UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

Probleme de cautare si agenti adversariali

Inteligenta Artificiala

Autori: Oprisor Paul si Turda Sorin
Grupa: 30232

Facultatea de Automatica
si Calculatoare

3 Decembrie 2024

Cuprins

1 Tutorial

1.1 Subtitlu 1

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

1.1.1 Subsubtitlu

Born into the noble Crownguard family, along with his younger sister Lux, Garen knew from an early age that he would be expected to defend the throne of Demacia with his life. His father, Pieter, was a decorated military officer, while his aunt Tianna was Sword-Captain of the elite Dauntless Vanguard—and both were recognized and greatly respected by King Jarvan III. It was assumed that Garen would eventually come to serve the king's son in the same manner.

1.2 Tutorial elemente de baza

O propozitie normala. Daca vrem sa adaugam text in **bold**, *italic* sau ***bold italic***.
Astfel se poate utiliza o **lista numerotata**:

1. primul element
2. al doilea element
3. al treilea element

Astfel se poate utiliza o **lista neumerotata**:

- primul element
- al doilea element
- al treilea element

Asa se adauga o **imagine**.

Figura 1: Baba Voss

Daca vrem sa referim imaginea 1 in text se procedeaza astfel.

Asa se adauga **cod**:

```
def fib(n):    # write Fibonacci series up to n
    """Print a Fibonacci series up to n."""
    a, b = 0, 1
    while a < n:
        print(a, end=' ')
        a, b = b, a+b
    print()
```

2 Uninformed search

2.1 Question 1 - Gasirea unui punct unde se afla mancare folosind Depth First Search

Gasirea unui punct unde se afla mancare folosind Depth First Search

Depth First Search este un algoritm utilizat pentru explorarea grafurilor sau arborilor. Scopul acestuia este de a traversa toate nodurile unui graf, urmărind o cale cât mai adâncă înainte de a reveni și a explora căile neexplorate.

Cod **DFS**

```
def depthFirstSearch(problem: SearchProblem) -> List[Directions]:
        """
        Cautare in adancime
        """
        stack = Stack()
    node = problem.getStartState()
    visited = set()
    stack.push((node, []))
    while not stack.isEmpty():
        position, path = stack.pop()
```

Algorithm 1 Iterative Depth First Search (DFS)

```
1: procedure IterativeDFS(G, v)
2:     Initialize an empty stack S
3:     Push v onto S
4:     Mark v as visited
5:     while S is not empty do
6:         u ← Pop(S)
7:         for each neighbor w of u in G do
8:             if w is not visited then
9:                 Push w onto S
10:                Mark w as visited
11:            end if
12:        end for
13:    end while
14: end procedure
```

```
11          if position not in visited:
12              visited.add(position)
13              if problem.isGoalState(positoin):
14                  return path
15              for successor, action, cost in problem.getSuccessors(position):
16                  if successor not in visited:
17                      stack.push((successor, path + [action]))
18      return []
```

2.2 Question 2 - Breadth-first search

3 Informed search

3.1 Question 4 - A* search algorithm

4 Adversarial search

4.1 Question 9 - Improve the ReflexAgent