

# ESCOLA DE PRIMAVERA DA MARATONA SBC DE PROGRAMAÇÃO















APOIO:







Grupo de Computação Competitiva

## BUSCA EM LARGURA E FLOOD FILL



Por: Gabriela Emanuele de Araujo Amorim



### **CONTEÚDOS**

- 01 Problema motivador (Coloração de Cenários de Jogos 1907)
- 02 Analisando o Problema
- 03 Busca em Largura (BFS)
- 04 Funcionamento do algoritmo
- 05 Algoritmo
- 06 Flood Fill
- 07 Resolução do problema
- 08 Problemas



#### 01 - PROBLEMA MOTIVADOR

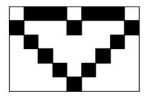
beecrowd | 1907

#### Coloração de Cenários de Jogos

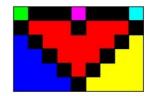
Por Leandro Zatesko, UFFS 🔯 Brazil

Timelimit: 1

O Prof. Fernando Bevilacqua está muito preocupado com os cenários do seu mais novo jogo. Os contornos dos cenários já foram desenhados por um artista, restando ao Prof. Fernando apenas colori-los. No momento, cada cenário é uma imagem em que cada *pixel* está preto ou branco. Assim, quando o Prof. Fernando, em seu programa de coloração de imagens, clica num *pixel* branco para ser colorido com uma cor **a**, toda a *região branca* em que está o *pixel* selecionado recebe a cor **a**. Dizemos que um *pixel* branco **A** está na mesma região branca que um *pixel* branco **B** se existe um caminho entre **A** e **B** que passa apenas por *pixels* brancos e que considera as adjacências apenas nos sentidos horizontal e vertical. Por exemplo, são necessários 6 cliques para colorir a figura da esquerda.









### 01 - PROBLEMA MOTIVADOR

#### Entrada

A primeira linha da entrada consiste de dois inteiros positivos  $\mathbf{N}$  e  $\mathbf{M}$  ( $\mathbf{N}$ ,  $\mathbf{M} \le 1.024$ ), os quais representam a *resolução* da imagem. Cada uma das  $\mathbf{N}$  linhas seguintes contém  $\mathbf{M}$  caracteres, os quais podem ser  $\cdot$  (ponto) ou  $\mathbf{o}$  (letra 'o' minúscula), representando respectivamente um *pixel* branco ou um *pixel* preto.

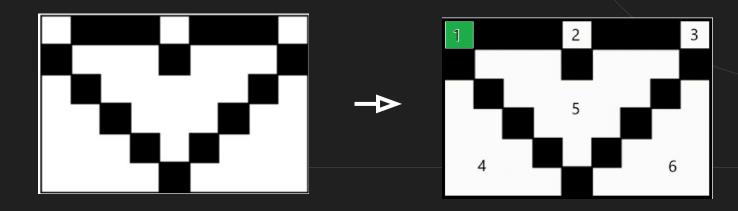
#### Saída

Imprima uma linha contendo um único inteiro que represente o número de cliques necessários para colorir toda a figura descrita na entrada.

Exemplos de Entrada		Exemplos de Saída
6 9	6	
.000.000.		
00		
.00.		
0.0		
1 8	4	
.0.0.0.0		
1 1	0	
0		



### 02 - ANALISANDO O PROBLEMA



Resposta: 6



### 03 - BUSCA EM LARGURA (BFS)

A Busca em Largura (BFS) explora um grafo expandindo todos os vértices de uma mesma distância antes de passar para a próxima. Ela encontra o menor caminho (em número de arestas) de um vértice de origem para os outros vértices acessíveis.

- Processo: Cada vértice pode estar em três estados:
  - Não visitado;
  - Processo de exploração;
  - Totalmente processado.
- Objetivo: Descobrir todos os vértices acessíveis e calcular a distância mínima até eles.



1	for cada vértice u ∈ V[G] - {s}	10 w	hile Q ≠ Ø	
2		11	$do\ u \leftarrow DEQUEUE(Q)$	
3		12	for cada v ∈ Adj[u]	(SOURCE)
4	$\pi[u] \leftarrow NULO$	13	do if cor[v] = BRANCO	(SOUNCE)
5	cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$	
	d[s] ← 0	15	$d[v] \leftarrow d[u] + 1$	B C D
	$\pi[s] \leftarrow NULO$	16	π[v] ← u	
	0 ← ∅	17	ENQUEUE(Q, v)	
9	ENQUEUE(Q, s)	18	cor[u] ← PRETO	E F
F	Fonte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s.	n.], 2012.	



1	for cada vértice $u \in V[G] - \{s\}$	10 w	hile Q ≠ Ø			
2	do cor[u] ← BRANCO	11	do u $\leftarrow$ DEQUEUE(Q)			
3	d[u] ← ∞	12	for cada v ∈ Adj[u]		A ← S (SOUF	OCE)
4	π[u] ← NULO	13	do if cor[v] = BRANCO		(300)	(CE)
	cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$			
	$d[s] \leftarrow 0$	15	$d[v] \leftarrow d[u] + 1$	В	C D	NULO
	$\pi[s] \leftarrow \text{NULO}$	16	π[v] ← u	NULO	NULO	œ
	0 ← Ø	17	ENQUEUE(Q, v)	<b>6 ° 6</b>	œ	
	ENQUEUE(Q, s)	18	cor[u] ← PRETO	E		
	onte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s. ı	n.], 2012.	NULO	NULO	
				<b>∞</b>	œ	



1	for cada vértice $u \in V[G] - \{s\}$	10 w	hile Q ≠ Ø			
2		11	do u $\leftarrow$ DEQUEUE(Q)			
3	d[u] ← ∞	12	for cada v ∈ Adj[u]		A S (SOU	DCE)
4	π[u] ← NULO	13	do if cor[v] = BRANCO		(300)	(CL)
5	cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$			
	d[s] ← 0	15	$d[v] \leftarrow d[u] + 1$	В	CD	NULO
	$\pi[s] \leftarrow \text{NULO}$	16	π[v] ← u	<b>A1</b>	NULO	<b>∞</b>
	$Q \leftarrow \emptyset$	17	ENQUEUE(Q, v)	E	<b>~</b> ∞	
	ENQUEUE(Q, s)	18	cor[u] ← PRETO	E	F /	
	onte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s. r	n.], 2012.	NULO	NULO	
				œ	œ	



1 for ca	da vértice u ∈ V[G] - {s}	10 w	hile Q ≠ Ø			
	cor[u] ← BRANCO	11	do u $\leftarrow$ DEQUEUE(Q)			
	[u] ← ∞	12	for cada v ∈ Adj[u]	(	A	URCE)
	r[u] ← NULO	13	do if cor[v] = BRANCO			URCE)
	← CINZA	14	then $cor[v] \leftarrow CINZA$			
6 d[s] ←		15	$d[v] \leftarrow d[u] + 1$	(B) (	CD	NULO
7 π[s] ←		16	π[v] ← u	A1	A1	<b>∞</b>
8 Q ← <sup>©</sup>		17	ENQUEUE(Q, v)			
	EUE(Q, s)	18	cor[u] ← PRETO	E	F )	
_	ME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s. r	1.], 2012.	NULO	NULO	
				œ	<b>∞</b>	



1 for ca	da vértice u ∈ V[G] - {s}	10 w	hile Q ≠ Ø			
	cor[u] ← BRANCO	11	$do\ u \leftarrow DEQUEUE(Q)$			
	[u] ← ∞	12	for cada v ∈ Adj[u]	(	A (SOUF	OCE)
	[u] ← NULO	13	do if $cor[v] = BRANCO$		(SOUR	(CE)
	← CINZA	14	then $cor[v] \leftarrow CINZA$			
6 d[s] ←		15	$d[v] \leftarrow d[u] + 1$	$\left( \mathbf{B} \right)$	(C)	
7 π[s] ←		16	π[v] ← u	A1	A1	<b>A1</b>
8 0 ← ∅		17	ENQUEUE(Q, v)			
·	EUE(Q, s)	18	cor[u] ← PRETO	E	F)	
	ルE, Thomas. <b>Algoritmos Teoria e Prática</b> .	[S. l.: s.	n.], 2012.	NULO	NULO	
				<b>co</b>	œ	



1 for cac	da vértice u ∈ V[G] - {s}	10 w	hile Q ≠ Ø			
	cor[u] ← BRANCO	11	$do\ u \leftarrow DEQUEUE(Q)$			
	[u] ← ∞	12	for cada v ∈ Adj[u]	(	A S (SOU	OCE)
	[u] ← NULO	13	do if cor[v] = BRANCO		(300)	(CE)
	← CINZA	14	then $cor[v] \leftarrow CINZA$			
6 d[s] ←		15	$d[v] \leftarrow d[u] + 1$	(B) (	(C)	
7 π[s] ←		16	π[v] ← u	A1	A1	A1
8 Q ← 2		17	ENQUEUE(Q, v)			
9 ENQU		18	cor[u] ← PRETO	E	F)	
	ME, Thomas. <b>Algoritmos Teoria e Prática</b> .	[S. l.: s.	n.], 2012.	NULO	NULO	
				œ	œ	



1	for cada vértice $u \in V[G] - \{s\}$	10 w	hile Q ≠ Ø			
2		11	$do\ u \leftarrow DEQUEUE(Q)$			
3		12	for cada v ∈ Adj[u]		A S (SOUF	CE)
4		13	do if cor[v] = BRANCO		(300)	(CL)
5	cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$			
	d[s] ← 0	15	$d[v] \leftarrow d[u] + 1$	В		
	π[s] ← NULO	16	π[v] ← u	<b>A1</b>	A1 A1	41
	$Q \leftarrow \varnothing$	17	ENQUEUE(Q, v)			
9	ENQUEUE(Q, s)	18	cor[u] ← PRETO	E	f /	
	Fonte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s.	n.], 2012.	B2	NULO	
					00	

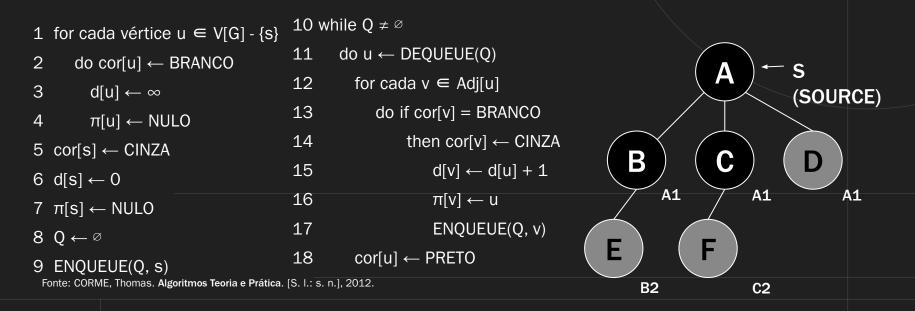


1	for cada vértice $u \in V[G] - \{s\}$	10 w	hile Q ≠ Ø			
2	do cor[u] ← BRANCO	11	do u $\leftarrow$ DEQUEUE(Q)			
3	d[u] ← ∞	12	for cada v ∈ Adj[u]		A S (SOUF	OCE)
4	π[u] ← NULO	13	do if cor[v] = BRANCO		(300)	(CE)
5	cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$			
	$d[s] \leftarrow 0$	15	$d[v] \leftarrow d[u] + 1$	$\left( B \right)$		
	π[s] ← NULO	16	π[v] ← u	A1	A1	41
	Q ← ∅	17	ENQUEUE(Q, v)			
	ENQUEUE(Q, s)	18	cor[u] ← PRETO	E		
	onte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s. r	n.], 2012.	B2	NULO	
					<b>∞</b>	



1 for cada vértice u ∈ V[G] - {s}	10 wł	nile Q ≠ Ø		
2 do cor[u] ← BRANCO	11	do u $\leftarrow$ DEQUEUE(Q)		\ <b>←</b> c
3 d[u] ← ∞	12	for cada v ∈ Adj[u]	A	(SOURCE)
4 $\pi[u] \leftarrow \text{NULO}$	13	do if cor[v] = BRANCO		(SOURCE)
5 cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$		
6 d[s] ← 0	15	$d[v] \leftarrow d[u] + 1$	(B)	
7 $\pi[s] \leftarrow \text{NULO}$	16	π[v] ← u	A1 /	A1 A1
8 Q ← Ø	17	ENQUEUE(Q, v)		
9 ENQUEUE(Q, s)	18	cor[u] ← PRETO	(E) (F)	
Fonte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s. n	.], 2012.	B2	C2







1	for cada vértice u ∈ V[G] - {s}	10 w	hile Q ≠ Ø	
2		11	$do\ u \leftarrow DEQUEUE(Q)$	
3	$d[u] \leftarrow \infty$	12	for cada v ∈ Adj[u]	A S (SOURCE)
4	π[u] ← NULO	13	do if cor[v] = BRANCO	(SOUNCE)
5	cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$	
	d[s] ← 0	15	$d[v] \leftarrow d[u] + 1$	$\left(\begin{array}{c} \mathbf{B} \end{array}\right) \left(\begin{array}{c} \mathbf{C} \end{array}\right) \left(\begin{array}{c} \mathbf{D} \end{array}\right)$
	$\pi[s] \leftarrow \text{NULO}$	16	π[v] ← u	A1 / A1 A1
	Q ← ∅	17	ENQUEUE(Q, v)	
	ENQUEUE(Q, s)	18	cor[u] ← PRETO	E
	onte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s. r	1.], 2012.	B2 C2



1	for cada vértice u ∈ V[G] - {s}	10 wh	nile Q ≠ Ø			
2		11	do u $\leftarrow$ DEQUEUE(Q)		<b>^</b> ← c	
3	d[u] ← ∞	12	for cada v ∈ Adj[u]		(SOURCE	ΕJ
4	π[u] ← NULO	13	do if $cor[v] = BRANCO$		(SOUNCE	-)
5	cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$			
	d[s] ← 0	15	$d[v] \leftarrow d[u] + 1$	$\left( \mathbf{B}\right) $	$(\mathbf{C})$	
	$\pi[s] \leftarrow \text{NULO}$	16	π[v] ← u	<b>A1</b>	A1 A1	
	Q ← Ø	17	ENQUEUE(Q, v)			
	ENQUEUE(Q, s)	18	$cor[u] \leftarrow PRETO$			
	Fonte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s. n.]	], 2012.	B2	C2	



1	for cada vértice $u \in V[G] - \{s\}$	10 wh	nile Q ≠ Ø			
2		11	do u $\leftarrow$ DEQUEUE(Q)		<b>→</b> c	
3	d[u] ← ∞	12	for cada v ∈ Adj[u]	A	SOUF	CE)
4	π[u] ← NULO	13	do if cor[v] = BRANCO		(3001)	ONOL)
5	cor[s] ← CINZA	14	then $cor[v] \leftarrow CINZA$			
	d[s] ← 0	15	$d[v] \leftarrow d[u] + 1$	(B)		
	$\pi[s] \leftarrow \text{NULO}$	16	π[v] ← u	A1 /	A1 /	<b>\1</b>
	$Q \leftarrow \emptyset$	17	ENQUEUE(Q, v)			
	ENQUEUE(Q, s)	18	cor[u] ← PRETO	(E)		
	onte: CORME, Thomas. Algoritmos Teoria e Prática.	[S. l.: s. n.]	], 2012.	B2	C2	



## 05 - ALGORITMO

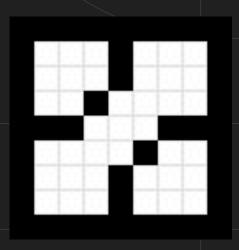


#### **06 - FLOOD FILL**

- Algoritmo usado para determinar uma área conectada em uma matriz bidimensional;
- Simula o comportamento da ferramenta "balde de tinta" em editores de imagem.

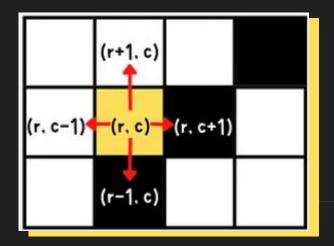
#### **Funcionamento:**

- 1. Ponto de Partida: Começa em uma posição inicial (x, y).
- 2. Direções: Decide se vai em 4 ou 8 direções.
- 3. Verificação:
  - a. Se a célula tem a cor de destino.
  - b. Substitui pela cor substituta.
- 4. Recursão/Iteração: Repete o processo até preencher a área conectada.

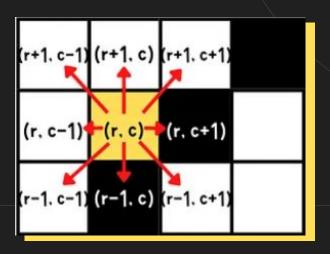




### 06 - FLOOD FILL







Eight-way

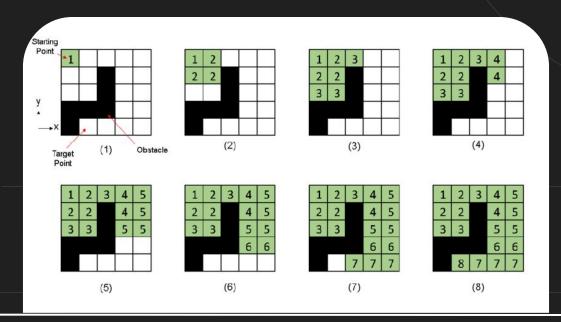


### 06 - FLOOD FILL (ALGORITMO)

```
1 flood_fill(x, y, check_validity)
                                               if (check_validity(x1 + 1, y1))
   Queue q
                                      8
                                                q.push(x1 + 1, y1)
   q.push((x, y))
                                               if (check_validity(x1 - 1, y1))
   while (q is not empty)
                                      10
                                                  q.push(x1 - 1, y1)
      (x1, y1) = q.pop()
                                              if (check_validity(x1, y1 + 1))
5
                                      11
      color(x1, y1)
                                      12
                                                  q.push(x1, y1 + 1)
                                      13
                                              if (check_validity(x1, y1 - 1))
                                                  q.push(x1, y1 - 1)
                                      14
```



### 06 - FLOOD FILL





## 07 - RESOLUÇÃO DO PROBLEMA MOTIVADOR

#### Estratégia:

- Utilizar o algoritmo Flood Fill para contar regiões conectadas de pixels brancos;
- Pixels brancos conectados ortogonalmente (esquerda, direita, cima, baixo) formam uma única região.

#### Solução:

- Percorrer a matriz;
- Ao encontrar um pixel branco (.), iniciar o Flood Fill (BFS) para marcar toda a região conectada;
- Cada novo Flood Fill indica um novo clique;
- Contar o número de cliques realizados.



## 07 - RESOLUÇÃO DO PROBLEMA MOTIVADOR



### 08 - PROBLEMAS

Duende Perdido - https://judge.beecrowd.com/pt/problems/view/2294

Colorindo - https://judge.beecrowd.com/pt/problems/view/2405

Manchas de Pele - https://judge.beecrowd.com/pt/problems/view/3061



### REFERÊNCIAS

[1] CORME, Thomas. Algoritmos Teoria e Prática. [S. I.: s. n.], 2012.

[2] FLOOD/SEED Fill Algorithm. [S. I.], 2022. Disponível em: https://yuminlee2.medium.com/flood-seed-fill-algorithm-21fba08a46e. Acesso em: 10 out. 2024.



# OBRIGADO PELA ATENÇÃO

Grupo de Computação Competitiva

