МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
Факультет комп'ютерних наук, фізики та математики
Кафедра інформатики, програмної інженерії та економічної кібернетики

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПОДІЛУ СТАТЕЙ РЕЦЕНЗЕНТАМ У ЗБІРНИКУ НАУКОВИХ ПРАЦЬ

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти «бакалавр»

Виконав: студент 4 курсу 431 групи
Спеціальності: 122 комп'ютерні науки
Освітньо-професійної програми:
Комп'ютерні науки
Ракітін Дмитро Андрійович
Керівник: д.п.н., доц. Валько Н.В.
доктор педагогічних наук, професор
Співаковський О.В.
Рецензент: к.п.н., доц. Гончаренко Т.Л

Херсон – 2021

ЗМІСТ

ВСТУП 3
РОЗДІЛ 1. Аналіз алгоритмів. Переваги та недоліки 6
1.1 Постановка задачі 6
1.2 Алгоритм стабільних шлюбів 7
1.3 Задача вибору оптимального місця роботи 8
1.4 Дослідження роботи алгоритму для розподілу студентів 12
Висновки до першого розділу 15
РОЗДІЛ 2. Побудова моделі розподілу основних алгоритмів стабільного узгодження 17
2.1 Концептуальна схема проектованого 17
2.2 Дослідження алгоритму для вибірки один до одного 18
2.3 Дослідження алгоритму для вибірки один до багатьох 22
РОЗДІЛ 3. Створення системи розподілу статей рецензентам 26
3.1 Обрані технології. PHP. Laravel 26
3.2 Проєктування бази даних 28
3.3 Застосування алгоритму та створення серверної частини 31
Висновки до третього розділу 35
ВИСНОВКИ 36
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 38
ДОДАТОК А 40

ВСТУП

Актуальність роботи

Задачу оптимізації, в умовах зростання кількісної характеристики інформації, набувають великого значення. Якісна оцінка стратегій чи діяльності можлива лише за умови початкового оптимізаційного розподілу ресурсів, інформації. З розвитком технологій у світі значно зросла кількість даних. Аналіз великих даних (big data) грає велику роль в сучасному суспільстві. В результаті дослідження методів обробки Big data зараз покращується загальний розвиток сучасних технологій в різних сферах: промисловість, енергетика, медицина, фінанси, туризм, екологія, розваги.

Багато алгоритмів мають справу з узгодженням шаблонів. Алгоритмом стабільного узгодження є такий, що вирішує проблему оптимальної позиції елементів. 1962 року математик Лойд Шеплі та економіст Девід Гейл представили алгоритм «стабільних пар» або алгоритм «стабільних шлюбів». За розробку даного алгоритму автори отримали нобелівську премію. Завданням алгоритму було вирішення таких проблем як, наприклад, однакова кількість чоловіків та жінок шукають пару, до моменту, поки не буде досягнуто оптимальної послідовності, де кожному елементу не існувало іншої стабільної пари. Пізніше даний алгоритм було використано для вирішення таких проблем як направлення нових лікарів на оптимальне місце роботи, або перспективних студентів до навчальних закладів. У 1974 році Лойд Ш. та Герберт С. застосували основний цикл даного алгоритму в економіці. Ці дослідження також допомогли швидко та ефективно підбирати біологічно-сумісних донорів для людей котрі цього потребували.

Алгоритм відкладеного прийняття, запропонований Гейлом і Шеплі мав глибокий вплив на дизайн ринку, як безпосередньо, адаптувавшись до практичних механізмів узгодження, так і, побічно, піднявши нові теоретичні

питання. Алгоритми стабільного узгодження лежать в основі ряду клірингових центрів (звірка та оновлення інформації) на ринку праці по всьому світу і нещодавно впроваджені в системах вибору шкіл у Бостоні та Нью-Йорку. Крім того, дослідження ринків, котрі зазнали невдачі та не могли бути виправлені за допомогою централізованих механізмів, призвело до глибшого розуміння деяких питань, на які потрібно звернути увагу для покращення ситуації.

Апробація. Роботу було направлено на участь у ІІ турі Всеукраїнського конкурсу студентських наукових робіт у 2020-2021 н.р.

Мета дослідження

Основною метою кваліфікаційної роботи є створення системи розподілу статей рецензентам у збірнику наукових праць. Дана програма повинна значно полегшувати та автоматизувати пошук оптимальних рецензентів до статей та спростити адміністрування системи.

Досягнення мети дослідження передбачало розв’язання таких **завдань:**

1. Дослідити роботу алгоритму Гейла-Шеплі.
2. Проєктування технічної частини, а також бази даних проєкту для підбору рецензентів до статей.
3. Застосування алгоритму для підбору рецензентів до статей та реалізація мовою PHP (laravel)

Об’єкт дослідження - багатокритеріальні задачі оптимального розподілу.

Предмет дослідження – система розподілу на основі алгоритму Гейла-Шеплі.

РОЗДІЛ 1
АНАЛІЗ АЛГОРИТМІВ. ПЕРЕВАГИ ТА НЕДОЛІКИ

1.1 Постановка задачі

Існують проблеми, котрі вимагають створення стабільних зв’язків, такі як підбір університетів для студентів, вибір місця роботи , підбір підходящих донорів для пацієнтів у лікарнях тощо. Наприклад, розміщення офісу або не оптимально підібраний штат працівників не завжди сприяє задовільній роботі. Девід Гейл та Лойд Шеплі розробили алгоритм, котрий дозволяє підбирати стабільні пари «один до багатьох» та «багато-до-багатьох», відомий як «алгоритм стабільних шлюбів».

Використовуючи алгоритм розподілу можна отримати дводольний граф, в котрому визначено дві взаємопов’язані множини з стабільних та оптимальних елементів. Метою є знайти стабільні пари з двох вибірок X та Y. Об’єкт кожної вибірки має власний список переваг, котрі впливають на вибір його пари. Було представлено алгоритм для створення пар n-об’єктів X з n-об’єктів Y використовуючи деякі умови[5, с. 60]. Утворена вибірка не є стабільною, якщо існує пара окремих елементів за пріоритетами яких вони б підходили краще ніж поточні. Такі пари X та Y можуть «блокувати», або бути «блокуючими» парами для іншої. Закономірно, що відповідність, для якої немає блокуючої пари є стабільною. Даний термін можна інтерпретувати іншими словами: пара (x, y) є стабільною парою, якщо є деяка стабільна відповідність, в якій X і Y є партнерами, в цьому випадку кожен є стабільним партнером іншого.

В залежності від умов та оптимізаційної функції існує кілька різних задач і варіантів алгоритму розподілу. Далі розглянемо кілька з таких задач, які використовують алгоритм розподілу Гейла-Шеплі.

1.2 Алгоритм стабільних шлюбів

Перший набір включає в себе чоловіків, другий – жінок, кожний з яких має список партнерів протилежної статі, котрим вони віддають перевагу. Кожен елемент такого списку має свій пріоритет. В процесі роботи алгоритму чоловіки роблять пропозицію кожній жінці. Кожен чоловік залишається вільним, однак має альтернативну пару. Жінки постійно мають пару, однак їх партнер може змінюватись до завершення алгоритму. Чоловіки роблять пропозицію жінкам за пріоритетом, від найбільшого до найменшого. Кожного разу, коли жінка отримує пропозицію, вона порівнює її з своїми пріоритетами. Якщо попередня пропозиція має нижчий пріоритет, ніж нова, жінка відхиляє її та обирає нового партнера. Коли пропозицію чоловіка відхилено, він стає вільним та може обрати іншу жінку з найвищим пріоритетом. Алгоритм закінчує свою роботу, коли кожен об’єкт має свою пару[11, с.1253].

Псевдокод даного алгоритму має наступну структуру [5, с. 61].:

```
function stableMatching {
    Initialize all m ∈ M and w ∈ W to free
    while ∃ free man m who still has a woman w to propose to {
       w = first woman on m’s list to whom m has not yet proposed
       if w is free
         (m, w) become engaged
       else some pair (m', w) already exists
         if w prefers m to m'
            m' becomes free
           (m, w) become engaged
         else
           (m', w) remain engaged
    }
}
```

Рисунок 1.1 – Приклад псевдокоду алгоритму стабільних шлюбів

1.3 Задача вибору оптимального місця роботи

Коли певна установа наймає нового працівника та призначає його на певну посаду, має бути узгоджено багато критеріїв. Вручну розрахунки заважатимуть точності розміщення працівників на підприємстві. Взаємозв'язок між співробітниками також впливає на якість роботи. Оптимально вибраний на посаду працівник буде виконувати роботу більш ефективно.

На успіх компанії чи установи значною мірою впливає правильно підібраний штат працівників. Кожен у компанії займає своє місце згідно з його кваліфікації та навичок. Проте, є випадки, коли компанія не достатньо ретельно підбирає нових працівників. В результаті, це спричиняє дисбаланс в роботі всього закладу. Оцінка якості роботи працівника відіграє важливу роль у прийнятті рішень з різних питань. Це залежить від типу роботи та основних цілей кампанії. Компанія має узагальнити інформацію про те, якого прогресу було досягнуто в процесі роботи штату за певний період. Висококваліфіковані працівники, очікувано, будуть виконувати свої обов’язки з повною відповідальністю, та матимуть змогу розкрити свій потенціал. Компетентна людина може зробити вагомий внесок в загальну продуктивність компанії. Використання вмінь та навичок для досягнення цілей та подолання труднощів є важливим елементом для покращення працездатності. Професійні навички потрібні для досягнення успіху. Кожен людина, котра подає своє резюме, має набір навичок та буде порівнюватись з іншими саме за цим списком. Адекватна працездатність передбачає покращення в роботі працівника для підтримки професійного та кваліфікованого виконання завдань. Продуктивність виконання також

залежить від емоційного стану працівника, чи задоволений він своєю роботою.

Однією з проблем є те, що багато компаній не використовують стабільних технік під час направлення людей на нові посади. Досвід кожної людини має бути дослідженим та проаналізованим перед остаточним призначенням на певну позицію установи. Алгоритм Гейла-Шеплі може вирішити цю проблему за очікуваними критеріями в конкретній позиції кожної вибірки. Цей алгоритм визначає положення змінної на основі значення зважування або бажання установи. Кожного разу на певну посаду буде найнято більш кваліфікованого працівника. Якщо є працівник кращий за вже встановленого, його буде призначено на цю посаду, та ще раз виконано аналіз. Відкликаний (звільнений) працівник отримає нове, більш підходяще місце. В результаті буде визначено положення роботи на основі бажання установи та здібностей працівників. Таким чином заклад матиме більше шансів найняти кваліфікованих працівників.

Критерії роботи алгоритму зручніше всього зберігати у таблицях, в котрих буде відтворено зацікавлення людини в отриманні роботи, та бажання компанії найняти певного фахівця. Для реалізації потрібно проаналізувати та заповнити дві таких таблиці. Для стабільної роботи кількість стовпців та рядків кожної таблиці має співпадати. Кожен елемент a_{ij} буде відповідати пріоритету зацікавленого працівника щодо певної роботи або ж зацікавленого закладу в певних вакансіях. Пріоритети мають бути унікальними, тому числа в комірках не можуть повторюватись. В даних таблицях визначено також визначено зв’язок між пріоритетами майбутніх співробітників та зацікавленого закладу. Робота цього прикладу алгоритму Гейле-Шеплі буде також відбуватись протягом декількох ітерацій. Загальна кількість таких ітерацій залежить від того, наскільки швидко буде досягнуто

найбільш стабільного набору пар робітник-посада в результаті роботи алгоритму.

Таблиця з пріоритетами людей котрі подають свої резюме:

	Programmer	Manager	Marketing	Mechanic	Post Man	Supervisor
Pritz Brown	3	5	2	1	6	4
Sheryl	4	5	3	2	5	1
Robin Hood	1	2	3	5	6	4
Charles	5	2	3	4	1	6
Andysah	2	5	3	6	4	1
Keysha	6	1	4	5	3	2

Рисунок 1.2 – Пріоритети працівників

Таблиця з пріоритетами компанії щодо працівників:

	Pritz Brown	Sheryl	Robin Hood	Charles	Andysah	Keysha
Programmer	1	2	4	5	3	6
Manager	5	4	2	3	1	6
Marketing	4	1	2	5	6	3
Mechanic	3	1	6	5	2	4
Post Man	1	6	4	5	3	2
Supervisor	2	3	4	6	5	1

Рисунок 1.3 – Пріоритети компанії

Перша ітерація буде виглядати наступним чином:

```
Employee = Pritz Brown       <1>
       Placed to      = Mechanic     <4>
       Status         = Permitted
       Reason         = Position Mechanic is available

Employee = Sheryl     <2>
       Placed to      = Supervisor   <6>
       Status         = Permitted
       Reason         = Position Supervisor is available

Employee = Robin Hood <3>
       Placed to      = Programmer   <1>
       Status         = Permitted
       Reason         = Position Programmer is available

Employee = Charles    <4>
       Placed to      = Post Man     <5>
       Status         = Permitted
       Reason         = Position Post Man is available

Employee = Andysah    <5>
       Placed to      = Supervisor   <6>
       New Weight     = 5
       Old Weight     = 3
       Replace        = Sheryl
       Status         = Unchanged
       Reason         = Weight Andysah is lower than Sheryl

Employee = Keysha     <6>
       Placed to      = Manager      <2>
       Status         = Permitted
       Reason         = Position Manager is available

Employee [1] = Pritz Brown   Position [4] = Mechanic
Employee [2] = Sheryl        Position [6] = Supervisor
Employee [3] = Robin Hood    Position [1] = Programmer
Employee [4] = Charles       Position [5] = Post Man
Employee [5] = Andysah       Position [0] = -
Employee [6] = Keysha        Position [2] = Manager
```

Рисунок 1.4 – Перша ітерація алгоритму підбору працівників

В наступних кроках алгоритм буде аналізувати та змінювати позиції елементів, таким чином щоб пріоритет обох вибірок був максимально високим[5, с. 62-67].

Остаточний результат матиме наступну послідовність:

```
Employee [1] = Pritz Brown    Position [3] = Marketing
Employee [2] = Sheryl         Position [4] = Mechanic
Employee [3] = Robin Hood     Position [2] = Manager
Employee [4] = Charles        Position [5] = Post Man
Employee [5] = Andysah        Position [1] = Programmer
Employee [6] = Keysha         Position [6] = Supervisor
```

Рисунок 1.5 – Отримана стабільна вибірка працівник-посада

Робітник ‘Pritz Brown’ отримав позицію Marketing, котра була другою в його списку пріоритетів. Sheril – Mechanic, також отримала позицію котра була другою за списком своїх пріоритетів та першою в списку бажань компанії. Наступні співробітники також отримали перше та друге місце за своїми пріоритетами. В даному випадку пріоритет співробітника грає вищу роль, тому в більшості випадків робітники були на посередніх місцях в пріоритетах компанії, тому що якість роботи більшою мірою залежить від бажання та старань працівників.

Проаналізувавши результати роботи алгоритму можна стверджувати, що було досягнуто оптимальну варіацію пар з працівників та їхніх посад виходячи з їх пріоритетів.

1.4 Дослідження роботи алгоритму для розподілу студентів

Алгоритм Гейле Шеплі також може бути використаний для вирішення задачі розподілу студентів по навчальним закладам. На відміну від оригінального алгоритму стабільних шлюбів, в такому випадку результатом буде вибірка один до багатьох. Існує стабільний розподіл студентів до університетів, який можна знайти в результаті роботи деяким чином видозміненого алгоритму Гейла-Шеплі. У версії алгоритму, в якому пропозиції внесли абітурієнти, єдина відмінність полягає в тому, що університети можуть прийняти декілька пропозицій, але в межах квоти. А у версії алгоритму, в який внесли пропозиції університети, останні можуть направити кілька пропозицій одночасно, але так само в межах квоти.

Нехай A – вибірка студентів, C – набір навчальних закладів, q_u – кількість студентів котрих може прийняти вуз C_u. Так як студент може бути зарахований лише до одного вузу, а вуз при цьому може бути поєднаним з групою абітурієнтів, можна припустити, що кожен заклад, у котрого q більше одиниці має можливість порівняти групи абітурієнтів як альтернативні розподіли. В ситуаціях, коли студент не прийнятий до жодного вузу то A_u є «self-matched». Кількість розподілів для кожного вузу не може перевищувати кількості можливих місць – q. Розподіл студентів по вузам є «двостороннім». Це означає, що студент може бути зарахованим, лише у випадку, якщо він відповідає вимогам навчального закладу. Якщо студент не відповідає вимогам, і при цьому не бажає вступати до конкретного закладу, такий розподіл є індивідуально нераціональним. Розподіл є стабільним, якщо кожного учасника влаштовує його позиція, та не може бути покращено одним або кількома учасниками [3, с. 10-15].

Так як результатом алгоритму мають бути оптимальні розподіли, можна стверджувати що студентам не вигідно вводити до свого списку пріоритетів не правдиві дані. Така проблема узгодження з відповідними пріоритетами є некооперативною «грою» між учнями. Їх виграш – отримання рекомендації до бажаного закладу, а стратегія – заявлені пріоритети. Модель розподілу, що підтверджує цю стратегію, дозволяє студентам визначати свої справжні уподобання, без будь-яких маніпуляцій, для отримання оптимального вибору зі списку їхніх уподобань. Таким чином учні не зможуть отримати кращі рекомендації сфальсифікувавши свої списки пріоритетів [4, с. 2].

Також не існує набору учнів («коаліції») котрі б могли маніпулювати своїми пріоритетами на користь кожного співучасника. Враховуючи вподобання та оцінки студентів можна визначити рейтинг студентів та репутацію навчальних закладів. Таким чином можливо скоординувати роботу програми для максимальної оптимізації даних. При цьому буде

задоволено умову, що кожен учень отримає рекомендацію в один з ВНЗ, і в гіршому випадку останній за пріоритетом із свого списку[4, с. 5].

Стабільність алгоритму також має враховувати чесний розподіл між учасниками. Нехай студент A1, A2 мають оцінки 71 та 72 відповідно з наступною схемою пріоритетів :

[A1: c1 > c2 > c3 ;; A1: 71] , [A2: c1 > c3 > c2 ;; A2: 72]

Враховуючи їхні пріоритети можна помітити, що незважаючи на бали кожного учасника заклад C1 є на першому місці в обох випадках. Для того, щоб модель була стабільною заклад C1 в своєму рейтингу учнів має віддати перевагу A2. При цьому не важливо в якій послідовності працював би алгоритм, в кінцевому результаті, за жодних обставин студент з меншою кількістю балів не може бути вище у рейтингу. У іншому випадку, за перевищення квоти, заклад відхилятиме заявки учасників котрі мають більше балів, що не є прийнятним.

Алгоритм припиняється, коли кожен студент отримує своє остаточне призначення(рекомендацію) до певного закладу. У разі відмови з відведених місць, алгоритм починається з перерозподілу місць іншим претендентам. Централізована система потребує єдиної централізованої форми заявки для всіх закладів. У цій формі студенти класифікують пріоритету у відповідних ВНЗ відповідно до своїх уподобань.

Така централізована система вимагає створення двох механізмів:

1. Механізм виявлення уподобань учнів

2. Механізм для об'єднання цих визначених уподобань та розподілу місць для відповідних університетів

Діаграма роботи алгоритму виглядає наступним чином [4, с. 6]:

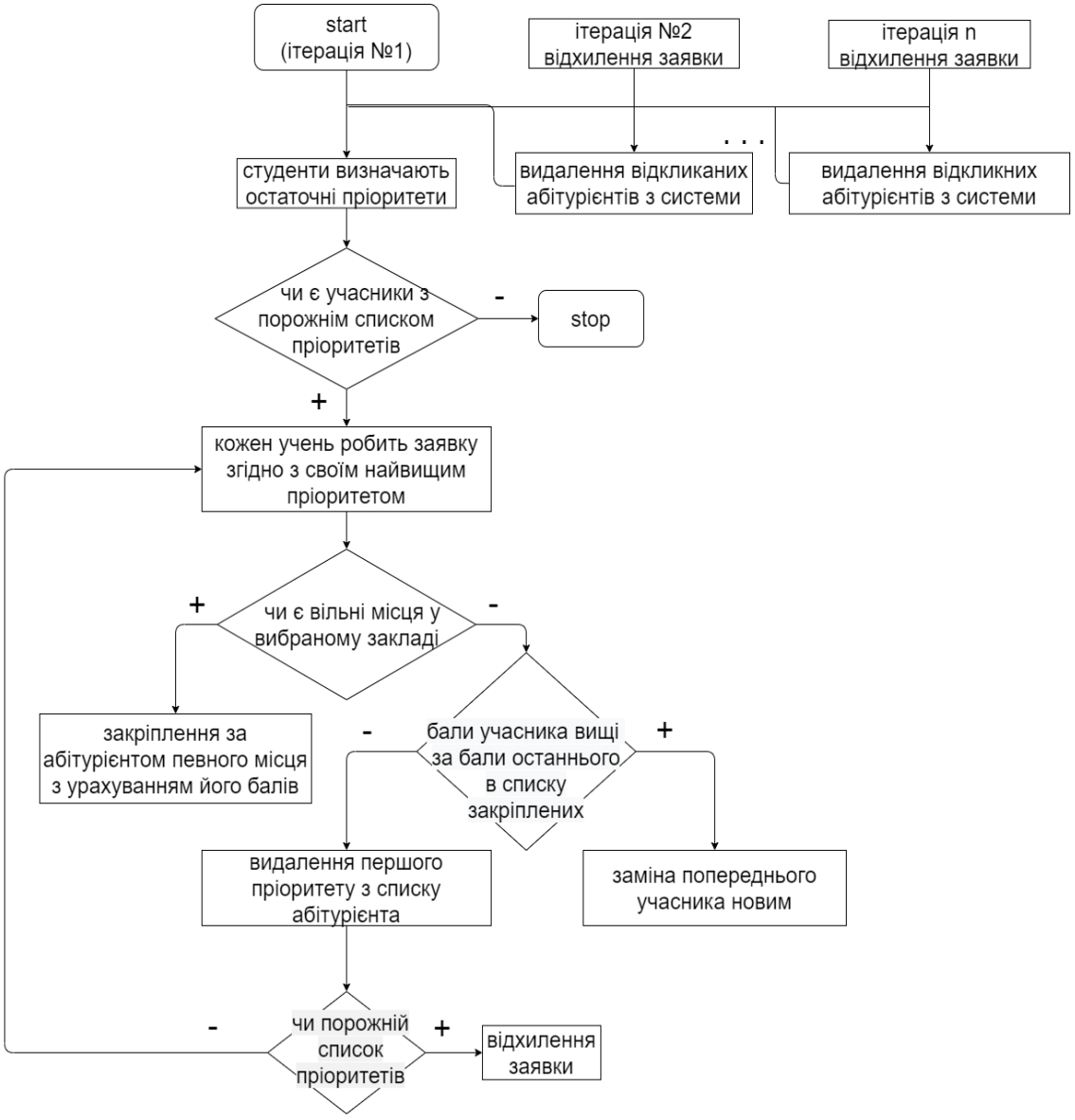

Рисунок 1.6 – Алгоритм розподілу студентів

Висновки до першого розділу

В результаті дослідження алгоритму Гейла-Шеплі було виявлено деякі особливості, переваги та недоліки. Однією з головних особливостей даного алгоритму є те, що стабільні пари елементів будуть утворені з більшою вигодою для однієї зі сторін. Не зважаючи на те, що отримані результати

можна вважати оптимальними, не можна виключати можливість знаходження більш стабільного розв’язку. Крім того даний факт свідчить про те, що одна зі сторін, з не великою вірогідністю, але може маніпулювати пріоритетами для власної вигоди. Також в залежно від поставлених цілей робота алгоритму може різнитись. Наприклад, на це може впливати розмір вибірки, пріоритетів, пріоритети якої вибірки грають важливішу роль на результати та інші.

До переваг можна віднести:

- Завжди є можливість знайти оптимальний розв’язок
- Не великий шанс фальсифікації пріоритетів для отримання власної вигоди
- Складність алгоритму для знаходження одного оптимального рішення складає O(n^2).

Недоліки алгоритму:

- Оптимальний розв’язок завжди більш вигідний для однієї зі сторін
- Існує невеликий шанс маніпулюванням пріоритетами
- Для великих розмірів початкової вибірки ускладняється робота алгоритму.

РОЗДІЛ 2
ПОБУДОВА МОДЕЛІ РОЗПОДІЛУ ОСНОВНИХ АЛГОРИТМІВ СТАБІЛЬНОГО УЗГОДЖЕННЯ

2.1 Концептуальна схема проектованого

Основним завданням роботи алгоритму Гейла-Шеплі є вирішення проблем пов'язаних із знаходженням стабільних зв'язків між двома масивами елементів. Можна зазначити, що стабільні пари елементів не є обов'язково унікальними, так як не завжди можна досягнути ідеального співпадіння пріоритетів між членами вибірки. З використанням алгоритму Gale-Shapley можлива реалізація програм для вибору не тільки стабільних пар, але складних вибірок із зв'язком один до багатьох, та багато до багатьох.

При підвищенні розміру вибірки, наприклад, двох масивів чоловіків та жінок робота алгоритму ускладняється. Тому було б не розумно для ста жінок послідовно ранжувати сто чоловіків. В такому випадку, для знаходження оптимального розв'язку раціонально обирати пари елементів з порівнянням трьох найвищих:1,2,3 та трьох найнижчих пріоритетів 98, 99 , 100.

Складність алгоритму Гейла-Шеплі поліноміальна – для кількості елементів n – O(p(n)) >= n^2 , де n – розмір вибірки елементів. Очевидно, що для при збільшенні n розрахунки стануть занадто складними, та громіздкими, для того хто захоче скористатись цим алгоритмом. Тому для здійснення таких задач краще всього використовувати комп'ютерну програму, в котрій реалізовано алгоритм Gale-Shapley. Найпростішою та найпершою проблемою з використанням даного підходу була «проблема стабільних шлюбів». Алгоритм Гейла-Шеплі був розроблений для поєднання чоловіків і жінок, які мали висловлювати свої індивідуальні уподобання один про одного. В результаті роботи алгоритму мають утворитись стабільні пари з елементів. Вибірка вважається стабільною, якщо жодному елементу,

виходячи з його пріоритетів не вигідно змінювати свого партнера. Якщо є сто жінок та сто чоловіків з кожним пріоритетом, алгоритм справді зможе утворити 100 стабільні пари. Така проблема не є досить реалістичною, однак змогла привернути великий інтерес серед математиків.

2.2 Дослідження алгоритму для вибірки один до одного

Для прикладу, під час дослідження алгоритму Гейла-Шеплі можна взяти вибірку з 4 пар чоловіків та жінок. Для зручності, краще всього заповнити таблицю з елементами вибірки та їх перевагами.

Наступна таблиця описує пріоритети кожного елемента вибірки:

	Michelle	Teresa	Kim	Ashley
Brent	1,3	2,2	3,1	4,3
Mike	1,4	2,3	3,2	4,4
Jason	3,1	1,4	2,3	4,2
Dennis	2,2	3,1	1,4	4,1

Рисунок 2.1 – Пріоритети чоловіків та жінок

Перше число кожної комірки відповідає за пріоритет чоловіка до певної жінки. Друге число відповідає за пріоритети жінок. В програмі буде використано ці пріоритети для вибору стабільних пар. На першому кроці буде чоловіки обирають жінок з найвищим пріоритетом. Якщо два чоловіки обирають одну і ту ж саму жінку, вона обирає чоловіка згідно зі своїми пріоритетами. Чоловік, котрий залишився без пари обирає наступну претендентку з найвищим пріоритетом [6, с. 2-5].

Після першої ітерації буде утворено наступні пари:

Brent, Michelle

Mike, Michelle

Jason, Teresa

Dennis,Kim

Так як Brent та Mike обрали Michele вона має визначити хто підходить їй більше. Перші два пріоритети Michele тимчасово зайняті, тому вона не може обрати партнера із третім пріоритетом. Після цього Mike обирає свій другий пріоритет – Teresa.

На другому кроці пари виглядають наступним чином:

Brent, Michelle

Mike, Teresa

Jason, Teresa

Denis, Kim

Тепер Mike та Jason обирають Teresa, тому вона обирає між ними. Пріоритет Mike – 3, вищий ніж Jason, тому вона обирає за свою пару Mike. Далі Jason має обрати наступну за списком жінку – другий пріоритет Kim.

Третій крок:

Brent, Michelle

Mike, Teresa,

Jason, Kim

Даний процес буде повторюватись до тих пір, поки не буде знайдено стабільну варіацію з пар чоловік-жінка. В остаточній ітерації буде отримано такі пари:

Brent, Kim

Mike, Ashley

Jason, Michelle

Denis, Teresa

На рис 2.2 можна побачити яким чином відбувався розподіл між елементами масивів для вибірки чоловіків та жінок.

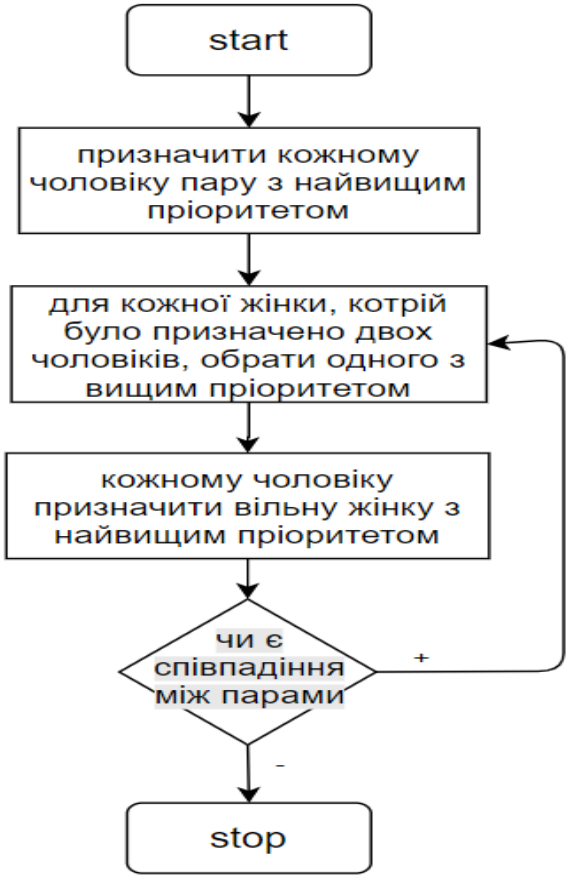

Рисунок 2.2 – Діаграма роботи програми для жінок

```
Couples are :
Jason Michelle
3       1
Dennis Teresa
3       1
Brent Kim
3       1
Mike Ashley
4       4
```

Рис 2.3 Пари утворені для жінок

В результаті роботи алгоритму було отримано пари елементів чоловік-жінка (рис 2.3). Під кожним елементом виведено номер пріоритету кожного

учасника до його пари. Дану варіацію можна вважати стабільною. Проте, слід звернути увагу, що жінки переважно отримали пару з вищим пріоритетом – 1, в той час як чоловіки - 3. Така інтерпретація алгоритму, можливо, в залежності від обставин експерименту та пріоритетів, не завжди даватиме найбільш оптимальний результат. Згідно з такими міркуваннями, слід також розглядати дослідити алгоритм, в результаті роботи якого чоловіки отримають пару з вищим пріоритетом.

У даній інтерпретації алгоритму кожному чоловіку було призначено вільну жінку з найвищим пріоритетом. Під час збігу пріоритетів чоловіків, жінки могли обирати одного з вищим пріоритетом. Таким чином жінки є в більш вигідній позиції. Розглянемо можливість створення програми, в результаті роботи якої буде отримано більш вигідні оцінки з точки зору іншої вибірки.

Проаналізуємо роботу алгоритму для вибору пари чоловікам:

```
Couples are :
Brent Michelle
3   1
Mike Teresa
3   2
Jason Kim
3   2
Dennis Ashley
1   4
```

Рисунок 2.4 – Пари утворені для чоловіків

З виведених даних вибірки для чоловіків (рис 2.4) можна побачити, що пріоритети їх пар вищі ніж минулого разу : 1, 2, 2, 4. Отже, перед інтерпретацією алгоритму потрібно визначити цілі, та пріоритети якої вибірки грають більшу роль під час знаходження оптимального узгодження між елементами. В залежності від цілей можна використовувати перший або другий спосіб реалізації. У випадку, коли оптимальне рішення потрібно

знайти не залежно від того яка вибірка має бути основною, можна порахувати суму всіх пріоритетів. Чим менша сума пріоритетів, тим стабільніші пари було отримано. Під час дослідження було виявлено, що для знаходження максимально стабільного набору пар об’єктів у рівнозначних вибірках краще за все проводити два експерименти. Такий підхід до виконання подібних задач є складнішим, однак гарантує можливість обрання максимальної варіації стабільного узгодження.

2.3 Дослідження алгоритму для вибірки один до багатьох

Яскравим прикладом для реалізації вибірки зі зв’язком одного до багатьох є розподіл студентів по навчальним закладам. Логіка роботи алгоритму в даному випадку є набагато складнішою ніж у вибірці один до одного.

Розглянемо яким чином відбуватиметься розподіл десяти студентів по трьом закладам. Спершу, потрібно визначити списки студентів та університетів. Окрім пріоритетів, у даному випадку основні об’єкти вибірок характеризуватимуть також статус заявки для студентів, та кількість вільних місць для університетів.

Параметри студентів:

- ім’я;
- статус(«вільний» / «зайнятий») для відображення, чи прийнято заявку студента;
- Список бажаних університетів (пріоритети).

Параметри університетів:

- назва;
- кількість вільних місць (квота);
- рейтинг студентів (пріоритети);

Кожен студент подаватиме заявку до університету згідно з його максимальним пріоритетом. Якщо в університеті є вільні місця, то студента буде закріплено, надано йому статус «зайнятий» (matched), та виконано перехід до наступної ітерації. Після кожної ітерації, для максимально точної роботи програми, на кожному навчальному закладі потрібно відсортувати студентів за рейтингом. Якщо під час подачі, заявки вільних місць не залишається, то поточний студент порівнюється з уже призначеними. У випадку коли його рейтинг у списку преференцій університету нижчий, ніж у останнього з тих, хто уже призначений на місця, його заявку буде відхилено. При цьому немає потреби у повторному сортуванні вибірки. У іншому випадку, коли рейтинг студента, який подає заявку є вищим, ніж в останнього у списку, заявку буде прийнято, а попереднього – видалено зі списку прийнятих. Статус нового при цьому також зміниться на «зайнятий», а попереднього на «вільний». Так як рейтинг нового студента може бути вищим ніж тих, хто вже є в списку з попередньої ітерації, потрібно знову виконати сортування за рейтингом. Даний функціонал описано в діаграмі роботи алгоритму розподілу студентів (рис. 1.6).

Логічна складова алгоритму та досліджуваних об’єктів:

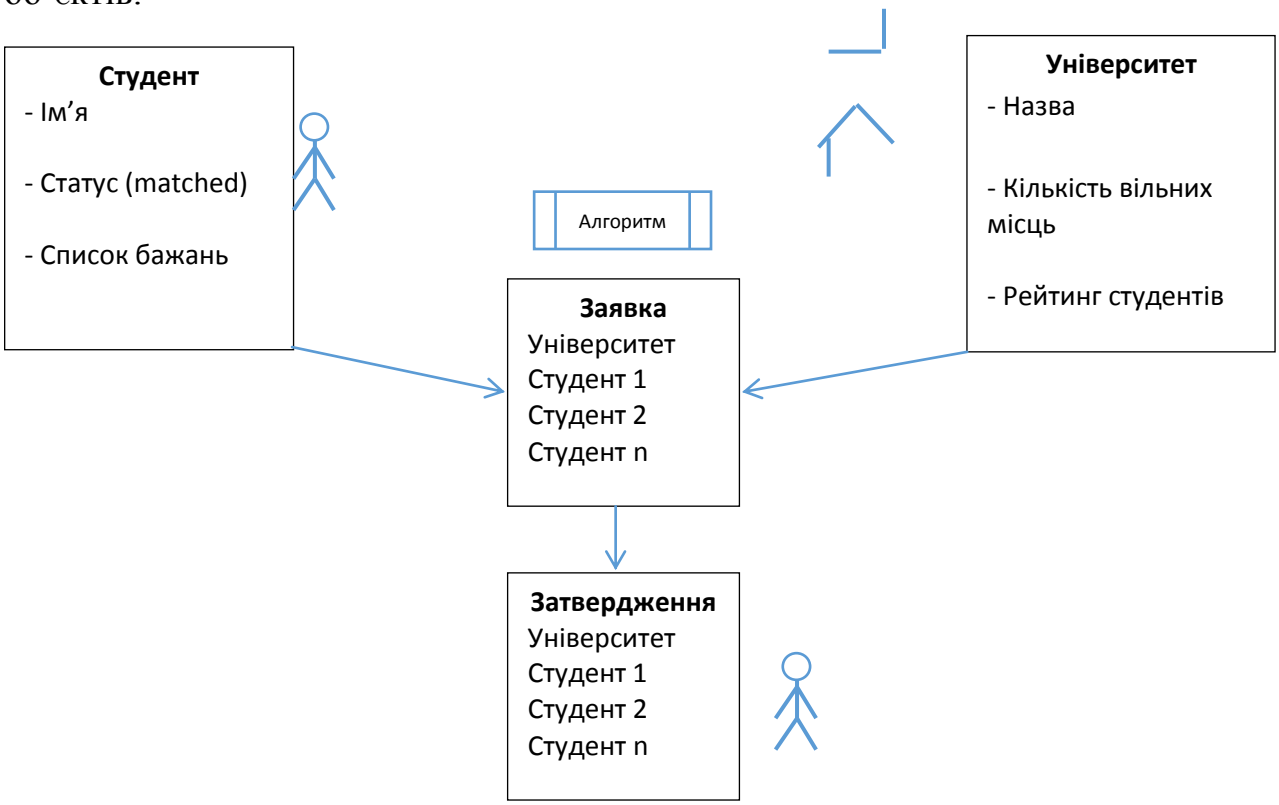

Рисунок 2.5 – Взаємозв’язки та параметри об’єктів

Після закінчення роботи алгоритму було утворено такі вибірки:

```
Matchings are:
G-University:[Alex, Jane, Elzabeth]
S-University:[Nina, Amily, Adam]
K-University:[Bob, Kevin, Robert]
```

Рисунок 2.6 – Оптимальні набори з університетів та студентів

Порівнявши пріоритети студентів та університетів, помітно, що майже кожного студента було зараховано до бажаного закладу. У випадках, коли кількість студентів перевищувала максимальну, а рейтинг нового учня, котрий подає заявку вищий ніж в учня з найнижчим рейтингом , то навчальний заклад відхиляє заявку другого, та приймає того, у кого рейтинг вищий.

Згідно з цим, можна стверджувати, що такий алгоритм дає змогу отримати стабільні списки абітурієнтів та навчальних закладів. При цьому бажання учнів грають вищу роль в розподіленні, однак при заповненні вільних місць в університетах, останні могли зараховувати студентів з вищим рейтингом.

Висновки до другого розділу

Під час написання другого розділу було проаналізовано та використано результати дослідження першого розділу. В другому розділі було описано технології реалізації різних варіацій алгоритму Гейла-Шеплі. Під час написання програми з використанням даного алгоритму було взято до уваги його особливості. У вибірці один до одного, та один до багатьох алгоритм дає різні результати, в залежності до того, пріоритети якої вибірки були на першому місці. Для створення програми з використанням зв’язку один до багатьох обрано вибірки студентів та університетів. Бажання студенів при цьому грає вищу роль у створенні розподілу. Пріоритет студентів поставлено на перше місце, саме тому , що такий підхід максимально наближений до реальності. Окрім того, під час такого розподілу забезпечується здорова конкуренція між студентами.

РОЗДІЛ 3
СТВОРЕННЯ СИСТЕМИ РОЗПОДІЛУ СТАТЕЙ РЕЦЕНЗЕНТАМ

3.1 Обрані технології. PHP. Laravel

PHP - (рекурсивний акронім для PHP : гіпертекстовий препроцесор) - це широко використовувана мова програмування загального призначення з відкритим кодом, яка особливо добре підходить для веб-розробки та вбудови в HTML. PHP – є потужною мовою програмування та інтерпретатором, що може бути вбудована до веб-серверу як модуль або запущена як окрема бібліотека CGI, що має доступ до файлів, може виконувати команди та відкривати підключення до веб-серверу. Дана мова програмування створена враховуючи особливість бути безпечною для написання CGI програм. З правильним налаштуванням конфігурації компілятора, інтерпретатора та правильному підходу до написання коду можна досягнути створення вільного та безпечного веб-сайту. Так само як і велика кількість способів використання PHP також є і великий вибір конфігурацій контролюючих його роботу. Велика кількість опцій гарантує, що ви можете створювати додатки з різним функціоналом та з різною метою. Окрім цього, гнучкість налаштувань конфігурації PHP конкурує з гнучким написанням коду. PHP може бути використаним для створення повноцінних веб додатків.

Замість того щоб писати велику кількість команд для виводу в HTML - наприклад як в C, файли мови програмування PHP вміщують в собі вбудований код, що виконує деякі функції. У файл HTML можна підключити php , відкривши спеціальний тег : <?php ?>. Це дуже зручно, адже в будь-який момент дозволяє переключатись між html та php. В даний блок можна вставити код , котрий буде оброблено гіпертекстовим препроцесором як php. Наприклад, для виводу деякої інформації в браузері можна використати команду echo :

```
<?php echo "This is some information"; ?>
```

Що відрізняє PHP, від клієнт-частини , такої як javascript , це те , що код обробляється на серверній частині, генеруючи файли html, котрі лише потім відправлено до клієнт-частини. Користувач може отримати результати котрі надійшли з сервера, однак не матиме доступу до основного коду , з допомогою якого було отримано дані. Крім того, можливо навіть налаштувати свій власний веб-сервер , для обробки усіх html файлів проекту , і тоді користувач не матиме жодної можливості отримати доступ до коду.

Однією з переваг мови програмування php є її легкість в освоєнні для нового користувача. Однак при цьому, в досвідчених користувачів є можливість впровадження додаткового функціоналу до свого веб-сайту. В офіційній документації можна знайти інформацію про основні та додаткові функції для розширення функціоналу. Новий користувач може швидко ознайомитись з основами php, та вже за кілька годин приступити до створення проекту.

PHP головною мірою призначений для програмування серверної частини сайтів, тому з його допомогою можна робити, все, що й інші CGI-програми можуть досягти.

Основні функції php:

- збір даних серверу;
- генерація динамічного контенту;
- надсилання та отримання даних з cookies;

Є три основні підходи, де використовуються сценарії php. Програмування серверної частини сайту - це традиційний та найбільш розповсюджений спосіб використання php. Вам потрібно три основних речі для того, щоб почати використання : парсер php (модуль серверу або CGI), веб-сервер та браузер. Для роботи потрібно запустити веб-сервер з

встановленим та підключеним php. У вас при цьому є доступ до програмного виводу з веб-браузеру, переглядаючи сторінки через сервер.

Програмування з допомогою консолі – можливість запустити PHP код без попереднього запуску серверу та без потреби у браузері. Для цього потрібно лише встановити парсер PHP. Такий підхід зазвичай використовують для створення додатків Linux систем, а також для «планувальника задач» Windows. Також можна створювати прості додатки для обробки тексту.

Створення додатків з інтерфейсом користувача – мова PHP більшою мірою не призначена для створення повноцінних додатків, однак з використанням додаткових модулів це можливо. Для досвідчених користувачів є можливість використання розширення PHP-GTK, що надає доступ до багатьох додаткових функцій пов’язаних з інтерфейсом користувача.

PHP можна використовувати на усіх основних операційних системах, таких як Linux, Unix, Microsoft Windows, macOS та ін. Також дана мова підтримує зв’язок з більшістю сучасних веб-серверів (Apache, IIS), і таких що працюють на базі бібліотеки FastCGI PHP (lighttpd, nginx). При цьому PHP може працювати як модуль або як препроцесор CGI.

Тому з використанням PHP ви можете обрати операційну систему а також маєте свободу у виборі веб-серверу.

3.2 Проєктування бази даних

Перед початком застосування алгоритму для знаходження рецензентів для статей потрібно спроєктувати базу даних. Для цього визначимо сутності котрі будуть задіяні в програмі. Основними сутностями будуть таблиця статей і таблиця рецензентів.

Статті належатимуть такі поля : назва(name) – тип varchar, тема(theme) – тип varchar, мова статті (language) – тип varchar, статус (is_reviewed) – тип tinyint, автор (author) – тип varchar, ключові слова (keywords) – тип text. Статус is_reviewd використовується для позначення таких властивостей: нова (0-new), на рецензії (1-on review), відрецензована (2-reviewed). Коли статтю тільки додано до бази даних її статус встановлюється як нова (0-new), після вибору одного з рецензентів та його призначення до цієї статті статус встановлюється на рецензії (1-on review), а потім може бути встановленим як відрецензована(2-reviewed).

Рецензента характеризуватимуть наступні поля: ім’я (first_name, last_name) – тип varchar, теми(themes) – json, мови (languages) – json, кількість рецензій – тип int. Список тем (themes) – масив у котрому будуть знаходитись теми , з якими може працювати рецензент. Кожна тема в даному списку матиме пріоритет, за її номером у масиві. Даний пріоритет гратиме роль у роботі алгоритму.

Додатково також ще створено дві таблиці бази даних: пропозиції (suggestions) та призначені публікації(assigned_publications). Пропозиції матимуть поле з ідентифікатором(id) статті, та списком ідентифікаторів рецензентів, котрих було визначено в результаті роботи алгоритму. Таблиця призначених публікацій матиме ідентифікатор статті, та призначеного рецензента, одного із запропонованих. Коли статтю додано до таблиці призначених, запис із номером даної статті в таблиці пропозиції має бути видаленим, а статус статті позначено on_review. Структуру бази даних можна побачити на рис. 3.1 .

Нормалізація бази даних. Розглянемо можливість приведення бази даних нормальної форми. Для цього кожен елемент у таблицях сутностей має містити лише одне значення, тому для цього у рецензентів потрібно видалити поля мови і теми, та для них створено нові, додаткові таблиці. Розглянемо

зв’язки у базі даних. Із таблицею статей пов’язано дві інших: таблиця призначених статей, і таблиця пропозицій. Таким чином кожна пропозиція має номер статті, для котрої була призначена. Таблиця призначених статей має зв’язок з рецензентами та статями. Структуру нормалізованої бази даних зображено на рис. 3.2 .

Рисунок 3.1 – Діаграма бази даних

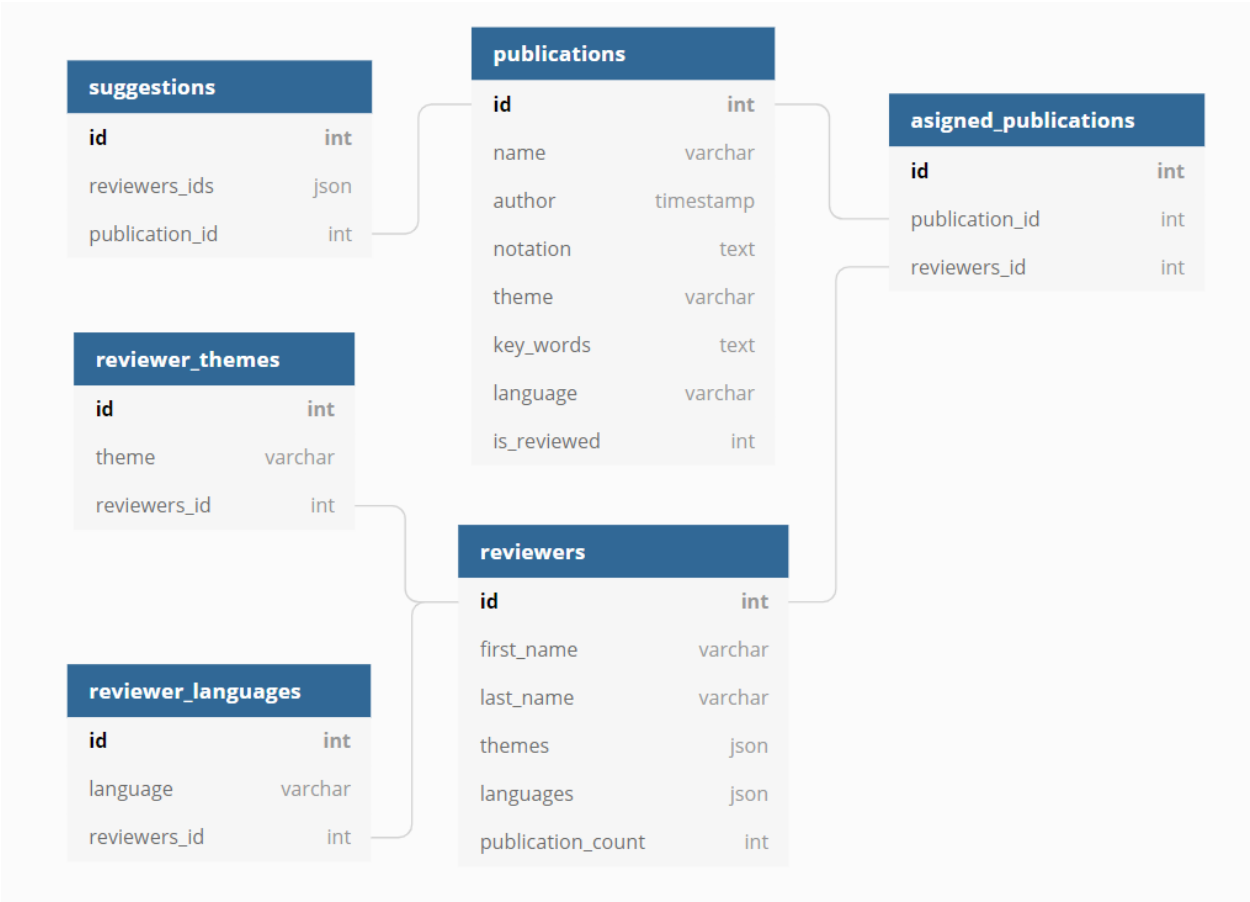

Рисунок 3.2 – Діаграма нормалізованої бази даних

3.3 Застосування алгоритму та створення серверної частини

Досліджений алгоритм застосуємо для підбору рецензентів до статей, інтегруючи його до веб системи. Програма має знаходити рецензентів до окремої статті або до декількох статей. Підбір буде відбуватись за такими критеріями:

1. Мова статті співпадає з мовою рецензента.
2. Тема статті є у списку тем (пріоритетів) рецензента.
3. Номер статті в списку рецензента впливає на роботу алгоритму.

Під час знаходження рецензентів для деякої статті буде порівнюватись співпадіння мови, тобто якщо в списку мов рецензента немає мови обраної статті, його не буде включено до списку запропонованих. Алгоритм буде рухатись далі, та перевірятиме наступний критерій. Якщо тема статті не співпадає з жодною з тем рецензента його не буде включено до списку, та

відбудеться перехід до наступного рецензента. Якщо перші два критерії успішно виконано, та перевірка рецензента задовольняє їх умови, буде застосовано останній критерій – вибір рецензента за його пріоритетами.

Ключову роль у варіаціях стабільного узгодження алгоритму Гейле-Шеплі грають саме пріоритети об’єктів до яких його застосовано. В даному випадку алгоритм працюватиме з вибіркою один до багатьох та багато до багатьох. Тобто, до однієї, обраної статті буде знайдено декількох підходящих рецензентів. Однак, знаходити рецензентів до кожної статті окремо є не достатньо зручним. Зважаючи на це в програмі має бути можливість обрати декілька статей та застосувати алгоритм до кожної з них, що створить вибірку багато до багатьох. В даному випадку пріоритетами статті є її тема та мова, однак у рецензента це список з тем та мов. В даній роботі розглянемо випадок, коли теми рецензента розміщено в його списку за порядком зростання. Таким чином чим менший номер у масиві тим вищий пріоритет даної теми для рецензента. Під час роботи алгоритму пріоритети тем кожного рецензента будуть порівнюватись. Якщо пріоритет теми одного рецензента нижчий ніж пріоритет іншого, його буде замінено. Таким чином до списку запропонованих буде внесено рецензента з вищим пріоритетом теми.

Якщо додавати до списку всіх рецензентів тема котрих співпадає, можливо, буде знайдено дуже багато кандидатів, тому краще всього підібрати лише тих, хто максимально підходить на дану роль. В програмі буде можливість обрати максимальну кількість запропонованих рецензентів. Оптимальним варіантом буде знайти від одно до п’яти рецензентів, за замовчуванням буде відбуватись пошук трьох рецензентів. Для забезпечення такої логіки, краще всього буде відсортувати рецензентів за номером їх пріоритету теми, після кожної ітерації, коли список було оновлено. Оптимальні рецензенти будуть внесені до списку, поки його максимальна кількість не перевищена. У випадку, коли максимальну кількість

перевищено, буде взято рецензента, пріоритет теми якого найнижчий, та замінено на нового. Так як список було відсортовано, буде достатньо обрати останнього зі списку. Після того як нового рецензента додано на місце останнього, його пріоритет теми не обов’язково є найнижчим зі списку, тому сортування потрібно виконати ще раз. Схему роботи даного алгоритму можна побачити на рис. 3.3 .

Під час аналізу алгоритму було виявлено, що одна зі сторін завжди є у більш вигідній позиції. В даному випадку така проблема не виникає. Алгоритм спрямовано на знаходження декількох рецензентів, та можливості обрати одного з них. При цьому в статей немає точного списку пріоритетів що до рецензентів, а лише для їх теми та мови. Основною метою програми є саме пошук оптимальних рецензентів, а не збігу між двома списками об’єктів, як в інших варіаціях алгоритму Гейла-Шеплі. Таким чином результатом алгоритму завжди буде оптимальний список рецензентів для однієї або декількох статей.

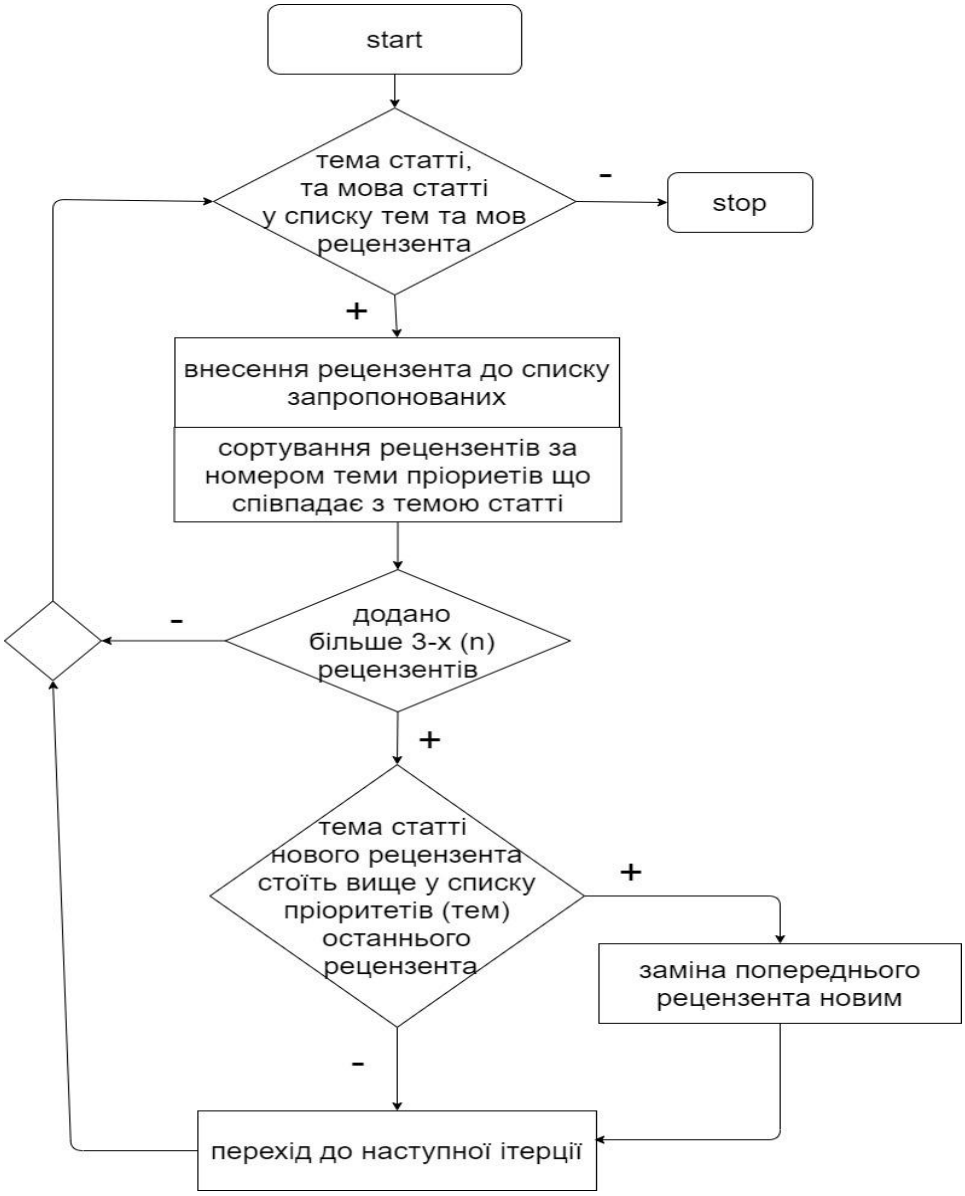

Рис 3.3 Схема роботи алгоритму.

Для застосування алгоритму підбору рецензентів на практиці творимо проект на базі мови PHP та платформи Laravel. Основними сутностями проєкту є такі файли: Controllers, Routes, Migrations, та файли що відповідають за створення візуальної частини для взаємодії з користувачем. У файлах Controllers створюються класи, котрі відповідають за групування логіки для обробки запитів до даних , отримання та надсилання інформації. Дані файли в проєкті розташовані за шляхом : « app/Http/Controllers ». Саме в

цих файлах застосовано логіку роботи алгоритму Гейла-Шеплі, а саме у класі SuggestionController. В цьому класі створено функцію (processData) вхідними даними котрої є номер статті а вихідними – список запропонованих рецензентів. Функцію можна застосувати одразу для декількох статей, таким чином отримавши зв’язок багато до багатьох. В файлах Routes налаштовано URL - посилання , котрі працюватимуть в майбутньому сайті. Migrations – файли, в котрих відбувається налаштування бази даних, а саме створення таблиць, полів та типів даних, забезпечення зв’язку між полями різних таблиць. В теці /views створено файли, котрі відповідають за візуальну частину сайту, та взаємодію з користувачем.

Висновки до третього розділу

Під час написання третього розділу було проаналізовано та застосовано алгоритм Гейла-Шеплі для розподілу рецензентів до статей. Було спроєктовано базу даних для статей та рецензентів. На базі мови PHP та платформи Laravel створено функціонал для запитів до бази даних та їх обробки. На прикладі веб-сайту алгоритм Гейла Шеплі застосовано для знаходження оптимального списку рецензентів до статей. Під час створення практичної частини також було реалізовано додатковий функціонал, а саме: можливість застосувати алгоритм як до окремої статті, так і для декількох одразу; додано можливість сортувати відрецензовані ,нові та статті на рецензії; додано можливість обрати максимальну кількість знайдених рецензентів.

ВИСНОВКИ

У процесі дослідження було досягнуто поставленої мети, а саме: проаналізовано алгоритм стабільного узгодження; спроєктовано базу даних статей та рецензентів; мовою PHP (Laravel) створено платформу для підбору рецензентів до статей з використанням алгоритму стабільного узгодження.

Алгоритм Гейла-Шеплі було розроблено для створення стабільних пар елементів, згідно з їх пріоритетами(бажаннями). Учені Девід Гейл та Лойд Шеплі вперше застосували алгоритм до проблеми «стабільних шлюбів», однак в подальшому їх праці було використано для більш актуальних проблем. Алгоритм «стабільних пар» використовують для пошуку оптимального місця роботи, призначення людей на певні посади, розподілу студентів по навчальним закладам, пошуку біологічно-сумісних донорів для хворих та в багатьох інших сферах. Особливістю алгоритму є те, що одна зі сторін завжди виявляється у більш вигідній позиції.

У другому розділі дипломної роботи було досліджено існуючі варіації застосування алгоритму Гейла-Шеплі. У вибірці один до одного, та один до багатьох алгоритм дає різні результати, в залежності до того, пріоритети якої вибірки були на першому місці. Для аналізу алгоритму з використанням зв’язку один до багатьох обрано вибірки студентів та університетів. Бажання студенів при цьому грає вищу роль у створенні розподілу. Пріоритет студентів поставлено на перше місце, саме тому, що такий підхід максимально наближений до реальності. Окрім того, під час такого розподілу забезпечується здорова конкуренція між студентами

Під час дослідження було виявлено переваги та недоліки алгоритму стабільного узгодження.

До переваг алгоритму можна віднести:

- Відносно висока швидкість роботи

- Завжди є можливість отримати стабільні вибірки елементів
- Учасникам вигідно вказувати свої правдиві бажання в пріоритетах

Недоліки:

- Одна зі сторін є у більш вигідній позиції
- В деяких випадках існує невеликий шанс маніпуляції пріоритетами
- Під час роботи з великим об’ємом даних ускладняється робота алгоритму

У практичній частині алгоритм стабільного узгодження було застосовано для підбору рецензентів до статей, на базі мови PHP та платформи Laravel. Базу даних до статей та рецензентів спроєктовано, та досліджено можливість її нормалізації. Після застосування алгоритму для підбору рецензентів, було виявлено, що одна зі сторін в даному випадку не є у більш вигідній позиції, через особливості розташування пріоритетів у вибірках. Для взаємодії з користувачем, з допомогою засобів Laravel та інструментів Bootstrap створений веб-сайт з простим дизайном. Інтерфейс користувача та в алгоритмі передбачено особливість підбору рецензентів як для однієї статті, так і для декількох. Результатом роботи алгоритму є вибірки статей та рецензентів, що є оптимальними. До кожної статті, можна призначити одного із запропонованих рецензентів. Розроблене програмне забезпечення значно полегшує та автоматизує пошук рецензентів та подальше призначення одного з них для рецензування статті.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ABOUOUF, Menatalla, et al. Gale-shapley matching game selection—A framework for user satisfaction. IEEE Access, 2018, 7: 3694-3703.
2. ARNOLD, Ken, et al. The Java programming language. Reading: Addison-wesley, 2000.
3. AXTELL, Robert L.; KIMBROUGH, Steven O. The high cost of stability in two-sided matching: How much social welfare should be sacrificed in the pursuit of stability. In: Proceedings of the 2008 world congress on social simulation (WCSS-08). 2008.
4. BHATIA, Abhishek; SHARMA, Chetan; GOYAL, Rinkaj. Development of an agent based model illustrating the usage of deferred acceptance algorithm in the admission process. PeerJ PrePrints, 2015.
5. ELVIWANI, E.; SIAHAAN, A. Putera Utama; FITRIANA, Liza. Performance-based Stable Matching using Gale-Shapley Algorithm. In: Proceedings of the Joint Workshop KO2PI and The 1st International Conference on Advance & Scientific Innovation. 2018.
6. EVERED, Lisa J.; MAY, William. A JAVA Program for the Gale-Shapley Algorithm.
7. FARRELL, Joyce. Java programming. Cengage Learning, 2011.
8. GALE, David; SOTOMAYOR, Marilda. Some remarks on the stable matching problem. Discrete Applied Mathematics, 1985, 11.3: 223-232.
9. ROTH, Alvin E. The college admissions problem is not equivalent to the marriage problem. Journal of economic Theory, 1985, 36.2: 277-288.
10. ROTH, Alvin E.; SOTOMAYOR, Marilda. Two-sided matching. Handbook of game theory with economic applications, 1992, 1: 485-541.
11. TEO, Chung-Piaw; SETHURAMAN, Jay; TAN, Wee-Peng. Gale-shapley stable marriage problem revisited: Strategic issues and applications. Management Science, 2001, 47.9: 1252-1267.

12. Офіційний сайт Bootstrap – режим доступу https://getbootstrap.com/ - дата доступу 04.01.2021.
13. Офіційний сайт PHP – режим доступу https://www.php.net/ - дата доступу 03.20.21.
14. Офіційний сайт Laravel – режим доступу https://laravel.com/ - дата доступу 03.20.21.
15. Портал вивчення технологій програмування – режим доступу https://www.w3schools.com/ - дата доступу 03.05.2021.

ДОДАТОК А