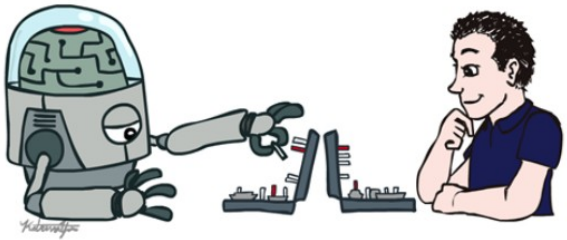
Введение в мобильную робототехнику
Часть 2 - плэннинг

Станислав Кикоть



LONDON METROPOLITAN UNIVERSITY

SUDOKU

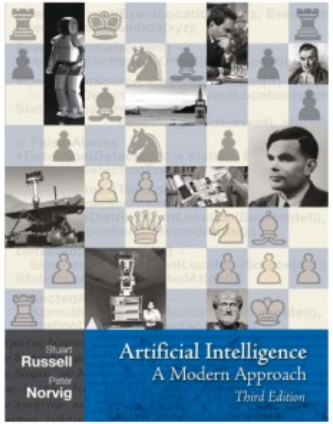
			9		5			
1		5		6		7		2
							3	8
2					1	9		4
	8				7			
3				9		1		5
			7				6	1
5	1				3		4	
7		3	8		4			

ANSWER

8	7	2	9	3	5	4	1	6
1	3	5	4	6	8	7	9	2
6	9	4	1	7	2	5	3	8
2	5	6	3	8	1	9	7	4
9	8	1	5	4	7	6	2	3
3	4	7	2	9	6	1	8	5
4	2	8	7	5	9	3	6	1
5	1	9	6	2	3	8	4	7
7	6	3	8	1	4	2	5	9

Artificial Intelligence

Dr. Stan Kikot

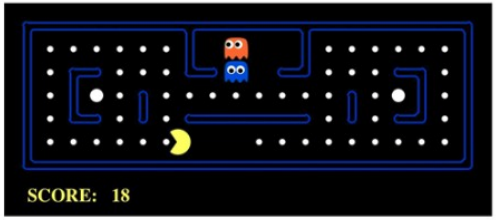


7	2	4
5		6
8	3	1

Start State

	1	2
3	4	5
6	7	8

Goal State



Состав второй части курса

- **3 лекции: 9, 16 и 23 ноября**
- **2 домашних задания**
  - **“Пакман”** (непроверяемое) будет выдано сегодня
  - “**Построение траектории**” (зачетное) будет выдано 16 ноября, дедлайн для сдачи **1 декабря (среда)**
- разбор решений зачетного задания **7 декабря (вторник)** (~45 минут)

Таймлайн второй части курса

- **9 ноября – лекция “Поиск на графах в контексте ИИ” + выдается непроверяемое Д/З 4 “Пакман”**
- **16 ноября – лекция “Построение и изменение траектории” + выдается зачетное Д/З 5 “Построение траектории”**
- **23 ноября – лекция “Модель безопасности RSS” + Q&A**

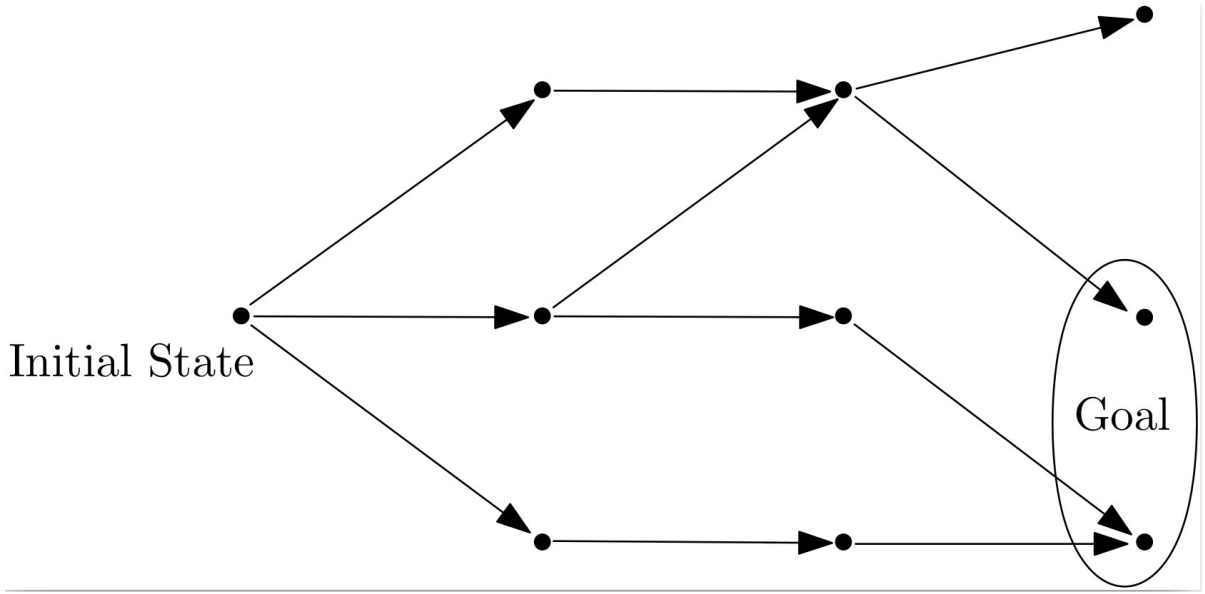
  (~120 минут)
- **30 ноября – нет лекции**

  (завтра – 1 декабря, дедлайн к зачетному Д/З 5)
- **7 декабря - разбор зачетного задания (~60 минут)**

ИИ = поиск пути на графе ?

- **Как ?**
  - **перевезти волка, козу и капусту на другой берег**
  - **доехать из пункта А в пункт Б**
  - **составить расписание занятий**
  - **сгруппировать вагоны по составам**
  - ~~**заработать $1.000.000**~~
  - ~~**избавиться от бедности и ковида**~~
  - **написать код планировщика для автономного автомобиля**

Метаалгоритм поиска пути



Graph search

```
function GRAPH-SEARCH(problem, frontier) returns a solution, or failure
    explored ← an empty set
    INSERT(ROOT-NODE(problem.INITIAL-STATE), frontier)
    while not EMPTY?(frontier) do
        node ← REMOVE(frontier)
        add node.STATE to explored
        if problem.GOAL-TEST applied to node.STATE succeeds return node
        for each action in problem.ACTIONS(node.STATE) do
            child ← CHILD-NODE(problem, node, action)
            if child.STATE is not in explored then
                INSERT(child, frontier)
    return failure
```

Yes, this node is the goal node, but a solution is a sequence of actions, which can be obtained from *node* by following the *parent* link until it points at None.

node = (state, parent, action, cost)