Лабораторна робота 3

Створено системою Doxygen 1.9.1

1 Звіт з лабораторної роботи 3

за дисципліною "Інтелектуальні інформаційні системи"
студента групи ПА-17-2
Панасенка Єгора Сергійовича
Кафедра комп'ютерних технологій
ФПМ, ДНУ, 2020-2021 навч.р.
Варант 17

Звіт доступний за посиланням
https://gaurapanasenko.github.io/unilab_opt/IIS_Lab3/html/index.html.
Вихідний код доступний за посиланням
https://github.com/gaurapanasenko/unilab/tree/master/08/IIS_Lab3

1.1 Постановка задачі

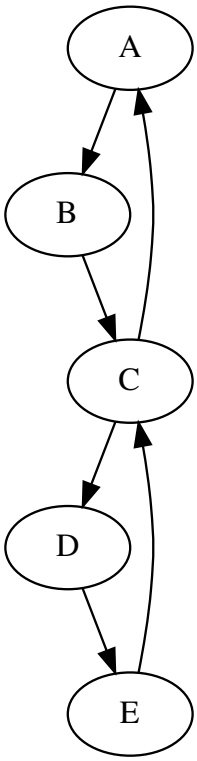
Розв'язати задачу про комівояжера, використовуючи такі методи пошуку:

1. Метод перебору.
2. Метод пошуку в глибину.
3. Метод пошуку в ширину.
4. Метод найкоротшого шляху.
5. Метод Мура.
6. Метод Дейкстри.
7. Метод гілок і меж.
8. Метод Нільсона. Розв'язати задачу про туриста, використовуючи такі методи пошуку:
9. Метод перебору.
10. Метод пошуку в глибину.
11. Метод пошуку в ширину.

12. Метод найкоротшого шляху.
13. Метод Мура.
14. Метод Дейкстри.
15. Метод гілок і меж.
16. Метод Нільсона.

1.2 Побудова графу

Побудуємо деяке сімейне дерево.



1.3 Результат роботи програми

Використаємо компілятор swipl, для початку зкомпілюємо нашу програму.

Тепер запустимо програму та перевримо роботу алгоритму пошуку перебору:
?- find_optimal(X, Y).
X = 0+1+2,
Y = [node(b, c, 2), node(a, b, 1)].

Та запустимо перевірку алгоритму для задачі про мандрівника:
?- travelable(a).
true.
?- travelable(f).
false.

1.4 Висновки

Було розроблено програму, яка здатна перебором вирішувати задачу комівояжера та задачу про мандрівника. Для обох задач був розроблений універсальні предикати, які можуть використовуватися для пошуку перебором у будь-якому двонаправленому графі.

2 Покажчик файлв

2.1 Файли

Повний список файлів.

3 Файли

3.1 Файл main.pl

3.2 main.pl

```
00001 node(a, b, 1).
00002 node(b, c, 2).
00003 node(c, d, 3).
00004 node(d, e, 4).
00005 node(e, c, 5).
00006 node(c, a, 6).
00007
00008 both(A, B, node(A, B, C)) :-
00009     node(A, B, C).
00010 both(A, B, node(B, A, C)) :-
00011     node(B, A, C).
00012
00013 is_in(X, [X | _]).
00014 is_in(X, [_ | Tail]):-
00015     is_in(X, Tail).
00016
00017 add(X, List, [X | List]).
00018
00019 rec(A, A, Path, Path).
00020 rec(A, B, Path, OutPath) :-
00021     both(A, C, N),
00022     not(is_in(N, Path)),
00023     add(N, Path, O),
00024     rec(C, B, O, OutPath).
00025
00026 sum_path([node(_, _, T) | Tail], A) :-
00027     count(Tail, B),
00028     A = B + T, !.
00029 sum_path(_,0) :- !.
00030
00031 min(X, A, Y, _, X, A) :-
00032     X =< Y.
00033 min(_, _, Y, B, Y, B).
00034
00035 find_min([X | Tail], Z, C) :-
00036     find_min(Tail, A, Y),
00037     sum_path(X, B),
00038     min(A, Y, B, X, Z, C), !.
00039 find_min([X | _], Z, X) :-
00040     sum_path(X, Z), !.
00041
00042 count([_ | Tail], A) :-
00043     count(Tail, B),
00044     A = B + 1, !.
00045 count(_,0) :- !.
00046
00047 count_paths(A, B) :-
00048     rec(A, A, [], C),
```

```
00049     count(C, B).
00050
00051 travelable_pred(A, B) :-
00052     count_paths(A, C),
00053     C = B.
00054
00055 count_all_nodes(A) :-
00056     findall(node(A,B,C), node(A,B,C), Lst1),
00057     count(Lst1, A).
00058
00059 find_optimal(X, Y) :-
00060     findall(A, rec(a, c, [], A), Lst), find_min(Lst, X, Y).
00061
00062 travelable(A) :-
00063     count_all_nodes(B),
00064     travelable_pred(A, B), !.
```

3.3 Файл mainpage.dox