Segundo Exercício-Programa

Prof. Luciano Antonio Digiampietri

Prazo máximo para a entrega: 17/12/2023

1 Coloração de Mapas

Neste trabalho, você deverá resolver o problema de coloração de mapas usando duas abordagens diferentes: tentativa e erro (*backtracking*) e gulosa.

A Coloração de Mapas tem por objetivo colorir os diferentes países de um dado mapa de forma que dois países adjacentes não tenham a mesma cor.

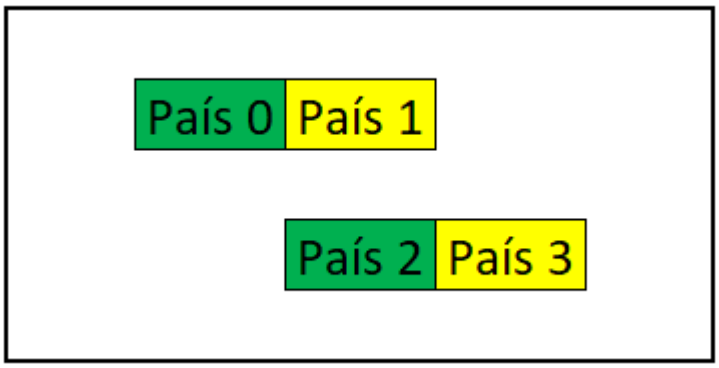
Existem diferentes formas de se fazer isso. Neste EP você deverá implementar uma solução Gulosa (que costuma ser bastante rápida, mas não garante o uso da menor quantidade possível de cores) e uma solução baseada em Tentativa e Erro que, dada uma quantidade de cores, irá verificar se é possível colorir o mapa.

Detalhamento: Para este EP, há duas estruturas de dados que serão utilizadas, chamadas *PAIS* e *MAPA*.

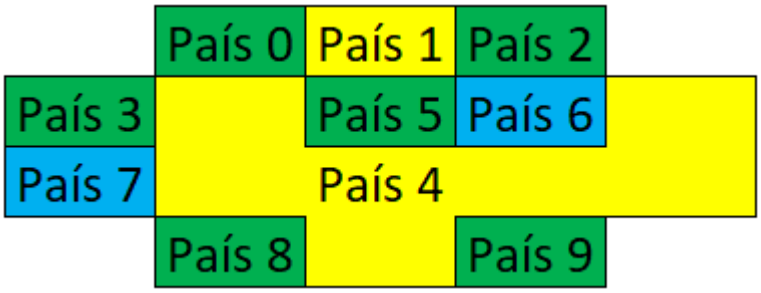
***PAIS*:** tem por objetivo representar um pais. É uma estrutura que possui quatro campos: *id* do tipo inteiro que corresponde a um identificador único de cada país (se houver dez países no mapa, os identificadores serão de zero a nove); *cor* do tipo inteiro que corresponde à cor que suas funções irão atribuir a cada países (as cores válidas correspondem a números inteiros não negativos [de zero em diante]); *numVizinhos* do tipo inteiro que contém a quantidade de países que são vizinhos do país atual; e *vizinhos* do tipo ponteiro para ponteiro de PAIS que corresponde a um ponteiro para um arranjo de ponteiros de países; neste arranjo serão colocados os endereços de cada um dos países que são vizinhos do país atual.

***MAPA*:** tem por objetivo representar um mapa, isto é, um conjunto de países. Esta estrutura possui dois campos: *numPaises* do tipo inteiro que contém a quantidade de países do mapa; e *paises* do tipo ponteiro para PAIS, correspondendo ao endereço do arranjo de países (de dados do tipo PAIS) do mapa.

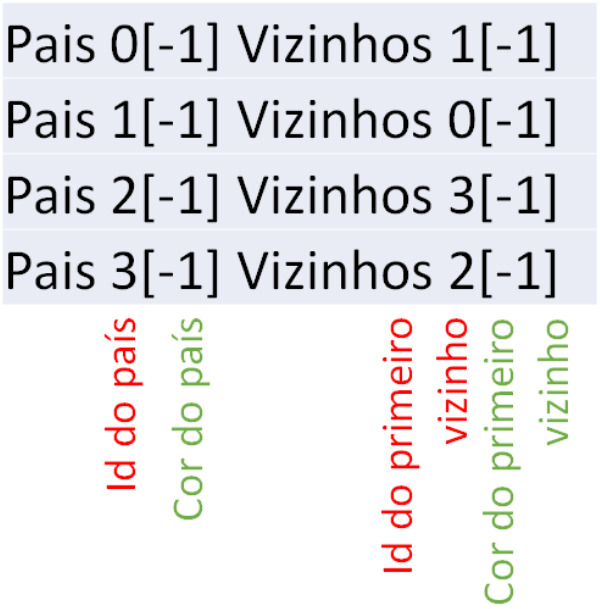
Exemplo gráfico de mapa com quatro países, no qual os países 0 e 1 são vizinhos, bem como os países 2 e 3. Países vizinhos não podem ser coloridos com a mesma cor. Mapa colorido com duas cores: verde e amarelo) .



Exemplo de mapa com dez países (colorido com três cores: verde, amarelo e azul).



Do ponto de visto deste EP, o mapa não precisa ser planar (e os países não possuem fronteiras necessariamente viáveis). O que temos é um conjunto de países e cada país contém a lista de seus vizinhos.



O código fornecido para este EP já possui várias funções implementadas, para gerar diferentes mapas e imprimir os mapas (incluindo a coloração).

Há duas funções que você deverá implementar/completar, podendo, se julgar conveniente, implementar funções adicionais/auxiliares:

- *int resolveGuloso(MAPA map)*: função que recebe um mapa como parâmetro, colore este mapa e retorna o número de cores utilizadas para colorir, de acordo com o critério guloso apresentado a seguir. Para cada país do arranjo de países (começando do primeiro, isto é, daquele da posição 0 (zero) do arranjo) colocar a primeira cor válida/viável (começando da cor zero em diante [um, dois ...]). Uma cor é considerada válida para o presente país, caso nenhum de seus vizinhos já esteja colorido com essa cor.

- *bool resolveTentativaEErro(MAPA map, int maxCor)*: função que recebe um mapa como parâmetro e um número de cores (*maxCor*) e tenta colorir o mapa usando o número de cores passado como parâmetro. Se for possível colorir o mapa, a função deverá atribuir cores aos países e retornar *true*, caso contrário, deverá retornar *false*. Esta função deve seguir estritamente a estratégia de Tentativa e Erro apresentada a seguir.

Estratégia - Tentativa e Erro: a partir do país atual (começando do primeiro) há *maxCor* ações potencialmente possíveis (que devem ser seguidas nesta ordem): colocar a cor 0 (zero) no país atual, colocar a cor 1 no país atual, colocar a cor 2 no país atual, ... colocar a cor *maxCor*-1 no país atual. Uma ação é considera viável caso nenhum dos países vizinhos do país atual já esteja colorido com a cor da ação atual. Caso seja possível tomar a ação, o país atual deverá receber a cor atual e o algoritmo deverá tentar colorir o próximo país. Se todos os países estiverem coloridos, a função deverá retornar *true*. Se não for possível colorir o país com a cor atual, o algoritmo deverá tentar colorir com a próxima cor. Se não for possível colorir o país atual com nenhuma cor, o algoritmo deverá marcar o país atual com a "cor" -1 e retornar (*backtracking*) para tentar outra coloração (tentar, por exemplo, colorir o país anterior com outra cor.

Observação: apesar de, potencialmente, existirem diversas soluções de coloração, só existe uma solução gulosa e uma de tentativa e erro considerando as regras descritas neste EP.

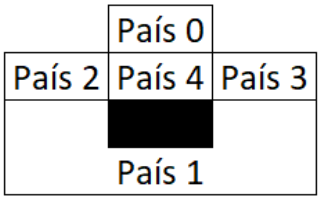
Exemplos: a seguir são apresentadas as soluções para diferentes mapas.

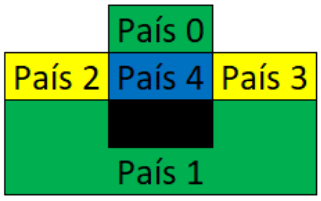
```
Solucao Gulosa (duas cores)
Pais 0[ 0] Vizinhos 1[ 1]
Pais 1[ 1] Vizinhos 0[ 0]
Pais 2[ 0] Vizinhos 3[ 1]
Pais 3[ 1] Vizinhos 2[ 0]
```

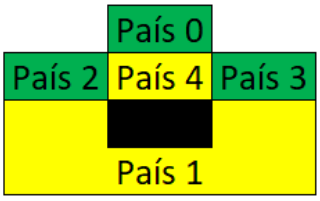
```
Solucao Gulosa (tres cores)
Pais 0[ 0] Vizinhos 4[ 2]
Pais 1[ 0] Vizinhos 2[ 1] 3[ 1]
Pais 2[ 1] Vizinhos 1[ 0] 4[ 2]
Pais 3[ 1] Vizinhos 1[ 0] 4[ 2]
Pais 4[ 2] Vizinhos 0[ 0] 2[ 1] 3[ 1]
```

```
Solucao BackTracking (duas cores)
Pais 0[ 0] Vizinhos 1[ 1]
Pais 1[ 1] Vizinhos 0[ 0]
Pais 2[ 0] Vizinhos 3[ 1]
Pais 3[ 1] Vizinhos 2[ 0]
```

```
Solucao BackTracking (duas cores)
Pais 0[ 0] Vizinhos 4[ 1]
Pais 1[ 1] Vizinhos 2[ 0] 3[ 0]
Pais 2[ 0] Vizinhos 1[ 1] 4[ 1]
Pais 3[ 0] Vizinhos 1[ 1] 4[ 1]
Pais 4[ 1] Vizinhos 0[ 0] 2[ 0] 3[ 0]
```







1.1 Material a Ser Entregue

Um arquivo, denominado *NUSP*.c (sendo NUSP o seu número USP, por exemplo: 123456789.c), contendo seu código, incluindo todas as funções solicitadas e qualquer outra função adicional que ache necessário. Para sua conveniência, `completeERenomeie.c` será fornecido, cabendo a você então completá-lo e renomeá-lo para a submissão.

Atenção!

1. Não modifique as assinaturas das funções já implementadas e/ou que você deverá completar!
2. Para avaliação, as diferentes funções serão invocadas diretamente (individualmente ou em conjunto com outras funções). Em especial, qualquer código dentro da função `main()` será ignorado.

2 Entrega

A entrega será feita única e exclusivamente via sistema e-Disciplinas, até a data final marcada. Deverá ser postado no sistema um arquivo .c, tendo como nome seu número USP:

```
seuNumeroUSP.c   (por exemplo, 12345678.c)
```

Não esqueça de preencher o cabeçalho constante do arquivo .c, com seu nome, número USP, turma etc.

A responsabilidade de postagem é exclusivamente sua. Por isso, submeta e certifique-se de que o arquivo submetido é o correto (fazendo seu download, por exemplo). Problemas referentes ao uso do sistema devem ser resolvidos com antecedência.

3 Avaliação

A nota atribuída ao EP será baseada nas funcionalidades solicitadas, porém não esqueça de se atentar aos seguintes aspectos:

1. Documentação: se há comentários explicando o que se faz nos passos mais importantes e para que serve o programa (tanto a função quanto o programa em que está inserida);
2. Apresentação visual: se o código está legível, identado etc;
3. Corretude: se o programa funciona.

Além disso, algumas observações pertinentes ao trabalho, que influenciam em sua nota, são:

- Este exercício-programa deve ser elaborado individualmente;
- Não será tolerado plágio;
- Exercícios com erro de sintaxe (ou seja, erros de compilação), receberão nota ZERO.