Домашно 1 по „Изкуствен Интелект“
8-puzzle

Проблемът/играта „8-puzzle” в общ вид представлява дъска с размерност 3x3, в която са разположени 9 плочки, съдържащи числата от 0 до 8. Целта на играта е да се наредят плочките последователно и в нарастващ ред, гледайки ред по ред, започвайки от 0. Целта на алгоритмите, които решават този проблем, е да успеят да го направят за минимален брой ходове.

При реализирането на решението по-долу се ползват следните **стратегии**:

- A*
- Best-first search

Оценяващите(евристични) функции, използвани в решението, са:

- Манхатъново разстояние
- Броят плочки, които не са си на мястото спрямо целевото състояние
- Квадратът на Евклидовото разстояние (с цел спестяване смятането на корени, които по никакъв начин не влияят на крайния резултат)

Псевдокод:

Тъй като програмата е реализирана в стил ООП, ще разгледам функцията main, в която се извиква съществената функция от програмата. За останалите, от които тя зависи, ще спомена накратко за какво са, защото са много кратки и не могат да предизвикат объркване.

Solver – клас, чиито обекти имат полета за начално и търсено състояние(Node), избрани евристика и стратегия, списък на съседствата на за всяка отделна клетка (за по-лесна обработка).
Pair – наредена двойка (състояние, евристична стойност)

solve – функция, която извиква основната изчислителна функция solveWith и чиято единствена цел е да я извика, подавайки ѝ евристиката, която сме избрали при създаване на обект от тип Solve.

Solver solver(startState, targetState, Strategy, Heuristic) – инициализира полетата на solver обекта

Heuristic – заема една от 3те си възможни стойности

Strategy – заема една от 2те си възможни стойности

За яснота са сложени в квадратни скоби полетата, които имплицитно са дадени на функцията:
solveWith(Heuristic function of choice[, Strategy of choice, startNode, targetNode, adjacencyList])

```
  visitedBoards← array[900 000 000]        //за константна проверка дали нещо е срещано, макар и
                                           //да не е оптимална откъм памет
  frontier ← empty priority queue of Pairs
  frontier ← (startNode, heuristic value of startNode)
  while frontier isn't empty do
    parentNode ← first element of frontier
    parentBoard ← board of parentNode
    if parentBoard matches the goal board then
      return the depth of parentNode
    visitedBoards ← parentBoard
    zeroID ← position of 0 on parentBoard
    adj ← adjacency list of zeroID
    for each ID in adj do
      childBoard ← parentBoard with swapped values in positions ID and zeroID
      if childBoard is in visitedBoards then
        skip to the next iteration
      childNode ← (childBoard, parentNode depth + 1)
      frontier ← (childNode, heuristic value of childNode)
```

Резултати при експериментиране с различните стратегии и евристики:

A*:

- В мнозинството от случаите алгоритъмът A* работи еднакво и с трите предоставени евристики. В изолирани случаи той намира оптимално решение за 1-5 стъпки по-бавно, когато работи с Евклидово разстояние. В останалите случаи няма осезаема разлика в броя стъпки, с които достигаме до решение, измежду трите евристики.
- При работата с Евклидово разстояние има разлика в зависимост от това дали разглеждаме нулата (празното поле) или не.

Best-first search:

- Във всеки случай евристиката „брой плочки не на място“ се оказва най-неподходяща. Този резултат не е съвсем неочакван, защото методът Best-first разчита само и единствено на евристиката, а в този случай тя е изключително бедна, заемаща цели стойности в интервала [0;8].
- Дали разглеждаме празното поле при изчислението на евристичните стойности на отделните състояния, се оказва съществен фактор. Допълнително, в някои случаи разглеждането на празното поле води до по-оптимален резултат, а в други – пропускането на празното поле. При Евклидовото и Манхатъновото разстояние е по-добре да не го разглеждаме. При броенето на разлики е по-добре да го разглеждаме. (Тествани са върху идентични дъски при двата варианта за нулата)
- Съществена роля играе редът, в който разглеждаме съседите на всяко текущо състояние, което изваждаме от Open list-а. Причината отново е малкото множество от допустими стойности на евристиките и съответно редът, в който разглеждаме елементи с еднакви оценки, е въпрос на подредба на списъка на съседствата.
- Най-неефикасно е гледането на броя плочки, които не са си по местата. След него се нарежда Манхатъновото разстояние и най-ефикасно е Евклидовото разстояние. Това прави доста интересна съпоставката между A* и Best-first, защото, в контраст с Best-first, при A* Евклидовото разстояние се оказва най-неефикасно.

Средно аритметични резултати при изпълнението на комбинациите стратегия-евристика, като в евристиките НЕ се разглежда празната клетка:

```
Average results of 1000 tests which ignore 0:
A* Manhattan     19.863
A* Misplacements 19.863
A* Euclidean     20.465
BestFirst Manhattan     56.029
BestFirst Misplacements 78.201
BestFirst Euclidean     42.061

Process returned 0 (0x0)   execution time : 612.088 s
Press any key to continue.
```

Средно аритметични резултати при изпълнението на комбинациите стратегия-евристика, като в евристиките се разглежда празната клетка:

```
Average results of 1000 tests which don't ignore 0:
A* Manhattan     20.43
A* Misplacements 20.392
A* Euclidean     21.23
BestFirst Manhattan     63.3
BestFirst Misplacements 82.118
BestFirst Euclidean     54.148

Process returned 0 (0x0)   execution time : 693.691 s
Press any key to continue.
```

Забележка: тестовете са извършени върху две различни извадки от по 1000 начални състояния