ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

INTRODUCTION TO ARTIFICIAL INTELLIGENCE

MID-TERM PRESENTATION

523H0011- Tôn Minh Đăng
523H0082 - Phan Việt Quan
523H0107 - Nguyễn Minh Triết

CONTENT

I. Task 1 - A* with 8-Puzzle

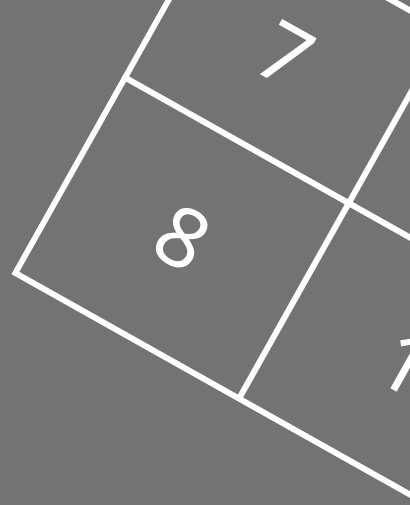
- Formulate problem.
- Describe 2 heuristics.
- Compare performance.

II. A* with Pacman

- Formulate problem.
- Implement GUI.
- Time & Space complexity of algorithm.

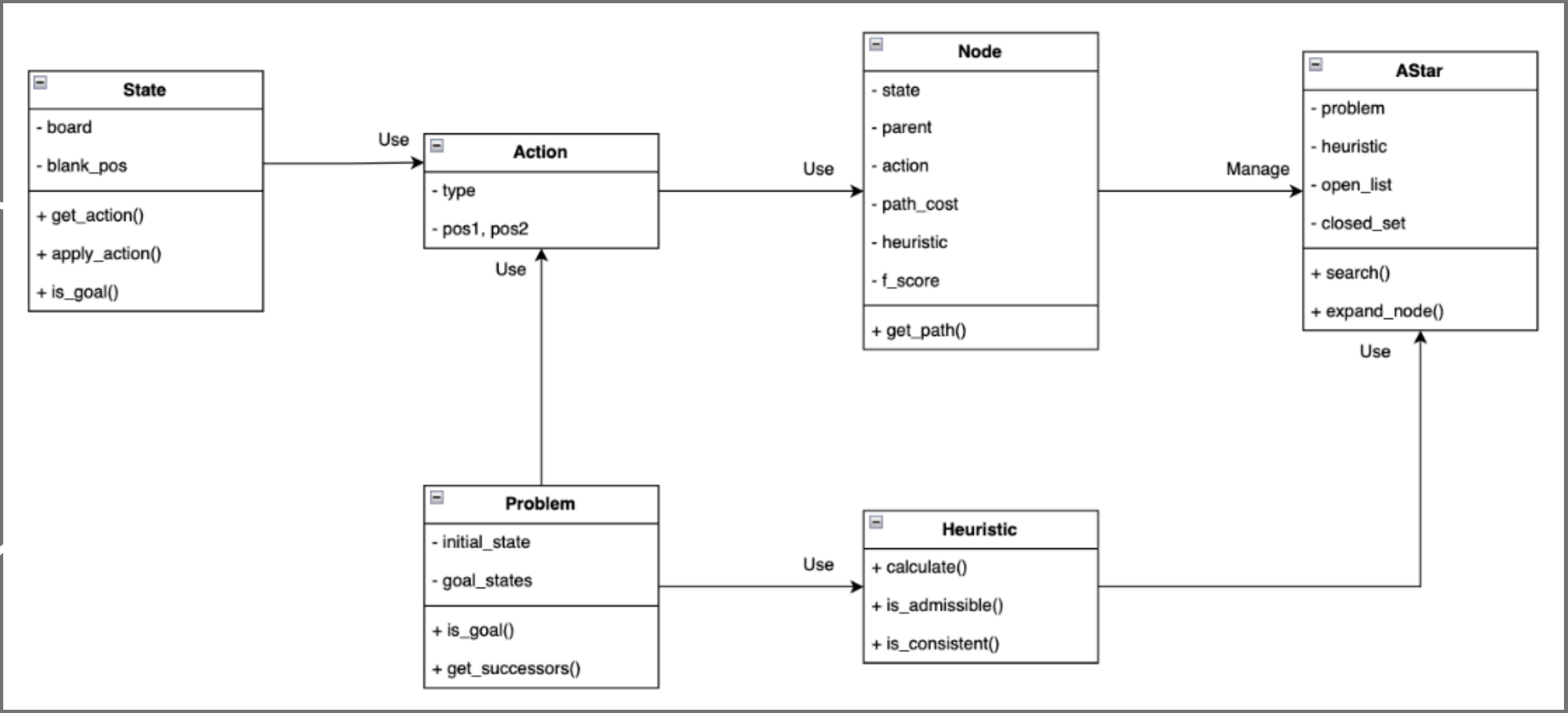
ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Task 1
A* with 8-Puzzle



A* WITH 8-PUZZLE

1.1 Formulate problem



02 HEURISTICS

Misplaced Tiles Heuristic: function would compare current state with all goals state. With each goal state, **number misplaced is number of wrong positions** from the goal state. Function **return minimum Misplaced value** of these goal states.

1	2	7
4	5	6
3	8	0

$h(n) = 2$

ADMISS & CONSIS

[[1,2,7], [4,5,6], [3,8,0]]

h(n) = 2

h*(n) = 1

h(n) > h*(n)

=> Not Admissibility

h(n) <= c(n, n') + h(n')

h(n) <= 1 + h(n')

|h(n) - h(n')| <= 1 (1)

|h(n) - h(n')| = 2 (2)

(1) >< (2)

=> Not Consistency

02 HEURISTICS

Break point Heuristic: function would compare current state with all goals state. With each goal state, number position that:

$$|x(n+1) - x(n)| \; != 1$$

Function return minimun value of 4 goal state.

1	3	2
4	5	6
7	8	0

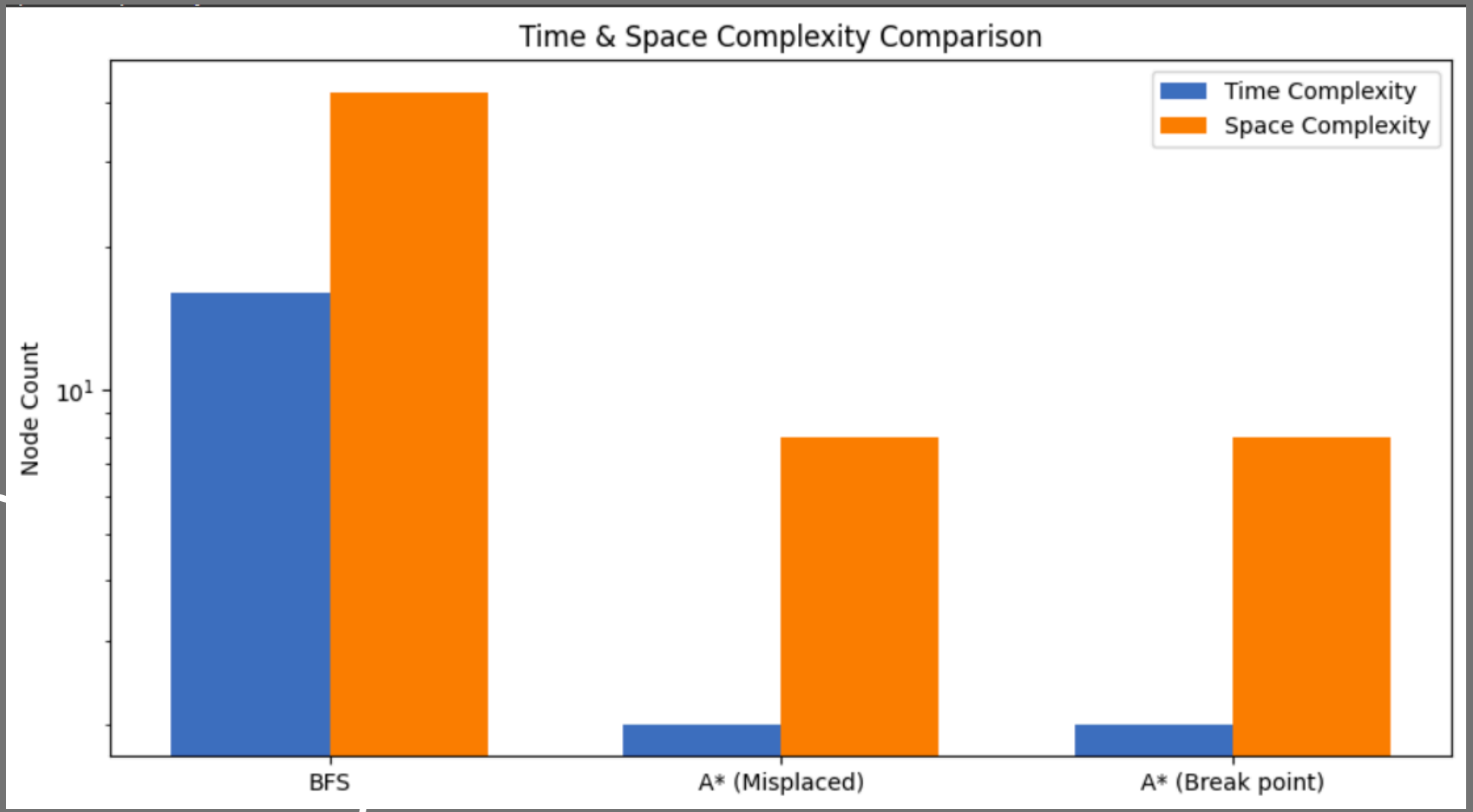
$$h(n) = 3$$

ADMISS & CONSIS

[1,2,3,4,5,6,7,0,8]
-> 2 break points
[1,2,3,4,5,6,7,8,0]
-> 0 break points
h(n) = 2 ; h*(n) = 1
h(n) > h*(n)
=> Not Admissibility

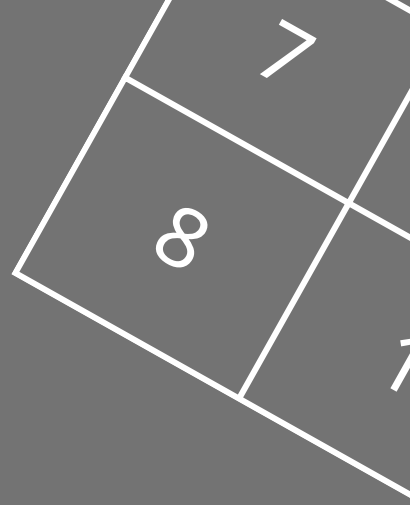
[1,2,3,4,5,6,7,0,8]
-> 2 break points
-> h(n) = 2
[1,2,3,4,5,6,7,8,0]
-> 0 break points
-> h(n') = 0
h(n) > c(n, n') + h(n')
~ 2 > 1 + 0
)
=> Not Consistency

TIME & SPACE COMPLEXITY



ĐẠI HỌC TÔN ĐỨC THẮNG
TON DUC THANG UNIVERSITY

Task 2
A* with Pacman



A* WITH PACMAN

2.1 Formulate problem

- **Reuse A*** algorithm from Task 1,
- **Change** Problem and relative class.
- Define **new heuristics that more suitable** with Pacman Problem.

A* WITH PACMAN

2.2 Pygame workflow

- **Reuse A*** algorithm from Task 1,
- **Change** Problem and relative class.
- Define **new heuristics that more suitable** with Pacman Problem.

A* WITH PACMAN

2.1 Run_auto_mode

Explain method --run_auto_mode(layout_lines, heuristic: str = "auto")-- step by step:
Input:
- initial state in List[str] form: --layout_lines--
- name of heuristic (mapping from --_select_heuristic--): --heuristic--
Proccess:
- Create `PacmanEnvironment` from --layout_lines--. `PacmanEnvironment` include `PacmanLayout.`
- Create `PacmanProblem` from `PacmanEnvironment.`
- Creaete a heuristic object by `_select_heuristic.` Specially, `_select_auto` could select the best heuristic base on complexity of problem.
- Run `AStar(problem, heuristic)` extend on Task 1, return (path, cost, expanded, frontier_max)`
Output:
- path, cost, expanded, frontier_max

A* WITH PACMAN

Result: compare with UCS