UNIVERSIDADE DE ÉVORA

Programação Declarativa

Relatório do trabalho prático

Autores:
Tiago Martinho, 35735
Petersen Figueira, 39022

Docente:
Salvador Abreu

Janeiro de 2020

Índice

1 Introdução **2**

2 *Nonograma* **2**

3 Desenvolvimento **3**
3.1 Verificar uma linha 3
3.2 Fazer a permutação de uma linha 5
3.2.1 Apenas uma pista 5
3.2.2 Mais que uma pista 6
3.3 Escolha das permutações 7
3.4 Testar permutações 7

4 Conclusão **8**

5 Espaço para melhorias **8**

6 Referências **9**

1 Introdução

No âmbito da unidade curricular de Programação Declarativa, pretende-se por em prática conhecimentos de programação funcional e de programação lógica. Com este propósito o grupo foi encarregue de desenvolver um programa capaz de resolver ***Nonogramas*** usando uma das seguintes linguagens: *Ocaml* ou *Prolog*. Este trabalho foi realizado em *Ocaml* pois foi a última matéria a ser leccionada na disciplina e consequentemente a linguagem com que o grupo estaria mais confortável.

2 *Nonograma*

Nonograma é um puzzle de origem japonesa que consiste em preencher uma grelha com blocos coloridos consoantes as pistas dadas. Existem dois tipos de nonogramas: a cores e a preto e branco, neste trabalho iremos apenas resolver nonogramas a **preto e branco**. A grelha a ser pintada pode variar de tamanho e pode nem ser quadrada.
Os nonogramas foram criados no fim dos anos 80 e foram principalmente divulgados em revistas de puzzles japoneses, mais tarde foram adoptados para o novo meio tecnológico que é os *video games*.
As pistas dadas para resolver um nonograma são numéricas e são referentes à coluna ou à linha. Cada número representa quantos **blocos coloridos** devem existir na linha ou coluna respectiva, sendo que entre cada conjunto de blocos pintados deve existir pelo menos **um espaço em branco**.

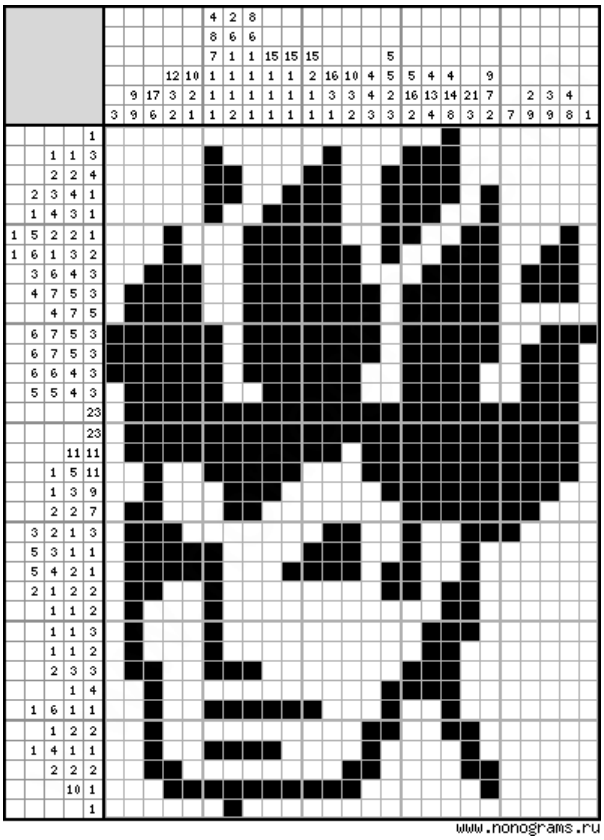

Figura 1: Nonograma a preto e branco

3 Desenvolvimento

A táctita abordada foi uma de ***Depth-first Search*** ou seja de *Bruteforce*. Em prática o programa faz todas as possíveis resposta para cada linha, de seguida faz as combinações de todas as linhas e verifica se a solução proposta é a correcta para as pistas das colunas.

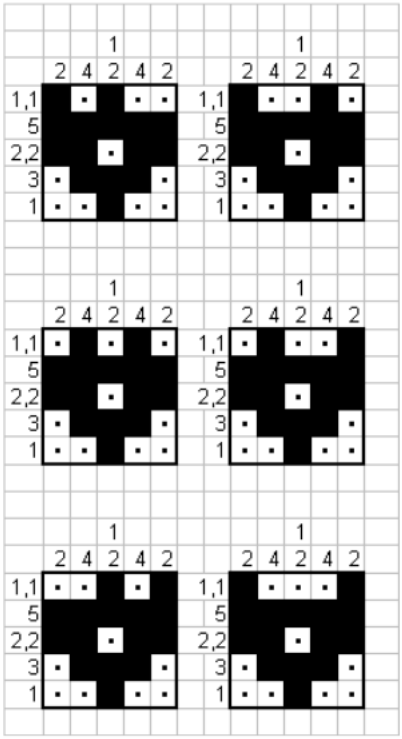

Figura 2: Resultados do *Bruteforce*

3.1 Verificar uma linha

Para verificar uma linha foi usada uma abordagem lógica. Sabemos que uma solução possível **começa** com um número indeterminado de espaços vazios, que é **seguido** de um número definido de blocos pintados e, caso existam mais pistas terá que ter um número **maior** que 1 de espaços vazios, caso seja a última pista sabemos que o espaço restante terá que ter **apenas** espaços vazios seguidos dos blocos preenchidos.
Isto numa linguagem matemática traduz-se para:

- Caso haja mais que uma pista: **(0, n)** espaços vazios :: **(1, pista)** espaços preenchidos :: **(1, m)** espaços vazios :: ...
- Caso exista apenas uma pista: **(0, n)** espaços vazios :: **(1, pista)** :: **(0, m)** espaços vazios.

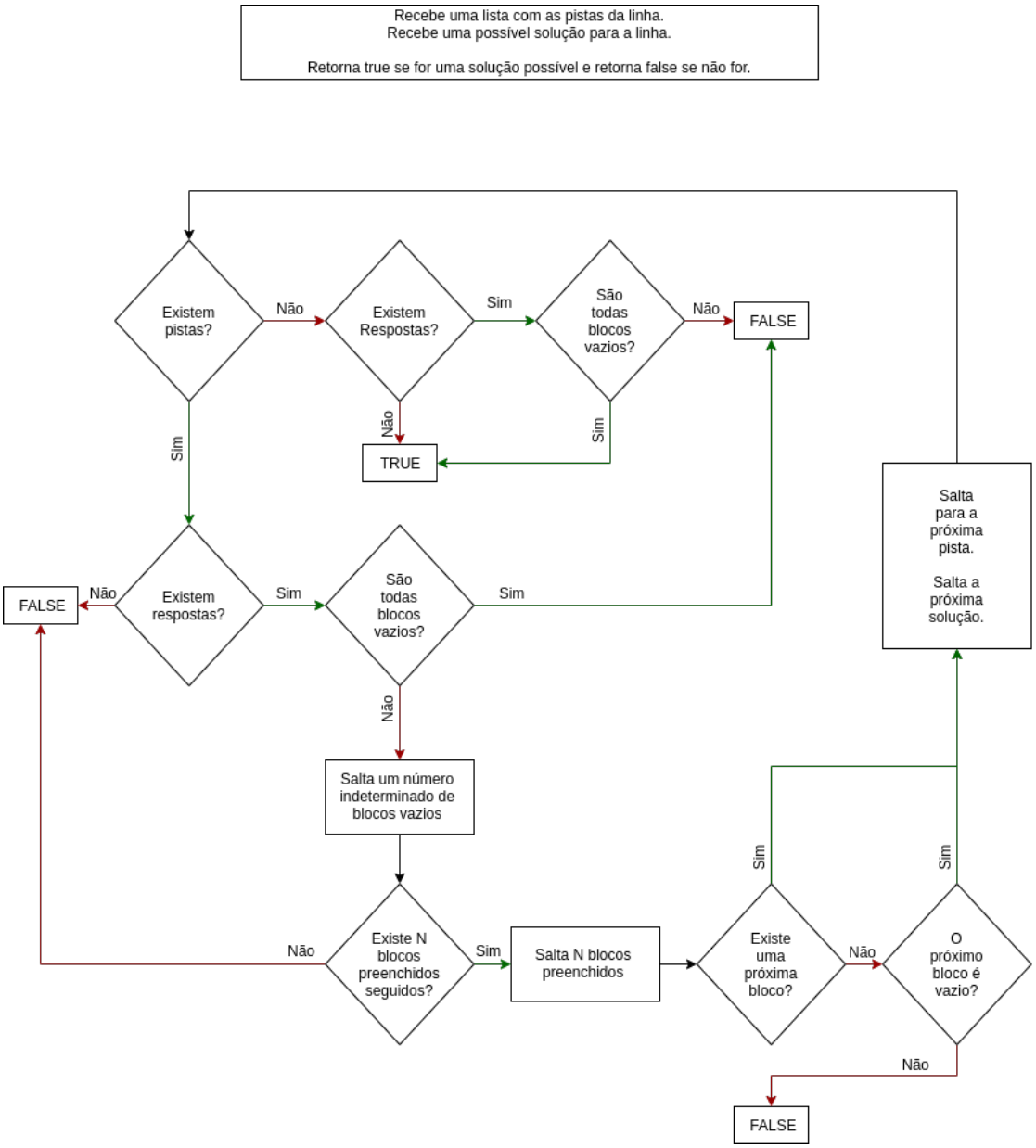

Figura 3: Diagrama da verificação de linha

3.2 Fazer a permutação de uma linha

Para fazer **todas as possibilidades** de uma linha dividiu-se em **dois estados**. Existe o caso de haver **apenas uma pista** ou de **múltiplas pistas**. No caso de múltiplas pistas segue-se as pistas até existir **apenas uma pista** e depois concatena-se essas possibilidades às **permutações anteriores**.

3.2.1 Apenas uma pista

Para fazer as permutações de **apenas uma pista**, como é o caso da figura 4 para a pista 1 numa tabela com 5 espaços, usa-se um **contador** que indica qual o valor onde a pista deve **começar** e preenche-se o espaço anterior com blocos **vazios** seguido do número da pista de blocos **preenchidos** e os **restantes** blocos com blocos vazios.

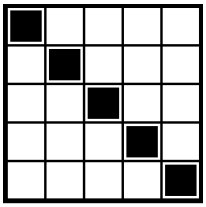

Figura 4: Possibilidades para a pista 1 numa tabela com 5 espaços

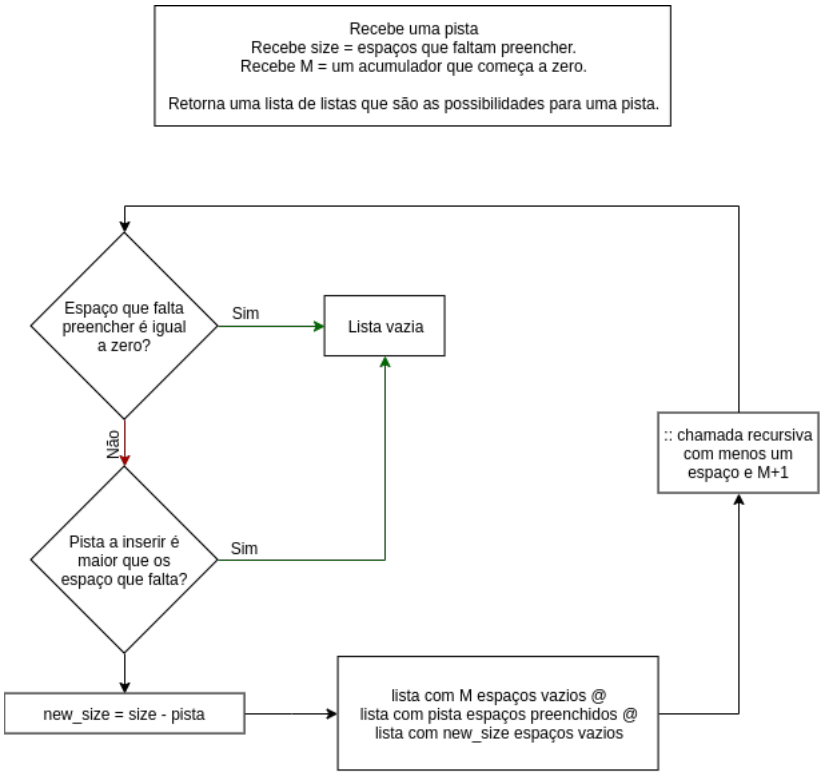

Figura 5: Diagrama da criação de permutações para apenas uma pista

3.2.2 Mais que uma pista

Para fazer as permutações de algo que tenha mais que uma pista usa-se um **acumulador** que incrementa um N, representativo dos espaços deixados em **vazio**, no inicio de cada pista, **segue-se** um número de espaços preenchidos igual ao da **pista atual** e um espaço vazio **obrigatório**. Calcula-se então uma lista de listas das **próximas** permutações das pistas seguintes e concatena-se a pista atual com as pistas seguintes.

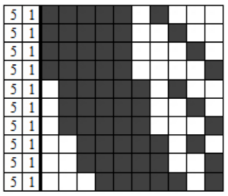

Figura 6: Permutações para uma pista 5 1 com 10 espaços totais

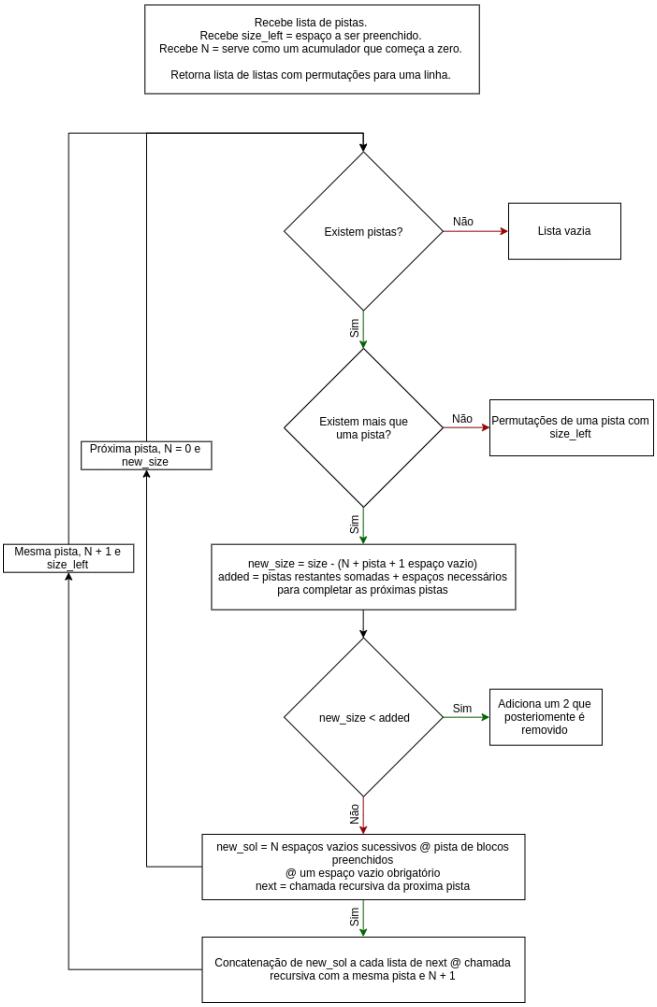

Figura 7: Diagrama das permutações com mais do que uma pista

3.3 Escolha das permutações

Para serem testadas **todas** as possibilidades decidiu-se fazer uma lista com todas as **escolhas possíveis** entre as permutações. Isto veio a tornar-se uma má decisão por parte do grupo perante o número de permutações possíveis.
Decidiu-se usar uma lista que guardaria as escolhas de permutações tomadas. Ou seja se a **permutação escolhida** fosse a primeira de todas num nonograma com 5 linhas seriam : [0; 0; 0; 0; 0].

3.4 Testar permutações

Para testar as permutações **junta-se** todas as linhas que poderão ser respostas numa só linha. A partir desta **linha** cria-se então uma **matriz transposta** que faz corresponder a solução a listas de colunas.
Finalmente é **testada** a matriz transposta com as pistas das **colunas** e caso todas as respostas sejam válidas a resposta é então imprimida em *standard output*.

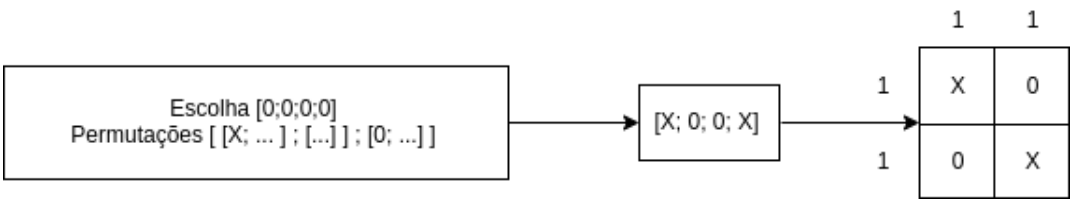

Figura 8: Passagem de escolha e de seguida a matriz transposta

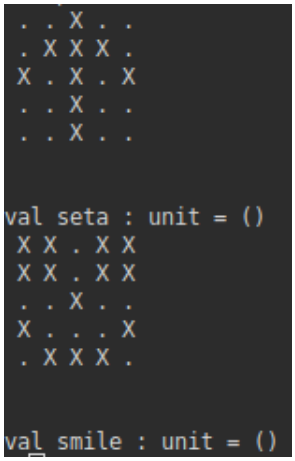

Figura 9: Output do programa

4 Conclusão

Ao longo do desenvolvimento deste trabalho o grupo foi ajustando perante as dificuldades encontradas.
Alcançou-se um limite de funcionalidade para nonogramas pequenos (5x6). Isto deve-se à paragem súbita do programa apresentando a mensagem *Stack overflow during evaluation.* Graças ao tamanho da lista de todas as escolhas possíveis ultrapassar o tamanho máximo da *Stack*. A solução proposta seria fazer permutações uma a uma e ir testando no mesmo ciclo.
Apesar de não serem abrangidos todos os nonogramas o grupo sente-se satisfeito ao apresentar um trabalho fruto de esforço e criatividade dos próprios.

5 Espaço para melhorias

Caso o grupo tivesse oportunidade de continuar o trabalho teria então aplicado as seguintes melhorias.

- Selecionar as pistas da coordenada com menor tamanho, ou seja, linhas ou colunas de modo a serem reduzidas as permutações de cada linha;
- Reduzir as possibilidades de solução verificando se a linha proposta como resposta é válida perante as pistas da coluna.

6 Referências

Para o desenvolvimento deste trabalho prático foram consultadas as seguintes páginas:

https://core.ac.uk/download/pdf/51698555.pdf
https://en.wikipedia.org/wiki/Nonogram#Solution_techniques