Projeto Nº 1: Época Normal

Inteligência Artificial - Escola Superior de Tecnologia de Setúbal
2020/2021

Prof. Joaquim Filipe
Eng. Filipe Mariano

1. Problema do Quatro: Descrição

O projeto destina-se a resolver o problema baseado no jogo do Quatro. O jogo é constituído por um tabuleiro de 4 x 4 casas e um conjunto de peças (até 16 dependente dos problemas) que possuem traços característicos de forma e de cor: Existem peças brancas ou pretas; parte delas são altas, a outra parte são baixas; umas são quadradas outras são redondas. Algumas peças são cheias outras são ocas.

Figura 1: Exemplo do Problema do Quatro.

1.1. Tabuleiro

O jogo é constituído por um tabuleiro de 4 x 4 casas e um conjunto de 16 peças, que possuem traços característicos de forma e de cor. Todas as peças de jogo terão as seguintes características:

1. Branca ou Preta
2. Alta ou Baixa
3. Quadrada ou Redonda
4. Cheia ou Oca

O que significa que as 16 peças do jogo encontram-se divididas da seguinte forma, pelas características supramencionadas:

- Existem 8 peças brancas e 8 peças pretas
- Existem 8 peças altas e 8 peças baixas
- Existem 8 peças quadradas e 8 peças redondas
- Existem 8 peças cheias e 8 peças ocas

1.2. Solução para o Problema

Para resolver o problema, é preciso completar uma linha de 4 peças (na horizontal, na vertical ou na diagonal) que compartilhem pelo menos um traço em comum (4 peças pretas, ou 4 peças altas, ou 4 peças ocas, etc).

1.3. Operadores

Como operadores do problema, ou movimentos possíveis de realizar, existe apenas um único movimento possível que é a colocação de uma peça da reserva, i.e. por colocar, no tabuleiro.

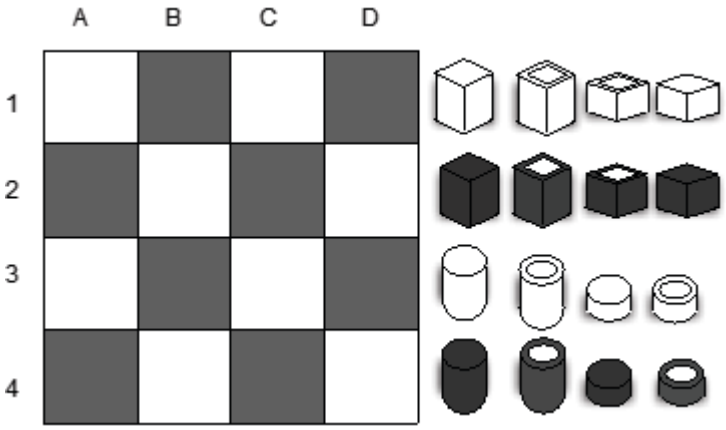

Figura 2: O tabuleiro vazio do problema "Quatro" com uma reserva de 16 peças.

2. Objetivo do Projeto

Pretende-se desenvolver um programa, em `LISP`, para indicar a sequência de passos que conduzem de uma posição inicial do problema até a uma posição final, em que existem 4 peças com características comuns alinhadas.

O principal objetivo consiste em explorar o espaço de possibilidades tentanto os vários caminhos possíveis até encontrar a solução, recorrendo aos algoritmos de procura lecionados nas aulas. Neste sentido, o intuito do projeto passa por testar diversos tabuleiros, aplicando métodos de procura em espaço de estados conforme se indica detalhadamente mais adiante, para apresentar a sequência de estados ou de jogadas que conduzem de uma posição inicial do puzzle até uma posição final.

3. Formulação do Problema

3.1. Tabuleiro

O tabuleiro é representado sob a forma de uma estrutura de `4 x 4` casas. A notação das colunas utiliza as letras `A, B, C, D` e a notação das linhas utiliza os números `1, 2, 3, 4`. Por exemplo, a casa B3 faz referência a casa situada na segunda coluna da terceira linha. A Figura 2 apresenta o tabuleiro vazio.

É assim possível representar em LISP o tabuleiro como uma lista de 4 listas compostas, cada uma composta por 4 elementos, em que cada elemento representa uma casa:

- O valor `0` (zero) representa uma casa vazia
- Uma lista de quatro elementos representa uma peça. Nesta lista, o **primeiro elemento representa a cor da peça** (`preta` ou `branca`), **o segundo elemento representa a forma da peça** (`redonda` ou `quadrada`), **o terceiro elemento representa a altura** (`alta` ou `baixa`) e **o quarto elemento representa a densidade** (`cheia` ou `oca`). Desta forma as 16 peças podem ser representadas com as 16 listas seguintes:

```
(branca redonda alta oca)
(preta redonda alta oca)
(branca redonda baixa oca)
(preta redonda baixa oca)
(branca quadrada alta oca)
(preta quadrada alta oca)
(branca quadrada baixa oca)
(preta quadrada baixa oca)
(branca redonda alta cheia)
(preta redonda alta cheia)
(branca redonda baixa cheia)
(preta redonda baixa cheia)
(branca quadrada alta cheia)
(preta quadrada alta cheia)
(branca quadrada baixa cheia)
(preta quadrada baixa cheia)
```

Tabuleiro sem reserva de peças:

Um exemplo de representação em LISP do tabuleiro vazio sem reserva de peças, seria da seguinte forma:

```
((0 0 0 0) (0 0 0 0) (0 0 0 0) (0 0 0 0))
```

Tabuleiro com reserva de peças:

Para representar um tabuleiro completo, é necessário incluir a lista de peças por colocar no tabuleiro (até 16 no inicio da resolução do problema). Desta forma um tabuleiro e a reserva de peças podem ser representados por uma lista de duas listas, na qual a primeira lista representa o tabuleiro e a segunda lista a reserva. Um exemplo de representação em LISP do tabuleiro vazio com reserva de peças, seria da seguinte forma:

```
(
    (
        (0 0 0 0) (0 0 0 0) (0 0 0 0) (0 0 0 0)
    )
    (
        (branca redonda alta oca)
        (preta redonda alta oca)
        (branca redonda baixa oca)
        (preta redonda baixa oca)
        (branca quadrada alta oca)
        (preta quadrada alta oca)
        (branca quadrada baixa oca)
        (preta quadrada baixa oca)
        (branca redonda alta cheia)
        (preta redonda alta cheia)
        (branca redonda baixa cheia)
        (preta redonda baixa cheia)
        (branca quadrada alta cheia)
```

```
        (preta quadrada alta cheia)
        (branca quadrada baixa cheia)
        (preta quadrada baixa cheia)
    )
)
```

3.2. Operadores

Cada movimento de colocação é composto por 4 elementos:

1. A coordenada `x` (coluna)
2. A coordenada `y` (linha)
3. A peça a colocar
4. O tabuleiro onde deve ser feita a colocação

Desta forma, se pretendemos colocar a peça (preta quadrada baixa cheia) na casa C2 do tabuleiro vazio, o operador a aplicar é:

```
(
    3
    2
    (preta quadrada baixa cheia)
    (
        (
            (0 0 0 0) (0 0 0 0) (0 0 0 0) (0 0 0 0)
        )
        (
            (branca redonda alta oca)
            (preta redonda alta oca)
            (branca redonda baixa oca)
            (preta redonda baixa oca)
            (branca quadrada alta oca)
            (preta quadrada alta oca)
            (branca quadrada baixa oca)
            (preta quadrada baixa oca)
            (branca redonda alta cheia)
            (preta redonda alta cheia)
            (branca redonda baixa cheia)
            (preta redonda baixa cheia)
            (branca quadrada alta cheia)
            (preta quadrada alta cheia)
            (branca quadrada baixa cheia)
            (preta quadrada baixa cheia)
        )
    )
)
```

O tabuleiro resultante será:

```
(
    (
        (0 0 0 0)
        (0 0 (preta quadrada baixa cheia) 0)
        (0 0 0 0)
        (0 0 0 0)
    )
    (
        (branca redonda alta oca)
        (preta redonda alta oca)
        (branca redonda baixa oca)
        (preta redonda baixa oca)
        (branca quadrada alta oca)
        (preta quadrada alta oca)
        (branca quadrada baixa oca)
        (preta quadrada baixa oca)
        (branca redonda alta cheia)
        (preta redonda alta cheia)
        (branca redonda baixa cheia)
        (preta redonda baixa cheia)
        (branca quadrada alta cheia)
        (preta quadrada alta cheia)
        (branca quadrada baixa cheia)
    )
)
```

Um movimento de colocação é considerado possível se:

- A casa de destino pertencer ao tabuleiro
- Não houver nenhuma peça nessa casa

3.3. Solução do Problema

A solução pode representar-se por uma sequência de operações realizadas sobre as peças da reserva, indicando as coordenadas da casa e a peça da reserva a colocar no tabuleiro.

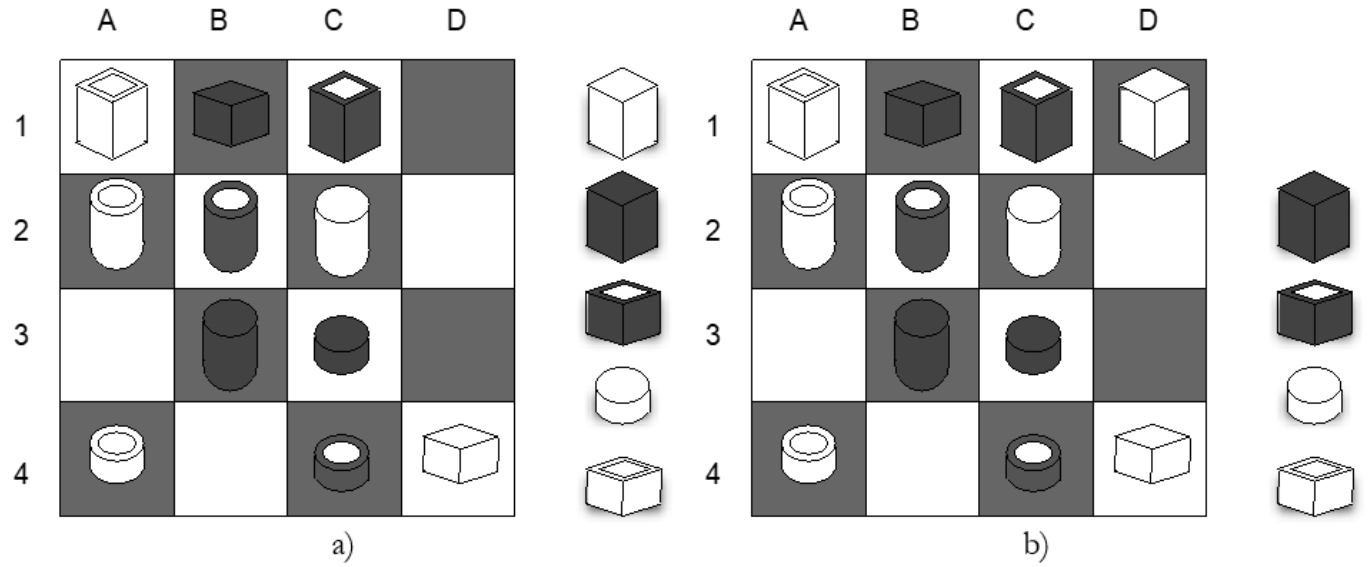

Figura 3: Exemplo de tabuleiro do problema do Quatro (a) e a sua respetiva solução (b).

Por exemplo, é possível encontrar uma das soluções para o problema da Figura 3 através da aplicação do seguinte operador:

```
(
    4
    2
    (branca quadrada alta cheia)
    (
        (
            ((branca quadrada alta oca) (preta quadrada baixa cheia) (preta
quadrada alta oca) 0)
            ((branca redonda alta oca) (preta redonda alta oca) (branca redonda
alta cheia) 0)
            (0 (preta redonda alta cheia) (preta redonda baixa cheia) (preta
quadrada baixa oca))
            ((branca redonda baixa oca) 0 (preta redonda baixa oca) 0)
        )
        (
            (branca quadrada alta cheia)(preta quadrada alta cheia)(branca
quadrada baixa cheia) (branca quadrada baixa oca) (branca redonda baixa cheia)
        )
    )
)
```

3.4. Estrutura do Programa

O programa deverá estar dividido em três partes, cada uma num ficheiro diferente:

1. Uma parte com a implementação dos métodos de procura, de forma independente do domínio de aplicação;
2. Outra para implementar a resolução do problema, incluindo a definição dos operadores e heurísticas, específicos do domínio de aplicação;
3. E a terceira parte para fazer a interação com o utilizador e para proceder à escrita e leitura de ficheiros.

Enquanto a primeira parte do programa deverá ser genérica para qualquer problema que possa ser resolvido com base no método de procura selecionado, a segunda parte é específico do domínio de aplicação, nomeadamente o problema do Quatro.

O projeto deverá apresentar um estudo comparativo do comportamento dos três métodos: procura em largura (BFS), procura em profundidade (DFS) e A*.

Para além destas três formas de procura, cada grupo pode programar, aplicar e estudar os algoritmos Simplified Memory A* (SMA*), Interactive Deepening A* (IDA*) e Recursive BestFirst Search (RBFS), que representarão um bónus para quem os implementar (ver Secção 7).

No caso dos métodos informados, o programa deverá utilizar funções heurísticas modulares, ou seja, que possam ser colocadas ou retiradas do programa de procura como módulos.

As heurísticas não devem estar embutidas de forma rígida no programa de procura. Exige–se a utilização de duas heurísticas, uma fornecida no fim do presente documento e outra desenvolvida pelos alunos.

O projeto deverá incluir a implementação de cada um dos métodos, de forma modular, permitindo que o utilizador escolha qualquer um deles, conjuntamente com os seus parâmetros (heurística, profundidade, etc.) para a resolução de um dado problema.

4. Experiências

Pretende-se que o projeto estude, para cada problema fornecido em anexo, o desempenho de cada algoritmo e, no caso dos algoritmos de procura informados, de cada uma das heurísticas propostas, apresentando, em relação a cada problema, a solução encontrada e dados estatísticos sobre a sua eficiência, nomeadamente o fator de ramificação média, o número de nós gerados, número de nós expandidos, a penetrância e o tempo de execução.

Os projetos deverão apresentar os dados acima referidos num ficheiro produzido automaticamente pelo programa, sendo descontado `0,5` valor por cada problema não resolvido. No caso de ser apresentada a solução, mas não o estudo de desempenho das heurísticas o desconto é de apenas `0,2` valor por cada caso.

Estes problemas deverão estar num ficheiro `problemas.dat`, utilizando a notação atrás indicada. O último problema (`G`) será apresentado durante a avaliação oral e inserido no ficheiro `problemas.dat` para verificar o funcionamento do projeto.

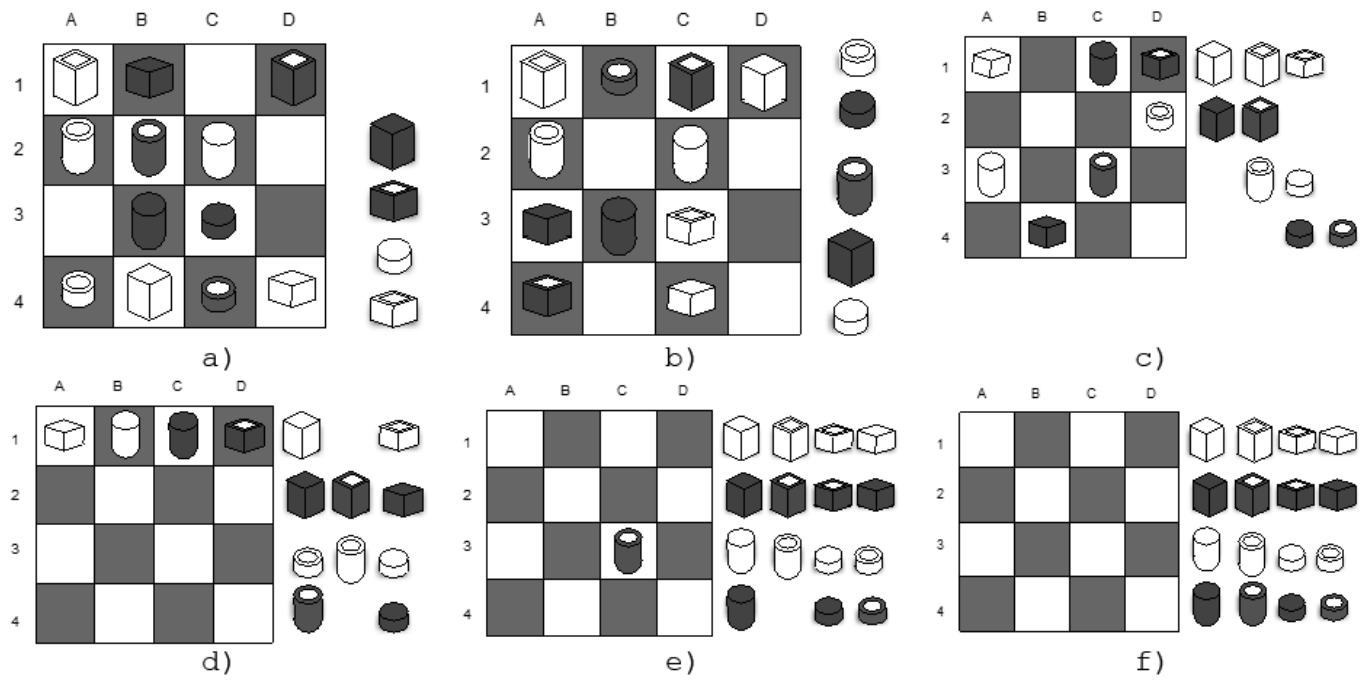

Figura 4: Conjunto de problemas de teste.

5. Heurísticas

Sugere-se usar como heurística de base, uma heurística que determina o número de peças já alinhadas no tabuleiro. Para um determinado tabuleiro x:

$h(x) = 4 - p(x)$

em que `p(x)` é o valor máximo do alinhamento de peças tomando em conta todas as possibilidades em termos de direção e características das peças, i.e., o número máximo de peças com características comuns já alinhadas na horizontal, na diagonal ou na vertical.

Esta heurística pode ser melhorada para refletir de forma mais adequada o conhecimento acerca do puzzle e assim contribuir para uma maior eficiência dos algoritmos de procura informados. Além da heurística acima sugerida, deve ser definida pelo menos uma segunda heurística que deverá melhorar o desempenho dos algoritmos de procura informados em relação à primeira fornecida.

6. Grupos

Os projetos deverão ser realizados em grupos de, no máximo, duas pessoas sendo contudo sempre sujeitos a avaliação oral individual para confirmação da capacidade de compreensão dos algoritmos e de desenvolvimento de código em LISP.

O grupo poderá ser constituido por alunos que frequentam turmas ou turnos diferentes.

7. Bónus

O projeto inclui uma parte opcional que permite atribuir um bónus aos alunos que a conseguirem implementar.

Para além dos três métodos de procura anteriormente mencionados, cada grupo tem a opção de programar, aplicar e estudar uma ou mais das seguintes estratégias *Simplified Memory–Bounded A** (SMA*), *Iterative Deepening A** (IDA*) ou *Recursive Best First Search* (RBFS).

A programação e estudo dos métodos opcionais vale 1 valor cada, a somar ao valor total do projeto, sendo a nota final do projeto limitada ao máximo de 20 valores.

8. Datas

Entrega do projeto: 21 de Dezembro de 2020, até as 23:00.
Discussão do projeto: Inicio de Fevereiro de 2021, após a entrega da 2ª fase do projeto.

9. Documentação a Entregar

A entrega do projeto e da respetiva documentação deverá ser feita através do Moodle, na zona do evento "Entrega do 1º Projeto". Todos os ficheiros a entregar deverão ser devidamente arquivados num ficheiro comprimido (`ZIP` com um tamanho máximo de 5Mb), até à data acima indicada. O nome do arquivo deve seguir a estrutura `nomeAluno1_numeroAluno1_nomeAluno2_numeroAluno2_P1`.

9.1. Código Fonte

Os ficheiros de código devem ser devidamente comentados e organizados da seguinte forma:

projeto.lisp Carrega os outros ficheiros de código, escreve e lê ficheiros, e trata da interação com o utilizador.

puzzle.lisp Código relacionado com o problema.

procura.lisp Deve conter a implementação de:

1. Algoritmo de Procura de Largura Primeiro (BFS)
2. Algoritmo de Procura do Profundidade Primeiro (DFS)
3. Algoritmo de Procura do Melhor Primeiro (A*)
4. Os algoritmos SMA*, IDA* e/ou RBFS (caso optem por implementar o bónus)

9.2. Exercícios

Deverá haver um ficheiro de exercícios, com a designação `problemas.dat`, contendo todos os exemplos de tabuleiro que se quiser fornecer ao utilizador, organizados de forma sequencial, e que este deverá escolher mediante um número inserido no interface com o utilizador.

Esse número representa o número de ordem na sequência de exemplos. O ficheiro deverá ter várias listas, separadas umas das outras por um separador legal, e não uma lista de listas. Essas listas serão tantas quantos os problemas fornecidos.

Na oral, os docentes irão solicitar que se adicione mais um exemplo, numa dada posição do ficheiro, que deverá imediatamente passar a ser selecionável através do interface com o utilizador e ser resolvido normalmente.

9.3. Manuais

No âmbito da Unidade Curricular de Inteligência Artificial pretende–se que os alunos pratiquem a escrita de documentos recorrendo à linguagem de marcação ***Markdown***, que é amplamente utilizada para os ficheiros ***ReadMe*** no ***GitHub***. Na Secção 4 do guia de Laboratório nº 2, encontrará toda a informação relativa à estrutura recomendada e sugestões de ferramentas de edição para ***Markdown***.

Para além de entregar os ficheiros de código e de problemas, é necessário elaborar e entregar 2 manuais (o manual de utilizador e o manual técnico), em formato PDF juntamente com os sources em MD, incluídos no arquivo acima referido.

Manual Técnico:

O Manual Técnico deverá conter o algoritmo geral, por partes e devidamente comentado; descrição dos objetos que compõem o projeto, incluindo dados e procedimentos; identificação das limitações e opções técnicas. Deverá ser apresentada uma análise critica dos resultados das execuções do programa, onde deverá transparecer a compreensão das limitações do projeto. Deverão usar uma análise comparativa do conjunto de execuções do programa para cada algoritmo e cada problema, permitindo verificar o desempenho de cada algoritmo e das heurísticas. Deverá, por fim, apresentar a lista dos requisitos do projeto (listados neste documento) que não foram implementados.

Manual de Utilizador:

O Manual do Utilizador deverá conter a identificação dos objetivos do programa, juntamente com descrição geral do seu funcionamento; explicação da forma como se usa o programa (acompanhada de exemplos); descrição da informação necessária e da informação produzida (ecrã/teclado e ficheiros); limitações do programa (do ponto de vista do utilizador, de natureza não técnica).

10. Avaliação

Tabela 1: Grelha de classificação.

Funcionalidade	**Valores**
Representação do estado e operadores	2
Funções referentes ao puzzle	3
Procura em profundidade e largura	2.5
Procura com A* e heurística dada	2.5
Implementação de nova heurística	1.5
Resolução dos problemas a) a f)	3
Resolução do problema g) (dado na avalição oral)	0.5
Qualidade do código	2
Interação com o utilizador	1
Manuais (utilizador e técnico)	2
Total	20
Bónus	3

O valor máximo da nota são 20 valores, já com a integração do valor do bónus.

Os problemas a) a f) estão descritos na Secção Experiências, enquanto que o problema g) será facultado durante a avaliação oral. A avaliação do projeto levará em linha de conta os seguintes aspectos:

- Data de entrega final – Existe uma tolerância de 4 dias em relação ao prazo de entrega, com a penalização de 1 valor por cada dia de atraso. Findo este período a nota do projeto será 0.
- Correção processual da entrega do projeto – (Moodle; manuais no formato correto). Anomalias processuais darão origem a uma penalização que pode ir até 3 valores.
- Qualidade técnica – Objetivos atingidos; Código correto; Facilidade de leitura e manutenção do programa; Opções técnicas corretas.
- Qualidade da documentação – Estrutura e conteúdo dos manuais que acompanham o projeto.
- Avaliação oral – Eficácia e eficiência da exposição; Compreensão das limitações e possibilidades de desenvolvimento do programa. Nesta fase poderá haver lugar a uma revisão total da nota de projeto.

11. Recomendações Finais

Com este projeto pretende–se motivar o paradigma de programação funcional. A utilização de variáveis globais, de instruções de atribuição do tipo set, setq, setf, de ciclos, de funções destrutivas ou de quaisquer funções com efeitos laterais é fortemente desincentivada dado que denota normalmente uma baixa qualidade técnica.

As únicas excepções permitidas a estas regras poderão ser a utilização da instrução loop para implementar os ciclos principais dos algoritmos implementados (como alternativa a uma solução puramente recursiva), em conjugação com as variáveis globais Abertos e Fechados para manutenção das listas de nós.

ATENÇÃO: Suspeitas confirmadas de plágio serão penalizadas com a anulação de ambos os projetos envolvidos (fonte e destino), e os responsáveis ficam sujeitos à instauração de processo disciplinar.