Instituto Tecnológico de Costa Rica

Área Académica de Ingeniería en Computadores
Programa de Licenciatura en Ingeniería en Computadores

CE1102-Taller de introducción a la programación

Tarea Programada 1: Filtro de medianas

Profesor: Saúl Calderón Ramírez

Estudiantes:
Federico Alfaro Chaverri 2022051002
Valentin Tissera Doncini 2022145010

Primer semestre 2022

Indice

1 Introducción

El siguiente proyecto se desarrolló en el mes de mayo, su objetivo era poder realizar un filtro el cual al ser aplicado en una imagen este haría que ciertos pixeles los cuales no corresponden en la imagen fueran eliminados y sustituidos por aquellos que rodeaban el área.El objetivo del código era demostrar cómo con la recursividad de cola y pila este filtro se podía crear aunque existiese la limitación del lenguaje Python al llenarse su pila de recursividad. La implementacion de este filtro se basa en el principio de la mediana, lo que se realizó fue un programa el cual al insertar una matriz de números (en este caso la matriz de números con la que se va a trabajar). El proyecto se dividió en tres partes:

i) La apertura de la imagen a trabajar y de la nueva imagen después de que se le aplique el filtro.

ii) El paso de la imagen principal a matriz definida el valor por la ventana la cual el usuario ingreso

iii) El ordenamiento de las matrices y sustitución de los valores incorrectos.

Estas últimas dos partes fueron creadas en clases para que el proceso se hiciera más sencillo y fácil de entender.

2 Análisis del problema

- Entradas: Ninguna. Durante el progreso del programa se debe solicitar mediante la función **input()** que se introduzca la dirección de una foto con el problema de sal y pimienta, también se debe preguntar al usuario el tamaño de la ventana deseada y cuál de los dos algoritmos de ordenamiento prefiere usar.

- Salidas: Una matriz con la mediana de los valores que se encuentran dentro de una ventana de cierta posición. Además, se espera que el programa produzca errores si no se introduce una dirección de imagen válida. Otros posibles errores se deberán a elegir un tamaño de ventana impar o elegir un número de algóritmo que no sea uno o dos. Se debe tomar en cuenta que si la imagen es muy grande también se puede producir un error por llenar la pila de llamados recursivos, sin embargo, con las imágenes esperadas en este proyecto no se produce tal error.

- Restricciones:

  - Al trabajar con recursividad, no se pueden trabajar con matrices muy grandes ya que se puede llenar el límite de llamador recursivos.
  - Se debe agregar manualmente la dirección de la imagen a la que se le aplicará el filtro de medianas, por lo que se debe estar seguro de insertar la dirección correcta cada vez.

- Subproblemas:

  - Solicitar la dirección de una imagen a la cual se desee realizar el filtro de medianas y verificar que se encuentre una imagen
  - Transformar la imagen en una matriz.
  - Solicitar el tamaño de la ventana con la que se quiera trabajar.
  - Recorrer las filas de la matriz.
  - Recorrer cada elemento de la fila y tomar los valores en la ventana y transformarlos en un vector.
  - Verificar que las ventanas no se salgan de la matriz, en caso de que se salgan, solo tomar los elementos de la matriz que se encuentren dentro de la ventana para calcular su mediana.
  - Ordenar los elementos de la lista en orden creciente y tomar el valor central de dicha lista ordenada.
  - Crear una lista con los elementos medios de los valores de la ventana de cada fila.
  - Agregar cada una de las listas anteriores para hacer una matriz de los valores medios dentro de cada ventana.
  - Devolver la matriz generada y transformarla nuevamente en una imagen.

3 Diseño de la solución

Para poder solicitar la dirección de la imagen a la cual se le desea realizar el filtro de medianas se crea una clase llamada Apertura() que contiene las siguientes funciones.

1) **probar_apertura_archivo()** la cual e encarga de preguntar la dirección de la imagen a la que se le desea sacar el filtro de medianas y transforma dicha imagen en una matriz.

   1.1) Pregunta al usuario mediante la función **input()** que introduzca la dirección de la imagen a filtrar.

   1.2) Usa la librería PIL para abrir una imagen y transformarla en una matriz

2) **abrir_nuevo_archivo(matriz)** Se encarga que usar funciones de la librería PIL para transformar una matriz en una imagen y abrirla en pantalla.

Para resolver los problemas sobre recorrer una matriz, seleccionar la ventana alrededor de cada posición de la matriz y construir el resultado final en una matriz se usa la clase Calculo_Media.

1) Las instancias se inicializan recibiendo una matriz, para generar una matriz donde se guarda el resultado de las demás funciones

2) **calcular_matriz_de_medias(matriz, contador_fila, ventana, selector)** recorre cada fila de la matriz original llamando a la función **calcular_media_ventana** y retorna el resultado final.

3) **calcular_media_ventana(matriz, fila, contador_columna,ventana, selector)** dada una lista que representa una fila de la matriz original y el número de dicha fila se recorre cada elemento y se calcula la mediana de los elementos dentro de la ventana.

   3.1) Se calcula la variable *radio* que representa la cantidad máxima de posiciones de distancia hacia arriba, abajo, izquierda o derecha que puede estar un elemento dentro de la ventana.

   3.2) Se ponen las condiciones para verificar si en cada posición específica de la matriz la ventana se sale lateralmente por uno o más lados. Dependiendo de si la matriz se sale se recorta el tamaño de la ventana para que solo se tomen los elementos de la matriz.

   3.3) Se toman todos los elementos dentro de la matriz como una submatriz mediante el slicing.

   3.4) Mediante la función **.flatten()** de numpy se transforma la ventana en una lista, la cual llama a la función **media(matriz)** de la clase Ordenamiento o Ordenamiento_2 encontrar la mediana de dicha matriz

3.5) Se sustituyen los valores de *matriz_de_medias* por cada una de las medianas encontradas.

4) calcular_media_arriba(matriz, fila, contador_columna, radio, selector) es una función auxiliar de **calcular_media_ventana** que reutiliza parte del código pero cambiando el tamaño de la ventana contemplando que la ventana se sale por arriba de la matriz.

5) calcular_media_abajo(matriz, fila, contador_columna, radio, selector) es una función auxiliar de **calcular_media_ventana** que reutiliza parte del código pero cambiando el tamaño de la ventana contemplando que la ventana se sale por abajo de la matriz.

6) calcular_media_abajo_arriba(matriz, fila, contador_columna, radio, selector) es una función auxiliar de **calcular_media_ventana** que reutiliza parte del código pero cambiando el tamaño de la ventana contemplando que la ventana se sale por arriba y por abajo de la matriz.

Para calcular la mediana de una lista se crean las clases Ordenamiento y Ordenamiento_2 que comparten la función:

1) **media(lista)** que se encarga de llamar a otra función que ordena la lista. Una vez ordenada la lista se toma el valor de la matriz que se encuentra a la mitad y se retorna.

En la clase Ordenamiento se encuentra

1) **quick_sort(lista)** se encarga de tomar el menor elemento de la lista y añadirlo al final de una segunda lista y eliminarlo de la lista inicial y se repite el proceso hasta que no queden elementos en la lista original, de modo que la segunda lista queda con todos los elementos de la lista original ordenados.
NOTA: para facilitar la escritura del código se utilizó este nombre, sin embargo, esta función específica no corresponde al algoritmo de ordenamiento quick_sort.

En la clase Ordenamiento_2 se encuentra

1) **quick_sort(lista)** implementa el algoritmo quick sort que se encarga de dividir una lista en sublistas que contienen los valores mayores y menores al primer elemento de la lista, luego se repite el proceso con cada una de las sublistas hasta que solo queden listas de 1 elemento y durante el proceso suma la sublista de menores del lado izquierdo y la sublista de mayores del lado derecho del primer elemento de cada lista.

2) **mayores(valor, lista, contador, lista_mayores)** toma los elementos mayores al primer elemento de la lista original, los agrega en una nueva lista y llama nuevamente a la función **quick_sort(lista)**.

3) **menores(valor, lista, contador, lista_menores)** toma los elementos menores al primer elemento de la lista original, los agrega en una nueva lista y llama nuevamente a la función **quick_sort(lista)**.

En el archivo "main" se crean variables llamadas *inicio* y *fin* que utilizan la librería time para dar el tiempo actual del sistema y se restan para calcular el tiempo de ejecución del programa. También se solicita al usuario que ingrese el tamaño de la ventana, luego, se calcula el filtro de medias, se muestra en pantalla y se muestra el tiempo de ejecución del sistema.

En la Figura 1 se muestra el diagrama de de las clases donde se ilustra como funciona cada una.

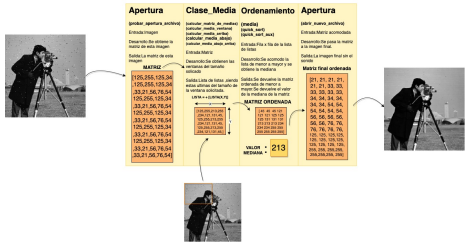

Figure 1: Diagrama de clases

4 Implementación y pruebas

Para poder desarrollar este proyecto se necesito de las librerías “numpy”, “matplotlib.pyplot”, “PIL” y la librería “time”. Las cuales todas son librerías propias del lenguaje python. Con la función nativa de la librería “PIL” se inicializa la imagen con la cual se va a trabajar, ya una vez que la imagen fue inicializada se obtiene la matriz con la función “asarray” de la librería “numpy”.Por ultimo la librería ”time” se usa para poder mostrarle al usuario cuanto tiempo duro,haciendo una resta del momento cuando se empezó a correr el programa y cuando se finalizo de correr el programa multiplicado por 1000 ya que se pidió que se mostrar en consola el tiempo que tardo en ejecutarse el programa en milisegundos y el resultado nos lo da en segundos.

En resumen lo que se hace es pasar la imagen al número de la escala de gris de cada pixel de la imagen a una matriz la cual se va almacena como una lista de listas donde cada valor es la columna n de la lista. Ya una vez se haya terminado esta fase ,lo que sigue es ordenar y crear las ventanas del tamaño solicitado por el usuario y por lo tanto ahora se llama a la clase “Calculo_Media” la cual está almacenada matrices con los elementos de todas las ventanas posibles de la imagen y el resto que no entro en una ventana se almacena en una matriz final. Ya una vez se tiene la lista de listas lo que hace falta es acomodarla y encontrar la mediana de cada una de estas.

Por lo tanto se pasa a la siguiente clase “Ordenamiento” la cual se encarga de ordenar de forma ascendente los números de la matriz para poder obtener el valor del medio, cabe aclarar que si la lista no es impar se tomará el número de la derecha del medio para usar de mediana. Una vez ya nos encontramos en esta sección del programa, solamente nos queda calcular la media y pasar los resultados de nuevo a la segunda clase para que estos hagan el intercambio de los valores no correspondidos.

En la Figura 2 se puede ver la imagen original a la que se le aplicará el filtro de medianas.

Figure 2: manNoisy

En la Figura 3 se muestra una comparación entre la imagen original y la imagen filtrada con una ventana de tamaño tres.

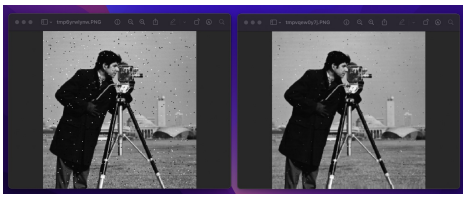

Figure 3: Comparación con ventana 3

En la Figura 4 se aprecia la duración del tiempo de ejecución del filtro con una ventana de tamaño tres.

Figure 4: Tiempo de ejecución con ventana de 3

En la siguiente imagen 5 se muestra la filtrada tres veces con una ventana de tamaño tres. Se puede apreciar que el problema de la sal y pimienta se ha corregido casi por completo, sin embargo, se pierde cierta nitidez en la imagen. Realizar este filtro tuvo una duración de 3000 milisegundos

Figure 5: Ventana de 3 (filtrada 3 veces)

En la Figura 6 se muestra una comparación entre la imagen original y la imagen filtrada con una ventana de tamaño veintiuno.

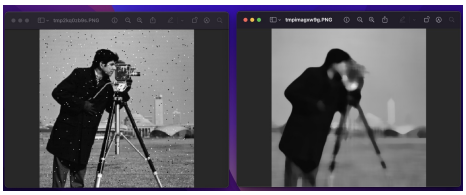

Figure 6: Comparación con ventana de 21

En la Figura 7 se aprecia la duración del tiempo de ejecución del filtro con una ventana de tamaño veintiuno.

Figure 7: Tiempo de ejecución con ventana de 21

En la siguiente imagen 8 se muestra la filtrada tres veces con una ventana de tamaño veintiuno. Se puede apreciar que el problema de la sal y pimienta se ha corregido por completo, sin embargo, la imagen es indistinguible. Aplicar este filtro tuvo una duración de 1800000 milisegundos

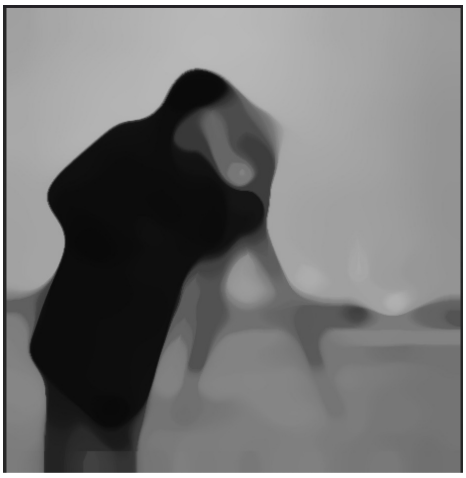

Figure 8: Ventana de 21 (filtrada 3 veces)

4.1 main

Archivo que sirve para correr el programa. Instancia todas las clases para poder obtener la matriz de medias , se encarga de pedirle al usuario todos los requisitos para poder correr el programa(imagen,tamaño de la ventana y algoritmo de ordenamiento).

4.1.1 probar_apertura_de_archivo()

- Descripción: Esta función se encarga de obtener la imagen con la que se va a trabajar, transformar los colores de esta imagen a la escala de grises, mostrar la imagen para que el usuario pueda ver los cambios y por ultimo pasar esta imagen a una matriz de números.
- Salidas: La función retorna la matriz de la imagen después de que se le aplique el filtro de la escala de grises
- Restricciones: El nombre de la imagen que se ingreso tiene que existir en la carpeta o repositorio donde se esta trabajando.La extensión tiene que ser de tipo .PNG o .JPG.

4.1.2 abrir_nuevo_archivo(matriz)

- Descripción: El objetivo de esta función es el de recibir la matriz ya organizada y abrirla en el dispositivo para mostrarle al usuario la nueva imagen.
- Entradas:La matriz ordenada de la imagen

  Parametro 1: matriz la cual se va a abrir
- Salidas: La funcion no tiene salida ya que esta es la ultima funcion que se ejecuta.
- Restricciones: La variable que entra tiene que ser una matriz de la libreria numpy.

4.1.3 calcular_matriz_de_medias(matriz, contador_fila, ventana)

- Descripción: Esta función se encarga de devolver la matriz de media después de ser ordenada y trabajada por ventanas
- Entradas: La matriz de la imagen inicial ,s

  Parametro 1:La matriz inicial de la imagen

  Parametro 2:Contador el cual va a ser usado para la recursividad

  Parametro 3:Tamaño de la ventana con el que se va a trabajar
- Salidas: La matriz después de aplicarle el filtro de medianas
- Restricciones: La variable que entra tiene que ser una matriz de la libreria numpy y el contador tiene que estar inicializado en 0 para que funcione bien

4.2 Ordenamiento

Clase que contienen las funciones que se encargan de ordenar una lista numérica en orden numérico y encontrar la media de dicha lista.

4.2.1 media(self,lista)

- Descripción: Esta función se encarga de obtener el dígito del medio de la matriz insertada en el parámetro.Si la lista no tiene una cantidad de valores impar ,se usara el numero que esta a la derecha del numero de la mitad de la lista.
- Entradas:La única que entrada que existe en esta función seria la matriz a la cual se le quiere obtener la mediana.

  Parámetro 1:El único parámetro con el que cuenta esta función es la lista(matriz) la cual se va a calcular la mediana.
- Salidas: Esta función devuelve una variable entero ,el cual es la mediana.
- Restricciones: La lista ingresada tiene que ser la matriz ya acomodada para que la mediana sea la correcta

4.2.2 quick_sort(self, lista)

- Descripción: Esta función se encarga de instanciar la matriz entrada por el parámetro verificando que esta sea una lista ,también esta función sirve para verificar que la matriz tenga mas de un valor .
- Entradas: La entrada de esta función es la lista la cual se va a ordenar.

  Parámetro 1:El parámetro ingresado es la matriz/lista a ordenar.
- Salidas: La función devuelve el llamado de la función auxiliar de esta etapa.

- Restricciones: El parámetro ingresado tiene que ser de tipo lista o array de la librería numpy.

4.2.3 quick_sort_aux(self, lista, lista_ord)

- Descripción: Esta función obtiene el mínimo valor adentro del primer parámetro para ir almacenándolo en el otro parámetro haciendo que la lista se vaya creando de una forma de recursiva de cola
- Entradas: La entrada de esta función seria la lista con la que se va trabajar y la lista donde se va ir guardando el resultado.

  Parámetro 1:Lista ingresada desde la función principal(quick_sort) para acomodar.

  Parámetro 2:Lista vacía en la cual se va ir ingresando el resultado
- Salidas: La función devuelve la matriz ordenada en el parámetro lista_ord
- Restricciones: La primera vez que se llama a la función se tiene que inicializar con la matriz a ordenar como parámetro 1 y en el segundo parámetro se inicializa vacía

4.3 Calculo_Media

Clase que contiene las funciones que recorren la matriz de la imagen y calculan una nueva matriz la cual contiene el cálculo de la mediana de los elementos de la matriz de la imagen.

4.3.1 calcular_matriz_de_medias(matriz, contador_fila, ventana, selector)

- Descripción: Esta función se encarga de devolver ya todo la imagen después de aplicarle el filtro de medianas por el tamaño de la ventana
- Entradas:La entrada de esta función seria la matriz de la imagen,en conclusión la matriz con la que se va a trabajar

  Parámetro 1: La matriz de la imagen creada en la clase apertura

  Parámetro 2: Contador que se su funcionalidad se ve mas implicada en la recursividad para devolver el resultado

  Parámetro 3: Variable que almacena la elección del usuario para el ordenamiento de la matriz.
- Salidas: La salida de esta función seria la matriz después de trabajarla por el tamaño de la ventana y pasar por el algoritmo de ordenamiento.
- Restricciones: Esta función tiene que ser inicializada con la matriz de la imagen para que funcione bien.Todos los demas parámetros tienen que inicializarse de manera correcta

4.3.2 calcular_media_ventana(matriz, fila, contador_columna, ventana, selector)

- Descripción: Esta funcion se encarga de el proceso de las ventanas que estaria posicionadas en el centro de la imagen.
- Entradas: La imagen y el tamaño de la ventana son las única entrada necesaria para poder correr esta funcion

  Parámetro 1: La matriz con la que se va trabajar

  Parámetro 2: La fila por la cual se va ejecutando la recursividad

  Parámetro 3: Contador que indica por cual columna de la ventana se va ejecutando el proceso

  Parámetro 4: Variable que almacena la elección del usuario para el ordenamiento de la matriz.
- Salidas: Esta función devuelve la matriz de la ventana ya ordenada.
- Restricciones: La matriz tiene que ser de una imagen y la inicialización de los otros paraḿetros tiene que estar de forma correcta.

4.3.3 calcular_media_arriba(matriz, fila, contador_columna, radio, selector)

- Descripción: Esta función se encarga de acomodar la matriz con el quick sort y de devolver esta matriz ya ordenada.
- Entradas: La entrada de esta función seria la matriz de las ventanas de la parte de arriba de la imagen

  Parámetro 1: La matriz con la que se va trabajar

  Parámetro 2: La fila por la cual se va ejecutando la recursividad

  Parámetro 3: Contador que indica por cual columna de la ventana se va ejecutando el proceso

  Parámetro 4: El radio de la ventana con la que se esta trabajando

  Parámetro 5: Variable que almacena la elección del usuario para el ordenamiento de la matriz.
- Salidas: Esta función devuelve la matriz de la ventana ya ordenada.
- Restricciones: La matriz tiene que ser de una imagen y la inicialización de los otros paraḿetros tiene que estar de forma correcta.

4.3.4 calcular_media_abajo(matriz, fila, contador_columna, radio, selector)

- Descripción: Esta función se encarga de ordenar las ventanas creadas de la parte de abajo de la imagen.Devolviendo las ventanas ya ordenas

- Entradas: La entrada de esta función es la matriz con la que se va a trabajar.

  Parámetro 1: La matriz con la que se va trabajar

  Parámetro 2: La fila por la cual se va ejecutando la recursividad

  Parámetro 3: Contador que indica por cual columna de la ventana se va ejecutando el proceso

  Parámetro 4: El radio de la ventana con la que se esta trabajando

  Parámetro 5: Variable que almacena la elección del usuario para el ordenamiento de la matriz.

- Salidas: Esta función devuelve la matriz de la ventana ya ordenada.

- Restricciones: La matriz tiene que ser de una imagen y la inicialización de los otros paraḿetros tiene que estar de forma correcta.

4.4 Apertura

4.4.1 probar_apertura_archivo()

- Descripción: Función que pregunta la dirección de la imagen a la que se le desea sacar el filtro de medianas y transforma dicha imagen en una matriz.

- Salidas: matriz que contiene los valores de la imagen.

- Restricciones: La función le solicita al usuario que digite la dirección de la imagen a filtrar, por lo que si no se ingresa una dirección válida que contenga una imagen el programa no funcionará.

4.4.2 abrir_nuevo_archivo(matriz)

- Descripción: Función que transforma una matriz en una imagen y la muestra.

- Entradas: La entrada de esta función es la matriz con la que se va a trabajar.

  Parámetro 1: Matriz (matriz de medias calculada).

- Salidas: Esta función abre la imagen filtrada.

5 Conclusiones y recomendaciones

Para implementar este problema se recomienda utilizar algoritmos de ordenamiento de listas de lo más eficientes posibles disminuir el tiempo de ejecución del código, al mismo tiempo, se debe tomar en cuenta que dependiendo del tamaño de la imagen y la ventana el tiempo de ejecución puede aumentar sustancialmente. Es importante siempre considerar el tipo de datos ingresados en las variables para poder manipularlos correctamente. En caso de trabajar con listas, como existen las listas de listas, al trabajar con ambas y utilizarlas en diferentes funciones se puede llegar a tener errores al confundir las indexaciones.

En general, el proyecto permitió un acercamiento a la programación con listas y matrices, además, permitió aprender a utilizar la programación orientada a objetos para resolver problemas. Al aplicar el filtro de medianas, permite disminuir el problema de las imágenes con sal y pimienta, adicionalmente, se pudo notar que las ventanas pequeñas dan un mejor resultado. Para mejorar el filtro se recomienda filtrar varias veces la imagen con una ventana pequeña, siempre tomando en cuenta que el filtro puede hacer que la imagen llegue a verse borrosa y por consiguiente se puedan perder detalles importantes.

6 Bibliografía

numpy array: IndexError: too many indices for array. (2017, December 9). Stack Overflow. https://stackoverflow.com/questions/47733704/numpy-array-indexerror-too-many-indices-for-array Gallagher, J. (2021, January 4). Python Flatten List: A How-To Guide. Career Karma. https://careerkarma.com/blog/python-flatten-list/

GeeksforGeeks. (2021, May 22). Python - Call function from another file. https://www.geeksforgeeks.org/python-call-function-from-another-file/ using variables in class functions in another class (python). (2012, June 1). Stack Overflow. https://stackoverflow.com/questions/10842996/using-variables-in-class-functions-in-another-class-python

ImportError: cannot import name ". . ." from partially initialized module ". . ." (most likely due to a circular import). (2020, November 12). Stack Overflow. https://stackoverflow.com/questions/64807163/importerror-cannot-import-name-from-partially-initialized-module-m

Convert an array into image. (2015, November 11). Stack Overflow. https://stackoverflow.com/questions/33658709/convert-an-array-into-image