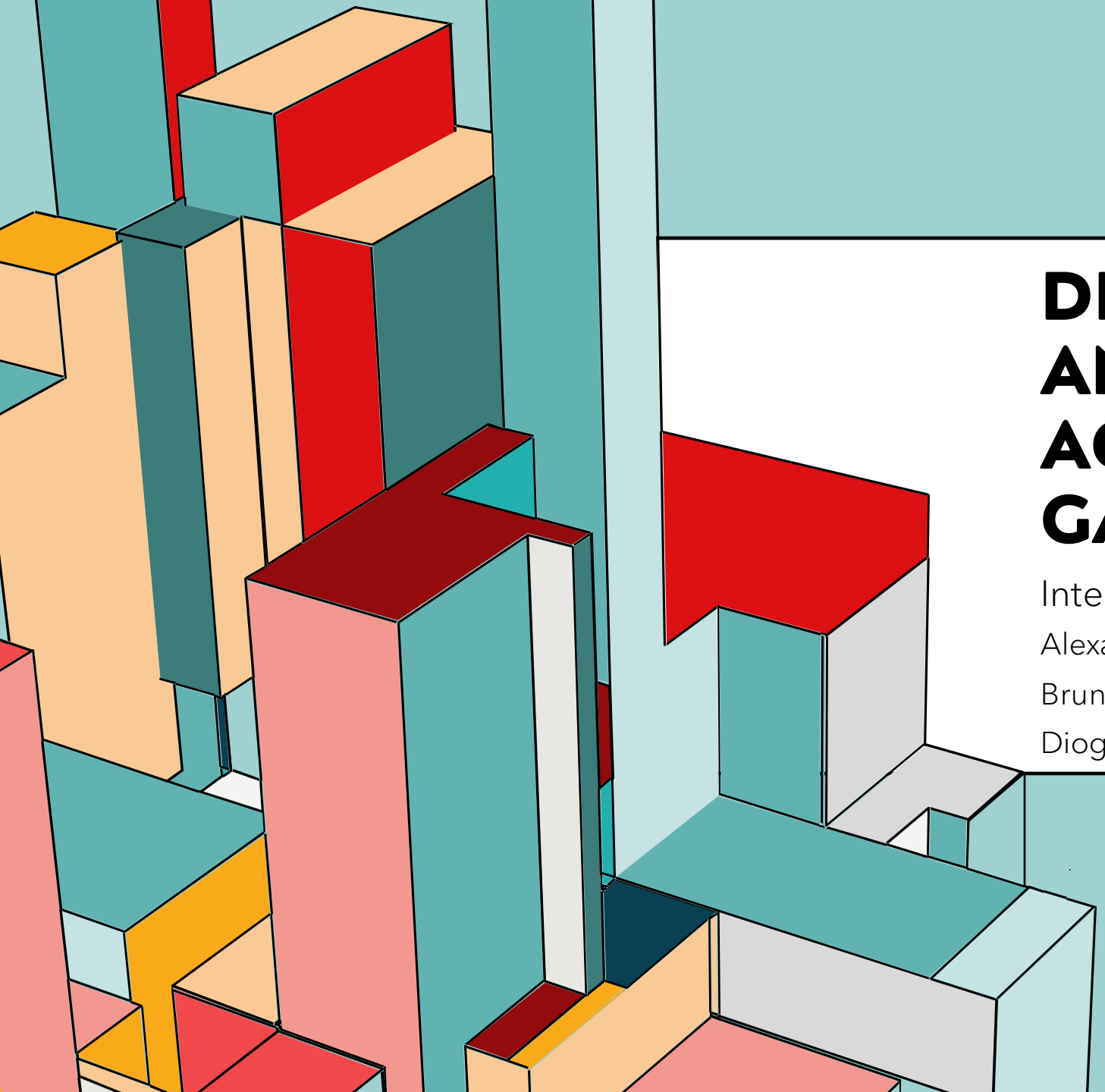


DEVELOPMENT OF AN AUTONOMOUS AGENT FOR THE GAME RUSH HOUR

Inteligência Artificial

Alexandre Martins 103552

Bruno Gomes 103320

Diogo Silva 104341

ARQUITETURA

Map

- car_actions : devolve uma lista de tuplos com as possíveis ações de uma peça.
- get_car_orientation : devolve a orientação da peça.
- move_two : move uma peça até às coordenadas indicadas

Dominio

- actions : Utiliza o car_actions para obter um set com todas as ações possíveis de todos os carros
- result_two : atualiza o Map após executar um movimento
- heuristic : distancia do carro vermelho até à saída

tree_search

- search : é criada uma variável para armazenar todos os estados conhecidos para evitar loops infinitos. Por cada ação possível de cada carro é devolvida uma simulação da grid em string. sSe esta nova grid não estiver na variável de estados conhecidos é criado um objeto Map e de seguida um SearchNode, é calculado a heurística caso o algoritmo exija.

ARQUITETURA

student

- cursor_on_car: Coloca o cursor no carro
- cursor_selecting_car: diz se o cursor está a selecionar um carro
- cursor_is_selected_on_to_something: diz se está a selecionar algo
- get_to_car : faz com que o cursor chege até ao carro
- agent_loop: vai buscar o primeiro elemento do set de actions e após o completar a ação dá pop econtinua para a próxima action.

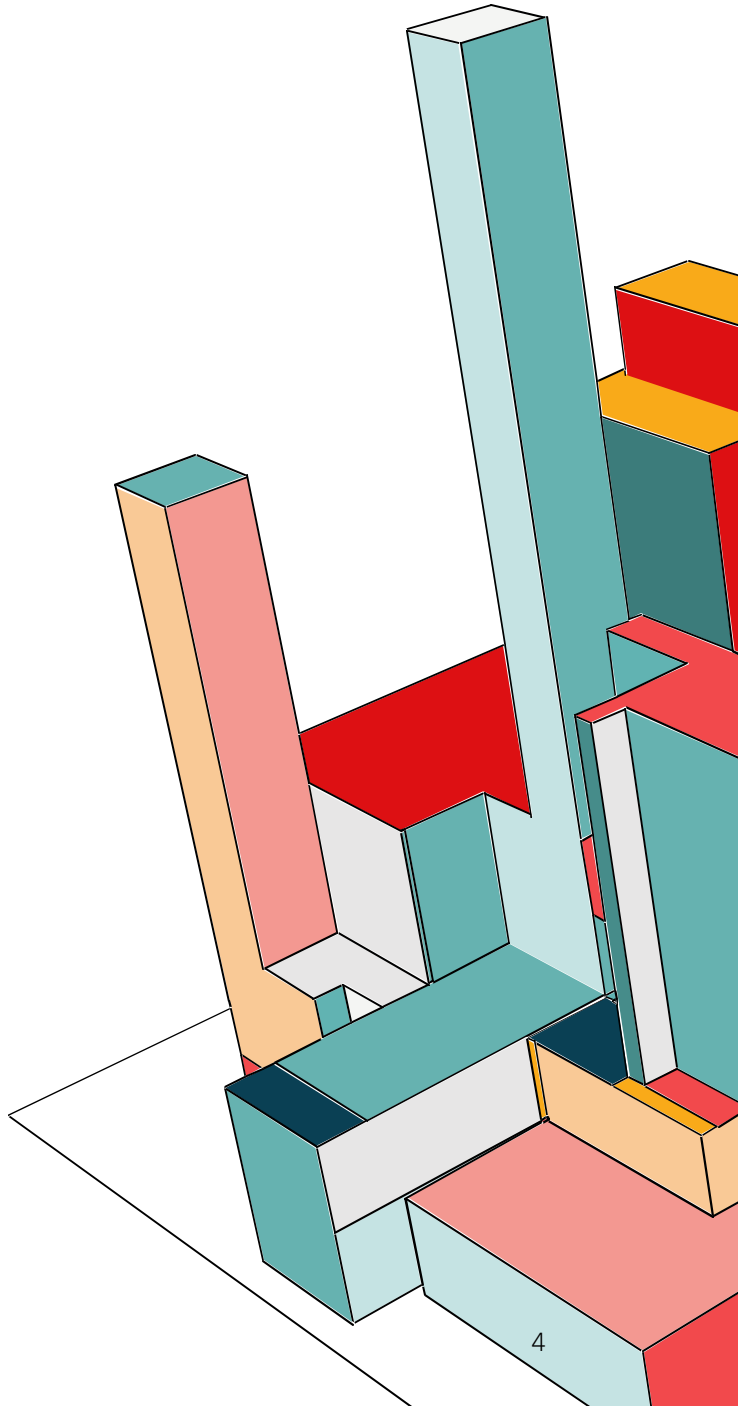
ALGORITMO UTILIZADO

Greedy

Um algoritmo greedy é um algoritmo que utiliza heurística de resolução de problemas para fazer a melhor escolha.

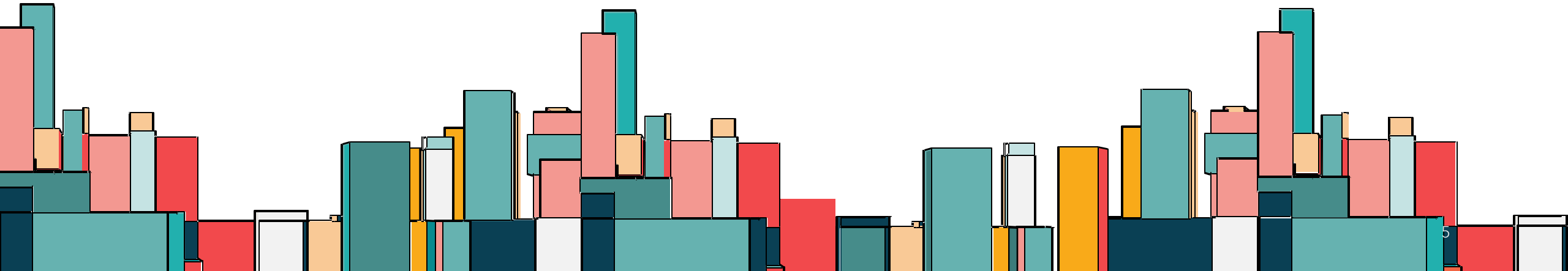
Heurística utilizada

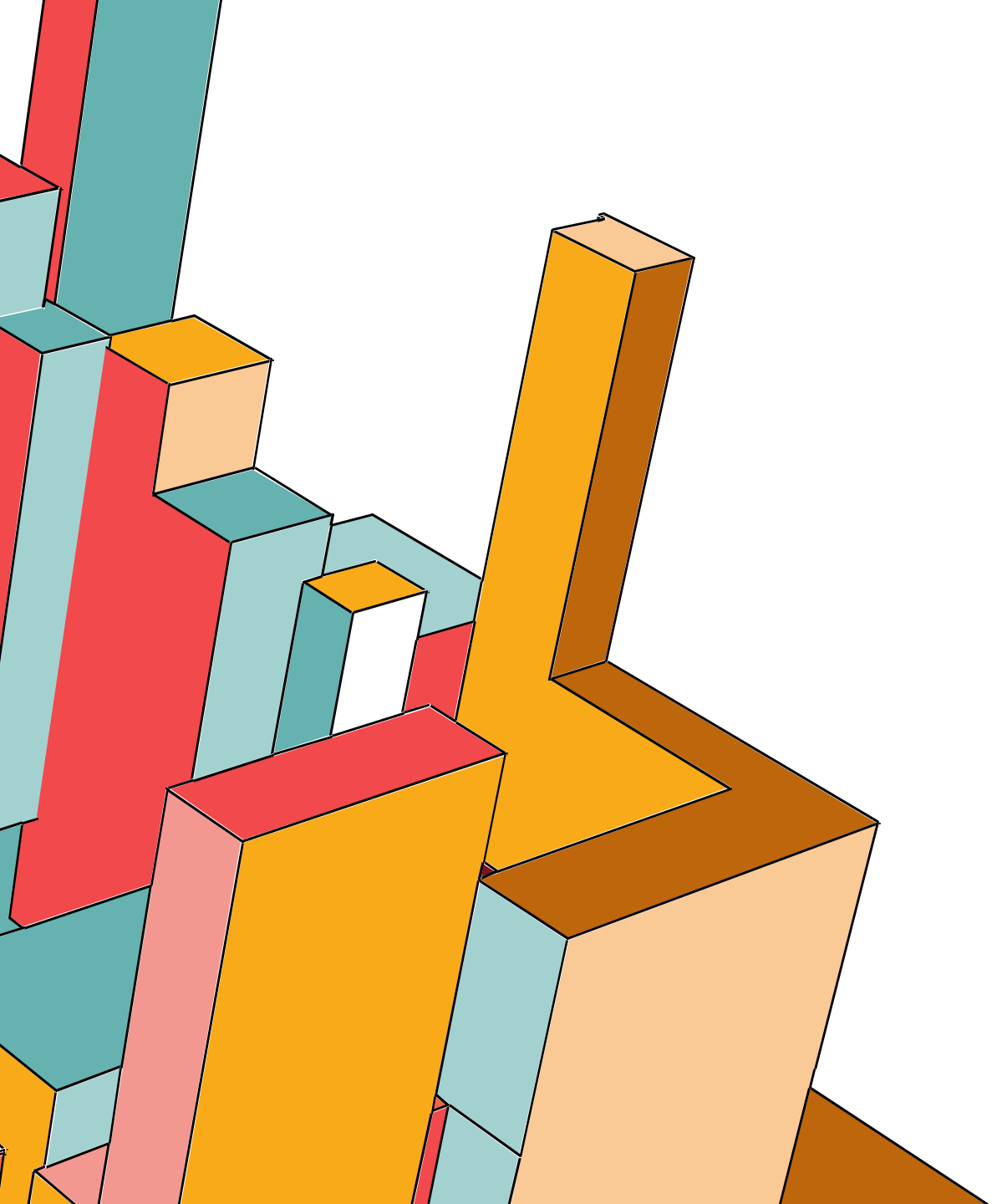
A nossa heurística devolve a distância do carro vermelho até à saída, é admissível pois não sobrestima a quantidade de jogadas necessárias.



REPLANEAMENTO

Quando detetamos que uma peça vai tentar fazer uma jogada impossível corremos novamente o search para popular o array de jogadas válidas baseadas no novo mapa.





BENCHMARK DA SOLUÇÃO

tests.py

Fizemos um ficheiro de testes onde experimentamos ao longo do projeto as várias partes do nosso código