# **Метод гілок та меж для розв’язання задачі комівояжера**

Підготували студенти групи ПМІм-13:  
Загакайло Софія,

Лєснік Уляна,

Малашняк Марія,

Байкенич Ірина-Анастасія,

Сабадашка Олександр

# **Вступ**

Задача комівояжера (Traveling Salesman Problem, TSP) є однією з найвідоміших і найвивченіших задач комбінаторної оптимізації. У своїй основі вона ставить запитання: як комівояжер може обійти певний набір міст, відвідавши кожне з них рівно один раз, та повернутися до початкового міста, при цьому мінімізуючи загальні витрати (наприклад, відстань або час).

Вперше задача була сформульована ще в 1800-х роках як практична задача для комівояжерів (торгових агентів), яким потрібно було розробити маршрут для відвідування декількох міст із мінімальними витратами на подорож. Вона стала популярною серед математиків у XX столітті завдяки своїй універсальності та складності. Задача має важливе значення не тільки в теорії оптимізації, але й у практичних застосуваннях — від логістики та планування маршрутів до молекулярної біології, телекомунікацій та навіть машинного навчання.

Одним з найбільш ефективних підходів до розв'язання TSP є метод гілок та меж (branch and bound), який дозволяє знаходити оптимальні рішення шляхом систематичного перебору можливих маршрутів з використанням обчислень меж для скорочення пошукового простору. У цьому рефераті ми розглянемо історію методу, його концепції та принципи роботи, його переваги та недоліки, а також застосування в контексті задачі комівояжера.

## **Постановка задачі комівояжера**

Задача комівояжера формулюється таким чином:

* Є набір міст, кожне з яких повинно бути відвідане лише один раз.
* Комівояжер повинен почати і завершити маршрут у певному місті (наприклад, у місті A).
* Відомі витрати або відстані для кожної пари міст, що визначають "ціну" переміщення між містами.
* Необхідно знайти найкоротший можливий маршрут, який дозволить комівояжеру відвідати кожне місто один раз і повернутися в початкову точку.

Математично задача комівояжера може бути представлена у вигляді зваженого графа, де кожна вершина (місто) з'єднана з іншими вершинами ребрами, що мають певну вагу (вартість переходу між містами).

Початкова умова може задаватись також як матриця суміжностей, де значення - відстані між містами чи витрати на переміщення, тобто "ціна".

## 

## **Оцінка складності задачі**

Кожен можливий розв’язок задачі представлено Гамільтоновим циклом. У графі з n вузлами загальна кількість можливих циклів Гамільтона визначається як, (n-1)!, якщо розглядати шляхи від А до Б та від Б до А як два різні шляхи. Кількість можливих маршрутів, відповідно, теж (n−1)!, оскільки для кожного міста потрібно перевірити всі інші як наступні точки маршруту. Це робить повний перебір варіантів неефективним, навіть для відносно невеликих значень n.

З точки зору обчислювальної складності, задачі поділяють на два класи - із так званою P-складністю та NP-складні задачі. Під перший варіант підпадають ті, де проблему прийняття рішення можна розв’язати за поліноміальний час за допомогою детермінованої машини Тьюрінга (DTM). Проблеми, які можна розв’язати за допомогою алгоритму з поліноміальним часом, називаються проблемами, що піддаються обробці. Прикладами таких алгоритмів є лінійний пошук (O(n)) , двійковий пошук (O(log n)) , сортування вставкою (O(n²)) , сортування злиттям (O(n log n)) і множення матриць (O( n³)).

Задача комівояжера належить до класу NP-складних задач. Це означає, що для великих розмірів задачі не існує поліноміального алгоритму, який міг би точно її розв'язати в розумний час (якщо не брати до уваги евристичні підходи або випадкові розв'язки). Коли для проблеми NP надано потенційне рішення, детермінована машина Тьюрінга може перевірити його правильність за поліноміальний час.

З огляду на все, сказане вище, що якщо існує велика кількість міст, неможливо оцінити кожне можливе рішення протягом «розумного» часу

Для практичного застосування TSP часто використовують наближені алгоритми, наприклад, евристики або метаевристики — жадібний алгоритм, генетичні алгоритми, алгоритм мурашиних колоній та інші. Вони дають рішення, близькі до оптимальних, за прийнятний час. У цьому рефераті розглянуто метод гілок та меж, як один із прикладів таких алгоритмів

## **Історія методу гілок та меж**

Метод гілок та меж почав формуватися в середині XX століття, коли вчені почали активно досліджувати проблеми оптимізації. Основи методу були закладені Річардом Беллманом у 1950-х роках, який розробив принципи динамічного програмування. Беллман показав, що багато задач оптимізації можуть бути розбиті на підзадачі, рішення яких можуть бути використані для знаходження оптимального розв’язання загальної задачі.

Протягом 1960-70-х років метод гілок та меж отримав подальший розвиток завдяки роботам таких вчених, як Лоуренс Шредер та Джонатан Лемпель. Вони формалізували ідеї про гілки та межі, продемонструвавши, як можна застосовувати ці концепції до широкого спектра задач, включаючи TSP, задачу про рюкзак, задачі на мережах і багато інших. Одним з ключових моментів стало розуміння того, що використання оцінок меж може суттєво зменшити обсяг обчислень, необхідних для пошуку оптимального рішення.

На початку 1980-х років метод гілок та меж став стандартним підходом до розв'язання багатьох комбінаторних задач. Розробка нових алгоритмів та вдосконалення існуючих методів призвели до появи більш ефективних реалізацій, які можуть справлятися з великими обсягами даних. У сучасному світі метод гілок та меж активно використовується у практичних задачах, таких як планування маршрутів для доставки, оптимізація виробничих процесів та управління логістикою.

## 

## **Загальні концепції методу гілок та меж**

Метод гілок та меж є систематичним підходом до пошуку оптимальних рішень, який включає в себе три основні компоненти: гілки, межі та відсічення. Розглянемо їх детальніше.

### **1. Гілки (Branching)**

Гілки представляють процес розбиття задачі на менші підзадачі. У контексті задачі комівояжера, на початку алгоритму вибирається стартове місто, з якого комівояжер розпочинає свій маршрут. Це місто вважається коренем дерева рішень. У цей момент всі шляхи ще відкриті для розгляду, оскільки жодне інше місто ще не відвідано. На кожному кроці алгоритм обирає нове місто для відвідування, формуючи відповідну гілку. Наприклад, на початку комівояжер може вибрати будь-яке місто як наступну точку призначення. З кожним кроком число доступних міст скорочується, і формуються нові підзадачі, що містять інформацію про всі вже відвідані та невідвідані міста.

Кожне нове місто, вибране для відвідування, розгалужує пошук на кілька можливих шляхів, формуючи дерево рішень. Процес розгалуження продовжується доти, поки комівояжер не відвідає всі міста. У результаті, дерево рішень містить усі можливі маршрути, які може пройти комівояжер.

### **2. Межі (Bounding)**

Для кожної з підзадач, що виникають у процесі гілок, розраховується нижня межа витрат (наприклад, відстані). Ця межа відображає мінімальні можливі витрати для даної підзадачі, виходячи з відомих даних про вже відвідані міста та залишені для відвідування.

Для обчислення меж можна використовувати різні методи, такі як:

* **Евристичні оцінки**: вони можуть дати швидку оцінку витрат, не проходячи через усі можливі маршрути.
* **Метод динамічного програмування**: він дозволяє зберігати результати обчислень для вже розглянутих підзадач і використовувати їх повторно.

Якщо обчислена межа для певної підзадачі перевищує вже знайдене оптимальне рішення, ця гілка (підзадача) може бути відсічена, оскільки немає сенсу продовжувати її дослідження.

### **3. Відсічення (Pruning)**

Відсічення є критично важливим етапом методу гілок та меж, оскільки воно дозволяє зменшити кількість обчислень шляхом відкидання неперспективних маршрутів. Якщо межа підзадачі вказує на те, що вона не може дати кращого результату, ніж уже знайдене оптимальне рішення, вона відсікається з подальшого розгляду.

Процес відсічення значно знижує обсяг даних, які потрібно проаналізувати, і допомагає зосередитися на найбільш перспективних варіантах рішень. У випадку задачі комівояжера, це може суттєво зменшити кількість маршрутів, які потрібно перевірити, що, в свою чергу, скорочує час на розв'язання.

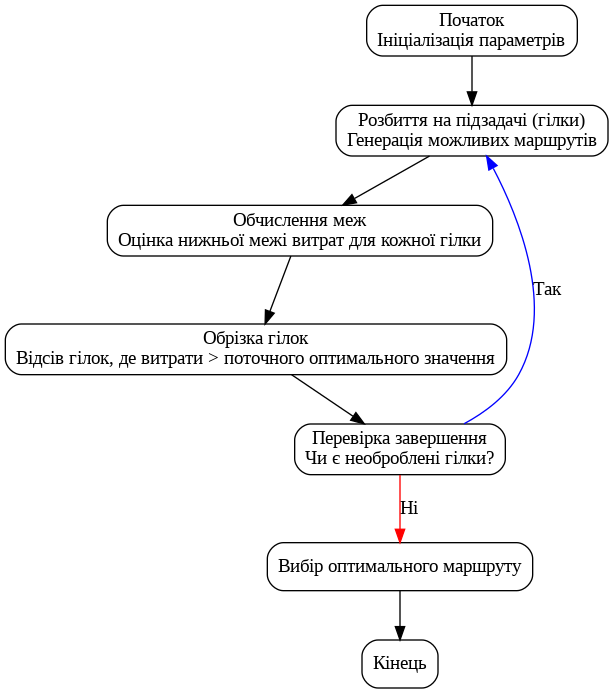
### **4. Пошук оптимального рішення**

Метод гілок та меж продовжує своє функціонування до тих пір, поки не буде знайдено оптимальне рішення. Алгоритм зберігає найкраще знайдене рішення та порівнює його з межами нових підзадач.

Коли всі можливі гілки досліджені, а відсічені неефективні маршрути, алгоритм зупиняється, і знайдене оптимальне рішення оголошується остаточним.

Розв’язком вважається отримана впорядкована послідовність вузлів, що представляють міста.

Загалом послідовність виконання алгоритму виглядає наступним чином:



## **Переваги та недоліки**

### **Переваги методу гілок та меж**

* **Оптимальність рішення**. Метод дозволяє знайти точне оптимальне рішення. Це робить його корисним для задач, де важливо досягти максимальної ефективності (наприклад, оптимізація маршрутів у логістиці).
* **Зниження обчислювальної складності**. Завдяки процесу відсічення метод відкидає неперспективні гілки, що не зможуть покращити результат. Це допомагає зменшити обсяг обчислень, які необхідно виконати, порівняно з повним перебором.
* **Гнучкість у використанні обмежень**. Метод гілок та меж дозволяє легко інтегрувати додаткові обмеження. Це корисно у задачах з реальними обмеженнями (наприклад, обмеження на витрати чи час доставки).
* **Застосування у різних сферах**. Метод широко використовується у логістиці, виробничих процесах, інформаційних технологіях та навіть у біології. Він допомагає оптимізувати не тільки маршрути, але й розподіл ресурсів, управління транспортом тощо.

### **Недоліки методу гілок та меж**

* **Висока обчислювальна складність для великих задач**. Хоча метод знижує кількість перевірених варіантів, для задач з дуже великим числом можливих рішень (наприклад, для графів з великою кількістю вузлів) він може залишатися обчислювально затратним і вимагати значних ресурсів.
* **Чутливість до вибору оцінок меж**. Ефективність методу значною мірою залежить від правильного обчислення меж (оцінки мінімальних можливих витрат). Неправильний вибір меж може призвести до перевірки багатьох неперспективних варіантів.
* **Складність реалізації**. Реалізація методу гілок та меж вимагає знання специфічних алгоритмічних технік та розуміння задачі. Це може бути ускладненням для задач з високим рівнем складності чи великою кількістю обмежень.

## **Порівняння методу гілок та меж із іншими методами розв’язання**

### **1. Точні методи**

Ці методи гарантують знаходження оптимального розв’язку, але можуть бути обчислювально затратними для великих задач.

* **Метод повного перебору**: Цей метод передбачає перевірку всіх можливих варіантів розв'язання, що гарантує знаходження оптимального результату. У випадку задачі комівояжера, потрібно перевірити всі можливі маршрути, яких у графі з n містами буде (n−1)!. Це надзвичайно трудомісткий процес, і для великих задач цей метод неефективний через експоненційний ріст варіантів. У порівнянні з ним, метод гілок та меж може бути ефективнішим, оскільки дозволяє відсіювати неперспективні маршрути на основі оцінок меж.
* **Метод динамічного програмування**: Динамічне програмування розбиває задачу на підзадачі і зберігає результати для уникнення повторних обчислень. У випадку задачі комівояжера використовується алгоритм Беллмана-Хелда-Карпа, що має складність O(n2⋅2n). Хоча цей метод оптимальний, його обчислювальна складність все ще залишається високою. Метод гілок та меж також розбиває задачу на підзадачі, але завдяки використанню оцінок меж може відкидати частини пошукового простору, знижуючи обчислювальні витрати.

### **2. Евристичні методи**

* **Жадібний алгоритм (Greedy Algorithm)**: Жадібний алгоритм починає з довільного міста та кожного разу вибирає наступне місто, яке є найближчим. Хоча цей підхід швидкий, він не завжди дає оптимальне рішення, оскільки на кожному кроці не враховує глобальної оптимальності маршруту.
* **Алгоритм найближчого сусіда (Nearest Neighbor):** Це спрощений жадібний алгоритм, який починає з певного міста, кожного разу відвідує найближче ще не відвідане місто та завершує маршрут, повертаючись до початкового міста. Цей метод швидкий і простий, але, як і жадібний алгоритм, часто не дає оптимального результату.
* **Мурашиний алгоритм**: Заснований на поведінці мурах, які відкладають феромони, прокладаючи маршрут. Під час обчислень алгоритм оцінює відстані між містами, а також «феромонний слід», що впливає на ймовірність вибору певного шляху. Алгоритм мурашиних колоній є набагато швидшим за метод гілок та меж і краще підходить для великих задач.
* **Генетичні алгоритми**: Базуються на принципах природної еволюції, генетичні алгоритми використовують популяцію можливих рішень (маршрутів), яка розвивається шляхом схрещування та мутацій. Генетичний алгоритм може бути ефективнішим для великих задач, коли методи точного обчислення (такі як гілки та межі) занадто повільні.

Евристичні методи відрізняються від методу гілок та меж тим, що вони націлені на швидке знаходження приблизного рішення, тоді як метод гілок та меж шукає оптимальне рішення, систематично перевіряючи варіанти та відсіюючи неперспективні. Евристики скорочують час, досліджуючи лише частину пошукового простору, але не гарантують точності. Натомість метод гілок та меж охоплює весь простір варіантів, тому є повільнішим, але забезпечує оптимальність, якщо розмір задачі дозволяє.

### **Переваги та недоліки методу гілок та меж у порівнянні з іншими методами**

### **Переваги:**

* **Оптимальність**: Метод гілок та меж забезпечує оптимальне рішення, на відміну від більшості евристичних методів.
* **Скорочення пошукового простору**: Відсічення підзадач із неперспективними межами суттєво зменшує обсяг необхідних обчислень у порівнянні з повним перебором.

### **Недоліки:**

* **Обчислювальна складність**: Для великих задач цей метод залишається повільним через експоненційний ріст пошукового простору, що обмежує його застосування.
* **Залежність від ефективності меж**: Результативність методу залежить від того, наскільки точно визначені межі для відсічення неперспективних гілок.

Таким чином, метод гілок та меж є ідеальним вибором для задач середнього розміру, які потребують точного розв’язання, тоді як для великих задач краще використовувати евристичні або гібридні методи.

## **Сфери застосування методу**

Метод гілок та меж є ефективним інструментом для розв’язання задачі комівояжера, що знаходить широке застосування в різних сферах, де важливо оптимізувати маршрути чи процеси з багатьма варіантами. У сучасному світі цей метод допомагає вирішувати складні задачі в таких галузях, як логістика, виробництво та складські системи, а також в інформаційних технологіях і мережах.

### **Сфера 1: Логістика та Транспортні Системи**

Метод гілок та меж широко застосовується в логістиці та транспортних системах для оптимізації маршрутів і зменшення витрат. Логістика охоплює транспортування, зберігання та обробку товарів, і задача комівояжера в цьому контексті моделює процес планування оптимальних маршрутів для транспортних засобів. Ось кілька прикладів використання цього методу в логістиці:

**Оптимізація маршрутів для доставки товарів**

Однією з головних задач є знаходження найкоротших шляхів для доставки з урахуванням дорожніх умов, вартості пального та специфічних потреб замовників. Це дозволяє зменшити час доставки, знизити витрати на паливо і мінімізувати негативний вплив на довкілля. Наприклад, метод гілок та меж допомагає уникнути заторів і критичних ділянок на маршруті, що покращує ефективність доставки для великих компаній.

**Автоматизація управління транспортними засобами**

Метод також дозволяє автоматизувати перенаправлення транспортних засобів в реальному часі, що є важливим для адаптації до змін дорожніх умов і мінімізації затримок. Сучасні системи управління транспортом (TMS) використовують ці алгоритми для динамічного перенаправлення вантажівок, що покращує ефективність і скорочує час доставки.

**Зниження операційних витрат**

Оптимізація маршрутів за допомогою методу гілок та меж дозволяє знизити витрати на пальне та технічне обслуговування, оскільки кожен зайвий кілометр призводить до додаткових витрат. Це дає можливість компаніям економити значні кошти на транспортуванні, підвищуючи загальну ефективність бізнесу.

**Поліпшення екологічної ситуації**

Оптимізовані маршрути знижують витрати пального, що також веде до зменшення викидів CO2. Це дозволяє компаніям відповідати екологічним стандартам і сприяти покращенню екологічної ситуації, що є важливим аспектом для багатьох сучасних підприємств.

**Поліпшення задоволеності клієнтів**

Швидка та точна доставка товарів без затримок підвищує рівень задоволеності клієнтів. Використання методу гілок та меж дозволяє компаніям дотримуватися обіцянок щодо термінів доставки і забезпечувати своєчасне отримання товарів клієнтами, що зміцнює їх лояльність.

Отже, метод гілок та меж є важливим інструментом для покращення логістичних процесів, зниження витрат і підвищення ефективності в транспортних системах.

### **Сфера 2: Виробничі та Складські Системи**

Метод гілок та меж має велике значення у виробничих і складських системах, де важливо оптимізувати шляхи переміщення товарів та планувати графіки роботи для персоналу та обладнання.

**Оптимізація переміщення товарів на складі**

Метод дозволяє швидко знаходити найефективніші маршрути для транспортування товарів, що зменшує час простоїв та ризик затримок у великих складах чи розподільчих центрах. Це важливо для компаній, таких як Amazon, які мають численні локації.

**Планування графіків роботи для обладнання та персоналу**

Оптимізація графіків допомагає уникнути простоїв і перевантажень, забезпечуючи плавну роботу на виробничих лініях. У таких галузях, як автомобілебудування, це критично для підтримки ефективності виробництва.

**Управління потоками матеріалів і сировини**

Метод гілок та меж допомагає оптимізувати доставку матеріалів до виробничих станцій, що зменшує витрати на транспортування та знижує ризик затримок у процесі обробки.

**Оптимізація складських операцій з робототехнікою**

Сучасні склади використовують роботів для виконання завдань. Метод гілок та меж дозволяє створювати оптимальні маршрути для роботів, що підвищує ефективність та зменшує ймовірність зіткнень.

**Зниження витрат на управління запасами**

Оптимізація переміщення товарів і розташування їх на складі дозволяє знизити витрати на зберігання та обробку, що сприяє економії часу та коштів на обслуговування складу.

Загалом, метод гілок та меж покращує ефективність виробничих і складських процесів, автоматизуючи їх і знижуючи витрати.

### **Сфера 3: Інформаційні технології та мережі**

Метод гілок та меж широко застосовується в інформаційних технологіях для оптимізації різних задач, таких як маршрутизація даних, балансування навантаження, управління ресурсами та забезпечення безпеки мереж.

**Оптимізація маршрутизації даних**

У великих мережах з численними вузлами метод допомагає визначати найефективніші маршрути для передачі даних, зменшуючи час затримки та знижуючи навантаження на основні канали зв'язку. Це особливо важливо для оптимізації інтернет-маршрутизації, де важливо зменшити навантаження на сервери та дата-центри.

**Балансування навантаження в розподілених системах**

Метод допомагає рівномірно розподіляти навантаження серед серверів у хмарних платформах або дата-центрах. Це забезпечує ефективну обробку запитів, підвищуючи ефективність використання ресурсів та стабільність роботи систем навіть за умов високих навантажень.

**Оптимізація ресурсів у хмарних обчисленнях**

У хмарних системах, де ресурси обмежені, метод гілок та меж дозволяє оптимально розподіляти обчислювальні задачі між різними сервісами, враховуючи як поточне навантаження, так і майбутні зміни в системі. Це дозволяє знижувати витрати на інфраструктуру та забезпечувати ефективну роботу додатків.

**Організація ефективного доступу до даних**

Метод застосовується для оптимізації запитів до баз даних, зменшуючи час доступу до інформації та знижуючи навантаження на систему. Це важливо у великих розподілених системах або при аналітиці великих даних, де запити можуть бути дуже складними.

**Підвищення безпеки мереж**

Метод гілок та меж також застосовується для забезпечення безпеки мереж. Наприклад, для вибору найбезпечніших маршрутів для передачі даних та оптимізації заходів шифрування, що мінімізує ймовірність перехоплення інформації.

Метод гілок та меж є потужним інструментом для оптимізації складних ІТ-операцій. Він дозволяє знижувати витрати, підвищувати ефективність систем та забезпечувати їх стабільну роботу, що особливо важливо в умовах зростання обсягів даних та складності мереж.

## **Приклад**

Розв'язати задачу комівояжера, що визначається матрицею**С**(таблиця 2.2.1).

Таблиця 2.2.1

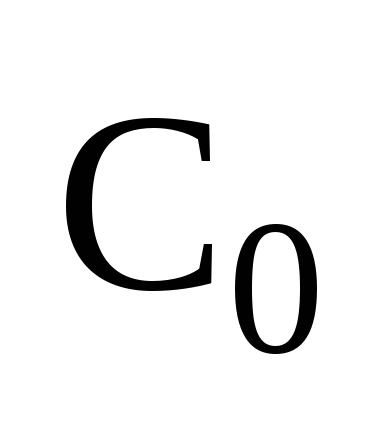
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | i i | 0 | 1 | 2 | 3 | 4 | 5 |
|  | 0 | Х | 5 | 8 | 2 | 7 | 3 |
|  | 1 | 2 | Х | 9 | 7 | 1 | 5 |
| С= | 2 | 5 | 3 | Х | 2 | 6 | 7 |
|  | 3 | 7 | 8 | 8 | Х | 2 | 10 |
|  | 4 | 9 | 3 | 7 | 9 | Х | 2 |
|  | 5 | 10 | 1 | 5 | 2 | 7 | Х |

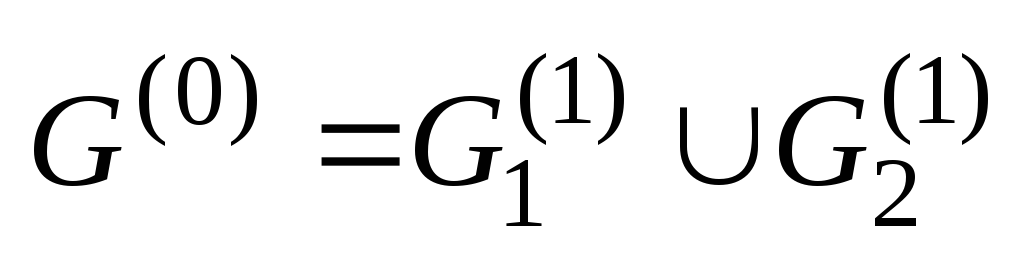
Нульовий крок. Застосувавши процедуру зведення, отримаємо матрицю **С0**(таблиця 2.2.2).

Таблиця 2.2.2

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i i | 0 | 1 | 2 | 3 | 4 | 5 | hi | | αi |
|  | 0 | Х | 3 | 2 | 0 | 5 | 1 | 2 | | 1 |
|  | 1 | 0 | Х | 4 | 6 | 0 | 4 | 1 | | 0 |
|  | 2 | 2 | 1 | Х | 0 | 4 | 5 | 2 | | 1 |
| С0= | 3 | 4 | 6 | 2 | Х | 0 | 8 | 2 | | 2 |
|  | 4 | 6 | 1 | 1 | 7 | Х | 0 | 2 | | 1 |
|  | 5 | 8 | 0 | 0 | 1 | 6 | Х | 1 | | 0 |
|  | Hj | 1 | 0 | 4 | 0 | 0 | 0 | |  | |
|  | βj | 2 | 1 | 1 | 0 | 0 | 1 | |  | |

Знаходимо оцінку .

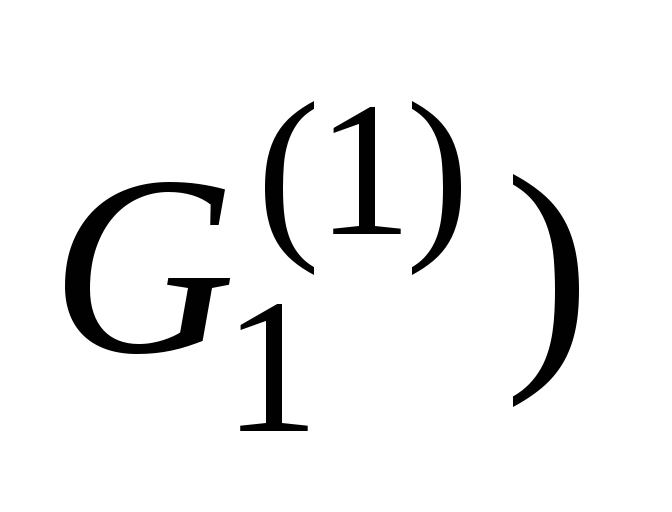
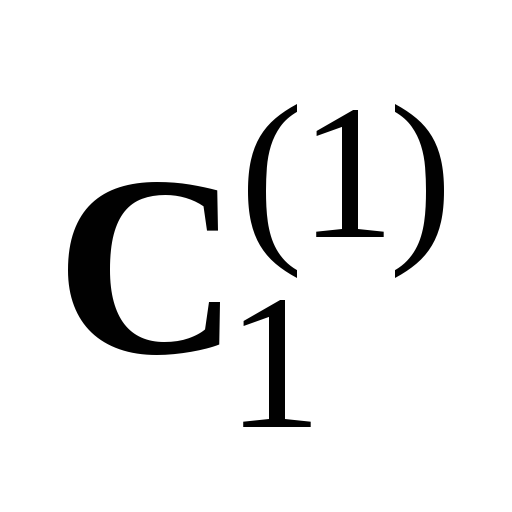
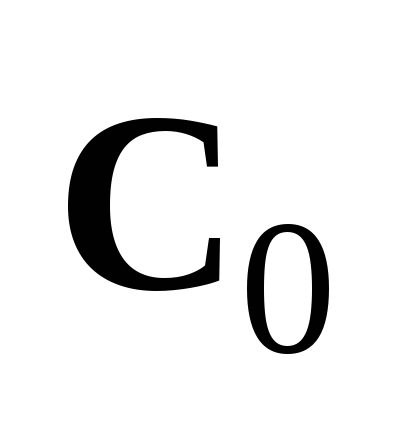
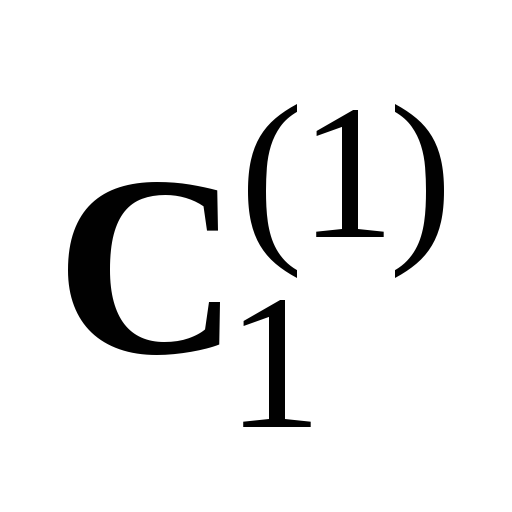
Оскільки, то вибираємо для першого розбиття пару (3;4) ((3;4)=0).

Перший крок. Проводимо розбиття , де.



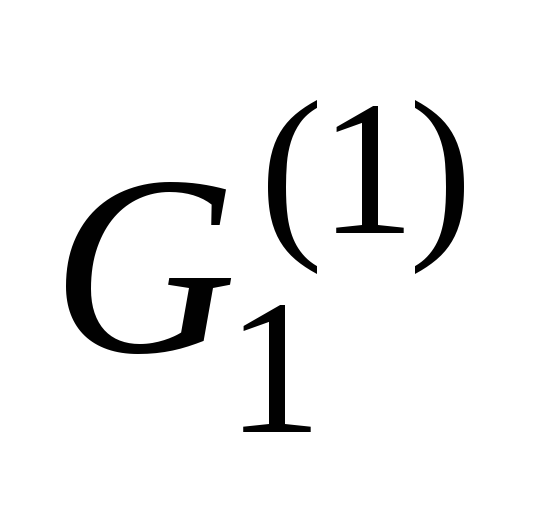


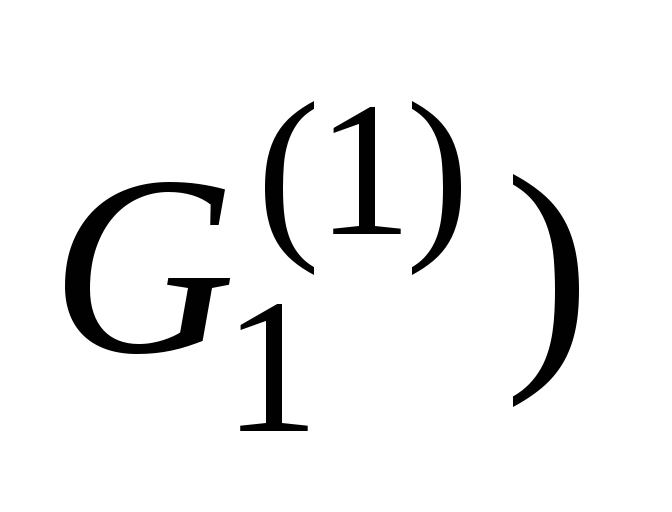
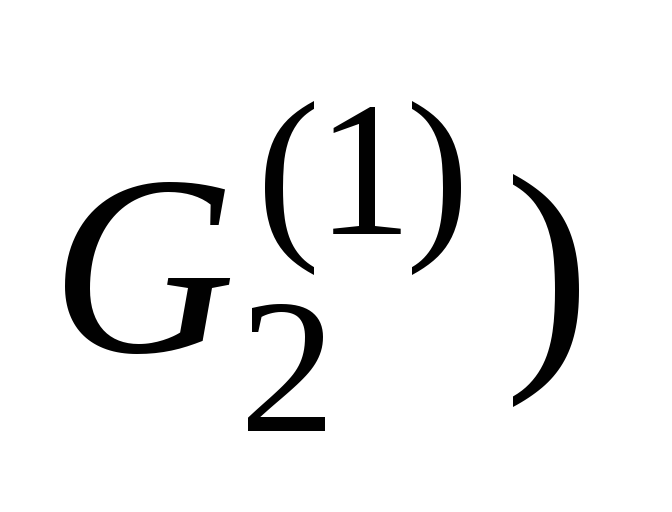
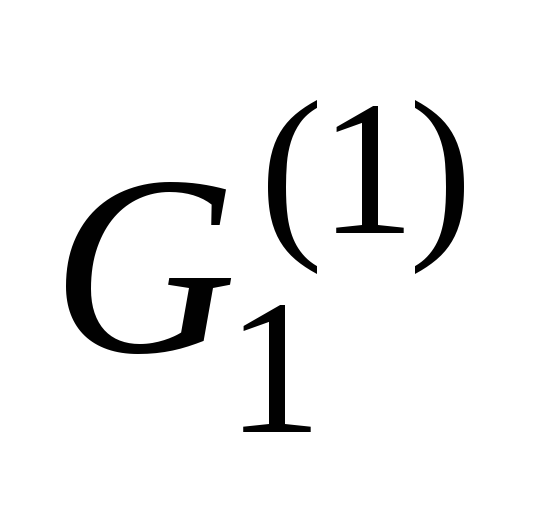
Обчислюємо оцінку 

Для обчислення (треба побудувати матрицю. Для цього в матриці викреслюємо 3-й рядок і 4-й стовпець і беручи елементC34=, здійснюємо процедуру зведення. В результаті одержуємо матрицю(òàáë.2.2.3).

Таблиця 2.2.3.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i i | 0 | 1 | 2 | 3 | 5 | Hi | | Αi |
|  | 0 | Х | 3 | 2 | 0 | 1 | 0 | | 1 |
|  | 1 | 0 | Х | 4 | 6 | 4 | 0 | | 4 |
|  | 2 | 2 | 1 | Х | 0 | 5 | 0 | | 1 |
| = | 4 | 6 | 1 | 1 | Х | 0 | 0 | | 1 |
|  | 5 | 8 | 0 | 0 | 1 | Х | 0 | | 0 |
|  | Hj | 0 | 0 | 0 | 0 | 0 | |  | |
|  | βj | 2 | 1 | 1 | 0 | 1 | |  | |

визначимо оцінку для множини :.

Оскільки (<(, то на наступному кроці розбиттю підлягає підмножина



Другий крок.



Проводимо розбиття ,де





Знаходимо оцінку 

Побудуємо матрицю (табл.2.2.4.)

Таблиця 2.2.4

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | i i | 1 | 2 | 3 | 5 | hi | | αi |
|  | 0 | Х | 2 | 0 | 1 | 0 | | 1 |
| = | 2 | 1 | Х | 0 | 5 | 0 | | 1 |
|  | 4 | 1 | 1 | Х | 0 | 0 | | 1 |
|  | 5 | 0 | 0 | 1 | Х | 0 | | 0 |
|  | Hj | 0 | 0 | 0 | 0 | |  | |
|  | βj | 1 | 1 | 0 | 1 | |  | |



обчислимо оцінку 

Оскільки , то на наступному кроці розбиттю підлягає підмножина.

Третій крок.

Оскільки ,, то вибираємо для розбиття пару (4;5)..Проводимо розбиття

де,





Знаходимо оцінку



Побудуємо матрицю.викреслимо 4-й рядок і 5-й стовпець, покладемо (таблиця 2.2.5)

Таблиця 2.2.5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | I i | 1 | 2 | 3 | hi | | αi |
|  | 0 | Х | 2 | 0 | 0 | | 2 |
| = | 2 | 1 | Х | 0 | 0 | | 1 |
|  | 5 | 0 | 0 | Х | 0 | | 0 |
|  | Hj | 0 | 0 | 0 | |  | |
|  | βj | 1 | 2 | 0 | |  | |

Знаходимо оцінку, oскільки.

Оскільки , то на наступному кроці розбиттю підлягає підмножина.

Четвертий крок.

оскільки (0;3) =2 =max(*p*,*q*).Вибираємо для розбиття пару**(0,3)**.



Знаходимо оцінку 

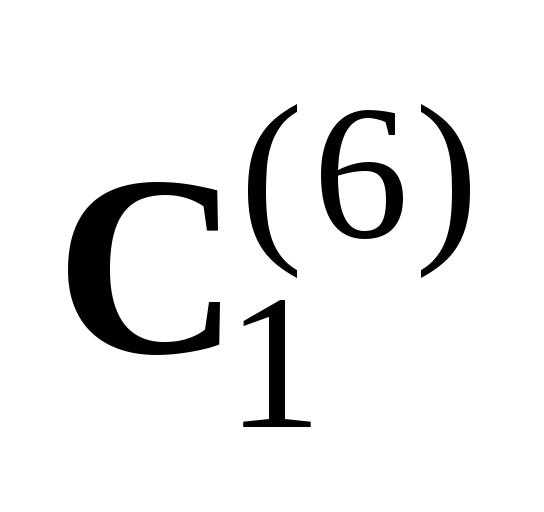
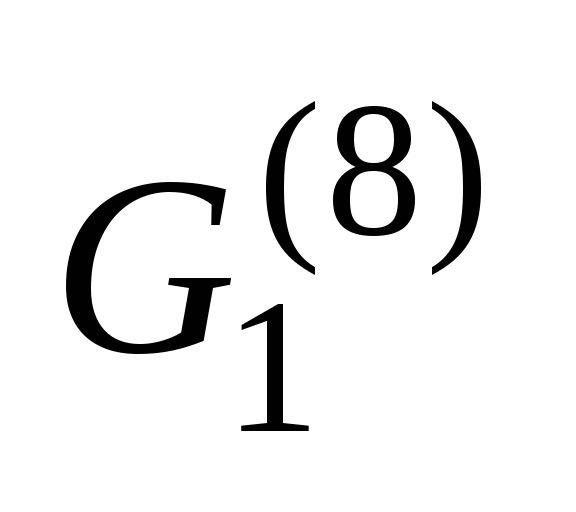
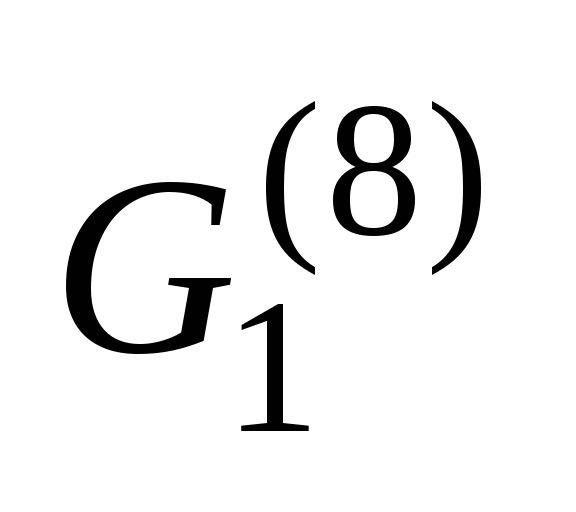
Побудуємо матрицю, викреслимо 0-й рядок і 3-й стовпець, покладемо . Виконаємо процедуру зведення(таблиця 2.2.6)

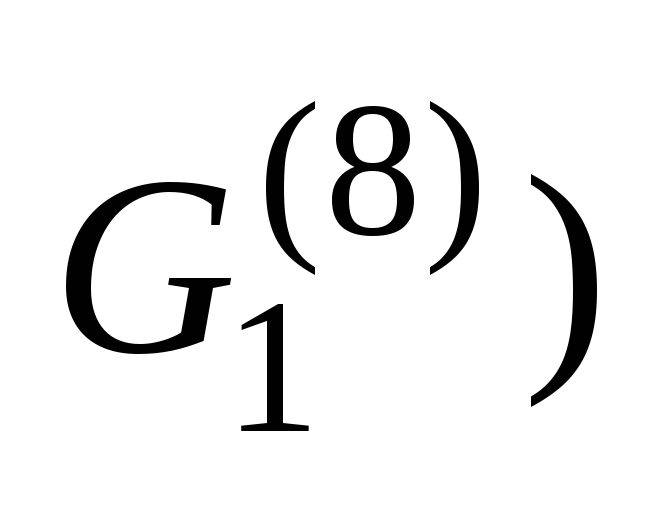
Таблиця 2.2.6

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | i i | 1 | 2 | hi | |
|  | 2 | 0 | Х | 1 | |
| = | 5 | Х | 0 | 0 | |
|  | Hj | 0 | 0 | |  |



Знаходимо оцінку .

Із матриці вибираємо дві останні пари (2;1) і (5;2). В результаті отримаємо цикл, що відповідає підмножині= {(2;1); (3;4); (1;0); (0;3); (4;5);(5.2)}. Довжина цього циклу дорівнює оцінці для множини.

У цьому можна переконатися безпосередньою перевіркою. Оцінка (є найменшою серед оцінок для всіх висячих вершин дерева розв'язків (рис.2.2.1), отже, знайдено оптимальний цикл. Процес побудови дерева розв'язків наведено на рис. 2.2.1, а знайдений мінімальний цикл — на рис. 2.2.2.

