МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
"КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ"

Індивідуальне завдання
курсу “Логічне програмування” НаУКМА

Вирішення задач на переливання застосовуючи Prolog

Роботу виконала
студентка 3 року навчання
спеціальності “Комп’ютерні науки”
факультету інформатики
Національного університету “Києво-Могилянська академія”
Шулакова Катерина

Київ-2025

1. Постановка задач

Задачі на переливання – це класичні головоломки, які полягають у переміщенні рідини між декількома ємностями з різною ємністю з метою досягнення заданого об’єму рідини в одній із них. Задача має такі основні особливості:

Параметр	**Задача з джерелом та зливом**	**Задача без джерела та зливу**
На вході задається	- **Набір ємностей**, де кожну описуємо структурою bucket (ідентифікатор, максимальна ємність, початкова кількість рідини). - **Спеціальні ємності**: • Джерело bucket(0, 0, _) – безкінечне джерело, яке завжди наповнює до максимальної ємності приймаючої. • Злив bucket(-1, -1, _) – безкінечний злив, в який можна виливати лише всю рідину з ємності. - **Ціль**: заданою величиною GoalAmount (кількість рідини, яку потрібно отримати в одній з звичайних ємностей).	- **Набір ємностей**, кожна з яких описується структурою bucket (ідентифікатор, максимальна ємність, початкова кількість рідини). - **Ціль**: визначений цільовий стан (GoalState), який задає, що в одній з ємностей має бути певна кількість рідини (наприклад, bucket(_, _, бажана_кількість)).
Операції переливання	- Рідина переливається з безкінечного джерела до звичайної ємності до її повного заповнення. - Рідина переливається з ємності до безкінечного зливу, при цьому ємність повністю спорожнюється. - Рідина переливається між ємностями, причому переливання триває до повного спорожнення ємності-джерела або до повного заповнення приймаючої.	- Рідина переливається за правилами, де переливання триває до повного спорожнення ємності-джерела або до повного заповнення приймаючої. Додаткові спеціальні ємності (джерело, злив) не використовуються.
Цільовий стан	- В одній із звичайних ємностей має бути рівно GoalAmount літрів рідини. - При перевірці цільового стану спеціальні ємності (джерело та злив) ігноруються.	- В одній із ємностей має бути задана кількість рідини, як визначено в GoalState.

2. Мови програмування та особливості Prolog

Задачу на переливання рідин можна реалізувати на різних мовах програмування, таких як C/C++, Java, Python, Haskell тощо. Проте існують специфічні особливості, які роблять Prolog ідеальним кандидатом для подібних задач:

- **Природна підтримка логічного висновку та рекурсії.**
  Prolog заснований на декларативному підході, де описується логіка задачі через факти і правила. Це дозволяє сформулювати просте визначення станів і операцій (переливань), а також використовувати механізм рекурсії для обходу простору можливих станів. Такий підхід значно спрощує розробку алгоритму, оскільки сам процес пошуку рішення реалізується завдяки вбудованим механізмам мови.

- **Пошук з використанням backtracking.**

  Програми на Prolog автоматично використовують механізм відкату (backtracking) при невдалих спробах переходу до наступного стану. Це дозволяє ефективно досліджувати простір усіх можливих рішень, уникаючи повторного перебору вже відвіданих станів. Завдяки цьому зменшується складність коду, оскільки алгоритм не вимагає додаткової реалізації перевірок на зациклення.

- **Інтуїтивність опису задачі.**
  Завдяки можливості описувати факти і правила в дуже природній формі, Prolog дозволяє безпосередньо відобразити умови задачі (наприклад, правила переливання між ємностями) у вигляді логічних тверджень. Це значно полегшує налагодження і розуміння програми.

Незважаючи на всі переваги, варто враховувати деякі ризики:

- **Комбінаційна експлозія:** У задачах з великим числом можливих станів може виникнути експоненціальне зростання кількості варіантів, що значно впливає на час виконання.
- **Оптимальність рішень:** Використання стандартного пошуку (наприклад, глибинного перебору) не гарантує знаходження оптимального за кількістю кроків рішення. Для цього може знадобитися модифікація алгоритму або використання альтернативних методів пошуку, таких як пошук у ширину чи евристичні алгоритми.
- **Ускладнене налагодження:** Логічні програми, зокрема ті, що використовують backtracking, інколи важко налагоджувати через нетрадиційний підхід до керування потоком виконання.

Ці аспекти роблять Prolog ефективним інструментом для вирішення задач на переливання, однак вимагають уважного підходу до оптимізації та управління простором станів.

3. Розв’язання задач

Розв’язання задач на переливання рідини ґрунтується на представленні кожного стану системи як списку ємностей, де кожна ємність описується структурою, що містить її ідентифікатор, максимальну місткість та поточний об’єм рідини. Алгоритм пошуку рішення реалізовано у вигляді рекурсивного процесу з використанням механізму backtracking, що дозволяє відслідковувати вже відвідані стани та уникати зациклення.

3.1. Базова версія – задача лише з 3 ємностями

Постановка задачі:

На вхід задається список із трьох звичайних ємностей. Наприклад, ми маємо такі дані:

Ємності представлені як `[bucket(1, 12, 8), bucket(2, 8, 0), bucket(3, 5, 0)]` де `bucket(1, 12, 8)` означає, що перша ємність має максимальну місткість 12 літрів і зараз містить 8 літрів рідини.

Ціль: досягти такого стану, в якому хоча б одна з ємностей містить задану кількість рідини (наприклад, 3 літри), що визначається як `bucket(_, _, 3)`.

Алгоритм розв’язання може виглядати так:

1. **Ініціалізація стану**
   На першому етапі формується початковий стан задачі, що є списком ємностей із їхніми початковими значеннями. Цей стан слугує відправною точкою для подальшого пошуку рішення.
2. **Визначення операцій переливання**
   Створюються правила, які описують можливі операції переливання між двома ємностями. За умовою задачі переливання триває до того моменту, коли або ємність, з якої переливають, стає порожньою, або приймаюча ємність заповнюється до своєї максимальної місткості.

3. **Рекурсивний пошук та механізм backtracking**
    - На кожному кроці алгоритм обирає пару ємностей і застосовує відповідне правило переливання, отримуючи новий стан.
    - Після кожного переходу перевіряється, чи досягнуто цільового стану (тобто, чи містить хоча б одна з ємностей заданий об’єм рідини).
    - Якщо отриманий стан вже був відвіданий або не веде до досягнення цілі, алгоритм відкидає цей шлях і повертається назад (backtracking), щоб спробувати інший варіант переливання.
    - Під час процесу виводяться всі проміжні стани, що дозволяє відслідковувати послідовність операцій.
4. **Виведення кроків та результату**

    Послідовність переходів від початкового стану до фінального, де застосовано як стандартні, так і розширені операції, виводиться для детального аналізу послідовності дій.

3.2. Розширена версія – задача із джерелом та зливом

У розширеній версії до звичайних ємностей додаються дві спеціальні:

- **Джерело,** яке позначається як `bucket(0, 0, _)` і символізує безкінечне джерело рідини, яке завжди може забезпечити приймаючу ємність необхідною кількістю рідини до її максимального значення.
- **Злив,** що позначається як `bucket(-1, -1, _)` і представляє безкінечний злив, що дозволяє повністю спорожнити будь-яку ємність.

При цьому цільова умова залишається незмінною – потрібно отримати задану кількість рідини в одній із звичайних ємностей. Спеціальні ємності використовуються лише для розширення можливостей переливання і не враховуються при перевірці фінального стану.

Алгоритм розв’язання

1. **Ініціалізація розширеного стану**

   Формується список ємностей, до якого додаються записи про джерело та злив. Це дозволяє алгоритму враховувати додаткові операції з поповнення та спорожнення.
2. **Розширення правил переливання**
   - Окрім стандартних операцій між звичайними ємностями, вводяться нові операції:
     - **Переливання з джерела:** При застосуванні цього правила приймаюча ємність заповнюється до свого максимального значення, оскільки джерело є безкінечним.
     - **Переливання до зливу:** Це правило дозволяє повністю спорожнити ємність, виливаючи всю рідину до безкінечного зливу.
   - Ці правила інтегруються в загальний алгоритм, що дозволяє використовувати як стандартні, так і розширені операції переливання.
3. **Пошук рішення із розширеними можливостями**

   Алгоритм залишається рекурсивним і використовує механізм backtracking, проте під час перевірки фінального стану спеціальні ємності ігноруються, щоб враховувати лише звичайні ємності, в яких має бути досягнуто заданий об’єм рідини.
4. **Виведення кроків рішення**

   Послідовність переходів від початкового стану до фінального, де застосовано як стандартні, так і розширені операції, виводиться для детального аналізу послідовності дій.

Таким чином, початковий підхід із трьома звичайними ємностями дозволяє зрозуміти базову логіку роботи алгоритму, а розширення коду для задачі з джерелом і зливом демонструє, як незначними доповненнями можна значно

збільшити функціональність системи та її здатність моделювати більш реалістичні сценарії.

4. Альтернативні варіанти вирішення

Це алгоритм використовує пошук у просторі станів із рекурсією та backtracking. Існують також альтернативні підходи, які можуть бути використані залежно від специфічних вимог задачі, таких як оптимальність рішення або обмеження ресурсів.

Пошук у ширину (BFS)
У цьому підході дослідження простору станів здійснюється рівень за рівнем.

+ Забезпечує знаходження найкоротшого шляху до цільового стану, оскільки кожен можливий хід досліджується систематично.
- Може вимагати значно більше оперативної пам’яті, особливо якщо простір станів є великим.

Евристичні алгоритми (наприклад, A*)
Додавання евристичної функції, що оцінює "відстань" від поточного стану до цілі, дозволяє спрямовувати пошук у більш перспективні області простору станів.

+ Може суттєво скоротити кількість перевірених станів, що прискорює знаходження рішення.
- Ефективність алгоритму сильно залежить від правильно обраної евристики; невдалий вибір може призвести до неефективного пошуку.

Комбіновані підходи
Інколи доцільно використовувати гібридні методи, що інтегрують переваги декількох алгоритмів. Наприклад, можна спочатку застосувати BFS для попередньої оцінки оптимального шляху, а потім за допомогою DFS з backtracking провести детальний аналіз варіантів.

+ Забезпечують баланс між точністю пошуку і витратами ресурсів.

- Реалізація таких методів може бути складнішою і вимагати додаткових налаштувань.

Застосування альтернативних алгоритмів може бути особливо корисним у випадках, коли простір станів надто великий для простого рекурсивного пошуку, або коли важливо знайти не просто будь-яке рішення, а оптимальне за певним критерієм (наприклад, з мінімальною кількістю кроків).

5. Реальне застосування подібних задач

Задачі на переливання рідини мають не лише навчальну цінність, але й знаходять практичне застосування в різних галузях.

Навчальний аспект

Демонстрація алгоритмічних методів
Задачі на переливання є класичним прикладом для вивчення методів пошуку у просторі станів, рекурсії та backtracking.

Логічне програмування
Реалізація подібних задач на Prolog демонструє переваги декларативного підходу, де важливі не кроки виконання програми, а опис логічних залежностей і правил.

Практичне застосування

Розподіл ресурсів
Моделі, що лежать в основі задач на переливання, застосовуються для оптимального розподілу обмежених ресурсів. Наприклад, у виробництві чи енергетиці такі підходи можуть використовуватись для визначення оптимального розподілу енергії або матеріалів між різними системами.

Автоматизація процесів
У системах автоматичного управління подібні алгоритми допомагають регулювати потоки рідин, газів або інших ресурсів, що є важливим у технологічних процесах.

Планування та логістика
Принципи, закладені в задачах на переливання, можуть використовуватись для моделювання та оптимізації логістичних процесів, таких як маршрутизація, планування запасів і розподіл вантажів.