ИМЕ: **Антония Тодорова Тодорова**

EMAIL: **a.t.todorova97@gmail.com**

ДИСЦИПЛИНА: **Мрежово програмиране**

ТЕМА: **Реализиране на клиент и сървър приложения чрез UDP протокол за търсене на зависимост, използвайки алгоритъм „FPMax Algorithm for Mining Frequent Maximal Itemsets”**

**СЪДЪРЖАНИЕ**

ТЕМА - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 3 )

Описание на темата с използвания алгоритъма - - - - - - - - - - - - - - - - - - - ( 3 )

РЕШЕНИЕ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 4 )

Описание на решението на проблема - - - - - - - - - - - - - - - - - - - - - - - - - - ( 4 )

ПРОЕКТИРАНЕ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 4 )

РЕАЛИЗАЦИЯ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 5 )

Описание на програмната реализация - - - - - - - - - - - - - - - - - - - - - - - - - - ( 5 )

РЕЗУЛТАТ - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 6 )

Резултат от работата на приложението - - - - - - - - - - - - - - - - - - - - - - - - - - ( 6 )

ЛИТЕРАТУРА - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 6 )

Източници - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - ( 6 )

**ТЕМА**

ОПИСНИЕ НА ТЕМАТА С ИЗПОЛЗВАНИЯ АЛГОРИТЪМ

Тема, която се разработва в проекта е реализиране на клиент и сървър приложения чрез UDP протокол за търсене на зависимост, използвайки алгоритъм „FPMax Algorithm for Mining Frequent Maximal Itemsets”.

UDP е абривиатура на User Datagram Protocol като това е минимален транспортен пакетен протокол. Той представлява прост интерфейс между мрежовия протокол под него и приложните протоколи над него. В същността си, при работа не гарантира доставката на данните, тоест не се пази информация за изпратено съобщение. UDP добавя към IP пакета единствено мултиплексиране на приложения, чрез номерата на портовете и проверка на целостта на данните, чрез контролна сума. В случай, че е необходимо надеждност на предаването, тя трябва да се реализира в потребителското приложение.

FPMax Algorithm е алгоритъм за намиране на максималното множеството от елементи, което се среща най-често в колекция от множества наречена транзакция.

Входа за алгоритъма представлява така наречената транзакция от база данни – transaction database. Това е множество от множества, като всяко множество само по себе си е транзакция от елементи – transaction set. Важно е да се отбележи, че в едно множество (set), даден елемент не може да се дублира, както и е нуждо елементите да са подредени във възходящ ред. Транзакцията от база данни се предоставя на алгоритъма под формата на фийл, записан в системата. Неговия формат е текстов и представлява редове от транзакции, като всяко множество се идентифицира от елементи с положителни уникални числа, разделени с интервал. Към него се добавя и стойност minsup, което представлява стойност в проценти между 0 и 100, указваща минималната граница за търсене на зависимост.

Изходът от алгоритъма представлява най-често срещаното максимално множество. Също така се дава и информация за честотата на срещанията под формата на число, което е убозначено със support, указващо точния брой на транзакциите съдържащи това множество. Всяко едно множество има съответстващ му съпорт. Най-често срещано множество е такова, което се поява поне minsup на брой транзакции в базата данни. Най-често срещано затворено множество, е като горното, но с допълнение, че не е част от никое по-голямо множество, което има съшия съпорт. В ресултата всяко максимално най-често множество се идентифицира с честота си. Това се случва, като на всеки ред се извежда намереното множество, след това специфичната комбинация от символи “#SUP:” и накрая числото, показващо честотата.

Този алгоритъм е изключително надежден и ефикасен за намирането на най-често срещано максимално множество.

**РЕШЕНИЕ**

ОПИСНИЕ НА РЕШЕНИЕТО НА ПРОБЛЕМА

За да се реши дадения проблем, е необходимо създаването на кодировка, която да преработи входната информация в удобен за алгоритъма формат, така че да може да бъде пуснат. Кодировката, която се използва в решението е чрез даване на уникален идентификационен номер на всеки един различен стринг. Така се получава напълно еднозначно определяне на двойката (String, Integer), което позволява възстановяването на данните, след прехвърлянето от един формат в друг.

**ПРОЕКТИРАНЕ**

{the network}

request

receive()

Server

send()

Client

STRING

List<Information>

max

integer

string

Map < String , Pair < Integer, Integer > >

Map < String , Integer >

result.txt

source.txt

integer

integer

string

List<Information>

file.txt

class INFORMATION

date : string

time : string

event context : string

component : string

event name : string

description : string

origin : string

ip address : string

response

receive()

send()

**РЕАЛИЗАЦИЯ**

ОПИСНИЕ НА ПРОГРАМНАТА РЕАЛИЗАЦИЯ

Първата стъпка от решаването на проблема е да се създаде проект, в който да има сървър и клиент, които да изпращат и получават информация помежду си. Това са класове, които са активни и могат да бъдат стартирани (run-ати). В тях се създават сокети, които ще усъществят връзката, като приемат и предават пакети. Един пакет се състои от информацията, която ще бъде предадена, дължината на тази информация, ip адрес на получателя и порт, на който ще се прати пакетът. Когато сървърът и клиентът приключат с предаването на всички пакети, сокетите се затварят.

След като се осъществи всичко написано в горния абзац, се преминава към реалното решение на проблема. Създава се клас Information, който ще държи цялата информация за един обект. Първо клиентът праща име на файл към сървъра, от който иска да бъдат прочетени данните и съответно ще очаква анализ на точно тези данни. След това сървърът прави преработка и анализ чрез алгоритъма по следния начин:

* отваря се файла, съдържащ цялата информация и се чете ред по ред от него, понеже всеки ред ще съответсва на един обект
* създават се обекти от клас Information, които съдържат в себе си date, time, event context, component, event name, description, origin, ip address, които определят едно действие
* всички създадени обекти, от клас Information, се записват в List<Information>
* създава се Map<String, Pair<Integer, Integer>>, който ще съдържа string от информация за съответно поле в обекта от тип Information, както и две числа съответно уникален идентификационен номер на стринга и номер показващ полето в обекта Information
* паралелно със създаването на Map-а, съхраняващ цялата кодирана информация, се създава и файл с име source.txt, който ще съдържа информацията, преработа в удобед вид за алгоритъма, а именно редове от сортирани уникални числа, идентифициращи конкретен стринг
* върху създадения филе с име source.txt се пуска алгоритъмът, като резултатът от него се записва в нов файл с име result.txt
* резултатът се анализира, като се преминава през всеки ред и се създа обект Map<String, Integer>, съдържащ кодираната информация под формата на стринг от числа и едно единствено съответстващо число, показващо честота на срещане на множеството
* намира се най-често срещаното множество, което съответства на максималното число, върнато от алгоритъма, като е възможно да има няколко множества с единакъв максимален съпорт
* правят се обекти от клас Information съдържащи декодирана информация, запазена в интовите числа, като се намират съответните им стрингове в Map-а с цялата информация
* всички създадени обекти, от клас Information, се записват в List<Information>
* обектите се форматират в стрринг, удобен за четене

Накрая информацията се праща от сървъра към клиента по подходящ начин, а именно записана в получения стринг от целия анализ на програмата, който клиентът може да прочете без никакви проблеми.

**РЕЗУЛТАТ**

РЕЗУЛТАТ ОТ РАБОТАТА НА ПРИЛОЖЕНИЕТО

Разултатът от работата на приложението е намиране на закономерност за най-извършваните действия от потребителите. Тази информация съдържа датата, на която са се логвали различни потребители, времето, което е най-натоварено, контекст на най-ползвано събитие, най-използваните компоненти, името на най-усъществяваното събитие, както и неговото описание, от къде се е извършвало най-често логване и кой ip адрес е най-активе. Цялата тази информация се записва в string, който се праща от сървъра на клиента.

**ЛИТЕРАТУРА**

ИЗТОЧНИЦИ

SPMF – an open source data mining library

http://www.philippe-fournier-viger.com/spmf/FPMax.php