

Fundamentos de Sistemas de Información

Dr. Marco Aurelio Nuño Maganda

Universidad Politécnica de Victoria
Maestría en Ingeniería
Agosto – Diciembre 2024

mnunom@upv.edu.mx

9 de septiembre de 2024

- Doctor en Ciencias Computacionales por parte del INAOE (2009).
- Profesor de Tiempo Completo de la UPV desde 2009.
- Miembro del Sistema Nacional de Investigadores - Nivel Candidado (2014-2016), Nivel I (2020-2022), Nivel I (2023-2027)
- 17 tesis dirigidas a nivel maestría.
- Asignaturas impartidas en el pasado
 - Licenciatura: Cómputo en Dispositivos Mviles, Graficación por Computadora Avanzada, Lenguajes y Automátas, Programación Orientada a Objetos
 - Maestría: Visión por computadora, Tópicos Selectos de Imagenología, Fundamentos de Sistemas de Información
- Miembro del Núcleo Académico Básico (NAB) de la maestria en Ingeniería de la UPV.

- Nombre de la clase: **Fundamentos de Sistemas de Información-Agosto – Diciembre 2024**
- Código de clase en Classroom: **siudfqm**
- Enlace Meet para sesiones no presenciales:
<https://meet.google.com/gxn-insx-bbf>

- Días y horas de clase

Clave de Grupo	Lunes	Martes	Miércoles	Jueves	Viernes
IM-20375	12:05 - 13:55		12:05 - 13:55		

- Fechas Importantes:

- Inicio de Cursos: 2/Septiembre
- Fin de Cursos: 13/Diciembre
- Dias no hábiles oficiales: 16 de septiembre (lunes), 1 de octubre (martes) y 18 de noviembre (lunes)

- Se recomienda puntualidad y asistencia a las sesiones.
- Respecto hacia el profesor y hacia sus compañeros y compañeras.
- No se permite el ingreso y/o ingestión de **Alimentos** ni **Bebidas**.

- Pase de Lista al Inicio de cada sesión.
- Para actividades en línea, debe utilizar su cuenta institucional de la universidad, ya que no se autorizará el ingreso de ningún participante externo a la universidad.
- Para justificar una inasistencia, es necesario cumplir con los siguientes pasos:
 - Mandar correo al profesor, justificando su inasistencia (solo por motivos médicos)

Asistencias mínimas para aprobar asignatura

- Es obligatorio tener un 90 % de asistencia. De no ser así, el estudiante PIERDE su derecho de ser EVALUADO

Alumnos VIPs

En caso de tener un empleo formal dentro o fuera de la ciudad, es necesario entregar una **constancia laboral** que acredite el horario que se esta cubriendo (en el caso de locales, este horario se debe empalmar con el de la materia). En esa constancia debe acreditar que se esta haciendo labores de manera presencial en tal ubicacion. Esto lo dispensa solo del requisito de las asistencias, mas no de los proyectos que deban entregarse. Incluso pudiera solicitarle presentar avance de manera “remota” durante alguna de las clases. Enviar esa constancia con copia para el director de carrera.

Justificación de Faltas Extraordinaria

- Todo aquel estudiante que realice una estancia de investigación, visita académica, visita a empresa, evento deportivo (por parte de la Universidad), o cualquier otra actividad académica, debe negociar una extesión máxima de **7 días naturales** de prórroga para entrega de proyectos individuales, en equipo o asignaciones especiales, respaldado por algun documento oficial que acredite dicha actividad.
- Ninguna actividad de las anteriormente mencionadas **es causa de EXENSIÓN** de los proyectos solicitados en la clase, ni generará ningún cambio o ajuste a sus calificaciones por participar en dicho evento. De no cumplir con los plazos de entrega establecidos, deberá asumir las consecuencias de no entregar los proyectos o trabajo solicitado en el tiempo establecido.
- Las inasistencias a clase quedan justificadas solo **durante el desarrollo del evento y los traslados**. Para justificar dichas inasistencias, debe hacerlo mediante el procedimiento establecido para dicha justificación.

- 1 Introducción a las tecnologías de la información
- 2 Herramientas para escritura de documentos científicos
- 3 Programación orientada a objetos e interfaces de usuario
- 4 Programación en dispositivos móviles
- 5 Tópicos avanzados

Para cada unidad del curso, se consideran los siguientes aspectos:

- Exposición de Tema Selecto 1- 10 %
- Exposición de Tema Selecto 2- 10 %
- Proyecto Individual 1 - 20 %
- Proyecto Individual 2 - 20 %
- Proyecto Individual 3 - 20 %
- Proyecto Individual 4 - 20 %

Para aprobar el curso, es obligatorio entregar todos los proyectos solicitados

Para cada unidad, habra sesiones de “teoria”, sesiones de seguimiento de proyectos y sesiones de esparcimiento

- En las sesiones de teoria, el profesor presentara uno o varios temas
- En las sesiones de seguimiento de proyectos, de manera aleatoria se nombrara al integrante de equipo individual o en equipo. En el caso de que un integrante individual no responda, se le bajarán 5 puntos a su calificación del proyecto
- En las sesiones de esparcimiento, se permitirá a los estudiantes trabajar en proyectos pendientes, pero se contabilizará la asistencia.

Sesiones de Seguimiento de proyectos

- Programadas generalmente una semana despues de haber sido asignado el proyecto
- Los estudiantes deben exponer el avance del proyecto, así como los retos y dificultades sorteados

Acerca de los proyectos

- Proyectos diferentes para los integrantes de la clase
- Asignados de manera aleatoria para cada integrante de la clase

- Para cada práctica realizada, entregar un documento (**únicamente en formato PDF***) con las siguientes secciones:
 - Introducción
 - Desarrollo Experimental
 - Resultados
 - Conclusiones
 - **Referencias**
- Para GENERAR este reporte es necesario utilizar la plantilla en LATEX (**únicamente usando LATEX***) localizada en el siguiente enlace:
<https://www.overleaf.com/read/dgkhvfwynygvc>

- Bajo ninguna circunstancia deben incluir **CÓDIGO FUENTE**. Si pueden incluir diagrama de flujo, Pseudocódigo, Diagrama E-R, Diagrama de Clases, de Casos de USO, etc. De incluir código fuente, solo generará una penalización a la calificación del proyecto.
- En caso de trabajos individuales o en EQUIPO, deben emplear la plantilla LaTeX que se provee. En caso de utilizar algo diferente a LaTeX u otra plantilla de LaTeX, la calificación proporcional del informe será **DESESTIMADA**.

Ponderación del Informe en la Calificación del Proyecto

- Proyecto: 66 Puntos
 - Ejecución y Funcionalidad: 45 Puntos
 - Modularidad: 13 Puntos
 - Documentación: 8 Puntos
- Informe: 34 Puntos
 - Uso adecuado de Latex: 5 Puntos
 - Organización y Redacción: 6 Puntos
 - Referencias en formato adecuado: 8 Puntos
 - Evidencia del trabajo realizado: 8 Puntos
 - Sin faltas de ortografía ni errores de dedo: 7 Puntos

Entregables de proyecto individual

En cada entrega, subir un archivo .ZIP, cuyo nombre de archivo debe seguir la siguiente especificación (todo en minúsculas , sin ESPACIOS):

- **Clave de GRUPO (incluir guión medio)**
- **Nombre del integrante iniciando por apellido paterno, SIN ESPACIOS y separado por guion bajo**
- *clavegrupo_nuno_maganda_marco_aurelio.zip*,
clavegrupo_nuno_maganda_marco_aurelio.pdf, etc.
- El contenido de dicho archivo debe ser el siguiente:
 - 1 Archivo .ZIP con el código fuente.
 - 2 Archivo instalador de la aplicación (.APK).
 - 3 Archivo PDF con el informe.
- Mismo formato de NOMBRE de archivo del ENTREGABLE principal para el nombre de los archivos al interior del ZIP

- Este es un ejemplo, pero nunca falta el listo que lo entrega tal cual!
- **Archivo Principal:** iti-271086_nuno_maganda_marco_aurelio.zip
- Contenido de dicho archivo:
 - **iti-271086_nuno_maganda_marco_aurelio.zip** (Codigo fuente)
 - **iti-271086_nuno_maganda_marco_aurelio.apk** (Instalable)
 - **iti-271086_nuno_maganda_marco_aurelio.pdf** (Informe)

Cuatro cuatrimestres ignorando estas instrucciones, ya debería poner nombres y apellidos. El que lo haga serán sentadillas o páginas, ustedes escojan!

En el caso de nombres y apellidos acentuados, con diéresis o con virgulilla (~), sustituir de acuerdo con las siguientes reglas:

- Sustituir N/n por Ñ/ñ
- Sustituir A/a por Á/á
- Sustituir E/e por É/é
- Sustituir I/i por Í/í
- Sustituir O/o por Ó/ó
- Sustituir U/u por Ú/ú
- Sustituir U/u por Ü/ü

- Se debe extraer utilizando el SCRIPT para dicho propósito (SOLO EXISTE HAY PARA LINUX, usuarios Windows deben programar el SUYO o replicar los pasos del SCRIPT Linux para sus entrega) que se proveerá en el CLASSROOM para extraer los archivos necesarios
- **De ninguna manera se debe compactar la carpeta completa del PROYECTO. De hacer esto, habrá una penalización de 20 PUNTOS**

Fechas importantes de entrega de proyectos (1)

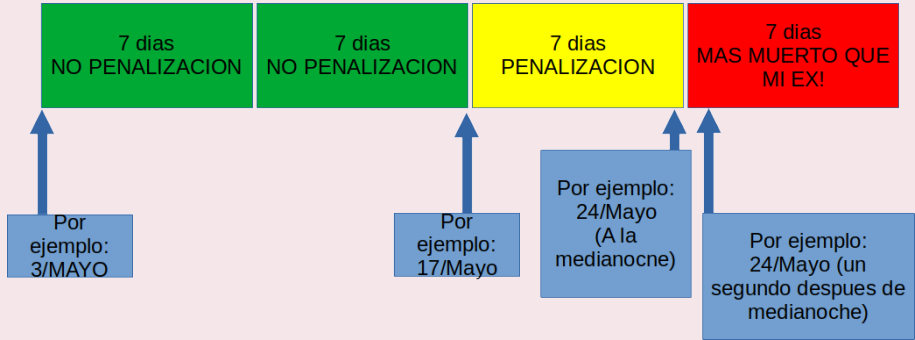
- Fecha de asignación: fecha en que se da a conocer al grupo el trabajo a elaborar
- Fecha de entrega sin penalización: 14 días naturales después de la fecha de asignación
- Proyecto entregado después de la fecha de penalización se le aplica una penalización de 25 PUNTOS
- Fecha de cierre: 21 días naturales después de la fecha de asignación.

Regla “CANTU”

- Ningún proyecto será revisado después de la fecha de cierre. Se programarán las entregas para cerrar y no permitir entregas tardías.

Fechas importantes de entrega de proyectos (2)

Grafo "DAFNE"



- En el momento de publicar la tarea, se incluirá la fecha para no penalización y fecha de cierre.

¿Es posible obtener una calificación negativa?

SI

Calificación asignada después de revisión	70
(-) Debería entregarse 17/Mayo, lo entregó 24/Mayo (1 minuto antes de la medianoche)	25
(-) Nombre Archivo MAL	8
(-) Tipo de Archivo MAL	8
(-) Estructura de Directorios INCORRECTA	8
(-) Faltas o Errores en Script	8
(-) ZIPs dentro del ZIP	8
(-) Incluir ejecutables (aplica solo cuando el lenguaje es C++)	8
(-) Penalización por Fragmentación de Equipo	25
Calificación Final	-28

LINUX (Recomendado)

Recomendaciones

- Es recomendado instalarlo, por cuestión de desempeño y por las herramientas que se utilizarán.
- Si no quieren formatear computadora, se recomienda utilizar un HD booteable (SSD con persistencia) y bootear desde su laptop o computadora.
- Si lo instalan de manera nativa, puede ser cualquier distribución (**Mint, Ubuntu, Lubuntu, Xubuntu, Debian**).

Uso de Windows

Bajo su propia responsabilidad y riesgo

Sobre una instalación de Linux, se debe instalar lo siguiente:

- Python3
- PyQt6
- LaTeX para edición de reportes
- Android Studio
- Scrcpy (<https://github.com/Genymobile/scrcpy>)

Se buscan integrantes para ingresar al
Salon de la fama del PLAGIO

- Reprobación automática a quien reproduzca códigos de otros compañeros y los reporte como suyos, además de una nota en su expediente con copia para el consejo de calidad



- Reprobación automática a quien copie códigos de Internet y los reporte como suyos, además de una nota en su expediente con copia para el consejo de calidad



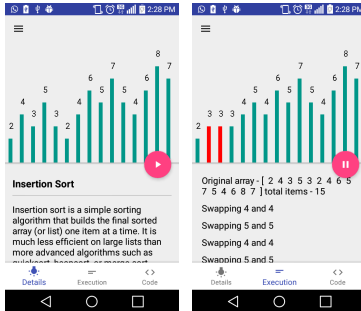
```

14 #include <iostream>
15 #include <vector>
16 #include <string>
17 #include <algorithm>
18 #include <math>
19 #include <conio.h>
20 #include <windows.h>
21 #include <opencv2/opencv.hpp>
22 #include <opencv2/imgproc/imgproc.hpp>
23 #include <opencv2/highgui/highgui.hpp>
24 using namespace cv;
25 using namespace std;
26 using namespace cv;
27
28 //funcion que inicializa todos los valores a 0
29 void initHist(Mat image, int histogram[])
30 {
31     // initialize all intensity values to 0
32     for(int i = 0; i < 256; i++)
33     {
34         histogram[i] = 0;
35     }
36 }
37
38 //calculando el numero de pixeles de cada nivel de intensidad
39 int getHist(Mat image)
40 {
41     for(int i = 0; i < image.rows; i++)
42     {
43         for(int j = 0; j < image.cols; j++)
44         {
45             histogram[image.at<uchar>(i,j)]++;
46         }
47     }
48 }
49
50 //funcion para calcular el histograma
51 void combiHist(Mat image, int histogram[])
52 {
53     combiHistogram[0] = histogram[0];
54     for(int i = 1; i < 256; i++)
55     {
56         combiHistogram[i] = combiHistogram[i-1] + histogram[i];
57     }
58 }
59
60 //funcion que muestra el histograma
61 void showHist(Mat image, int histogram[])
62 {
63     int hist[256];
64     for(int i = 0; i < 256; i++)
65     {
66         hist[i] = histogram[i];
67     }
68 }
69
70 //calculando los histogramas
71 int hist_x = 0; int hist_y = 0;
72 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
73
74 //calculando la suma de intensidades del histograma
75 int sum = 0;
76 for(int i = 0; i < 256; i++)
77 {
78     sum = hist[i];
79 }
80
81 //normalizando el histograma entre 0 y 255
82 for(int i = 0; i < 256; i++)
83 {
84     hist[i] = (hist[i] * 255) / sum;
85 }
86
87 //calculando el histograma normalizado
88 void normHist(Mat image, int histogram[])
89 {
90     for(int i = 0; i < 256; i++)
91     {
92         histogram[i] = hist[i];
93     }
94 }
95
96 //funcion que muestra el histograma normalizado
97 void showHist(Mat image, int histogram[])
98 {
99     int hist[256];
100     for(int i = 0; i < 256; i++)
101     {
102         hist[i] = histogram[i];
103     }
104 }
105
106 //calculando los histogramas
107 int hist_x = 0; int hist_y = 0;
108 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
109
110 //calculando la suma de intensidades del histograma
111 int sum = 0;
112 for(int i = 0; i < 256; i++)
113 {
114     sum = hist[i];
115 }
116
117 //normalizando el histograma entre 0 y 255
118 for(int i = 0; i < 256; i++)
119 {
120     hist[i] = (hist[i] * 255) / sum;
121 }
122
123 //calculando el histograma normalizado
124 void normHist(Mat image, int histogram[])
125 {
126     for(int i = 0; i < 256; i++)
127     {
128         histogram[i] = hist[i];
129     }
130 }
131
132 //funcion que muestra el histograma normalizado
133 void showHist(Mat image, int histogram[])
134 {
135     int hist[256];
136     for(int i = 0; i < 256; i++)
137     {
138         hist[i] = histogram[i];
139     }
140 }
141
142 //calculando los histogramas
143 int hist_x = 0; int hist_y = 0;
144 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
145
146 //calculando la suma de intensidades del histograma
147 int sum = 0;
148 for(int i = 0; i < 256; i++)
149 {
150     sum = hist[i];
151 }
152
153 //normalizando el histograma entre 0 y 255
154 for(int i = 0; i < 256; i++)
155 {
156     hist[i] = (hist[i] * 255) / sum;
157 }
158
159 //calculando el histograma normalizado
160 void normHist(Mat image, int histogram[])
161 {
162     for(int i = 0; i < 256; i++)
163     {
164         histogram[i] = hist[i];
165     }
166 }
167
168 //funcion que muestra el histograma normalizado
169 void showHist(Mat image, int histogram[])
170 {
171     int hist[256];
172     for(int i = 0; i < 256; i++)
173     {
174         hist[i] = histogram[i];
175     }
176 }
177
178 //calculando los histogramas
179 int hist_x = 0; int hist_y = 0;
180 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
181
182 //calculando la suma de intensidades del histograma
183 int sum = 0;
184 for(int i = 0; i < 256; i++)
185 {
186     sum = hist[i];
187 }
188
189 //normalizando el histograma entre 0 y 255
190 for(int i = 0; i < 256; i++)
191 {
192     hist[i] = (hist[i] * 255) / sum;
193 }
194
195 //calculando el histograma normalizado
196 void normHist(Mat image, int histogram[])
197 {
198     for(int i = 0; i < 256; i++)
199     {
200         histogram[i] = hist[i];
201     }
202 }
203
204 //funcion que muestra el histograma normalizado
205 void showHist(Mat image, int histogram[])
206 {
207     int hist[256];
208     for(int i = 0; i < 256; i++)
209     {
210         hist[i] = histogram[i];
211     }
212 }
213
214 //calculando los histogramas
215 int hist_x = 0; int hist_y = 0;
216 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
217
218 //calculando la suma de intensidades del histograma
219 int sum = 0;
220 for(int i = 0; i < 256; i++)
221 {
222     sum = hist[i];
223 }
224
225 //normalizando el histograma entre 0 y 255
226 for(int i = 0; i < 256; i++)
227 {
228     hist[i] = (hist[i] * 255) / sum;
229 }
230
231 //calculando el histograma normalizado
232 void normHist(Mat image, int histogram[])
233 {
234     for(int i = 0; i < 256; i++)
235     {
236         histogram[i] = hist[i];
237     }
238 }
239
240 //funcion que muestra el histograma normalizado
241 void showHist(Mat image, int histogram[])
242 {
243     int hist[256];
244     for(int i = 0; i < 256; i++)
245     {
246         hist[i] = histogram[i];
247     }
248 }
249
250 //calculando los histogramas
251 int hist_x = 0; int hist_y = 0;
252 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
253
254 //calculando la suma de intensidades del histograma
255 int sum = 0;
256 for(int i = 0; i < 256; i++)
257 {
258     sum = hist[i];
259 }
260
261 //normalizando el histograma entre 0 y 255
262 for(int i = 0; i < 256; i++)
263 {
264     hist[i] = (hist[i] * 255) / sum;
265 }
266
267 //calculando el histograma normalizado
268 void normHist(Mat image, int histogram[])
269 {
270     for(int i = 0; i < 256; i++)
271     {
272         histogram[i] = hist[i];
273     }
274 }
275
276 //funcion que muestra el histograma normalizado
277 void showHist(Mat image, int histogram[])
278 {
279     int hist[256];
280     for(int i = 0; i < 256; i++)
281     {
282         hist[i] = histogram[i];
283     }
284 }
285
286 //calculando los histogramas
287 int hist_x = 0; int hist_y = 0;
288 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
289
290 //calculando la suma de intensidades del histograma
291 int sum = 0;
292 for(int i = 0; i < 256; i++)
293 {
294     sum = hist[i];
295 }
296
297 //normalizando el histograma entre 0 y 255
298 for(int i = 0; i < 256; i++)
299 {
300     hist[i] = (hist[i] * 255) / sum;
301 }
302
303 //calculando el histograma normalizado
304 void normHist(Mat image, int histogram[])
305 {
306     for(int i = 0; i < 256; i++)
307     {
308         histogram[i] = hist[i];
309     }
310 }
311
312 //funcion que muestra el histograma normalizado
313 void showHist(Mat image, int histogram[])
314 {
315     int hist[256];
316     for(int i = 0; i < 256; i++)
317     {
318         hist[i] = histogram[i];
319     }
320 }
321
322 //calculando los histogramas
323 int hist_x = 0; int hist_y = 0;
324 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
325
326 //calculando la suma de intensidades del histograma
327 int sum = 0;
328 for(int i = 0; i < 256; i++)
329 {
330     sum = hist[i];
331 }
332
333 //normalizando el histograma entre 0 y 255
334 for(int i = 0; i < 256; i++)
335 {
336     hist[i] = (hist[i] * 255) / sum;
337 }
338
339 //calculando el histograma normalizado
340 void normHist(Mat image, int histogram[])
341 {
342     for(int i = 0; i < 256; i++)
343     {
344         histogram[i] = hist[i];
345     }
346 }
347
348 //funcion que muestra el histograma normalizado
349 void showHist(Mat image, int histogram[])
350 {
351     int hist[256];
352     for(int i = 0; i < 256; i++)
353     {
354         hist[i] = histogram[i];
355     }
356 }
357
358 //calculando los histogramas
359 int hist_x = 0; int hist_y = 0;
360 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
361
362 //calculando la suma de intensidades del histograma
363 int sum = 0;
364 for(int i = 0; i < 256; i++)
365 {
366     sum = hist[i];
367 }
368
369 //normalizando el histograma entre 0 y 255
370 for(int i = 0; i < 256; i++)
371 {
372     hist[i] = (hist[i] * 255) / sum;
373 }
374
375 //calculando el histograma normalizado
376 void normHist(Mat image, int histogram[])
377 {
378     for(int i = 0; i < 256; i++)
379     {
380         histogram[i] = hist[i];
381     }
382 }
383
384 //funcion que muestra el histograma normalizado
385 void showHist(Mat image, int histogram[])
386 {
387     int hist[256];
388     for(int i = 0; i < 256; i++)
389     {
390         hist[i] = histogram[i];
391     }
392 }
393
394 //calculando los histogramas
395 int hist_x = 0; int hist_y = 0;
396 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
397
398 //calculando la suma de intensidades del histograma
399 int sum = 0;
400 for(int i = 0; i < 256; i++)
401 {
402     sum = hist[i];
403 }
404
405 //normalizando el histograma entre 0 y 255
406 for(int i = 0; i < 256; i++)
407 {
408     hist[i] = (hist[i] * 255) / sum;
409 }
410
411 //calculando el histograma normalizado
412 void normHist(Mat image, int histogram[])
413 {
414     for(int i = 0; i < 256; i++)
415     {
416         histogram[i] = hist[i];
417     }
418 }
419
420 //funcion que muestra el histograma normalizado
421 void showHist(Mat image, int histogram[])
422 {
423     int hist[256];
424     for(int i = 0; i < 256; i++)
425     {
426         hist[i] = histogram[i];
427     }
428 }
429
430 //calculando los histogramas
431 int hist_x = 0; int hist_y = 0;
432 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
433
434 //calculando la suma de intensidades del histograma
435 int sum = 0;
436 for(int i = 0; i < 256; i++)
437 {
438     sum = hist[i];
439 }
440
441 //normalizando el histograma entre 0 y 255
442 for(int i = 0; i < 256; i++)
443 {
444     hist[i] = (hist[i] * 255) / sum;
445 }
446
447 //calculando el histograma normalizado
448 void normHist(Mat image, int histogram[])
449 {
450     for(int i = 0; i < 256; i++)
451     {
452         histogram[i] = hist[i];
453     }
454 }
455
456 //funcion que muestra el histograma normalizado
457 void showHist(Mat image, int histogram[])
458 {
459     int hist[256];
460     for(int i = 0; i < 256; i++)
461     {
462         hist[i] = histogram[i];
463     }
464 }
465
466 //calculando los histogramas
467 int hist_x = 0; int hist_y = 0;
468 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
469
470 //calculando la suma de intensidades del histograma
471 int sum = 0;
472 for(int i = 0; i < 256; i++)
473 {
474     sum = hist[i];
475 }
476
477 //normalizando el histograma entre 0 y 255
478 for(int i = 0; i < 256; i++)
479 {
480     hist[i] = (hist[i] * 255) / sum;
481 }
482
483 //calculando el histograma normalizado
484 void normHist(Mat image, int histogram[])
485 {
486     for(int i = 0; i < 256; i++)
487     {
488         histogram[i] = hist[i];
489     }
490 }
491
492 //funcion que muestra el histograma normalizado
493 void showHist(Mat image, int histogram[])
494 {
495     int hist[256];
496     for(int i = 0; i < 256; i++)
497     {
498         hist[i] = histogram[i];
499     }
500 }
501
502 //calculando los histogramas
503 int hist_x = 0; int hist_y = 0;
504 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
505
506 //calculando la suma de intensidades del histograma
507 int sum = 0;
508 for(int i = 0; i < 256; i++)
509 {
510     sum = hist[i];
511 }
512
513 //normalizando el histograma entre 0 y 255
514 for(int i = 0; i < 256; i++)
515 {
516     hist[i] = (hist[i] * 255) / sum;
517 }
518
519 //calculando el histograma normalizado
520 void normHist(Mat image, int histogram[])
521 {
522     for(int i = 0; i < 256; i++)
523     {
524         histogram[i] = hist[i];
525     }
526 }
527
528 //funcion que muestra el histograma normalizado
529 void showHist(Mat image, int histogram[])
530 {
531     int hist[256];
532     for(int i = 0; i < 256; i++)
533     {
534         hist[i] = histogram[i];
535     }
536 }
537
538 //calculando los histogramas
539 int hist_x = 0; int hist_y = 0;
540 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
541
542 //calculando la suma de intensidades del histograma
543 int sum = 0;
544 for(int i = 0; i < 256; i++)
545 {
546     sum = hist[i];
547 }
548
549 //normalizando el histograma entre 0 y 255
550 for(int i = 0; i < 256; i++)
551 {
552     hist[i] = (hist[i] * 255) / sum;
553 }
554
555 //calculando el histograma normalizado
556 void normHist(Mat image, int histogram[])
557 {
558     for(int i = 0; i < 256; i++)
559     {
560         histogram[i] = hist[i];
561     }
562 }
563
564 //funcion que muestra el histograma normalizado
565 void showHist(Mat image, int histogram[])
566 {
567     int hist[256];
568     for(int i = 0; i < 256; i++)
569     {
570         hist[i] = histogram[i];
571     }
572 }
573
574 //calculando los histogramas
575 int hist_x = 0; int hist_y = 0;
576 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
577
578 //calculando la suma de intensidades del histograma
579 int sum = 0;
580 for(int i = 0; i < 256; i++)
581 {
582     sum = hist[i];
583 }
584
585 //normalizando el histograma entre 0 y 255
586 for(int i = 0; i < 256; i++)
587 {
588     hist[i] = (hist[i] * 255) / sum;
589 }
590
591 //calculando el histograma normalizado
592 void normHist(Mat image, int histogram[])
593 {
594     for(int i = 0; i < 256; i++)
595     {
596         histogram[i] = hist[i];
597     }
598 }
599
600 //funcion que muestra el histograma normalizado
601 void showHist(Mat image, int histogram[])
602 {
603     int hist[256];
604     for(int i = 0; i < 256; i++)
605     {
606         hist[i] = histogram[i];
607     }
608 }
609
610 //calculando los histogramas
611 int hist_x = 0; int hist_y = 0;
612 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
613
614 //calculando la suma de intensidades del histograma
615 int sum = 0;
616 for(int i = 0; i < 256; i++)
617 {
618     sum = hist[i];
619 }
620
621 //normalizando el histograma entre 0 y 255
622 for(int i = 0; i < 256; i++)
623 {
624     hist[i] = (hist[i] * 255) / sum;
625 }
626
627 //calculando el histograma normalizado
628 void normHist(Mat image, int histogram[])
629 {
630     for(int i = 0; i < 256; i++)
631     {
632         histogram[i] = hist[i];
633     }
634 }
635
636 //funcion que muestra el histograma normalizado
637 void showHist(Mat image, int histogram[])
638 {
639     int hist[256];
640     for(int i = 0; i < 256; i++)
641     {
642         hist[i] = histogram[i];
643     }
644 }
645
646 //calculando los histogramas
647 int hist_x = 0; int hist_y = 0;
648 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
649
650 //calculando la suma de intensidades del histograma
651 int sum = 0;
652 for(int i = 0; i < 256; i++)
653 {
654     sum = hist[i];
655 }
656
657 //normalizando el histograma entre 0 y 255
658 for(int i = 0; i < 256; i++)
659 {
660     hist[i] = (hist[i] * 255) / sum;
661 }
662
663 //calculando el histograma normalizado
664 void normHist(Mat image, int histogram[])
665 {
666     for(int i = 0; i < 256; i++)
667     {
668         histogram[i] = hist[i];
669     }
670 }
671
672 //funcion que muestra el histograma normalizado
673 void showHist(Mat image, int histogram[])
674 {
675     int hist[256];
676     for(int i = 0; i < 256; i++)
677     {
678         hist[i] = histogram[i];
679     }
680 }
681
682 //calculando los histogramas
683 int hist_x = 0; int hist_y = 0;
684 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
685
686 //calculando la suma de intensidades del histograma
687 int sum = 0;
688 for(int i = 0; i < 256; i++)
689 {
690     sum = hist[i];
691 }
692
693 //normalizando el histograma entre 0 y 255
694 for(int i = 0; i < 256; i++)
695 {
696     hist[i] = (hist[i] * 255) / sum;
697 }
698
699 //calculando el histograma normalizado
700 void normHist(Mat image, int histogram[])
701 {
702     for(int i = 0; i < 256; i++)
703     {
704         histogram[i] = hist[i];
705     }
706 }
707
708 //funcion que muestra el histograma normalizado
709 void showHist(Mat image, int histogram[])
710 {
711     int hist[256];
712     for(int i = 0; i < 256; i++)
713     {
714         hist[i] = histogram[i];
715     }
716 }
717
718 //calculando los histogramas
719 int hist_x = 0; int hist_y = 0;
720 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
721
722 //calculando la suma de intensidades del histograma
723 int sum = 0;
724 for(int i = 0; i < 256; i++)
725 {
726     sum = hist[i];
727 }
728
729 //normalizando el histograma entre 0 y 255
730 for(int i = 0; i < 256; i++)
731 {
732     hist[i] = (hist[i] * 255) / sum;
733 }
734
735 //calculando el histograma normalizado
736 void normHist(Mat image, int histogram[])
737 {
738     for(int i = 0; i < 256; i++)
739     {
740         histogram[i] = hist[i];
741     }
742 }
743
744 //funcion que muestra el histograma normalizado
745 void showHist(Mat image, int histogram[])
746 {
747     int hist[256];
748     for(int i = 0; i < 256; i++)
749     {
750         hist[i] = histogram[i];
751     }
752 }
753
754 //calculando los histogramas
755 int hist_x = 0; int hist_y = 0;
756 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
757
758 //calculando la suma de intensidades del histograma
759 int sum = 0;
760 for(int i = 0; i < 256; i++)
761 {
762     sum = hist[i];
763 }
764
765 //normalizando el histograma entre 0 y 255
766 for(int i = 0; i < 256; i++)
767 {
768     hist[i] = (hist[i] * 255) / sum;
769 }
770
771 //calculando el histograma normalizado
772 void normHist(Mat image, int histogram[])
773 {
774     for(int i = 0; i < 256; i++)
775     {
776         histogram[i] = hist[i];
777     }
778 }
779
780 //funcion que muestra el histograma normalizado
781 void showHist(Mat image, int histogram[])
782 {
783     int hist[256];
784     for(int i = 0; i < 256; i++)
785     {
786         hist[i] = histogram[i];
787     }
788 }
789
790 //calculando los histogramas
791 int hist_x = 0; int hist_y = 0;
792 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
793
794 //calculando la suma de intensidades del histograma
795 int sum = 0;
796 for(int i = 0; i < 256; i++)
797 {
798     sum = hist[i];
799 }
800
801 //normalizando el histograma entre 0 y 255
802 for(int i = 0; i < 256; i++)
803 {
804     hist[i] = (hist[i] * 255) / sum;
805 }
806
807 //calculando el histograma normalizado
808 void normHist(Mat image, int histogram[])
809 {
810     for(int i = 0; i < 256; i++)
811     {
812         histogram[i] = hist[i];
813     }
814 }
815
816 //funcion que muestra el histograma normalizado
817 void showHist(Mat image, int histogram[])
818 {
819     int hist[256];
820     for(int i = 0; i < 256; i++)
821     {
822         hist[i] = histogram[i];
823     }
824 }
825
826 //calculando los histogramas
827 int hist_x = 0; int hist_y = 0;
828 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
829
830 //calculando la suma de intensidades del histograma
831 int sum = 0;
832 for(int i = 0; i < 256; i++)
833 {
834     sum = hist[i];
835 }
836
837 //normalizando el histograma entre 0 y 255
838 for(int i = 0; i < 256; i++)
839 {
840     hist[i] = (hist[i] * 255) / sum;
841 }
842
843 //calculando el histograma normalizado
844 void normHist(Mat image, int histogram[])
845 {
846     for(int i = 0; i < 256; i++)
847     {
848         histogram[i] = hist[i];
849     }
850 }
851
852 //funcion que muestra el histograma normalizado
853 void showHist(Mat image, int histogram[])
854 {
855     int hist[256];
856     for(int i = 0; i < 256; i++)
857     {
858         hist[i] = histogram[i];
859     }
860 }
861
862 //calculando los histogramas
863 int hist_x = 0; int hist_y = 0;
864 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
865
866 //calculando la suma de intensidades del histograma
867 int sum = 0;
868 for(int i = 0; i < 256; i++)
869 {
870     sum = hist[i];
871 }
872
873 //normalizando el histograma entre 0 y 255
874 for(int i = 0; i < 256; i++)
875 {
876     hist[i] = (hist[i] * 255) / sum;
877 }
878
879 //calculando el histograma normalizado
880 void normHist(Mat image, int histogram[])
881 {
882     for(int i = 0; i < 256; i++)
883     {
884         histogram[i] = hist[i];
885     }
886 }
887
888 //funcion que muestra el histograma normalizado
889 void showHist(Mat image, int histogram[])
890 {
891     int hist[256];
892     for(int i = 0; i < 256; i++)
893     {
894         hist[i] = histogram[i];
895     }
896 }
897
898 //calculando los histogramas
899 int hist_x = 0; int hist_y = 0;
900 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
901
902 //calculando la suma de intensidades del histograma
903 int sum = 0;
904 for(int i = 0; i < 256; i++)
905 {
906     sum = hist[i];
907 }
908
909 //normalizando el histograma entre 0 y 255
910 for(int i = 0; i < 256; i++)
911 {
912     hist[i] = (hist[i] * 255) / sum;
913 }
914
915 //calculando el histograma normalizado
916 void normHist(Mat image, int histogram[])
917 {
918     for(int i = 0; i < 256; i++)
919     {
920         histogram[i] = hist[i];
921     }
922 }
923
924 //funcion que muestra el histograma normalizado
925 void showHist(Mat image, int histogram[])
926 {
927     int hist[256];
928     for(int i = 0; i < 256; i++)
929     {
930         hist[i] = histogram[i];
931     }
932 }
933
934 //calculando los histogramas
935 int hist_x = 0; int hist_y = 0;
936 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
937
938 //calculando la suma de intensidades del histograma
939 int sum = 0;
940 for(int i = 0; i < 256; i++)
941 {
942     sum = hist[i];
943 }
944
945 //normalizando el histograma entre 0 y 255
946 for(int i = 0; i < 256; i++)
947 {
948     hist[i] = (hist[i] * 255) / sum;
949 }
950
951 //calculando el histograma normalizado
952 void normHist(Mat image, int histogram[])
953 {
954     for(int i = 0; i < 256; i++)
955     {
956         histogram[i] = hist[i];
957     }
958 }
959
960 //funcion que muestra el histograma normalizado
961 void showHist(Mat image, int histogram[])
962 {
963     int hist[256];
964     for(int i = 0; i < 256; i++)
965     {
966         hist[i] = histogram[i];
967     }
968 }
969
970 //calculando los histogramas
971 int hist_x = 0; int hist_y = 0;
972 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
973
974 //calculando la suma de intensidades del histograma
975 int sum = 0;
976 for(int i = 0; i < 256; i++)
977 {
978     sum = hist[i];
979 }
980
981 //normalizando el histograma entre 0 y 255
982 for(int i = 0; i < 256; i++)
983 {
984     hist[i] = (hist[i] * 255) / sum;
985 }
986
987 //calculando el histograma normalizado
988 void normHist(Mat image, int histogram[])
989 {
990     for(int i = 0; i < 256; i++)
991     {
992         histogram[i] = hist[i];
993     }
994 }
995
996 //funcion que muestra el histograma normalizado
997 void showHist(Mat image, int histogram[])
998 {
999     int hist[256];
1000     for(int i = 0; i < 256; i++)
1001     {
1002         hist[i] = histogram[i];
1003     }
1004 }
1005
1006 //calculando los histogramas
1007 int hist_x = 0; int hist_y = 0;
1008 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
1009
1010 //calculando la suma de intensidades del histograma
1011 int sum = 0;
1012 for(int i = 0; i < 256; i++)
1013 {
1014     sum = hist[i];
1015 }
1016
1017 //normalizando el histograma entre 0 y 255
1018 for(int i = 0; i < 256; i++)
1019 {
1020     hist[i] = (hist[i] * 255) / sum;
1021 }
1022
1023 //calculando el histograma normalizado
1024 void normHist(Mat image, int histogram[])
1025 {
1026     for(int i = 0; i < 256; i++)
1027     {
1028         histogram[i] = hist[i];
1029     }
1030 }
1031
1032 //funcion que muestra el histograma normalizado
1033 void showHist(Mat image, int histogram[])
1034 {
1035     int hist[256];
1036     for(int i = 0; i < 256; i++)
1037     {
1038         hist[i] = histogram[i];
1039     }
1040 }
1041
1042 //calculando los histogramas
1043 int hist_x = 0; int hist_y = 0;
1044 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
1045
1046 //calculando la suma de intensidades del histograma
1047 int sum = 0;
1048 for(int i = 0; i < 256; i++)
1049 {
1050     sum = hist[i];
1051 }
1052
1053 //normalizando el histograma entre 0 y 255
1054 for(int i = 0; i < 256; i++)
1055 {
1056     hist[i] = (hist[i] * 255) / sum;
1057 }
1058
1059 //calculando el histograma normalizado
1060 void normHist(Mat image, int histogram[])
1061 {
1062     for(int i = 0; i < 256; i++)
1063     {
1064         histogram[i] = hist[i];
1065     }
1066 }
1067
1068 //funcion que muestra el histograma normalizado
1069 void showHist(Mat image, int histogram[])
1070 {
1071     int hist[256];
1072     for(int i = 0; i < 256; i++)
1073     {
1074         hist[i] = histogram[i];
1075     }
1076 }
1077
1078 //calculando los histogramas
1079 int hist_x = 0; int hist_y = 0;
1080 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
1081
1082 //calculando la suma de intensidades del histograma
1083 int sum = 0;
1084 for(int i = 0; i < 256; i++)
1085 {
1086     sum = hist[i];
1087 }
1088
1089 //normalizando el histograma entre 0 y 255
1090 for(int i = 0; i < 256; i++)
1091 {
1092     hist[i] = (hist[i] * 255) / sum;
1093 }
1094
1095 //calculando el histograma normalizado
1096 void normHist(Mat image, int histogram[])
1097 {
1098     for(int i = 0; i < 256; i++)
1099     {
1100         histogram[i] = hist[i];
1101     }
1102 }
1103
1104 //funcion que muestra el histograma normalizado
1105 void showHist(Mat image, int histogram[])
1106 {
1107     int hist[256];
1108     for(int i = 0; i < 256; i++)
1109     {
1110         hist[i] = histogram[i];
1111     }
1112 }
1113
1114 //calculando los histogramas
1115 int hist_x = 0; int hist_y = 0;
1116 int hist_x = cvt_2u(hist_x); int hist_y = cvt_2u(hist_y);
1117
1118 //calculando la suma de intensidades del histograma
1119 int sum
```

- Reprobación automática a quien copie códigos de Internet y los reporte como suyos, además de una nota en su expediente con copia para el consejo de calidad

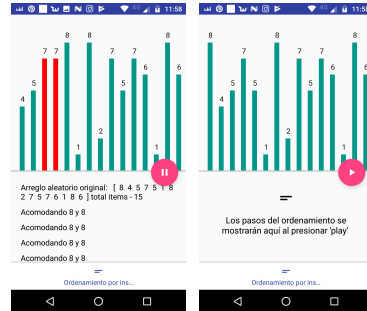


<https://github.com/naman14/AlgorithmVisualizer-Android>



Proyecto Original

Dr. Marco Aurelio Nuño Maganda



Proyecto "clonado"

Fundamentos de Sistemas de Información

“Finalmente son jóvenes que están en la preparatoria y que deben de leer su convocatoria con toda claridad, si no cumplen con los requisitos, si no pueden leer una convocatoria que dice tienes que traer número uno esto, número dos esto, número tres esto, no están listos para ser **estudiantes de educación superior**, así lo digo con toda claridad”.

Sara Ladrón de Guevara.

Rectora de la Universidad Veracruzana (2013-2017 y 2017-2021).

ODIAME AHORA
POR EXIGIRTE
COMO ESTUDIANTE
PARA QUE NO ME TENGAS QUE
ODIAR DESPUÉS
POR HACERTE
UN PROFESIONAL MEDIOCRE