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 № 5

з дисципліни

«Основи штучного інтелекту»

Варіант 4

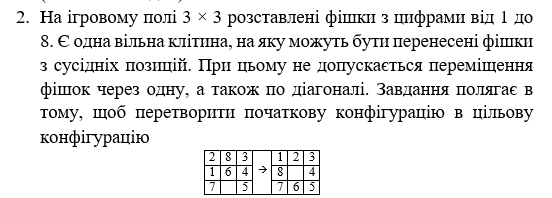
|  |  |
| --- | --- |
| Перевірила:  Ст. вик. Мажара О.О. | Виконала:  Студентка групи ІС-71  Вознюк Олександра |
|  |  |

Київ 2020

1. **Мета роботи**

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

1. **Завдання до роботи**

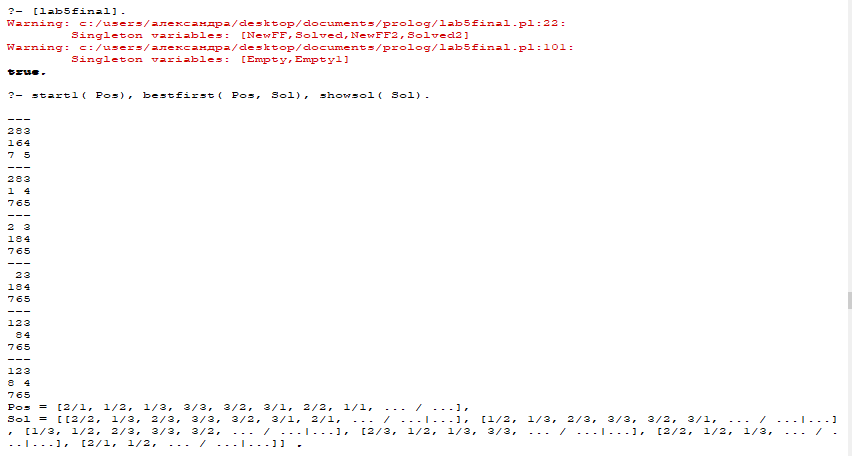


RBFS – Рекурсивний пошук за першим найкращим збігом

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

<https://github.com/oleksandravozniuk/ArtificialIntelligence/tree/master/AI4>

1. **Результати роботи програмного забезпечення, що включають результати тестування та копії екранних форм.**



1. **Відповіді на контрольні запитання**

* Напишіть на псевдомові процедури пошуку A\*, RBFS

**RBFS**

**function** Recursive-Best-First-Search (problem) **return** решение result

или индикатор неудачи failure

RBFS (problem, Make-Node (Initial-State[problem]),)

**Function** RBFS (problem, node, f\_limit) **return** решение result

или индикатор неудачи failure и новый предел f-стоимости f\_limit

**if** Goal-Test [problem] (State [node]) **then return** узел node

successors Expand(node, problem)

**if** множество узлов-преемников successors пусто

**then return** failure,

**for each** s in successors **do**

f[s] max(g(s)+h(s), f[node])

**repeat**

best узел с наименьшим f-значением в множестве successors

**if** f[best]>f\_limit **then return** failure, f[best]

alternative второе после наименьшего f-значение в множестве successors

result, f[best] RBFS(problem, best, min(f\_limit, alternative))

**if** result failure **then return** result

**A\***

bool A\*(start, goal):

U = ∅ //множество рассмотренных вершин

Q = ∅ //множество вершин, которые требуется рассмотреть

Q.push(start)

g[start] = 0 //стоимость пути от начальной вершины до x

f[start] = g[start] + h(start) //значение эвристической функции "расстояние + стоимость" для вершины x

while Q.size() != 0

current = вершина из Q с минимальным значением f

if current == goal

return true // нашли путь до нужной вершины

Q.remove(current)

U.push(current)

for v : смежные с current вершины

tentativeScore = g[current] + d(current, v) // d(current, v) — стоимость пути между current и v

if v∈U and tentativeScore >= g[v]

continue

if v∉U or tentativeScore < g[v]

parent[v] = current

g[v] = tentativeScore

f[v] = g[v] + h(v)

if v∉Q

Q.push(v)

return false

* Яка відмінність між алгоритмами інформативного та неінформативного пошуку?

Неінформативні стратегії пошуку дозволяють знаходити рішення задач шляхом систематичного виробництва нових станів і їх перевірки на досягнення результату. На жаль, в більшості випадків ці стратегії є вкрай неефективні. Оскільки в інформативній стратегії пошуку використовуються знання, що відносяться до конкретної задачі, вона забезпечує більш ефективний пошук рішення.

* Поясніть необхідність використання евристичних функцій при інформативному пошуку.

Тому що евристичні функції представляють собою найбільш спільну форму, в якій до алгоритму пошуку підключаються додаткові знання про задачу, а суть інформативного пошуку як раз в тому, щоб використовувати знання, що відносяться до певної задачі.

* Якою повинна бути евристична функція для того, щоб алгоритми інформативного пошуку були оптимальними?
* Ефективний коефіцієнт розгалуження b\*

N+1=1+b\* + (b\*)^2 + … + (b\*)^d

Де N – загальна кількість вузлів, що створена в процесі пошуку A\* рішення конкретної задачі

d – глибина рішення

Добре спроектована евристична функція повинна мати b\*

* Які алгоритми інформативного пошуку є повними? Оптимальними? Чому?
* A\* є повним, оптимальним та оптимально ефективним.

Оптимально ефективний тому що даний алгоритм не гарантує розгортання меншої кількості вузлів ніж в пошуку А\* за допомогою якого-небудь іншого оптимального алгоритму.

Повний тому що по мірі додавання полос зі зростаючими значеннями f ми повинні в кінцевому результаті досягти полоси, де значення f буде дорівнювати вартості шляху до цільового стану.

Оптимальним, тому що інтуїтивно зрозуміло, що перше знайдене рішення повинно бути оптимальним, оскільки, цільові вузли у всіх наступним контурах будуть мати більш високе значення f-вартості і тому більш високе значення g-вартості (оскільки усі цільові вузли мають значення h(n)=0).

* RBFS оптимальний, якщо евристична функція h(n) допустима і повний так як не йде по безкінечному шляху і має можливість йти по альтернативним шляхам.
* SMA\* є повним, якщо існує яке-небудь рішення, яке можна досягти і оптимальним, якщо існує яке-небудь оптимальне рішення, якого можна досягнути
* Поясніть принцип роботи алгоритмів, які реалізовані у лабораторній роботі.

A\* - порядок обходу вершин визначається евристичною функцією “відстань+вартість”. А\* покроково проглядає усі шляхи, що ведуть від початкової вершини в кінцеву, поки не знайде мінімальний. Алгоритм спочатку проглядає ті маршрути, котрі “здаються” такими, що ведуть до цілі. При виборі вершини А\* враховує також увесь пройдений шлях до неї шлях.

RBFS(реалізовано) – рекурсивний алгоритм, який намагається імітувати роботу стандартного пошуку по першому найліпшому збігу, але з використанням тільки лінійного простору. Він має таку ж структуру, що і пошук у глибину, але замість безкінечного прямування вниз по поточному шляху даний алгоритм контролює f-значення найкращого альтернативного шляху, що доступний з предка поточного вузла. Якщо поточний вузол більший ніж даний ліміт, то поточний етап рекурсії скасовується і рекурсія продовжується з альтернативного шляху.

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

При розробці алгоритму рекурсивного пошуку за першим найкращим збігом була взята h = d + s\*3 (d – манхетенська відстань між клітинками, вимірюється як сума відстаней між двома клітинками в горизонтальному та вертикальному напрямах. s – оцінка упорядкованності) як евристичну функцію. Ця евристична функція ефективно направляє пошук до цілі, але її недоліком є те, що вона не є допустима: вона не гарантує, що найкоротший шлях вирішення завжди буде знайдений до виявлення якогось більш довгого рішення. Ця функція не задовільняє умовам допустимості при якому h<=h\* для усіх вузлів. В загальному алгоритм є достатньо ефективний, але часто доводиться повторно формувати вузли, оскільки алгоритм може змінювати рішення щодо вибраного шляху і переключатися на альтернативні шляхи.