Міністерство освіти і науки, молоді та спорту України

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Кафедра АСОІУ

ЗВІТ

про виконання лабораторної роботи № 1

з дисципліни

«Основи штучного інтелекту»

Варіант 4

|  |  |
| --- | --- |
| Перевірила:  Ст. вик. Мажара О.О. | Виконала:  Студентка групи ІС-71  Вознюк Олександра |
|  |  |

Київ 2020

1. **Мета роботи**

Ознайомитися з методами неінформативного пошуку та розробити алгоритм на їх основі для вирішення задачі згідно варіанту.

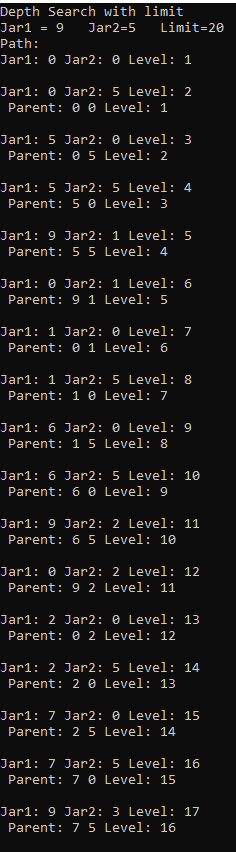
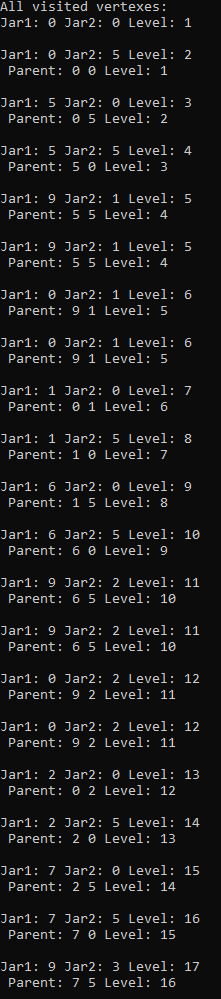
1. **Завдання до роботи**

Розробити алгоритм вирішення старовинної логічної задачі: "Як за допомогою 5-ти літрового і 9-ти літрового відра набрати з річки 3 літри води?"

За допомогою пошуку в глибину з обмеженням глибини

1. **Текст розробленого програмного забезпечення з коментарями.**

<https://github.com/oleksandravozniuk/ArtificialIntelligence>

1. **Результати роботи програмного забезпечення, що включають результати тестування та копії екранних форм.**  
2. **Відповіді на контрольні запитання**

* Напишіть на псевдомові процедури пошуку в ширину і глибину, поясніть їх відмінність з алгоритмічної точки зору

Пошук у глибину – завжди розгортається найглибший вузол, що далі вже не має спадкоємців, в поточній периферії дерева.

**function** Tree-Search (problem, LIFO-Queue) **returns** решение solution

или индикатор неудачи failure

LIFO-Queue Insert (Make-Node (Initial-State [problem]), LIFO-Queue)

**loop do**

**if** Empty? (LIFO-Queue) **then return** индикатор неудачи failure

node Remove-First (LIFO-Queue)

**if** Goal-Test [problem] применительно к State [node]

завершается успешно

**then return** Solution (node)

LIFO-Queue Insert-All (Expand (node, problem), LIFO-Queue)

**function** Expand (node, problem) **returns** множество узлов successors

successors пустое множество

**for each** <action,result> **in** Successor-Fn[problem] (State [node]) **do**

s новый узел

State [s] result

Parent-Node [s] node

Action [s] action

Path-Cost [s] Path-Cost [node] + Step-Cost (node, action, s)

Depth [s] Depth [node] + 1

Добавить узел s к множеству successors

**return** successors

Пошук в ширину – розгортаються спочатку всі спадкоємці кореня, потім всі спадкоємці тих спадкоємців.

**function** Tree-Search (problem, FIFO-Queue) **returns** решение solution

или индикатор неудачи failure

FIFO-Queue Insert (Make-Node (Initial-State [problem]), FIFO-Queue)

**loop do**

**if** Empty? (FIFO-Queue) **then return** индикатор неудачи failure

node Remove-First (FIFO-Queue)

**if** Goal-Test [problem] применительно к State [node]

завершается успешно

**then return** Solution (node)

FIFO-Queue Insert-All (Expand (node, problem), FIFO-Queue)

**function** Expand (node, problem) **returns** множество узлов successors

successors пустое множество

**for each** <action,result> **in** Successor-Fn[problem] (State [node]) **do**

s новый узел

State [s] result

Parent-Node [s] node

Action [s] action

Path-Cost [s] Path-Cost [node] + Step-Cost (node, action, s)

Depth [s] Depth [node] + 1

Добавить узел s к множеству successors

**return** successors

* Сформулюйте принципи пошуку, використовувані в алгоритмах пошуку в глибину:  з обмеженням глибини і з ітеративним поглибленням.

Пошук в глибину з обмеженням глибини має границю глибини l. Це означає, що вузли на глибині l розглядаються як вузли, що не мають спадкоємців. Використання ліміту глибини дозволяє вирішити проблему безкінечного шляху.

Пошук з ітеративним поглибленням являє собою спільну стратегію, що часто використовується разом с пошуком у глибину, що допомагає знайти найкращий ліміт глибини. Це досягається шляхом покрокового збільшення ліміта до тих пір, поки не буде знайдена ціль.

* Які алгоритми неінформативного пошуку є повними? Оптимальними?
* Пошук в ширину – повний. Якщо самий поверхневий цільовий вузол знаходиться на деякій кінцевій глибині d, то пошук в ширину рано чи пізно знайде відповідь (при умові, що коефіцієнт гіллястості b є кінцевим). Самий поверхневий цільовий вузол не обов’язково є оптимальним; формально пошук в ширину буде оптимальним, якщо вартість шляху виражається у вигляді неспадної функції глибини вузла.
* Пошук в глибину з обмеженням глибини буде оптимальним при l<=d, де l – ліміт глибини, d – глибина самого поверхневого цільового вузла.
* Пошук з ітеративним заглибленням, якщо коефіцієнт гіллястості скінченний і оптимальний, якщо вартість шляху являє собою неспадну функцію глибини вузла.
* Двонаправлений пошук – повний і оптимальний

1. **Висновки, що відображають особисто отримані результати виконання роботи, їх критичний аналіз.**

Метод пошуку в глибину з обмеженням на глибину не дає оптимального результату, якщо поставити ліміт, на глибині якого не існує вирішення задачі. Якщо не контролювати виділення пам’яті для породження наступних станів може статися її переповнення. Особисто отримані мною результати показують, що метод призводить до правильного рішення. Але в отриманих мною результатах ви можете побачити що в списку усіх відвіданих вершин деякі вершини повторюються. Думаю, це потрібно розуміти як те, що якщо пошук в глибину потрапляє в стан, що вже є раніше відвіданим, то ми починаємо рухатися не вглиб, а горизонтально по одному рівню перебираючи дітей доки не знайдемо підходящий вузол або завершимо роботу.