УДК 004.658

Я.І. Корнага, ст. викладач
Національний технічний університет України «Київський політехнічний інститут», м. Київ, Україна

МЕТОДИ ПІДВИЩЕННЯ ШВИДКОСТІ ЗАПИСУ ТА ПОШУКУ ДАНИХ У БАЗАХ ДАНИХ

Запропоновано нові методи оптимізації запису та пошуку інформації в базах даних. Проведено аналітичне та експериментальне дослідження збільшення швидкості запису та пошуку інформації. Здійснене порівняння швидкості пошуку зі стандартними методами, що базуються на основі індексів, які складаються з B+-дерев.

Ключові слова: *запис даних, пошук даних, індекси, B-дерева.*

Предложены новые методы оптимизации записи и поиска информации в базах данных. Проведено аналитическое и экспериментальное исследование увеличения скорости записи и поиска информации. Осуществлено сравнение скорости поиска со стандартными методами, которые базируются на основе индексов, какие складаються из B+-деревьев.

Ключевые слова: *запись данных, поиск данных, индексы, B-деревья.*

There are represented new methods of record optimization and information retrieval in the database. An analytical and experimental study of speed recording and retrieval information increase was conducted. The comparison of new retrieval rate methods with the standard one (have B+-trees index base) had been done.

Key words: *data recording, data retrieval, indexes, B-tree.*

Вступ. Найбільш важливими процесами в роботі з базами даних є запис інформації та пошук її в базі даних. Основною характеристикою цих процесів є швидкість. У сучасних СУБД при кожному зберіганні даних у таблицю відбувається перебудова індексів, що значно впливає на швидкість запису в БД. Клієнт змушений витрачати час на закінчення попереднього збереження та перебудову індексів. Іноді час затримки перед наступною операцією запису (модифікації) даних доходить до кількох хвилин, що не зовсім допустимо при роботі баз даних, в яких кілька сотень клієнтів [1; 2; 3]. А якщо їх тисячі і мільйони, то збереження буде тривати досить тривалий час та продуктивність роботи серверу відповідно буде падати [4; 5; 6]. Таким чином, є потреба в створенні методів підвищення ефективності запису та пошуку в базах даних.

Метод об'єднаних індексів. Для підвищення ефективності швидкості пошуку однакових даних у різних таблицях створюються об'єднуючі індекси. Це ефективно в тому випадку, коли в базі даних присутнє використання однакових полів у різних таблицях [7].

У листі дерева такого об'єднаного індексу потрібно зберігати запис-посилання на однакове поле в різних таблицях у форматі, як для звичайних B+-дерев, але з тією відмінністю, що запис, крім посилання на розміщення даних у таблиці, повинен містити ще і посилання на саму таблицю, як це показано на рисунку 1.

ключ 1 запис 1.i	ключ 2 запис 2.j	…	ключ k запис k.m

Рис. 1. Структура листової сторінки дерева об'єднаного індексу.

Значення запису-посилання 1, 2,…,k показують конкретне посилання на дані, а значення запису-посилання i, j,…, m – номер таблиці, в якій знаходяться дані.

Це дає можливість суттєво економити час пошуку даних у різних таблицях одночасно, адже посилання на записи будуть зберігатися в одному місці.

Для описання алгоритму використання цього об'єднаного індексу приведемо приклад його роботи. Будемо проводити пошук-порівняння всіх студентів, які навчаються, та студентів, які працюють в університеті. Для цього приведемо приклад запиту, за яким буде відбуватися пошук цієї інформації:

«SELECT прізвище, ім'я, по батькові студента

FROM таблиця Студенти, таблиця Співробітники

WHERE прізвище студента = прізвище співробітника та ім'я студента = ім'я співробітника та дата народження студента = дата народження співробітника»

З цього запиту видно, що в зв'язку з тим, що пошук відбувається за двома таблицями, то відповідно будуть використовуватися індекси, які створенні по полях "прізвище" та "ім'я", двох таблиць. При використанні об'єднаного індексу запит буде використовувати один і той самий індекс, який буде читатися в оперативну пам'ять один раз та швидше видасть результат співпадання ПІБ, що й призведе до зменшення часу пошуку майже вдвоє.

Після розгляду прикладу приведемо метод пошуку на основі алгоритму використання об'єднаних індексів:

1. Клієнт створює запит на пошук даних за кількома таблицями.
2. Запит обробляється компілятором та перевіряється на наявність об'єднаних індексів за даними полів та таблиць.
3. У разі наявності об'єднаного індексу, використовується він, а якщо об'єднаний індекс відсутній, то використовують стандартні табличні індекси для пошуку даних.
4. Клієнт отримує результати пошуку даних за кількома таблицями.

Метод хешування індексів. При частому пошуку даних і використанні одних і тих самих індексів доцільно було б зберігати найчастіше використані індекси в оперативній пам'яті, це б зменшило час доступу до фізичного диску і відповідно зменшило б час, потрачений на пошук інформації.

Наприклад, для практично всіх таблиць БД характерною особливістю є наявність ідентифікаторів, адже порівняння таблиць, для вибору даних з них, проводиться практично завжди саме за допомогою ідентифікаторів основних полів.

Для написання алгоритму методу хешування індексів визначимо основні причини зберігання індексів у пам'яті:

1. Часте використання їх для пошуку даних.
2. Зчитування індексу з жорсткого диску займає набагато більше часу, ніж з оперативної пам'яті.
3. Після операцій запису (модифікації) даних проводиться перебудова індексів, яка займає набагато більше часу, ніж пошук даних.

Відповідно до названих причин створимо метод пошуку на основі алгоритму використання методу хешування індексів за умови, що індекси, які використовуються найчастіше, зберігаються в оперативній пам'яті:

1. Запит, що був створений клієнтом БД, про пошук даних передається на оброблення компілятором.
2. Компілятор визначає таблиці та індекси в них, за допомогою яких потрібно проводити пошук.
3. Перевіряє наявність чи відсутність індексу в оперативній пам'яті.
4. У разі, якщо індекс присутній в оперативній пам'яті, пошук проводиться по ньому, а якщо відсутній, то проводиться зчитування його в пам'ять для подальшого проведення пошуку.
5. Знайдені дані передаються клієнту БД.

Під час запису (модифікації) даних проводяться ті самі операції, що і під час пошуку, з тією відмінністю, що спочатку дані обробляються, а потім відбувається перебудова індексів.

Метод тимчасових таблиць. Для того, щоб не відбувалось затримок у роботі серверу БД, потрібно створити тимчасові таблиці для зберігання даних. Потік інформації з запитами запису, який іде від клієнтів, буде записуватися в них. Тимчасові таблиці створюватимуться для таблиць, в які проводяться операції запису даних найчастіше та в яких використовується багато індексів для проведення пошуку даних.

Тимчасові таблиці створюються на основі існуючих основних таблиць та мають абсолютно ідентичний вигляд з ними. Єдиною відмінністю може бути додавання службо-

вих полів для визначення історії записів, у які можуть входити: час запису та ідентифікатор користувача, який проводив запис.

Наприклад, під час сесії користувачі БД активно вносять оцінки протягом робочого часу в таблицю *Відомості*. Доцільно створити тимчасову таблицю *Відомості_тимчасова_N (де N – номер тимчасової таблиці),* як зображено на рисунку 2, в якій будуть зберігатися предмети з оцінками, які було введено протягом робочого дня та службові поля (ІД користувача, час внесення зміни), а в період найменшої активності користувачів проводити перенесення даних з тимчасової таблиці в основну.

Таблиця Відомості_тимчасова_N

ІД студента	ІД відомості	ІД дисципліни
Оцінка	Оцінка ECTS	Кількість балів
Номер перездачі	Тип занять	Семестр
ІД Викладача	Дата проведення	Дата видачі
Факультет	Курс	Сесія
Підстава перездачі	Навчальна група	ІД користувача
Час внесення змін		

Рис. 2. Приклад тимчасової таблиці для основної таблиці Відомості

Перенесення даних з тимчасової таблиці в основну може відбуватися за чотирма варіантами:

1. Перенесення за кількістю полів заповнення тимчасової таблиці.
2. Перенесення за плином часу заповнення.
3. Перенесення при найменшій активності користувачів.
4. Перенесення за вимогою адміністратора (користувача) БД.

Перший варіант перенесення в основну таблицю здійснюється при перевищенні мінімальної кількості записів у тимчасовій таблиці. Перед початком перенесення завершуються всі транзакції та створюється нова тимчасова таблиця, в яку і будуть писатися дані. Перенесення даних відбувається звичайним дописування в кінець таблиці. Після перенесення обов'язково відбувається перебудова індексів основних таблиць, в які було записано дані.

Другий варіант аналогічний першому, тільки замість мінімального заповнення тимчасової таблиці точкою перенесення є проходження заданого проміжку часу.

При третьому варіанті основною характеристикою процесу перенесення даних є активність користувачів бази даних. Перенесення відбувається при найменшій активності. В основному найменша активність користувачів БД припадає на нічну частину доби, а також, як запасний варіант, під час обідньої перерви.

Четвертий варіант служить як резервний, тому що зазвичай його використовують під час роботи з таблицями у відключеній від користувачів базі даних та проведення профілактики чи відновлювання даних. Тоді для збереження цілісності даних потрібно перенести всі дані з тимчасових таблиць в основні та аж потім проводити роботу з ними.

Для кращого розуміння роботи тимчасових таблиць опишемо їх життєвий цикл, який показаний на рисунку 3.

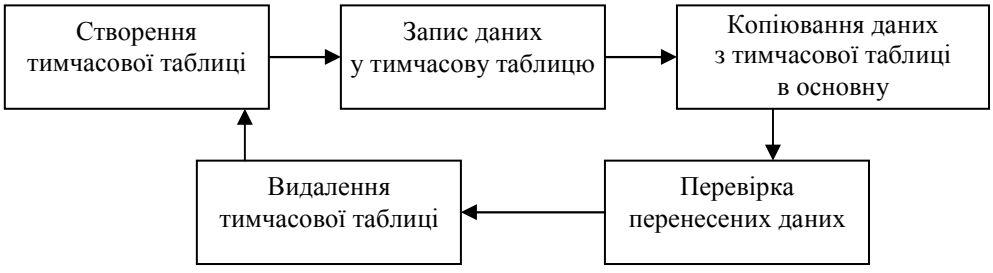

Рис. 3. Схема роботи тимчасових таблиць

Для того, щоб почати роботу з тимчасовими таблицями, СУБД створює їх у форматі, який описаний вище, та фізично розміщує у робочому табличному файлі. Дані, які потрапляють на сервер від користувачів, записуються в них протягом часу, аж поки не настане момент копіювання даних в основні таблиці. Перед самим початком копіювання створюється нова тимчасова таблиця для того, щоб не переривалася робота по внесенню даних користувачами СУБД. Після завершення копіювання даних в основну таблицю відбувається перевірка даних, що були перенесенні. У разі, якщо все пройшло успішно, то відбувається перебудова індексів основної таблиці. Після проведення всіх вищевказаних операцій тимчасова таблиця видаляється, у зв'язку з тим, що на її місці уже працює нова тимчасова таблиця, в яку уже заповнюються нові дані.

Враховуючи вищесказане, опишемо метод запису (модифікації) даних на основі алгоритму використання методу тимчасових таблиць відповідно до варіантів перенесення даних до основних таблиць:

1. Клієнт створює запит на запис даних.
2. Компілятор обробляє запит та визначає таблиці, в які потрібно проводити запис (модифікацію).
3. Якщо для таблиці, в яку потрібно проводити запис, існує тимчасова таблиця, то запис проводиться у неї, а якщо відсутня – то запис проводиться в основну таблицю.
4. Якщо дані, які потрібно модифікувати, знаходяться в основній таблиці, то проводиться модифікація даних та перебудова індексів основної таблиці, а якщо дані, які потрібно модифікувати, не було знайдено в основній таблиці, то проводиться пошук і модифікація даних у додатковій таблиці.
5. Після завершення запису (модифікації) клієнту передається кількість даних, які були записані (модифіковані).

Аналітична оцінка параметрів методів запису та пошуку даних. Під час розрахунку часу пошуку даних за ***методом об'єднаних індексів (МОІ)*** візьмемо до уваги той факт, що, як зображено на рисунку 4, пошук може відбуватися за двома варіантами. Для першого варіанта характерно те, що пошук посилань на дані з різних таблиць у дереві відбувається по одних і тих самих вузлах, тоді економиться час $T_Д$ на зчитування з жорсткого диску вузлів індексу (дерева) для однієї з таблиць, у зв'язку з тим, що вони уже були зчитані для другої таблиці. Відповідно, час пошуку $T_{ПО}$ буде розраховуватися за формулою:

$$T_{ПО} = m(T_Д + 2T_В), \quad (1)$$

де $T_Д$ – час зчитування вузлів дерева індексу пошуку з жорсткого диску, $T_В$ – час пошуку у вузлі дерева для першої та другої таблиці та m – висота дерева.

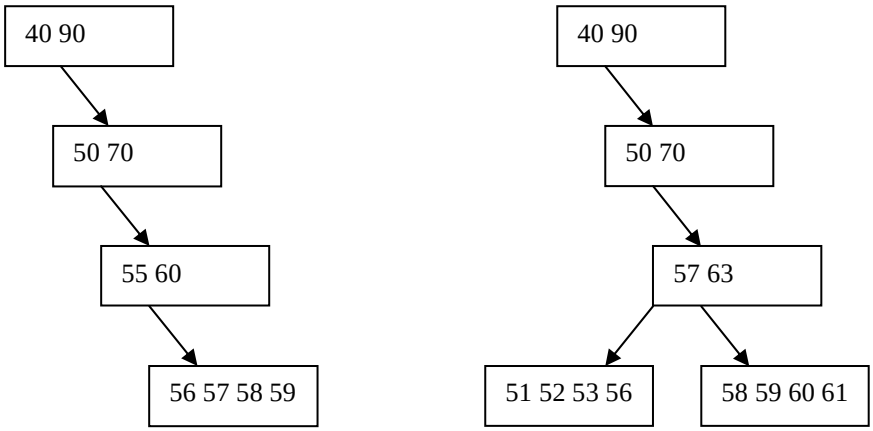

Рис. 4. Приклади пошуку даних у деревах за методом об'єднаних індексів

Для другого варіанта характерно те, що пошук посилань на дані з різних таблиць у дереві відбувається не по одних і тих самих вузлах, тоді економиться час тільки на зчитування з жорсткого диску вузлів індексу (дерева) для однієї з таблиць до того моменту, поки не розходяться шляхи пошуку по дереву. Відповідно, час пошуку $T_П$ буде розраховуватися за формулою:

$$T_{ПО} = m(T_Д + 2T_В) + dT_Д, \quad (2)$$

де $T_Д$ – час зчитування вузлів дерева індексу пошуку з жорсткого диску для першої таблиці, $T_В$ – час пошуку у вузлі дерева для першої та другої таблиці, m – висота дерева та d – кількість вузлів, що додатково потрібно зчитати з жорсткого диску (d завжди менше m).

Отже, для розрахунку часу пошуку даних за методом об'єднаних індексів, потрібно використовувати формулу 2, а в разі відсутності розбіжності у вузлах пошуку, то d = 0 та час пошуку розраховується за першим варіантом.

Фактично відбувається економія часу пошуку даних, що складає: для першого варіанта розрахунку $mT_Д$, а для другого варіанта – $(m-d)T_Д$.

Для оцінки часу пошуку при застосуванні ***методу хешування індексів (МХІ),*** потрібно враховувати той факт, що перед застосування індексу потрібно один раз потратити час $T_{ДЗ}$ для знаходження та зчитування індексу з жорсткого диску. В подальшому буде використовуватися тільки час $T_В$ на пошук у вузлах дерева індексу, яке зберігається в оперативній пам'яті. Відповідно до формули оцінки часу $T_П$ пошуку даних у таблиці можна виключити $T_{ДЗ}$, так як він тратиться тільки один раз. Отже, формула оцінки часу пошуку даних за методом хешування індексів має такий вигляд:

$$T_{ПХ} = mT_В, \quad (3)$$

де $T_В$ – час пошуку у вузлах дерева та m – висота дерева.

Застосування методу хешування індексів дозволяє значно економити час пошуку, підвищити ефективність застосування індексів у БД та в зв'язку з малою кількістю звернень зменшує навантаження на жорсткий диск.

Для оцінки параметрів запису інформації в БД за допомогою ***методу тимчасових таблиць*** потрібно ввести такі позначення:

$T_З$ – час запису інформації в БД,

$T_{ПР}$ – час однієї перебудови індексів для відповідної таблиці,

$T_{ЗТ}$ – час одного запису інформації,

$T_{ПД}$ – час переносу даних з тимчасової таблиці в основну,

n – кількість записів, які проводяться у відповідну таблицю.

У загальному випадку під час запису даних час запису розраховується за формулою:

$$T_З = n(T_{ЗТ} + T_{ПР}), \quad (4)$$

тобто, під час кожного запису витрачається час на сам запис інформації $T_{ЗТ}$ та час на перебудову індексів $T_{ПР}$.

Для методу використання тимчасових таблиць характерною особливістю є відсутність багаторазової перебудови індексів для таблиці, тому формула часу запису інформації в БД матиме вигляд:

$$T_З = nT_{ЗТ} + T_{ПР} + T_{ПД}, \quad (5)$$

тобто, замість виконання n перебудов індексів ми перебудовуємо їх тільки один раз.

Так як час $T_{ПР}$ перебудови індексів значно більший за час $T_{ПД}$ запису з тимчасової таблиці в основну, то $T_{ПД}$ можна знехтувати, тоді виграш часу запису даних з використання методу тимчасових таблиць складає $(n-1)T_{ПР}$, що дозволяє виконувати більше операцій запису одночасно.

Експериментальні оцінки параметрів методів пошуку даних. Візьмемо час $T_Д$, за який відбувається пошук та зчитування відповідного вузла дерева на жорсткому диску рівним 10 мс, а час $T_В$ пошуку у цьому вузлі рівним 2 мс та підставимо його у формули для розрахунку методів пошуку і порівняємо їх з часом пошуку по B+-деревах (табл. 1).

Таблиця 1

Час пошуку в мс для різної кількості записів та середньої наповненості дерева

Кількість записів	B+-дерева	MOI	MXI
5000	24	7	2
10000	24	10.5	3
50000	24	10.5	3
100000	24	10.5	3
500000	36	10.5	3
1000000	36	14	4
5000000	36	14	4

На рисунку 5 зображений графік, який показує відмінність у часі пошуку для різних методів пошуку.

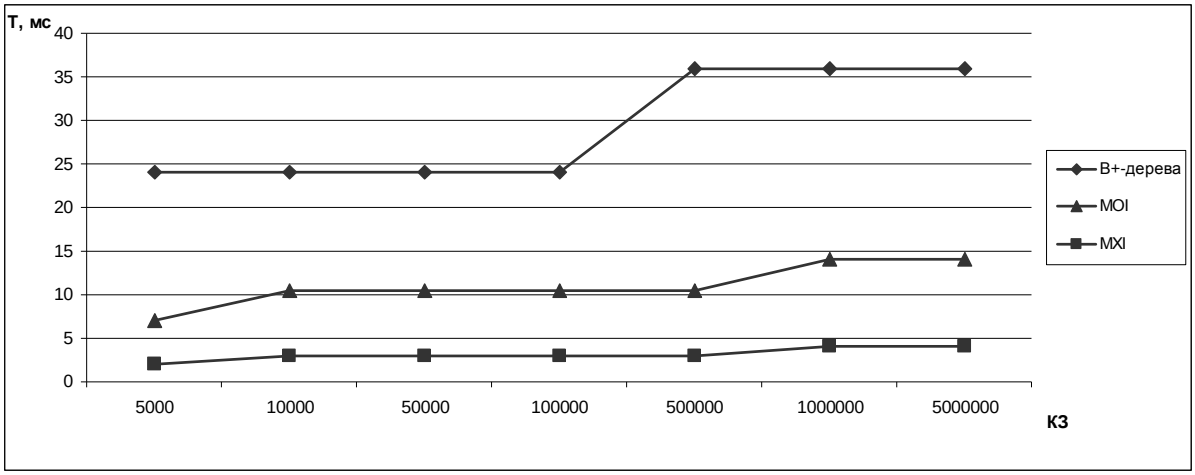

Рис 5. Залежність часу пошуку від кількості записів для різних методів

Висновки. Пошук на основі методу об'єднаних індексів доцільно використовувати, для тих таблиць, які найчастіше попадають під пошук та поля, в яких найчастіше порівнюють між собою. Зазвичай він буде використовуватися для двох таблиць, але зі збільшенням об'ємів БД та розподіленістю її елементів він може використовуватися і для більшої кількості таблиць. Використання методу хешування індексів призводить до покращення часу пошуку та швидкості перебудови індексів під час запису (модифікації) даних у БД у тих випадках, коли один і той самий індекс використовується в багато разів за короткий проміжок часу. Використання методу тимчасових таблиць для запису даних у БД веде до зменшення часу запису, в зв'язку з відсутністю в тимчасовій таблиці БД індексів та потребою в їх перебудові.

Таким чином, вибір певного методу запису та пошуку даних залежить від декількох параметрів побудови бази даних, а саме: кількості таблиць та записів у них, кількості звернень до відповідної таблиці, кількості побудованих індексів на відповідну таблицю та типів даних у відповідних полях БД.

Список використаних джерел

1. Кренке Д. Теорія й практика побудови баз даних / Д. Кренке. – СПб.: Питер, 2008. – С. 800.
2. Хомоненко А. Д. Бази даних / А. Д. Хомоненко, В. М. Циганок, М. Г. Мальцев. – СПб.: Корона, 2007. – С. 736.
3. Портякин І. Ввведення в системи баз даних / І. Портякін. – М.: Вільямс, 2006. – С. 1328.

4. Джен Л. Проектування реляційних баз даних / Л. Джен, С. Харрингтон. – М.: Лорі, 2006. – С. 230.
5. Горєв А. Ефективна робота із СУБД / А. Горєв, Р. Ахаян, С. Макашарипов. – СПб., 2006. – С. 704.
6. Коннолі Т. Бази даних. Проектування, реалізація та супровід / Т. Коннолі, К. Бегг. – М.: Вільямс, 2008. – С. 1440.
7. Мишра С. Секрети Oracle SQL / С. Мишра, А. Бьюли. – СПб.: Символ-плюс, 2006. – С. 368.

УДК 519.161

А.Н. Подоляка, ст. преподаватель
НАУ «ХАИ», г. Харьков, Украина

О.А. Подоляка, канд. техн. наук
ХНАДУ, г. Харьков, Украина

Е.В. Скакалина, канд. техн. наук
ПНТУ ім. Ю. Кондратюка, г. Полтава, Украина

ЭФФЕКТИВНОЕ РЕШЕНИЕ ЗАДАЧИ ПОКРЫТИЯ ДВУДОЛЬНОГО ГРАФА ЗВЕЗДАМИ И НЕКОТОРЫХ ЕЕ ОБОБЩЕНИЙ

В работе рассматривается класс полиномиально разрешимых задач, сводимых к задаче покрытия двудольного графа звездами, а также некоторые похожие труднорешаемые задачи, в частности, коммивояжёра, трехмерного сочетания и задача о покрытии множеств.

***Ключевые слова**: реберное покрытие графа, звезды, циклы, паросочетания.*

У роботі розглядається клас поліноміально вирішуваних задач, які зводяться до задачі покриття дводольних графа зірками, а також деякі схожі складновирішувані задачі, зокрема, комівояжера, тривимірного сполучення і задачі про покриття множин.

***Ключові слова:** реберне покриття графа, зірки, цикли, паросполучення.*

We consider the class of polynomial problems reducible to the problem of the bipartite graph stars covering, as well as some of hard problems: traveling salesman, three-dimensional machining and the set covering problem.

***Key words:** edge graph covering, stars, cycles, machining.*

Постановка задачи покрытия двудольного графа звездами. Рассмотрим оптимизационную задачу покрытия взвешенного двудольного графа $G(V_1,V_2)$ звездами одинаковой степени.

h-звезда – это связный двудольный граф $G_{1,h}$, одна из долей которого имеет степень равную h и называется вершиной звезды, а каждая из вершин другой доли – единичную степень. Звезду можно также рассматривать как дерево, у которого количества ребер и листьев равны степени звезды.

h-звездным покрытием графа назовем множество $E_h \in E$ ребер графа инцидентных звездам, степень которых не превосходит h.

Пусть w – весовая функция, тогда вес звездного покрытия определяется как сумма весов его ребер.

$$w(E_h) = \sum_{e \in E_h} w(e). \qquad (1)$$

Наибольшим h-звездным покрытием (НЗП) графа назовем h-звездное покрытие $E_h^{'} \in E$ максимальной мощности, т.е. для любого $E_h^{'}, \left|E_h^{'}\right| \geq \left|E_h\right|$. НЗП, покрывающее все вершины графа, назовем *совершенным ЗП*. Множество всех возможных наибольших h-звездных покрытий графа обозначим $E_h^{'all}$.

В рассматриваемой задаче первой доли исходного графа соответствуют вершины звезд, а вершины второй доли – это листья звезд. Для удобства будем считать, что ко-