



PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2133 — Estructuras de Datos y Algoritmos
2020 - 1

Tarea 1

Fecha de entrega código: 9 de Mayo

Fecha de entrega informe: 10 de Mayo

Objetivos

- Modelar un problema y sus propiedades de manera recursiva, usando una estructura de datos adecuada para resolverlo eficientemente.
- Investigar de manera personal acerca de una estructura de datos para resolver un problema específico.

Introducción

Tras haber logrado superar el asalto de los piratas espaciales, los antiguos pasajeros del Space Titanic se encuentran enfrentados ante el más grande enemigo jamás conocido por la humanidad: el aburrimiento. Luego de varios intentos de vencer a RoboKasparov en el SpaceChess, discutir la legalidad de la SpacePizza con Spiña y demostrar el último teorema espacial de Fermat, se han dado por vencidos (la demostración era demasiado larga para el papel que les quedaba en el pod).

Todo parecía estar perdido hasta que — entre los antiguos componentes de navegación — encontraron una antigua consola Spoogle Spacetadia conectada directamente a los DCCervidores en la Tierra. Sin embargo, han descubierto que su conexión de SpaceNet no es lo suficientemente poderosa como para recibir un *stream* de imágenes en HD, por lo que necesitan que las imágenes que reciban los pods desde la tierra sean comprimidas antes de ser enviadas. Dado que el equipo docente sigue en cuarentena tras contraer la *Space Malaria* en sus vacaciones en el sudeste galáctico, ¡Es tu labor como programador de *Space-C* el enviar las texturas comprimidas de sus preciados juegos!



Imagen original, 7.2MB



Imagen comprimida, 49KB

Imágenes

Para un computador, una imagen no es más que una matriz de colores y existen diversos modelos matemáticos para representar colores en este formato, tales como RGB, HSV y CIE-Lab.

Para esta tarea, utilizaremos el espacio de color CIE-Lab, el cual permite mejores resultados al hacer operaciones de comparaciones entre colores, ya que se ajusta más a la percepción humana.

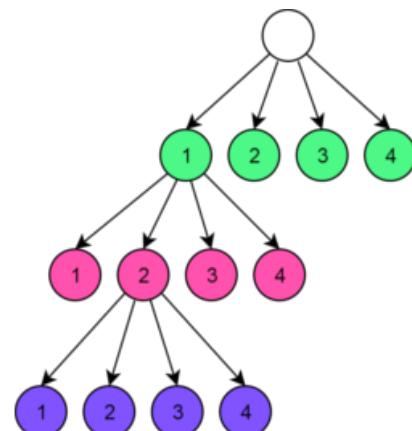
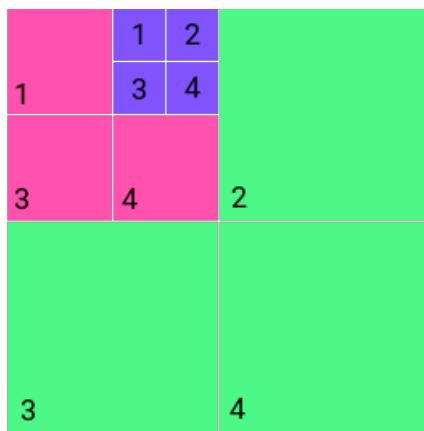
Esto significa que cada color está representado por una 3-tupla con los siguientes valores:

- L: canal de luminosidad, de 0 a 100
- a: canal cromática verde-magenta, de -128 a 128
- b: canal cromática azul-amarillo, de -128 a 128

Problema

Para esta tarea deberás implementar un algoritmo de compresión llamado, a falta de un mejor nombre, “Compresión mediante **QuadTree**”.

El Quadtree es una estructura de datos que divide los datos en 2 dimensiones en 4 cuadrantes iguales, donde cada cuadrante a su vez se organiza recursivamente como un QuadTree.



En este caso, el Quadtree tiene una profundidad de 3

En el caso de una imagen, cada píxel de esta correspondería a una hoja del árbol, mientras que los demás nodos representan grupos de píxeles dentro de un cuadrante dado.

La lógica detrás del algoritmo de compresión es agrupar los píxeles de colores similares dentro de un mismo cuadrante, para luego reemplazarlos por un solo color, que en este caso será el promedio entre los elementos de este.

Similitud

Para determinar qué tan similares son los colores de un cuadrante, utilizaremos como métrica la desviación estándar. Para un set de n datos $\{x_1, \dots, x_n\}$, esta se define de la siguiente manera:

$$\sigma = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_i - \mu)^2}$$

Donde μ es el promedio de los datos,

$$\mu = \frac{1}{n} \sum_{i=1}^n x_i$$

Este cálculo debe repetirse para cada uno de los canales de la imagen a tratar, es decir, debes calcular σ_L , σ_a , σ_b .

Definiremos arbitrariamente la desviación estándar de un cuadrante como el promedio entre las desviaciones estándar de cada uno de los canales:

$$\gamma = \frac{\sigma_L + \sigma_a + \sigma_b}{3}$$

Para calcular estos valores de manera eficiente puedes utilizar métodos incrementales¹.

Filtración del Quadtree

Este sub-algoritmo tiene un parámetro, que llamaremos α , el cual indica la desviación estándar máxima que puede tener un cuadrante. Dado α , para cada cuadrante que tenga $\gamma \leq \alpha$, se deja ese cuadrante como hoja en el árbol y el color que le corresponde en la imagen es μ , es decir, el promedio entre todos los colores del cuadrante. En caso contrario, el cuadrante se subdivide en 4, y se aplica este criterio recursivamente para cada uno de estos sub-cuadrantes.²

Llamaremos a esta operación un “filtro” del árbol según α .

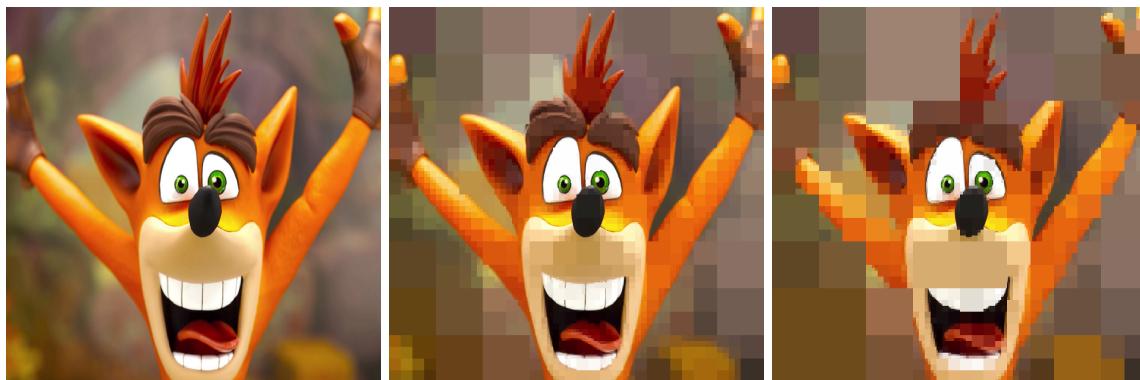


Imagen original

Imagen filtrada con $\alpha = 5$

Imagen filtrada con $\alpha = 10$

¹Puedes revisar el desarrollo de la fórmula [aquí](#)

²Puedes ver [aquí](#) un ejemplo interactivo de la subdivisión de los nodos del árbol (los cuadrantes están estilizados como círculos pero el principio es el mismo)

Compresión mediante Quadtree

El algoritmo de compresión mediante Quadtree recibe como parámetro la cantidad máxima de hojas h que pueden existir en el árbol comprimido.

Lo que hace este algoritmo es buscar, usando búsqueda binaria, el $\alpha \in \mathbb{N}$ más pequeño tal que se cumpla que al filtrar el árbol según este α , la cantidad de hojas del árbol sea menor a h . Los límites posibles para α van de 0 a 128.

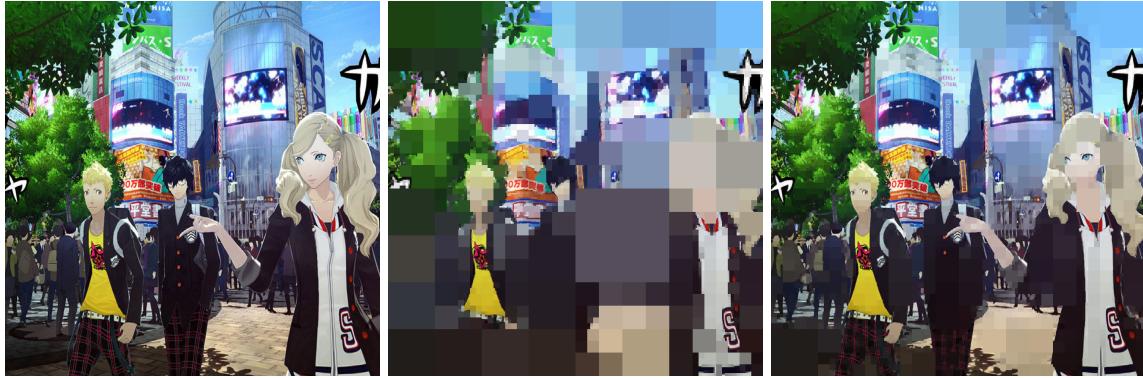


Imagen original

Imagen comprimida con $h = 1000$

Imagen comprimida con $h = 10000$

Se espera que desarrolles los algoritmos de filtro y compresión usando un Quadtree.

Se espera que al iniciar tu programa construyas el árbol a partir de la imagen, y que luego lo utilices mediante consultas para llevar a cabo los distintos pasos del algoritmo.

Es tu responsabilidad el investigar acerca de esta estructura así como sus detalles de implementación.

Librería y Código Base

Para el manejo de imágenes, tus ayudantes han preparado una librería a tu disposición. Esta se encarga de todo lo que es lectura y escritura de imágenes. Recuerda leerla atentamente y familiarizarte con su interfaz, así no perderás tiempo implementando funciones que te han sido entregadas.

Input

Tu programa deberá responder llamadas de la forma

```
./quadtree input_image output_image command param
```

En donde **command** indica el modo de funcionamiento y puede tener dos valores:

1. **filter**: Tu programa deberá **filtrar** la imagen **input_image** (en formato png) con **param** como α .
2. **compress**: Tu programa deberá **comprimir** la imagen **input_image** (en formato png) con **param** como h .

Puedes asumir que la imagen de input siempre será cuadrada y tanto su ancho como su largo serán siempre una misma potencia de 2.

Output

El output de tu programa es la imagen resultante, la cual deberás guardar en la ruta `output_image` que se te ha otorgado.

Análisis

Deberás escribir un informe de análisis donde menciones los siguientes puntos:

- Calcula y justifica la complejidad en notación \mathcal{O} para la construcción del árbol así como cada una de las funciones (filtrar y comprimir) en función de:
 - Número de píxeles de la imagen
 - El parámetro de la función (h o α).

Evaluación

La nota de tu tarea se descompone como se detalla a continuación:

- 80% a la nota de tu código, dividido en:
 - 40% que el output de los tests de `filter` sean correctos.
 - 30% que el output de los tests de `compress` sean correctos.
 - 5% valgrind reporta en tu código 0 *memory leaks*.
 - 5% valgrind reporta en tu código 0 errores de memoria.
- 20% a la nota del informe, basada en el cálculo de la complejidad teórica y análisis general del desempeño del algoritmo de compresión.

Si la imagen que entregas no coincide con la imagen esperada, tendrás automáticamente 0 puntos en ese test. Si tu algoritmo demora más de 10 segundos en un test (sin considerar el tiempo que toma leer / escribir el archivo de imagen), será cortado y tendrás 0 puntos en ese test. En el repositorio base de su tarea está explicado cómo medir el tiempo de su función sin considerar el leer / escribir el archivo de imagen. Tu programa se probará con diversas imágenes de tamaños crecientes.

Entrega

Código: GIT - Repositorio asignado. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

Informe: SIDING - En el cuestionario correspondiente, en formato PDF. Sigue las instrucciones del cuestionario. Se entrega a más tardar el día de entrega a las 23:59 hora de Chile continental.

BONUS

Buen uso del espacio y del formato (+5 décimas a la nota de Informe): La nota de tu informe aumentará en 5 décimas si tu informe está bien presentado y usa el espacio y formato a favor de entregar la información. Este bonus es entregado a criterio del corrector; no admite corrección.

Política de recorrección

Una vez entregadas las notas, todo alumno puede solicitar una recorrección. Si desean solicitar recorrección, deben seguir los pasos que se detallan a continuación:

1. Deben abrir un issue en el repositorio de su tarea solicitando formalmente recorregir su tarea y explicando por qué. Si hacen cambios en su código, deben indicar qué cambiaron y qué commit quieren que se revise. Si estos cambios no cambian la lógica de su algoritmo, no llevan asociado descuento. En caso contrario, hay un descuento aplicado en proporción a la cantidad de cambios que hagan a su algoritmo.
2. Deben enviar el link de dicha issue a caespinoza5@uc.cl con el encabezado [IIC2133] - Recorregir T1 - [Su nombre]. Ejemplo: [IIC2133] - Recorregir T1 - Juan Perez.