# Теоретичні відомості

1. **Загальні відомості про поставлену задачу**

Задача про 8 ферзів була запропонована німецьким гравцем у шахи Максом Беззелом у 1848 році. Вона полягає у розміщенні восьми ферзів на шаховій дошці, розміром 8 × 8 у взаємно-неатакуючому розміщенні. Іншими словами, потрібно щоб фігури не були у одному рядку, одному стовпці та на одній діагоналі.

1. **Вирішення проблем способом пошуку**

Представлений спосіб здійснюється за допомогою агентів, які вирішують задачі. Такі агенти здатні визначити подальші кроки, завдяки знаходженню послідовності дій, які призводять до бажаних станів. Робота агента поділяється на декілька етапів:

* Формулювання цілі: задання кінцевого стану системи, при якому задача є вирішеною (усіх ферзів розставлено на відповідних позиціях)
* Формулювання задачі: процес визначення того, які дії та стани потрібно буде розглянути для досягнення цілі.
* Пошук: процес дослідження різних варіантів послідовностей дій, які ведуть до цільового стану та вибір найкращої послідовності.
* Виконання: викнання зайденої послідовності.

**Пошук в ширину (Breath First Search)**

Цей алгоритм побудований на простій стратегії, де спочатку розгортається та відвідується кореневий вузол дерева пошуку, а після – усі його нащадки. Так продовжується доти, доки вузли мають дочірні елементи.

Структура даних, яка використовується для цього алгоритму – черга, що працює за принципом FIFO (First-In-First-Out). Це гарантує розгортання в першу чергу тих вузлів, які були відвідані першими.

Складність алгоритму можна оцінити як , де – це коефіцієнт розгалуження, а – це глибина, на якій знаходиться цільовий вузол, що знаходиться найближче до поверхні.

Головним недоліком пошуку в ширину є необхідність зберігання оброблених вузлів у пам'яті. Також цей алгоритм є досить часозатратним, якщо шуканий стан знаходиться досить далеко від початкового.

Псевдокод пошуку в ширину:

Вхідні дані: початковий стан *node*, шуканий стан *goal.*

Вихідні дані: шлях від *node* до *goal*.

**BFS (node, goal)**

Нехай Q – черга

Позначити root як пройдений

Додати root до Q

**Повторити** поки Q не порожня

v: = останній елемент черги

**Якщо** v == goal, то

Повернути шлях від node до v

**Все якщо**

**Повторити** для усіх ребер від v до w, де вершина v суміжна до w

**Якщо** w не є дослідженою, то

Позначити w як досліджену

Додати w до Q

**Все якщо**

**Все повторити**

**Все повторити**

 Cormen, Thomas H. "22.2 Breadth-first search". *[Introduction to algorithms](http://worldcat.org/oclc/1006880283)*. *[ISBN](https://en.wikipedia.org/wiki/ISBN_(identifier)" \o "ISBN (identifier))* *[978-81-203-4007-7](https://en.wikipedia.org/wiki/Special:BookSources/978-81-203-4007-7" \o "Special:BookSources/978-81-203-4007-7)*. *[OCLC](https://en.wikipedia.org/wiki/OCLC_(identifier)" \o "OCLC (identifier))* *[1006880283](https://www.worldcat.org/oclc/1006880283)*.

**https://en.wikipedia.org/wiki/Breadth-first\_search#cite\_note-7**

**Пошук із обмеженням глибини (Limited Depth First Search)**

Якщо при розв’язанні задачі виявляється, що дерево пошуку є необмеженим, то вводиться зарання визначене обмеження глибини . Тобто, вузли, що знаходяться на цій глибині розглядаються як листя дерева пошуку. Обмеження глибини вирішує проблему нескінченного шляху. Якщо зарання невідома, то потрібно вводити додаткове джерело неповноти, адже може виникнути ситуація, що (шуканий вузол знаходиться нижче глибини пошуку). Неоптимальним також буде вибір значення , такого що .

Часова складність такого пошуку - .

Просторова складність - .

Пошук із обмеженням глибини може призводити до невдач, які є двох типів: **failure** – відсутнє рішення, **cutoff** – на заданій глибині рішення немає.

Псевдокод пошуку із обмеженням глибини:

Вхідні дані: початковий стан *node*, шуканий стан *goal,* обмеження глибини *l.*

Вихідні дані: шлях від *node* до *goal* або індикатори невдачі *cutoff* / *failure*

**Функція** LDFS (node, goal, l) **повертає** result (шлях від *node* до *goal*) **або** failure/ cutoff (індикатор невдачі)

**Якщо** (!cutoff\_occured)

**Якщо** ( l >= 0) **то**

**Якщо** (node == goal) **то**

**Повернути** шлях від *node* до *goal*

**Інакше якщо (**Depth[node] == l) **то**

**Повернути** cutoff

**Інакше**

**Повторити** для усіх вершин v суміжних до node

result = **LDFS (v, goal, l - 1)**

**Якщо** (result = cutoff)

cutoff\_occured = true

**Інакше якщо** (result != failure)

**Повернути** result

**Все якщо**

**Все повторити**

**Все якщо**

**Все якщо**

**Все якщо**

**Якщо** (cutoff\_occured) **то**

**Повернути** cutoff

**Інакше**

**Повернути** failure

**Все якщо**

[**https://enacademic.com/dic.nsf/enwiki/446185**](https://enacademic.com/dic.nsf/enwiki/446185) **(псевдокод)**

**Пошук в глибину із ітеративним заглибленням (IDS)**

Пошук в глибину із ітеративним заглибленням має переваги за умови, якщо простір пошуку є досить великим, а глибина, на якому знаходиться рішення, не відома зарання.

Стратегія даного пошуку полягає у послідовному збільшені обмеження доти, доки не буде знайдений шуканий результат. При цьому пошуку, розгортаються лише ті вузли, які знаходяться на заданому обмеженні. На кожній наступній ітерації розгортання вузлів відбувається знову.

Цей метод включає у себе переваги пошуку у глибину та пошуку у ширину. Повнота пошуку досягається при скінченному , чим IDS завдячує BFS. Подібно до DFS, IDS потребує досить невеликий ресурс пам’яті, який можна визначити як .

KORF, Richard E. (1985). ["Depth-first iterative deepening"](https://cse.sc.edu/~mgv/csce580f09/gradPres/korf_IDAStar_1985.pdf)

Псевдокод пошуку із обмеженням глибини :

Вхідні дані: початковий стан *node*, шуканий стан *goal*

Вихідні дані: шлях від *node* до *goal* індикатор невдачі *failure*

**Функція IDS(node, goal)**

**Повторити для** l = 0 до

result = **LDFS(node, goal, l)**

**Якщо** (result != cutoff) **то**

**Повернути** result

**Все якщо**

**Все повторити**