

UNIVERSIDADE LUSÓFONA DO PORTO

TRABALHO DE LÓGICA/GRAFOS

MATEMÁTICA DISCRETA

NUNO PONTES - A 2 1 2 0 0 3 7 3

1. INTRODUÇÃO

Este relatório foi realizado no âmbito da disciplina de Matemática Discreta, em que se elaborou um algoritmo capaz de discernir quais vértices (no contexto deste trabalho, são regiões) são adjacentes uns dos outros, e de acordo com a sua cor, identificar quais os que se encontram em conflito.

Tenho como objetivo encontrar um problema, e resolvê-lo através da lógica ou na representação de um mapa/gráfo. Inicialmente para este projeto pensei em projetar apenas o gráfo, atribuir um algoritmo na sua resolução, representar a minha solução e apresentar uma conclusão.

No entanto pensei em fazer uso do GNU Prolog, uma vez que já me foi lecionado na disciplina de Inteligência Artificial, terceiro ano. Para este caso em particular, achei que se enquadrava no contexto de trabalho, uma vez que o conceito dos grafos me lembrou um pouco o que foi lecionado nas aulas de IA, nomeadamente as árvores de família, o desafio dos vizinhos, e distâncias, entre outros algoritmos que aprendi na disciplina.

Posteriormente tenho também os resultados da experiência em questão, assim como a conclusão tirada.

2. CONTEXTO

O prolog é uma linguagem de programação declarativa, e ao invés do programa estipular a maneira de chegar à solução passo-a-passo, com acontece nas linguagens procedimentais ou orientadas a objetos, o mesmo fornece uma descrição do problema que se pretende resolver, utilizando lógica, que auxilia na resolução do problema proposto.

Outra coisa que a distingue das outras linguagens de programação, é de não ter estruturas de controlo (if-else, do-while, for, switch) que são comuns na maioria das linguagens de programação.

Neste contexto, o programa é executado em um modo interativo, posso efetivamente formular as queries, ou seja, as perguntas, utilizando fatos e regras para produzir a solução do mecanismo de unificação.

3. PROBLEMA

O problema diz respeito à coloração de regiões planas adjacentes. Assim como os mapas cartográficos, é necessário que, sejam quais fossem as cores usadas, duas regiões adjacentes não devem ter a mesma cor. Ou seja, duas regiões só são consideradas adjacentes quando compartilham algum segmento de linha, como o mapa demonstra na figura abaixo.

A ideal forma que encontrei para resolver o problema, é não ter a mesma cor em regiões adjacentes. Para tal optei por me inspirar no algoritmo de coloração lecionado em uma das aulas de matemática discreta, na parte teórica.

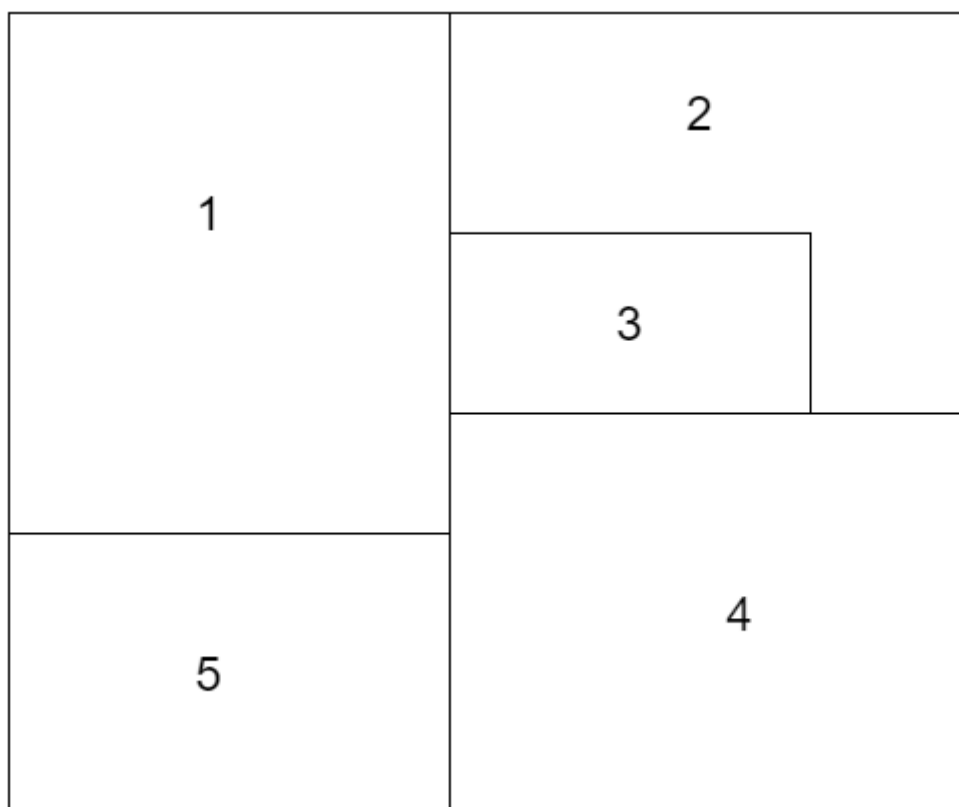


Figura 1 - Mapa

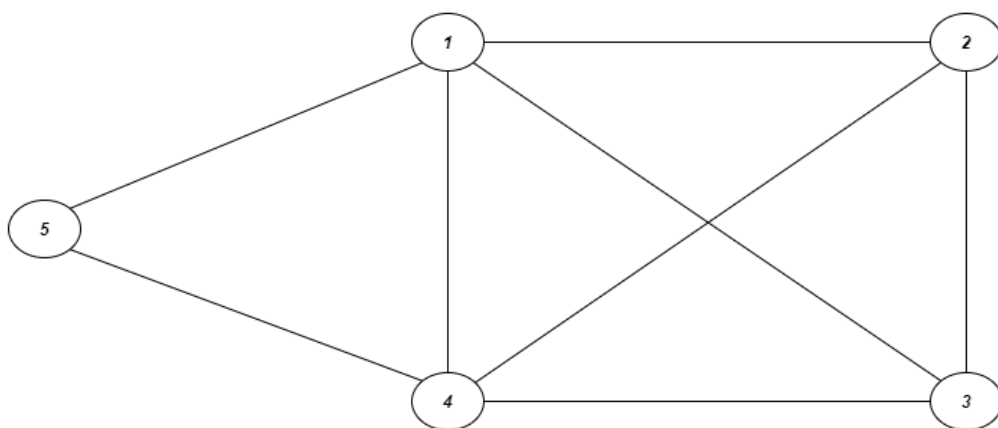


Figura 2 – Representação do Mapa em Grafo

Foram dados nomes numéricos a 5 regiões. Para representar quais as regiões adjacentes, tenho o seguinte grafo correspondente. Por exemplo tenho que a região 1, é adjacente da região (5, 2, 3, 4), no grafo comprova-se a minha afirmação, tal como no mapa correspondente.

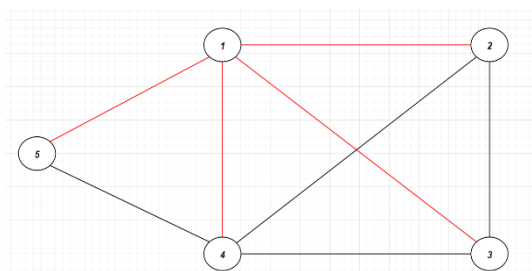


Figura 3 – Adjacência Vértice 1 (1, 2, 3, 4, 5)

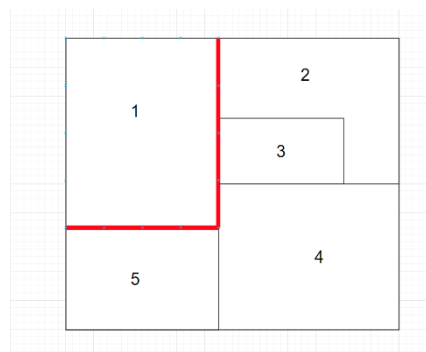
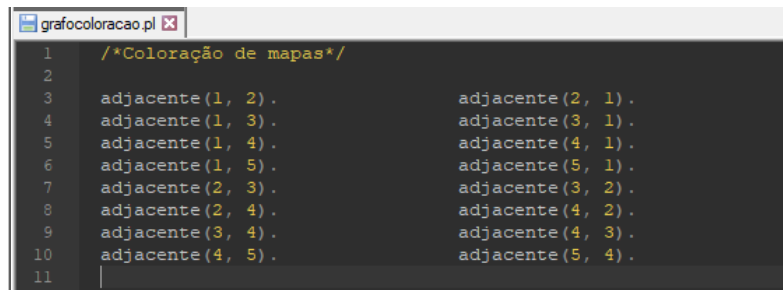


Figura 4 – Mapa Adjacência (1, 2, 3, 4, 5)

Outro exemplo é o 2, como a figura exemplifica, denota-se que o 2 não é adjacente com 5.

3.1 REPRESENTAÇÃO DO GRAFO

Para o mapa e grafo correspondente, optei por fazer uma representação do mesmo em prolog, como disse anteriormente. Onde adjacente representa a adjacência entre os vértices e, neste caso em concreto por exemplo, (1 – representa a região 1, 2 – representa a região 2) e (2 – representa a região 2, 1 – representa a região 1). Ou seja, serve para dizer que as regiões 1 e 2 são adjacentes de uma e outra.



```
1 /*Coloração de mapas*/
2
3 adjacente(1, 2).      adjacente(2, 1).
4 adjacente(1, 3).      adjacente(3, 1).
5 adjacente(1, 4).      adjacente(4, 1).
6 adjacente(1, 5).      adjacente(5, 1).
7 adjacente(2, 3).      adjacente(3, 2).
8 adjacente(2, 4).      adjacente(4, 2).
9 adjacente(3, 4).      adjacente(4, 3).
10 adjacente(4, 5).      adjacente(5, 4).
11
```

Figura 5 - Prolog

3.2 ESQUEMA DE CORES

Optei por representar o mapa por cores diferentes, nomeadamente o esquema de cores (A) e (B). É possível notar, na figura 6, que as regiões adjacentes possuem cores distintas.

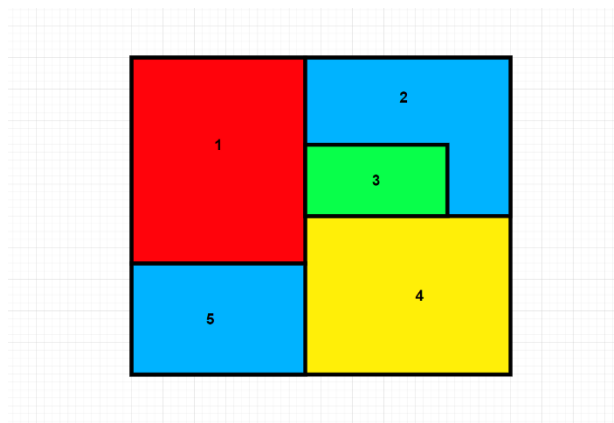


Figura 6 – Esquema de cores (A)

Por exemplo a região 1, encontra-se ligada a todas as regiões representadas no esquema, e todas as regiões do esquema adjacentes a região 1, não possuem a mesma cor, mas sim cores distintas entre si. Por exemplo, no caso da região 5 e 2, embora possuam a mesma cor, não são adjacentes um do outro. Portante nota-se particularmente que este esquema de cores (A), é uma válida representação de regiões, nos quais duas adjacentes, não têm a mesma cor.

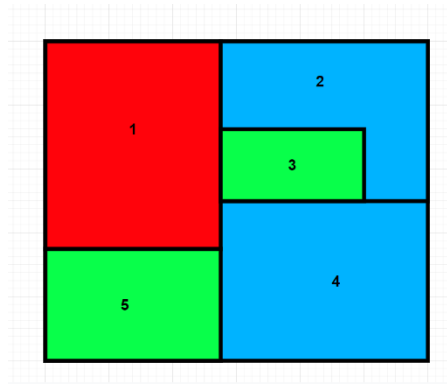


Figura 7 – Esquema de cores (B)

No entanto considerando o esquema de cores (B), nota-se que duas regiões 2 e 4, possuem a mesma cor, têm isso em comum, e são adjacentes.

```

11
12 cor(1, vermelho, a). cor(1, vermelho, b).
13 cor(2, azul, a). cor(2, azul, b).
14 cor(3, verde, a). cor(3, verde, b).
15 cor(4, amarelo, a). cor(4, azul, b).
16 cor(5, azul, a). cor(5, verde, b).
17

```

Figura 8 – Representação dos esquemas (A) e (B) em Prolog

A nomenclatura que utilizei, por exemplo cor (1, vermelho, a), representa a região 1, que é uma cor vermelha, e o esquema de cores é o (A). O mesmo é aplicável ao esquema (B). Estes representam os fatos em linguagem de prolog.

3.3 CONFLITOS

Como mencionei no ponto anterior, para a representação do grafo (B), existe adjacência entre as regiões 2 e 4, 4 e 2. No entanto as mesmas padecem de ter a mesma cor, ou seja, existe um conflito entre essas duas regiões.

```

18 conflito(Pintar) :-
19     adjacente(X, Y),
20     cor(X, Cor, Pintar),
21     cor(Y, Cor, Pintar).
22

```

Figura 9 - Conflito

Na primeira clausula significa que se o X ou Y, representados por (A) e (B) forem adjacentes, é suposto verificar, se existem regiões que são da mesma cor ou não, mas como um todo. Desta forma consigo obter o output desejado. Para tal criei duas regras, que estejam de acordo com os fatos descritos que mencionei anteriormente.

Analiso o esquema (A) ou o (B), e verifico se existe efetivamente algum conflito entre adjacências ou não. Basicamente significa que, com os parâmetros descritos nos fatos, se X e Y forem adjacentes e possuem a mesma cor, então o conflito ocorre. Mas se X e Y forem adjacentes e não tiverem a mesma cor, para todas as regiões, nesse caso, recebe-se um output False, que significa que não existe conflito.

```

22
23     conflito(R1, R2, Pintar) :-
24         adjacente(R1, R2),
25         cor(R1, Cor, Pintar),
26         cor(R2, Cor, Pintar).

```

Figura 10 – Conflito

A segunda clausula que descrevo é simplesmente outra forma de escrever esta condição, do conflito, R1 – região 1, R2 – região 2, no entanto mais pormenorizadas com três argumentos. Descrevo a que regiões, e de que grafo pertencem essas mesmas regiões, e são posteriormente analisadas. Verifica-se justamente o mesmo, se as regiões são adjacentes ou não, e se têm a mesma cor.

4. RESULTADOS OBTIDOS

Nesta fase foi onde efetuei os testes para verificar se tudo funciona como esperado. Como mencionei anteriormente, utilizei a consola do GNU Prolog, a fim de efetuar os testes necessários, para resolução do problema apresentado.

Optei por fazer um teste inicial, para verificar se efetivamente existe conflito no esquema (A), sem qualquer particularidade de regiões, mas como um todo, o grafo todo.

```

(47 ms) yes
| ?- conflito(a).

no
| ?- |

```

Figura 11 – Conflito (A)

Como se pode verificar o programa diz-me, (No), ou seja, não existe qualquer conflito no esquema (A). O esquema de cores confirma precisamente este resultado, uma vez que todas as regiões se mantêm perfeitamente descritas na sua natureza de adjacência, com cores não iguais. No entanto como se verifica na figura abaixo indicada, existe de fato conflito no esquema (B).

```

| ?- conflito(b).

true ?

yes
| ~

```

Figura 12 – Conflito (B)

Isto porque, quando fazemos uma análise no esquema (B), é possível denotar que as regiões 2 e 4, embora sejam adjacentes, possuem a mesma cor, logo gera conflito. E isso é o que o programa diz, quando o resultado é retornado como (True).

Para a segunda clausula, a pergunta feita ao programa é mais pormenorizada, ou seja, pergunto se a região N e X do esquema (A) ou (B), são efetivamente adjacentes.

```
yes
| ?- conflito(3,2,a).
no
```

Figura 13 – Conflito (A)

Verifica-se que o esquema (A), nas adjacências das regiões 3 e 2, não existe qualquer conflito. No entanto o mesmo já não se verifica com o esquema (B).

```
no
| ?- conflito(2,4,b).
yes
```

Figura 14 – Conflito (B)

O programa diz-me que existe efetivamente um conflito, neste esquema (B), em particular para as duas regiões que menciono, 2 e 4. O mesmo se verifica que corresponde a realidade apresentada no grafo correspondente do mapa (B).

5. OBSERVAÇÕES FINAIS

Para este projeto optei por implementar os conhecimentos adquiridos posteriormente ao 1º ano, ou seja, antes de aprender a teoria dos grafos, adquiri alguns conhecimentos na cadeira de Inteligência Artificial, ao qual me foi possível aliar o conteúdo que aprendi para aplicar em matemática discreta.

Sei inclusive, que um dos focos da teoria dos grafos é permitir a modelação de uma variedade de sistemas, em que a estrutura de dados começa por ser primeiro um modelo matemático ou logico que posteriormente adquire a forma de um grafo.

Neste projeto consegui, com sucesso, aplicar a logica dos grafos às tecnologias e de certa forma criar um algoritmo que permite identificar conflitos entre cores com adjacências em comum.