Estruturas de Dados

Cristina Gomes Fernandes

Pilha

Lista linear em que todas as inserções e remoções são feitas numa mesma extremidade (topo).

Implementação sequencial:
um vetor $P[1\,..\,\text{MAX}]$ e uma variável inteira $topo$.

Operações:

- Inicialize$(P, topo)$
- Empilhe$(P, topo, x)$
- Desempilhe$(P, topo)$
- Topo$(P, topo)$
- Vazia$(P, topo)$

Implementação das operações

Inicialize $(P, topo)$

1 $topo \leftarrow 0$

Implementação das operações

Inicialize $(P, topo)$

1 $topo \leftarrow 0$

Empilhe $(P, topo, x)$

1 $topo \leftarrow topo + 1$

2 $P[topo] \leftarrow x$

Desempilhe $(P, topo)$

1 $topo \leftarrow topo - 1$

2 **devolva** $P[topo + 1]$

Implementação das operações

Inicialize $(P, topo)$

1 $topo \leftarrow 0$

Empilhe $(P, topo, x)$

1 $topo \leftarrow topo + 1$

2 $P[topo] \leftarrow x$

Desempilhe $(P, topo)$

1 $topo \leftarrow topo - 1$

2 **devolva** $P[topo + 1]$

Topo $(P, topo)$

1 **devolva** $P[topo]$

Implementação das operações

Inicialize $(P, topo)$

1 $topo \leftarrow 0$

Empilhe $(P, topo, x)$

1 $topo \leftarrow topo + 1$
2 $P[topo] \leftarrow x$

Desempilhe $(P, topo)$

1 $topo \leftarrow topo - 1$
2 **devolva** $P[topo + 1]$

Topo $(P, topo)$

1 **devolva** $P[topo]$

Vazia $(P, topo)$

1 **se** $topo = 0$
2 **então devolva** VERDADE
3 **senão devolva** FALSO

Aplicações

- cálculo de expressões em notação posfixa
- conversão de expressões em notação infixa para posfixa
- administração de memória em tempo de execução
- algoritmo de Graham para cálculo de casco convexo
- backtrack

Implementação com lista ligada

Inicialize (p)

1 $p \leftarrow \text{NIL}$

Empilhe (p, x)

1 $q \leftarrow$ NovaCélula(x)
2 $prox(q) \leftarrow p$
3 $p \leftarrow q$

Desempilhe (p)

1 $x \leftarrow info(p)$
2 $q \leftarrow p$
3 $p \leftarrow prox(p)$
4 LiberaCélula(q)
5 **devolva** x

Implementação com lista ligada

Inicialize (p)

1 $p \leftarrow$ NIL

Empilhe (p, x)

1 $q \leftarrow$ NovaCélula(x)
2 $prox(q) \leftarrow p$
3 $p \leftarrow q$

Topo (p)

1 **devolva** $info(p)$

Desempilhe (p)

1 $x \leftarrow info(p)$
2 $q \leftarrow p$
3 $p \leftarrow prox(p)$
4 LiberaCélula(q)
5 **devolva** x

Implementação com lista ligada

Inicialize (p)

1 $p \leftarrow \text{NIL}$

Empilhe (p, x)

1 $q \leftarrow$ NovaCélula(x)
2 $prox(q) \leftarrow p$
3 $p \leftarrow q$

Topo (p)

1 **devolva** $info(p)$

Vazia (p)

1 **se** $p = \text{NIL}$
2 **então devolva** VERDADE
3 **senão devolva** FALSO

Desempilhe (p)

1 $x \leftarrow info(p)$
2 $q \leftarrow p$
3 $p \leftarrow prox(p)$
4 LiberaCélula(q)
5 **devolva** x

LDL circular com cabeça

Inicialize (p)

1 $p \leftarrow$ NovaCélula(x)
2 $prox(p) \leftarrow p$
3 $prev(p) \leftarrow p$

Empilhe (p, x)

1 $q \leftarrow$ NovaCélula(x)
2 $prox(q) \leftarrow prox(p)$
3 $prev(q) \leftarrow p$
4 $prev(prox(p)) \leftarrow q$
5 $prox(p) \leftarrow q$

Vazia (p)

1 **se** $prox(p) = p$
2 **então devolva** VERDADE
3 **senão devolva** FALSO

Topo (p)

1 **devolva** $info(prox(p))$

Desempilhe (p)

1 $x \leftarrow info(prox(p))$
2 $q \leftarrow prox(p)$
3 $prox(p) \leftarrow prox(q)$
4 $prev(prox(p)) \leftarrow p$
5 LiberaCélula(q)
6 **devolva** x

Backtrack

Problema: Dado n gerar todas as subseqüências de $1, 2, \ldots, n$ em ordem lexicográfica.

Exemplo: $n = 4$

Saída:

1
1 2
1 2 3
1 2 3 4
1 2 4
1 3
1 3 4
1 4
2
2 3
2 3 4
2 4
3
3 4
4

Quantas subseqüências há?

Algoritmo

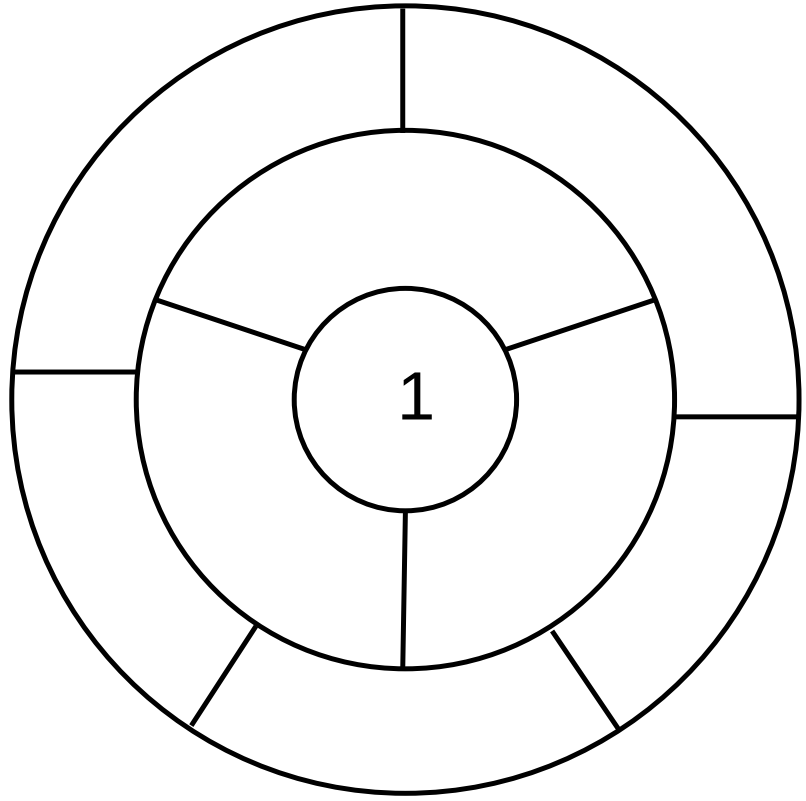
Gera Subseq Lex (n)

```
 1  Inicialize(P, topo)
 2  Empilhe(P, topo, 0)
 3  enquanto não Vazia(P, topo) faça
 4     x ← Desempilhe(P, topo)
 5     Empilhe(P, topo, x + 1)
 6     Imprime(P, topo)
 7     enquanto Topo(P, topo) < n faça
 8        Empilhe(P, topo, Topo(P, topo) + 1)
 9        Imprime(P, topo)
10     Desempilhe(P, topo)      ▷ desempilha o n
```

Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

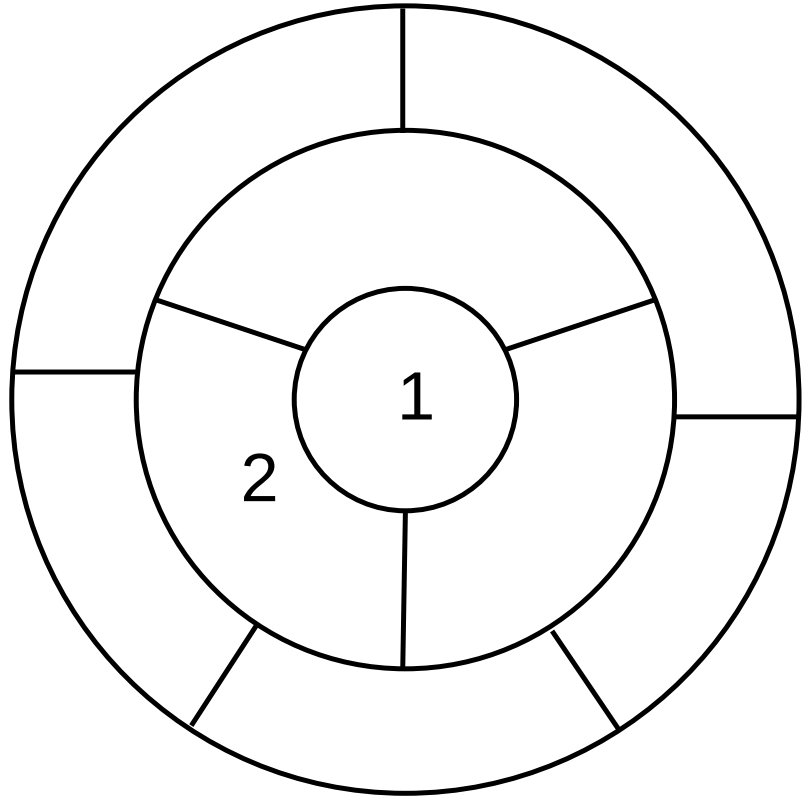
Exemplo:



Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

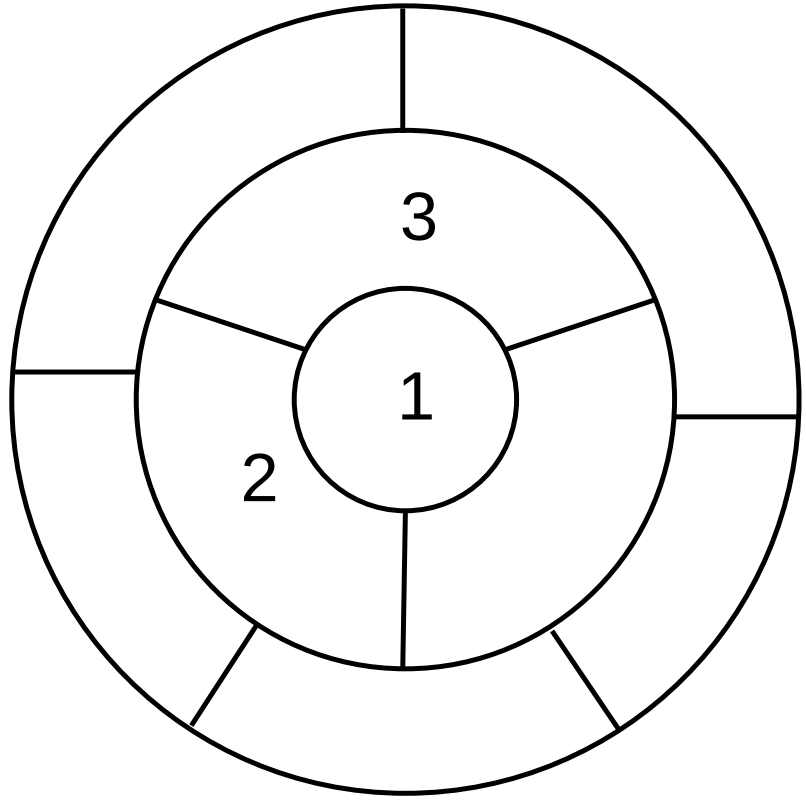
Exemplo:



Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

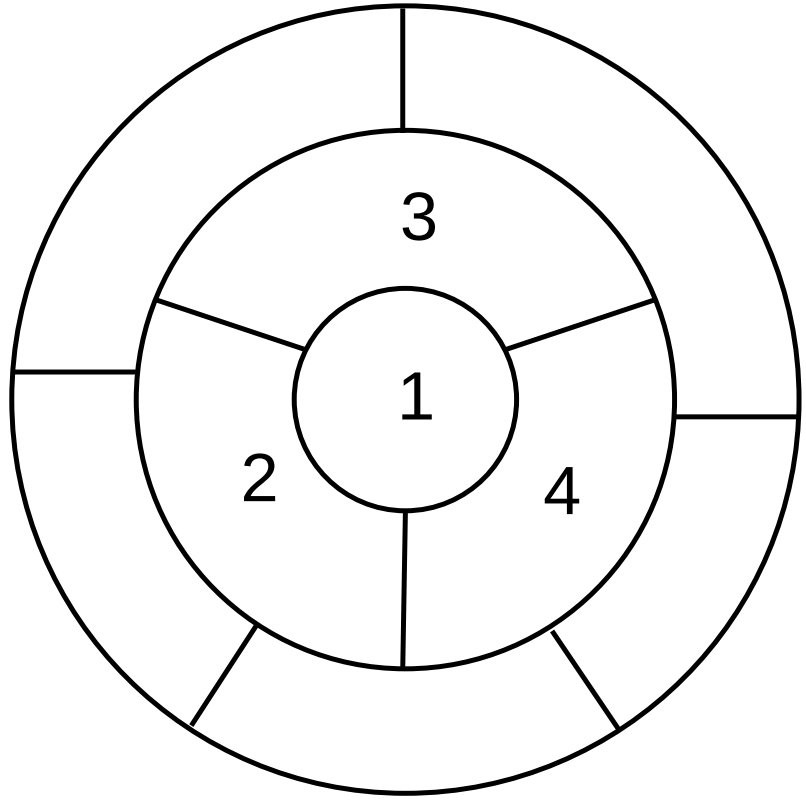
Exemplo:



Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

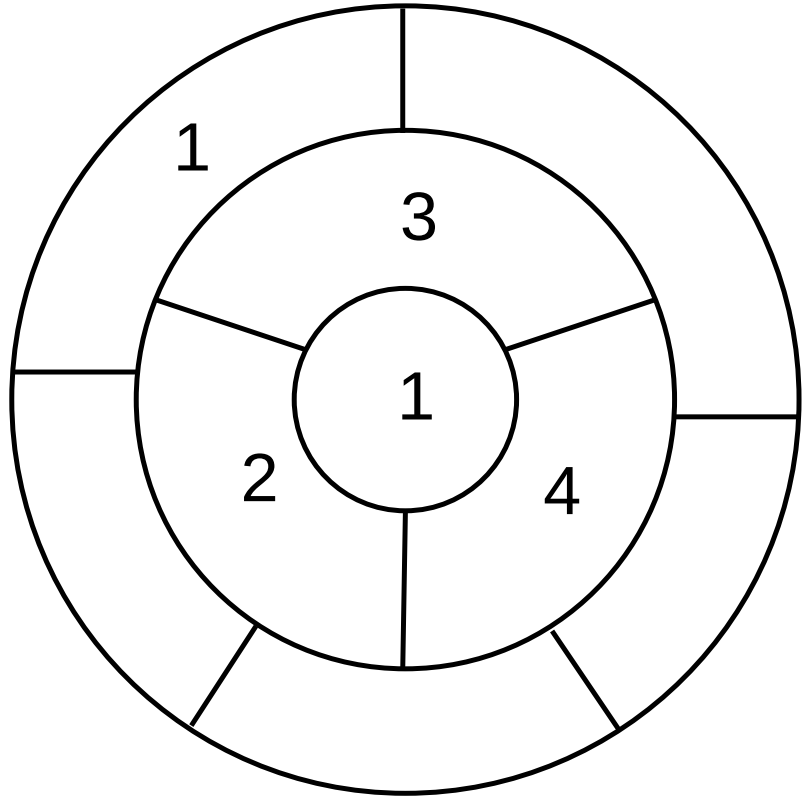
Exemplo:



Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

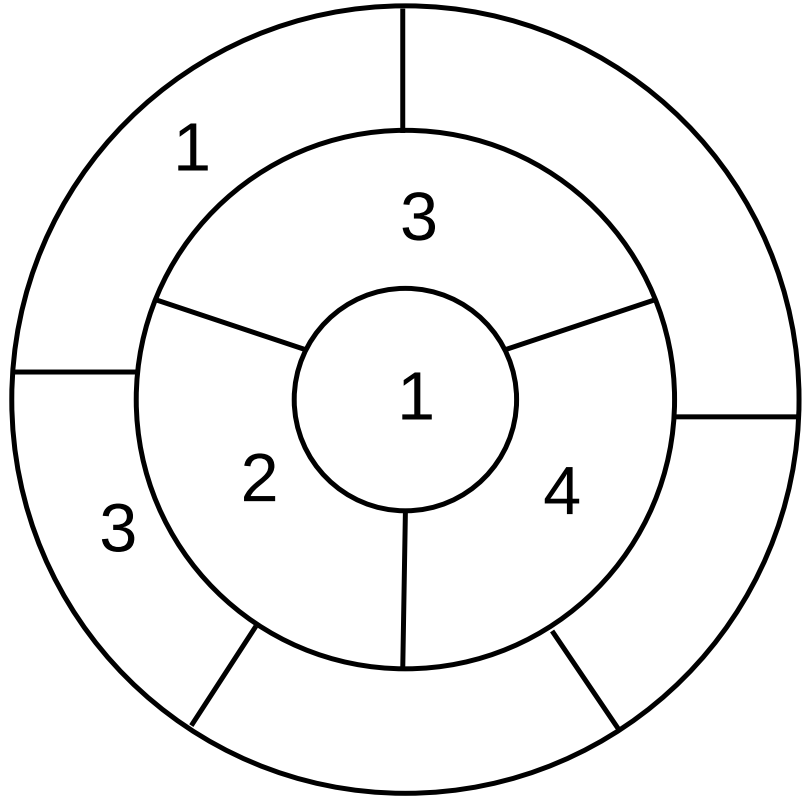
Exemplo:



Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

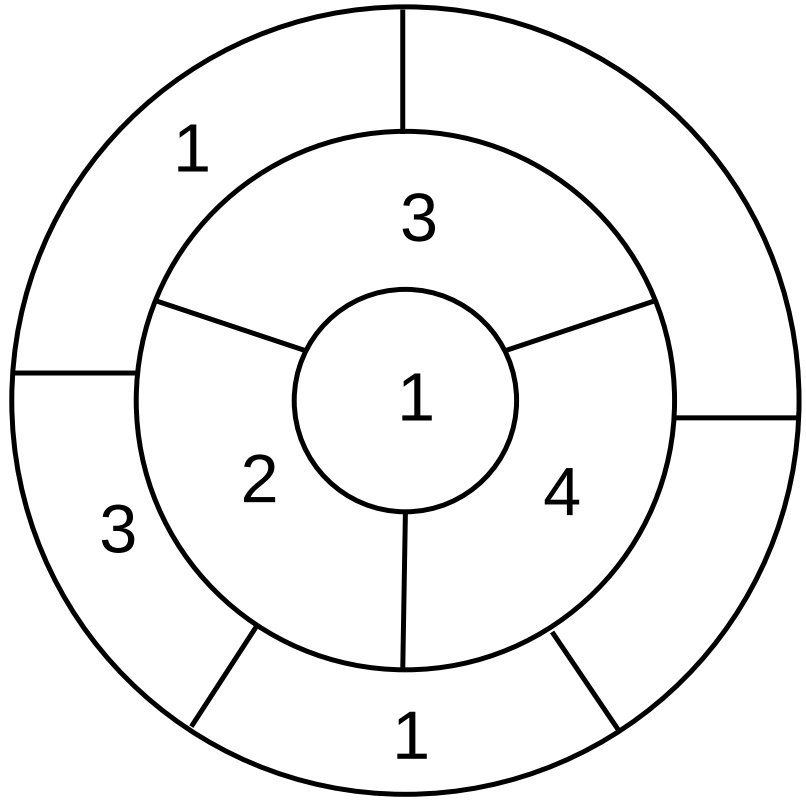
Exemplo:



Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

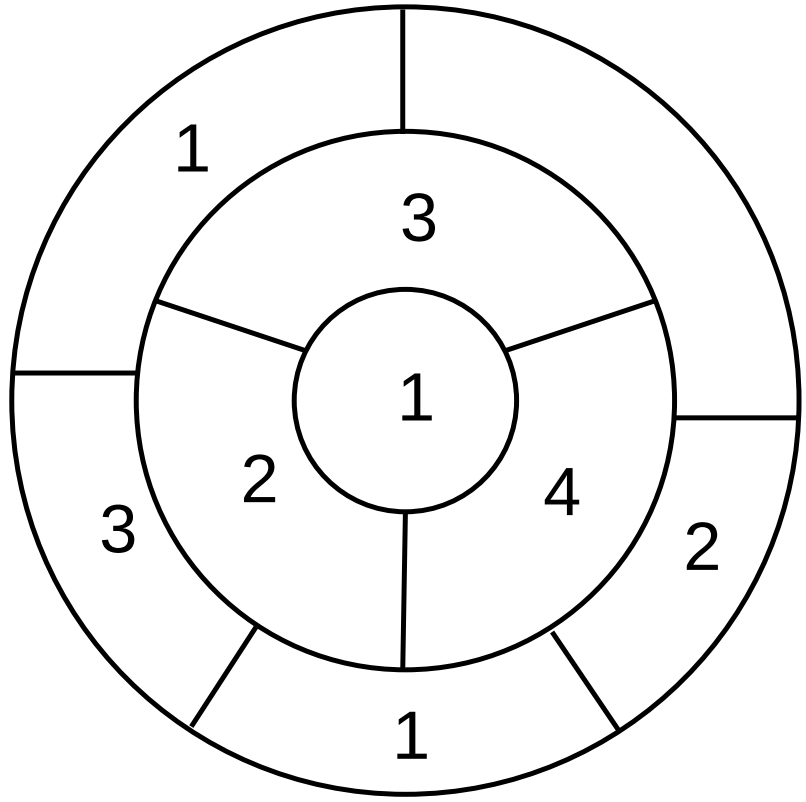
Exemplo:



Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

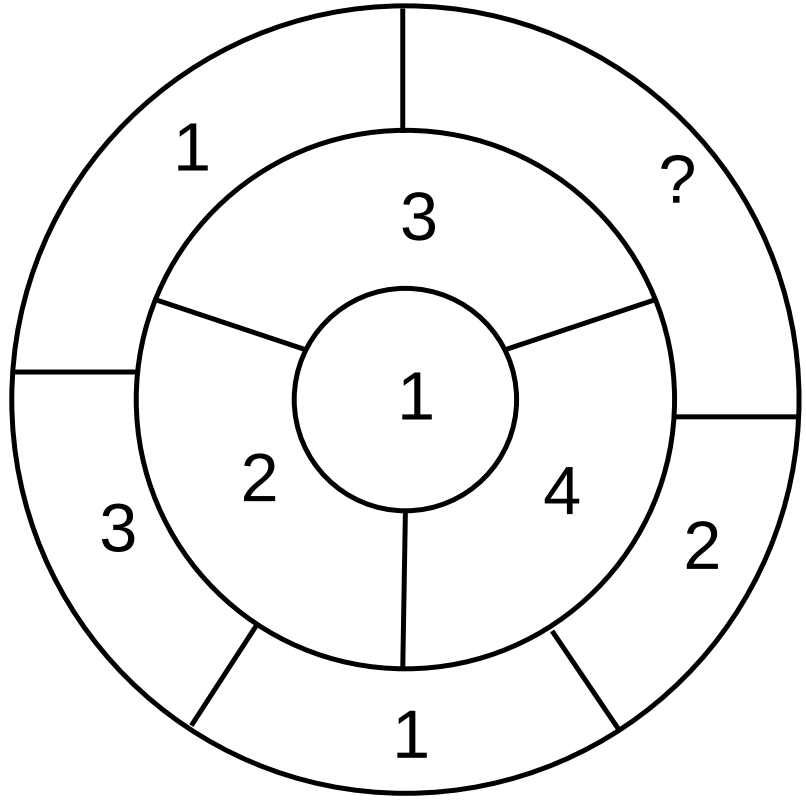
Exemplo:



Coloração de mapas

Problema: Dado um mapa com n países, cada país com a lista de seus vizinhos, imprimir uma colocarção do mapa que use no máximo 4 cores.

Exemplo:



Algoritmo

Pinta Mapa (n, viz)

```
 1  Inicialize(Pcor, t)
 2  Empilhe(Pcor, t, 1)
 3  c ← 1
 4  enquanto t < n faça
 5     se Pode Colorir(Pcor, t, c, viz)
 6        então Empilhe(Pcor, t, c)
 7              c ← 1
 8        senão c ← c + 1
 9              enquanto c > 4 faça
10                 c ← Desempilhe(Pcor, t) + 1
11  Imprime(Pcor, t)
```

Pode Colorir

Esta função recebe a lista de cores usadas para colorir os países de 1 a t, uma cor c e a lista de vizinhos de cada país e devolve SIM se o país $t+1$ pode ser colorido com a cor c, devolve NÃO caso contrário.

Pode Colorir $(Pcor, t, c, viz)$

```
1  para cada vizinho i de t+1 faça
2     se i ≤ t e Pcor[i] = c
3        então devolva NÃO
4  devolva SIM
```

Exercícios

Problema das 8 rainhas:

Dado um inteiro n, determinar se existe uma maneira de colocar n rainhas num tabuleiro de xadrez $n \times n$ sem que nenhuma delas ataque a outra.

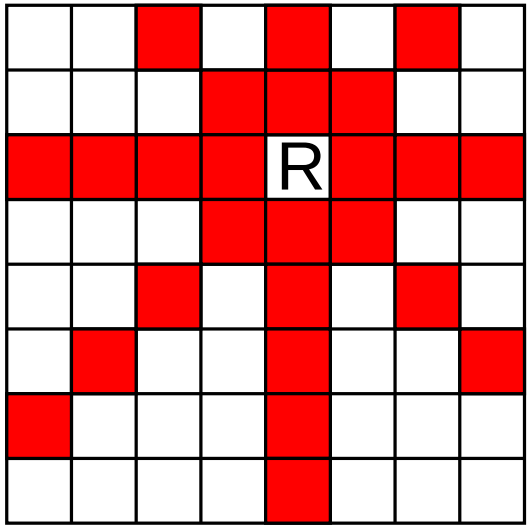
Escreva um algoritmo que resolva o problema das 8 rainhas.

Caso exista uma maneira de colocar n rainhas, seu algoritmo deve imprimir as posições das n rainhas para uma destas maneiras válidas de colocá-las.

Exercícios

Problema do passeio do cavalo:
Dado um inteiro n, determinar se existe uma maneira de um cavalo se movimentar por todas as posições do tabuleiro de xadrez, e terminar onde começou, sem passar duas vezes por nenhuma das posições.

Ataque da rainha



Movimento do cavalo