ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ОЛЕСЯ ГОНЧАРА

Факультет прикладної математики

Кафедра обчислювальної математики та математичної кібернетики

Кваліфікаційна робота

перший (бакалаврський) рівень вищої освіти

спеціальність 124 Системний аналіз

АНАЛІЗ АЛГОРИТМІВ ОТРИМАННЯ ЙМОВІРНІСНИХ ВИСНОВКІВ В БАЙЄСОВСЬКИХ МЕРЕЖАХ

Виконавець

студентка групи ПС-17-1

Маркова Анастасія Олександрівна \_\_\_\_\_\_\_\_\_\_\_\_

підпис

Керівник

доц. каф. ПОМ,

канд. фіз.-мат. наук

Турчина В.А. \_\_\_\_\_\_\_\_\_\_\_\_

підпис

Завідувач випускової кафедри

канд. фіз.-мат. наук

Турчина В.А. \_\_\_\_\_\_\_\_\_\_\_\_

підпис

2021

**ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА**

Факультет \_\_\_Прикладної математики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра \_обчислювальної математики та математичної кібернетики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Рівень вищої освіти \_перший (бакалаврський)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Спеціальність \_124 Системний аналіз\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ЗАТВЕРДЖУЮ**

**Завідувач кафедри обчислювальної математики \_\_\_та\_\_\_ математичної кібернетики\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

\_\_\_\_\_\_\_\_\_ Валентина ТУРЧИНА\_\_

(підпис)

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021року

**ЗАВДАННЯ**

**НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Маркова Анастасія Олександрівна\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема роботи «Аналіз алгоритмів отримання ймовірнісних висновків в байєсовських мережах»

Керівник роботи Турчина Валентина Андріївна, к.ф.-м.н., доцент

Затверджені наказом по Університету від «19» березня 2021 року № 332с.

1. Строк подання роботи 04.06.2021 року.
2. Вхідні дані до роботи постановка задачі оцінки рівня компетентності та ациклічні орієнтовані графи, що описують байєсовські мережі, та алгоритми отримання ймовірнісних висновків.
3. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) вивчення областей застосування байєсовських мереж та прикладних задач, що постають в цих областях, розробка алгоритму отримання ймовірнісного висновку в байєсовських мережах, реалізація програмного продукту з використанням парадигми ООП, аналіз отриманих результатів.
4. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень) обов’язкові креслення не передбачені.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
5. Консультанти розділів роботи

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
| Завдання видав | Завдання прийняв |
|  | Консультанти не передбачені |  |  |
|  |  |  |  |

1. Дата видачі завдання 26 жовтня 2021року.

**КАЛЕНДАРНИЙ ПЛАН**

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Назва етапів кваліфікаційної роботи | Термін виконання етапів роботи | Примітка |
| 1. | Опрацювання літературних джерел за тематикою | з 26.10.2020 по 20.11.2020 | виконано |
| 2. | Огляд сфер застосування байєсовських мереж | з 23.11.2020 по 10.12.2020 | виконано |
| 3. | Постановка задачі оцінки рівня компетентності | з 20.01.2021 по 19.03.2021 | виконано |
| 4. | Розробка програмного продукту, пов’язаного з застосуванням байєсовської мережі | з 26.04.2021 по 10.05.2021 | виконано |
| 5. | Дослідження алгоритмів ймовірнісного висновку в байєсовських мережах | з 11.05.2021 по 25.05.2021 | виконано |
| 6. | Програмна реалізація алгоритму виключення змінних | з 25.05.2021 по 1.06.2021 | виконано |
| 7. | Тестування програми, аналіз результатів | з 20.05.2021 по 02.06.2021 | виконано |
| 8. | Оформлення кваліфікаційної роботи | з 11.05.2021 по 03.06.2021 | виконано |
| 9. | Здача кваліфікаційної роботи на кафедру | 04.06.2021 | виконано |

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Анастасія МАРКОВА

Керівник роботи \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Валентина ТУРЧИНА

РЕФЕРАТ

Кваліфікаційна робота: 53с., 20 рис., 1 табл., 11 джерел, 1 додаток.

Об’єкт дослідження: алгоритми формування ймовірнісного висновку, прикладні задачі, що використовують мережі Байєса.

Мета роботи: застосування байєсовських мереж до задачі оцінки рівня компетентності, вивчення та реалізація алгоритму отримання ймовірнісних висновків.

Методи дослідження: методи теорії ймовірностей, методи побудови сіткових мереж, елементи теорії графів.

Отримані результати: проведено порівняння ефективності точних алгоритмів ймовірнісного висновку, дані практичні рекомендації щодо їх застосування.

Ключові слова: БАЙЄСОВСЬКА МЕРЕЖА, АЦИКЛІЧНИЙ ОРГРАФ, ЙМОВІРНІСНИЙ ВИСНОВОК, АЛГОРИТМ ВИКЛЮЧЕННЯ ЗМІННИХ.

ANNOTATION

The graduation research of the 2021-year student Markova A. (DNU, Faculty of Applied Mathematics, Department of Calculating Mathematics and Mathematical Cybernetics) deals with “Analysis of algorithms for obtaining probabilistic conclusions in Bayesian networks”.

The work is interesting due to the novelty of Bayesian networks and methods for constructing accurate probabilistic inferences with its help.

The relevance of the work lies in the use and creation of new methods for constructing probabilistic inference for solving the task of assessing the level of competence.

Modern systems have a large number of factors and various causal relationships. They require special processing to identify patterns. The ideas of clustering methods were presented in this work and taken into account when describing the algorithm for eliminating variables.

The results of the work can be applied in curricula to assess the level of mastering by students of the material.

Bibliography 53, pictures 20, tables 1, supplement 1.

List of keywords: BAYESIAN NETWORK, ACYCLIC ORGRAPH, PROBABILITY CONCLUSION, VARIABLE EXCLUTION ALGORITHM.

ЗМІСТ

[ВСТУП 7](#_Toc74425509)

[1 ПОНЯТТЯ БАЙЄСОВСЬКОЇ МЕРЕЖІ 8](#_Toc74425510)

[1.1 Історія появи поняття байєсовської мережі та її суть 8](#_Toc74425511)

[1.2 Використання ациклічних графів в байєсовських мережах 10](#_Toc74425512)

[1.3 Типи байєсовських мереж та їх застосування 12](#_Toc74425513)

[2 ПРИКЛАДНІ СФЕРИ ЗАСТОСУВАННЯ 15](#_Toc74425514)

[3 ВИКОРИСТАННЯ МАТЕМАТИЧНОГО АПАРАТУ 18](#_Toc74425515)

[4 АЛГОРИТМИ ЙМОВІРНІСНОГО ВИСНОВКУ 20](#_Toc74425516)

[4.1 Алгоритм визначаючого перетину 22](#_Toc74425517)

[4.2 Алгоритм виключення змінних 23](#_Toc74425518)

[5 ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ВИКЛЮЧЕННЯ ЗМІННИХ 27](#_Toc74425519)

[5.1 Опис структури програмного продукту 27](#_Toc74425520)

[5.2 Опис основних модулів програми 28](#_Toc74425521)

[5.3 Опис інтерфейсу користувача 29](#_Toc74425522)

[5.4 Результати тестування 30](#_Toc74425523)

[5.5 Інструкція користувачеві 33](#_Toc74425524)

[ВИСНОВКИ 35](#_Toc74425525)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 36](#_Toc74425526)

[ДОДАТОК А 37](#_Toc74425527)

# ВСТУП

В даній роботі була розглянута відома задача оцінки рівня компетентності та запропоновано метод отримання для неї ймовірнісного висновку. Для цього було досліджено існуючі точні та наближені алгоритми ймовірнісного висновку, розглянуто ідеї двох точних алгоритмів, які послугували розробці модифікованого алгоритму виключення змінних. Отриманий алгоритм може бути застосований до задачі оцінки рівня компетентності у випадку зовнішнього впливу на систему оцінки рівня знань студентів.

Поставлена задача виникла в результаті складнощів під час оцінки знань студентів. Для її розв’язання потрібно було визначитись з моделлю студента, знання якого перевіряються. Результат засвоєння ним матеріалу навчального курсу є компетенцією, яку перевіряють за допомогою тестових завдань. Ця задача вирішується за допомогою представлення її у вигляді байєсовської мережі.

Байєсовська мережа представляється у вигляді графа. Для її опису використовуються дискретні оцінки для кожної вершини та двошарова структура мережі, яка відображає зв’язки компетенцій, що перевіряються, та тестових завдань, які видаються студентам наприкінці певного етапу навчання. Після побудови байєсовської мережі відбувається процес виключення змінних та зміна значень в таблицях ймовірностей для компетенцій. До мережі зі зміненою структурою можна застосувати метод розв’язання задачі оцінки компетентності на основі використання формули Байєса та визначити апостеріорні ймовірності сформованості в студента компетенцій.

# ПОНЯТТЯ БАЙЄСОВСЬКОЇ МЕРЕЖІ

## Історія появи поняття байєсовської мережі та її суть

Байєсовські мережі з’явилися на стику двох наук: теорії ймовірностей та теорії графів (розділ дискретної математики). Байєсовська мережа є одним з двох основних різновидів графових моделей. До ймовірнісних графових моделей також відносяться марковські мережі.

Графові моделі вперше застосовувалися в області статистичної фізики Гіббсом у 1902р., в області генетики – Райтом у 1921р., в області економіки – Уолдом у 1954р., в області соціології – Блейлоком у 1971р., в області статистики – Бартлеттом у 1935р[1].

У 1972 році була побудована система de Bombal, яка використовувала ймовірнісні методи на основі дуже примітивної байєсовської мережі для медичної діагностики[2]. Ця система правильно поставила діагноз гострої болі в животі в 90% випадках, тоді як досвідчені лікарі – лише в 65%.

Широке розповсюдження ймовірнісні методи отримали в 1980-х роках. Саме тоді теоретична розробка байєсовських мереж дала плідні результати. Найбільш впливовими серед них є робота Джуді Перла «Ймовірнісне мислення в інтелектуальних системах» 1988 року та експертна система Pathfinder, розроблена Хекерманом у 1992 році, яка використовувала байєсовську мережу для діагностики зразків патологій.

Саме Перл вперше ввів термін «баєсова мережа» у 1985 році. На сьогоднішній день байєсовські мережі застосовуються в діагностиці несправностей, аналізі даних, розпізнаванні мови та багато ще де. Графові моделі та, зокрема, байєсовські мережі наразі активно розвиваються і розробляються, тож їх вивчення та дослідження є актуальним.

Байєсовська мережа – це ймовірнісна графова модель, яка представляє набір випадкових змінних та їхніх умовних залежностей за допомогою ациклічного орієнтованого графу. Вузли графа представляють собою випадкові змінні, які можуть бути дискретними або неперервними. Дуги графа відображають причинно-наслідкові зв’язки між змінними.

Мережі Байєса використовуються для аналізу процесів різної природи, діяльності людини та функціонування технічних систем. Вони надають можливість враховувати та використовувати в системі вхідні дані будь-якої природи, наприклад, це можуть бути експертні оцінки. Ці дані можуть передаватися «в режимі реального часу», або у вигляді статистичної інформації. Змінні, що використовуються у байєсовських мережах, можуть бути як дискретними, так і неперервними. Зв’язки між змінними носять назву причино-наслідкових зв’язків через те, що ребра графа є орієнтованими, тобто вони є дугами. Завдяки візуалізації вхідної інформації стає дуже легко розуміти структуру взаємодії змінних в байєсовській мережі. Саме це відрізняє мережі Байєса від інших методів інтелектуального аналізу даних[3].

Серед існуючих методів інтелектуального аналізу даних, таких, як популярні моделі “чорних скриньок” та інші, саме мережі Байєса надають найзрозуміліше пояснення своїх висновків, припускають логічну інтерпретацію і модифікацію структури відношень між змінними задачі, а також дозволяють у явній формі враховувати попередній апріорний досвід експертів. Байєсовські мережі дуже зручні для розв’язання прикладних задач користувачів завдяки представленню у вигляді графів.

Перевагами використання мереж Байєса для інтелектуального аналізу даних є можливості врахування невизначеностей статистичного, структурного і параметричного характеру. Це особливо важливо для формування висновку за допомогою різних методів – наближених і точних.

Основною ідеєю побудови графічної моделі мережі Байєса є поняття модульності, тобто розкладання складної системи на прості елементи. Для об'єднання окремих елементів у систему використовуються результати теорії ймовірностей, які дають можливість поєднувати графічні моделі з базами даних. Такий підхід до побудови моделі дає досліднику можливість будувати моделі з множини сильно взаємодіючих змінних. Ефективним також буде створення структури даних для наступної розробки алгоритмів їхньої обробки.

Ідея застосування мереж Байєса не обмежується використанням умовних ймовірностей в орієнтованих графах. Вона включається і в моделі з симетричними зв’язками (випадкові поля та решітки), моделі динамічних процесів (ланцюги Маркова), а також має місце в широкому класі моделей із прихованими змінними, що дозволяють вирішувати ймовірнісні задачі класифікації, розпізнавання образів та прогнозування.

## Використання ациклічних графів в байєсовських мережах

Використання ациклічних орієнтованих графів в байєсовських мережах є узагальненням структури даних дерева, а точніше лісу як їх об’єднання. Дерево (рис. 1.1) – ациклічний орієнтований граф, в якому тільки одна вершина може бути кореневою. Ліс (рис. 1.2) – множина дерев.

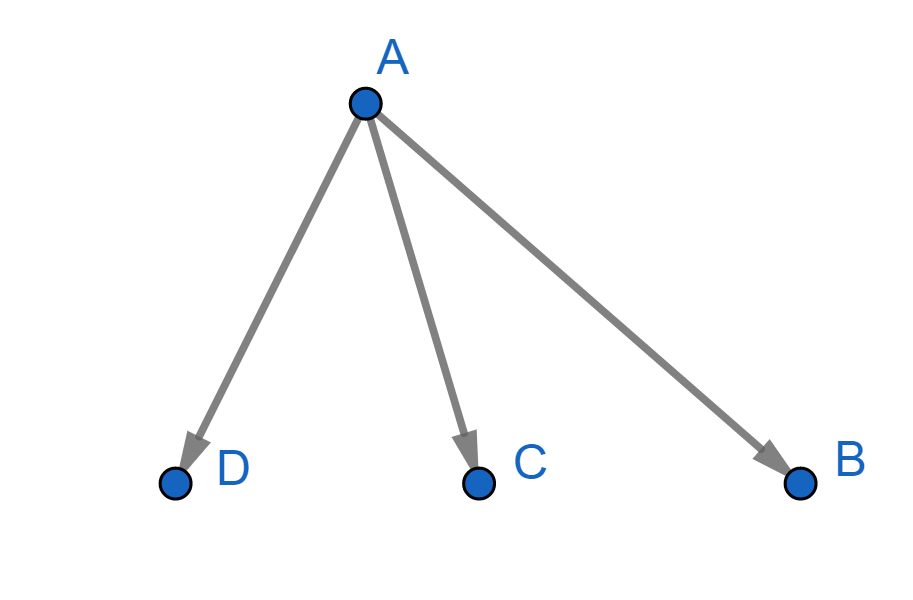


Рисунок 1.1 – Приклад дерева

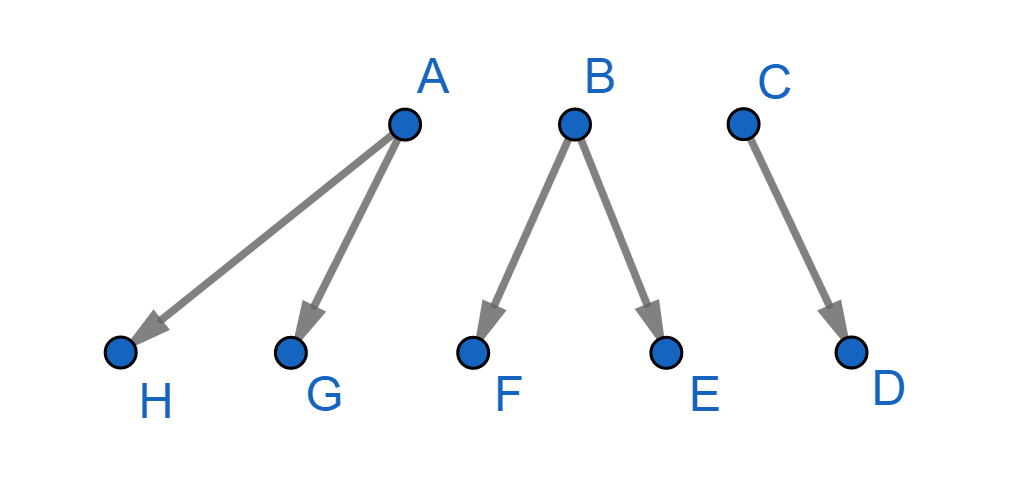


Рисунок 1.2 – Приклад лісу

Ациклічні графи зручні у використанні в задачах пошуку подібних структур. Їх зручно використовуються в задачах інтелектуального аналізу даних таких як пошуку слова в онлайн-словнику (рис. 1.3), категорії на сайті (рис. 1.4) та ін. Інтелектуальний аналіз даних – це процес пошуку закономірностей та прихованих зв’язків між змінними в великих масивах даних, які неможливо обробити без використання комп’ютера[8].

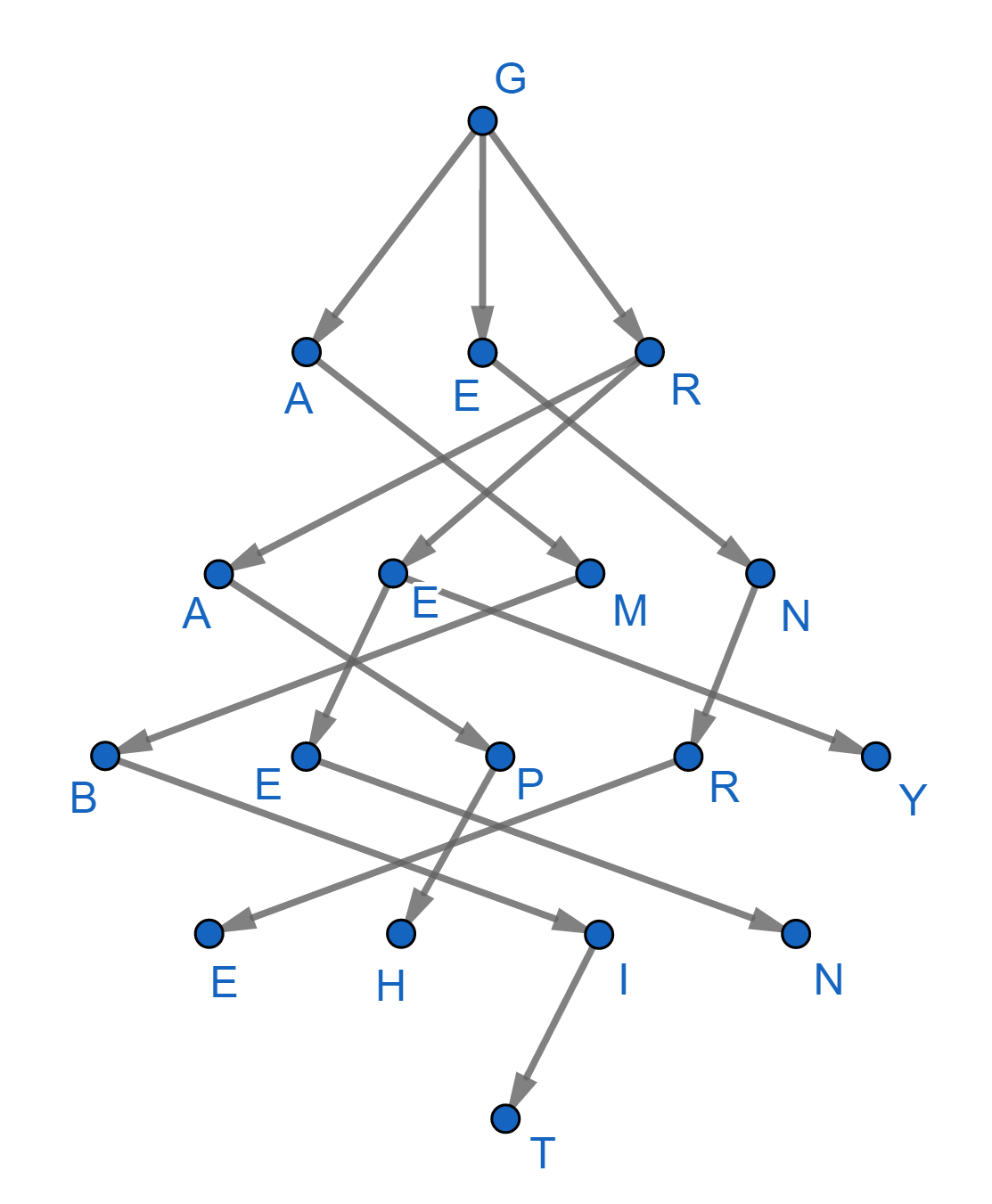


Рисунок 1.3 – Приклад словника

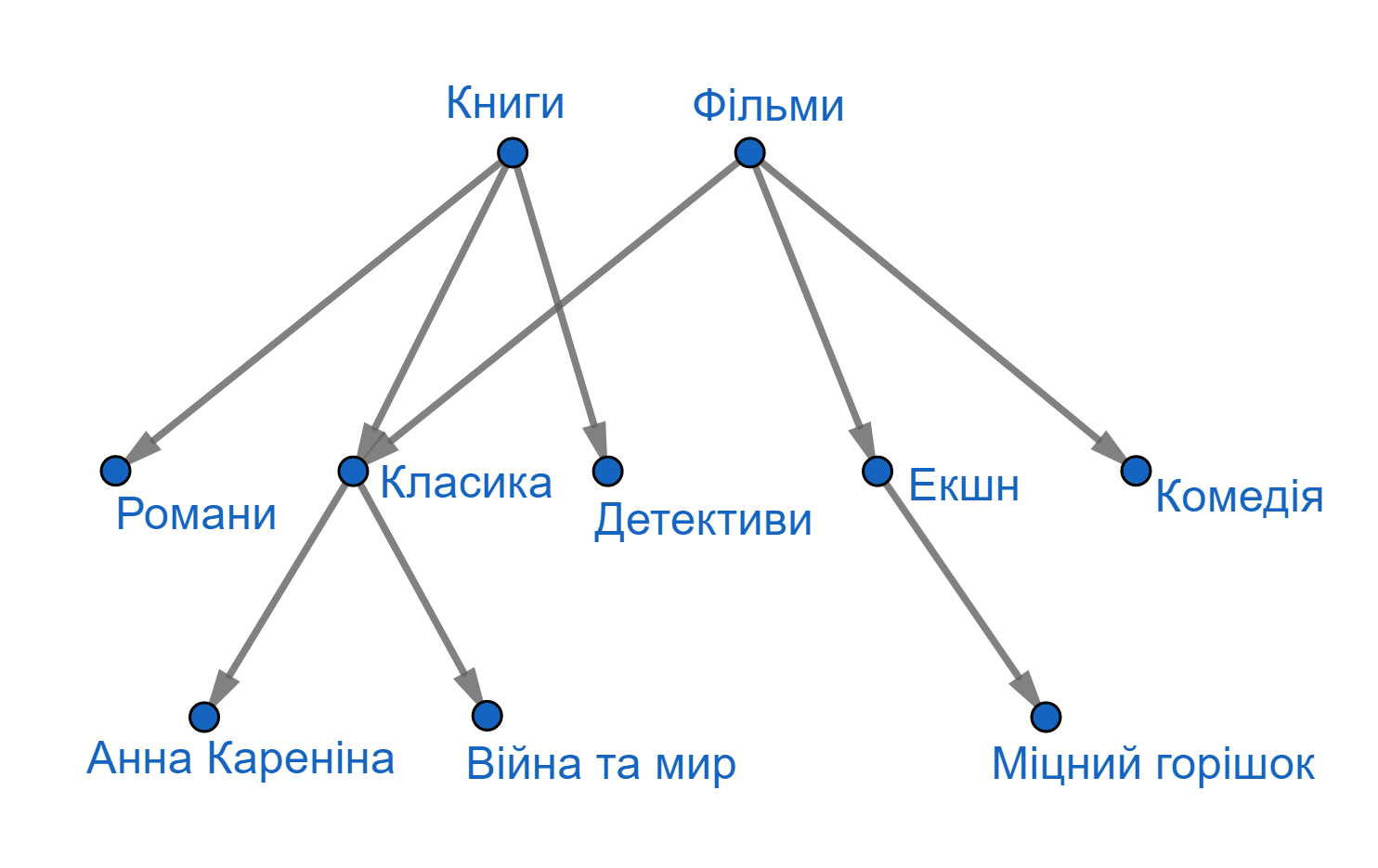


Рисунок 1.4 – Приклад категорій на сайті

## Типи байєсовських мереж та їх застосування

Байєсовські мережі можна поділити на наступні типи за змінними, що відповідають вузлам графа: дискретні, динамічні, неперервні, гібридні та байєсовські нейронні:

*Дискретні байєсовські мережі*

В даному випадку змінні можуть приймати тільки дискретні значення[7]. Кожна вершина графа може мати декілька станів. Якщо вершина має батька (вона не є кореневою), то для неї складається таблиця або функція умовних ймовірностей, а для батьківських вершин – безумовних ймовірностей . Для змінних, що відповідають таблицям умовних ймовірностей, вони складаються відповідно до конкретних станів батьківських вершин.

*Неперервні байєсовські мережі*

Змінні даного типу мереж можуть приймати неперервні значення, тобто описуватися діапазоном вхідних даних. За цих умов вершини можуть приймати нескінчену кількість станів, тобто ймовірність кожного стану наближається до нуля. Щоб уникнути такої некоректності, розподіл ймовірностей для вершин мережі описують не таблицями, а функціями. Такі байєсовські мережі використовують в системах, що моделюють стохастичні процеси.

*Гібридні байєсовські мережі*

Цей тип байєсовських мереж об’єднує властивості двох попередніх. В гібридній байєсовській множині змінні можуть мати як дискретні, так і неперервні значення. Однак ця перевага при формуванні ймовірнісного висновку надається лише за певних обмежень:

* вершини, що мають дискретні значення, не можуть бути дочірніми до вершин, що мають неперервні значення;
* неперервні змінні повинні мати нормальний закон розподілу;
* розподіл неперервної змінної з дискретними та неперервними батьками є нормальним.

*Динамічні байєсовські мережі*

Цей тип мереж застосовується до динамічних систем, в яких фактори змінюються у часі. Зазвичай термін «динамічна» описує зміну параметрів у часі, а не зміну структуру мережі. Але в даному випадку треба мати на увазі можливість існування прихованих мереж, які при їх виявленні точніше опишуть стан системи та змінять структуру мережі.

*Нейронні байєсовські мережі*

Байєсовська нейронна мережа так само, як і нейронна мережа, представляє собою орієнтований ациклічний граф, в якому вершинами є нейрони, а дуги представлені різними типами даних. Їхня різниця полягає в тому, що дуги нейронної мережі представлені числами (вагами), а дуги нейронної байєсовської мережі – ймовірнісними розподілами[4].

Загалом можна сказати, що мережа Байєса використовується для ймовірнісного моделювання процесів довільної природи і подій з невизначеностями різних типів. Мережа Байєса дає змогу достатньо точно описувати функціонування досліджуваних систем та будувати прогнози. Вона є особливо корисною при розробці та аналізі машинних алгоритмів навчання.

Також популярною на сьогоднішній день є когнітивістика – наука, що займається дослідженням та описом процесу людського та тваринного мислення. Вона також може бути застосована до штучної системи, здатної зберігати та передавати інформацію. Зокрема, в її основу покладено вивчення слабкоструктурованих систем, в яких представлено фактори та причинно-наслідкові зв’язки між ними. Когнітивна карта, яка утворюється як результат обробки даних щодо поведінки досліджуваної системи, представляється у графічному вигляді, який легко зрозуміти та інтерпретувати в процесі навчання, наприклад.

# ПРИКЛАДНІ СФЕРИ ЗАСТОСУВАННЯ

Професійна діяльність фахівців в області бізнес-аналітики, медицини, інженерії тощо в 21-му столітті не обмежується знаннями в сферах їхньої діяльності. Майже всі фахівці повинні так чи інакше стикатися на роботі з комп’ютерами, тобто володіти певними навичками та вміннями застосовувати комп’ютерні інформаційні системи[5].

На роботу фахівців може впливати наявність невизначеностей, нечіткої та неструктурованої інформації, що ускладнює їхню діяльність. Вирішенням цієї проблеми послуговує розвиток сучасних інформаційних технологій. Окрім цього, він надає користувачам широкий спектр інструментів для роботи з інформаційними системами. Це дозволяє працювати з інформацією набагато швидше та якісніше. Інтелектуальний аналіз даних, що є однією із таких сучасних технологій, має основну мету у виявленні нетривіальних зв’язків між факторами досліджуваного процесу. Про ці зв’язки можуть не мати уяви навіть експерти у відповідній предметній області. Мережі Байєса є засобом інтелектуального аналізу даних та вони можуть розв’язувати даного роду задачі, які виникають в багатьох сферах людської діяльності.

*Медицина та біологія*

Це найперша на найскладніша область застосування байєсовських мереж. Для діагностики стану пацієнта потрібно попередньо провести аналіз впливу багатьох факторів на і без того складну систему «людина». Окрім того, інформація, отримана в процесі дослідження, може бути неповною або неточною, що ускладнює процес діагностики. Якість отриманих результатів залежить від часу, витраченого на дослідження. Тому при діагностиці часто доводиться вибирати або якість, або оперативність.

Накопичена під час проведених досліджень інформація потребує обробки. Для застосування на практиці результатів теоретичних досліджень необхідно провезти низку тестів, що теж займає багату часу. В цьому випадку до роботи долучається більша кількість фахівців-медиків.

Важливість розробки правильної байєсовської мережі для складної системи захворювань та шляхів лікування особливо добре помітно на початковій стадії постановки попереднього діагнозу та визначення напрямку подальших досліджень. На цій стадії особливо важливо вибрати правильний шлях, що призведе до правильний висновків щодо можливого лікування пацієнта. Це може бути суттєвим у двох випадках: в разі важливості оперативно визначити лікування або в разі фінансового обмеження. Розробка найбільш точних байєсовських мереж допоможе уникнути проблем, пов’язаних з некомпетентністю деяких лікарів, ризиком для пацієнтів та необхідністю проводити зайві дослідження.

*Військове застосування, техніка, безпека*

Основним завданням в даній області, з яким повинна впоратись байєсовська мережа, є розробка шляхів боротьби з тероризмом. Антитерор розвивається, поглинаючи знання про вже існуючі стратегії боротьби проти чітко описаного ворога. Як наслідок, за допомогою байєсовської мережи можливо розробити нові ефективні стратегії, знайти слабкі місця суперника, зменшити ризик несприятливих ситуацій. Це, звичайно, робиться з ціллю знешкодити ворога, запобігти терористичним актам, зменшити ймовірність знаходження слабких місць антитерористичної організації, тобто в захисних цілях.

Головною перевагою ворога є напад на погано підготовленого суперника. Завданням антитерористів є виявлення можливих місць нападу та запобігання можливих наслідків. Для цього й використовуються мережі Байєса.

*Фінанси та економіка*

В даній області Байєсовські мережі допомагають розв’язати проблему накопичення інформації, яку неможливо опрацювати вручну. Вони дозволяють враховувати велику кількість змінних – різноманітних показників, які змінюються в процесі роботи. Структура мережі також підлягає підлаштуванню під зміни в фінансово-економічній сфері, появу нової інформації, необхідність моделювання ринку.

*Комп’ютери і системне програмне забезпечення*

Байєсовські мережі застосовуються в даній області переважно для діагностики несправностей складних технічних пристроїв: принтерів тощо. Вони також відіграють ключову роль в довідкових та пошукових системах. Байєсовські мережі допомагають фільтрувати поштові програми, виділяючи спам серед важливих листів.

*Освіта*

Важливу роль відіграє застосування байєсовських мереж у процесі навчання студентів. За допомогою байєсовської мережі можливо не тільки здійснювати контроль отриманих студентом знань та навичок, а й планувати процес навчання. Наприклад, системи, розроблені науковими центрами відомих університетів, такі як Andes та POET здатні розробляти план вивчення певних дисциплін студентами, допомагати їм в виборі дисциплін для засвоєння на основі здобутих ними навичок, отриманих при вже вивчених дисциплінах, та особистих побажань щодо напрямку розвитку. Такі системи враховують успіхи студента на кожному етапі навчання та здійснюють контроль його успіхів. В процесі роботи з такими системами студент може самостійно визначити в який момент навчання він «випав» з курсу, чи варто йому пройти його повторно або залишити цю ідею та змінити напрямок підготовки, щоб стати фахівцем в іншій області.

# ВИКОРИСТАННЯ МАТЕМАТИЧНОГО АПАРАТУ

Побудова байєсовської мережі передбачає використання знань з теорії ймовірностей та основ теорії графів. Розуміння структури побудови баз даних також допомагає при побудові таблиць ймовірностей для вершин графа.

Байєсовська мережа має вигляд ациклічного орієнтованого графа, який задається у вигляді двох множин:

, (3.1)

де – множина вершин, – множина дуг.

Ймовірності для кореневих вершин, що відповідають компетенціям, вважаються невідомими. Значення, які може приймати кожна компетенція, є рівноймовірними. Таблиці ймовірностей для кореневих вершин заповнюються за формулою:

, (3.2)

де – кількість оцінок в шкалі оцінювання кожної вершини.

В основу мереж Байєса покладено використання формули Байєса, яка має наступний вигляд:

, (3.3)

де та – події,, – ймовірності подій та незалежно одна від одної, – ймовірність події за умови істинності , – ймовірність події за умови істинності .

Ймовірності для всіх інших вершин записуються до таблиць, які відповідають кожній вершині. Вони заповнюються за формулою Байєса, до якої значення беруться з таблиці повних спільних ймовірностей вершин. Ці ймовірності обчислюються за ймовірностями значень вершин-компетенцій та результатами тестувань. Повна спільна ймовірність для прикладу з рисунку 1.1 обчислюється за формулою:

(3.4)

Для прикладу з рис. 3.1 були застосовані формули, аналогічні (3.3). Вони застосовуються до кожної з вершин-компетенцій та мають вигляд:

(3.5)

(3.6)

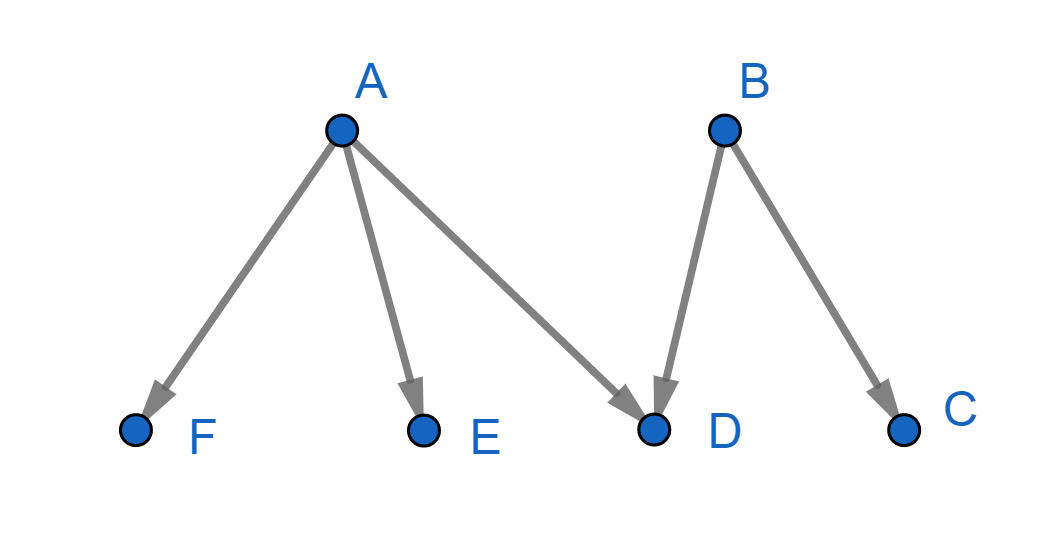


Рисунок 3.1 – Приклад мережі Байєса

# АЛГОРИТМИ ЙМОВІРНІСНОГО ВИСНОВКУ

Після побудови мережі Байєса її можна використовувати та тестувати на різних вхідних даних, моделюючі різноманітні ситуації. Задаючи деякий набір початкових даних, ми отримаємо, що значення деяких вершин більш ймовірні, ніж в інших вершинах. Ймовірнісний висновок – результат, отриманий в процесі оцінки стану вершини на основі апріорних ймовірностей станів інших вершин. Формування ймовірнісного висновку є заключним етапом при застосуванні мережі Байєса до експертної системи.

В загальному випадку, процес формування (обчислення тощо) ймовірнісного висновку є складним процесом. На цей процес впливають найрізноманітніші фактори: кількість дуг та вершин впливають на обсяг обчислень та час роботи програми, різні методи ймовірнісного висновку можуть приносити різні результати, а іноді не всі методи можна застосувати через алгоритмічну складність задачі.

Для невеликих мереж Байєса, які мають вигляд простих дерев, нескладно зрозуміти, що для побудови ймовірнісного висновку треба задати стан кореневих вершин та застосувати до інших вершин теорему Байєса. Також можлива обернена ситуація, коли задано стани дочірніх вершин, а потрібно обчислити значення стану кореневої вершини за допомогою оберненої теореми Байєса. Для великих мереж Байєса, що задані у вигляді об’єднання дерев та мають складну структуру, застосовуються алгоритми, метою яких є перетворення структури мережі для більш зручної роботи з нею.

Взагалі кажучи, немає єдиного алгоритму, який би показував найефективнішу роботу на різних мережах та давав би найточніші результати. Отже розглянемо дві групи алгоритмів ймовірнісного висновку за обчислювальною складністю: точні та наближені. Точні методи формування ймовірнісного висновку для великих мереж Байєса побудувати неможливо[6]. Це пов’язано з обмеженістю пам’яті комп’ютера, в якій потрібно зберігати великі масиви ймовірностей та інформацію про граф. Сучасні комп’ютери дозволяють частково вирішити цю проблему, але складнощі, пов’язані зі швидкістю обчислень, залишаються. Тому суть алгоритмів даного типу полягає в тому, щоб перетворити структуру мережі на більш просту, а потім застосувати до неї метод, який не потребує складних обчислень.

Суть алгоритмів наближеного ймовірнісного висновку полягає в можливості робити обчислення для мереж, в яких кількість вершин перевищує тисячі. Складність такої задачі близька до експоненціальної. Тому ці алгоритми жертвують точністю обчислень у випадку роботи з великими масивами даних. В наступній таблиці наведені основні групи алгоритмів, поділені на точні і наближені.

Таблиця 4.1 – Алгоритми ймовірнісного висновку

|  |  |
| --- | --- |
| **Алгоритми точного ймовірнісного висновку** | **Алгоритми наближеного ймовірнісного висновку** |
| Алгоритм Перла розповсюдження повідомлення в полі-деревах | Алгоритми неповного висновку |
| Алгоритм кластерізації дерева клік | Варіаційні алгоритми |
| Алгоритм визначаючого перетину |
| Алгоритм послідовного виключення змінних | Алгоритми стохастичного вибору |
| Алгоритм символьного ймовірнісного висновку | Пошукові алгоритми, засновані на евристичних алгоритмах пошуку |
| Алгоритм з використанням диференціального підходу |

## Алгоритм визначаючого перетину

Цей алгоритм застосовують до багатозв’язної мережі, перетворюючи її на однозв’язну або декілька однозв’язних. Багатозв’язна мережа – граф, в якому існують підграфи, що утворюють цикли для -повідомлень і -повідомлень. Однозв’язна мережа – мережа, що не є багатозв’язною. Інстанційовані вершини – вершини, що можуть приймати декілька станів, та перетворюють багатозв’язну мережу в однозв’язну. У випадку передачі повідомлень через мережу (-повідомлення посилаються від дочірніх до батьківських вершин, -повідомлення – від батьківських до дочірніх) можуть виникати зациклення. Розглянемо наступний приклад (рис. 4.1).

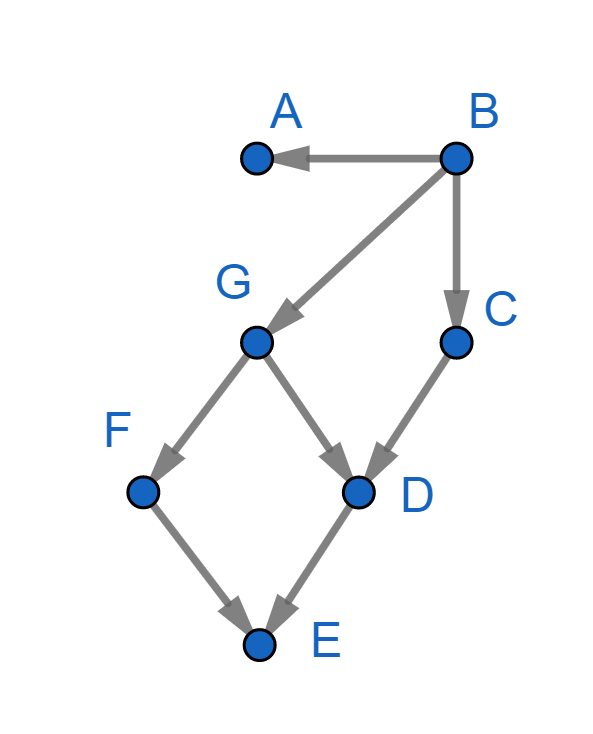


Рисунок 4.1 – Багатозв’язний граф

Мінімальний визначаючий перетин складається з однієї вершини G. В підграфах GBCD та GDEF відбувається зациклення повідомлень. Тоді, припускаючи, що вершина G може бути в двох станах, робимо її інстанційованою та утворюємо дві нових однозв’язних мережі (рис. 4.2, 4.3).

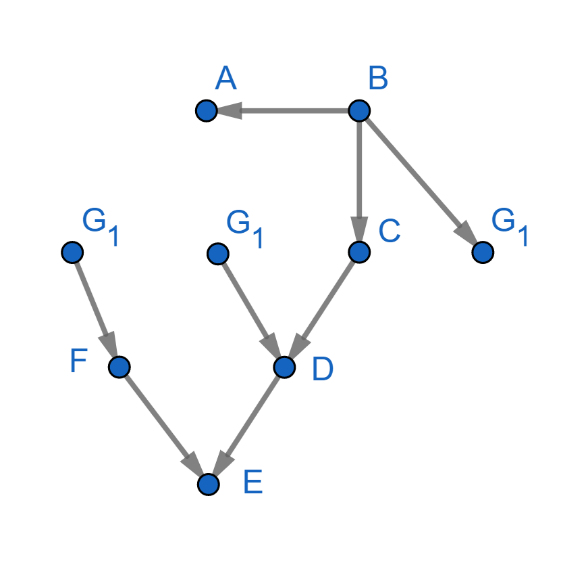


Рисунок 4.2 – Перший однозв’язний граф

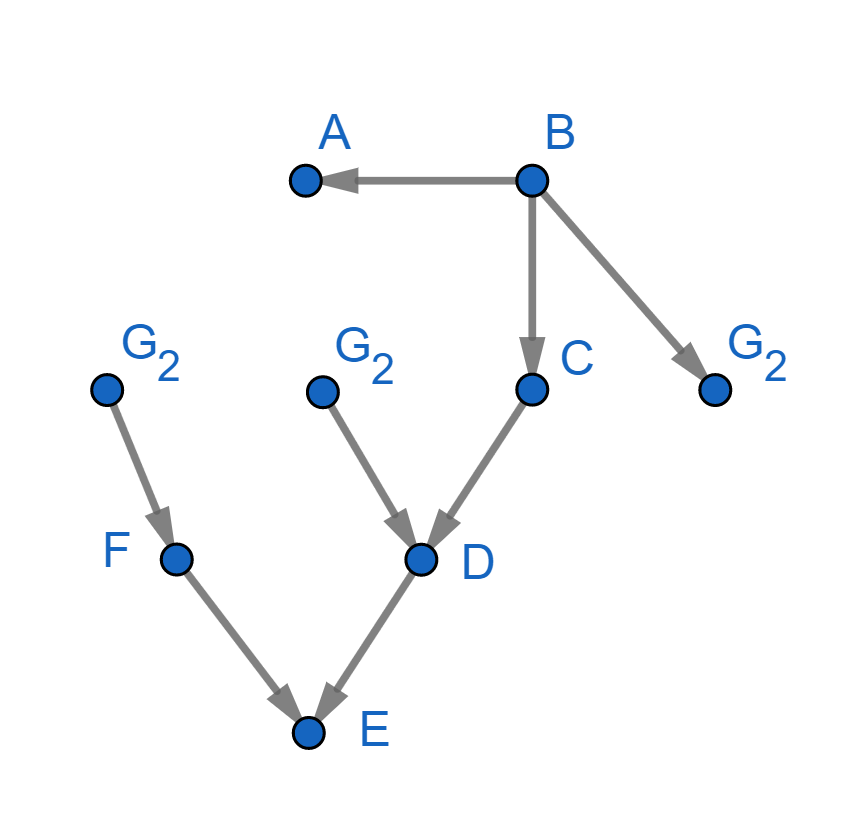


Рисунок 4.3 – Другий однозв’язний граф

## Алгоритм виключення змінних

Суть цього алгоритму полягає в тому, щоб обчислити суму ймовірностей батьківських вершин та перетворити їх на нову вершину. Остаточно мережа Байєса повинна перетворитися на кліку (рис. 4.6). Кліка – підмножина вершин неорієнтованого графа, в якій кожні дві вершини пов’язані ребром. Розглянемо приклад (рис. 4.4). Послідовно виключаємо вершини мережі (рис. 4.5) до утворення кліки.

Спочатку обчислюємо сукупний розподіл вершин : Виключаємо вершину : . І так далі…

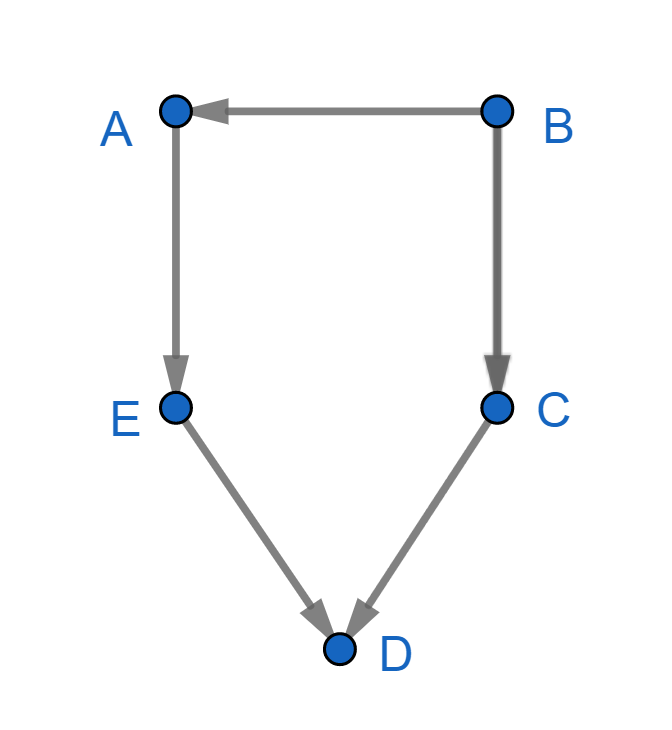


Рисунок 4.4 – Приклад роботи алгоритму виключення змінних. Початковий граф

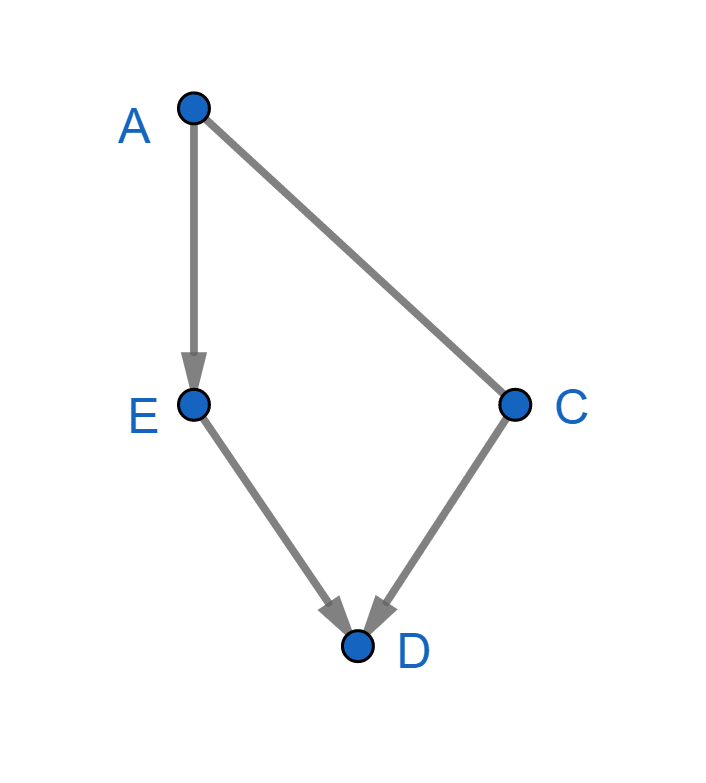


Рисунок 4.5 – Граф з виключеною вершиною B

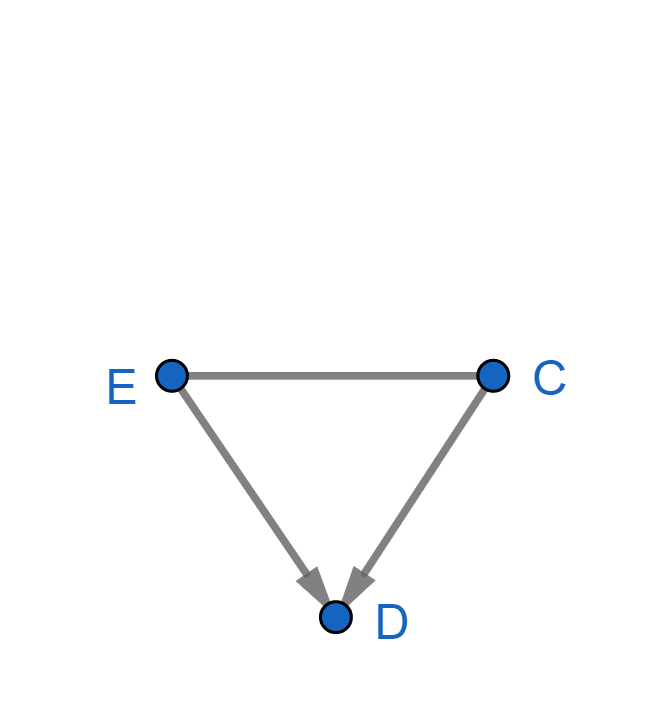


Рисунок 4.6 – Кліка

В рамках даної роботи модифікуємо алгоритм. Будемо вважати, що нам потрібно замінити структуру мережі, перетворивши її не на кліку, а на двошарову байєсовську мережу, в якої перший шар вершин, не зв’язаних між собою, складається з компетенцій, а другий шар вершин, не зв’язаних між собою, складається с тестових питань. Нехай задано мережу, показану на рис. 4.7. Нам потрібно привести її до вигляду, показаному на рис. 4.8.

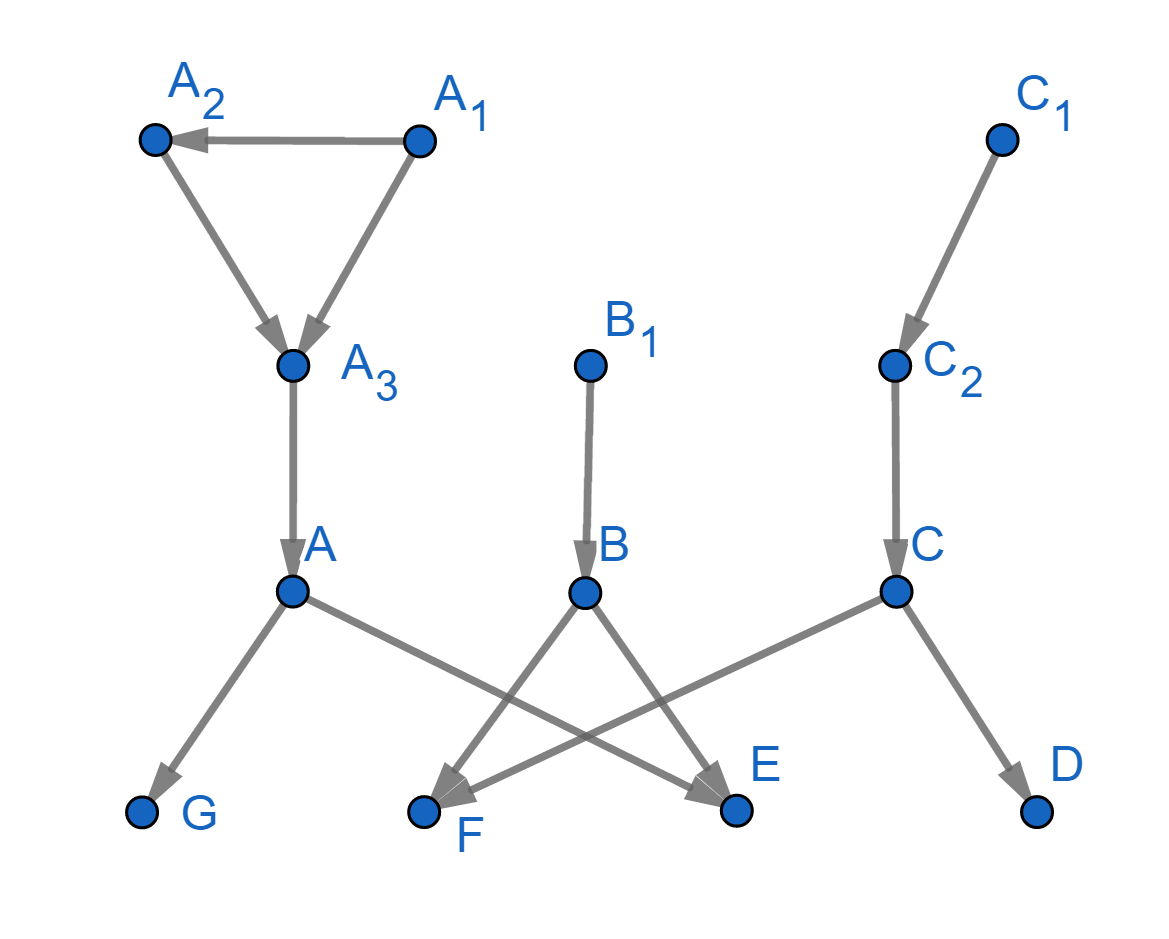


Рисунок 4.7 – Приклад застосування модифікованого алгоритму. Початковий граф

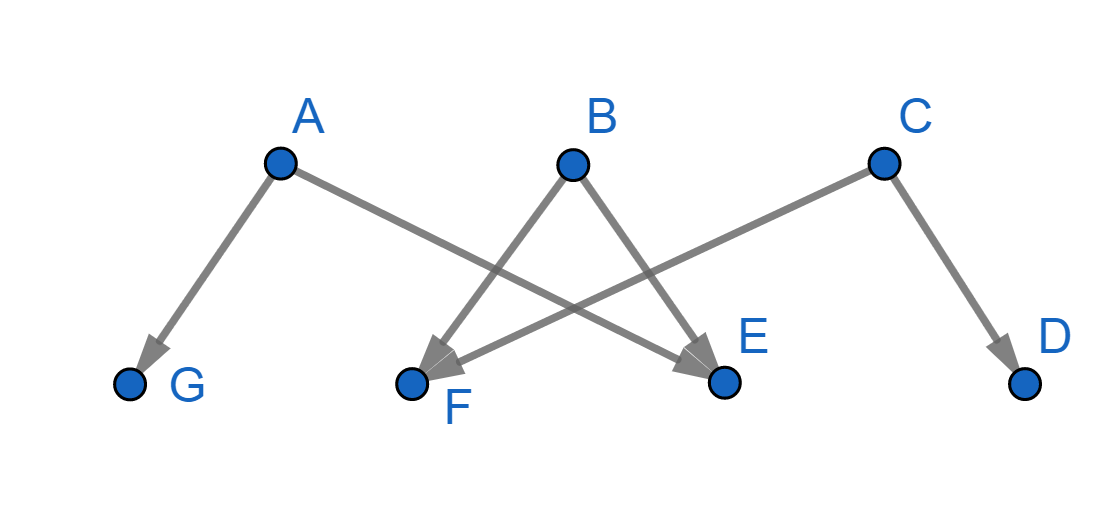


Рисунок 4.8 – Приклад застосування модифікованого алгоритму. Кінцевий граф

*Алгоритм виключення змінних в задачі оцінки рівня компетентності*

Крок 1. Ідентифікація структури байєсовської мережі, в якій зображено тільки залежності між компетенціями та тестовими запитаннями. Якщо ця структура і є даним графом, то алгоритм завершує роботу. Інакше – крок 2.

Крок 2. Виділення множини вершин, що є компетенціями, та вершин, що є питаннями.

Крок 3. Якщо вершина з першої множини, тобто вона є компетенцією, то виявлення підграфу, в якому ця вершина є дочірньою. Перехід до кроку 4. Якщо вершина з другої множини, тобто є питанням, то виявлення підграфу, в якому вона є батьківською вершиною. Перехід до кроку 8.

Крок 4. Для підграфу, виявленому на кроці 3, застосувати ідею алгоритму виключення змінних, виключивши всі змінні окрім вершини, що була дочірньою, так, щоб вона не мала батьків. Для цього перейти до кроку 5.

Крок 5. Якщо підграф складається з однієї вершини, то перехід до кроку 3. Інакше, виявлення батьківської вершини в даному підграфі.

Крок 6. Обчислення сукупного розподілу ймовірностей вершин, які мають спільні дуги з даною вершиною, та даної батьківської вершини.

Крок 7. Виключення батьківської вершини. Перехід до кроку 5.

Крок 8. Для підграфу, виявленого на кроці 3, застосувати ідею алгоритму виключення змінних, виключивши всі змінні окрім вершини, що була батьківською, так, щоб вона не мала дочірніх вершин. Для цього перейти до кроку 9.

Крок 9. Якщо підграф складається з однієї вершини, то перехід до кроку 3. Інакше, виявлення вершини, що не має дітей, в даному підграфі.

Крок 10. Обчислення сукупного розподілу ймовірностей вершин, які мають спільні дуги з даною вершиною, та даної вершини, що не має дітей.

Крок 11. Виключення даної вершини. Перехід до кроку 9.

# ПРОГРАМНА РЕАЛІЗАЦІЯ АЛГОРИТМУ ВИКЛЮЧЕННЯ ЗМІННИХ

## Опис структури програмного продукту

Програма містить два класи:

* основний клас «Main.java», в якому відбувається основна робота програми: зчитування вхідних даних, формування таблиць ймовірностей та виконується робота алгоритму ймовірнісного висновку;
* клас «Data.java», який допомагає зберігати таблиці ймовірностей різних розмірів для кожної вершини графа.

Вхідні дані зчитуються з трьох файлів:

* «InputData.TXT», в якому зберігається інформація щодо максимального значення, яким може бути оцінена кожна компетенція та кожне питання тесту. Градація починається з нуля та закінчується цілим числом.
* «IncidenceMatrix.TXT», в якому записана матриця інцидентності даного графа;
* «Tests.TXT», який містить інформацію щодо оцінок значень, що відповідають кожній вершині графа

Вихідні дані представлені файлом:

* «Output.TXT», в який записується інформація про змінену ймовірність вершин, що відповідають компетентностям, та її початкову оцінку відповідно до кожного набору вхідних даних з файлу «Tests.TXT».

## Опис основних модулів програми

1. Зчитування значення максимальної оцінки з файлу «InputData.TXT» в змінну «maxGrade».
2. Обчислення кількості вершин графа, яка записується у змінну «numVertexes», з файлу «IncidenceMatrix.TXT». Перевірка чи є матриця інцидентності квадратною.
3. Зчитування матриці інцидентності з файлу «IncidenceMatrix.TXT» у двовимірну матрицю «incidenceMatrix[][]».
4. Підрахунок кількості компетенцій та кількості тестових питань з матриці інцидентності, які відповідно записуються в змінні «numCompetences» та «numTestQuestions».
5. Обчислення кількості тестових датасетів, яка записується в змінну «numTests», з файлу «Tests.TXT».
6. Зчитування датасетів з результатами тесту в двомірну матрицю «tests[][]» з файлу «Tests.TXT».
7. Заповнення таблиць ймовірностей для кожної вершини як двомірних матриць «probabilitiesNum[][]» та їх запис разом з номером вершини як об’єкт «data» класу «Data.java» в масив інтерфейсу «List», який має назву «dataFromList».
8. Робота алгоритму виключення замінних. Перерахунок таблиць ймовірностей для вершин компетенцій та перезапис частини матриці інцидентності до нової двомірної матриці «incidenceMatrixChanged[][]».
9. Підрахунок безумовних ймовірностей для вершин, що є компетенціями, та запис їх у двомірний масив «posteriorProbabilities[][]», в якому по вертикалі ідуть компетенції, а по горизонталі – номери датасетів з відповідями на тест.
10. Запис результатів у файл «Output.TXT».

## Опис інтерфейсу користувача

Вхідні дані користувач повинен внести у три файла за правилами, наведеними в інструкції:

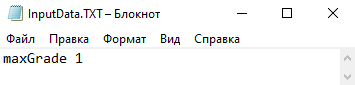


Рисунок 5.1 – Файл з вхідними даними «InputData.TXT»

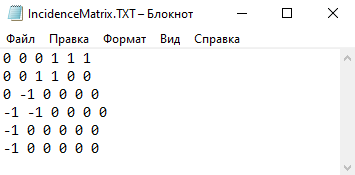


Рисунок 5.2 – Файл з вхідними даними «IncidenceMatrix.TXT»

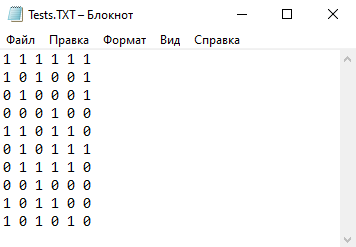


Рисунок 5.3 – Файл з вхідними даними «Tests.TXT»

На рис.5.4 показано результат роботи програми, який користувач отримує у вигляді файлу. Набір вхідних даних позначений номером після ключового слова «Test», номер вершини-компетенції позначений номером після ключового слова «Competence», далі подано початкове значення кожної компетенції та значення початкової (вона однакова для кожної компетенції та дорівнює 0.5) та кінцевої ймовірності для даної компетенції.

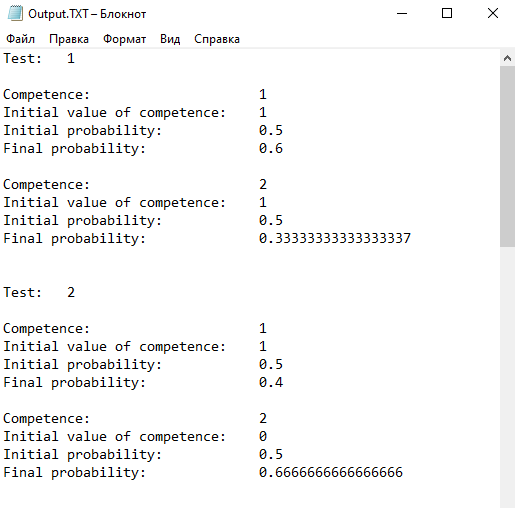


Рисунок 5.4 – Файл з вихідними даними «Output.TXT»

## Результати тестування

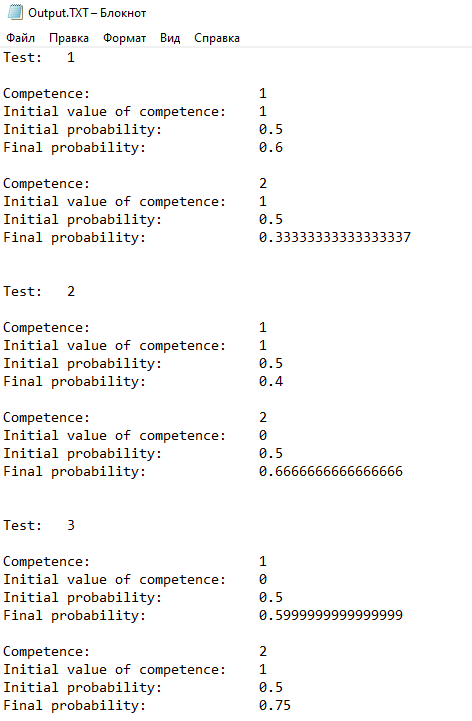


Рисунок 5.5 – Результати тестування. Скріншот 1

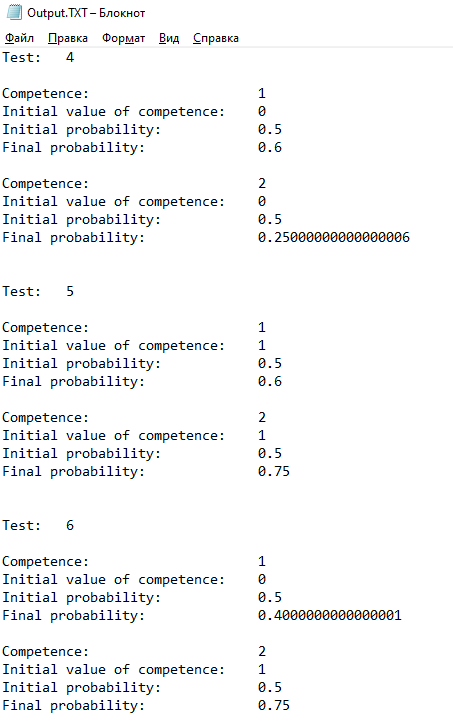


Рисунок 5.6 – Результати тестування. Скріншот 2

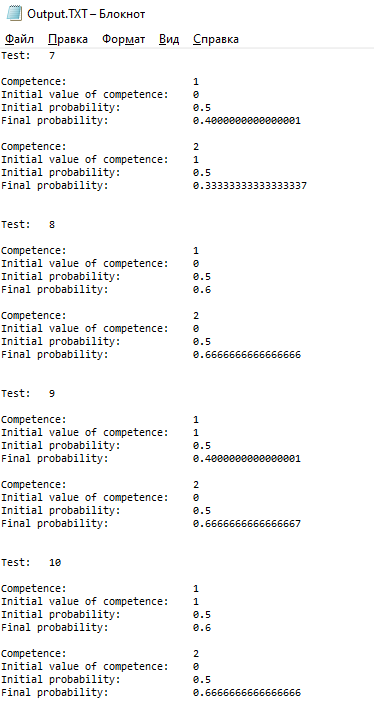


Рисунок 5.7 – Результати тестування. Скріншот 3

## Інструкція користувачеві

Правильність введених даних перевіряється програмою. Якщо будь-яке з нижченаведених правил заповнення файлів буде порушене, програма зупинить свою роботу та повідомить користувачу про його помилку:

* файл з матрицею інцидентності може містити лише цифри «-1», «0» та «1»;
* матриця інцидентності заповнюється дозволеними цифрами за загальними правилами розташування рядків та стовпців зі знаками пробілу та перенесення на новий рядок;
* файл зі значенням максимальної оцінки може містити лише ціле додатне число через знак пробілу після ключового слова «maxGrade»;
* файл з результатами тестувань може містити цілі числа більші або рівні нулю та менші або рівні значення максимальної оцінки;
* результати тестувань в файлі розміщуються наступним чином: по вертикалі ідуть номери вершин графа, по горизонталі – номери тестувань.

# ВИСНОВКИ

В даній роботі була розглянута дискретна байєсовська мережа. Значення змінних компетенцій та тестових питань подавалися наборами вхідних даних. Ці датасети оброблялися програмою таким чином, що до кожної кореневої вершини прикріплялась таблиця ймовірностей, в якій значення були рівноймовірними, а до інших – таблиця умовних ймовірностей. Після використання формули Байєса обчислювалися апостеріорні ймовірності, які і були результатом роботи програми.

Для даної байєсовської мережі було розроблено модифікацію алгоритму виключення змінних. Ідея полягає у перетворенні початкової структури мережі у двошарову байєсовську мережу, де першим шаром є компетенції, а другим шаром – тестові запитання. Після перетворення мережа має простий вигляд та її структуру легко зрозуміти. При цьому вершини мережі не просто «викидаються», а враховуються при перерахуванні значень таблиць ймовірностей вершин компетенцій. Це означає, що вони не будуть рівноймовірними як при першій постановці задачі оцінки рівня компетентності.

Для даної задачі були вибрані саме точні алгоритми ймовірнісного висновку, бо наближені алгоритми застосовуються до мереж зі складною структурою та кількістю вершин, більшу за тисячу. Очевидно, опис такої мережі виходить за рамки написання кваліфікаційної роботи. Серед розглянутих алгоритмів було вибрано два популярних, ідеї яких підштовхнули до створення нового алгоритму, який дав результати застосування до практичної задачі оцінки рівня компетентності.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Хлопотов М.В. Применение байесовской сети при построении моделей обучающихся для оценки уровня сформированности компетенций. Интернет-журнал “Науковедение”. – 2014. – №5 (24) – 28 с.
2. Koller D., Friedman N. Probabilistic graphical models. – 2009. – 1270 с.
3. Згуровський М.З., Бідюк П.І., Терентьєв О.М., Просянкіна-Жарова Т.І. Байєсівські мережі в системах підтримки прийняття рішень. – 2015. – 300с.
4. Байесовская нейронная сеть – потому что а почему бы и нет (часть 1) [Електронний ресурс]. – Режим доступу: <https://habr.com/ru/post/276355/>
5. Cooper A. The inmates are running the asylum. – 1999. – 261с.
6. Бідюк Б.І., Кузнєцов Н.В. Наближені методи формування ймовірнісного висновку та їх модифікація для гібридних мереж Байєса. Наукові праці. Випуск 104. Том 117. – 2009. – С.17-30.
7. Бідюк П.І., Терентьєв О.М., Коновалюк М.М. Байєсівські мережі в технологіях інтелектуального аналізу даних. Штучний інтелект. №2. – 2010. – С.104-113.
8. Марченко О.О., Россада Т.В. Актуальні проблеми Data Mining. Навчально-методичний посібник. – 2017. – 150с.
9. Руководство по языку программирования Java [Електронний ресурс]. – Режим доступу: <https://metanit.com/java/tutorial/>
10. Schildt H. Java. The complete reference. Tenth edition. – 2018. – 1923с.
11. Darwin I.F. Java cookbook. Problems and solutions for java developers. Fourth edition. – 2020. – 949с.

# ДОДАТОК А

Main.java

**package** com.company;  
  
**import** java.io.File;  
**import** java.io.FileWriter;  
**import** java.io.IOException;  
**import** java.text.NumberFormat;  
**import** java.text.ParseException;  
**import** java.util.ArrayList;  
**import** java.util.Locale;  
**import** java.util.Scanner;  
  
**public class** Main {  
  
 **public static void** main(String[] args) **throws** IOException, ParseException {  
 *//building BN  
 //getting input data* NumberFormat nf = NumberFormat.*getInstance*(**new** Locale(**"uk"**, **"UA"**));  
 **int** numCounter;  
 **int** maxGrade = 0;  
 File fileInputData = **new** File(**"G:\\.учеба\\ДНУ. Бакалавриат\\8 семестр\\.практика\\BN\_practice\\src\\com\\company\\InputData.TXT"**);  
 Scanner fileInputDataScanner = **new** Scanner(fileInputData);  
 **while** (fileInputDataScanner.hasNextLine()) {  
 String line = fileInputDataScanner.nextLine();  
 String[] dataString = line.split(**" "**);  
 **if** (!(dataString[0].equals(**"maxGrade"**))) {  
 System.***out***.println(**"Your input data is not enough! Check it!"**);  
 System.*exit*(0);  
 }  
 **if** (dataString[0].equals(**"maxGrade"**)) {  
 **if** (Integer.*parseInt*(dataString[1]) <= 0) {  
 System.***out***.println(**"Your maximum grade value is incorrect! Check it!"**);  
 System.*exit*(0);  
 }  
 maxGrade = Integer.*parseInt*(dataString[1]);  
 }  
 }  
 fileInputDataScanner.close();  
  
 File fileIncidenceMatrixCount = **new** File(**"G:\\.учеба\\ДНУ. Бакалавриат\\8 семестр\\.практика\\BN\_practice\\src\\com\\company\\IncidenceMatrix.TXT"**);  
 Scanner fileIncidenceMatrixCountScanner = **new** Scanner(fileIncidenceMatrixCount);  
 **int** numVertexes = 0;  
 **int** counter = 0;  
 numCounter = 0;  
 **while** (fileIncidenceMatrixCountScanner.hasNextLine()) {  
 String line = fileIncidenceMatrixCountScanner.nextLine();  
 String[] numbersString = line.split(**" "**);  
 numVertexes = numbersString.**length**;  
 **if** (counter != 0) {  
 **if** (numVertexes != counter) {  
 System.***out***.println(**"Your incidence matrix is incorrect! Check it!"**);  
 System.*exit*(0);  
 }  
 }  
 counter = numbersString.**length**;  
 numCounter++;  
 }  
 **if** (numCounter != numVertexes) {  
 System.***out***.println(**"Your incidence matrix is incorrect! Check it!"**);  
 System.*exit*(0);  
 }  
 fileIncidenceMatrixCountScanner.close();  
  
 *//building graph* File fileIncidenceMatrix = **new** File(**"G:\\.учеба\\ДНУ. Бакалавриат\\8 семестр\\.практика\\BN\_practice\\src\\com\\company\\IncidenceMatrix.TXT"**);  
 Scanner fileIncidenceMatrixScanner = **new** Scanner(fileIncidenceMatrix);  
 **int**[][] incidenceMatrix = **new int**[numVertexes][numVertexes];  
 numCounter = 0;  
 **while** (fileIncidenceMatrixScanner.hasNextLine()) {  
 String line = fileIncidenceMatrixScanner.nextLine();  
 String[] numbersString = line.split(**" "**);  
 **int**[] numbers = **new int**[numbersString.**length**];  
 counter = 0;  
 **for** (String number : numbersString) {  
 numbers[counter++] = nf.parse(number.trim()).intValue();  
 }  
 **for** (**int** i = 0; i < numbers.**length**; i++) {  
 **if** (!(numbers[i] == 0 || numbers[i] == 1 || numbers[i] == (-1))) {  
 System.***out***.println(**"Your incidence matrix is incorrect! Check it!"**);  
 System.*exit*(0);  
 }  
 incidenceMatrix[numCounter][i] = numbers[i];  
 }  
 numCounter++;  
 }  
 fileIncidenceMatrixScanner.close();  
  
 **int** beginOfStructure = 0, endOfStructure = 0;  
 **for** (**int** i = 1; i < numVertexes; i++) {  
 **if** (incidenceMatrix[i - 1][i] == 0 && incidenceMatrix[i][i - 1] == 0) {  
 **for** (**int** j = 0; j < numVertexes; j++) {  
 **if** ((incidenceMatrix[i - 1][j] == 1 && j > (i - 1)) || (incidenceMatrix[i - 1][j] == (-1) && j < (i - 1))) {  
 numCounter = 0;  
 **for** (**int** k = 0; k < numVertexes; k++) {  
 **if** (incidenceMatrix[j][k] == 1) {  
 numCounter++;  
 }  
 }  
 **if** (numCounter == 0) {  
 beginOfStructure = i - 1;  
 }  
 }  
 }  
 }  
 }  
 ArrayList<Integer> secondLayer = **new** ArrayList<>();  
 **for** (**int** i = beginOfStructure; i < numVertexes; i++) {  
 **for** (**int** j = 0; j < numVertexes; j++) {  
 **if** (!(incidenceMatrix[i][j] == 1)) {  
 secondLayer.add(i);  
 **break**;  
 }  
 }  
 }  
 **for** (**int** i = secondLayer.get(0); i < numVertexes; i++) {  
 **for** (**int** j = 0; j < numVertexes; j++) {  
 **if** (incidenceMatrix[i][j] == (-1) && !secondLayer.contains(j)) {  
 **if** (!secondLayer.contains(i)) {  
 secondLayer.add(i);  
 }  
 } **else if** (secondLayer.contains(j)) {  
 **break**;  
 }  
 }  
 }  
 endOfStructure = secondLayer.get(secondLayer.size() - 1);  
 **int** numCompetences = 0, numTestQuestions = 0;  
 **int** flagHasNoParent;  
 **for** (**int** i = beginOfStructure; i < endOfStructure; i++) {  
 flagHasNoParent = 0;  
 **for** (**int** j = 0; j < numVertexes; j++) {  
 **if** (incidenceMatrix[i][j] == 1) {  
 flagHasNoParent = 1;  
 }  
 }  
 **if** (flagHasNoParent == 1) {  
 numCompetences++;  
 } **else if** (flagHasNoParent == 0) {  
 numTestQuestions++;  
 }  
 }  
  
 File fileTestsToCheck = **new** File(**"G:\\.учеба\\ДНУ. Бакалавриат\\8 семестр\\.практика\\BN\_practice\\src\\com\\company\\Tests.TXT"**);  
 Scanner fileTestsToCheckScanner = **new** Scanner(fileTestsToCheck);  
 **int** numTests = 0;  
 **while** (fileTestsToCheckScanner.hasNextLine()) {  
 String line = fileTestsToCheckScanner.nextLine();  
 String[] numbersString = line.split(**" "**);  
 **if** (numbersString.**length** != numVertexes) {  
 System.***out***.println(**"You have wrong number of tests! Check them!"**);  
 System.*exit*(0);  
 }  
 numTests++;  
 }  
 fileTestsToCheckScanner.close();  
  
 File fileTests = **new** File(**"G:\\.учеба\\ДНУ. Бакалавриат\\8 семестр\\.практика\\BN\_practice\\src\\com\\company\\Tests.TXT"**);  
 Scanner fileTestsScanner = **new** Scanner(fileTests);  
 **int**[][] tests = **new int**[numTests][numVertexes];  
 numCounter = 0;  
 **while** (fileTestsScanner.hasNextLine()) {  
 String line = fileTestsScanner.nextLine();  
 String[] numbersString = line.split(**" "**);  
 **int**[] numbers = **new int**[numbersString.**length**];  
 counter = 0;  
 **for** (String number : numbersString) {  
 numbers[counter++] = nf.parse(number.trim()).intValue();  
 }  
 counter = 0;  
 **for** (**int** i = 0; i < numVertexes; i++) {  
 **for** (**int** j = 0; j < (maxGrade + 1); j++) {  
 **if** (numbers[i] == j) {  
 counter++;  
 }  
 }  
 **if** (counter == 0) {  
 System.***out***.println(**"Your competences are incorrect! Check them!"**);  
 System.*exit*(0);  
 }  
 }  
 **for** (**int** i = 0; i < numbersString.**length**; i++) {  
 tests[numCounter][i] = numbers[i];  
 }  
 numCounter++;  
 }  
 fileTestsScanner.close();  
  
 *//building probabilities* **int**[] probabilities = **new int**[numVertexes];  
 ArrayList<Data> dataFromList = **new** ArrayList<>();  
 ArrayList<Integer> incomeVertexes = **new** ArrayList<Integer>();  
 Data data;  
 **for** (**int** i = 0; i < numVertexes; i++) {  
 numCounter = 0;  
 incomeVertexes.clear();  
 **for** (**int** j = 0; j < numVertexes; j++) {  
 **if** (incidenceMatrix[i][j] == (-1)) {  
 numCounter++;  
 incomeVertexes.add(j);  
 }  
 }  
 probabilities[i] = (**int**) Math.*pow*((maxGrade + 1), numCounter);  
 **double**[][] probabilitiesNum = **new double**[(probabilities[i])][maxGrade + 1];  
 counter = 0;  
 **int** numLineCounter;  
 **for** (**int** k = 0; k < (probabilities[i]); k++) {  
 **for** (**int** m = 0; m < (maxGrade + 1); m++) {  
 **if** (numCounter == 0) {  
 probabilitiesNum[k][m] = (1.0 / (**double**) (maxGrade + 1));  
 } **else if** (numCounter == 1) {  
 numLineCounter = 0;  
 **for** (**int** n = 0; n < numTests; n++) {  
 **for** (**int** z = 0; z < incomeVertexes.size(); z++) {  
 **if** (tests[n][incomeVertexes.get(z)] == k && tests[n][i] == m) {  
 numLineCounter++;  
 }  
 }  
 }  
 probabilitiesNum[k][m] = (**double**) (numLineCounter) / (**double**) (numTests);  
 } **else if** (numCounter == 2) {  
 numLineCounter = 0;  
 **for** (**int** n = 0; n < numTests; n++) {  
 **for** (**int** z = 1; z < incomeVertexes.size(); z++) {  
 **if** (((maxGrade + 1) \* tests[n][incomeVertexes.get(z - 1)] + tests[n][incomeVertexes.get(z)]) == k && tests[n][i] == m) {  
 numLineCounter++;  
 }  
 }  
 }  
 probabilitiesNum[k][m] = (**double**) (numLineCounter) / (**double**) (numTests);  
 } **else if** (numCounter == 3) {  
 numLineCounter = 0;  
 **for** (**int** n = 0; n < numTests; n++) {  
 **for** (**int** z = 2; z < incomeVertexes.size(); z++) {  
 **if** (((maxGrade + 1) \* tests[n][incomeVertexes.get(z - 2)] + (maxGrade + 1) \* tests[n][incomeVertexes.get(z - 1)] + tests[n][incomeVertexes.get(z)]) == k && tests[n][i] == m) {  
 numLineCounter++;  
 }  
 }  
 }  
 probabilitiesNum[k][m] = (**double**) (numLineCounter) / (**double**) (numTests);  
 }  
 counter++;  
 }  
 }  
 **for** (**int** k = 0; k < (probabilities[i]); k++) {  
 **for** (**int** m = 0; m < (maxGrade + 1); m++) {  
 System.***out***.print(probabilitiesNum[k][m] + **" "**);  
 }  
 System.***out***.println();  
 }  
 System.***out***.println();  
 data = **new** Data(i, probabilitiesNum);  
 dataFromList.add(i, data);  
 }  
  
 *//algorithm* **if** (beginOfStructure != 0) {  
 ArrayList<Integer> structureToExclude = **new** ArrayList<>();  
 **double**[][] tempTable = **new double**[1][maxGrade + 1];  
 **for** (**int** i = beginOfStructure; i < beginOfStructure + numCompetences; i++) {  
 structureToExclude.clear();  
 **for** (**int** j = 0; j < numVertexes; j++) {  
 **if** (incidenceMatrix[i][j] == (-1)) {  
 **if** (!structureToExclude.contains(i)) {  
 structureToExclude.add(i);  
 }  
 **if** (!structureToExclude.contains(j)) {  
 structureToExclude.add(j);  
 }  
 }  
 }  
 **for** (**int** j = 1; j < structureToExclude.size(); j++) {  
 **for** (**int** k = 0; k < numVertexes; k++) {  
 **if** (incidenceMatrix[j][k] == 1 || incidenceMatrix[j][k] == (-1)) {  
 **if** (!structureToExclude.contains(k)) {  
 structureToExclude.add(k);  
 }  
 }  
 }  
 }  
 **for** (**int** j = 0; j < structureToExclude.size(); j++) {  
 **if** (structureToExclude.get(j)>=beginOfStructure) {  
 **for** (**int** z = 0; z < maxGrade + 1; z++) {  
 numCounter = 0;  
 **for** (**int** d = 0; d < numTests; d++) {  
 **if** (tests[d][j] == z) {  
 numCounter++;  
 }  
 }  
 tempTable[0][z] = (**double**) (numCounter) / (**double**) (numTests);  
 }  
 data = **new** Data(i, tempTable);  
 dataFromList.set(i, data);  
 }  
 }  
 }  
 **for** (**int** i = beginOfStructure + numCompetences; i < endOfStructure; i++) {  
 *//* }  
 }  
  
 *//Bayes theorem* **double** sumNumerator, sumDenominator;  
 **double**[][] posteriorProbabilities = **new double**[numTests][numCompetences];  
 ArrayList<Integer> index = **new** ArrayList<>();  
 **for** (**int** g = 0; g < numTests; g++) {  
 **for** (**int** i = 0; i < numCompetences; i++) {  
 sumNumerator = 0;  
 sumDenominator = 0;  
 index.clear();  
 **for** (**int** j = 0; j < numVertexes; j++) {  
 **if** (incidenceMatrix[i][j] == 1) {  
 **if** (!index.contains(j)) {  
 index.add(j);  
 }  
 **for** (**int** k = 0; k < numVertexes; k++) {  
 **if** (incidenceMatrix[j][k] == (-1)) {  
 **if** (!index.contains(k)) {  
 index.add(k);  
 }  
 }  
 }  
 }  
 }  
  
 **for** (**int** m = 0; m < (maxGrade + 1); m++) {  
 **double** tempSum = 1;  
 **for** (**int** k = numCompetences; k < (numCompetences + numTestQuestions); k++) {  
 **if** (index.contains(k)) {  
 numCounter = 0;  
 **for** (**int** h = 0; h < numVertexes; h++) {  
 **if** (incidenceMatrix[k][h] == (-1)) {  
 numCounter++;  
 }  
 }  
 **if** (numCounter == 1) {  
 tempSum \*= dataFromList.get(k).**matrix**[tests[g][i]][tests[g][k]];  
 } **else if** (numCounter == 2) {  
 tempSum \*= dataFromList.get(k).**matrix**[(maxGrade + 1) \* tests[g][i] + m][tests[g][k]];  
 } **else if** (numCounter == 3) {  
 *//* }  
 }  
 }  
 **for** (**int** h = beginOfStructure; h < beginOfStructure+numCompetences; h++) {  
 **if** (index.contains(h)) {  
 **if** (h == i) {  
 tempSum \*= dataFromList.get(h).**matrix**[0][tests[g][i]];  
 } **else** {  
 tempSum \*= dataFromList.get(h).**matrix**[0][m];  
 }  
 }  
 }  
 sumNumerator += tempSum;  
 }  
 **for** (**int** n = 0; n < (maxGrade + 1); n++) {  
 **for** (**int** m = 0; m < (maxGrade + 1); m++) {  
 **double** tempSum = 1;  
 **for** (**int** k = numCompetences; k < (numCompetences + numTestQuestions); k++) {  
 **if** (index.contains(k)) {  
 numCounter = 0;  
 **for** (**int** h = 0; h < numVertexes; h++) {  
 **if** (incidenceMatrix[k][h] == (-1)) {  
 numCounter++;  
 }  
 }  
 **if** (numCounter == 1) {  
 tempSum \*= dataFromList.get(k).**matrix**[n][tests[g][k]];  
 } **else if** (numCounter == 2) {  
 tempSum \*= dataFromList.get(k).**matrix**[(maxGrade + 1) \* n + m][tests[g][k]];  
 } **else if** (numCounter == 3) {  
 *//* }  
 }  
 }  
 **for** (**int** h = beginOfStructure; h < beginOfStructure+numCompetences; h++) {  
 **if** (index.contains(h)) {  
 **if** (h == i) {  
 tempSum \*= dataFromList.get(h).**matrix**[0][n];  
 } **else** {  
 tempSum \*= dataFromList.get(h).**matrix**[0][m];  
 }  
 }  
 }  
 sumDenominator += tempSum;  
 }  
 }  
 posteriorProbabilities[g][i] = (sumNumerator) / (sumDenominator);  
 }  
 }  
  
 **try** (FileWriter writer = **new** FileWriter(**"Output.TXT"**, **false**)) {  
 **for** (**int** g = 0; g < numTests; g++) {  
 writer.write(**"Test: \t"**);  
 writer.write(Integer.*toString*(g + 1));  
 writer.append(**'\n'**);  
 writer.append(**'\n'**);  
 **for** (**int** i = 0; i < numCompetences; i++) {  
 writer.write(**"Competence: \t\t\t"**);  
 writer.write(Integer.*toString*(i + 1));  
 writer.append(**'\n'**);  
 writer.write(**"Initial value of competence: \t"**);  
 writer.write(Integer.*toString*(tests[g][i]));  
 writer.append(**'\n'**);  
 writer.write(**"Initial probability: \t\t"**);  
 writer.write(Double.*toString*(dataFromList.get(i).**matrix**[0][tests[g][i]]));  
 writer.append(**'\n'**);  
 writer.write(**"Final probability: \t\t"**);  
 writer.write(Double.*toString*(posteriorProbabilities[g][i]));  
 writer.append(**'\n'**);  
 writer.append(**'\n'**);  
 }  
 writer.append(**'\n'**);  
 }  
 }  
 }  
}

Data.java

**package** com.company;  
  
**public class** Data {  
 **public int index**;  
 **public double**[][] **matrix**;  
  
 **public int** getIndex() {  
 **return index**;  
 }  
  
 **public double**[][] getMatrix() {  
 **return matrix**;  
 }  
  
 **public void** setIndex(**int** index) {  
 **this**.**index** = index;  
 }  
  
 **public void** setMatrix(**double**[][] matrix) {  
 **this**.**matrix** = matrix;  
 }  
 **public** Data(**int** index, **double**[][] matrix) {  
 **this**.**index**=index;  
 **this**.**matrix**=matrix;  
 }  
}