INF221 – Algoritmos y Complejidad

Clase #24
Algoritmo de Kruskal II

Aldo Berrios Valenzuela

Martes 8 de noviembre de 2016

1. Algoritmo de Kruskal (continuación)

Recordamos algunas observaciones de la clase pasada.

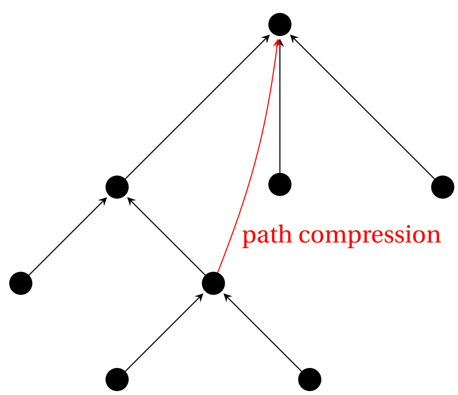

Figura 1: En path compression cambia el puntero al padre original de un nodo por la raíz del árbol más grande (véase el arco rojo).

- Si el nodo es raíz y tiene rank r, su árbol tiene a lo menos 2^r nodos. (si el nodo es una raíz, entonces el número de descendientes es 2^r)
- A lo largo de un camino a la raíz, rank aumenta.

Definición 1.1. Sea $g : \mathbb{R} \to \mathbb{R}$ tal que para $x > 1$, $g(x) < x$. Definimos:

$$g^*(x) = \begin{cases} 0 & x \leq 1 \\ 1 + g^*\left(g(x)\right) & x > 1 \end{cases} \tag{1.1}$$

En el fondo, g^* nos permite obtener cuántas veces debo aplicar la función g para obtener un resultado menor o igual a 1.

Análisis clásico (Hopcroft y Ullman, Tarjan) es lioso... Seidel y Sharir dan un análisis simple (para un valor adecuado de "simple")

Idea general:

- Reordenar operaciones para simplificar análisis. Todos los *unión* al comienzo, todos los *final* después ⇒ ya no llegamos a la raíz, nueva operación *compress* que toma un rango de nodos.

  El costo de la secuencia no cambia. Solo durante los *union* cambia el rank.

- Dividir el bosque "alto" (rank alto) y "bajo" (rank bajo), analizar por separado.

  Para impedir que se mezcle "la chusma", operación *shatter*: Los nodos en el camino quedan de raíces.

  Idea:

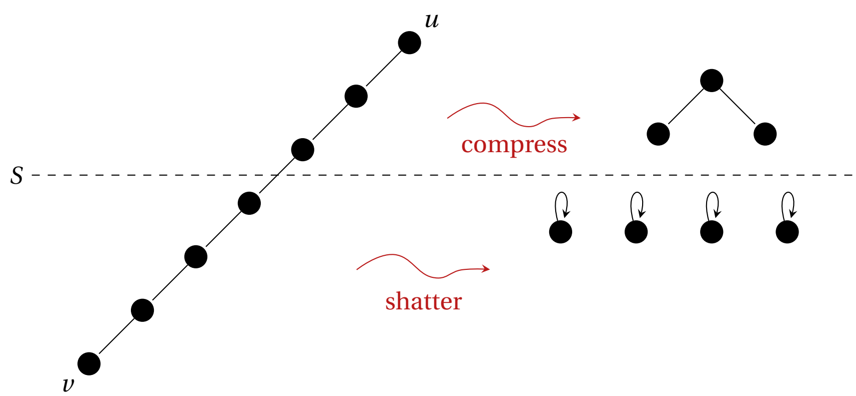

Figura 2: Todos los nodos cuyo rango es menor o igual que S, se les aplica la operación *shatter*. Por el contrario, aquellos que sean mayores se les aplica *compress*. Del bosque total $\mathscr{F}$, se originan dos bosques $\mathscr{F}_-$ y $\mathscr{F}_+$ respectivamente.

Sea $T(m,n,r)$ el costo máximo en m operaciones compress en un bosque de n nodos con rank a lo más r. Una cota trivial es el teorema 1.1.

Teorema 1.1. $T(m,n,r) \le nr$.

Demostración. Cada nodo puede cambiar a lo más r veces de padre. □

Sea $\mathscr{F}$ un bosque de rank máximo r, con n nodos, C una secuencia de m operaciones compress, $T(\mathscr{F},C)$ el número total de asignaciones *parent* al aplicar C sobre $\mathscr{F}$. Dividimos $\mathscr{F}$ en $\mathscr{F}_-$ (con nodos de rank $\le s$) y $\mathscr{F}_+$ (nodos de rank $> s$). Como rank aumenta al seguir punteros a padres, el padre de un nodo alto es a su vez alto. La secuencia C sobre $\mathscr{F}$ puede descomponerse en secuencias C_+ sobre $\mathscr{F}_+$ y C_- sobre $\mathscr{F}_-$, del mismo costo total, usando:

```
procedure compress(u, v, F)
  if rank[u]>s then
    compress(u, v, F+)
  else if rank[v]≤s then
    compress(u, v, F-)
  else
    z←u
    while rank[parent_F[z]]≤s do
      z← parent[z]
    end
    compress(parent_F[z], v, F+)
    shatter(u, z, F-)
    parent[z] ← z
```

En la figura 2 partimos el algoritmo desde el nodo u y comprimimos hasta el nodo v.