Fundamentos de Análisis y Diseño de Algoritmos

Estructuras de Datos

Carlos Alberto Ramirez Restrepo

Programa de Ingeniería de Sistemas
Escuela de Ingeniería de Sistemas y Computación,
home page: `http://eisc.univalle.edu.co/`
`carlos.a.ramirez@correounivalle.edu.co`

Plan

Generalidades

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

Árboles Binarios de Búsqueda

Estructuras de Datos

Generalidades

- Una estructura de datos es una forma de organizar un conjunto de datos con el objetivo de facilitar su manipulación.
- Una estructura de datos define la organización e interrelación de los datos y un conjunto de operaciones que se pueden realizar sobre ellos.
- Las principales operaciones sobre una estructura de datos son:
  - Adicionar un nuevo elemento a la estructura
  - Remover un elemento de la estructura
  - Buscar un valor en la estructura

Estructuras de Datos

Generalidades

- Una estructura de datos es una forma de organizar un conjunto de datos con el objetivo de facilitar su manipulación.
- Una estructura de datos define la organización e interrelación de los datos y un conjunto de operaciones que se pueden realizar sobre ellos.
- Las principales operaciones sobre una estructura de datos son:
  - Adicionar un nuevo elemento a la estructura
  - Remover un elemento de la estructura
  - Buscar un valor en la estructura

Estructuras de Datos

Generalidades

- Una estructura de datos es una forma de organizar un conjunto de datos con el objetivo de facilitar su manipulación.
- Una estructura de datos define la organización e interrelación de los datos y un conjunto de operaciones que se pueden realizar sobre ellos.
- Las principales operaciones sobre una estructura de datos son:
  - Adicionar un nuevo elemento a la estructura
  - Remover un elemento de la estructura
  - Buscar un valor en la estructura

Estructuras de Datos

Generalidades

- Una estructura de datos es una forma de organizar un conjunto de datos con el objetivo de facilitar su manipulación.
- Una estructura de datos define la organización e interrelación de los datos y un conjunto de operaciones que se pueden realizar sobre ellos.
- Las principales operaciones sobre una estructura de datos son:
    - Adicionar un nuevo elemento a la estructura
    - Remover un elemento de la estructura
    - Buscar un valor en la estructura

Estructuras de Datos

Generalidades

- Una estructura de datos es una forma de organizar un conjunto de datos con el objetivo de facilitar su manipulación.
- Una estructura de datos define la organización e interrelación de los datos y un conjunto de operaciones que se pueden realizar sobre ellos.
- Las principales operaciones sobre una estructura de datos son:
  - Adicionar un nuevo elemento a la estructura
  - Remover un elemento de la estructura
  - Buscar un valor en la estructura

Estructuras de Datos

Generalidades

- Una estructura de datos es una forma de organizar un conjunto de datos con el objetivo de facilitar su manipulación.
- Una estructura de datos define la organización e interrelación de los datos y un conjunto de operaciones que se pueden realizar sobre ellos.
- Las principales operaciones sobre una estructura de datos son:
    - Adicionar un nuevo elemento a la estructura
    - Remover un elemento de la estructura
    - Buscar un valor en la estructura

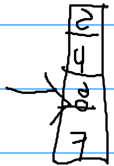
Estructuras de Datos

Generalidades

- Cada estructura de datos posee ventajas y desventajas en relación a la simplicidad y eficiencia en la realización de cada operación.
- De esta manera, la elección de la estructura de datos apropiada para cada problema depende de factores como la frecuencia y el orden en que se realiza cada operación sobre los datos.

Pila como arreglo

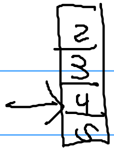
Por ejemplo 3 elemento



p.pop() p.push(4)

p.pop() p.push(8)

Caso cola



c.enqueue()

c.enqueue()

c.front()

Para retornar al estado inicial, toca sacar todos los elementos y volverlos a encolar

Estructuras de Datos

Generalidades

- Cada estructura de datos posee ventajas y desventajas en relación a la simplicidad y eficiencia en la realización de cada operación.
- De esta manera, la elección de la estructura de datos apropiada para cada problema depende de factores como la frecuencia y el orden en que se realiza cada operación sobre los datos.

Estructuras de Datos

Generalidades

Estructuras de Datos Elementales:

- Pilas
- Colas
- Listas Enlazadas
- Árboles
- Tablas Hash

Estructuras de Datos

Generalidades

No obstante, nos enfocaremos en algunas estructuras de datos más avanzadas como:

- Estructuras para conjuntos disyuntos
- Árboles de búsqueda
- Árboles Rojinegros, árboles B
- Heap binomial

Plan

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

- Una estructura de datos de conjuntos disyuntos mantiene una colección de conjuntos $S = \{S_1, S_2, \ldots, S_k\}$ de conjuntos disyuntos dinámicos.
- Cada conjunto se identifica mediante un elemento representativo.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

- Una estructura de datos de conjuntos disyuntos mantiene una colección de conjuntos $S = \{S_1, S_2, \ldots, S_k\}$ de conjuntos disyuntos dinámicos.
- Cada conjunto se identifica mediante un elemento representativo.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

- En algunas aplicaciones, no importa cual elemento de cada conjunto es usado como elemento representativo.
- Solo es necesario que al consultar dos veces sin que hayan modificaciones entre ellas, se obtenga la misma respuesta en ambos casos.
- Sin embargo, en otras aplicaciones se requiere una regla para escoger el elemento representativo, por ejemplo, escoger el elemento más pequeño.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

- En algunas aplicaciones, no importa cual elemento de cada conjunto es usado como elemento representativo.
- Solo es necesario que al consultar dos veces sin que hayan modificaciones entre ellas, se obtenga la misma respuesta en ambos casos.
- Sin embargo, en otras aplicaciones se requiere una regla para escoger el elemento representativo, por ejemplo, escoger el elemento más pequeño.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

Sea x un elemento, la estructura de datos de conjuntos disyuntos soporta las siguientes operaciones:

- `Make-Set`(x): Esta operación crea un nuevo conjunto cuyo único elemento (y por consiguiente representativo) es x. Dado que los conjuntos son disyuntos, es requerido que x no esté en los otros conjuntos.
- `Find-Set`(x): Esta operación retorna un puntero al elemento representativo del conjunto que contiene a x.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

Sea x un elemento, la estructura de datos de conjuntos disyuntos soporta las siguientes operaciones:

- `Union`(x, y): Esta operación une los conjuntos que contienen a x y y, denotados como S_x y S_y. El elemento representativo del conjunto resultante es cualquier elemento.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

En el análisis del tiempo de ejecución de las estructuras de conjuntos disyuntos se consideran 2 parámetros:

- n, el número de operaciones `Make-Set`(x) y
- m, el total de operaciones `Make-Set`(x), `Find-Set`(x) y `Union`(x, y).

Puesto que los conjuntos son disyuntos, el número máximo de operaciones `Union`(x, y) es $n - 1$.

Además, dado que las operaciones `Make-Set`(x) son incluidas en el número total de operaciones m, entonces se tiene que $m \geq n$.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

En el análisis del tiempo de ejecución de las estructuras de conjuntos disyuntos se consideran 2 parámetros:

- n, el número de operaciones `Make-Set`(x) y
- m, el total de operaciones `Make-Set`(x), `Find-Set`(x) y `Union`(x, y).

Puesto que los conjuntos son disyuntos, el número máximo de operaciones `Union`(x, y) es $n - 1$.

Además, dado que las operaciones `Make-Set`(x) son incluidas en el número total de operaciones m, entonces se tiene que $m \geq n$.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos

En el análisis del tiempo de ejecución de las estructuras de conjuntos disyuntos se consideran 2 parámetros:

- n, el número de operaciones `Make-Set`(x) y
- m, el total de operaciones `Make-Set`(x), `Find-Set`(x) y `Union`(x, y).

Puesto que los conjuntos son disyuntos, el número máximo de operaciones `Union`(x, y) es $n - 1$.

Además, dado que las operaciones `Make-Set`(x) son incluidas en el número total de operaciones m, entonces se tiene que $m \geq n$.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos: Ejemplo de aplicación

- Una posible aplicación de las estructuras de conjuntos distyuntos es el problema de determinar los componentes conectados de un grafo no dirigido.
- El procedimiento `CONNECTED-COMPONENTS` usa las operaciones de conjuntos disyuntos para computar los componentos conexos de un grafo.
- Después de que `CONNECTED-COMPONENTS` ha preprocesado el grafo, el procedimiento `SAME-COMPONENT` determina si dos vértices están en el mismo componente conexo.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos: Ejemplo de aplicación

- Una posible aplicación de las estructuras de conjuntos distyuntos es el problema de determinar los componentes conectados de un grafo no dirigido.
- El procedimiento `CONNECTED-COMPONENTS` usa las operaciones de conjuntos disyuntos para computar los componentos conexos de un grafo.
- Después de que `CONNECTED-COMPONENTS` ha preprocesado el grafo, el procedimiento `SAME-COMPONENT` determina si dos vértices están en el mismo componente conexo.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos: Ejemplo de aplicación

- Una posible aplicación de las estructuras de conjuntos distyuntos es el problema de determinar los componentes conectados de un grafo no dirigido.
- El procedimiento `CONNECTED-COMPONENTS` usa las operaciones de conjuntos disyuntos para computar los componentos conexos de un grafo.
- Después de que `CONNECTED-COMPONENTS` ha preprocesado el grafo, el procedimiento `SAME-COMPONENT` determina si dos vértices están en el mismo componente conexo.

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos: Ejemplo de aplicación

Los procedimientos anteriores pueden ser escritos así:

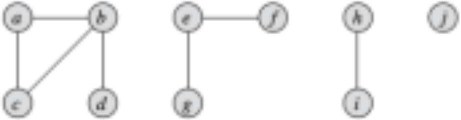
```
CONNECTED-COMPONENTS(G)
for each vertex v ∈ G.V
    MAKE-SET(v)
for each edge (u, v) ∈ G.E
    if FIND-SET(u) ≠ FIND-SET(v)
        UNION(u, v)
```

```
SAME-COMPONENT(u, v)
if FIND-SET(u) == FIND-SET(v)
    return TRUE
else return FALSE
```

Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos: Ejemplo de aplicación

Por ejemplo, el siguiente grafo tiene cuatro componentes conexos:



Estructuras de Datos

Estructuras de Datos para Conjuntos Disyuntos: Ejemplo de aplicación

La siguiente tabla ilustra como el procedimiento `CONNECTED-COMPONENTS` computa los conjuntons disyuntos:

Edge processed	Collection of disjoint sets									
initial sets	$\{a\}$	$\{b\}$	$\{c\}$	$\{d\}$	$\{e\}$	$\{f\}$	$\{g\}$	$\{h\}$	$\{i\}$	$\{j\}$
(b,d)	$\{a\}$	$\{b,d\}$	$\{c\}$		$\{e\}$	$\{f\}$	$\{g\}$	$\{h\}$	$\{i\}$	$\{j\}$
(e,g)	$\{a\}$	$\{b,d\}$	$\{c\}$		$\{e,g\}$	$\{f\}$		$\{h\}$	$\{i\}$	$\{j\}$
(a,c)	$\{a,c\}$	$\{b,d\}$			$\{e,g\}$	$\{f\}$		$\{h\}$	$\{i\}$	$\{j\}$
(h,i)	$\{a,c\}$	$\{b,d\}$			$\{e,g\}$	$\{f\}$		$\{h,i\}$		$\{j\}$
(a,b)	$\{a,b,c,d\}$				$\{e,g\}$	$\{f\}$		$\{h,i\}$		$\{j\}$
(e,f)	$\{a,b,c,d\}$				$\{e,f,g\}$			$\{h,i\}$		$\{j\}$
(b,c)	$\{a,b,c,d\}$				$\{e,f,g\}$			$\{h,i\}$		$\{j\}$

Plan

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Es posible implementar conjuntos disyuntos a través de listas enlazadas.
- De esta manera, cada conjunto es representado por su propia lista enlazada.
- El objeto para cada conjunto tiene atributos *head*, que apunta al primer objeto en la lista y *tail*, que apunta al último objeto.
- Cada objeto en la lista contiene un elemento, un puntero al siguiente objeto en la lista y un puntero al objeto del conjunto.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Es posible implementar conjuntos disyuntos a través de listas enlazadas.
- De esta manera, cada conjunto es representado por su propia lista enlazada.
- El objeto para cada conjunto tiene atributos *head*, que apunta al primer objeto en la lista y *tail*, que apunta al último objeto.
- Cada objeto en la lista contiene un elemento, un puntero al siguiente objeto en la lista y un puntero al objeto del conjunto.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Es posible implementar conjuntos disyuntos a través de listas enlazadas.
- De esta manera, cada conjunto es representado por su propia lista enlazada.
- El objeto para cada conjunto tiene atributos *head*, que apunta al primer objeto en la lista y *tail*, que apunta al último objeto.
- Cada objeto en la lista contiene un elemento, un puntero al siguiente objeto en la lista y un puntero al objeto del conjunto.

Estructuras de Datos para Conjuntos Disyuntos

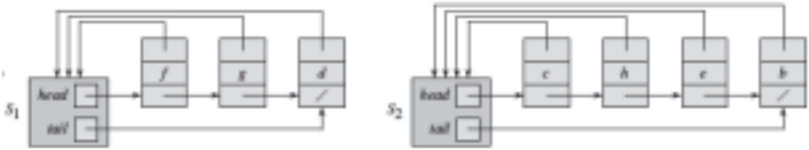
Representación con Listas Enlazadas

- En cada lista enlazada, los objetos pueden estar en cualquier orden.
- El elemento representativo es el elemento en el primero objeto en la lista.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

La siguiente es la representación con listas enlazadas de una estructura con dos conjuntos disyuntos:



Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Con esta representación, los procedimientos `Make-Set`(x) y `Find-Set`(x) son sencillos y requieren tiempo de ejecución $O(1)$.
- Para el procedimiento `Make-Set`(x), se crea una nueva lista enlazada cuyo único objeto es x.
- Para el procedimiento `Find-Set`(x), se utiliza el puntero de x al objeto del conjunto y luego se retorna el elemento que corresponde al atributo *head*.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Con esta representación, los procedimientos `Make-Set`(x) y `Find-Set`(x) son sencillos y requieren tiempo de ejecución $O(1)$.
- Para el procedimiento `Make-Set`(x), se crea una nueva lista enlazada cuyo único objeto es x.
- Para el procedimiento `Find-Set`(x), se utiliza el puntero de x al objeto del conjunto y luego se retorna el elemento que corresponde al atributo *head*.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Con esta representación, los procedimientos `Make-Set`(x) y `Find-Set`(x) son sencillos y requieren tiempo de ejecución $O(1)$.
- Para el procedimiento `Make-Set`(x), se crea una nueva lista enlazada cuyo único objeto es x.
- Para el procedimiento `Find-Set`(x), se utiliza el puntero de x al objeto del conjunto y luego se retorna el elemento que corresponde al atributo *head*.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Para el procedimiento `Union`(x, y), se concatena la lista correspondiente al conjunto de y al final de la lista correspondiente al conjunto de x.
- El elemento representativo del conjunto resultante es el elemento representativo de la lista correspondiente al conjunto de x.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Para el procedimiento `Union`(x, y), se concatena la lista correspondiente al conjunto de y al final de la lista correspondiente al conjunto de x.
- El elemento representativo del conjunto resultante es el elemento representativo de la lista correspondiente al conjunto de x.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Dado que todos los elementos de la lista correspondiente al conjunto de y son puestos en la lista asociada al conjunto de x, es posible destruir el objeto de la lista asociada a y.
- No obstante, es necesario actualizar el puntero al objeto del conjunto para cada uno de los elementos que originalmente estaban en la lista de y, lo cual toma un tiempo lineal con respecto al tamaño de la lista de y.

Estructuras de Datos para Conjuntos Disyuntos

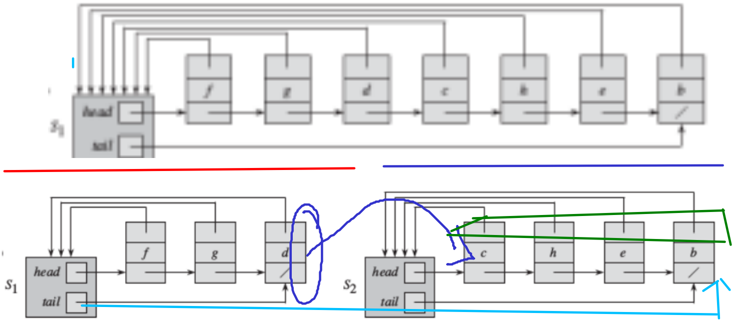
Representación con Listas Enlazadas

- Dado que todos los elementos de la lista correspondiente al conjunto de y son puestos en la lista asociada al conjunto de x, es posible destruir el objeto de la lista asociada a y.
- No obstante, es necesario actualizar el puntero al objeto del conjunto para cada uno de los elementos que originalmente estaban en la lista de y, lo cual toma un tiempo lineal con respecto al tamaño de la lista de y.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

La siguiente es la representación con listas enlazadas del conjunto resultante al unir los conjuntos S_1 y S_2 mencionados anteriorememte:



Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Para analizar esta implementación, consideramos una secuencia de m operaciones en n objetos.
- Vamos a suponer que se tienen n objetos $x_1, x_2, \ldots, x_n$.
- Si se ejecutan n operaciones `Make-Set` seguidas por $n-1$ operaciones `Union`, luego se tiene que $m = 2n - 1$.
- Las n operaciones `Make-Set` toman un tiempo $\Theta(n)$.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Para analizar esta implementación, consideramos una secuencia de m operaciones en n objetos.
- Vamos a suponer que se tienen n objetos $x_1, x_2, \ldots, x_n$.
- Si se ejecutan n operaciones `Make-Set` seguidas por $n - 1$ operaciones `Union`, luego se tiene que $m = 2n - 1$.
- Las n operaciones `Make-Set` toman un tiempo $\Theta(n)$.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Para analizar esta implementación, consideramos una secuencia de m operaciones en n objetos.
- Vamos a suponer que se tienen n objetos $x_1, x_2, \ldots, x_n$.
- Si se ejecutan n operaciones `Make-Set` seguidas por $n-1$ operaciones `Union`, luego se tiene que $m = 2n - 1$.
- Las n operaciones `Make-Set` toman un tiempo $\Theta(n)$.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

- Para analizar esta implementación, consideramos una secuencia de m operaciones en n objetos.
- Vamos a suponer que se tienen n objetos $x_1, x_2, \ldots, x_n$.
- Si se ejecutan n operaciones `Make-Set` seguidas por $n - 1$ operaciones `Union`, luego se tiene que $m = 2n - 1$.
- Las n operaciones `Make-Set` toman un tiempo $\Theta(n)$.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

Puesto que la i-esima operación `Union` actualiza i objetos, luego el número total de objetos actualizados por las $n-1$ operaciones `Union` es

$$\sum_{i=1}^{n-1} i = \Theta(n^2)$$

El número total de operaciones es $2n-1$ y de esta manera cada operación en promedio requiere un tiempo $\Theta(n)$. Esto es, el tiempo amortizado de una operación es $\Theta(n)$.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

Puesto que la i-esima operación `Union` actualiza i objetos, luego el número total de objetos actualizados por las $n-1$ operaciones `Union` es

$$\sum_{i=1}^{n-1} i = \Theta(n^2)$$

El número total de operaciones es $2n-1$ y de esta manera cada operación en promedio requiere un tiempo $\Theta(n)$. Esto es, el tiempo amortizado de una operación es $\Theta(n)$.

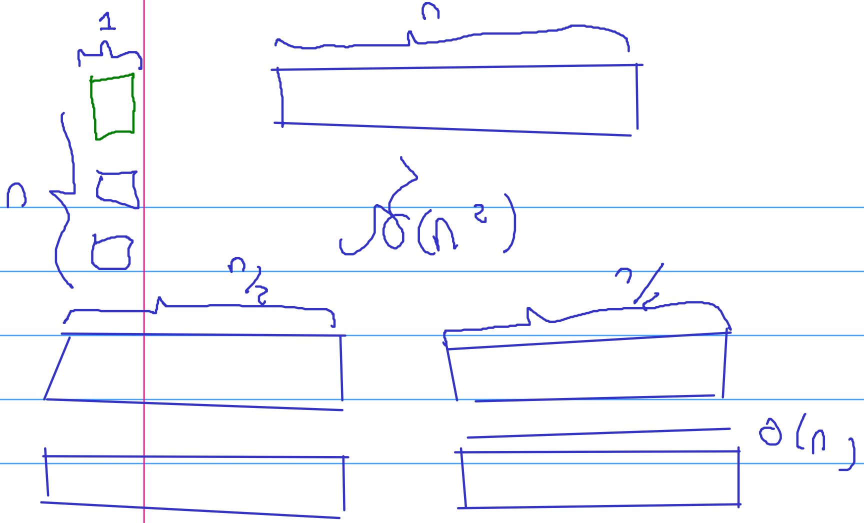
Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

Puesto que la i-esima operación `Union` actualiza i objetos, luego el número total de objetos actualizados por las $n-1$ operaciones `Union` es

$$\sum_{i=1}^{n-1} i = \Theta(n^2)$$

El número total de operaciones es $2n-1$ y de esta manera cada operación en promedio requiere un tiempo $\Theta(n)$. Esto es, el tiempo amortizado de una operación es $\Theta(n)$.



1
n
n
n/2
n/4
O(n)

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas: Heurística de Unión con peso

- De acuerdo a lo anterior, la implementación del procedimiento `Union` requiere en promedio $\Theta(n)$.
- Esto se debe a es posible que sea necesario concatenar una lista más grande con una más pequeña. De esta manera, se debe actualizar el puntero al objeto del conjunto para cada elemento de la lista mas grande.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas: Heurística de Unión con peso

- De acuerdo a lo anterior, la implementación del procedimiento `Union` requiere en promedio $\Theta(n)$.
- Esto se debe a es posible que sea necesario concatenar una lista más grande con una más pequeña. De esta manera, se debe actualizar el puntero al objeto del conjunto para cada elemento de la lista mas grande.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas: Heurística de Unión con peso

- Supongamos que en el objeto correspondiente a cada lista también se incluye el tamaño de la lista.
- Luego, siempre que se va a realizar una operación `Union`, se concatena la lista más pequeña en la lista más grande.
- Con esta heurística de unión con peso, cada operación `Union` puede tomar un tiempo $\Omega(n)$ si ambos conjuntos tienen $\Omega(n)$ elementos.
- Sin embargo, con esta mejora una secuencia de m operaciones `Make-Set`(x), `Find-Set`(x) y `Union` donde n operaciones son `Make-Set`(x), toma un tiempo $O(m + nlgn)$.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas: Heurística de Unión con peso

- Supongamos que en el objeto correspondiente a cada lista también se incluye el tamaño de la lista.
- Luego, siempre que se va a realizar una operación `Union`, se concatena la lista más pequeña en la lista más grande.
- Con esta heurística de unión con peso, cada operación `Union` puede tomar un tiempo $\Omega(n)$ si ambos conjuntos tienen $\Omega(n)$ elementos.
- Sin embargo, con esta mejora una secuencia de m operaciones `Make-Set`(x), `Find-Set`(x) y `Union` donde n operaciones son `Make-Set`(x), toma un tiempo $O(m + nlgn)$.

Estructuras de Datos para Conjuntos Disyuntos

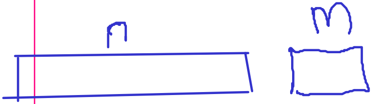
Representación con Listas Enlazadas: Heurística de Unión con peso

- Supongamos que en el objeto correspondiente a cada lista también se incluye el tamaño de la lista.
- Luego, siempre que se va a realizar una operación `Union`, se concatena la lista más pequeña en la lista más grande.
- Con esta heurística de unión con peso, cada operación `Union` puede tomar un tiempo $\Omega(n)$ si ambos conjuntos tienen $\Omega(n)$ elementos.
- Sin embargo, con esta mejora una secuencia de m operaciones `Make-Set`(x), `Find-Set`(x) y `Union` donde n operaciones son `Make-Set`(x), toma un tiempo $O(m + nlgn)$.

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas: Heurística de Unión con peso

- Supongamos que en el objeto correspondiente a cada lista también se incluye el tamaño de la lista.
- Luego, siempre que se va a realizar una operación `Union`, se concatena la lista más pequeña en la lista más grande.
- Con esta heurística de unión con peso, cada operación `Union` puede tomar un tiempo $\Omega(n)$ si ambos conjuntos tienen $\Omega(n)$ elementos.
- Sin embargo, con esta mejora una secuencia de m operaciones `Make-Set`(x), `Find-Set`(x) y `Union` donde n operaciones son `Make-Set`(x), toma un tiempo $O(m + nlgn)$.



1) Mejor caso tiene $m = 1$
2) Caso promedio $m = n/2$

$$T(n) = T(n/2) + O(1) = n\log(n)$$

Plan

Generalidades

Estructuras de Datos para Conjuntos Disyuntos

Representación con Listas Enlazadas

Árboles Binarios de Búsqueda

Estructuras de Datos

Árboles Binarios de Búsqueda

- Un árbol binario de búsqueda es un caso especial de árbol binario.
- Un árbol binario de búsuqeda puede ser representado como una estructura de datos enlazada en la cúal cada nodo es un objeto.
- Cada nodo contiene un dato y atributos *left*, *right* y *p*, que corresponden a punteros a los nodos hijos izquierdo y derecho y al nodo padre.
- Si un nodo no tiene padre o algún hijo, el atributo correspondiente contiene el valor NIL.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Un árbol binario de búsqueda es un caso especial de árbol binario.
- Un árbol binario de búsuqeda puede ser representado como una estructura de datos enlazada en la cúal cada nodo es un objeto.
- Cada nodo contiene un dato y atributos *left*, *right* y *p*, que corresponden a punteros a los nodos hijos izquierdo y derecho y al nodo padre.
- Si un nodo no tiene padre o algún hijo, el atributo correspondiente contiene el valor NIL.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Un árbol binario de búsqueda es un caso especial de árbol binario.
- Un árbol binario de búsuqeda puede ser representado como una estructura de datos enlazada en la cúal cada nodo es un objeto.
- Cada nodo contiene un dato y atributos *left*, *right* y *p*, que corresponden a punteros a los nodos hijos izquierdo y derecho y al nodo padre.
- Si un nodo no tiene padre o algún hijo, el atributo correspondiente contiene el valor NIL.

Estructuras de Datos

Árboles Binarios de Búsqueda

Las claves en un árbol binario de búsqueda son almacenadas de tal forma que siempre se satisface la siguiente propiedad:

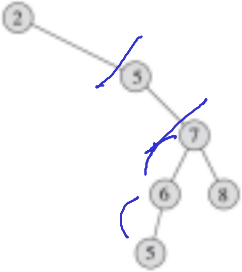
Sea x un nodo en un árbol binario de búsqueda. Si y es un nodo en el subárbol izquierdo de x entonces $y.key \leq x.key$. Si y es un nodo en el subárbol derecho de x, entonces $y.key \geq x.key$.

Estructuras de Datos

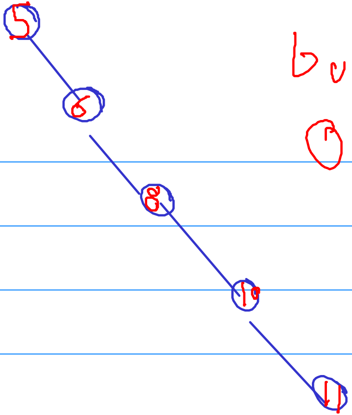
Árboles Binarios de Búsqueda

Los siguientes son dos ejemplos de árboles binarios de búsqueda que contienen los mismos datos:

2 5 5 6 7 8



2 5 5 6 7 8



buscar

$O(n)$

Estructuras de Datos

Árboles Binarios de Búsqueda

Es posible imprimir todos los elementos de un árbol binario de búsqueda mediante un recorrido in-orden,. El siguiente algoritmo implementa dicho recorrido:

```
INORDER-TREE-WALK(x)
if x ≠ NIL
    INORDER-TREE-WALK(x.left)
    print x.key
    INORDER-TREE-WALK(x.right)
```

Cúal es la complejidad de este algoritmo?

Estructuras de Datos

Árboles Binarios de Búsqueda

Es posible imprimir todos los elementos de un árbol binario de búsqueda mediante un recorrido in-orden,. El siguiente algoritmo implementa dicho recorrido:

```
INORDER-TREE-WALK(x)
if x ≠ NIL
    INORDER-TREE-WALK(x.left)
    print x.key
    INORDER-TREE-WALK(x.right)
```

Cúal es la complejidad de este algoritmo?

Estructuras de Datos

Árboles Binarios de Búsqueda

- Una operación frecuente en un árbol binario de búsqueda consiste en buscar un elemento.
- Además, los árboles binarios de búsqueda también soportan consultas como el mínimo o máximo elemento del árbol y como el sucesor o predecesor de un elemento dado.
- En general, estas operaciones se pueden realizar en tiempo $O(h)$ para cualquier árbol binario de búsqueda de altura h.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Una operación frecuente en un árbol binario de búsqueda consiste en buscar un elemento.
- Además, los árboles binarios de búsqueda también soportan consultas como el mínimo o máximo elemento del árbol y como el sucesor o predecesor de un elemento dado.
- En general, estas operaciones se pueden realizar en tiempo $O(h)$ para cualquier árbol binario de búsqueda de altura h.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Una operación frecuente en un árbol binario de búsqueda consiste en buscar un elemento.
- Además, los árboles binarios de búsqueda también soportan consultas como el mínimo o máximo elemento del árbol y como el sucesor o predecesor de un elemento dado.
- En general, estas operaciones se pueden realizar en tiempo $O(h)$ para cualquier árbol binario de búsqueda de altura h.

Estructuras de Datos

Árboles Binarios de Búsqueda

El siguiente procedimiento permite buscar un nodo en un árbol binario de búsqueda que tenga una clave dada.

```
TREE-SEARCH(x, k)
if x == NIL or k == x.key
    return x
if k < x.key
    return TREE-SEARCH(x.left, k)
else return TREE-SEARCH(x.right, k)
```

Dados un puntero a la raíz del árbol y una clave k, el procedimiento `TREE-SEARCH` retorna un puntero a un nodo con clave k si existe. De lo contrario, retorna NIL.

Estructuras de Datos

Árboles Binarios de Búsqueda

El siguiente procedimiento permite buscar un nodo en un árbol binario de búsqueda que tenga una clave dada.

```
TREE-SEARCH(x, k)
if x == NIL or k == x.key
    return x
if k < x.key
    return TREE-SEARCH(x.left, k)
else return TREE-SEARCH(x.right, k)
```

Dados un puntero a la raíz del árbol y una clave k, el procedimiento `TREE-SEARCH` retorna un puntero a un nodo con clave k si existe. De lo contrario, retorna NIL.

Estructuras de Datos

Árboles Binarios de Búsqueda

Es posible reescribir el anterior procedimiento de forma iterativa de la siguiente forma:

```
ITERATIVE-TREE-SEARCH(x, k)
while x ≠ NIL and k ≠ x.key
    if k < x.key
        x = x.left
    else x = x.right
return x
```

Estructuras de Datos

Árboles Binarios de Búsqueda

- Siempre es posible encontrar el mínimo elemento de un árbol binirio de búsqueda siguiendo los punteros al hijo izquierdo desde la raíz hasta encontrar un NIL.
- El procedimiento `TREE-MINIMUN` retorna un puntero al mínimo elemento de un árbol con raíz en el nodo x.
- El procedimiento `TREE-MAXIMUN` retorna un puntero al máximo elemento de un árbol con raíz en el nodo x.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Siempre es posible encontrar el mínimo elemento de un árbol binirio de búsqueda siguiendo los punteros al hijo izquierdo desde la raíz hasta encontrar un NIL.
- El procedimiento `TREE-MINIMUN` retorna un puntero al mínimo elemento de un árbol con raíz en el nodo x.
- El procedimiento `TREE-MAXIMUN` retorna un puntero al máximo elemento de un árbol con raíz en el nodo x.

Estructuras de Datos

Árboles Binarios de Búsqueda

El procedimiento `TREE-MINIMUN` es definido como sigue:

```
TREE-MINIMUM(x)
while x.left ≠ NIL
      x = x.left
return x
```

Estructuras de Datos

Árboles Binarios de Búsqueda

El procedimiento `TREE-MAXIMUN` es definido como sigue:

```
TREE-MAXIMUM(x)
1  while x.right ≠ NIL
2      x = x.right
3  return x
```

Estructuras de Datos

Árboles Binarios de Búsqueda

- En algunas aplicaciones, es necesario encontrar el nodo sucesor de otro de acuerdo al orden determinado por un recorrido in-orden en un árbol binario de búsqueda.
- Si todas las claves son diferentes, el sucesor de un nodo x es el nodo con la clave más pequeña que x.
- La estructura de los árboles binarios de búsqueda nos permite determinar el sucesor de un nodo sin necesidad de comparar las claves.

Estructuras de Datos

Árboles Binarios de Búsqueda

- En algunas aplicaciones, es necesario encontrar el nodo sucesor de otro de acuerdo al orden determinado por un recorrido in-orden en un árbol binario de búsqueda.
- Si todas las claves son diferentes, el sucesor de un nodo x es el nodo con la clave más pequeña que x.
- La estructura de los árboles binarios de búsqueda nos permite determinar el sucesor de un nodo sin necesidad de comparar las claves.

Estructuras de Datos

Árboles Binarios de Búsqueda

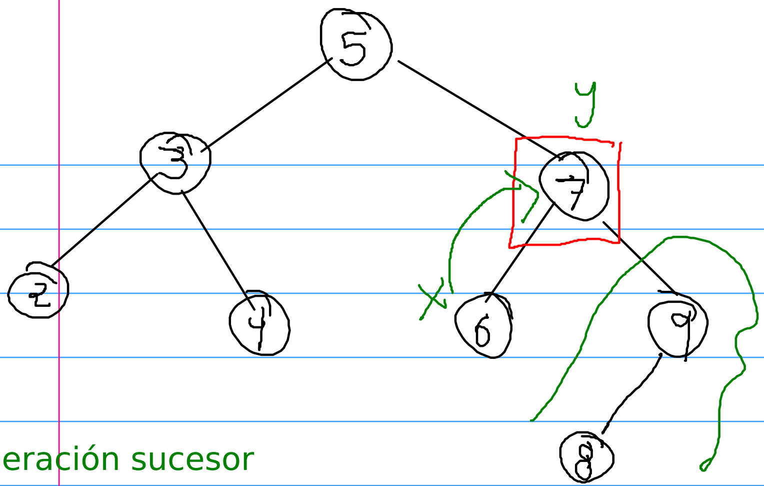
- En algunas aplicaciones, es necesario encontrar el nodo sucesor de otro de acuerdo al orden determinado por un recorrido in-orden en un árbol binario de búsqueda.
- Si todas las claves son diferentes, el sucesor de un nodo x es el nodo con la clave más pequeña que x.
- La estructura de los árboles binarios de búsqueda nos permite determinar el sucesor de un nodo sin necesidad de comparar las claves.

Estructuras de Datos

Árboles Binarios de Búsqueda

El siguiente procedimiento retorna el sucesor de un nodo x en un árbol binario de búsqueda en caso de que exista y NIL si x tiene la clave más grande en el árbol.

```
TREE-SUCCESSOR(x)
if x.right ≠ NIL
    return TREE-MINIMUM(x.right)
y = x.p
while y ≠ NIL and x == y.right
    x = y
    y = y.p
return y
```



Operación sucesor

1) Si el hijo derecho no es NIL, entonces
busco el minimo del subarbol derecho de x
2) Si el hijo derecho es NIL, (Y = P.X) itero hasta que
X NO sea el hijo derecho de Y

Estructuras de Datos

Ejercicio: Árboles Binarios de Búsqueda

Escriba el pseudocódigo del procedimiento `TREE-PREDECESSOR` que recibe un nodo de un árbol binario de búsqueda y retorna el elemento que lo precede.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Las operaciones de inserción y eliminación de un nodo ocasionan que el conjunto dinámico representado por el árbol binario de búsqueda cambie.
- La estructura de datos debe ser modificada para reflejar dicho cambio, pero dicha modificación debe realizarse de tal manera que se preserve la propiedad de los árboles binarios de búsqueda.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Para insertar un nuevo valor en un árbol binario de búsqueda, se utiliza el procedimiento `TREE-INSERT`.
- Este procedimiento recibe un nodo z para el cual $z.key = v$, $z.left = NIL$ y $z.right = NIL$.
- Luego, modifica T y algunos de los atributos de z de tal manera que inserta z en una posición apropiada en el árbol.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Para insertar un nuevo valor en un árbol binario de búsqueda, se utiliza el procedimiento `TREE-INSERT`.
- Este procedimiento recibe un nodo z para el cual $z.key = v$, $z.left = NIL$ y $z.right = NIL$.
- Luego, modifica T y algunos de los atributos de z de tal manera que inserta z en una posición apropiada en el árbol.

Estructuras de Datos

Árboles Binarios de Búsqueda

- Para insertar un nuevo valor en un árbol binario de búsqueda, se utiliza el procedimiento `TREE-INSERT`.
- Este procedimiento recibe un nodo z para el cual $z.key = v$, $z.left = NIL$ y $z.right = NIL$.
- Luego, modifica T y algunos de los atributos de z de tal manera que inserta z en una posición apropiada en el árbol.

Estructuras de Datos

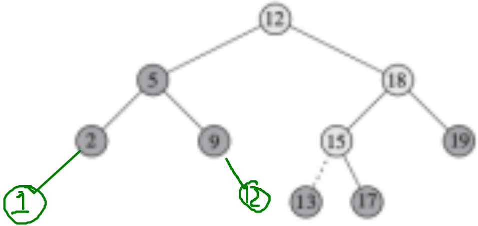
Árboles Binarios de Búsqueda

La siguiente es la definición del procedimiento `TREE-INSERT`:

```
TREE-INSERT(T, z)
y = NIL
x = T.root
while x ≠ NIL
    y = x
    if z.key < x.key
        x = x.left
    else x = x.right
z.p = y
if y == NIL
    T.root = z          // tree T was empty
elseif z.key < y.key
    y.left = z
else y.right = z
```

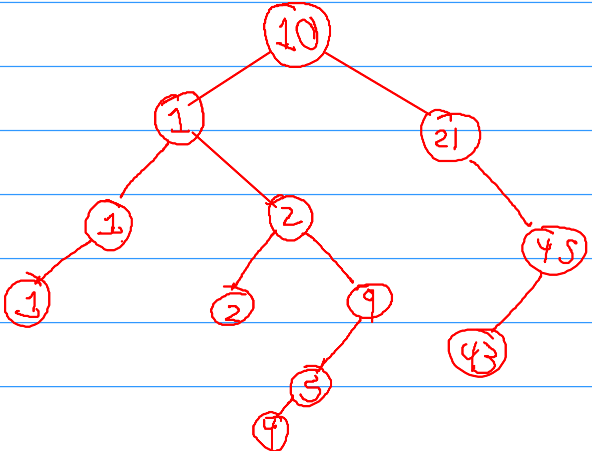
Estructuras de Datos

Árboles Binarios de Búsqueda



Crear un arbol Binario de Búsqueda, operaciones de inserción

{10,1,2,1,9,1,2,21,5,45,43,4}



Estructuras de Datos

Árboles Binarios de Búsqueda

La estrategia global para borrar un nodo z de un árbol binario de búsqueda T tiene 3 casos base:

- Si z no tiene hijos, entonces z es removido modificando su nodo padre con NIL como su hijo.
- Si z tiene un solo nodo hijo, entonces este nodo hijo es colocado en la posición de z colocando dicho nodo como hijo del padre de z.
- Si z tiene dos hijos, entonces es necesario encontrar el sucesor y de z, el cual debe estar en el subarbol derecho de z. Luego, y debe tomar la posición de z en el árbol. El resto del subárbol derecho original de z pasa a ser el subárbol derecho de y y el subárbol izquierdo de z pasa a ser el subárbol izquierdo de y.

Estructuras de Datos

Árboles Binarios de Búsqueda

La estrategia global para borrar un nodo z de un árbol binario de búsqueda T tiene 3 casos base:

- Si z no tiene hijos, entonces z es removido modificando su nodo padre con NIL como su hijo.
- Si z tiene un solo nodo hijo, entonces este nodo hijo es colocado en la posición de z colocando dicho nodo como hijo del padre de z.
- Si z tiene dos hijos, entonces es necesario encontrar el sucesor y de z, el cual debe estar en el subarbol derecho de z. Luego, y debe tomar la posición de z en el árbol. El resto del subárbol derecho original de z pasa a ser el subárbol derecho de y y el subárbol izquierdo de z pasa a ser el subárbol izquierdo de y.

Estructuras de Datos

Árboles Binarios de Búsqueda

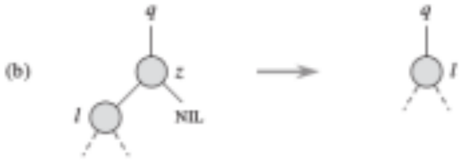
Los anteriores tres casos corresponden a las siguientes cuatro situaciones, donde se requiere eliminar el nodo z:

(a)

Estructuras de Datos

Árboles Binarios de Búsqueda

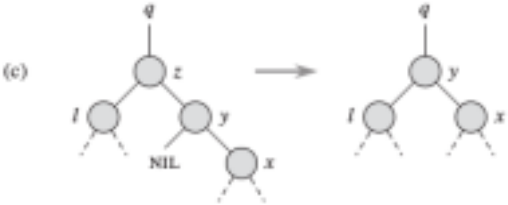
La situación (a) y la situación (b) corresponden a los casos donde el nodo a eliminar (z) no tiene nodos hijos o solamente tiene un nodo hijo.



Estructuras de Datos

Árboles Binarios de Búsqueda

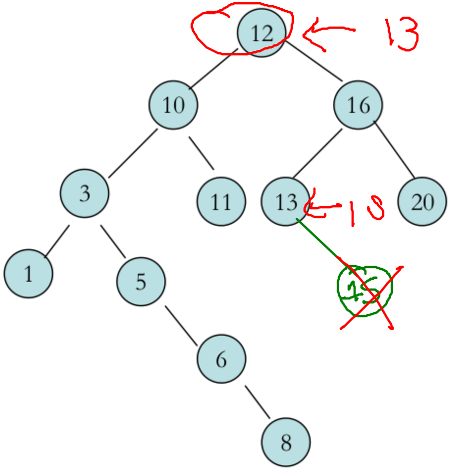
La situación (c) corresponde al caso donde el nodo z tiene dos nodos hijos y donde su sucesor es su hijo derecho.



Estructuras de Datos

Árboles Binarios de Búsqueda

La situación (d) corresponde al caso donde el nodo z tiene dos nodos hijos pero su nodo sucesor es diferente a su hijo derecho.


12
13
10
16
3
11
13
13
20
1
5
15
6
8

Estructuras de Datos

Árboles Binarios de Búsqueda

Para mover subárboles dentro del árbol binario de búsqueda de acuerdo a los casos mencionados anteriormente, se define el procedimiento `TRANSPLANT`, el cual reemplaza un subárbol u como hijo de su nodo padre por otro subárbol v.

```
TRANSPLANT(T, u, v)
if u.p == NIL
    T.root = v
elseif u == u.p.left
    u.p.left = v
else u.p.right = v
if v ≠ NIL
    v.p = u.p
```

1) Pilas: LIFO. Push y Pop --> O(1)

2) Colas: FIFO: Queue, EnQueue --> O(1)

3) Listas enlazadas

    3.1) Inserción de un sólo elemento a la derecha O(1) Mejor caso

    3.2) Inserción de un elemento (variable) caso promedio O(nlog(n))

    3.3) Inserción de igual o mayor, Peor caso O(n^2)

    3.4) Búsqueda: Empezo por el elemento que está apunto por head, y empezo a buscar. En el peor caso tiene O(n), caso promedio suponemos que está en la mitad O(n), mejor caso O(1)

Arboles binarios de búsqueda

1) Caracteristica: Dado nodo un X, el sub arbol izquierda contiene menores o iguales y el sub-arbol derecho mayores iguales. (EN caso igual se puede ir por cualquier lado)

2) El recorriendo inorden (izq raiz der) da una lista ordenada

3) Complejidad operaciones

1) Buscar: O(h) donde h es la altura del arbol
Peor caso: O(n), mejor caso está balanceado O(log(n))

2) Maximo y minimo: O(h), ir a la mayor profundidad izquierda o derecha.

3) Sucesor: O(h) peor caso (buscar sucesor del máximo)

4) Inserción: O(h) cuando se intentar insertar un número más grande el máximo o más pequeño que el mínimo

5) Eliminación: O(h) sucesor (caso 3)

Descripción de las operaciones

1) Insertar: Preguntar si es menor o igual O es mayor si es mayor busco en el subarbol derecho si no es NIL si NIL inserto como hijo derecho, de forma análoga en el izquierdo

2) Eliminar

Caso 1: Nodo a eliminar es hoja, lo borramos

Caso 2: Nodo a eliminar tiene un hijo, lo reemplazamos por el hijo

Caso 3: Nodo a eliminar tiene dos hijos, lo reemplazados por su sucesor y eliminamos el sucesor

Estructuras de Datos

Árboles Binarios de Búsqueda

Luego, tenemos que el procedimiento para borrar un nodo de un árbol binario de búsqueda es el siguiente:

```
TREE-DELETE(T, z)
if z.left == NIL
    TRANSPLANT(T, z, z.right)
elseif z.right == NIL
    TRANSPLANT(T, z, z.left)
else y = TREE-MINIMUM(z.right)
    if y.p ≠ z
        TRANSPLANT(T, y, y.right)
        y.right = z.right
        y.right.p = y
    TRANSPLANT(T, z, y)
    y.left = z.left
    y.left.p = y
```

Preguntas

?