Union Find Disjoint Set

FACUNDO GALÁN

15/10/2015

Definición

- Union Find Disjoint Sets

  UFDS es una estructura de datos que maneja conjuntos disjuntos (conjuntos que no comparten elementos), y permite realizar las operaciones de búsqueda (a qué conjunto pertenece un elemento dado) y unión (de dos conjuntos en uno más grande) de forma eficiente ≈ O(1).

Motivación

- El UFDS puede ser usado para resolver problemas como el de encontrar las componentes conexas de un grafo no dirigido, o para saber si un vértice está en la misma componente que otro, etc.

- ¿Se les ocurre algún algoritmo clásico de grafos que lo use?

Funcionamiento

- La estrategia que sugiere la estructura es la de escoger un elemento de cada conjunto, como representante del mismo.
- Para esto, el UFDS crea un árbol para cada conjunto disjunto, formando éstos un bosque, donde la raíz de cada árbol, simboliza el elemento identificador de un conjunto.

Funcionamiento

- De esta manera, determinar si dos elementos pertenecen al mismo conjunto, es comparar la raíz del árbol que contiene a cada uno, verificando si son iguales (pertenecen al mismo conjunto) o diferentes (pertenecen a diferentes conjuntos).
- Esto se puede codificar como un arreglo de padres, donde cada elemento indica su padre, e inicialmente cada elemento apunta a sí mismo.

Funcionamiento

- Con esta estructura, podemos realizar la operación de unión de conjuntos en O(1), ya que sólo debemos cambiar el padre de una de las dos raíces de los conjuntos que queremos unir.

- ¿Problema?

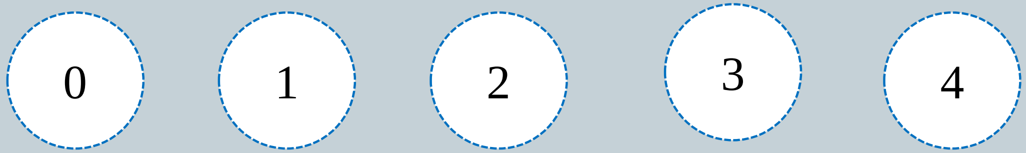
  La búsqueda sigue siendo de O(n) en el peor de los casos.

Funcionamiento

- Para optimizar la búsqueda, se le realizan dos modificaciones a la estructura: (1) se le agrega un arreglo extra, que lleva (el límite superior de) la altura del nodo raíz de cada conjunto (‘union by rank’); (2) se modifica la operación de búsqueda, para que al retornar su raíz, se asigne como nuevo padre a su raíz, logrando en la próxima operación devolverlo directamente (‘path compression’).

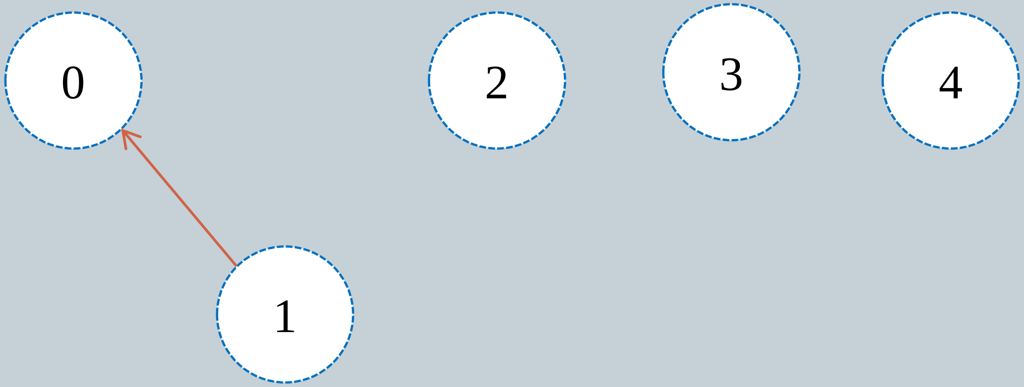
Ejemplo (0)

- N = 5 , siendo N la cantidad de nodos.
- int p[5] = {0, 1, 2, 3, 4};
- int rank[5] = {0, 0, 0, 0, 0};



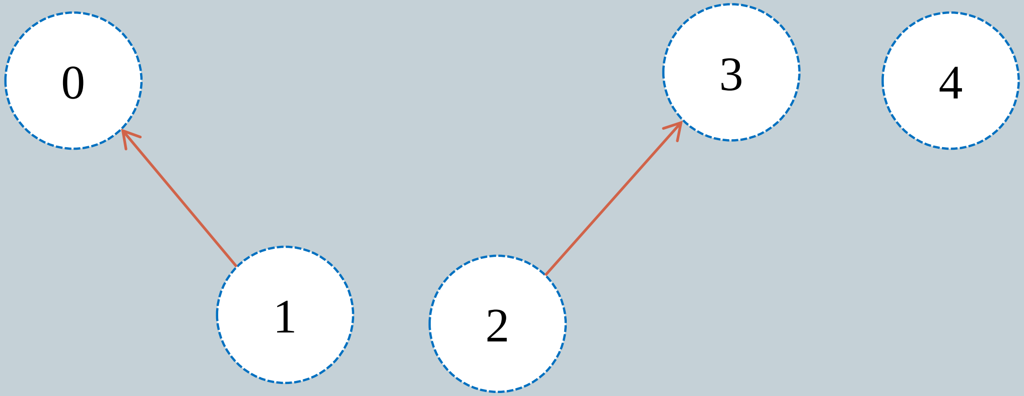
Ejemplo (1)

- Operación: ‘unión de 0 y 1’.
- int p[5] = {0, 0, 2, 3, 4};
- int rank[5] = {1, 0, 0, 0, 0};



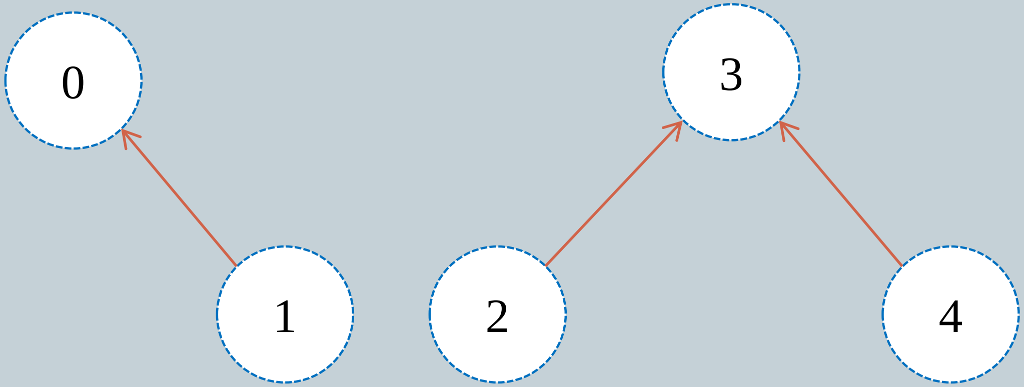
Ejemplo (2)

- Operación: ‘unión de 3 y 2’.
- int p[5] = {0, 0, 3, 3, 4};
- int rank[5] = {1, 0, 0, 1, 0};



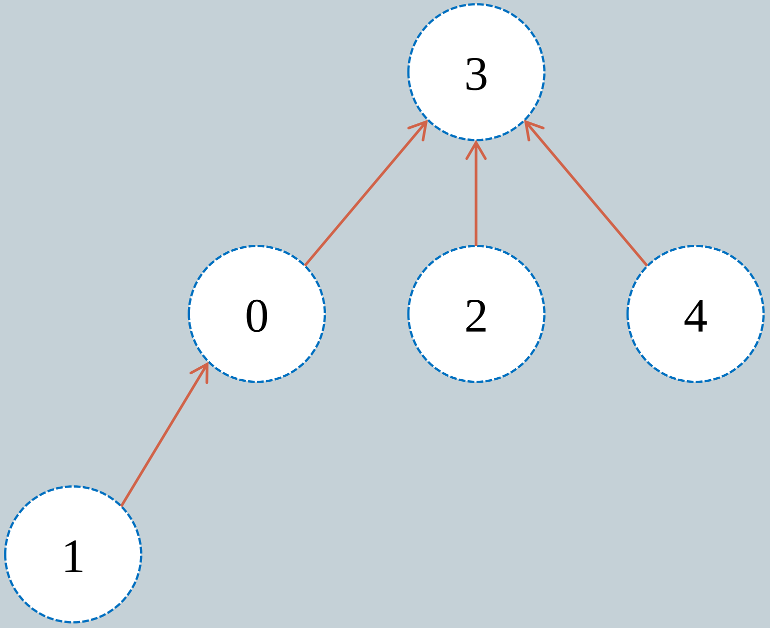
Ejemplo (3)

- Operación: ‘unión de 4 y 3’.
- int p[5] = {0, 0, 3, 3, 3};
- int rank[5] = {1, 0, 0, 1, 0};



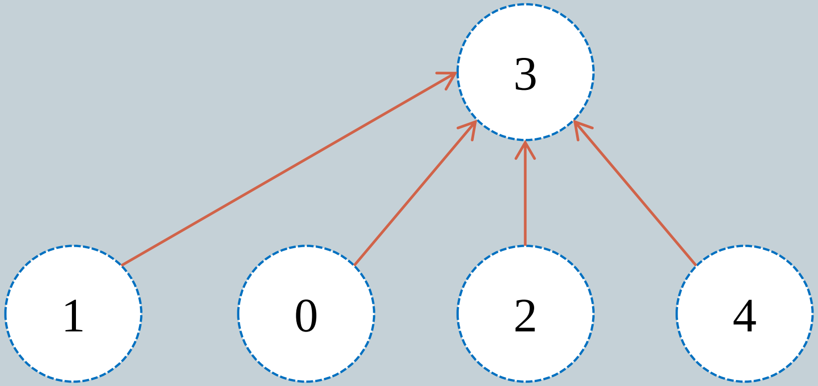
Ejemplo (4)

- Operación: ‘unión de 3 y 0’.
- int p[5] = {3, 0, 3, 3, 3};
- int rank[5] = {1, 0, 0, 2, 0};



Ejemplo (5)

- Operación: ‘el elemento 1 pertenece al mismo conjunto que el elemento 4?’.
- int p[5] = {3, 3, 3, 3, 3};
- int rank[5] = {1, 0, 0, 2, 0};



Tarea

- Agregar al código de la estructura dada por la cátedra, lo que sea necesario para contestar (de forma eficiente) consultas del tipo: '¿cuántos conjuntos hay hasta ahora?', y '¿cuántos elementos contiene el conjunto al que pertenece el elemento X?'.