# Trabalho prático individual nº 1

# Inteligência Artificial
# Ano Lectivo de 2024/2025

25-26 de Outubro de 2024

## I Important remarks

1. This assignment should be submitted via *GitHub* within 28 hours after the publication of this description. The assignment can be submitted after 28 hours, but will be penalized at 5% for each additional hour.

2. Complete the requested functions in module `"tpi1.py"`, provided together with this description.

3. Include your name and number and comment out or delete non-relevant code (e.g. test cases, print statements); submit only the mentioned module `"tpi1.py"`.

4. You can discuss this assignment with colleagues, but you cannot copy their programs neither in whole nor in part. Limit these discussions to the general understanding of the problem and avoid detailed discussions about implementation.

5. Include a comment with the names and numbers of the colleagues with whom you discussed this assignment. If you turn to other sources, identify them as well.

6. All submitted code must be original; although trusting that most students will do this, a plagiarism detection tool will be used. Students involved in plagiarism will have their submissions canceled.

7. The submitted programs will be evaluated taking into account: performance; style; and originality / evidence of independent work. Performance is mainly evaluated concerning correctness and completeness, although efficiency may also be taken into acount. Performance is evaluated through automatic testing. If necessary, the submitted modules will be analyzed by the teacher in order to appropriately credit the student's work.

## II Exercices

Together with this description, you can find the `tree_search` module. You can also find attached the modules `cidades`, `strips`, and `blocksworld`, containing the `Cidades(SearchDomain)`,

`STRIPS(SearchDomain)` and other related classes. These modules are similar to the ones initially provided for the practical classes, but with some changes and additions, namely:

- Loop prevention (repeated states along a path) is implemented
- The `Cidades(SearchDomain)` is given fully implemented.
- The `STRIPS(SearchDomain)` is given fully implemented, except for the heuristic.

Don't change the `tree_search`, `cidades`, `strips` and `blocksworld` modules.

The module `tpi1_tests` contains several test cases. If needed, you can add other test code in this module.

Module `tpi1` contains the classes `MyTree(SearchTree)`, `MyNode(SearchNode)` and `MyBlocksWorld(STRIPS)`. In the following exercices, you are asked to complete certain methods in these classes. All code that you need to develop should be integrated in the module `tpi1`.

1. Create a new method `search2()` in class `MyTree`, similar to the original search method, and make sure that nodes (class `MyNode`) have the attributes `depth`, `cost`, and `heuristic`, with the usual meaning, and also `action`, the action that led from the parent state to the current state. In addition, make sure the search tree (class `MyTree`) contains the following counters:
   - `num_open` - number of open nodes, i.e. nodes in the queue;
   - `num_solution` - number of solution nodes found (one of the techniques below finds several solutions);
   - `num_skipped` - number of skipped nodes, i.e. nodes removed from the queue but not expanded (one technique skips some nodes).
   - `num_closed` - number of closed nodes, i.e. nodes removed from the queue and expanded.
2. In `MyTree`, implement the method `astar_add_to_open(lnewnodes)`, which manages the queue of open nodes according to the values of the A* evaluation function. In the case of a tie in the evaluation function, use the node depth as a second creterion and the alphabetical order of states as a third criterion.
3. The *Informed Depth-First Search* technique consists of normal depth-first search, but the children produced when expanding a node are sorted according to the A* evaluation function. Therefore, the first child to be expanded will be the one with lowest evaluation function. In `MyTree`, implement the method `informeddepth_add_to_open(lnewnodes)`, to manage the queue when this technique is being used. In the case of a tie in the evaluation function, use the alphabetical order of states as a second criterion.
4. The `MyTree` constructor has an optional argument `improve`, which by default is `false`. In the default case, the search stops when it finds the first solution. When `improve=True`, the search will continue until the queue is empty, and at that point it returns the best solution found. To prevent exhaustive search, any visited node, with A* evaluation function equal or higher than the cost of the best solution so far, will be skipped (not expanded).
5. Develop the method `check_admissible(node)` in `MyTree`, that, given a solution node, checks if all nodes in the path from the root have heuristic values lower or equal to the actual cost of reaching the solution node.

6. Develop the method `get_plan(node)` in `MyTree`, which, given some node, returns the sequence of actions from the root to that node.

7. In class `MyBlocksWorld`, develop the method `heuristic(state,goal)`, which, given a state and the goal, estimates the cost of reaching the goal from that state. This exercice will be scored taking into acount, not only if the solution found with A* is correct and optimal, but also how large/small is the search tree.

# III Clarification of doubts

This work will be followed through `http://detiuaveiro.slack.com`. The clarification for the main doubts will be added here.

1. Os valores das heurísticas dos testes do BlocksWorld são meramente indicativos é isso?

   **Resposta**: Nesse exercício, vocês têm que inventar uma heurística, e cada uma dará valores diferentes; quanto mais aproximada for a heurística, relativamente ao custo real, melhor.

2. No exercício 4, quando `improve=True`, o `num_skipped` refere-se ao "número de nodes removidos da queue e não expandidos"ou somente ao "número de nodes removidos da queue pois a avaliação A* do node é maior ou igual ao custo da melhor solução"?

   **Resposta**: Significa "número de nodes removidos da queue e não expandidos [...] pois a avaliação A* do node é maior ou igual ao custo da melhor solução".

3. For function 4 what exactly is meant by any visited node. Is it a node that was poped from the `open_nodes` array or does a node already count as visited if it was added to the lnewnodes list?

   **Resposta**: Yes, by visited note we mean a node poped from the queue.

4. Na pesquisa com `improve=True`, se for encontrado um nó solução com função A* superior à solução atual, esse nó deve ser contabilizado para `num_solution` ou `num_skipped` ?

   **Resposta**: Se for encontrado um nó solução, esse nó deve ser contabilizado para `num_solution`, claro; se a função de avaliação for menor do que a melhor solução até ao momento, deve actualizar a melhor solução.

5. No ex4 fazemos sort pelo nome(state) como terceiro critério de desempate, mas que nome devemos usar no caso do blocksWorld? O state dos nos da tree no caso do `BlocksWorld`, são do género: {Floor(b), Floor(d), Floor(a), Free(e), Free(c), Free(b), HandFree(),On(e,a),On(c,d)}

   **Resposta**: Para os estados do mundo dos blocos, convertem o estado todo para cadeia de caracteres e continuam a funcionar com a ordem alfabetica.

6. As minhas heurísticas iniciais são iguais às do professor, assim como todos os exercícios que envolviam o A*, no entanto, nenhum dos meus resultados referentes ao blocksWorld, são iguais aos dos exemplos. São constantemente inferiores, admissíveis e o planos são iguais.

   **Resposta**: Se funciona assim tão bem para todos os testes, óptimo. Veja no entanto quais poderão ser os pontos fracos, para o caso de eu colocar testes adicionais na correção do trabalho.

7. Na última questão, o valor da heurística proveniente no results.txt é meramente uma referência, certo? Pelo que entendi no enunciado, diz que, desde que a heurística seja admissível, para a solução ser ótima, o que importa é minimizar o número de nós abertos e fechados. Eu tenho valores diferentes para a heurística e path diferentes também, mas não quer dizer que esteja mal, certo?

   **Resposta**: Os valores não têm que ser iguais; interessa minimizar o tamanho da árvore, preservando a admissibilidade da heurística e a optimalidade das soluções.

8. Quanto é que o último exercício vale?

   **Resposta**: Aqui estão as cotações: 1 - 20%; 2 - 5%; 3 - 5%; 4 - 25%; 5 - 10%; 6 - 10%; 7 - 25%.

9. O que é a heuristica no contexto do exercicio 7, são os movimentos necessários para transformar o state no goal?

   **Resposta**: A heurística é uma estimativa do custo; como o custo de cada acção está definido como sendo 1, a sua interpretação está correcta.

10. No exercício 7, ser admissível ou não e o tamanho da árvore têm o mesmo peso na avaliação?

    **Resposta**: O tamanho da árvore (quanto menor melhor) tem um peso grande, desde que a heurística seja admissível; o critério exacto irei definir na altura da correcção. Se nao for admissivel, depende; aqui há uma variedade de casos, e terá que ser visto com bom senso.

11. Os exercícios do TPI têm uma cotação fixa máxima para cada um ou será avaliado apenas em função do desempenho geral?

    **Resposta**: todos os exercícios vão ser avaliados separadamente, de forma automática, tendo em conta vários critérios

12. No exercício 7, através do argumento `State` da função heurística, como é que eu poderia determinar se um bloco está por exemplo, simultaneamente no `Floor` e `Free`?

    **Resposta**: Para um dado bloco `x` que quer saber, tem que ver se existe no `State`, um elemento `p` tal que `isinstance(p,Floor) and p.args[0]==x`, etc.

13. Porque é que neste em alguns testes do mundo dos blocos, parte dos blocos desaparecem do objetivo?

    **Resposta**: A descrição do objectivo não tem que ser uma descrição exaustiva do estado final; refere apenas o que for relevante