Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
IIC2133 - Estructura de datos y algoritmos

Pauta Tarea 1

1 de noviembre de 2021

2º semestre 2021 - Profesores Yadran Eterovic, Martín Muñoz

Component Tree:

- ¿Qué es?:

  Es una estructura de datos, específicamente un árbol, en el que la raíz contiene a la información completa, luego se divide este en trozos y cada trozo es contenido por un hijo de la raíz, y así se continúa dividiendo la información donde el padre siempre contiene toda la información de los hijos, hasta que no se pueda dividir más la información.

  Es útil para realizar análisis sobre trozos más pequeños de información, tales como segmentación o identificación.

  Evaluación :

  - (3 pts) si describe correctamente el tipo de estructura, como se guarda la información y su principal caso de uso
  - (1.5 pts) si le falta alguno de los elementos mencionados.

- Ventajas:
  - El tener todos los píxeles del vecindario en el nodo permite realizar un mejor análisis sobre el subconjunto, sirve especialmente cuando se busca analizar sub-objetos dentro del objeto principal, como partes específicas de una imagen, o trozos de información de un documento.

- Desventajas:
  - Utiliza mayor espacio, al guardar píxeles repetidos.
  - Demora más tiempo en su construcción, al tener que incluir píxeles repetidos.

- Puede ser redundante en problemas en los que el objeto de interés es el objeto principal, y no algún sub-objeto. Tal y como en filtros de color, o de filtros de interferencia, en los que se busca modificar el objeto principal.

Evaluación :

- (1.5 pts) Si menciona correctamente las ventaja, que presenta el árbol.
- (1.5 pts) Si menciona correctamente las desventajas que presenta el árbol.
- Consideración : Puntaje parcial (0.75 pts) en caso de que se le olvide mencionar alguna de estas, por otro lado si menciona otras no incluidas en el reporte pero que hagan sentido se da puntaje completo

Complejidad:

- Construcción árbol:

  Siguiendo la construcción de Max Tree con flood de Salembier, se obtiene que la construcción del árbol es $O(n \cdot k)$, con n la cantidad de píxeles, y k la cantidad de niveles de grises (profundidad del árbol).

  Luego, transformar un Max Tree a Component Tree se hace en $O(n \cdot k)$, simplemente mapeando los píxeles desde las hojas hacia la raíz. Es decir, los píxeles de las hojas se agregan a sus padres, luego los píxeles de estos padres a sus padres correspondientes. Cada píxel puede pertenecer a un máximo de k nodos, ya que hay solo k niveles de grises, y como este algoritmo toma un tiempo por pixel agregado, el tiempo máximo posible es $O(n \cdot k)$.

  Como transformar el Max Tree a Component Tree es también $O(n \cdot k)$, el tiempo total de construcción es $O(n \cdot k)$.

  Evaluación :

  - (3.5 pts) Si describe y explica la complejidad de construcción de su árbol (no debe ser necesariamente el método explicado arriba)
  - (1.5 pts) Si tiene error en el calculo de su complejidad , no explica como llega a esta o se olvida de diferenciar cantidad de píxeles con profundidad del árbol.
  - (0.75 pts) En caso de tener 2 de los 3 errores de arriba
  - Consideración : En caso de que su implementación sea mas ineficiente no se descuenta puntaje, siempre y cuando este bien calculado y argumentado.

- filtro:

El filtro es $O(n+N)$ con n cantidad de píxeles, y N cantidad de nodos, ya que se tienen que recorrer todos los píxeles para calcular los $n(N,G)$ y $ps(N,G)$. Se recorre cada píxel solo una vez, ya que se utiliza el valor del nodo hijo para calcular el del padre, así no se repiten píxeles. Luego, hay que recorrer cada nodo para resolver la segmentación, para decidir cuál nodo guardar y cuál nodo no. Pero como siempre ocurre que $n \geq N$, se tiene que el tiempo del filtro es simplemente $O(n)$.

- (2.5 pts) Si describe y explica correctamente la complejidad del filtro.
- (1.0 pts) En caso de tener un error de calculo, no tener una explicación clara u olvidar diferenciar cantidad de píxeles con cantidad de nodos.
- (0.5 pts) En caso de tener 2 de los 3 errores de arriba

Calculo de nota :

(Puntaje total / 12) * 6 + 1