Práctica 2: Divide y Vecenrás

Algoritmos básicos
Busca Forma

Dpto. Ciencias de la Computación e Inteligencia Artificial
E.T.S. de Ingenierías Informática y de Telecomunicación
Universidad de Granada

DECSAI

Algorítmica

Grado en Ingeniería Informática D

Índice de contenido

1. Introducción

Los objetivos de este guión de prácticas son los siguientes:

1. Entender la técnica de diseño de algoritmos Divide y Vencerás DyV . Hacer uso de esta técnica para resolver un problema
2. Implementar algoritmos básicos usando esta técnica, y otros algoritmos más complejos.
3. Seguir practicando con la obtención de la eficiencia de algoritmos.

2. Algoritmos básicos.

2.1. Máximo y Mínimo en un Vector

Dado un conjunto de enteros, dispuestos en un vector, diseñar un algoritmo DyV para encontrar el máximo y el mínimo. Para ello implementar un programa que genere un vector con un determinado número de enteros y obtener el máximo y mínimo de este conjunto. Para ello debéis implementar la función:

pair<int ,int> Max_Min(int * v,int n);

1. Obtener la eficiencia teórica y empírica del algoritmo.
2. Dar un nivel entre [0,10] del nivel de dificultad de como has percibido este ejercicio. Para ello **0** es igual a no tiene ningún nivel de dificultad y **10** tiene mucha dificultad

2.2. Máximo y Mínimo en una Matriz

Dado un conjunto de enteros dispuestos en una matriz cuadrada diseñar un algoritmo DyV para encontrar el máximo y el mínimo. Para ello implementar un programa que genere una Matriz con un determinado número de enteros y obtener el máximo y mínimo de este conjunto. Para ello debéis implementar la función:

pair<int ,int> Max_Min(int ** M,int inicioi,int inicioj,int n);

siendo ***M*** la matriz cuadrada, ***inicioi*** la fila de inicio (en la primera llamada 0), ***inicioj*** la columna de inicio (en la primera llamada 0) y ***n*** el número de filas o columnas.

1. Obtener la eficiencia teórica y empírica del algoritmo.
2. Dar un nivel entre [0,10] del nivel de dificultad de como has percibido este ejercicio. Para ello **0** es igual a no tiene ningún nivel de dificultad y **10** tiene mucha dificultad

2.3. Zapatos con sus pies.

Se dispone de un conjunto de n pares de zapatos y un conjunto de niños/as de una guardería a los que le queremos poner los zapatos de su talla. La maestra de la guardería sigue el siguiente algoritmo para calzar a un niño/niña:

*Dado el niño/a y un par de zapatos la maestra es capaz de determinar si los zapatos son menores o mayores que el pie del niño, o encaja exactamente con el pie. Sin embargo, no hay forma de comparar los pies de dos niños o dos pares de zapatos entre sí. Se desea ordenar los dos conjuntos de forma que los elementos que ocupan la misma posición en los dos conjuntos se emparejen entre sí.

IDEA: Se procederá mediante un algoritmo Divide y Vencerás de la siguiente forma. Representaremos los conjuntos de zapatos y los pies de los niños mediante dos vectores de tamaño n que contienen los zapatos (tamaños) y los tamaños de los pies de los niños. El algoritmo comienza escogiendo un zapato aleatoriamente, este será el pivote para dividir los pies de los niños en tres conjuntos: **S1** será el conjunto de pies que tienen un tamaño menor al

zapato ; **S2**: el conjunto de pies que tienen igual tamaño al zapato; y **S3** el conjunto de pies que tienen tamaño mayor al zapato. A continuación usando el pie del niño que encaja perfectamente con el zapato elegido, realizamos una partición similar en el conjunto de zapatos dividiéndolo en tres conjuntos (aquellos zapatos menores que el pie, zapatos iguales al pie y zapatos mayores que el pie). El algoritmo, de forma recursiva, aplica el mismo procedimiento a cada uno de los dos grupos que se han formado, es decir, el de los zapatos y pies menores que el par encontrado, y el de los mayores.

1. Obtener la eficiencia teórica y empírica del algoritmo.
2. Dar un nivel entre [0,10] del nivel de dificultad de como has percibido este ejercicio. Para ello **0** es igual a no tiene ningún nivel de dificultad y **10** tiene mucha dificultad

3. Como seguir con la Práctica.

Si al hacer el ejercicio 1-2 tu percepción de dificultad es mayor a 6 debes hacer el ejercicio 5 en otro caso debes hacer el ejercicio 4 y 5.

4. Moda de un conjunto de enteros

Dado un conjunto de enteros diseñar un algoritmo DyV para encontrar el elemento moda del conjunto. El elemento moda es aquel que aparece con más frecuencia.

1. Obtener la eficiencia teórica y empírica del algoritmo.
2. Dar un nivel entre [0,10] del nivel de dificultad de como has percibido este ejercicio. Para ello **0** es igual a no tiene ningún nivel de dificultad y **10** tiene mucha dificultad

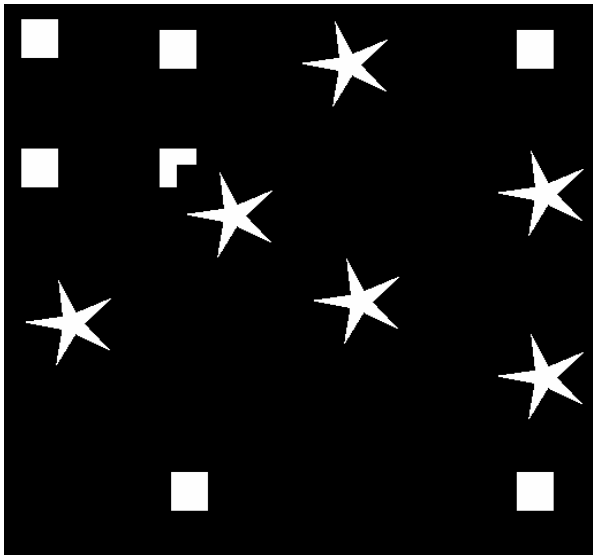
IDEA.- Definido un elemento pivote, dividir el conjunto en tres conjuntos:

1. Conjuntos de elementos menores y diferentes al pivote. Este conjunto se almacena en Heteregoneos.
2. Conjuntos de elementos igual al pivote. Este conjunto se almacena en Homogeneos
3. Conjunto de elementos mayores y diferentes al pivote. Este conjunto se almacena en Heteregoneos.

Con estas definiciones los pasos a seguir son:

1. Sea **He** el conjunto en heterogeneos de mayor cardinalidad (el conjunto más grande).
2. Sea **Ho** el conjunto de homogeneos de mayor cardinalidad (el conjunto más grande)
3. Si Ho tiene más elementos que He entonces hemos terminado y la moda es el elemento que aparece en Ho.
4. En otro caso volver a dividir según 1-3 He.

5. Contando Formas.

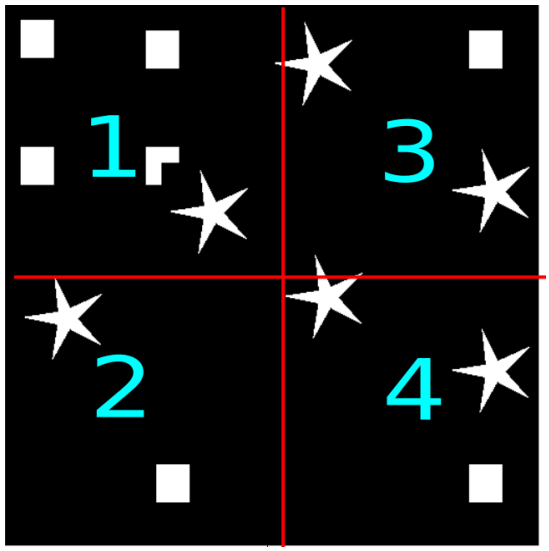


5.1. Descripción del Problema

Dada una imagen cuadrada (no es más que una matriz) con formas básicas como la que se muestra en la imagen de la derecha queremos contar las formas básicas que tienen. Así en este ejemplo la formas básicas son cuadrados y estrellas. Contabilizando que tenemos 6 cuadrados y 6 estrellas.

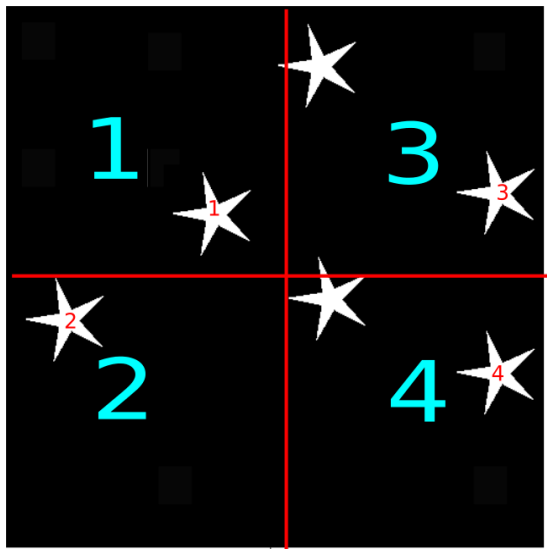
Una imagen binaria, no es más que una matriz rellena de 0 (negro) o 255 (blanco, si existe en esa posición una parte de una forma).

El objetivo es que el alumno/a construya un algoritmo basado en la técnica DyV para contabilizar cuantas ocurrencias hay de una forma.



IDEA.- Para llevar a cabo este algoritmo, supongamos que dividimos en cuatro subcuadrantes una imagen como se muestra en la derecha. Para resolver por ejemplo el problema de contabilizar cuantas estrellas hay en mi imagen, contabilizo cuantas estrellas hay en el subcuadrante 1, cuantas hay en el subcuadrante 2, cuantas hay en el subcuadrante 3 y cuantas en el subcuadrante 4. Esta partición se repite en cada subcuadrante de forma recursiva hasta llegar a una dimensión de subcuadrantes que es menor que la dimensiones de la forma que estoy buscando, en ese caso el número de formas que devuelvo es 0.

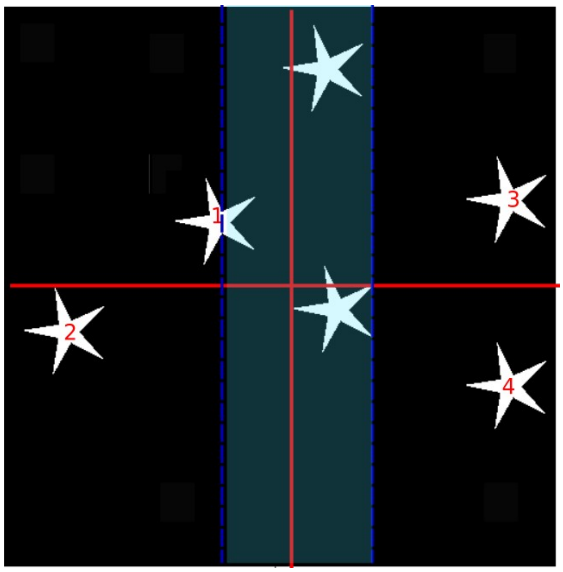
En la imagen de la derecha tengo una primera división de mi problema en cuatro subproblemas, a su vez cada subcuadrante se divide en otros subcuadrantes..

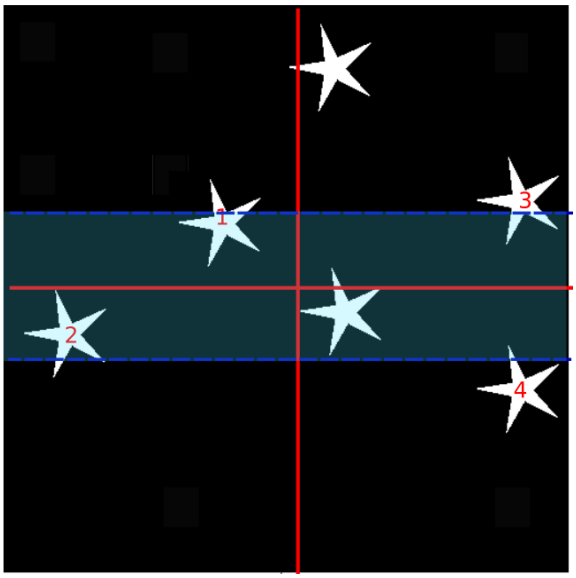


En este ejemplo el primer subcuadrante me ha devuelto una estrella, el segundo otra, el tercero otra y el cuarto otra

.

Existe dos estrellas que no se han contabilizado en los límites del subcuadrante 3 y el subcuadrante 4. Por lo tanto cuando volvamos de las cuatro llamadas recursivas debemos ver si alrededor de los limites de los cuadrantes había alguna estrella. De forma que analizo las siguientes franjas:





5.2. Programa Formas

Este programa contabiliza sobre una imagen binaria, cuantos formas existen de una forma básica dada otra imagen binaria. Un ejemplo de llamada seria la siguiente:

```
prompt% formas datos/estrellas_cuadrados.pgm datos/estrella.png
```

Los parámetros de entrada son los siguientes:

1. La imagen con el conjunto de formas
2. La imagen con la forma a buscar.

6. Eficiencia

El alumno aportará un análisis de la eficiencia de cada uno de los programas. Para ello construirá el documento eficiencia.pdf donde se explique el cálculo de la eficiencia de cada ejercicio. Cada sección debe aportar el análisis de la eficiciencia realizado para cada uno de los problemas. Se hará un estudio de la eficiencia teórica, centrándonos en la función que implementa el algoritmo DyV, obteniendo la formulación del tiempo de ejecución Esta función se resolverá usando las técnicas aprendidas en el tema de eficiencia.

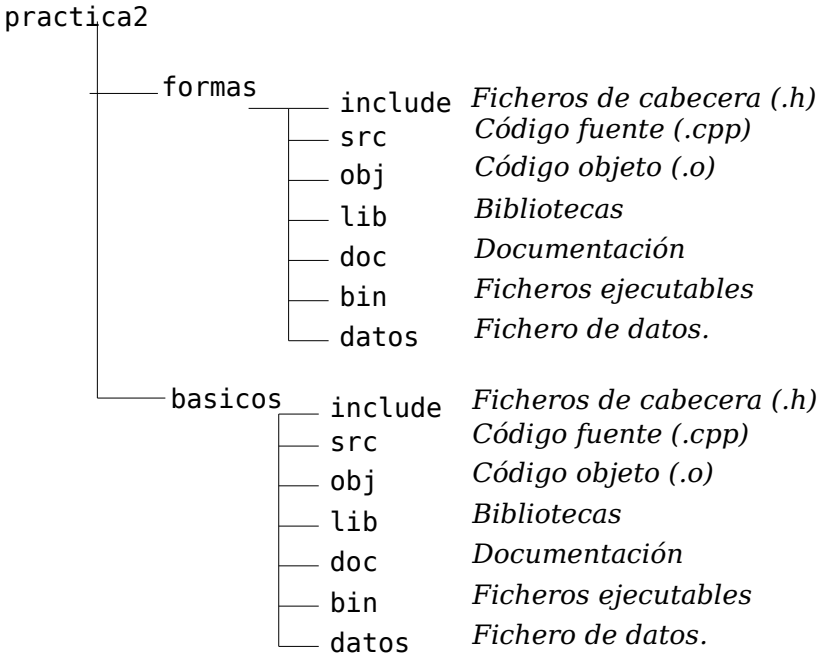
7. Material asociado

Para llevar a cabo la práctica el alumno tiene a su disposición tres módulos: imagen (imagen.h, imagen.cpp), imageES (imagenES.h y imagenES.cpp), con objeto de obtener los resultados usando las imágenes para el ejercicio 5.

8. Práctica a entregar

El alumno deberá empaquetar todos los archivos relacionados en el proyecto en un archivo con nombre "practica2.*tgz*" y entregarlo antes de la fecha que se publicará en la página web de la asignatura. Tenga en cuenta que no se incluirán ficheros objeto, ni ejecutables, ni la carpeta datos. Es recomendable que haga una "limpieza" para eliminar los archivos temporales o que se puedan generar a partir de los fuentes. El ejercicio 5 se desarrollara en le carpeta formas. El resto se desarrollara en la carpeta básicos.

El alumno debe incluir el archivo *Makefile* para realizar la compilación. Tenga en cuenta que los archivos deben estar distribuidos en directorios*:*



Para realizar la entrega, en primer lugar, realice la limpieza de archivos que no se incluirán en ella, y sitúese en la carpeta superior (en el mismo nivel de la carpeta "*practica2*") para ejecutar:

```
prompt% tar zcv practica2.tgz practica2
```

tras lo cual, dispondrá de un nuevo archivo practica2.tgz que contiene la carpeta practica2 así como todas las carpetas y archivos que cuelgan de ella.