DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Algoritmos e Estruturas de Dados

Licenciatura em Engenharia Electrónica

2022/23 (2º Sem. P4)

Relatório do Projecto

PADRÃO SECRETO

Grupo de Projecto nº: 07

Número	Nome
105917	Guilherme Garcia
105988	Mariana Domingos
106963	Rodrigo Caldeira

Docente: Prof. Paulo Flores

INSTITUTO SUPERIOR TÉCNICO
Departamento de Engenharia Electrotécnica e de Computadores

Índice

1. O Problema

Um padrão secreto encontra-se codificado por um conjunto de números obtidos a partir de um padrão original descodificado. O padrão é representado por um conjunto de pontos numa matriz com L linhas e C colunas, com um total de L×C pontos.

Pretende-se então desenvolver um programa que descodifique um padrão secreto que aceita como entrada, através de um ficheiro, o tamanho do padrão e uma codificação que permitirá obter o padrão descodificado.

O resultado final será então escrito num ficheiro de saída com a seguinte representação: um caracter ’-’ (menos) nos quadrados não pintados e um ’#’ (cardinal) nos quadrados pintados.

2. A Solução

Após uma análise ponderada, o nosso grupo decidiu implementar um programa que resolve o problema com o seguinte método:

O programa começa por verificar (para cada linha e tendo em conta a dimensão da “peça”) quantas possibilidades tem de colocar essa mesma peça sendo considerado como uma peça o conjunto de quadrados adjacentes de cada linha.

Prossegue para o ciclo principal onde, começando na primeira linha, avalia a primeira possibilidade verificando se a mesma é válida. Se o for, coloca a peça nessa posição e passa à próxima linha, caso contrário, tenta a próxima posição até não existirem mais possibilidades. Se o programa não encontrar uma posição válida na linha em que está, este volta à linha anterior e muda a peça (desta linha) para a posição válida seguinte de modo a conseguir encontrar então uma posição válida para a linha inicial em que se encontrava antes de voltar atrás.

Este processo repete-se até uma de duas coisas acontecerem: todas as linhas terem sido avaliadas e terem peças OU não ser possível resolver o problema dado e o programa ter saído do ciclo.

O programa termina de duas maneiras: se a codificação foi descodificada, prossegue para escrever o padrão descodificado (a solução) no ficheiro de saída OU quando não encontra nenhuma solução, prossegue para escrever 0 no ficheiro de saída.

3. Descrição da arquitetura do programa

3.1. Subsistemas e lógica do programa

A estrutura do programa está dividida em 4 subsistemas:

- Leitura e obtenção de dados iniciais a partir do ficheiro que contém codificação do padrão;
- Cálculo das possibilidades de colocar cada conjunto de quadrados na respectiva linha ou coluna;
- Ciclo principal responsável pela procura da solução através dum algoritmo do tipo DPS (Depth First Search) pré-fixado e com recurso a uma pilha onde serão guardadas as informações necessárias para obter a solução final;
- Escrita da solução final no ficheiro de saída (criado pelo programa com o nome adequado a cada padrão fornecido)

Leitura e obtenção de dados iniciais: Esta tarefa é a primeira a ser realizada visto que, obviamente, sem os dados iniciais o processo de descodificação do padrão não pode ser iniciado.

Esta leitura é realizada com recurso a 3 passos:

1. Abertura do ficheiro que contém codificação (main);
2. Leitura da primeira linha do ficheiro (contém dimensão do padrão) de modo a poder criar estruturas de dados com dimensão adequada (main);
3. Leitura do resto da codificação (read_file);

Cálculo das possibilidades: Fundamental para execução do algoritmo de procura de solução. As possibilidades são calculadas com base em duas fórmulas análogas que foram deduzidas de forma prática com recurso a exemplos (possibility_calculator).

$$PR = Nº\ Colunas - Nº\ quadrados\ pintados\ da\ linha + 1$$
$$PC = Nº\ Linhas - Nº\ quadrados\ pintados\ da\ coluna + 1$$

Exemplo (linha com 2 quadrados pintados parte de um padrão de 5 colunas):

$$PL = 5 - 2 + 1 = 4$$

Nota: Durante a realização deste relatório apercebemo-nos de que o cálculo de PC poderia ser removido contribuindo para a otimização do programa visto que, como o programa coloca as peças linha-a-linha, os valores obtidos e guardados nesta estrutura de dados são irrelevantes para a resolução do problema.

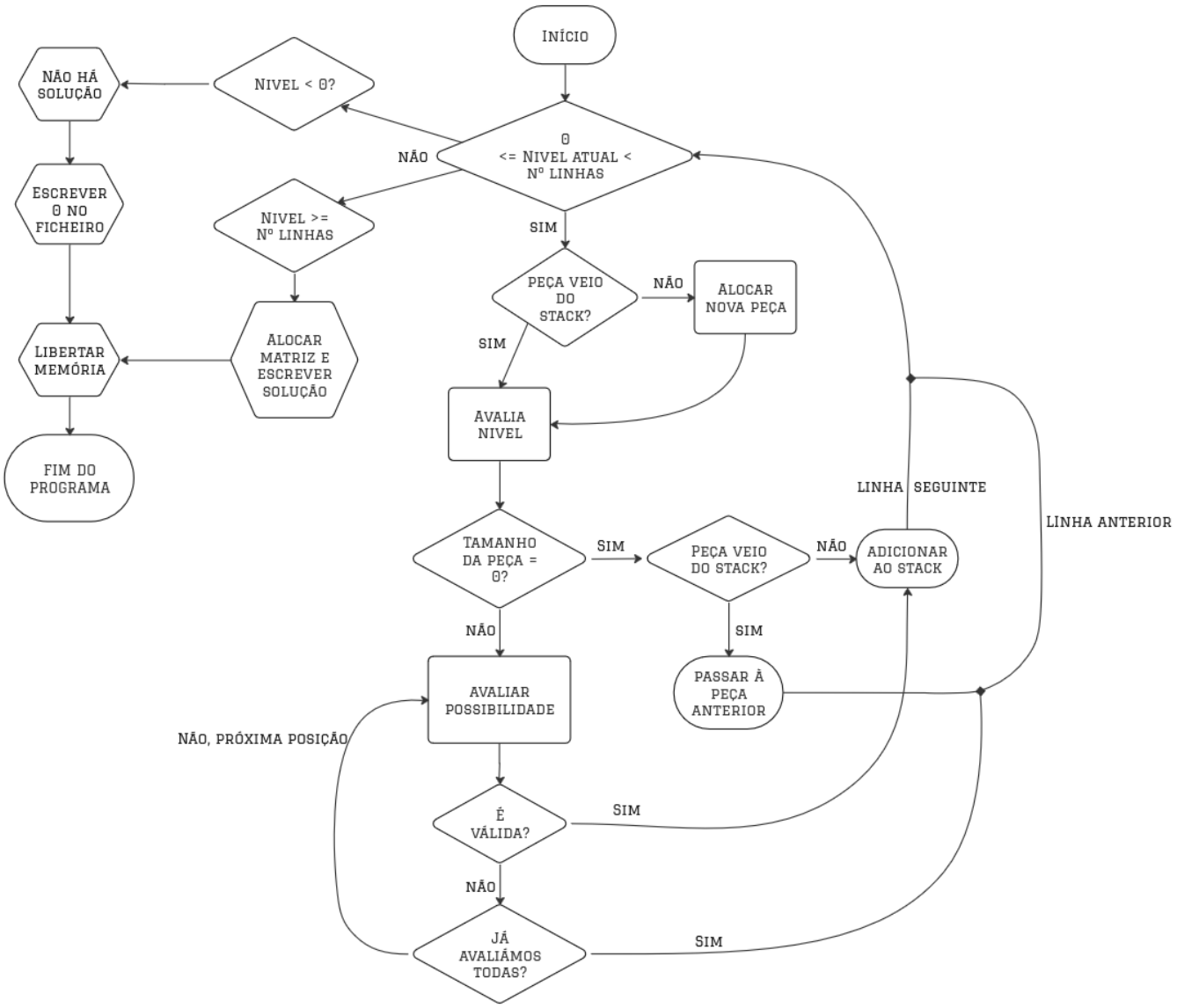
Ciclo principal: Essência do processo de resolução, é o algoritmo que permite com base na codificação, obter o padrão. O modo como tal é feito pode ser estudado com mais detalhe através do fluxograma presente na secção 3.1.1. (Funções relevantes: pieceCreator, add2Stack, evaluatePosition, updateState).

Escrita da solução: Fase final, consiste em "montar" o padrão numa matriz de dimensão LxC partindo das L peças que foram inseridas no stack durante o processo de resolução.

Após percorrer todo o stack, o padrão fica completo e a matriz pode ser percorrida "quadrado-a-quadrado", sendo cada quadrado escrito no ficheiro de saída que pode depois ser visualizado pelo utilizador sendo essa a representação do padrão descodificado (solutionWriter).

Apesar de não ser considerada um subsistema é importante salientar que existe uma fase final extremamente importante que consiste em fechar os ficheiros abertos e libertar toda a memória utilizada pelo programa (stackRelease/main).

3.1.1. Fluxograma



3.2. Estruturas de dados

Durante a projecção do programa fomos utilizando diferentes tipos de estruturas de dados. Para lermos os dados do ficheiro de entrada utilizamos dois vetores que se chamam v_rows, e v_columns, que guardam o valor do número de quadrados preenchidos por linha e por coluna, respectivamente.
Para o cálculo das possibilidades existentes em cada linha e em cada coluna tivemos a necessidade de criar outros dois vetores, o PR que guarda as possibilidades das linhas e o PC que guarda as possibilidades das colunas.

No processo de descodificação sentimos a necessidade da criação de um vetor que guardasse quantos quadrados faltavam pintar em cada coluna, e por isso, criámos o vetor squareCounter. Criámos ainda um stack que guardasse a posição da última peça e o seu tamanho para sempre que o programa chegasse ao ponto de não haver mais opções o mesmo conseguir ir buscar ao stack a posição da peça anterior e alterar essa posição para a próxima possível.

Para escrevermos a solução no ficheiro de saída tivemos de criar uma matriz para conseguirmos a representação pedida.

3.3. Algoritmos

O algoritmo utilizado na procura da solução do padrão é do tipo Depth-First-Search pŕe-fixado, realizando a procura duma solução (caminho completo) explorando as subárvores em função da validez de cada vértice.

Desta maneira, as opções todas podem ser interpretadas como uma grande árvore em que cada nível da árvore é uma linha, cada vértice uma possibilidade de colocar a peça dessa linha e cada caminho uma potencial solução.

4. Análise de requisitos computacionais

Melhor caso: Não é preciso ir buscar peças anteriores ao stack visto que há sempre uma posição válida em cada linha (preferencialmente umas das primeiras possibilidades) tendo em conta o caminho escolhido.

Pior caso: Para cada linha (exceto a 1ª) temos de testar todas as posições possíveis para uma peça sendo apenas a última destas válida. Ao chegar à última linha verificar que nenhuma

possibilidade era válida tendo em conta o caminho anteriormente escolhido e, por isso, ter de voltar atrás no stack até à 1ª linha repetindo este processo para cada possibilidade até, finalmente, detectar que não existe solução possível.

Memória: Tendo em contas os dados a que temos acesso, o valor da memória utilizada durante a execução do programa foi de cerca de 308 KBytes. Durante a análise dos valores obtidos pelo nosso grupo e por outros grupos, ficamos a perceber que a quantidade de memória gasta pelo nosso programa acabou por ser um valor baixo levando a que concluímos que o programa foi bastante eficiente nesse campo.

Complexidade temporal: O algoritmo concebido tem um tempo de execução dependente de um único parâmetro N (dimensão da matriz). Após a elaborar o algoritmo, realizamos uma análise de acordo com o estudado na aula e chegámos à conclusão que o nosso programa tem uma complexidade de $N \log N$ em que N é a dimensão da matriz.

5. Exemplo de aplicação

Anteriormente foi explicado a maneira de funcionamento do programa, agora vamos apresentar alguns exemplos da sua atuação, mostraremos os dados de entrada e os respectivos dados de saída esperados e o descodificado pelo nosso programa.

```
5 8
1
2
4
3
1
1 3 2 1 0 1 2 1
```

```
-#------
-##-----
####----
-----###
------#-
```

```
-#------
-##-----
####----
-----###
------#-
```

Dados de entrada ex. 1 | Dados de saída esperados ex. 1 | Dados de saída do programa ex. 1

Como pudemos verificar em cima o nosso programa cumpriu os requisitos pedidos e conseguiu descodificar este exemplo sem problemas.

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

Dados de entrada ex. 2 | Dados de saída esperados ex. 2 | Dados de saída do programa ex. 2

Como pudemos verificar em cima o nosso programa cumpriu os requisitos pedidos e conseguiu descodificar este exemplo, mas ao contrário do exemplo 1 a descodificação deste código não ocorreu sem problemas inicialmente, visto que quando o nosso programa era posto à prova este falhava e apresentava uma mensagem de erro: seg fault. Viemos a apercebemo-nos que este erro acontecia porque o ciclo da main não conseguia ultrapassar certos obstáculos, o que nos levou a ter que alterar o fluxo principal do algoritmo de resolução.

5	gp7	0/5	06-16 22:06	00935569	{Time Limit Exceeded}	13	60	59995	308

Após submetermos o nosso programa no Mooshak obtivemos estas informações de tempo, uso de memória, etc. Ao analisarmos os valores, verificamos que em termos de quantidade de memória utilizada estávamos bem, e que o nosso principal problema era não ser rápido o suficiente a resolver as codificações e por este motivo o nosso programa não foi capaz de descodificar todos os padrões usados na testagem realizada pelo sistema automático de avaliação.