Processamento XML com Imagens Binárias

Leonardo Gideão Costa Rocha
Nicolas Vanz
03/11/2020

Índice Geral

1. Introdução

Esse projeto foi desenvolvido com o intuito de apresentar um método para contagem de componentes conexos de imagens binárias extraídas de arquivos XML.

Segue um conjunto de definições para melhor entendimento do projeto.

- Vizinhança-4: a vizinhança-4 de um pixel P de coordenada (x, y) é o conjunto de pixels adjacentes a P, isto é, os pixels cujas coordenadas são: (x + 1, y), (x – 1, y), (x, y + 1), ou (x, y – 1).
- Caminho: Um caminho de um pixel P ao Q é o conjunto de pixels $P = a_1, a_2, \ldots, a_n = Q$, tal que a_{i+1} está na vizinhança-4 de a_i. No contexto desse projeto o caminho pode apenas ser composto por pixels de cor branca, representados pelo valor 1 nos arquivos xml.
- Componente conexo: é o maior conjunto de pixels tal que para todo par de pixels P, Q desse conjunto é possível encontrar um caminho de P a Q.

A figura 1.1 ilustra um exemplo de matriz de pixels de entrada processada pela solução proposta. A figura 1.2 traz uma abordagem mais visual do que são os componentes conexos (representados em branco) que seriam encontrados considerando a figura 1.1 como uma imagem do arquivo xml usado como entrada do programa.

Figura 1.1

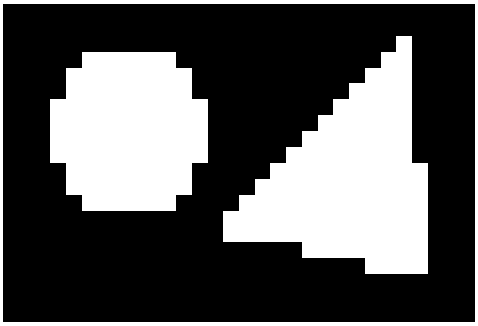


Figura 1.2

2. Desafios e Soluções

2.1. Validação dos Arquivos

2.1.1. Contextualização do Problema

Arquivos XML são construídos com base em tags de abertura (Ex.: <img>) e de fechamento (Ex.: </img>). Desse modo, para consistência da solução, devemos verificar se o arquivo a ser processado é válido ou não, isto é, se todas as tags foram abertas e fechadas corretamente.

2.1.2. Solução

Utilizamos uma estrutura de pilha para verificar se as tags de abertura e fechamento foram utilizadas de forma correta. Para maiores detalhes, ver a documentação e/ou código fonte da função *validate_file()*.

2.2. Organização dos Dados do Arquivos

2.2.1. Contextualização do Problema

Para facilitar a manipulação dos dados lidos do arquivo XML. Faz-se necessário a organização dos dados de entrada em conjuntos de dados. Assim pode-se processar cada imagem descrita no arquivo separadamente.

2.2.2. Solução

Criamos uma estrutura, que chamamos de Dataset, para representar cada imagem que é lida do arquivo XML. O processamento de cada imagem e sua atribuição a um Dataset é realizado pela função get_datasets(). Para maiores detalhes, ver a documentação e/ou código fonte da função *get_datasets()* e da estrutura Dataset.

2.3. Cálculo da Quantidade de Componentes Conexos

2.3.1. Contextualização do Problema

Esse é o desafio principal do projeto: contar a quantidade de componentes conexos de cada imagem do arquivo XML.

2.3.2. Solução

Utilizamos um algoritmo recursivo para o preenchimento dos componentes conexos. Sendo assim, toda vez que é encontrado um pixel não processado e com valor igual a 1, preenchemos a região e obtemos um componente conexo. Para maiores detalhes, ver a documentação e/ou código fonte das funções *get_conexes()* e *flood_fill()*.

3. Documentação

Índice dos módulos

Módulos

Lista de todos os módulos:

Índice dos namespaces

Lista de namespaces

Lista dos namespaces com uma breve descrição:

Índice dos componentes

Lista de componentes

Lista de classes, estruturas, uniões e interfaces com uma breve descrição:

Índice dos ficheiros

Lista de ficheiros

Lista de todos os ficheiros com uma breve descrição:

Documentação do módulo

estruturas de dados

Variáveis

structures::LinkedQueue< std::string > **fila** {}
structures::LinkedQueue< struct **Dataset** * > **datasets** {}
structures::LinkedStack< std::string > **stack** {}

Descrição detalhada

Documentação das variáveis

structures::LinkedQueue<struct Dataset*> datasets {}

Fila de todas as imagens organizadas em datasets.

structures::LinkedQueue<std::string> fila {}

Fila usada para armazenar todas as linhas do arquivo.

structures::LinkedStack<std::string> stack {}

Pilha para armazenamento das tags na validação do arquivo.

Documentação dos namespaces

Referência ao namespace structures

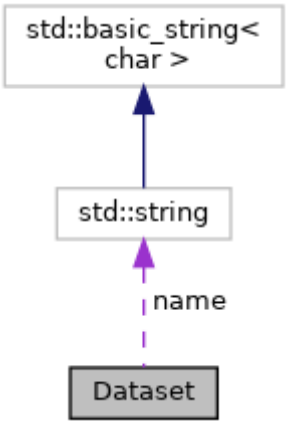
Componentes

class **LinkedQueue**
class **LinkedStack**

Documentação da classe

Referência à estrutura Dataset

Diagrama de colaboração para Dataset:



Atributos Públicos

std::string **name**
int **height**
int **width**
int ** **data**

Descrição detalhada

Dataset.

Representa uma imagem descrita no arquivo xml.

Documentação dos dados membro

int** Dataset::data

matriz de 0 e 1 da imagem

int Dataset::height

Altura da imagem.

std::string Dataset::name

Nome de imagem.

int Dataset::width

Largura da imagem.

A documentação para esta estrutura foi gerada a partir do seguinte ficheiro:

file_functions.cpp

Referência à classe Template structures::LinkedQueue< T >

`#include <fila.h>`

Membros públicos

LinkedQueue ()
~LinkedQueue ()
void **clear** ()
void **enqueue** (const T &data)
T **dequeue** ()
T & **front** () const
T & **back** () const
bool **empty** () const
std::size_t **size** () const
void **display** ()

Descrição detalhada

template<typename T>

class structures::LinkedQueue< T >

Classe que opera como uma fila encadeada

Documentação dos Construtores & Destrutor

template<typename T > structures::LinkedQueue< T >::LinkedQueue

Rotina construtora da fila.

template<typename T > structures::LinkedQueue< T >::~LinkedQueue

Rotina destrutora da fila.

Documentação dos métodos

template<typename T > T & structures::LinkedQueue< T >::back

Retorna o último elemento da fila.

template<typename T > void structures::LinkedQueue< T >::clear

Retira todos os elementos da fila.

template<typename T > T structures::LinkedQueue< T >::dequeue

Retira um elemento da fila.

Retorna

Dado removido da fila

Nota

O primeiro elemento da fila é o retirado

template<typename T > void structures::LinkedQueue< T >::display

Imprime o conteúdo da fila.

Nota

Utilizado como recurso para debugar

template<typename T > bool structures::LinkedQueue< T >::empty

Verifica se a fila está vazia.

Retorna

true caso a lista esteja vazia ou false caso não esteja vazia

template<typename T > void structures::LinkedQueue< T >::enqueue (const T & *data*)

Adiciona um elemento à fila.

Parâmetros

data	dado tipo T a ser enfileirado

Nota

O novo elemento é adicionado no final da fila

template<typename T > T & structures::LinkedQueue< T >::front

Retorna o primeiro dado da fila.

template<typename T > std::size_t structures::LinkedQueue< T >::size

Retorna a quantidade de elementos da fila.

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

fila.h

Referência à classe Template structures::LinkedStack< T >

`#include <pilha.h>`

Membros públicos

LinkedStack ()
~LinkedStack ()
void **clear** ()
void **push** (const T &data)
T **pop** ()
T & **top** () const
bool **empty** () const
std::size_t **size** () const
void **display** ()

Descrição detalhada

template<typename T>

class structures::LinkedStack< T >

Classe que opera como uma pilha encadeada

Documentação dos Construtores & Destrutor

template<typename T > structures::LinkedStack< T >::LinkedStack

Rotina construtora da pilha.

template<typename T > structures::LinkedStack< T >::~LinkedStack

Rotina destrutora da pilha.

Documentação dos métodos

template<typename T > void structures::LinkedStack< T >::clear

Remove todos os elementos da pilha.

template<typename T > void structures::LinkedStack< T >::display

Imprime o conteúdo da pilha.

Nota

Utilizado como recurso para debugar

template<typename T > bool structures::LinkedStack< T >::empty

Verifica se a pilha está vazia.

Retorna

true caso a pilha esteja vazia ou false caso não esteja vazia

template<typename T > T structures::LinkedStack< T >::pop

Retira um dado da pilha.

Retorna

Dado retirado da pilha

Nota

O dado é retirado do topo da pilha

template<typename T > void structures::LinkedStack< T >::push (const T & *data*)

Adiciona um elemento à pilha.

Parâmetros

data	dado a ser empilhado

Nota

O dado é colocado no topo da pilha

template<typename T > std::size_t structures::LinkedStack< T >::size

Retorna a quantidade de elementos da pilha.

template<typename T > T & structures::LinkedStack< T >::top

Retorna o dado no topo da pilha.

A documentação para esta classe foi gerada a partir do seguinte ficheiro:

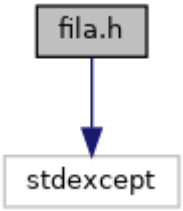
pilha.h

Documentação do ficheiro

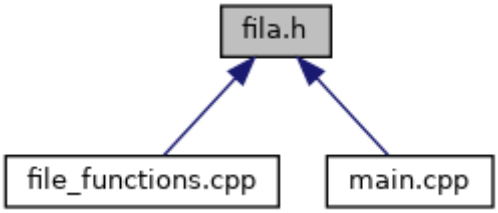
Referência ao ficheiro fila.h

```
#include <stdexcept>
```

Diagrama de dependências de inclusão para fila.h:



Este grafo mostra quais são os ficheiros que incluem directamente ou indirectamente este ficheiro:



Componentes

class **structures::LinkedQueue< T >**

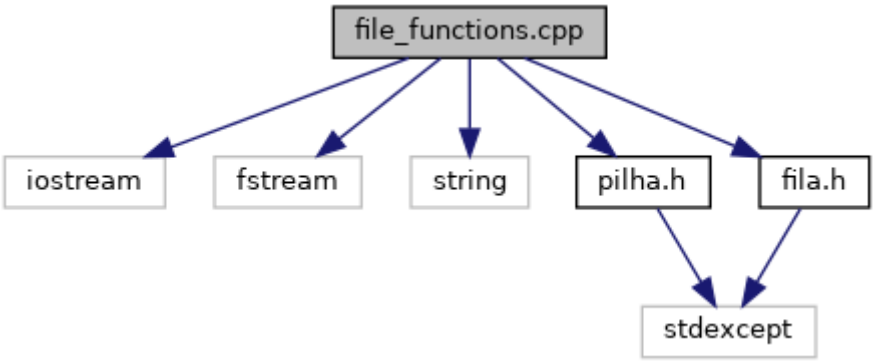
Namespaces

structures

Referência ao ficheiro file_functions.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include "pilha.h"
#include "fila.h"
```

Diagrama de dependências de inclusão para file_functions.cpp:



Componentes

struct **Dataset**

Funções

bool **is_open_tag** (std::string tag)
bool **validate_file** (char *filename)
void **get_tags** (char *filename)
void **get_datasets** ()
void **flood_fill** (struct **Dataset** *dataset, int x, int y)
void **display** (struct **Dataset** *p)
int **get_conexes** (struct **Dataset** *dataset)
void **results** ()

Variáveis

structures::LinkedQueue< std::string > **fila** {}
structures::LinkedQueue< struct **Dataset** * > **datasets** {}
structures::LinkedStack< std::string > **stack** {}

Documentação das funções

void display (struct Dataset * *p*)

Imprime todos os dados de um dataset.

Nota

Utilizado como ferramenta para debugar

Parâmetros

p	Ponteiro do dataset a ser mostrado

void flood_fill (struct Dataset * *dataset*, int *x*, int *y*)

Preenche um componente conexo da matriz.

Percorre todos os elementos iguais a 1 adjacentes à coordenada (x , y)

Parâmetros

dataset	Ponteiro para o conjunto de dados a ser analisado
x	Linha da matriz a ser analisada
y	Coluna da matriz a ser analisada

Nota

Essa função usa um meio recursivo para preencher o componente

int get_conexes (struct Dataset * *dataset*)

Retorna a quantidade de componentes conexos da imagem.

Parâmetros

dataset	Ponteiro do dataset a ser analisado

Retorna

Quantidade de componentes conexos da imagem representada pelo dataset

Nota

É chamado um algoritmo recursivo para o preenchimento do componente conexo

Veja também

flood_fill()

void get_datasets ()

Organiza as tags e seus valores em conjuntos de dados.

Nota

É utilizada uma fila para armazenar os conjuntos de dados (datasets)

Essa função só é chamada depois que as tags e seus valores foram processados por **get_tags()**

Veja também

get_tags()

void get_tags (char * *filename*)

Extrai cada elemento do arquivo Percorre todo o arquivo e extrai as tags e seus valores.

Parâmetros

filename	Nome do arquivo de onde serão retiradas as tags.

Nota

Uma fila é utilizada para armazenar cada elemento (tag ou valor da tag)

Essa rotina só é chamada depois do arquivo ser validado por **validate_file()**

Veja também

validate_file()

bool is_open_tag (std::string *tag*)

Verifica se uma tag é de abertura ou de fechamento.

Parâmetros

tag	tag a ser analisada

Retorna

true caso a tag seja de abertura ou false caso a tag seja de fechamento

void results ()

Imprime os resultados dos cálculos de cada conjunto de dados.

bool validate_file (char * *filename*)

Verifica se o arquivo é válido.

Parâmetros

filename	Nome do arquivo que sera analisado

Retorna

true caso o arquivo seja válido ou false caso o arquivo não seja válido

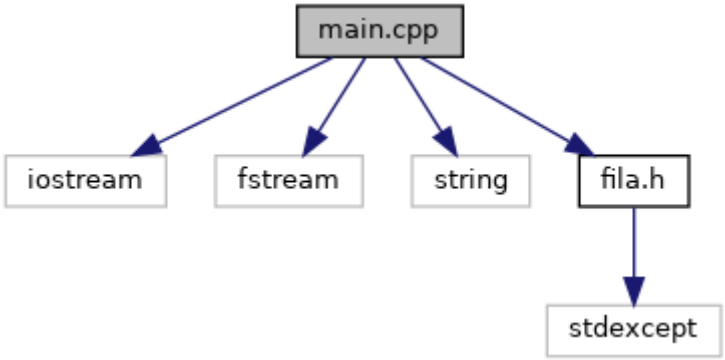
Nota

Uma estrutura de pilha é utilizada para a validação

Referência ao ficheiro main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include "fila.h"
```

Diagrama de dependências de inclusão para main.cpp:



Funções

bool **validate_file** (char *filename)
void **get_tags** (char *filename)
void **get_datasets** ()
void **results** ()
int **main** ()

Documentação das funções

void get_datasets ()

Organiza as tags e seus valores em conjuntos de dados.

Nota

É utilizada uma fila para armazenar os conjuntos de dados (datasets)

Essa função só é chamada depois que as tags e seus valores foram processados por **get_tags()**

Veja também

get_tags()

void get_tags (char * *filename*)

Extrai cada elemento do arquivo Percorre todo o arquivo e extrai as tags e seus valores.

Parâmetros

filename	Nome do arquivo de onde serão retiradas as tags.

Nota

Uma fila é utilizada para armazenar cada elemento (tag ou valor da tag)

Essa rotina só é chamada depois do arquivo ser validado por **validate_file()**

Veja também

validate_file()

int main ()

void results ()

Imprime os resultados dos cálculos de cada conjunto de dados.

bool validate_file (char * *filename*)

Verifica se o arquivo é válido.

Parâmetros

filename	Nome do arquivo que sera analisado

Retorna

true caso o arquivo seja válido ou false caso o arquivo não seja válido

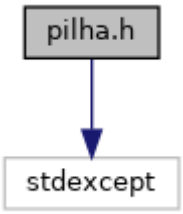
Nota

Uma estrutura de pilha é utilizada para a validação

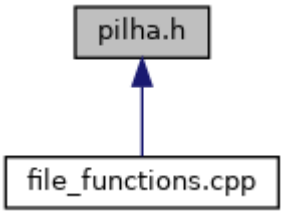
Referência ao ficheiro pilha.h

`#include <stdexcept>`

Diagrama de dependências de inclusão para pilha.h:



Este grafo mostra quais são os ficheiros que incluem directamente ou indirectamente este ficheiro:



Componentes

class **structures::LinkedStack< T >**

Namespaces

structures