**Pontificia Universidad Javeriana**
**Departamento de Ingeniería de Sistemas**
**Estructuras de Datos**
**Taller 4: Árboles de Partición, 2023-10**

# 1 Objetivo

Decodificar imágenes representadas como *quadtrees* en archivos de texto.

# 2 Recordatorio: compilación con g++

La compilación con g++ (compilador estándar que será usado en este curso para evaluar y calificar las entregas) se realiza con los siguientes pasos:

1. **Compilación**: de todo el código fuente compilable (**ÚNICAMENTE LOS ARCHIVOS CON EXTENSIONES** `*.c`, `*.cpp`, `*.cxx`)
   `g++ -std=c++11 -c *.c *.cxx *.cpp`
2. **Encadenamiento**: de todo el código de bajo nivel en el archivo ejecutable
   `g++ -std=c++11 -o nombre_de_mi_programa *.o`

   Nota: Estos dos pasos (compilación y encadenamiento) pueden abreviarse en un sólo comando:
   `g++ -std=c++11 -o nombre_de_mi_programa *.c *.cxx *.cpp`
3. **Ejecución**: del programa ejecutable anteriormente generado
   `./nombre_de_mi_programa`

**ATENCIÓN**: Los archivos de encabezados (*.h, *.hpp, *.hxx) **NO SE COMPILAN**, se incluyen en otros archivos (encabezados o código). Así mismo, los archivos de código fuente (*.c, *.cpp, *.cxx) **NO SE INCLUYEN**, se compilan. Si el programa entregado como respuesta a este Taller no atiende estas recomendaciones, automáticamente se calificará la entrega sobre un 25% menos de la calificación máxima.

# 3 Desarrollo del taller

Como ya se revisó en clase, los *quadtrees* se usan para separar el espacio bidimensional representado como una matriz de valores binarios (imagen binaria), como muestra la figura 1.

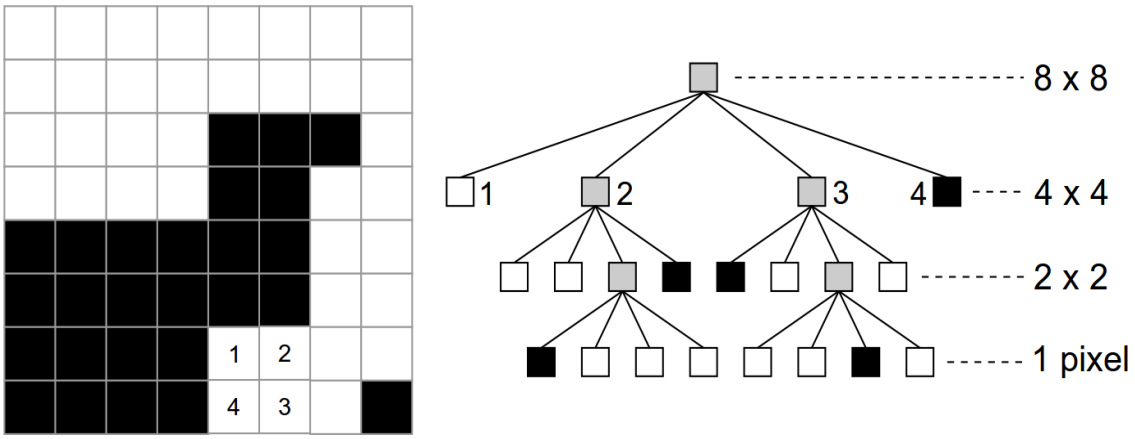


Figure 1: Ejemplo de un *quadtree*. Tomado de `https://en.wikipedia.org/wiki/Quadtree`

Las imágenes pueden almacenarse en disco de forma sencilla usando el formato PBM (*Portable BitMap format*, `https://en.wikipedia.org/wiki/Netpbm_format`). Este formato describe imágenes binarias en un archivo de texto, donde un valor de '1' representa el color negro y un valor de '0' representa el color blanco. El formato se define como:

```
P1
# Un comentario sobre la imagen ...
W H
v_11 v_12 ... v_WH
```

Donde:

- `P1` es conocido como un número mágico de dos bits, representa el tipo de archivo de imagen y la codificación en la que ésta se encuentra almacenada.
- El signo `'#'` indica líneas con comentarios.
- `W` y `H` son dos números enteros positivos que representan el ancho y el alto de la imagen, respectivamente.
- `v_ij` es el valor binario (`1` o `0`) del píxel `ij` de la imagen (ubicado en la fila `i` y la columna `j`).

Por ejemplo, para la imagen de la figura 1, su archivo PBM podría ser:

```
P1
# Esta es una imagen de prueba
8 8
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 1 1 1 0
0 0 0 0 1 1 0 0
1 1 1 1 1 1 0 0
1 1 1 1 1 1 0 0
1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 1
```

El recorrido en preorden del *quadtree* que representa esta imagen (derecha de la figura 1), se puede almacenar en un archivo de texto de la siguiente forma:

```
8 8
2020021000121020021001
```

Donde los dos primeros valores representan nuevamente el tamaño de la imagen (ancho y alto, respectivamente); `'0'` o `'1'` son los valores binarios de los nodos terminales (hojas) y `'2'` representa los nodos "grises" (o de mezcla de colores).

Teniendo en cuenta esto, el desarrollo del taller consistirá en diseñar (utilizando la plantilla de diseño de TADs vista en clase) e implementar un programa que permita generar imágenes PBM a partir del recorrido en preorden de su *quadtree* asociado. En particular, deben realizarse las siguientes tareas:

- Recibir dos parámetros por línea de comandos: el nombre de un archivo de texto que almacena el recorrido en preorden de un *quadtree* y el nombre de un archivo PBM donde se guardará la correspondiente imagen decodificada.
- Leer adecuadamente el recorrido en preorden desde el archivo de texto para reconstruir correctamente en memoria el *quadtree* correspondiente.
- Utilizar el *quadtree* reconstruido para identificar los bloques de color (0 ó 1) de la imagen, sus ubicaciones y tamaños; y con esta información generar la imagen PBM correspondiente.
- En el documento de diseño, incluir un corto comentario donde se analice la diferencia de tamaños de los archivos PBM contra los archivos de recorrido en preorden de los *quadtree*.
- En el documento de diseño, enumerar en un corto texto las descripciones de (lo que almacenan o representan) las imágenes reconstruídas a partir de los archivos de texto de prueba entregados junto con el presente enunciado.

# 4 Evaluación

La entrega se hará a través de la correspondiente actividad de UVirtual, antes de la medianoche del jueves 15 de abril. Se debe entregar un único archivo comprimido (único formato aceptado: .zip), nombrado con los apellidos de los integrantes del grupo. Este comprimido debe contener, dentro de un mismo directorio (sin estructura de carpetas interna), el documento de diseño (.pdf) y el código fuente del programa (.h, .hxx, .hpp, .c, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

La evaluación del taller tendrá la siguiente escala:

- **Excelente** (**5.0/5.0**): El estudiante diseñó correctamente (siguiendo la plantilla) e implementó una solución que utiliza correctamente el recorrido en preorden del *quadtree* para generar la correspondiente imagen PGM.
- **Bueno** (**3.5/5.0**): El estudiante diseñó correctamente (siguiendo la plantilla) e implementó una solución parcialmente correcta (guarda una imagen PGM invertida o rotada).
- **No fue un trabajo formal de ingeniería** (**3.0/5.0**): El estudiante implementó una solución completa o parcial, pero no la diseñó correcta o completamente.
- **Necesita mejoras sustanciales** (**2.0/5.0**): El estudiante diseñó y/o implementó una solución, pero no es completa o no soluciona lo pedido.
- **Malo** (**1.0/5.0**): El código entregado por el estudiante no compila en el compilador g++ (mínimo versión número 4.5).
- **No entregó** (**0.0/5.0**).