Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

Лабораторна робота № 4

З дисципліни «Основи штучного інтелекту»

на тему: “Неінформативний пошук”

Варіант - 13

Виконав студент гр. ІС-72

Кривохижа Р. А.

Перевірила ст. вик. каф. АСОІУ

Мажара О. О

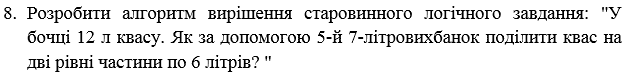
Київ 2020

**Мета:** дослідити роботу алгоритму неінформативного пошуку на мові програмування Prolog.

**Варіант:**



Застосовуємо пошук у ширину.



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

:- dynamic jars/2.

solve(jar(VolumeA, A), jar(VolumeB, B), jar(VolumeC, C), target(Target), Actions):-

retractall(jars(\_, \_, \_)),

bfs(volumes(VolumeA, VolumeB, VolumeC), target(Target), [state(jars(A, B, C), actions([init]))], Actions).

bfs(\_Volumes, target(Target), [state(jars(A, B, C), Actions)|\_Other], Actions):-

((A = Target, B = Target); (A = Target, C = Target); (B = Target, C = Target)).

bfs(Volumes, target(Target), [state(Jars, HeadActions)|Buffer], Actions):-

generate\_states(Volumes, Jars, HeadActions, NextStates),

append(Buffer, NextStates, NextBuffer),

bfs(Volumes, target(Target), NextBuffer, Actions).

generate\_states(Volumes, Jars, actions(PrevActions), NextStates):-

findall(NextState,

next\_unique\_state(Volumes, Jars, actions(PrevActions), NextState),

NextStates).

next\_unique\_state(Volumes, Jars, PrevActions, NextState):-

next\_state(Volumes, Jars, PrevActions, NextState),

NextState = state(jars(A, B, C), \_Actions),

\+ jars(A, B, C),

assert(jars(A, B, C)).

next\_state(volumes(VolumeA, VolumeB, VolumeC), jars(A, B, C), actions(PrevActions), NextState):-

((\+ B = 0), (\+ A = VolumeA)) , % з В в А

APlusB is min((A + B), VolumeA), % A = A + B

Delta is APlusB - A, % сколько смогли перелить

BMinusDelta is B - Delta, % B = B - Delta

NextState = state(jars(APlusB, BMinusDelta, C), actions([из\_B\_в\_A|PrevActions]));

((\+ B = 0), (\+ C = VolumeC)), % з В в С

CPlusB is min((C + B), VolumeC), % C = C + B

Delta is CPlusB - C, % сколько смогли перелить

BMinusDelta is B - Delta, % B = B - Delta

NextState = state(jars(A, BMinusDelta, CPlusB), actions([из\_B\_в\_C|PrevActions]));

((\+ A = 0), (\+ B = VolumeB)), % з А в В

BPlusA is min((A + B), VolumeB), % B = B + A

Delta is BPlusA - B, % сколько смогли перелить

AMinusDelta is A - Delta,

NextState = state(jars(AMinusDelta, BPlusA, C), actions([из\_А\_в\_В|PrevActions]));

((\+ A = 0), (\+ C = VolumeC)), % з А в С

CPlusA is min((A + C), VolumeC), % C = C + A

Delta is CPlusA - C, % сколько смогли перелить

AMinusDelta is A - Delta,

NextState = state(jars(AMinusDelta, B, CPlusA), actions([из\_А\_в\_C|PrevActions]));

((\+ C = 0), (\+ B = VolumeB)), % з С в В

BPlusC is min((C + B), VolumeB), % B = B + C

Delta is BPlusC - B, % сколько смогли перелить

CMinusDelta is C - Delta,

NextState = state(jars(A, BPlusC, CMinusDelta), actions([из\_C\_в\_В|PrevActions]));

((\+ C = 0), (\+ A = VolumeA)), % з С в А

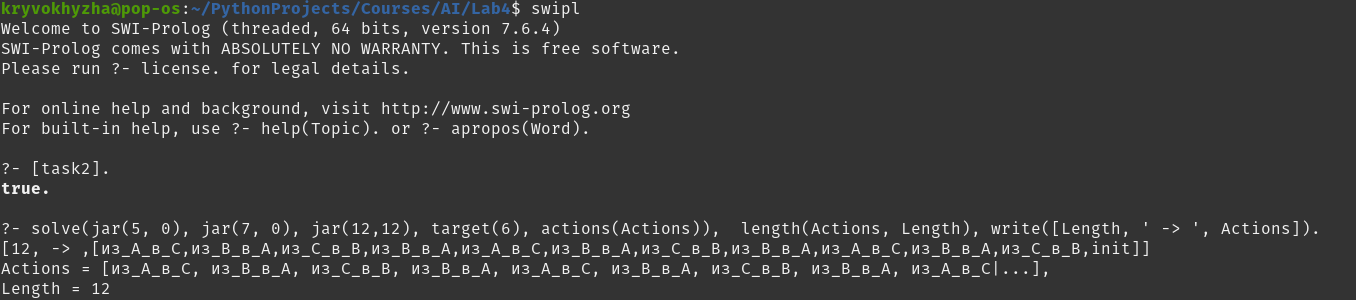
APlusC is min((A + C), VolumeA), % A = A + C

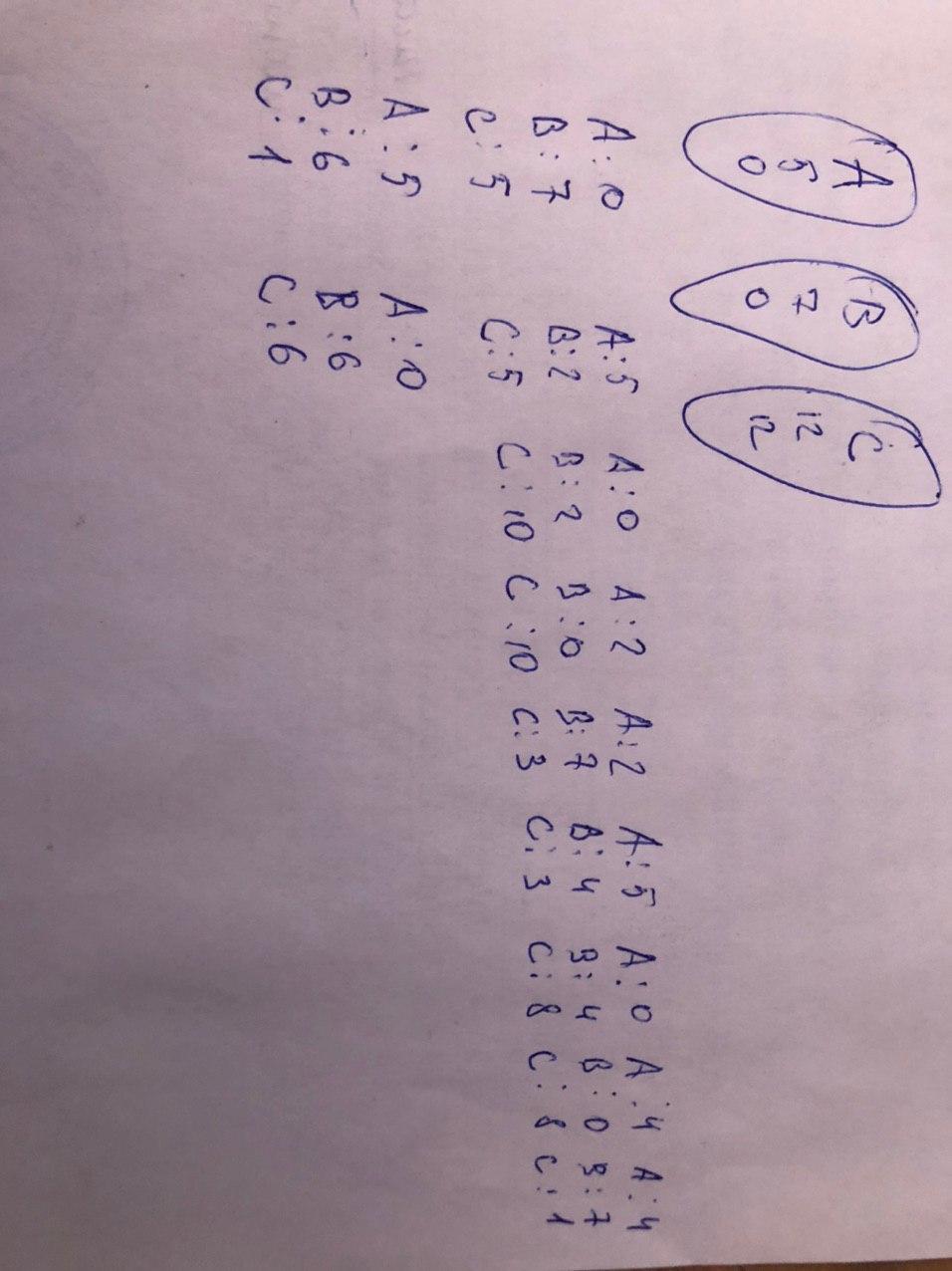
Delta is APlusC - A, % сколько смогли перелить

CMinusDelta is C - Delta, % C = C - Delta

NextState = state(jars(APlusC, B, CMinusDelta), actions([из\_C\_в\_A|PrevActions])).

1. **Результати роботи програмного забезпечення, що включають результати тестування та копії екранних форм.**
2. А – відповідає першій банці (5л)
3. В – відповідає другій банці (7л)
4. С – відповідає бочці (12л)

****

****

**Контрольні запитання**

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

Пошук в ширину:

function BFS(v : Node) : Boolean;

begin

enqueue(v);

while queue is not empty do

begin

curr := dequeue();

if is\_goal(curr) then

begin

BFS := true;

exit;

end;

mark(curr);

for next in successors(curr) do

if not marked(next) then

begin

enqueue(next);

end;

end;

BFS := false;

end;

Пошук в глибину:

function DFS(v : Node; depth : Integer) : Boolean;

begin

if is\_goal(v) then

begin

DFS := true;

exit;

end;

for next in successors(v) do

if DFS(next, depth + 1) then

begin

DFS := true;

exit;

end;

DFS := false;

end;

Пошук в ширину перед тим як спуститися на новий рівень огляне весь поточний рівень на відповідність кінцевій меті, а пошук в глибину спочатку заглибиться по гілці і потім буде поступово підніматися доки не нирне у нову, ще не досліджену гілку.

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

При пошуку з обмеженням глибини використовується заздалегідь задана межа глибини, що дозволяє вирішити проблему нескінченного шляху, а при пошуку з ітеративним поглибленням надається можливість знайти найліпшу межу глибини, оскільки на кожному кроці ця межа збільшується, поки не буде знайдена мета, що робить цей алгоритм повним та оптимальним.

1. Які алгоритми неінформативного пошуку є повними? Оптимальними?

Алгоритм називається оптимальним, якщо він гарантує знаходження кращого з можливих рішень, і до таких алгоритмів входять: пошук в ширину, пошук по критерію вартості при умові, що вартості етапів додатні, пошук в глибину з обмеженням при умові, що межа є найменшою можливою, при якій досягається мета, пошук в глибину з ітеративним поглибленням і т.д.

Алгоритм називається повним, якщо він завжди знаходить рішення, якщо таке існує, і до таких алгоритмів входять: пошук в ширину, пошук по критерію вартості при умові, що вартості етапів додатні, пошук в глибину, пошук в глибину з обмеженням при умові, що межа не менша за необхідну, пошук в глибину з ітеративним поглибленням і т.д.

Алгоритм пошуку з обмеженням глибини не є повним та оптимальним.

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

В даній лабораторній роботі я використовував пошук у ширину для вирішення задачі про переливання. Зрештою, було знайдено роз’язок, довжиной в 11 кроків.

Алгоритм пошуку в ширину перед тим як спуститися на новий рівень огляне весь поточний рівень на відповідність кінцевій меті, тобто отриманий розв’язок буде мати мінімальну кількість кроків, а отже, в даній задачі його можна вважати оптимальним.