МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РЕАЛІЗАЦІЯ АЛГОРИТМУ ОПТИМІЗАЦІЇ ЗМІСТУ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ»**

Виконала: студентка 2 курсу, групи 8.1229

спеціальності 122 комп'ютерні науки
(шифр і назва спеціальності)

освітньої програми комп'ютерні науки
(назва освітньої програми)

О. О. Милосердова
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, доцент, к. пед. н. Пшенична О. С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент декан математичного факультету, професор, д.т.н. Гоменюк С.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя
2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 комп'ютерні науки
(шифр і назва)

Освітня програма комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук, к.т.н., доцент

_____ Борю С.Ю.
(підпис)

« ____ » _____ 2020 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Милосердовій Олександрі Олександрівні
(прізвище, ім'я та по-батькові)

1. Тема роботи (проекту) Реалізація алгоритму оптимізації змісту навчальної дисципліни

керівник роботи (проекту) Пшенична Олена Станіславівна, к.пед.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » травня 2020 року № 576-с

2. Строк подання студентом роботи 01.12.2020

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Розробка алгоритму оптимізації.
4. Програмна реалізація алгоритму

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 23.05.2020

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	26.05.2020	
2.	Збір вихідних даних.	25.08.2020	
3.	Обробка методичних та теоретичних		
	джерел.	09.09.20	
4.	Розробка першого розділу.	15.10.20	
5.	Розробка другого розділу.	19.11.20	
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи.	01.12.20	
7.	Захист кваліфікаційної роботи.	15.12.20	

Студент _____ О.О. Милосердова
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.С. Пшенична
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Реалізація алгоритму оптимізації змісту навчальної дисципліни»: 58 с., 22 рис., 6 табл., 15 джерел, 2 додатки.

АЛГОРИТМИ ОПТИМІЗАЦІЇ, ЗМІСТ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ, КОНТЕНТ-АНАЛІЗ, ТЕОРІЯ ГРАФІВ, ОПТИМІЗАЦІЯ.

Об’єкт дослідження – алгоритм оптимізації змісту навчальної дисципліни.

Мета роботи – реалізація алгоритму оптимізації змісту навчальної дисципліни.

Методи дослідження – описовий, порівняльний, алгоритмізація, програмування.

Результат дипломного проекту – розроблений та програмно реалізований алгоритм оптимізації змісту навчальної дисципліни.

Потреба у оптимізації навчання с кожним днем стає все більш вираженою, оскільки зменшується час, що витрачається на вивчання окремих дисциплін, передбачених курсом, та збільшується обсяг інформації, яку можна вивчати. Існує багато шляхів оптимізації навчального процесу. В даній роботи розглянуто один з них, а саме вибір оптимальної послідовності вивчання розділів. Реалізація вибору проходить у два етапи. На першому етапі зміст дисципліни аналізується за допомогою засобів контент-аналізу. На другому етапі матриця суміжності, отримана з результатів контент-аналізу, піддається перетворенням, в результаті яких формується масив. Даний масив містить запропонований оптимальний порядок вивчання дисциплін. Таким чином, орієнтуючись на результати роботи програмного забезпечення, матимемо можливість автоматизувати аналіз змісту навчальних дисциплін та покращити їх вивчання.

SUMMARY

Master’s Qualification Thesis «Implementation of the algorithm for optimizing the academic discipline content»: 58 pages, 22 figures, 6 tables, 15 references, 2 supplements.

OPTIMIZATION ALGORITHMS, CONTENT OF THE COURSE, CONTENT ANALYSIS, GRAPH THEORY, OPTIMIZATION.

The object of the study is the algorithm for optimizing the content of the discipline.

The aim of the study is the implementation of the algorithm for optimizing the content of the discipline.

The methods of research are descriptive, comparative, algorithmization and programming.

The result of the diploma project is developed and implemented algorithm for optimizing the content of the discipline.

The need to optimize learning is becoming more pronounced every day, as the time spent on studying certain disciplines provided by the course decreases, and the amount of information that can be studied increases. There are many ways to optimize the learning process. This paper considers one of them, namely choosing an optimal sequence of study sections. The implementation of this choice takes is made in two steps. In the first step, the content of the discipline is analyzed using content analysis tools. In the second step, the adjacency matrix obtained from the results of content analysis is subjected to transformations, as a result of which an array is formed. This array contains the proposed optimal order of study sections in the discipline. Thus, focusing on the results of the software, we will be able to automate the analysis of the content of disciplines and improve their studying.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ	програмне забезпечення
КА	контент-аналіз

ВСТУП

Згідно з багатьма джерелами, час на вивчення дисциплін, передбачених курсом, зокрема у вищих навчальних закладах, має тенденцію до скорочення. В цей самий час, об’єм інформації збільшується. Таким чином, виникає необхідність у інтенсифікації навчання. Одним зі способів такої інтенсифікації є стиснення матеріалу, що викладається.

Варто зазначити, що досвідченість викладачів, які складають методики та навчальні плани, не віднімає необхідності у постійному удосконалені змісту навчальних дисциплін.

Даний дипломний проект спрямований на оцінку змісту навчальної дисципліни та розробку та програмну реалізацію алгоритму, результатом якого є запропонований оптимальний порядок тем, що вивчаються.

Мета роботи – реалізація алгоритму оптимізації змісту навчальної дисципліни.

Завдання:

а) ознайомитися з проблемою;

б) здійснити добір методів реалізації алгоритму оптимізації змісту навчальної дисципліни;

в) розробити алгоритм оптимізації змісту навчальної дисципліни;

г) програмно реалізувати розроблений алгоритм и провести серію тестових перетворень матриць суміжностей для визначення оптимального порядку розділів дисциплін.

Розділ 1 присвячений огляду основ теорії графів, їх застосування для обраної задачі. Визначається поняття оптимізації. Розглядається необхідність оптимізації навчальних матеріалів. Визначається два відповідних способи визначення зв’язків між розділами дисципліни, що базуються на контент-аналізі – за допомогою ручного та автоматичного кодування.

У розділі 2 детально описується розроблений алгоритм, для нього складається схема. Також детально розбирається програмна реалізація розробленого алгоритму.

Розділ 3 спрямований на експериментальну перевірку роботи розробленого ПЗ.

1 ОГЛЯД ПІДХОДІВ ДО ОПТИМІЗАЦІЇ ЗМІСТУ НАВЧАННЯ

В цьому розділі розглянемо основи теорії графів, застосуємо їх для обраної задачі. Визначимо, як саме встановлюються зв’язки між розділами дисципліни, а саме розглянемо два способи встановлення зв’язків. Буде проведений огляд можливих засобів визначення ефективності оптимізації.

1.1 Оптимізація навчального матеріалу

Перш за все, розберемося, що собою являє таке поняття, як оптимізація.

Оптимізація [8] – Вибір найкращого (оптимального) варіанту з множини можливих; покращення якогось процесу з метою досягнення його максимальної ефективності; підвищення інтенсивності чого-небудь з метою досягнення високих результатів.

Наступним кроком необхідно з’ясувати, чому навчальні матеріали потребують оптимізації. Як зазначає [12], час на вивчення дисциплін, передбачених курсом, зокрема у вищих навчальних закладах, має тенденцію до скорочення. В цей самий час, об’єм інформації збільшується. Таким чином, виникає необхідність у інтенсифікації навчання. Одним зі способів такої інтенсифікації є стиснення матеріалу, що викладається.

У [5] говориться про те, що методи та засобі планування, а також керування організаційними ресурсами визначають ефективне функціонування навчальних систем. Хоча на сьогоднішній день підготовка навчальних планів покращена за допомогою введення нормативних документів, однак, їх при складанні тільки на основі цих документів досі залишаються складнощі. У своїй роботі [5] говорить про те, що складнощі зумовлені обмеженістю навчальними системами, які реалізують традиційні

форми отримання вищої освіти, оскільки дані стандарти не можуть застосовуватися в навчальних системах з термінами навчання, що відрізняються від нормативних. Прикладами виступають отримання другої вищої освіти, навчання з використанням дистанційних технологій тощо.

Автори [13] погоджуються з ідеєю про те, що навчальний процес вже досить тривалий час потребує оптимізації, зокрема ця необхідність обумовлена проблемою проектування технологій синтезу знань. Оптимізація навчального процесу є ключовим елементом переходу від інтеграції знань до їх синтезу і охоплює цілі навчання, його зміст, форми, методи та аналіз результатів.

У джерелі [13] робиться висновок, що оптимізація дійсно є одним з головних та найбільш ефективних методів синтезу знань, а також стосується як фундаментального ядра будь-якої теорії, так і її методів, тобто, універсальних наукових та навчальних дій.

Про оптимізацію змісту навчального матеріалу також говорить [6]. Автор вважає, що впровадження дистанційних технологій навчання в усі форми неможливо без таких дій, як:

– аналіз інформаційно-професійної сутності завдань;

– моделювання міждисциплінарної взаємодії;

– рішення задачі визначення оптимальної послідовності вивчення окремих модулів курсу (розробки алгоритму);

– рішення задачі ефективності засвоєння навчального матеріалу.

Автор [6] наголошує на тому, що тенденція сьогоднішнього суспільства – перехід від дискретно-ступінчатих форм освіти до цілісної системи безперервної освіти, що буде забезпечувати потреби людини в різних освітніх траєкторіях.

Завдання вибору оптимальної реалізації навчального процесу зводиться до розподілу заданого набору структурних одиниць по заданих організаційно-тимчасових етапах.

На думку [6], передумовами реалізації оптимальної моделі навчання в рамках відповідного освітнього стандарту є:

– можливість подання змісту навчального курсу у вигляді безлічі окремих елементів;

– ранжування цих елементів;

– ранжування навчальних дисциплін відповідно до навчальних планів.

З представлених теоретичних джерел можемо зробити висновок, що оптимізація навчального процесу та змісту навчання – необхідний крок на шляху покращення якості навчальної системи.

1.2 Використання теорії графів у оптимізації

Як зазначає [12], для роботи з навчальною інформацією дуже добре підходять методи математичного моделювання, що охоплюють різні розділи математики: теорія графів, теорія ймовірностей, математична статистика та логіка тощо.

Авторами [12] запропоновані наступні методи відбору та організації навчальної інформації:

– укрупнення;

– ущільнення;

– вибір оптимальної послідовності відрізків навчального матеріалу;

– виділення основного (головного) в навчальному матеріалі.

Реалізація кожного з цих методів потребує попереднього аналізу матеріалу. Для цього доцільно використовувати теорію графів.

Теорію графів можна вважати простим, доступним та потужним засобом для вирішення питань, що пов’язані з пошуком оптимального рішення.

Узагальнивши відомості з [4][7][14], сформулюємо основні положення теорії графів, необхідні для вирішення задачі даного дипломного проекту.

Граф G – це сукупність двох кінцевих множин: множини точок, які називаються вершинами, і множини ребер. Число вершин графа не може бути рівним нулю. Кожне ребро можна роздивлятися як пару точок. Граф G, що містить хоча б одну дугу, називається орієнтованим графом. Приклад орієнтованого графу зображений на рисунку 1.

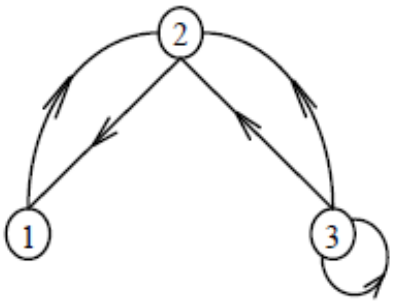


Рисунок 1.1 – Орієнтований граф G1, що містить 3 вершини та 5 ребер

Дві вершини, пов’язані ребром, називаються суміжними. З цього випливає, що можна скласти матрицю суміжності.

Матриця суміжності графу G – квадратна матриця [A] порядку n, кожен елемент aij якої дорівнює кількості ребер i-j графу (1.1):

$$[A] = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 2 & 1 \end{pmatrix}.$$

Використаємо ці визначення, щоб сформулювати задачу вибору оптимальної послідовності розділів навчальної дисципліни.

Припустимо, що ми розглядаємо деяку дисципліну, що містить 5 розділів, тобто x = {r1,r2,r3,r4,r5}. Нехай граф має такий вигляд (рисунок 1.2):

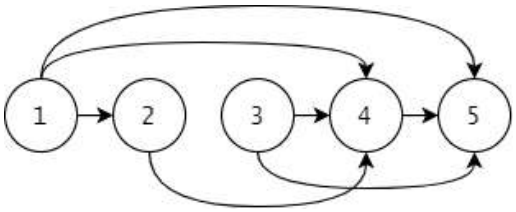


Рисунок 1.2 – Граф по розділам дисциплін

Представимо граф у вигляді матриці суміжності (таблиця 1.1):

Таблиця 1.1 – Матриця суміжності для графу з рисунку 1.2

	r1	r2	r3	r4	r5
r1	0	1	0	1	1
r2	0	0	0	1	0
r3	0	0	0	1	1
r4	0	0	0	0	1
r5	0	0	0	0	0

Якщо між розділами є зв’язок на графі, тоді в матриці цей зв’язок позначається як 1 (важливо враховувати напрямок зображеної на графі стрілки, оскільки у зворотному напрямку цей зв’язок буде помічений як 0). Можемо помітити, що перший та третій стовпчики містять в собі тільки нулі. Така ситуація показує, що розділи r1 та r3 можуть бути розміщені один за одним в довільному порядку (інакше кажучи, першим розділом може бути як r1, так і r3). Отримаємо наступну матрицю, викресливши з попередньої розділи r1 та r3. Отримана матриця представлена у таблиці 1.2.

Таблиця 1.2 – Нова матриця суміжності, отримана з попередньої

	r2	r4	r5
r2	0	1	0
r4	0	0	1
r5	0	0	0

На цій ітерації нульовий стовпець відповідає тільки одному розділу r2. Це означає, що розділ r2 не спирається на розділи, що йдуть за ним. Тобто, він має займати саме це місце в порядку вивчання розділів.

Підсумуємо: якщо виникає ситуація, коли ми маємо одночасно більше одного нульового стовпця, тоді й викреслювати потрібно всі стовпці та строки, що відповідають цим розділам, як у таблиці 1.1. Якщо ж за один прохід викреслюється тільки один розділ, це означає, що він має займати саме таке місце у послідовності розділів.

Процес перетворення матриці виконується, поки матриця не буде містити в собі 1 елемент (таблиця 1.3). На виході отримуємо усі варіанти оптимальної послідовності розділів.

Таблиця 1.3 – Кінцева матриця суміжності

	r5
r5	0

Таким чином, отримуємо, що для представленої дисципліни можливі послідовності наступні: {r1/r3, r2, r4, r5}. Вираз «r1/r3» аналогічний виразу «r1 або r3». Іншими словами, маємо дві можливі послідовності вивчання розділів: {r1, r3, r2, r4, r5} та {r3, r1, r2, r4, r5}.

1.3 Встановлення зв’язків між розділами дисципліни

Тепер, коли зрозуміло, як інтерпретувати граф та опрацьовувати його, виникає наступне питання: як встановити зв’язки між розділами, щоб мати можливість на їх основі побудувати орієнтований граф та матрицю суміжності.

Основою педагогічної діяльності є педагогічне проектування [3], що представляє собою сукупність видів професійної діяльності у сфері навчання,

що займається, в тому числі, формуванням змісту навчання. Історично склалося два способи побудови навчальних програм: концентричний та лінійний. При першому способі матеріал на наступному рівні вивчається в більш ускладненому вигляді, ніж на попередньому рівні (наприклад, в математиці). Лінійний спосіб вивчення матеріалу передбачає, що матеріал на наступному рівні є логічним продовженням попереднього рівня.

Встановлення зв’язків за допомогою онтологій використовується при вирішенні різноманітних задач. Вони надають можливість накопичувати та повторно використовувати знання, створювати моделі. Дуже часто для вилучення знань в різних системах використовуються статистичні методи [2], що базуються на частоті появи слів у тексті на природній мові.

Перехід від довільності в побудові та реалізації педагогічного процесу до обґрунтованості кожного його елементу та етапу, а також спрямованості на кінцевий результат, що об’єктивно діагностується, досягається за допомогою впровадження в процес педагогічної технології [3].

Нехай об’єкти, явища та методи діяльності, що були відібрані з науки та внесені до програми навчального предмета для вивчання, позначаються терміном «навчальні елементи» [3]. З таких елементів складається будь-яка навчальна програма, та предмети відрізняються між собою складом та кількістю навчальних елементів, що в них містяться. Тоді, за допомогою аналізу складу навчального предмету, оперуючи навчальними елементами матимемо змогу встановити як міжпредметні зв’язки, так і зв’язки між темами в рамках одного предмету.

Якщо ж розглядати проблему встановлення зв’язків як процес роботи з текстовими даними, одним з методів можна обрати контент-аналіз [10] [11].

В процесі аналізу теоретичних джерел було прийняте рішення визначити зв’язки за допомогою контент-аналізу, оскільки він в достатній мірі підходить під пошук та встановлення необхідних закономірностей.

1.3.1 Огляд програм для контент-аналізу

Контент-аналіз – стандартний метод дослідження в області соціальних наук. Предметом аналізу виступає зміст текстових масивів та продуктів комунікативної кореспонденції.

Контент-аналіз поділяється на кількісний та якісний. Перший проводиться з метою інтерпретації змісту знайдених чисельних закономірностей. Якісний контент-аналіз використовується, наприклад, для того, щоб виявити, скільки друкованого простору виділено для якогось предмету в обраному джерелі.

Докладніше методи аналізу тексту розглядаються в [10], також джерело містить деякі програмні продукти для здійснення КА та інших методів аналізу.

Роздивлялися декілька програм для аналізу текстової інформації:

LEXIMANCER. Проводить КА великих об’ємів тексту; дозволяє поєднувати в масиві тексти різних жанрів та стилів, в тому числі діалектичні та інші нетрадиційні форми мови.

QDAMiner. Проводить якісний аналіз текстових даних; дозволяє працювати з великою кількістю документів, що містять як текст, так и чисельні дані; надає широкий спектр пошукових засобів для виявлення кореляцій закодованих елементів.

WordStat. Обробляє такі матеріали, як журнальні статті, літературні твори, інтерв’ю; створює категоріальний апарат та словник контент-аналізу; дозволяє працювати з більш важкими методами статистичного аналізу (кластеризацією та багатомірним шкалюванням).

JFreq. Створює матриці частот використання слів у масиві; дозволяє виключити з масиву символи, що не читаються, та знаки, що не входять в алфавітну базу даних.

PROTAN. Дозволяє проводити КА масивів тексту за допомогою словників; ідентифікує сюжетні лінії, виявляючи кореляцію між словами через факторний аналіз та багато іншого.

ЛЕКТА. Проводить багатомірний КА тестових масивів; допомагає скласти словник на основі частотності або попередньо створеної системи категорій; розбиває тексти на рівні за об'ємом фрагменти.

TEXTPACK. Кодує тексти на основі створених користувачем словників;

проводить порівняння двох документів за словниковим наповненням; знаходить схожі фрагменти всередині документів.

Qualrus. Є інструментом проведення якісного аналізу даних, що кодує елементи масиву для подальшої обробки; може бути використаний для проведення повного спектра якісних досліджень, в тому числі в інтерпретації методів, емпіричному аналізі, аналізі оповідань і творів інших жанрів.

AnnoTape. Програмне забезпечення для запису і аналізу аудіо, відео, графічних і текстових даних; призначене для якісних досліджень, маркетингу, журналістики засобів масової інформації, архівних служб; виробляє запис звукових файлів – інтерв'ю, бесід, радіопередач безпосередньо на жорсткий диск комп'ютера; дозволяє зберігати до ста годин звуку разом з текстовими даними в одній інтегрованій базі даних; виробляє аналіз даних, анотування та індексування оригінальних звукових і текстових файлів; ефективно розбиває на фрагменти масиви аудіо-даних.

MonoConc. Здійснює пошук одиниць текстового аналізу; визначає кореляції між ними в масиві.

Для даного дипломного проекту в якості програмного забезпечення для КА був обраний **QDA Miner**[1)] з тієї причини, що він легкодоступний, має безплатну версію з частиною функціоналу, а також має сайт, що містить багато різних відео уроків по роботі з програмою, що полегшує її освоєння.

[1)] QDAMiner Official website https://provalisresearch.com/products/qualitative-data-analysis-software

1.3.2 Якісний контент аналіз у QDA Miner

Роздивимося більш детально, що представляє собою кодування в QDA Miner. На вході маємо документ `Method1.doc`, що містить якусь інформацію. В цьому випадку – лабораторні роботи з обраної дисципліни. Інтерфейс програмного забезпечення показаний на рисунку 1.3:

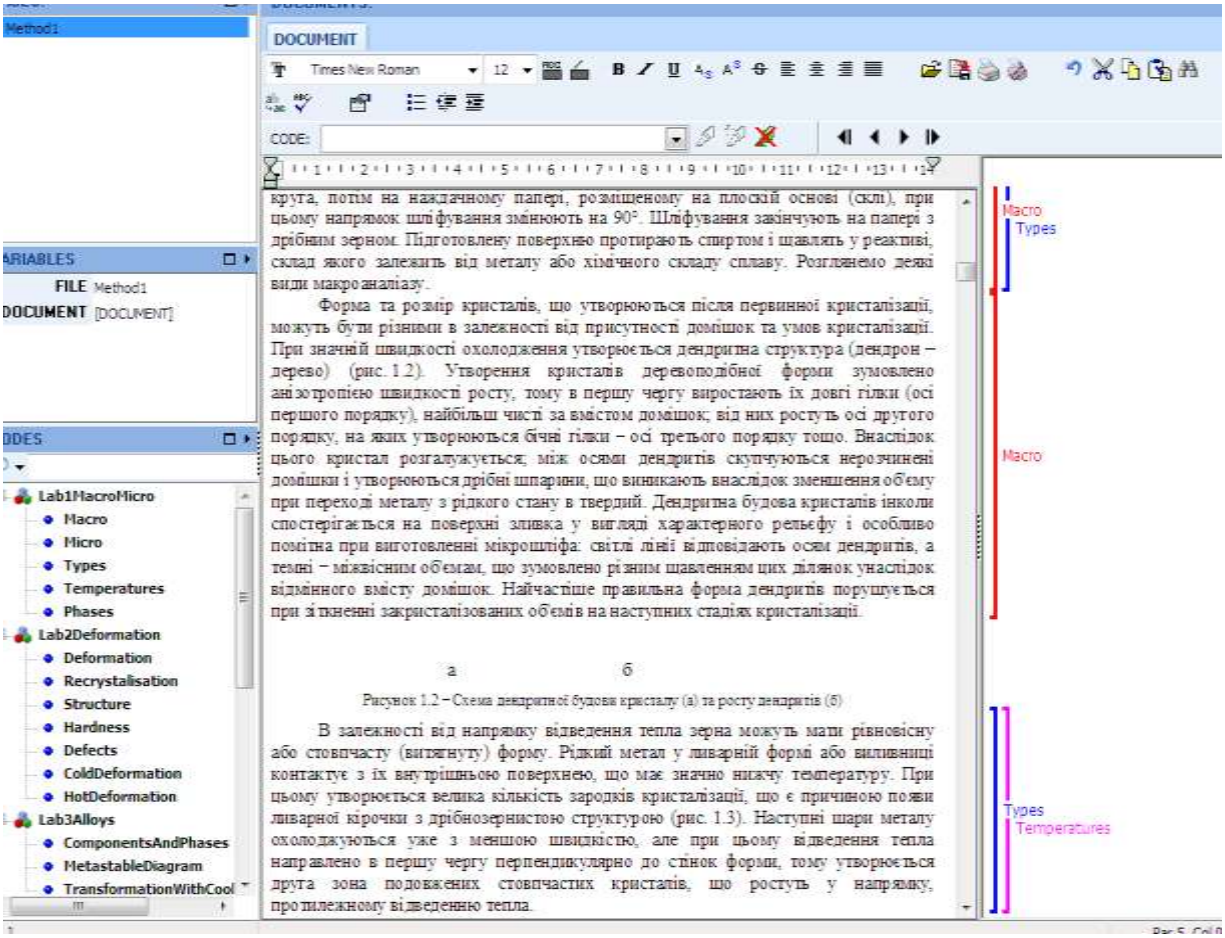


Рисунок 1.3 – Інтерфейс ПЗ QDA Miner: 1 – кодувальна книга в ієрархічному порядку; 2 – характеристика змінної (кількісна, категоріальна та ін.); 3 – список документів

Для створення кодувальної книги необхідно навести мишку на вікно CODES, натиснути праву кнопку на вільну область та вибрати опцію “Add code...”. Коди можна змінювати, видаляти, розділяти та ін. На рисунку 1.4 зображено вікно визначення коду.

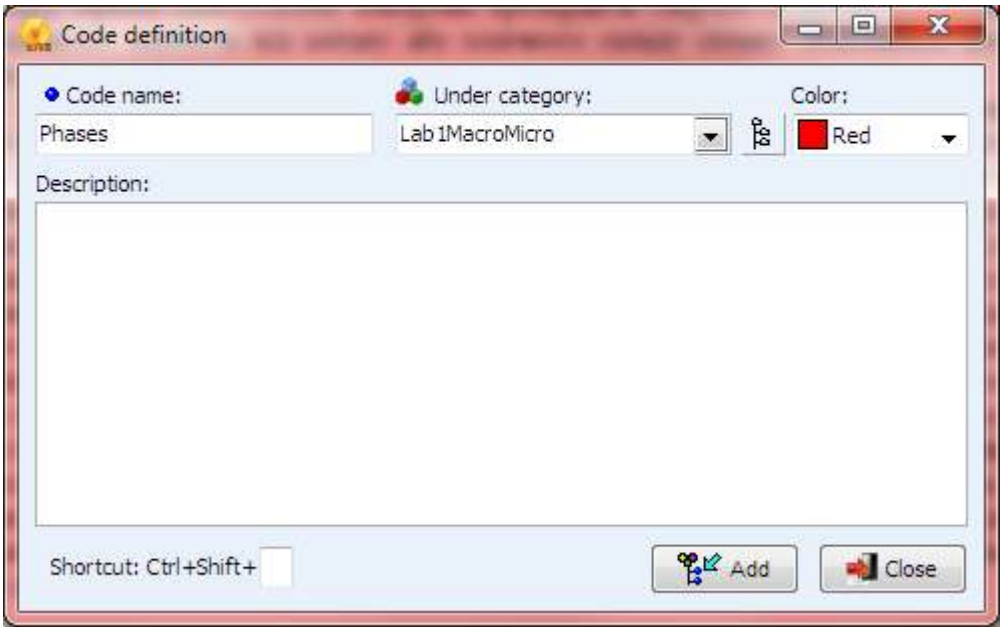


Рисунок 1.4 – Визначення нового коду

В даному дипломному проекті кодувальна книга формувалася за таким принципом: категорії – це безпосередньо теми, коди в них – окремі аспекти цієї теми (наприклад, підтеми, якщо вони наявні). Фрагмент ієрархії для обраного документа зображений на рисунку 1.5.

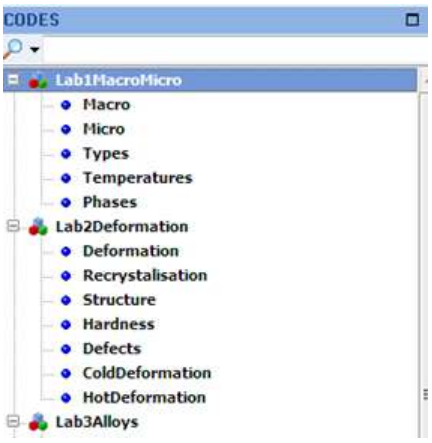


Рисунок 1.5 – Фрагмент ієрархії кодів для файлу Method1.doc

Після створення ієрархії кодів ми маємо можливість закодувати документ вручну. Для цього потрібно:

а) виділити бажаний сегмент в текстовому вікні;

б) двічі натиснути на потрібний код в кодувальній книзі (також можна перетащити з вікна кодувальної книги потрібний код в текстовий сегмент);

в) справа від тексту з’явиться примітка про те, до якого коду належить виділений сегмент (рисунок 1.6).

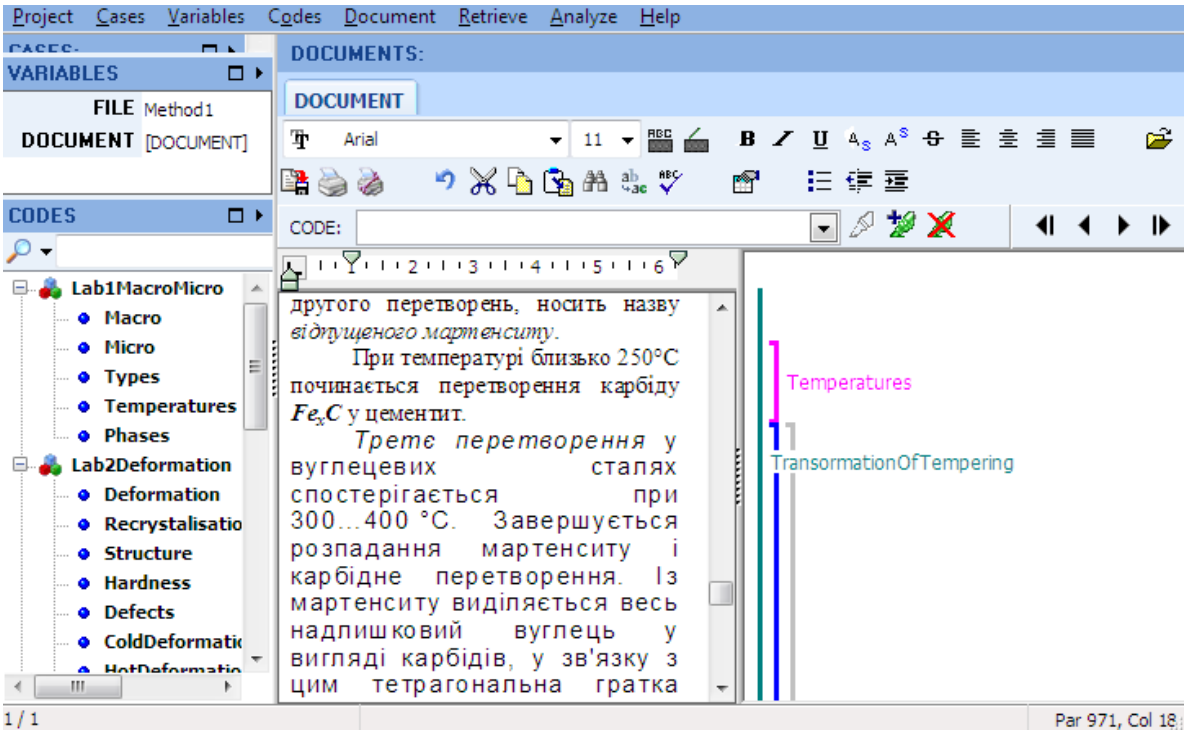


Рисунок 1.6 – Приклад закодованих сегментів тексту

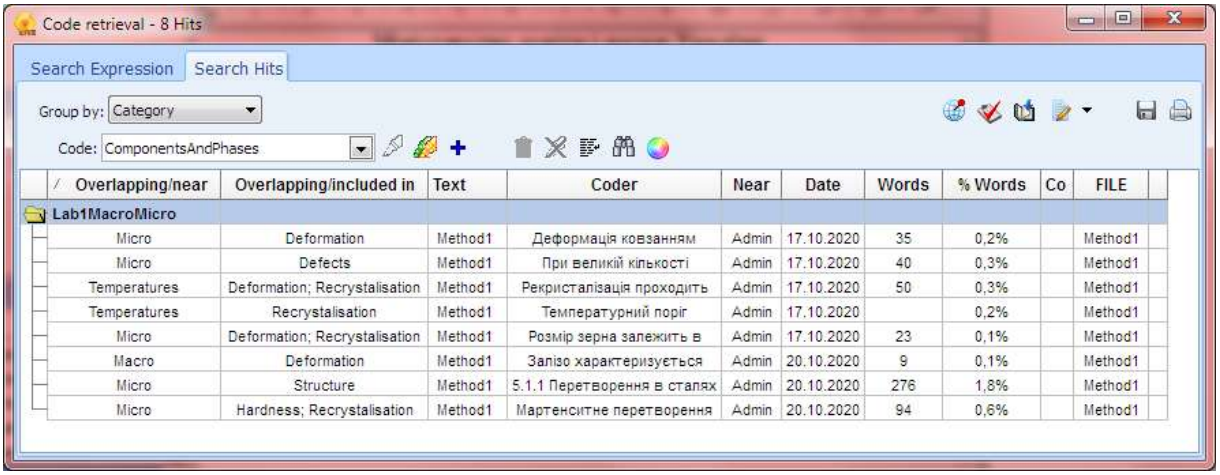
На рисунку 1.6 також маємо змогу чітко побачити, що коди можуть “переплітатися” між собою.

Варто зауважити, що QDA Miner пропонує інструменти, які спрощують процес кодування, наприклад, Retrieve -> Text Retrieval. Дана функція дозволяє подивитися всі документи на наявність ключових слів або фраз (рисунок 1.7).

Повна версія програми QDA Miner надає можливість автоматизувати та, тим самим, значно прискорити кодування сегментів за допомогою додатку **WordStat**[1)]. Зазначений додаток вміє автоматично категоризувати

[1)] https://provalisresearch.com/products/content-analysis-software/

сегменти, використовуючи словник-тезаурус. Це, звісно, означає, що для автоматичної категоризації дисципліни такий словник повинен існувати. На жаль, на сьогоднішній день його ще не склали. Проте, WordStat дозволяє створити його самостійно з допомогою експертів із обраної області.



Overlapping/near	Overlapping/included in	Text	Coder	Near	Date	Words	% Words	Co	FILE
Lab1MacroMicro									
Micro	Deformation	Method1	Деформація ковзанням	Admin	17.10.2020	35	0,2%		Method1
Micro	Defects	Method1	При великій кількості	Admin	17.10.2020	40	0,3%		Method1
Temperatures	Deformation; Recrystalisation	Method1	Рекристалізація проходить	Admin	17.10.2020	50	0,3%		Method1
Temperatures	Recrystalisation	Method1	Температурний поріг	Admin	17.10.2020		0,2%		Method1
Micro	Deformation; Recrystalisation	Method1	Розмір зерна залежить в	Admin	17.10.2020	23	0,1%		Method1
Macro	Deformation	Method1	Залізо характеризується	Admin	20.10.2020	9	0,1%		Method1
Micro	Structure	Method1	5.1.1 Перетворення в сталях	Admin	20.10.2020	276	1,8%		Method1
Micro	Hardness; Recrystalisation	Method1	Мартенситне перетворення	Admin	20.10.2020	94	0,6%		Method1

Рисунок 1.7 – Вікно функції Text Retrieval

Роздивимося роботу додатку WordStat на прикладі[1)], що пропонує офіційний сайт ПЗ. Послідовність дій наступна:

– отримуємо доступ до додатку WordStat через пункт меню Analyze -> Content Analysis (рисунок 1.8).

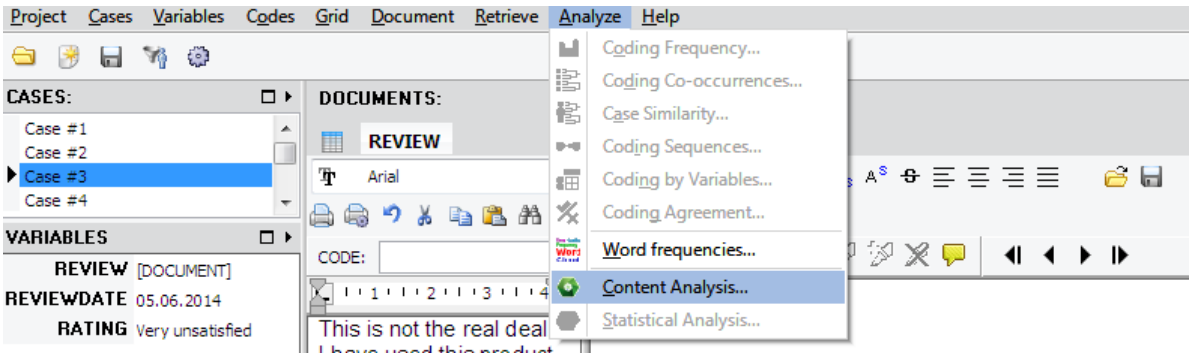


Рисунок 1.8 – Доступ до додатку WordStat

[1)] https://provalisresearch.com/resources/tutorials/automatically-code-segments-qdaminer/

– обираємо, які змінні аналізувати та по відношенню до чого. Можливо аналізувати по відношенню до інших змінних, зазначених кодів або категорій, або ж проводити тільки описовий аналіз (рисунок 1.9);

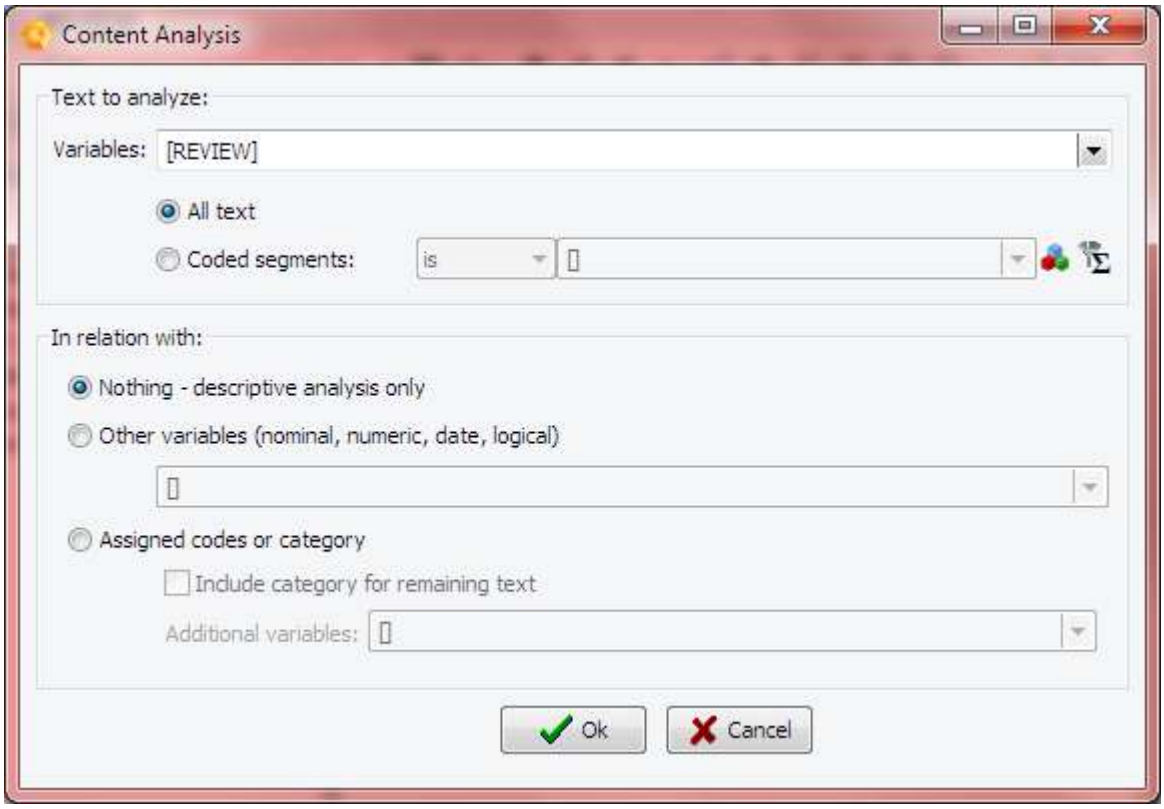


Рисунок 1.9 – Вибір параметрів аналізу

– відкривається WordStat. Перша вкладка, яку ми бачимо – Text Processing (Обробка тексту). Імпортуємо необхідний словник через кнопку Import. Для прикладу візьмемо словник RID, що наявний в бібліотеці прикладів. Після імпорту бачимо, що пакет містить 3 категорії – PRIMARY, SECONDARY та EMOTIONS. Оскільки ми обрали перелік відгуків на косметичні продукти, буде логічно обрати для аналізу категорію EMOTIONS (рисунок 1.10);

– переходимо на вкладку Frequencies, на якій ми можемо побачити частоту появи тих чи інших слів, що підпадають під підкатегорії. Натискаємо на кнопку Autocoding. Діалогове вікно дозволяє обрати вид кодування – за параграфами або за реченнями (рисунок 1.11).

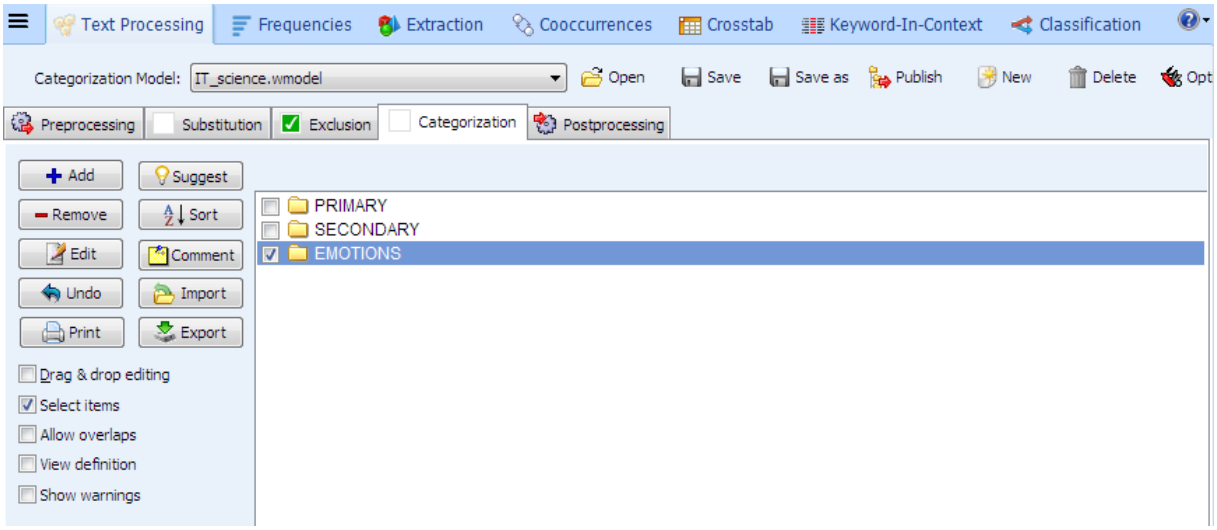


Рисунок 1.10 – Імпортований словник RID

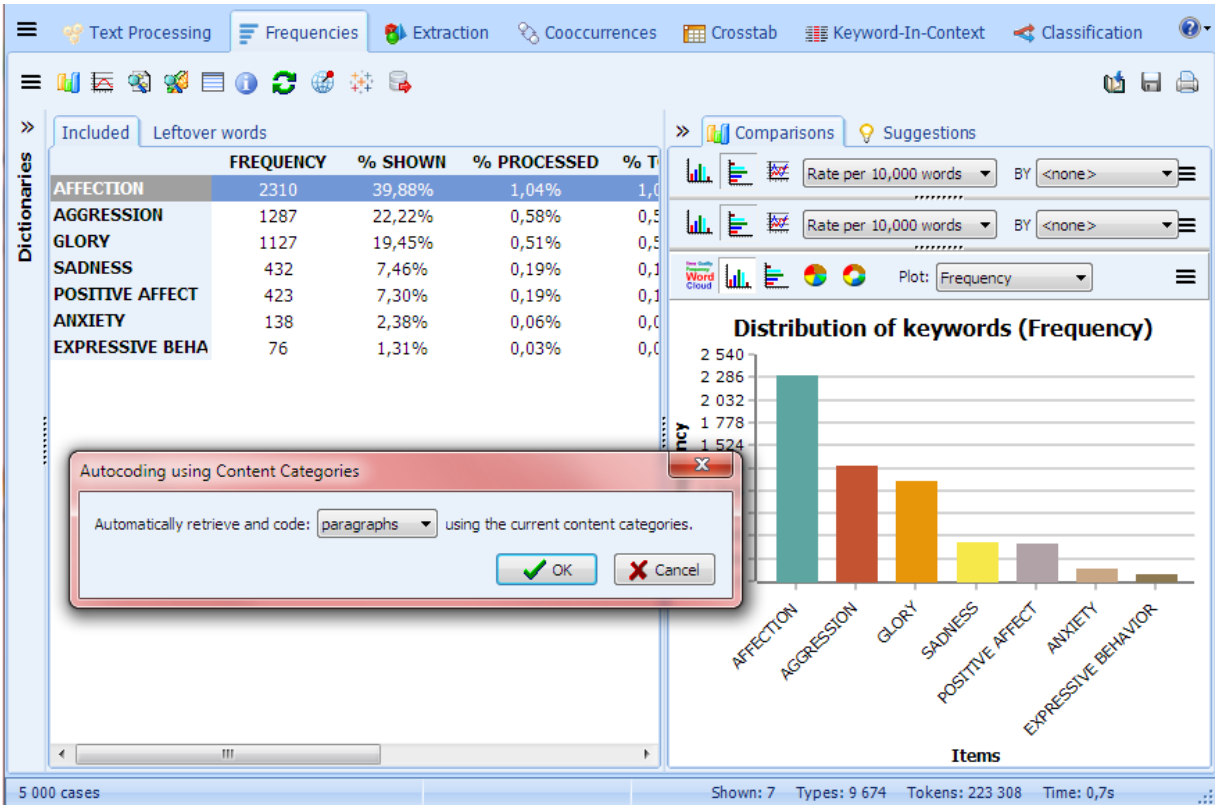


Рисунок 1.11 – Вкладка Frequencies. Автоматичне кодування

Нове діалогове вікно сповістить про те, скільки нових кодів було присвоєно. Після виходу з WordStat у QDA Miner з’являться всі зазначені коди, а також ієрархія цих кодів (рисунок 1.12).

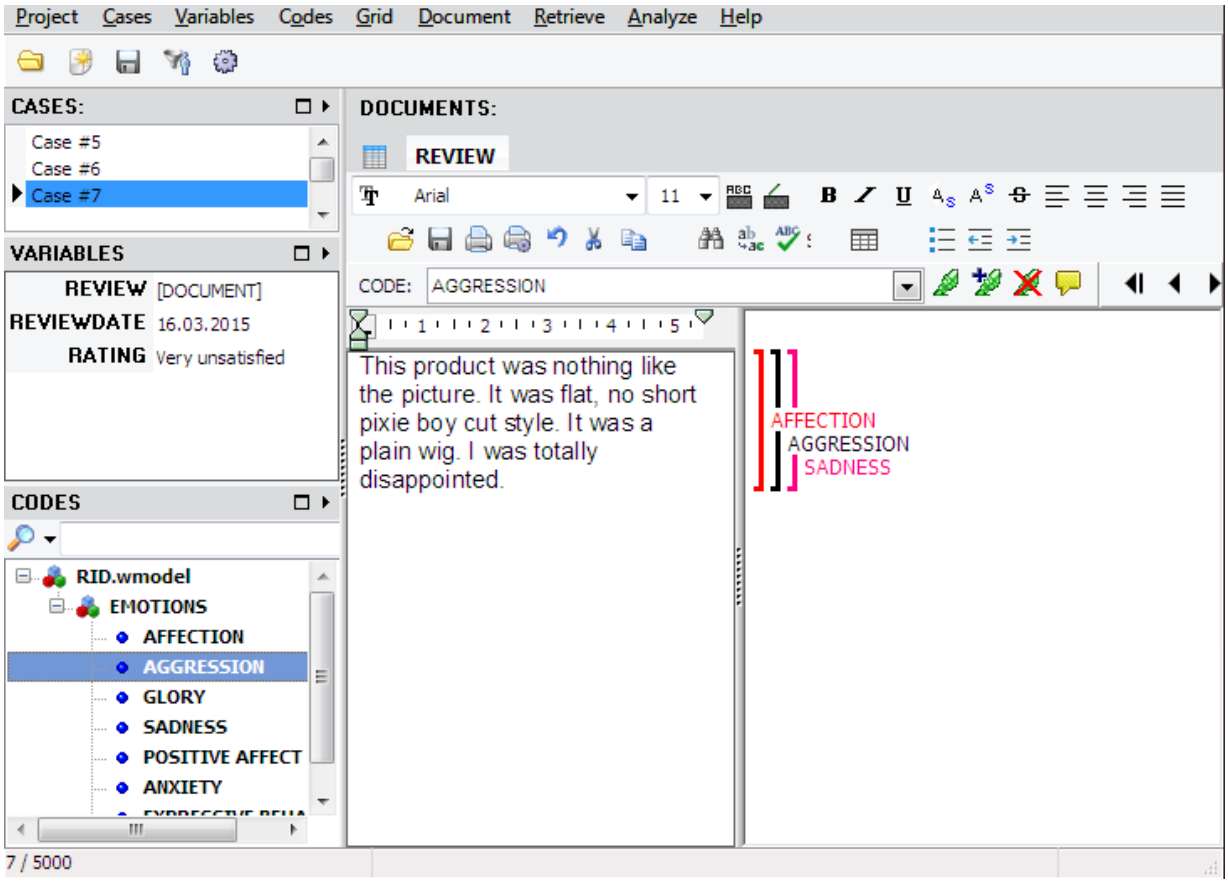


Рисунок 1.12 – Нові створені коди та ієрархія кодів

Коли кодування завершене, QDA Miner може вилучити ті сегменти документу, які пов’язані з конкретними кодами. Для цього в дипломній роботі була використана функція `Retrieve -> Coding Retrieval`. Саме таким чином ми зможемо встановити зв’язок між розділами дисципліни.

Наприклад, нам необхідно визначити, чи пов’язана Тема 1 обраного методичного посібника з Темою 2. Для цього визначаємо, чи є серед Теми 2 коди, що відповідають Темі 1 (тобто, чи є пересічення між кодами Теми 1 та Теми 2, або чи знаходяться вони поруч). Використана функція зображена на рисунку 1.13.

Результат пошуку показує, що між темами дійсно є зв’язок (рисунок 1.14).

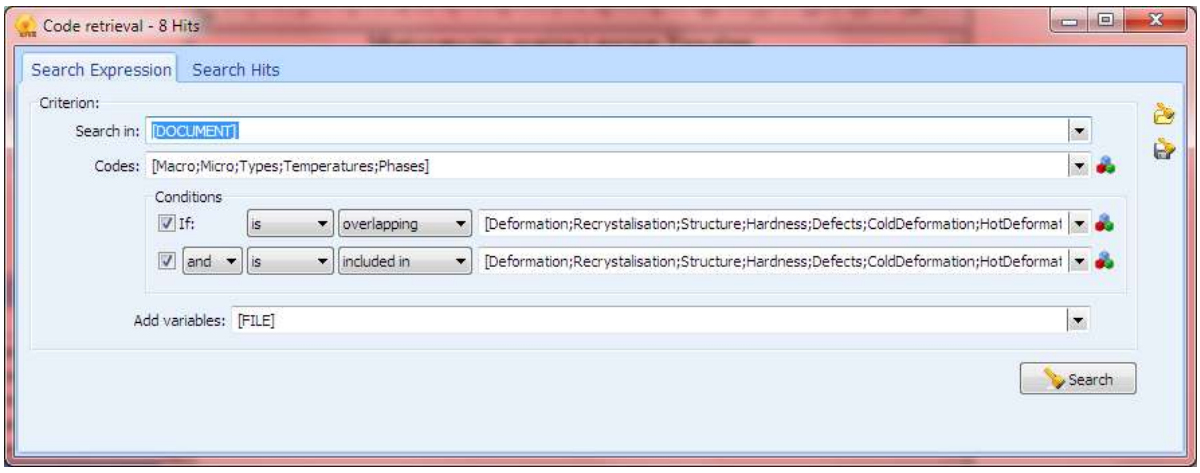
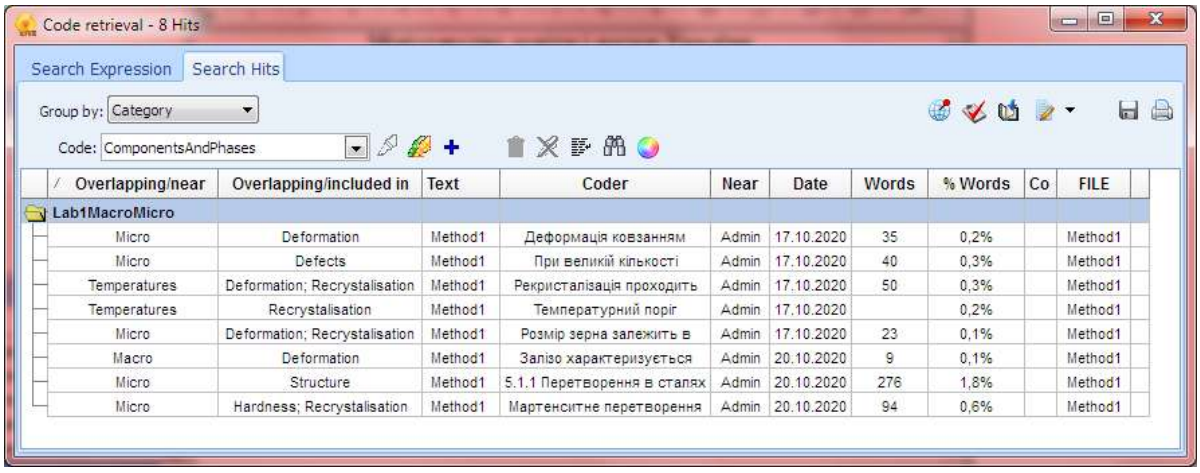


Рисунок 1.13 – Вікно функції Code Retrieval



Overlapping/near	Overlapping/included in	Text	Coder	Near	Date	Words	% Words	Co	FILE
Lab1MacroMicro									
Micro	Deformation	Method1	Деформація ковзанням	Admin	17.10.2020	35	0,2%		Method1
Micro	Defects	Method1	При великій кількості	Admin	17.10.2020	40	0,3%		Method1
Temperatures	Deformation; Recrystalisation	Method1	Рекристалізація проходить	Admin	17.10.2020	50	0,3%		Method1
Temperatures	Recrystalisation	Method1	Температурний поріг	Admin	17.10.2020		0,2%		Method1
Micro	Deformation; Recrystalisation	Method1	Розмір зерна залежить в	Admin	17.10.2020	23	0,1%		Method1
Macro	Deformation	Method1	Залізо характеризується	Admin	20.10.2020	9	0,1%		Method1
Micro	Structure	Method1	5.1.1 Перетворення в сталях	Admin	20.10.2020	276	1,8%		Method1
Micro	Hardness; Recrystalisation	Method1	Мартенситне перетворення	Admin	20.10.2020	94	0,6%		Method1

Рисунок 1.14 – Результат виконання пошуку за допомогою Code Retrieval

Таким чином встановлюється зв’язок між усіма наявними темами, після чого за результатами аналізу будується орієнтований граф.

Після завершення кодування для визначення зв’язків між розділами будемо використовувати `Code Retrieval` за таким принципом: якщо пошук дає більше одного-трьох перетинів, тоді робимо припущення, що між розділами дійсно існує стійкий зв’язок, якщо ж перетинів недостатня кількість, робиться висновок, що зв’язку між розділами немає.

1.3.3 Переваги та недоліки контент-аналізу

Серед переваг можна відмітити відносну простоту реалізації та достатньо якісний вихідний результат.

Оскільки для обраних дисциплін відсутній готовий тезаурус, кодинг можливо зробити тільки вручну, що може бути досить затратним за часом. Хоча ручний кодинг надає максимально достовірний результат, оскільки текст обробляється експертом з обраної дисципліни, обробляти таки чином велику кількість текстової інформації не ефективно. Саме в цьому випадку добре підходить автоматичний кодинг. Якість автоматичного кодингу залежить безпосередньо від повноти та всебічності словника-тезауруса.

1.4 Висновок

Розглянули поняття оптимізації та необхідність оптимізації навчальних матеріалів. Розглянули основи теорії графів, застосували їх для обраної задачі. Описали різні підходи до встановлення міжпредметних зв’язків і зв’язків між темами дисципліни. Визначили два відповідних способи визначення зв’язків між розділами дисципліни, що базуються на контент-аналізі – за допомогою ручного та автоматичного кодування.

2 ОПТИМІЗАЦІЯ ЗМІСТУ НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

2.1 Огляд алгоритму оптимізації змісту навчальної дисципліни. Формування гіпотези про ефективність алгоритму

У цьому розділі буде детально розглянуто алгоритм оптимізації змісту навчальної дисципліни, а також його програмну реалізацію. На основі теоретичних відомостей, отриманих у [12], був зроблений висновок, що спосіб виявлення оптимального порядку розділів у дисципліні релевантний та повинен мати програмну реалізацію для подальшого дослідження галузі оптимізації навчального матеріалу.

2.1.1 Опис розробленого алгоритму

З розділу 1 було визначено, що одним зі способів визначення оптимальної послідовності розділів у дисципліні є складання орієнтованого графу, завдяки якому матимемо можливість отримати матрицю суміжності та за допомогою перетворень визначити, в якому порядку мають вивчатися розділи дисципліни.

Базуючись на відомостях, описаних у пункті 1.1 та [12] було прийнято рішення створити алгоритм для програмної реалізації перетворення матриці суміжності з метою визначення оптимальної послідовності тем дисципліни.

Основні етапи алгоритму наступні:

а) на вході маємо документ `input.txt`, що містить у собі отриману з пункту 1.2 матрицю суміжності. На рисунку 2.1 представлена матриця за дисципліною `Aviation`.

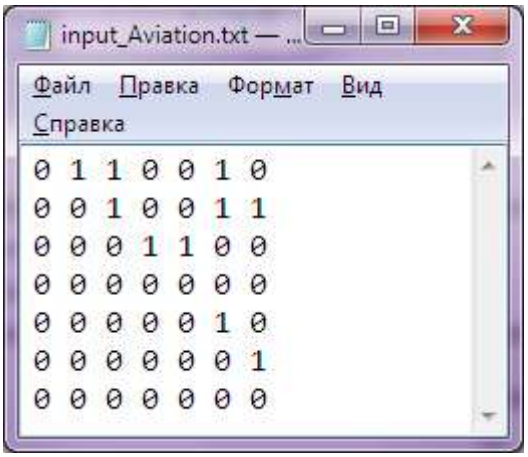


Рисунок 2.1 – Матриця суміжності для дисципліни Aviation

б) за допомогою функції `import2DArray` імпортуємо матрицю суміжності з файлу до основної програми;

в) в основній програмі вводимо кількість розділів (по факту – розмірність матриці);

г) на екран виводиться матриця, зчитана в пункті а);

д) програма знаходить нульові стовпці та прибирає їх разом із рядками за допомогою зміни розміру матриці на кількість знайдених стовпців та рядків з урахуванням їх положення;

е) новостворена матриця виводиться на екран.

ж) процес працює у циклі до моменту, поки матриця не буде складатися з одного елементу;

з) паралельно з пунктом д) в масив записуються номер прибраних рядку за стовпця.

Таким чином, в результаті роботи програми отримуємо масив з номерами прибраних рядків та стовпців.

Нижче представлено схему розробленого алгоритму (рисунок 2.2):

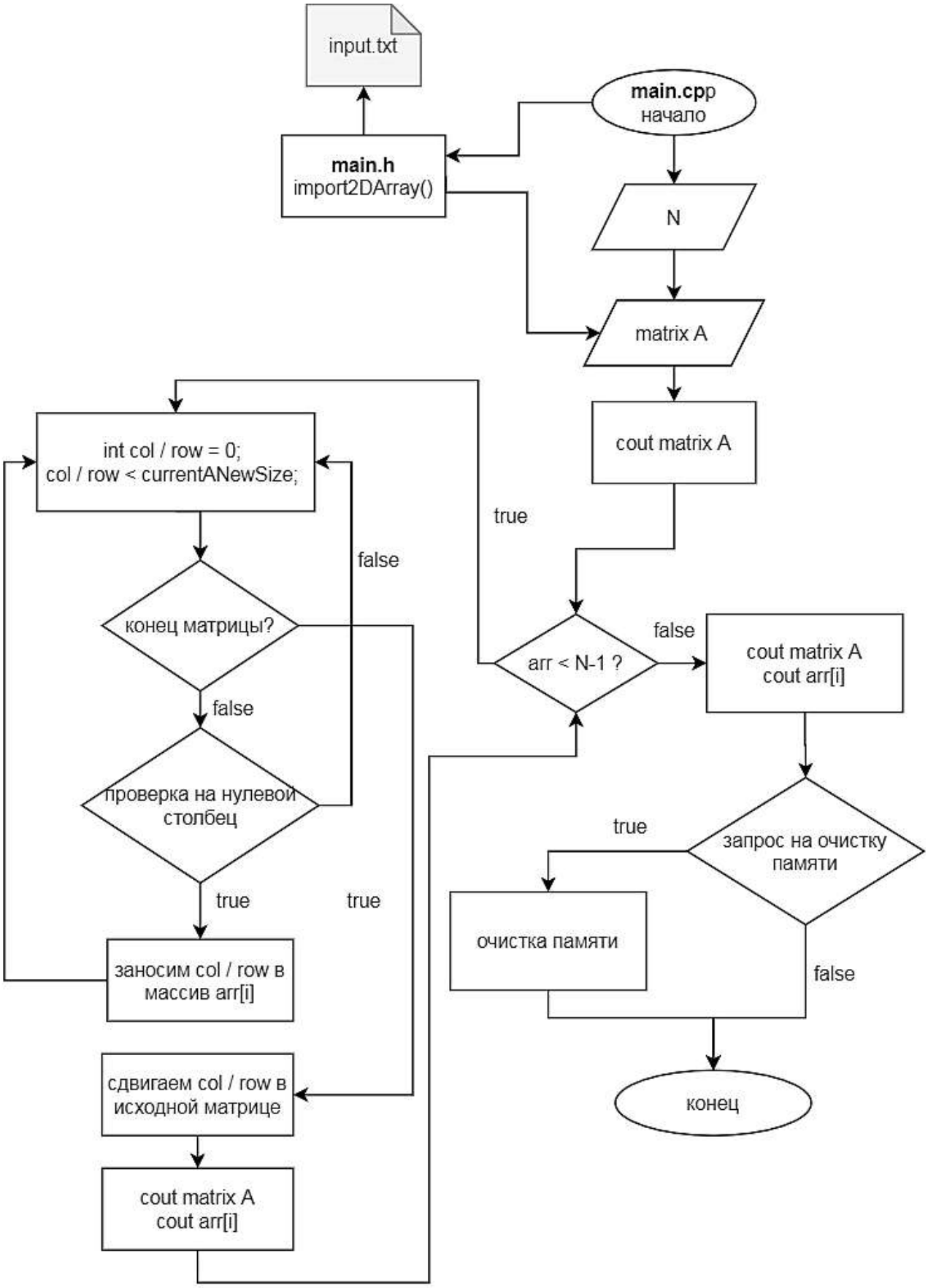


Рисунок 2.2 – Схема алгоритму, розробленого на основі теоретичних відомостей з розділу 1

2.2 Програмна реалізація алгоритму

Представлений вище алгоритм був реалізований за допомогою мови C++. Основний функціонал програми[3)] – перетворення матриці суміжності та створення масиву номерів прибраних рядків та стовпців. Розроблений код приведений у Додатку А.

Роздивимося детальніше безпосередньо перетворення матриці. Насамперед, отримана з файлу `input.txt` матриця за допомогою функції `import2DArray`, що міститься у хедері `main.h` імпортується до основної програми та виводиться на екран:

```
int** A = import2DArray(N);
cout << "\t\t\t Исходная матрица:\n";
cout << "_____________________________________\n\n";
for (int col=0; col<N; col++){
   for (int row=0; row<N; row++){
       cout<<A[col][row]<<" ";
   }
   cout<<endl;
}
cout << "_____________________________________\n";
```

Після цього в програмі визначаються змінні, необхідні для зменшення розміру матриці:

– змінна масиву, в який будуть заноситися номери прибраних рядків та стовпців (`arr[N]`);
– змінна, що визначає кількість заповнених позицій в масиві `arr[N]` (`arrI`);
– змінна, що містить поточний розмір матриці A (`currentANewSize`);

[3)] https://github.com/RawrMaster/MatrixReduction

– булева змінна для позначення знайдених нульових стовпців (`zeroCol`);
– змінна для підрахунку знайдених стовпців та строк на видалення (`deleteN`);
– змінна «наступного стовпця» (`nextCol`);
– змінна, що містить попереднє значення `arrI` (`arrIPrev`):

```
int arr[N] = { 0 };
int currentANewSize=N;
bool zeroCol;
int arrI = 0;
int deleteN;
int nextCol;
int arrIPrev = 0;
```

Цикл працює, поки позицій в масиві `arr` зайнято менше ніж N-1:

```
while (arrI<N-1) {
...
}
```

Роздивимося детальніше вміст даного циклу. Перший цикл `for`, що знаходиться у циклі `while`, проходиться по стовпцям та присвоює булевій змінній значення true.

```
for (int col = 0; col < currentANewSize; ++col) {
      zeroCol = true;
```

Вкладений до нього підцикл проходиться вже по рядкам. У випадку, коли елемент, що зустрівся при проході, не дорівнює нулю, змінна `zeroCol` змінює своє значення на `false`:

```
for (int row = 0; row < currentANewSize; ++row) {
     if (A[row][col] != 0) {
          zeroCol = false;
          break;
     }
}
```

Якщо після повного проходу змінна `zeroCol` залишається зі значенням `true`, це означає, що программа визначила нульовий стовпець. Тому його номер `arrI` заноситься до масиву `arr[]:`

```
          if (zeroCol == true)  {
             arr[arrI] = col;
             ++arrI;
           }
}
```

Наступний цикл `for` відповідає за «видалення» непотрібних рядків та стовпців шляхом зміщення рядків та стовпців матриці A. Проходимося по масиву `arr`, починаючи з попередньої ітерації та закінчуючи поточною:

```
for (int i = arrIPrev; i < arrI; i++)       {
     …
}
```

Насамперед, підраховуємо, скільки стовпців ми «видалили», щоб мати можливість зсувати рядки та стовпці на це значення. Поки кінець масиву arr не досягнутий, рахуємо ті випадки, коли прибирається декілька стовпців та рядків:

```
deleteN++;
int curI = i;
```

```
if (i != N - 1)  {
    while (arr[i] == arr[i + 1] - 1)   {
              ++deleteN;
              ++i;
    }
}
```

Після цього рахуємо, до якого стовпця та рядку зсувати. Якщо за видаленими стовпцями є ще якийсь стовпець, тоді виконується перша умова, інакше змінній наступного стовпця присвоюється значення поточного розміру матриці A:

```
if (i != arrI - 1)    {
    nextCol = arr[i+1];
}
else
    nextCol = currentANewSize;
```

Основні три цикли `for` відповідають за зміщення рядків та стовпців за схемою: першу частину зміщуємо вліво, другу – вверх, останню – вліво по діагоналі:

```
for (int row = arr[i] + 1; row < nextCol; row++) {
    for (int col = 0; col < arr[curI]; col++) {
        A[row - deleteN][col] = A[row][col];
    }
}

for (int row = 0; row < arr[curI]; row++)  {
    for (int col = arr[i]+1; col < nextCol; col++) {
         A[row][col - deleteN] = A[row][col];
    }
}
```

```
for (int row = arr[i]+1; row < nextCol; row++)  {
    for (int col =arr[i]+1; col < nextCol; col++) {
         A[row - deleteN][col - deleteN] = A[row][col];
    }
}
```

Приклад зміщення частин матриці зображений на рисунку 2.3:

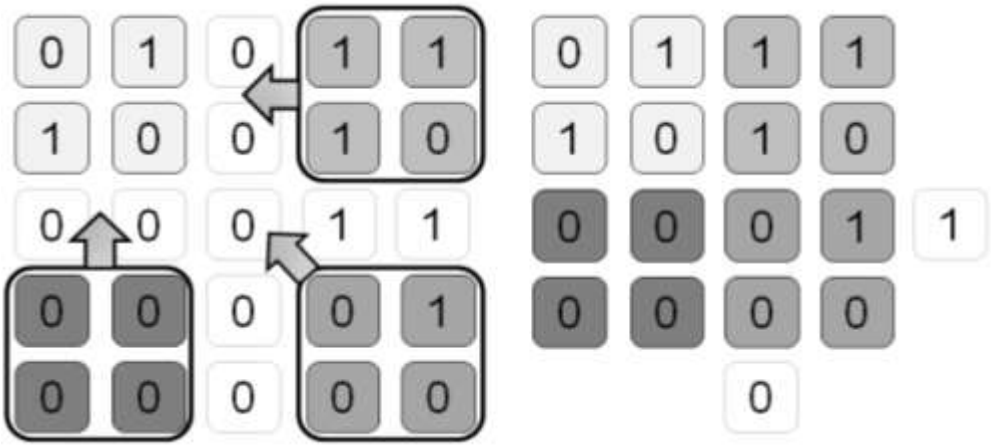


Рисунок 2.3 – Схема зміщення частин матриці

Останнє, що залишається зробити – змінити розмір матриці на значення, відповідне до прибраних рядків та стовпців:

```
currentANewSize -= (arrI - arrIPrev);
if (currentANewSize == 0)  {
   currentANewSize = 1;
   arrI--;
}
```

Після цього виводимо нову перетворену матрицю та масив з індексами прибраних стовпців та рядків:

```
for (int i = 0; i < arrI; ++i)    {
```

```
        cout << arr[i]+1 << " ";
    }
```

Варто зазначити що, оскільки програма збирає поточні індекси прибраних частин матриці, було прийнято рішення виводити вміст масиву на кожній ітерації, щоб, по-перше, можливо було визначити, скільки стовпців прибралося за ітерацію, та, по-друге, легше було інтерпретувати вміст масиву (перевести отримати індекси до відповідних номерів розділів дисципліни).

2.3 Висновок

У цьому розділі детально описали розроблений алгоритм, склали для нього схему. Також детально розібрали програмну реалізацію розробленого алгоритму.

3 ЕКСПЕРИМЕНТАЛЬНА ОЦІНКА РОЗРОБЛЕНОГО АЛГОРИТМУ

3.1 Аналіз коректності роботи алгоритму

Метою експерименту є перевірка гіпотези про ефективність розробленого алгоритму та коректності роботи алгоритму. Основою перевірки є оцінка результатів роботи ПЗ на основі відомостей у пункті 1.1.

Апаратне та програмне середовище виконання експерименту:

- процесор: Intel (R) Core (™) i5-4430 CPU 3.00 GHz;
- оперативна пам’ять: 16,00 Гб;
- тип системи: 64-розрядна ОС;

3.2 Перевірка гіпотези про ефективність.

Мірою ефективності роботи алгоритму є безпосередньо його здатність формувати послідовність розділів, що передбачається як оптимальна.

Для перевірки були обрані 3 довільних методичних посібників з різних дисциплін – «Інформатика», «Авіаційне матеріалознавство», «Інформаційні технології».

На основі якісного контент-аналізу, що описаний в розділі 1, для кожної дисципліни були складені орієнтовані графи та матриці суміжностей, що їм відповідають.

Орієнтовані графи були створені автоматично з матриць суміжностей за допомогою сервісу **Graph Online**[1)]. Для перевірки сервісу попередньо використовувалася матриця-приклад з розділу 1. Результати роботи для згаданої матриці представлені на рисунку 3.1.

[1)] Сайт Graph Online – https://graphonline.ru/create_graph_by_matrix

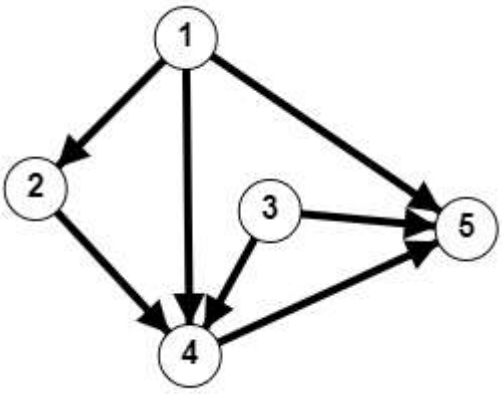


Рисунок 3.1 – Орієнтований граф для матриці-прикладу з таблиці 1.1

«Інформатика». Методичний посібник містить 16 тем, кожна з яких чітко поділена на підтеми. Поділення на підтеми важливе, оскільки значно спрощує кодування для контент-аналізу.

Орієнтований граф та матриця суміжності для дисципліни представлені нижче (див. рис. 3.2, табл. 3.1):

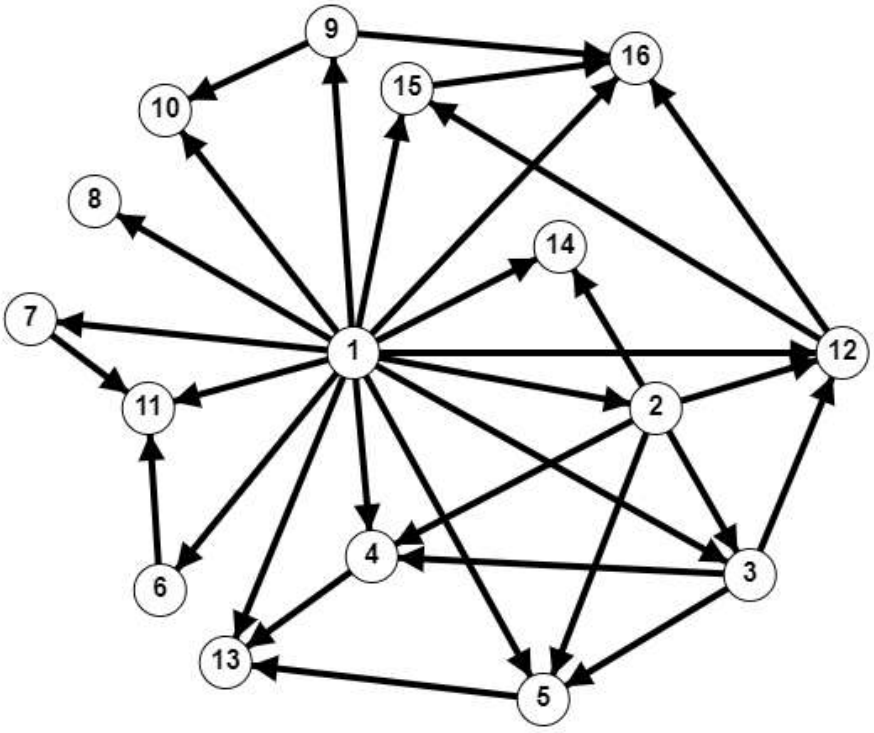


Рисунок 3.2 – Орієнтований граф для дисципліни «Інформатика»

Таблиця 3.1 – Матриця суміжності для дисципліни «Інформатика»

	r1	r2	r3	r4	r5	r6	r7	r8	r9	r10	r11	r12	r13	r14	r15	r16
r1	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
r2	0	0	1	1	1	0	0	0	0	0	0	1	0	1	0	0
r3	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0
r4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
r5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
r6	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
r7	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
r8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r9	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1
r10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
r13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
r16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

«Авіаційне матеріалознавство». Методичний посібник містить лабораторні роботи для відповідної дисципліни. В цьому посібнику відсутній поділ на підтеми, що сповільнює ручне кодування для контент-аналізу (див. рис.3.3, табл. 3.2).

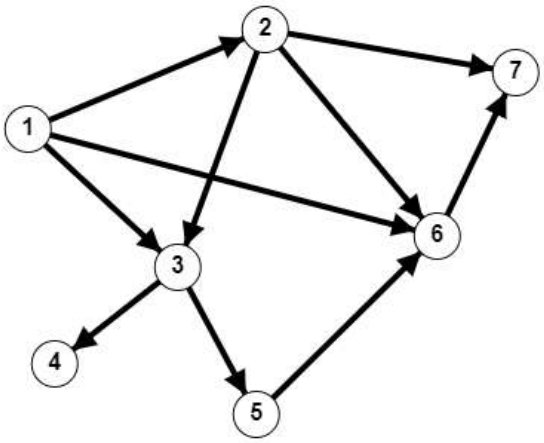


Рисунок 3.3 – Орієнтований граф для дисципліни «Авіаційне матеріалознавство»

Таблиця 3.2 – Матриця суміжності для дисципліни «Авіаційне матеріалознавство»

	r1	r2	r3	r4	r5	r6	r7
r1	0	1	1	0	0	1	0
r2	0	0	1	0	0	1	1
r3	0	0	0	1	1	0	0
r4	0	0	0	0	0	0	0
r5	0	0	0	0	0	1	0
r6	0	0	0	0	0	0	1
r7	0	0	0	0	0	0	0

«Інформаційні технології». Методичний посібник складається з 7 тем та поділений на підтеми аналогічно до посібнику для дисципліни «Інформатика» (див. рис. 3.4).

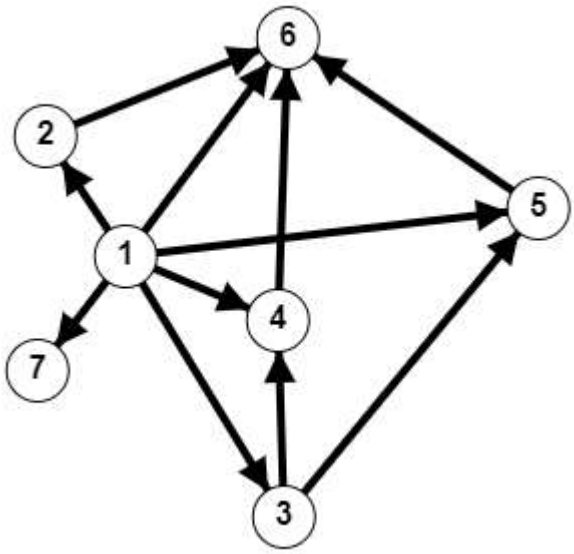


Рисунок 3.4 – Орієнтований граф для дисципліни «Інформаційні технології»

Матриця суміжностей для дисципліни представлена у таблиці 3.3.

Таблиця 3.3 – Матриця суміжностей для дисципліни «Інформаційні технології»

	r1	r2	r3	r4	r5	r6	r7
r1	0	1	1	1	1	1	1
r2	0	0	0	0	0	1	0
r3	0	0	0	1	1	0	0
r4	0	0	0	0	0	1	0
r5	0	0	0	0	0	1	0
r6	0	0	0	0	0	0	0
r7	0	0	0	0	0	0	0

3.2.1 Аналіз коректності роботи алгоритму

Як було зазначено в розділі 2, ПЗ видає результат через масив та показує саме поточні індекси прибраних частин матриці, тобто не номери розділів. Результаті роботи для кожної дисципліни наступні:

«Інформатика»: `1, ( 1 5 6 7 8 ), ( 1 4 5 8 ), (1 2 3 ), ( 1 2 ), 1`

«Авіаційне матеріалознавство»: `1, 2, 3, (4 5), 6, 7`

«Інформаційні технології»: `1, (2 3 7), (4 5), 6`

Повний результат роботи програми представлений у Додатку Б.

3.2.2 Визначення еталонної оцінки формування груп

Простим способом оцінки правильності рішення є порівняння його результатів з еталонною оцінкою. Для задачі перетворення матриць еталонною оцінку можна вважати, звісно, здійснення відповідних операцій вручну.

Нижче для прикладу представлені відповідні перетворення для дисципліни «Авіаційне матеріалознавство» (див. рис 3.5):

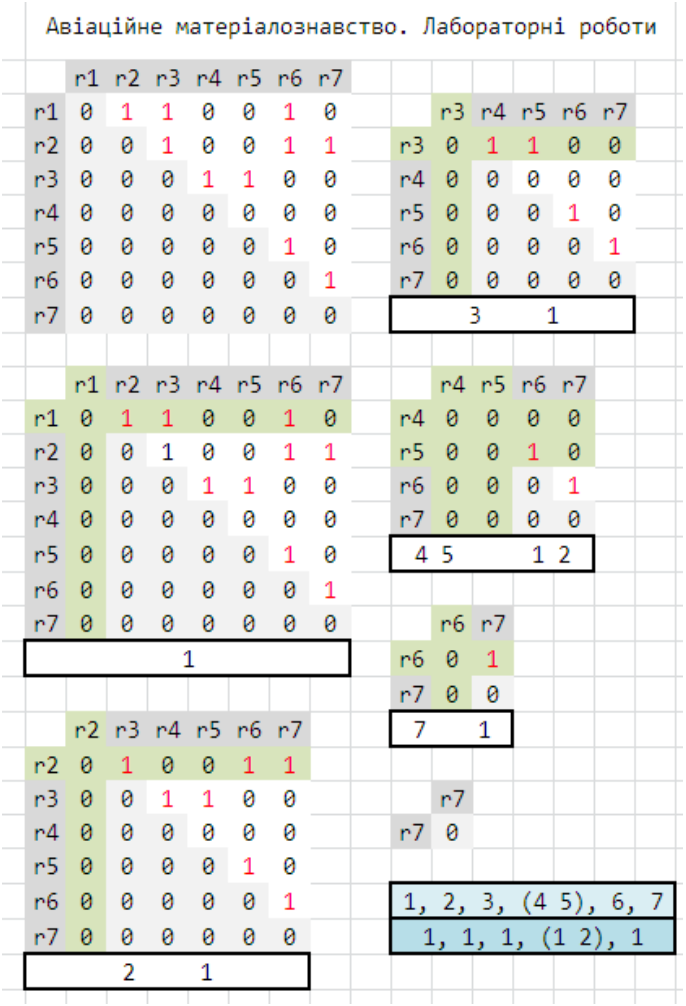


Рисунок 3.5 – еталонна оцінка для дисципліни «Авіаційні матеріали»

Як можна побачити з рисунку 3.5, результат роботи ПЗ відповідає результату при ручному перетворенні матриць.

3.4 Переваги та недоліки розробленого алгоритму

Перевагою розробленого алгоритму є його простота розуміння та реалізації, а також коректне виконання поставленої задачі.

Серед очевидних недоліків можна відзначити той факт, що програма збирає поточні індекси нульових рядків і стовпців без прив’язки до номера розділу.

3.5 Висновок

В цьому розділі експериментально перевірили коректність роботи розробленого ПЗ. Розглянули результати контент-аналізу для трьох дисциплін у вигляді орієнтований графів та матриць суміжностей. Визначили основні переваги та недоліки алгоритму.

ВИСНОВКИ

В розділі 1 розглянули поняття оптимізації та необхідність оптимізації навчальних матеріалів. Розглянули основи теорії графів, застосували їх для обраної задачі. Описали різні підходи до встановлення міжпредметних зв’язків і зв’язків між темами дисципліни. Визначили два відповідних способи визначення зв’язків між розділами дисципліни, що базуються на контент-аналізі – за допомогою ручного та автоматичного кодування.

В розділі 2 детально описали розроблений алгоритм, склали для нього схему. Також детально розібрали програмну реалізацію розробленого алгоритму.

В розділі 3 експериментально перевірили коректність роботи розробленого ПЗ. Розглянули результати контент-аналізу для трьох дисциплін у вигляді орієнтований графів та матриць суміжностей. Визначили основні переваги та недоліки алгоритму.

ПЕРЕЛІК ПОСИЛАНЬ

1. Kiv A., Soloviev V., Tarasova E., Koycheva T., Kolesnykova K. Semantic knowledge networks in education. *The International Conference on Sustainable Futures: Environmental, Technological, Social and Economic Matters (ICSF 2020). Kryvyi Rih, Ukraine,* May 20-22, 2020. Volume 166. Article 10022. URL: https://doi.org/10.1051/e3sconf/202016610022 (дата звернення: 16.10.2020)

2. Амурский, К.А. Проблема извлечения знаний в информационных системах. *Известия ПГПУ им. В.Г.Белинского*. 2010. №18 (22). С. 96–98.

3. Беспалько В. П. Слагаемые педагогической технологии. Москва : Педагогика, 1989. 191 с.

4. Домнин Л.Н. Элементы теории графов. Пенза: ПГУ, 2007. 144 с.

5. Кабальнов Ю. С., Кузьмина Е. А. Никин А. Д. Модели и оптимизация учебных планов в образовательных системах *Вестник УГАТУ*, 2002. Т. 4. № 1 (7). С. 181–189.

6. Калмыкова С.В., Соколицин А.С. Оптимизация содержания и структуры учебного процесса в вузе с использованием метамодели *Научно-технические ведомости Санкт-Петербургского государственного политехнического университета. Информатика, телекоммуникации и управление.* 2012. №6 (162). URL: https://cyberleninka.ru/article/n/optimizatsiya-soderzhaniya-i-struktury-uchebnogo-protsessa-v-vuze-s-ispolzovaniem-metamodeli (дата звернення 16.11.2020).

7. Калугин Н.А., Калугин А.Н. Элементы теории графов. Учебное пособие. Самара: Самар. гос. аэрокосм. ун-т, 2013. 48с.

8. Кузнецов С.А. Большой толковый словарь русского языка. изд. СПб.: Норинт, 1998. 1488 с.

9. Мальтекбасов М.Ж., Прокофьева М.А., Ескендиров Б.Н., Нурбосынова Г.С. Особенности применения теории графов при

проектировании образовательной траектории в вузе. *Международный журнал экспериментального образования.* 2014. № 1-1. С. 102-105; URL: http://www.expeducation.ru/ru/article/view?id=4526 (дата звернення: 12.10.2020).

10. Митина О.В., Евдокименко А.С. Методы анализа текста: методологические основания и программная реализация. *Психология. Психофизиология.* 2010. №40. URL: https://cyberleninka.ru/article/n/metody-analiza-teksta-metodologicheskie-osnovaniya-i-programmnaya-realizatsiya (дата звернення: 16.10.2020).

11. Олейник А.Н. Контент-анализ больших качественных данных. *International Journal of Open Information Technologies.* 2019. №10. URL: https://cyberleninka.ru/article/n/kontent-analiz-bolshih-kachestvennyh-dannyh (дата звернення: 20.11.2020).

12. Ощепкова Н. П., Поддубная М. Л. Моделирование учебной информации. *Математическое образование на Алтае*: Труды региональной научно-методической конференции МОНА 2001, Барнаул, 28 сентября 2001. URL : http://edu.secna.ru/media/f/Oshepkova.pdf. (дата звернення 12.10.2020).

13. Сечкин Г.И. Оптимизация как эффективный метод в педагогической технологии синтеза знаний *Омский научный вестник*. 2015. №1 (135). URL: https://cyberleninka.ru/article/n/optimizatsiya-kak-effektivnyy-metod-v-pedagogicheskoy-tehnologii-sinteza-znaniy (дата звернення: 12.10.2020).

14. Фрич Р., Перегуд Е.Е., Мациевский С.В. Избранные главы теории графов. Калининград: РГУ им. И. Канта, 2008. 205 с.

15. Шевченко А.И. Проектирование дисциплинарного образовательного пространства в вузе и методика его освоения. Ставрополь: СГУ, 2004. 176 с.

ДОДАТОК А

Програмна реалізація розробленого алгоритму

main.cpp

```
#include <cstdio>
#include <iostream>
#include "main.h"
#include <list>


using namespace std;

int main()
{
    setlocale(LC_ALL, "RUSSIAN");
    int N;

    cout << "К-во разделов> ";
    cin >> N;

    int** A = import2DArray(N);
    cout << "\t\t\t Исходная матрица:\n";
    cout << "_____________________________________\n\n";

    // Выводим матрицу x


    for (int col=0; col<N; col++)
    {
        for (int row=0; row<N; row++)
        {
            cout<<A[col][row]<<" ";
        }
        cout<<endl;
    }


    cout << "_____________________________________\n";
```

```
int arr[N] = { 0 };

int currentANewSize=N;

//int currentANewSizeM = 0;
bool zeroCol;
int arrI = 0;
int deleteN;
int nextCol;
int arrIPrev = 0;


while (arrI<N-1)
{
    for (int col = 0; col < currentANewSize; ++col)
    {

        zeroCol = true;
        for (int row = 0; row < currentANewSize; ++row)
        {
            if (A[row][col] != 0)
            {
                zeroCol = false;
                break;
            }
        }
        if (zeroCol == true)
        {
            arr[arrI] = col;
            ++arrI;

        }
    }

    if (arrI == arrIPrev)
        break;

    deleteN = 0;

    for (int i = arrIPrev; i < arrI; i++)
    {
        deleteN++;
        int curI = i;
        if (i != N - 1)
        {
```

```
                    while (arr[i] == arr[i + 1] - 1)
                    {
                        ++deleteN;
                        ++i;
                    }
                }
                if (i != arrI - 1)
                {
                    nextCol = arr[i+1];
                }
                else
                    nextCol = currentANewSize;
                for (int row = arr[i] + 1; row < nextCol; row++)
                {
                    for (int col = 0; col < arr[curI]; col++)
                    {
                        A[row - deleteN][col] = A[row][col];
                    }
                }
                for (int row = 0; row < arr[curI]; row++)
                {
                    for (int col = arr[i]+1; col < nextCol; col++)
                    {
                        A[row][col - deleteN] = A[row][col];
                    }
                }
                for (int row = arr[i]+1; row < nextCol; row++)
                {
                    for (int col =arr[i]+1; col < nextCol; col++)
                    {
                        A[row    -    deleteN][col    -    deleteN]    =
A[row][col];
                    }
                }
            }
            currentANewSize -= (arrI - arrIPrev);
            if (currentANewSize == 0)
            {
                currentANewSize = 1;
                arrI--;
            }


            cout << endl;
```

```
    for (int i = arrIPrev; i < arrI; ++i)
    {
        cout << arr[i]+1 << " ";
    }
    cout << endl;

    arrIPrev = arrI;


    cout << "_____________________________________________\n\n";

    // shows matrix

    for (int row = 0; row < currentANewSize; ++row)
    {
        for (int col = 0; col < currentANewSize; ++col)
        {
            cout << A[row][col] << " ";
        }
        cout << endl;
    }

    cout << "_____________________________________________\n";
}

cout << "\nПоследовательность удаления:\n" << endl;

for (int i = 0; i < arrI; ++i)
{
    cout << arr[i]+1 << " ";
}
cout << "\n_____________________________________________\n";


string ask;
cout << "\n Очистить память? (y/n)> ";
cin >> ask;
if (ask == "y")
{

    cout << "\n\nОчищаем память...";
    delete [] A;
    A = 0;
    printf("\n\nПамять очищена.\n\n\n");
}
```

```
    else if (ask=="n")
    {
        return 0;
    }

    return 0;

}
```

main.h

```
#include <cstdio>
#include <fstream>
#include <iostream>

using namespace std;

int** import2DArray(unsigned n)
{


    setlocale(LC_ALL, "RUSSIAN");

    ifstream in("input.txt");
        int** x = 0;
        x = new int*[n];
    if (in.is_open())
    {
        int count = 0;
        int temp;
        while (!in.eof())
        {
            in >> temp;
            count++;
        }

        in.seekg(0, ios::beg);
        in.clear();

        int count_space = 0;
        char symbol;
        while (!in.eof())
        {
```

```
            in.get(symbol);
            if (symbol == ‘ ‘) count_space++;
            if (symbol == ‘\n’) break;
        }

        in.seekg(0, ios::beg);
        in.clear();

        int n = count / (count_space + 1);
        int m = count_space + 1;

        for (int i = 0; i<n; i++)
            x[i] = new int[n];

        for (int i = 0; i < n; i++)
            for (int j = 0; j < m; j++)
                in >> x[i][j];
        in.close();
    }
    else
    {
        cout << "Файл не найден.";
    }

    return x;
}
```

ДОДАТОК Б

Вихідні матриці та результати їх обробки

Inf_Part1, Inf_Part2

```
К-во разделов> 16
                        Исходная матрица:
```

```
0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
0 0 1 1 1 0 0 0 0 0 0 1 0 1 0 0
0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
1
```

```
0 1 1 1 0 0 0 0 0 0 1 0 1 0 0
0 0 1 1 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

```
1 5 6 7 8
```

```
0 1 1 0 0 1 0 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 1
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0
```

```
1 4 5 8
```

```
0 0 0 1 0 0
0 0 0 1 0 0
0 0 0 0 1 1
0 0 0 0 0 0
0 0 0 0 0 1
0 0 0 0 0 0
```

```
1 2 3
```

```
0 0 0
0 0 1
0 0 0
```

```
1 2
```

```
0
```

Последовательность удаления:

1 1 5 6 7 8 1 4 5 8 1 2 3 1 2 1

Aviation

К-во разделов> 7

Исходная матрица:

---

0 1 1 0 0 1 0
0 0 1 0 0 1 1
0 0 0 1 1 0 0
0 0 0 0 0 0 0
0 0 0 0 0 1 0
0 0 0 0 0 0 1
0 0 0 0 0 0 0

1

---

0 1 0 0 1 1
0 0 1 1 0 0
0 0 0 0 0 0
0 0 0 0 1 0
0 0 0 0 0 1
0 0 0 0 0 0

1

---

0 1 1 0 0
0 0 0 0 0
0 0 0 1 0
0 0 0 0 1
0 0 0 0 0

```
1
```

```
0 0 0 0
0 0 1 0
0 0 0 1
0 0 0 0
```

```
1 2
```

```
0 1
0 0
```

```
1
```

```
0
```

```
Последовательность удаления:
1 1 1 1 2 1
```

IT_in_Learn

```
К-во разделов> 7
                        Исходная матрица:
```

```
0 1 1 1 1 1 1
0 0 0 0 0 1 0
0 0 0 1 1 0 0
0 0 0 0 0 1 0
0 0 0 0 0 1 0
0 0 0 0 0 0 0
0 0 0 0 0 0 0
```

```
1
```

```
0 0 0 0 1 0
0 0 1 1 0 0
0 0 0 0 1 0
0 0 0 0 1 0
0 0 0 0 0 0
0 0 0 0 0 0
```

```
1 2 6
```

```
0 0 1
0 0 1
0 0 0
```

```
1 2
```

```
0
```

Последовательность удаления:

```
1 1 2 6 1 2
```