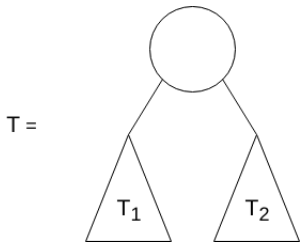
PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC 2133 — Estructuras de Datos y Algoritmos
Interrogación 2
Primer Semestre, 2017

Duración: 2 hrs.

1. a. (3 puntos) Ejemplo de solución correcta y su distribución de puntaje:

Por definición de AVL, las alturas de ambos subárboles no pueden diferir en más de 1 (0,25).
Sea T un AVL:



La altura h(T) está dada por

$$h\left(T\right) = 1 + max\left(h\left(T_1\right), h\left(T_2\right)\right)$$

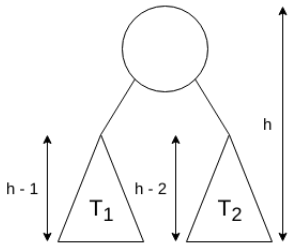
Pero por definición de AVL

$$|h\left(T_1\right) - h\left(T_2\right)| \leq 1 \quad \text{(0,25)}$$

En el peor caso, la igualdad es estricta.

$$|h\left(T_1\right) - h\left(T_2\right)| = 1 \quad \text{(0,5)}$$

Asumiendo que T_1 es más alto, $h(T_1) = h(T_2) + 1, \quad h(T) = h(T_1) + 1 = h(T_2) + 2$



Sea $N(h)$ la cantidad de nodos mínima para un árbol de altura h Podemos definirla en función de las alturas de sus subárboles:

$$N(h) = 1 + N(h-1) + N(h-2), \quad N(0) = 1 \quad \text{(0,5)}$$

Además se cumple que a mayor altura, más nodos. Esto significa que $N(h-1) > N(h-2)$ Luego,

$$\begin{aligned} N(h) &> 1 + N(h-2) + N(h-1) > 2N(h-2) \quad (0,5) \\ &> 2(1 + N(h-3) + N(h-4)) \\ &> 2^2 N(h-4) \\ &> 2^2(1 + N(h-5) + N(h-6)) \\ &> 2^3 N(h-6) \\ &> 2^3(1 + N(h-7) + N(h-8)) \\ &> 2^4 N(h-8) \\ &\vdots \\ &> 2^i N(h-2i) \quad (0,5) \\ &\text{Luego, para } i = \frac{h}{2} \\ &> 2^{h/2} N(0) \end{aligned} \tag{1}$$

Luego, como $N(0) = 1$ queda que $N(h) > 2^{h/2}$. Aplicamos logaritmo en base dos:

$$\begin{aligned} log_2(n) &> \frac{h}{2} \\ h \in O(log(n)) &\quad (0,5) \end{aligned}$$

b. (3 puntos) La demostración de esta pregunta fue vista en clases. En esta página hay un ejemplo de explicación:

http://www.mathcs.emory.edu/ cheung/Courses/323/Syllabus/Trees/AVL-insert.html

El puntaje se asignó:

(1) Explicar qué ocurre en un desbalanceo: diferencia de 2, 4 posibles configuraciones, etc.
(1) Dar a entender un caso general de desbalanceo (o cada caso), enseñando valores de alturas respectivas y detalles de cada nodo que forman parte de la rama desbalanceada.
(1) Mostrar qué ocurre al balancear (en un solo paso) y calcular los nuevos valores de altura. Explicar por qué se balancea.

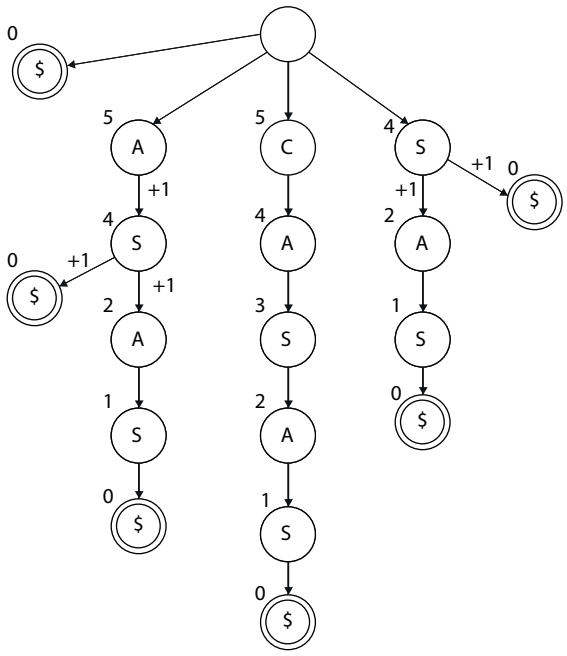
2. Existen distintas formas de implementar este algoritmo, a continuación se muestra una de las cuantas:

Debido a que el uso de memoria sería cuadrático si es que se intentan guardar todos los substrings, se debe pensar en poder representar los substrings de una manera eficiente, en este caso una forma ideal es implementar un `Suffix Trie`, ya que todo substring se puede representar como un prefijo de algún sufijo de w.

Es claro que la construcción del `Suffix Trie` toma tiempo $O(|w|^2)$, ya que se deben generar todos los sufijos de w y luego iterar sobre ellos para colocarlos en el árbol. También requiere espacio $O(|w|^2)$, ya que en el peor caso no hay intersecciones entre los sufijos y por lo tanto se guardarían todos los sufijos por separado.

Luego para hacer la consulta es necesario construir una forma eficiente los índices en memoria, una solución es por cada nodo v del árbol guardar la cantidad de nodos n que tendría el subárbol tomando como raíz a v. Esto se puede hacer en tiempo de construcción, basta con agregar un contador por cada nodo y sumar la cantidad de hijos hacia arriba una vez que se agrega un sufijo. Además consideramos agregar a cada referencia desde un padre un contador que diga las veces que se repitió el camino al momento de insertar el sufijo.

Considerando el ejemplo el `Trie` se vería así:



Finalmente para la consulta, basta recorrer el `Trie` haciendo *matching* de cada letra s_i de s, partiendo del nodo raíz, consideramos que se parte con una variable num inicialmente de valor 0 que representa el índice a retornar.

- Si s_i no hace *match* con ningún nodo entonces retornar **-1**
- Si s_i hace *match* con algún hijo entonces bajar a ese hijo, hacer $s_i = s_{i+1}$ y sumar a num uno más la cantidad de repeticiones en la referencia del *match*, más el valor n de cada uno de los nodos hermanos menores.
- Si s_i es el caracter de término entonces retornar num

Esto toma $O(\Sigma \cdot |s|)$, donde Σ es la cantidad de caracteres en $ASCII$, ya que en el peor caso un nodo tiene como hijo todas las letras posibles. Luego como Σ es constante la complejidad de consulta es $O(|s|)$.

Repartición puntaje:

- **(2 ptos):** Representar eficientemente los substrings posibles, mostrando que se cumplen las condiciones de tiempo y memoria. Además se considera que la consulta de verificar que s sea un substring de w tome $O(|w|)$.
- **(4 ptos):** Dar una estructura de representación y consulta que permita obtener por el índice (considerando las repeticiones) de s en S_w, mostrando que se cumplen todas las condiciones de tiempo y memoria.

3. *a*) Primero deben dar un ejemplo de un árbol 2-3-4 que en el cuál al eliminar un nodo, se producen dos fusiones. Las fusiones ocurren cuando al eliminar un nodo todos sus hermanos son nodo 2, es decir, tienen solo una clave. Esto, según el algoritmo visto en clases, hace que no puedan, los hermanos del nodo eliminado, prestarle una llave al espacio dejado por el nodo eliminado. Debido a esto, se produce una fusión entre el padre del nodo eliminado y algún hermano (bajando una clave del padre para unirse con la del hermano). Luego si el padre se queda sin claves, y sus hermanos también son nodos dos, ocurre la segunda fusión.

   Mostrar el árbol con las fusiones es **1 punto**.

   Dibujar su correspondiente árbol rojo-negro (dado que se hizo bien el árbol 2-3-4) con su correspondiente eliminación de nodo es **1 punto**. Es importante que el árbol rojo-negro ïmite"de alguna forma lo que está

asiendo el árbol 2-3-4 para que siga siendo su árbol asociado.

Finalmente, se debe hacer la relación de la fusión en el árbol 2-3-4 con lo que ocurre en el árbol rojo-negro. Es importante notar que una fusión corresponderá a un cambio de color en el árbol rojo-negro. Esto es porque un padre (que tenía solo una clave, es decir, que era negro) pasa a tener dos claves por haber sido unido con el hermano del nodo eliminado, por lo que se le cambia el color a rojo. (**2 puntos**).

b) En esta pueden haber varias soluciones dependiendo de cómo definieron el peor caso. Una solución era la siguiente:

- Peor caso: cuando la inserción produce mayor cantidad de rotaciones y de recoloraciones. (**1 punto**).
- Familia: Este es el caso de cuando al insertar un nodo, su padre será rojo y su tío también lo será. Y hacia arriba, los colores se van intercalando. Esto, al insertar un nodo, generará la mayor cantidad de cambios de color (hasta la raíz). (**1 punto**).