**UNIVERSIDADE FEDERAL DE SANTA CATARINA – UFSC- CTC**

**DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA**

**PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

**PROJETO E ANÁLISE DE ALGORITMO**

**Prof. Alexandre Gonçalves Silva**

**Aluno: Osmar de Oliveira Braz Junior**

# Questão 9

9. A MEDIANA(A) é um algoritmo que devolve o i-ésimo menor valor, sendo (mediana inferior) ou se preferir, (mediana superior), de um dado vetor *A* de n elementos.

(a) Implemente duas versões deste algoritmo em qualquer linguagem de programação:

– **versão 1:** em tempo Ω(n log n)

Foi desenvolvido utilizando o algoritmo heapsort em Java. O código fonte está na pasta **questao9\letra\_a\_nlogn.**

public class Principal {

private static void maxHeapify(int A[],int n, int i) {

int maior = 0;

int e = 2 \* i ;

int d = 2 \* i + 1;

if ((e < n) && (A[e]> A[i])){

maior = e;

} else {

maior = i;

}

if ((d < n) && (A[d]>A[maior])){

maior = d;

}

if (maior != i) {

troca(A,i,maior);

maxHeapify(A,n,maior);

}

}

private static void maxHeap(int A[],int n) {

int x;

if ((n%2)==0){

x = n/2;

} else {

x = (n -1)/2;

}

for(int i=x;i>=0;i--){

maxHeapify(A,n,i);

}

}

private static void heapsort(int A[],int n) {

maxHeap(A,n);

int m = n;

for(int i=n-1;i>=0;i--){

troca(A,0,i);

m = m - 1;

maxHeapify(A,m,0);

}

}

private static int medianaNLogN(int A[], int p, int r) {

int n = r;

int m = n/2;

heapsort(A,r); //Ordena todo o vetor para encontrar a mediana

if (n%2 == 1) {

return (p + m);

} else {

return ((p + m) + (p + m - 1))/2;

}

}

//Realiza a troca de posição de elementos

private static void troca(int A[],int i, int j) {

int aux;

aux = A[i];

A[i] = A[j];

A[j] = aux;

}

public static void main(String args[]) {

int r = 10;

int A[] = new int[r];

A[0]=99;

A[1]=33;

A[2]=55;

A[3]=77;

A[4]=11;

A[5]=22;

A[6]=88;

A[7]=66;

A[8]=33;

A[9]=44;

System.out.println(">>> Mediana <<<");

System.out.println("Vetor A antes: ");

for (int i=0; i < r; i++) {

System.out.println((i) + " - " + A[i]);

}

int q = medianaNLogN(A, 0, A.length);

System.out.println("q: "+q);

System.out.println("Mediana: "+A[q]);

System.out.println("Vetor A após: ");

for (int i=0; i < r; i++) {

System.out.println((i) + " - " + A[i]);

}

}

}

– **versão 2:** em tempo médio O(n)

Foi desenvolvido utilizando o algoritmo seleciona aleatório em Java. O código fonte está na pasta **questao9\letra\_a\_n.**

import java.util.Random;

public class Principal {

public static int aleatorio(int inicio, int fim) {

return (int) Math.ceil(Math.random() \* (fim - inicio + 1)) - 1 + inicio;

}

private static int particione(int A[],int p, int r) {

int pivot = A[r];

int i = p-1;

for(int j=p;j<=r-1;j++) {

if(A[j] <= pivot) {

i = i + 1;

troca(A,i,j);

}

}

troca(A,i+1,r);

return i+1;

}

private static int particionaAleatorio(int A[],int p, int r) {

int j=aleatorio(p,r);

troca(A,j,r);

return particione(A,p,r);

}

private static int selecionaAleatorio(int A[],int p, int r, int i) {

if(p==r){

return A[p];

}

int q = particionaAleatorio(A,p,r);

int k = q - p + 1;

if (i==k){

return A[q];

} else {

if (i < k){

return selecionaAleatorio(A,p,q-1,i);

} else {

return selecionaAleatorio(A,q+1,r,i-k);

}

}

}

private static int medianaN(int A[],int p, int r) {

int n = r;

int m = n/2;

selecionaAleatorio(A,p,n-1,m); //Particiona todo o vetor para encontrar a mediana

if (n%2 == 1) {

return (p + m);

} else {

//Particiona todo o vetor para encontrar a mediana

return ((p + m) + (p + m - 1))/2;

}

}

//Realiza a troca de posição de elementos

private static void troca(int A[],int i, int j) {

int aux;

aux = A[i];

A[i] = A[j];

A[j] = aux;

}

public static void main(String args[]) {

int r = 10;

int A[] = new int[r];

A[0]=99;

A[1]=33;

A[2]=55;

A[3]=77;

A[4]=11;

A[5]=22;

A[6]=88;

A[7]=66;

A[8]=33;

A[9]=44;

System.out.println(">>> Mediana <<<");

System.out.println("Vetor A antes: ");

for (int i=0; i < r; i++) {

System.out.println((i) + " - " + A[i]);

}

int q = medianaN(A, 0, A.length);

System.out.println("q: "+q);

System.out.println("Mediana: "+A[q]);

System.out.println("Vetor A após: ");

for (int i=0; i < r; i++) {

System.out.println((i) + " - " + A[i]);

}

}

}

(b) (**OPCIONAL** – pontuação extra) Implemente duas versões do **filtro de mediana**, considerando os dois algoritmos desenvolvidos no item (a), para matrizes bidimensionais m x n de inteiros 0 <= f (i, j) <= 255, sendo 0 <=i < m, 0 <= j < n, supondo janela de filtro com vizinhança parametrizável de p x q, sendo 2 <= p < m e 2 <= q <n. A técnica, exemplos e código (em C) podem ser consultados no seguinte documento: https://www.ime.usp.br/~reverbel/ccm118-12/eps/ep4.pdf.

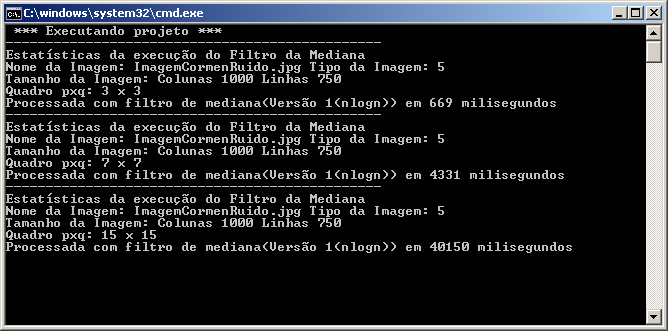
Avalie o tempo de execução real (por exemplo, em segundos) das duas versões implementadas do filtro para uma matriz (imagem) suficientemente grande (>= 640 x 480 pixels) e para diferentes escolhas de p e q (por exemplo, 3, 7, 15, . . . ).

**R.:**

O programa foi desenvolvido em Java para desktop e permite escolher quadros de p por q de tamanhos 3, 5 e 16. Além de permitir escolher qual o filtro de mediana a ser aplicado na imagem. Depois de escolher os parâmetros basta clicar no botão processar para executar o filtro nas imagens do diretório padrão. O usuário pode selecionar uma imagem a ser processada clicando no botão Abrir Imagem e depois clicando no botão processar. Após o processamento duas janelas serão exibidas com a imagem original e a imagem processada pelo filtro. O programa pode ser executado pelo arquivo **questao9b.jar** que se encontra no diretório **questao9/letra\_b\_imagem**.



As estatísticas são apresentadas no console de execução.

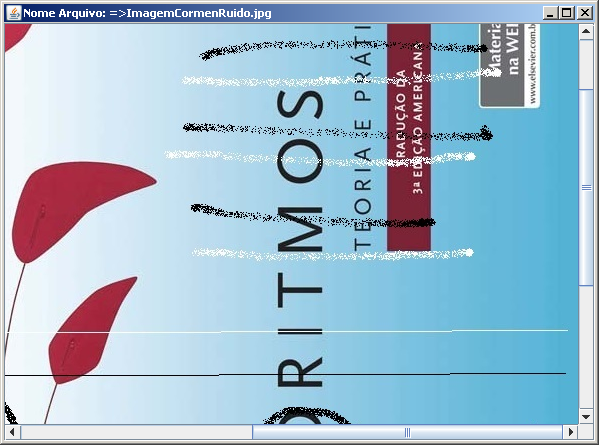


Foram realizados testes e estes são apresentados no quadro a seguir

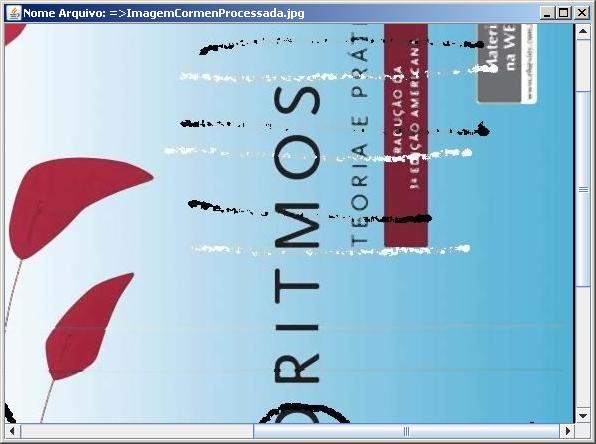
|  |  |  |  |
| --- | --- | --- | --- |
| **Imagem**  **ColunaxLinha** | **Filtro** | **Quadro**  **pxq** | **Tempo** |
| 1000x750 | n log n | 3x3 | 635 milissegundos |
| 1000x750 | n log n | 7x7 | 4353 milissegundos |
| 1000x750 | n log n | 15x15 | 28180 milissegundos |
| 1000x750 | n | 3x3 | 1134 milissegundos |
| 1000x750 | n | 7x7 | 4867 milissegundos |
| 1000x750 | n | 15x15 | 37530 milissegundos |

Resultado na imagem do filtro n log n sobre a imagem:

A seguir a imagem original com ruído:



E o resultado da imagem processada



O filtro da mediana está sendo aplicado nos 3 canais de cores da imagem (rgb). O código fonte que percorre a imagem aplicando o filtro da mediana está apresentado a seguir:

**// Processa valores da imagem de entrada e armazena na imagem de saida**

for(int i=(p/2); i < (alturaImagem - (p/2)); i++){

for(int j=(q/2); j <(larguraImagem - (q/2)); j++){ for(int canal=0;canal<=2; canal++){

**//Conta os elementos inseridos no vetor**

int x = 0;

**//Vetor base para achar a mediana**

int V[] = new int[p\*q];

**//Recupera os elementos para o quadro**

for (int k = i - (p/2); k <= i + (p/2); k++) {

**//Calcula o inicio do intervalo**

int inicio = j - (q/2);

**//Calcula o fim do intervalo**

int fim = j + (q/2) + 1;

for (int elemento = inicio; elemento < fim; elemento++){

V[x]=raster.getSample(elemento,k,canal);

x = x + 1;

}

}

int t = 0;

**//Seleciona o tipo do filtro da mediana**

if (tipoFiltro==0){

t = medianaNLogN(V,0,V.length);

} else {

t = medianaN(V,0,V.length);

}

int mediana = V[t];

**//Salva a mediana na imagem de saida**

wraster.setSample(j,i,canal,mediana);

}

}

}

O código fonte está em anexo na pasta **questao9\letra\_b\_imagem**