Реализация и использование бинарных деревьев

«Исскуственный интелеект»

- Игра «Угадай животное»
- Сценарий
- Анализ

Подходы к программированию

- Снизу вверх
- Сверху вниз

Структура данных

- СД на курсорах
- Заполнение базы для рассмотренного сценария

N	**data**	**Left**	**right**
0	Он рептилия?	2	3
1	Кот	-1	-1
2	У него длинная шея?	1	4
3	Крокодил	-1	-1
4	Жираф	-1	-1

Реализация

- Объявление класса и конструктор
- *# Структура данных: поле данных, курсор на левого сына, курсор на правого сына*

```
class BinaryTree:
# Инициализация. Сделать пустым
    def __init__(self):
        self._data_    = []
        self._left_    = []
        self._right_   = []
        self._root_    = -1
        self._current_ = -1
        self._path_=[]
```

Создать корень

```
def create_root(self,data):
    BinaryTree()
    self._root_= 0
    self._data_.append(data)
    self._left_.append(-1)
    self._right_.append(-1)
```

Встать в начало

```
def goto_first(self):
    self._current_=self._root_
    self._path_=[]
```

Проверка наличия сыновей и родителей

```
# Проверить, есть ли левый сын
  def is_right_empty(self):
    return self._right_[self._current_]==-1


# Проверить, есть ли правый сын
  def is_left_empty(self):
    return self._left_[self._current_]==-1

# Проверить, есть ли родитель
  def is_left_empty(self):
    return len(self._path_)==0
```

Движение по дереву

```
# Переместиться к левому сыну
def goto_left(self):
  if not self.is_left_empty:
    self._path_.append(self._current_)
    self._current_=self._left_[self._current_]

# Переместиться к правому сыну
def goto_right(self):
  if not self.is_right_empty():
     self._path_.append(self._current_)
     self._current_=self._right_[self._current_]

# Переместиться к родителю
def goto_parent(self):
    if not self.is_parent_empty():
           self._current_=self._path_.pop()
```

Это лист?

```
def is_leaf(self):
    return self.is_left_empty and \
           self.is_right_empty()
```

Добавление левого сына

```
def add_left(self,left_data):
  if self.is_left_empty:
    self._data_.append(left_data)
    self._left_.append(-1)
    self._right_.append(-1)
    self._left_[self._current_]= \
       len(self._data_)-1
    return True # если все прошло удачно
  return False
```

Добавление правого сына

```
def add_right(self,right_data):
    if self.is_right_empty():
        self._data_.append(right_data)
        self._left_.append(-1)
        self._right_.append(-1)
        self._right_[self._current_] = \
                        len(self._data_) - 1
        return True
return False
```

Сеттеры и геттеры

```
# Получить текущее поле данных
    def get_current_data(self):
        return self._data_[self._current_]

# Заменить текущее поле данных
    def set_current_data(self,cur_data):
        self._data_[self._current_] = cur_data

# Получить список полей данных
    def get_data_list(self):
        return self._data_

# Получить список курсоров на левых сыновей
    def get_left_list(self):
        return  self._left_

# Получить список курсоров на правых сыновей
    def get_right_list(self):
        return self._right_
```

Сеттеры списков

```
# Установить список полей данныъх
    def set_data_list(self,list_data):
        self._data_ = list_data


# Установить список курсоров на левых сыновей
    def set_left_list(self,list_left):
        self._left_ = list_left


# Установить список курсоров на правых сыновей
    def set_right_list(self,list_list_left):
        self._right_ = list_right
```

Класс ArtificialIntelligence

```
from binary_tree import *

class ArtificialIntelligence:
    def __init__(self):
        self._animal_bd_=BinaryTree()
        self._animal_bd_.create_root("Кот")

    def exist_question(self):
        return not self._animal_bd_.is_leaf()

    def ask(self):
        return  self._animal_bd_.get_current_data()

    def next_question(self,answer):
         if answer: self._animal_bd_.goto_right()
         else:      self._animal_bd_.goto_left()

    def learn(self,a,q):

self._animal_bd_.add_left(self._animal_bd_.get_current_data())
         self._animal_bd_.add_right(a)
         self._animal_bd_.set_current_data(q)

    def start(self):
        self._animal_bd_.goto_first()
```

Игра «Угадай животное»

```
from artificial_intelligenc import *

ai=ArtificialIntelligence()
exit_game=False;
while not exit_game:
    print("Загадай животное. Готов?")
    ans=int(input("(Да-1, Нет-0)"))
    ai.start()
    while ai.exist_question():
        print(ai.ask)
        ans=bool(input("(Да-1, Нет-0)"))
        ai.next_question(ans)
    print("Это", ai.ask() + "? (Да-1, Нет-0)")
    ans=int(input())
    if ans: print("Я победил!!! AI --- рулит!")
    else:
        a=input("Кто это? ");
        q=input("Задай вопрос, ответом ДА, на который будет " \
            + a +", а ответом НЕТ ---" + ai.ask() + ". ");
        ai.learn(a,q)
    exit_game=not int(input("Хочешь сыграть ещё?" + \
                            "(Да-1, Нет-0)"))
```

Обсуждение

- Что можно доделать?
- Достоинства подхода?
- Что дают АТД и классы?
  - Новая декомпозиция задачи
  - Повторное использование кода
  - Инкапсуляция

Анонсы сл. лекций

- Завершение АТД
- Объектно-ориентированнное программирование
- Введение в Объектно-ориентированный анализ и проектирование
- Введение в шаблоны проектирования
- Модульное тестирование
- Функциональное программирование
- Регулярные выражения
- Библиотеки Python
- Может быть GUI