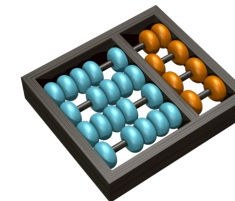


Instituto de Computação - UNICAMP

MC102 - Algoritmos e Programação de Computadores



Laboratório 06: It's me, Mario!

Segundo Semestre de 2017 - Turmas Coordenadas

Peso da Atividade: 2

Prazo de Entrega: 13 de Outubro de 2017 às 23:59:59

Conteúdo

[Contexto](#)

[Tarefa](#)

[Observações da Tarefa](#)

[Exemplos](#)

[Observações Gerais](#)

[Critérios Importantes](#)

Contexto

It's me, SuSy!

Seria um nome melhor para a atividade?



Figura 1: Tela final do jogo Super Mario World.

Mario é uma das franquias de jogos eletrônicos mais famosas da *Nintendo*, criada por Shigeru Miyamoto. Protagonista do jogo, Mario é um encanador que precisa geralmente passar por várias etapas para salvar alguma personagem. Sua primeira aparição foi no jogo *Donkey Kong* em 1981 e, desde então, várias outras séries foram dedicadas ao personagem.

Uma das séries mais famosas da franquia é a *Super Mario Bros*, na qual Mario precisa resgatar a Princesa Peach do seu archi-inimigo Bowser. Em alguns jogos desta série, Mario tem ajuda de seu irmão Luigi e do dinossauro Yoshi.

Por conta do sucesso dos vários jogos da série, Shigeru Miyamoto está planejando lançar mais uma versão de *Super Mario*. Nesta versão, o criador de *Mario* deseja incluir novas funcionalidades e também novos cenários para o jogo. Uma dessas funcionalidades é um estimador de tempo para finalização do jogo. Para isso são necessárias duas informações: o total de ilhas, ou regiões, em um mapa e o tempo médio de jogo em cada uma das ilhas.

Para a primeira etapa de desenvolvimento, Shigeru Miyamoto precisa apenas do número total de ilhas em um mapa. Você é um dos desenvolvedores da série *Mario* e precisa implementar a primeira etapa desta funcionalidade.

O uso de cores vivas é uma característica marcante do jogo e Shigeru já criou diversos mapas para esta nova versão. A princípio você se desespera pois não sabe muito bem como vai lidar com toda esta gama de cores empregadas no jogo. Luigi, um outro programador da Nintendo, é muito bom em processamento de imagens digitais e te tranquiliza. Ele, então, pré-processa todas as imagens dos mapas existentes, convertendo-as para imagens em preto e branco. Cada ilha é formada por regiões na cor preta e a água, em volta das ilhas,

pelas áreas de cor branca. Ele sugere que seja utilizada uma imagem binária no formato **PBM**. Os pixels de valor 1 são pretos e os de valor 0 são brancos.



Figura 2: Exemplo de um mapa do jogo e sua versão em preto e branco que será considerada para a implementação da nova funcionalidade do jogo. Neste mapa existem 10 ilhas ou regiões para o jogo.

O formato de imagem **PBM** é assim organizado:

P1 #Código que informa que esta é uma imagem composta por zeros e uns (binária)

W H #Dois inteiros que representam, respectivamente, a largura e altura da imagem, em pixels

#As linhas daqui pra baixo contém números 0 ou 1, no formato de uma matriz

#Os elementos são denominados pixels, e se o valor do pixel é zero, a cor é branca, caso contrário, a cor é preta

$p_{1,1}$	$p_{1,2}$...	$p_{1,w}$
$p_{2,1}$	$p_{2,2}$...	$p_{2,w}$
...
$p_{H,1}$	$p_{H,2}$...	$p_{H,w}$

Exemplo de uma imagem PBM:

```
P1
9 3
```

0	0	0	0	0	0	0	0	0
0	1	1	1	0	1	1	1	0
0	1	1	1	0	1	1	1	0



Figura 3: Representação ampliada da imagem do exemplo acima.

Agora que você está bem mais aliviado sobre o pré-processamento da imagem, e já conhece o formato da imagem que irá trabalhar, pode pegar a sua caixa de lápis de cor com 36 cores e várias folhas em branco para pensar no algoritmo que vai criar para determinar quantas ilhas há em cada mapa.

Imagine que a imagem do mapa do jogo é a exibida na Figura 4 (a). Se você tiver um lápis de cor **vermelho** e outro lápis de cor **azul**, uma forma de colorir a imagem é como a da Figura 4 (b). Em um determinado ponto, pode ser que, dois pixels vizinhos tenham cores diferentes, como na Figura 4 (c), e uma estratégia¹ possível é escolher uma das cores (por exemplo, a **vermelha**) para substituir tudo que havia sido pintado com a outra cor (por exemplo, a **azul**) pela primeira cor, como na Figura 4 (d).

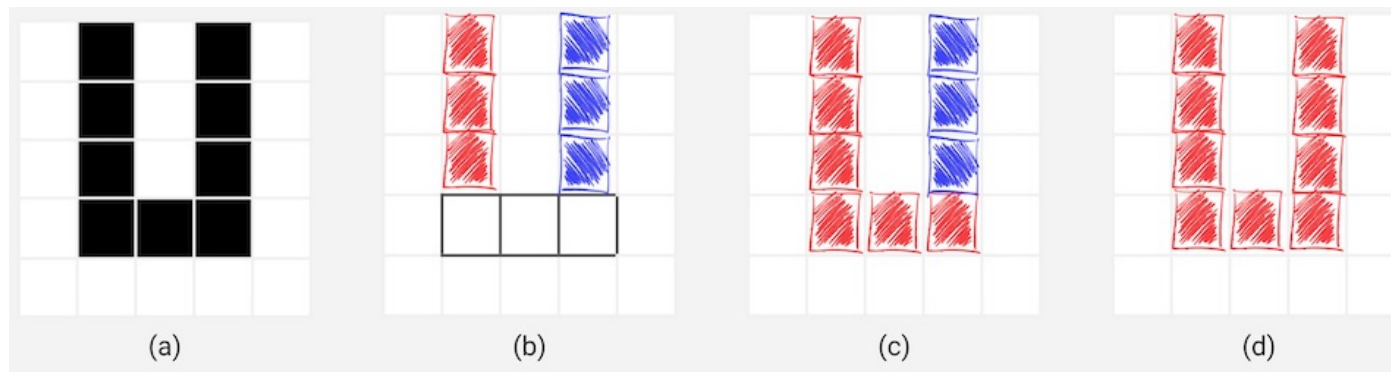


Figura 4: Exemplo de como se pode colorir uma ilha.

¹Esta é uma estratégia possível. Sugerimos que você pense em outras formas de resolver o problema.

Para visualizar imagens no formato PBM, ou converter uma imagem qualquer para este formato, você pode utilizar o [GIMP](https://www.gimp.org/).

Toda imagem PBM pode ser aberta em um editor de textos, e assim você pode visualizar os caracteres ASCII que constituem esse tipo de arquivo.

Tarefa

Neste laboratório, você deverá ler um mapa do jogo no formato de imagem **PBM** e determinar quantas ilhas (regiões) existem no mapa. A saída do seu programa será a quantidade de ilhas no mapa de entrada.

A primeira linha da entrada é sempre o identificador **P1**, referente ao formato do arquivo **PBM**. A segunda linha contém dois inteiros **W** e **H** que representam, respectivamente, a largura e altura da matriz de pixels que formam a imagem. A seguir, **H** linhas com **W** colunas cada contém os inteiros **0** e **1** que formam a imagem que será analisada.

A saída do seu programa será o número de ilhas contidas no mapa seguido de uma quebra de linha.

Observações da Tarefa

- Você *deve* inserir uma quebra de linha `\n` após imprimir o número total de ilhas do mapa.
- Um mapa tem como tamanho máximo **300px x 300px**.
- Os mapas não são, necessariamente, quadrados.
- Você pode utilizar `scanf("%c%c %d %d",&a,&b,&w,&h);` para ler as duas primeiras linhas da entrada.
- Uma ilha não contém buracos no seu interior, ou seja, não contém regiões brancas preenchidas pela ilha em toda sua volta.
- Considere que qualquer um dos oito pixels vizinhos conecta os pedaços de uma ilha. Por exemplo, a seguinte entrada apresenta apenas uma ilha:

```
P1
5 5
0 1 0 0 1
0 1 0 1 0
0 1 0 0 1
0 0 1 1 1
0 0 0 0 0
```

Exemplos

Notas:

Textos em azul designam dados de entrada, isto é, que devem ser lidos pelo seu programa.
Textos em preto designam dados de saída, ou seja, que devem ser impressos pelo seu programa.

Exemplo de execução 1:

P1

5	5			
0	1	0	0	0
0	1	0	0	0
0	1	0	0	1
0	0	1	1	1
1	0	0	0	0

2

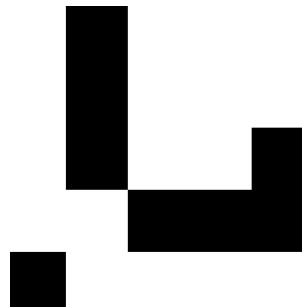


Figura 5: Mapa do exemplo 1.

Exemplo de execução 2:

[illegible]

10



Figura 6: Mapa do exemplo 2.

- O número máximo de submissões é 20.
- O arquivo **lab06.c** deve conter todo o seu programa.

- Para a realização dos testes automáticos, a compilação se dará da seguinte forma: `gcc lab06.c -o lab06 -Wall -Werror -ansi -pedantic`.
 - Não se esqueça de incluir no início do programa uma breve descrição dos objetivos, da entrada, da saída, seu nome, RA e turma.
 - Após cada submissão, você deve aguardar um minuto até poder submeter seu trabalho novamente.
 - Ao final deste laboratório, você terá aprendido como utilizar vetores multidimensionais.
-

CrITÉRIOS Importantes

O **não** cumprimento dos critérios abaixo acarretará em **nota zero na atividade**, independentemente dos resultados dos testes do SuSy.

- Sua solução deve atender todos os requisitos definidos no enunciado.
- Não serão aceitas soluções contendo estruturas não vistas em sala (para este laboratório, poderão ser utilizadas apenas variáveis simples, vetores, matrizes, operações de entrada e saída, operações aritméticas, desvios condicionais e laços). Observe que não podem ser implementadas funções além da `main`.
- Não é permitido o uso de `continue` e `break` (exceto em estruturas do tipo *switch-case*).
- Não é permitido o uso de variáveis globais.
- O único cabeçalho aceito para inclusão é `stdio.h`.