Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2133 — Estructuras de Datos y Algoritmos
2021 - 1

# Pauta Tarea 0

---

## a) Complejidades Teoricas

Sea $n$ el número de nodos totales
$\longrightarrow$ Concluir que la complejidad de buscar un paciente cero es de $\mathcal{O}(1)$. Justificando en base a las propiedades de la estructura utilizada. (0.2pts por complejidad, 0.3pts por justificación)

$\longrightarrow$ Para los eventos {`RECOVERED, POSITIVE, NEGATIVE, INFORM, CORRECT`}:

- Se calcula la complejidad del evento en base al número de nodos totales $n$, llegando a una complejidad de $\mathcal{O}(n)$ para el peor caso[1]. (0.3pts por evento)
- Se justifica la complejidad correctamente, ya sea con una descripción del algoritmo, un fragmento del código o mediante palabras. (0.6pts por evento)

$\longrightarrow$ Para el evento `ADD_CONTACTS`, dado un número de contactos a agregar $m$, se puede concluir que la complejidad es $\mathcal{O}(n+m)$ para el peor caso, **SIN EMBARGO** en el anunciado se pide que se calcule la complejidad en función de la cantidad de personas en el arbol por lo que se acepta como correcto concluir que la complejidad es $\mathcal{O}(n)$ . (0.2pts por complejidad, 0.3pts por justificación)

$\longrightarrow$ Para el evento `STATISTICS`, bajo una implementación óptima, Concluir que la complejidad es $\mathcal{O}(1)$ en cualquier caso[2]. (0.2pts por complejidad, 0.3pts por justificación)
**Consideraciones sobre el puntaje**

- Si se especifica que tiene algoritmos diferentes pero correctos con la complejidad bien calculada, se asigna puntaje completo
- Si da la complejidad correcta sin justificación se asigna la mitad del puntaje
- Si realiza cálculos en base a otras variables como la profundidad, se asigna puntaje siempre y cuando se concluya o mencione que en el peor caso $n = depth$. En caso contrario no se considerará correcto.

## b) Modelación

La Justificación ha de incluir lo siguiente

- Para el peor caso, tanto la inserción, busqueda y eliminacion de un elemento son iguales a la modelación original. ($depth = n$) (3 pts, uno cada uno)
- Comentar y argumentar que en el caso promedio la complejidad de la búsqueda es más eficiente, dado que no tiene que buscar a través de la lista ligada. (1 pts)

- Comentar las desventajas de utilizar un array vs una linked list. (Por ejemplo hacer `ADD_CONTACTS` dos veces sobre un mismo nodo, implica realizar operaciones de copiar el primer array a un array más grande). (2 pts)

## c) Comparacion C y Python

La respuesta ha de incluir lo siguiente

- Al menos una cuantificación numérica sobre tiempo de ejecución de un algoritmo en C vs el mismo en Python, ha de ser explícita. (Tablas o gráficos).
  Donde se ha de indicar que C es más rápido que Python (3 pts)
- Comentar el porque de la diferencia. (3 pts)

**Consideraciones del puntaje**

- Si los datos empíricos resultaron que los tiempos de python eran más rápidos a C. Se asigna puntaje completo siempre y cuando se argumente que dichos tiempos han sido afectados por una mala implementación o procesos externos.

**Sobre el Cálculo de la Complejidad**

1. Nótese que dado que todos los eventos inicialmente dependen de encontrar un nodo (o dos) para posteriormente operar sobre él, en el peor caso a lo más se han de recorrer $n$ nodos para encontrarlo (El nodo es la última *hoja* del árbol de nodos).

   Nótese que además, en los casos donde la operación posterior a encontrar el nodo no es de complejidad constante. Igualmente la suma de ambas operaciones en el peor caso será $\mathcal{O}(n)$. Tómese el caso de `NEGATIVE` como ejemplo, si cortamos un nodo que no sea el último que se encuentra luego de recorrer $m$ nodos, con una cantidad total de descendientes $o$, en el peor caso, o serian el resto de nodos no recorridos $n - m$ y por lo tanto el número total de operaciones será de $n$.
   (Por otro lado, se puede concluir que la operación de eliminar descendientes es a lo más $\mathcal{O}(n)$ y por lo tanto, la suma de ambas operaciones será igualmente $\mathcal{O}(n)$.
   Un proceso similar lleva a la conclusión de que la operación `CORRECT`, ha de tener complejidad On, solo que en vez de revisar el peor caso de eliminar los descendientes, se revisa el caso de cambiar el estado de los mismos.

2. La implementación más eficiente de STATISTICS, es tener memoria asignada para llevar cuenta de los datos mientras las operaciones ocurren. Luego la operación es solo entregar dichos datos. Si por otro lado se realiza un cálculo de forma recursiva sobre los nodos, se concluye que la complejidad en el peor caso es $\mathcal{O}(n)$