2)` távolságszámító algoritmus is implementálva kell legyen, mely annak számát adja meg, hogy hány szülőn megy keresztül, míg egyik egyedből a másikba nem jut. Például az 1. ábrán látható példában `AAAAA` és `ABCDE` közti távolság 2.
- Szükséges egy `int treeHeight()` magasság meghatározására alkalmas tagfüggvény, mely visszaadja, hogy mekkora távolságra van a gyökértől a legtávolabbi egyed. Az 1. ábrán látható példában ez a magasság 2.
- Szükséges egy `int treeSize()` függvény, mely visszaadja, hogy hány egyed található a fában.

Tudnivalók

- Minden típushoz van `double operator*(const T & t1, const T & t2)` függvény írva, mely a hasonlóságot adja vissza bizonyos metrika szerint. Ez az érték egy pozitív szám, és minél hasonlóbb, annál nagyobb.
- Minden típushoz van `T(std::string line)` konstruktor írva, mely a file egy sorából alkotja meg az adott típust.
- Minden típushoz van `T(const T & t1, const T & t2)` konstruktor implementálva, ami a két gyerekéből hozza létre a szülőt.
- Amennyiben több azonos távolságú egyed van, a listában előrébb lévő egyedpár kerüljön először a fába.
- Minden típushoz van `bool operator==(const T & t2)` egyenlőségvizsgálat írva.

Alkalmazása

Az ilyen clustering algoritmusoknak számos további felhasználása lehet, alkalmazható például koordinátákkal [3], neurális hálózatokban, sejtek osztályozásában, de akár még beadott házi feladat kódok hasonlóságának csoportosításában is.

Beadás: A teljes megoldás **mindkét rész** megoldását tartalmazza! A megadott flagekkel (**-Werror -Wall -Wextra -pedantic**) nem forduló kód 0 pontos házi feladatot eredményez. Ennél a házinál a repo szerveren automata tesztek is futnak, melyek ellenőrzik a feltöltött `String10` osztállyal is a helyes működést. A pontozásnál a kódminőség, és az algoritmus hatékonysága is számít a kód helyes működése mellett. Alkalmazzuk a kurzus során tanult módszereket és adatszerkezeteket! A kódba magyarázó kommentek is kerüljenek, amelyek leírják a kód működését!

Hivatkozások

[1] Cppreference, *Friend declaration*, 2024. cím: https://en.cppreference.com/w/cpp/language/friend.

[2] G. for Geeks, *Friend Class and Function in C++*, 2024. cím: https://www.geeksforgeeks.org/friend-class-function-cpp/.

[3] A. Ihler, *Clustering (2): Hierarchical Agglomerative Clustering*, 2015. cím: https://www.youtube.com/watch?v=OcoE7JlbXvY.

Jó kódolást!