Домашна работа по ИИ 1

8puzzle solver

Петко Каменов. Първа група. ФН:45546

1. За решаването на задачата съм използвал езика c++. Задачата е решена в ООП подход.

- Първото нещо което дефинирах бе, класът за State. В него имплементирах няколко метода с чиято цел, да пресмятаме манхатънско разстояние или брой на плочки които не са по местата си. Също така и метод който да генерира произволно състояние, както и други помощни методи за работа с него.
- След това дефинирах Node. Неговата идея е да пази състояние, указател към родителя и дълбочината на която се намира. Също имаме и помощни функции за взимане на родителя и други. С помощта на този клас имплементираме дървото на състоянията.
- Два прости класа със сравняващи функции, чиято идея е проста. Тези класове се наричат функтури. Те дефинират двата начина за сравняване, използвани при двата различни метод. Тоест двете различни евристики
  - При A* -> Manhattan cost + hamming cost + depth
  - При Best First Search -> Manhattan cost
- Последно дефиниран е класът Solver. Той има няколко основни функционалности:
  - Да пази отворения и затворения списък от върхове в него. В задачата ги пазя като vector от указатели към Nodes.
  - Да пази целта
  - Да разширява node. Тоест да открива всички негови наследници. - Това е сравнително просто. Един връх има максимално 4 наследника. Генерираме ги всичките и проверяваме дали всеки един от тях е в полето. Ако е, добавяме го. Ако не е, изхвърляме го.
  - Да проверява дали дадено начално състояние е решимо. Съществуват начални състояния които са нерешими. Алгоритъма за това го взех от сайта geekForGeeks.
  - Да завърта цикъла с който да решава цялата задача. - Идеята е проста:

    Първо се извиква функцията: getNextNode() която в зависимост от това кой алгоритъм сме избрали взима минималния такъв според евристиката. След това се извиква expandNode() идеята на който описах по-горе. Това се прави докато пъзела не е решен. След това се взима последния връх, решението и му се взима родителя, след това неговия и така нататък докато не се стигне до nullptr. Полученият път в дървото на решенията е самото решение.
- В main правим I/O операциите. Генерираме входа до валиден (решим). Принтираме решението.

2. Резултатите от приложението са, че при определен валиден(решим) вход. Програмата решава заданието за около 20 стъпки.

3. Не прилагам псевдо код, защото смятам, че c++ е достатъчно близко до такъв и той изразява съвсем ясно какъв бил псевдо кода.

4. В архива прикачвам: Този pdf с описание на решението ми. Архив с решението на c++. Кода на програмата е разбит в папките header source и файла main.cpp. Прикачва и cmakelists.txt файла за по-лесна компилация, ако използвате clion или друго (не VS) ide което използва такъв тип файлове.

Петко Каменов, 45546, I група