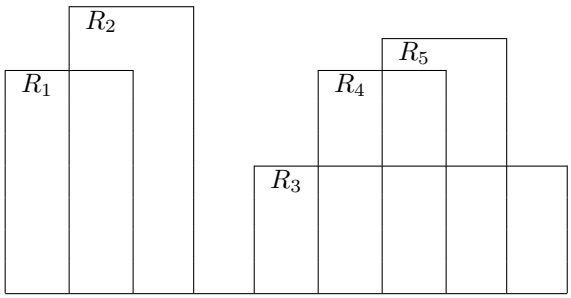
PONTIFICIA UNIVERSIDAD CATOLICA DE CHILE
ESCUELA DE INGENIERIA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACION

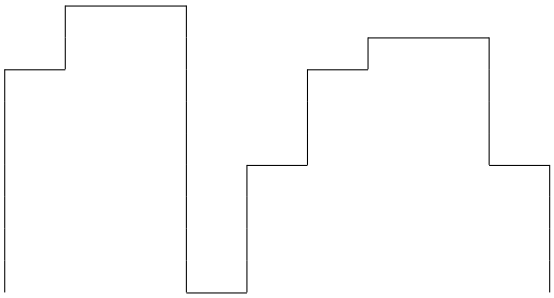
**Diseño y Análisis de Algoritmos - IIC2283**
**Tarea 2**
**Entrega: lunes 17 de octubre**

1. [1.5 puntos] Una manera de representar un rectángulo $R$ en el plano es dando sus coordenadas inferiores $(b, a)$, $(c, a)$ y su altura $h$, suponiendo que $b < c$ y $h > 0$. En esta pregunta vamos a suponer adicionalmente que para cada rectángulo $R$ se tiene que $a = 0$ y $b$, $c$, $h$ son números naturales. De esta forma representamos $R$ como una tupla $(b, h, c)$ de números naturales tales que $b < c$ y $h > 0$.

   En este problema vamos a utilizar rectángulos para representar edificios. Por ejemplo, suponga que $R_1 = (0, 7, 2)$, $R_2 = (1, 9, 3)$, $R_4 = (4, 4, 9)$, $R_4 = (5, 7, 7)$ y $R_5 = (6, 8, 8)$. Estos cinco rectángulos representan a los siguientes edificios:




   En esta pregunta usted debe construir un algoritmo que, dado un conjunto de edificios representados por rectángulos, construye la línea del horizonte para estos edificios. Por ejemplo, para los cinco rectángulos mostrados arriba el algoritmo debe construir la siguiente línea:

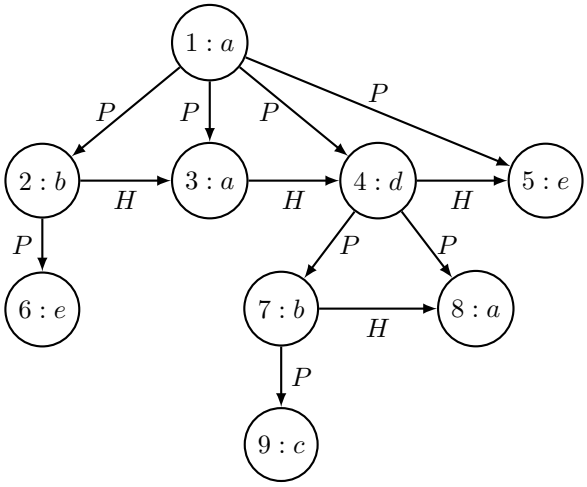


   Para representar la línea del horizonte se debe indicar los puntos donde la línea cambia de altura: (0, 7), (1, 9), (3, 0), (4, 4), (5,7), (6,8), (8,4) y (9,0), la cual debe estar ordenada de menor a mayor por

la primera coordenada. De esta forma, el algoritmo debe recibir un conjunto de rectángulos, y debe retornar la línea del horizonte para estos rectángulos en el formato descrito.

**Importante:** En esta pregunta debe desarrollar un algoritmo que en el peor caso sea $O(n \cdot \log_2(n))$, donde $n$ es el tamaño de la entrada. Además, debe demostrar que su algoritmo funciona en este tiempo. Para simplificar la resolución del problema puede suponer que la entrada siempre contiene un rectángulo con coordenadas $(0, h, b)$.

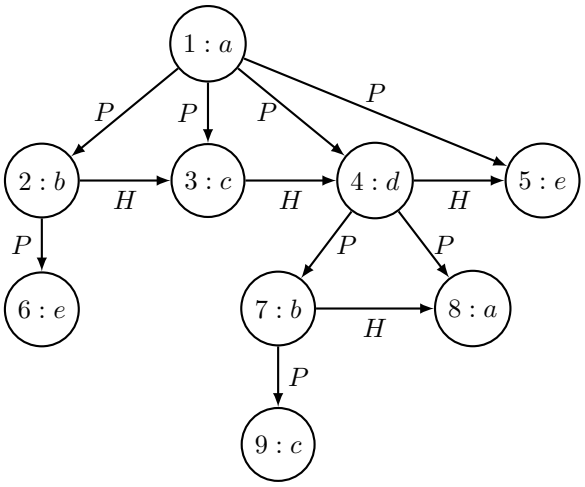
2. [1.5 puntos] Dado un alfabeto $\Sigma$, un *árbol ordenado sobre* $\Sigma$ (o *árbol ordenado* si $\Sigma$ está implícito en el contexto) es un árbol dirigido donde los hijos de cada nodo están ordenados y cada nodo recibe una etiqueta de $\Sigma$. Representamos un árbol ordenado $T$ como una tupla $(N, P, H, \lambda)$ donde $N \neq \emptyset$ es el conjunto de nodos del árbol, $P \subseteq N \times N$ es la relación de padre, $H \subseteq N \times N$ es la relación de hermano inmediato y $\lambda : N \to \Sigma$ es la función que asigna una etiqueta en $\Sigma$ a cada nodo. Por ejemplo, para el siguiente árbol:



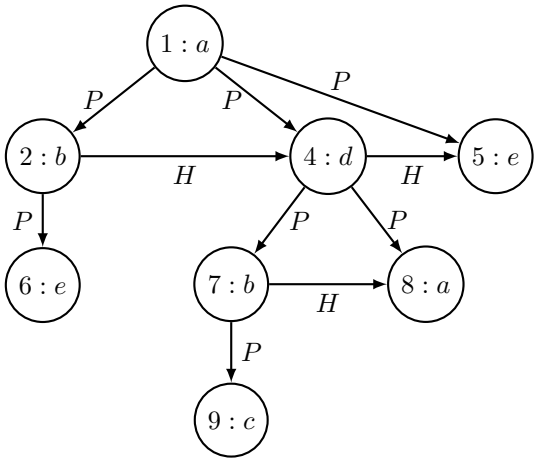

tenemos que $N = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $P = \{(1, 2), (1, 3), (1, 4), (1, 5), (2, 6), (4, 7), (4, 8), (7, 9)\}$, $H = \{(2, 3), (3, 4), (4, 5), (7, 8)\}$ y $\lambda(1) = a$, $\lambda(2) = b$, $\lambda(3) = a$, $\lambda(4) = d$, $\lambda(5) = e$, $\lambda(6) = e$, $\lambda(7) = b$, $\lambda(8) = a$, $\lambda(9) = c$. Vamos a utilizar este árbol ordenado en los siguientes ejemplos por lo que lo denotamos como $T_{ej}$.

Para modificar un árbol ordenado $T = (N, P, H, \lambda)$ podemos ejecutar tres operaciones.

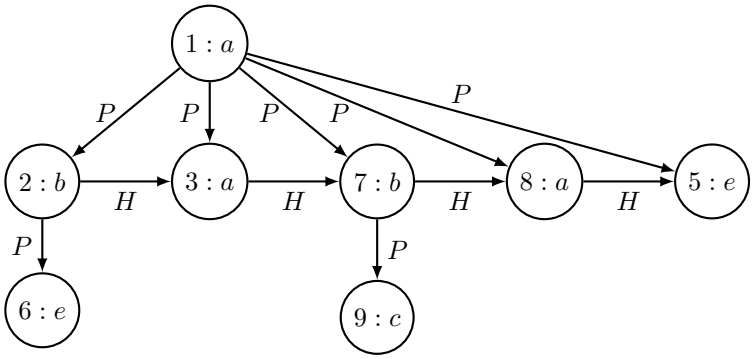
- Dado $v \in N$ y $a \in \Sigma$, la operación cambiar$(T, v, a)$ cambia la etiqueta de $v$ en $T$ por $a$. Por ejemplo, el siguiente es el resultado de ejecutar cambiar$(T_{ej}, 3, c)$:




- Dado $v_1, v_2 \in N$ tales que $v_1$ es el padre de $v_2$ en $T$, la operación eliminar$(T, v_2)$ elimina el nodo $v_2$ desde $T$, dejando los hijos de $v_2$ como hijos de $v_1$ en la posición que ocupaba $v_2$ y en el mismo orden en que aparecían como hijos de $v_2$. Por ejemplo, el siguiente es el resultado de ejecutar eliminar$(T_{ej}, 3)$:

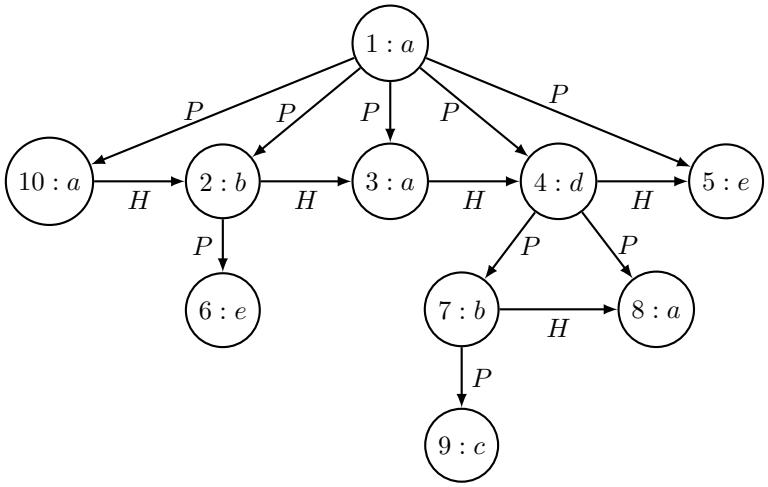



Mientras que el siguiente es el resultado de ejecutar eliminar$(T_{ej}, 4)$:

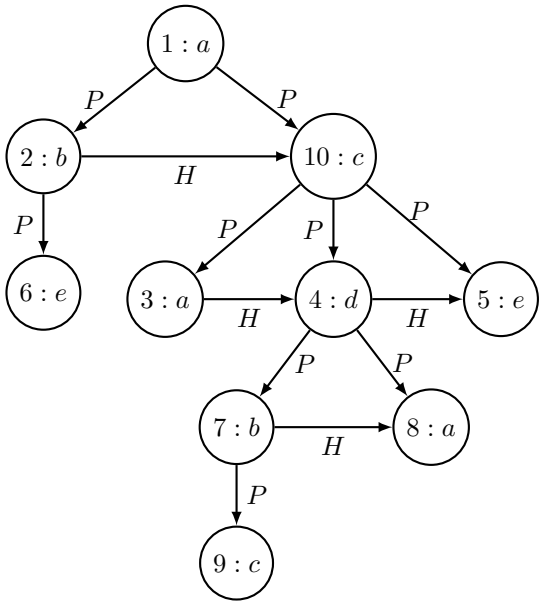



- Dado $b \in \Sigma$ y $v \in N$ tal que $v_1, \ldots, v_\ell$ es la secuencia de hijos de $v$ en $T$, la operación agregar$(T, v, b, j, k)$ es válida si $j \in \{0, \ldots, \ell\}$, $k \in \{0, \ldots, \ell\}$ y $j + k \leq \ell$. Si la operación agregar$(T, v, b, j, k)$ es válida, entonces al ejecutarla se crea un nuevo identificador $u$ para un nodo

($u \notin N$), se deja $b$ como la etiqueta de $u$, se reemplaza la secuencia de hijos $v_{j+1}, \ldots, v_{j+k}$ de $v$ por $u$ (en particular, se deja $u$ como hijo de $v$ en la posición $j+1$) y se deja $v_{j+1}, \ldots, v_{j+k}$ como la secuencia de hijos de $u$. Por ejemplo, el siguiente es el resultado de ejecutar agregar$(T_{ej}, 1, a, 0, 0)$:




Mientras que el siguiente es el resultado de ejecutar agregar$(T_{ej}, 1, c, 1, 3)$:




Dados dos árboles ordenados $T_1$ y $T_2$, definimos ted$(T_1, T_2)$ como el menor número de operaciones cambiar, eliminar y agregar que aplicadas desde $T_1$ generan $T_2$.

En esta pregunta debe utilizar la técnica de programación dinámica para desarrollar un algoritmo que calcule ted$(T_1, T_2)$ en tiempo polinomial en el peor caso. En particular, debe analizar la complejidad de su algoritmo y demostrar que funciona en este tiempo.

3. [1.5 puntos] Dada una secuencia $q_1$, ..., $q_\ell$ de números racionales y una secuencia $[a_1, b_1]$, ..., $[a_m, b_m]$ de intervalos de números racionales, decimos que $M \subseteq \{1, \ldots, m\}$ es un cubrimiento para $q_1$, ..., $q_\ell$ si:

$$\{q_1, \ldots, q_\ell\} \quad \subseteq \quad \bigcup_{j \in M} [a_j, b_j].$$

Además, decimos que $M$ es un cubrimiento mínimo para $q_1, \ldots, q_\ell$ si (i) $M$ es un cubrimiento para $q_1, \ldots, q_\ell$, y (ii) para todo cubrimiento $M'$ para $q_1, \ldots, q_\ell$ se tiene que $|M| \leq |M'|$.

Construya un algoritmo que reciba como entrada una secuencia $q_1, \ldots, q_\ell$ de números racionales y una secuencia $[a_1, b_1], \ldots, [a_m, b_m]$ de intervalos de números racionales, y retorne un cubrimiento mínimo para $q_1, \ldots, q_\ell$ si este cubrimiento existe, y retorne **no** si tal cubrimiento no existe.

**Importante:** El algoritmo desarrollado en esta pregunta debe ser codicioso y debe funcionar en tiempo polinomial. Además, debe demostrar que su algoritmo funciona correctamente, y debe analizar su complejidad.

4. [1.5 puntos] Un hipergrafo $H$ es una tupla $(N, A)$ donde $N$ es un conjunto de nodos y $A \subseteq 2^N$ es un conjunto de arcos. Nótese que cada arco en un hipergrafo es un subconjunto del conjunto de nodos del hipergrafo, y por lo tanto puede contener más de dos nodos. Los hipergrafos son considerados entonces como generalizaciones de los grafos.

   Dado un hipergrafo $H = (N, A)$ y $T \subseteq N$, decimos que $T$ es un transversal de $H$ si para cada $e \in A$ se tiene que $T \cap e \neq \emptyset$. Por ejemplo, si $H = (N, A)$ donde $N = \{1, 2, 3, 4\}$ y $A = \{e_1, e_2, e_3\}$ con $e_1 = \{1\}$, $e_2 = \{1, 2, 3\}$ y $e_3 = \{2, 3\}$, entonces tenemos que $T = \{1, 2\}$ es un transversal de $H$ puesto que $T \cap e_1 = \{1\}$, $T \cap e_2 = \{1, 2\}$ y $T \cap e_3 = \{2\}$, y tenemos que $T' = \{2, 3\}$ no es un transversal de $H$ puesto que $T' \cap e_1 = \emptyset$. Decimos además que $T$ es un transversal minimal de $H$ si (i) $T$ es un transversal de $H$, y (ii) para cada transversal $T'$ de $H$ no se tiene que $T' \subsetneq T$ (vale decir, no hay un transversal de $H$ propiamente contenido en $T$).

   En esta pregunta debe desarrollar un algoritmo de tiempo polinomial que, dado un hipergrafo $H$, calcula un transversal minimal de $H$.

   **Importante:** En esta pregunta debe analizar la complejidad de su algoritmo y demostrar que funciona en tiempo polinomial. Además, debe demostrar que el algoritmo desarrollado es correcto.