INF221 – Algoritmos y Complejidad

Clase #23
Algoritmo de Kruskal

Aldo Berrios Valenzuela

2 de noviembre de 2016

1. Algoritmo de Kruskal

Dado un grafo conexo $G = (V, E)$ con rótulos $\omega : E \to \mathbb{R}^+$ /* rotulamos los arcos, a cada arco le estamos asignando un peso) */, buscamos el árbol recubridor mínimo (mínima suma de los rótulos de los arcos). Minimal Spanning Tree, MST.

El algoritmo de kruskal es:

```
Ordenar $E$ en orden de costo creciente
$S \leftarrow \emptyset$
for $v \in V$ do
  Agregar {$v$} a $S$
end
$T \leftarrow \emptyset$
for $uv\in E$ do  /* En orden! */
  if $u$ y $v$ no pertenecen al mismo conjunto en $S$ then
    Agregar $uv$ a $T$
    Unir los conjuntos de $u$ y $v$ en $S$
  end
end
return ($V, T$)
```

Central: Manipulación de clases de equivalencia (reflexiva, transitiva, simétrica), conjunto S.

/* Grafo de pierola */

/* En el fondo el grafo resultante aparece en una clase de equivalencia o algo así (cambiar redacción) */

/* otro dibujo con el grafo minimo */

Tres operaciones centrales:

- Inicializar
- Find
- Unión

Idea:

- Representar la clase mediante un elemento representante.
- Un arreglo `parent`$[v]$: padre de v. /* en lugar de crear nodos con punteros al hijo, estos nodos tienen puntero al padre */

O sea:

/* Figura 1 apunte HvB: Clase 23 */

Find: Seguir punteros a padres hasta bucle → representante.

Unión: Hallar representantes (si no vienen dados), colgar el árbol menor del mayor (no aumente la altura).

Agregar rank[v]: Altura del árbol con raíz en v.

Algoritmos:

```
Make Set($V$)
  for $v \in V$ do
    parent[$v$] $\leftarrow v$
    rank[$v$] $\leftarrow 0$
  end
```

```
find ($u$)
  while $u \ne $parent[$u$] do
    $u \leftarrow $ parent[$u$]
  end
  return $u$

union ($u, v$)  /* representantes */
  if rank[$u$] $<$ rank[$v$] then
    parent [$u$] $\leftarrow v$
  else
    parent[$v$] $\leftarrow u$
    if rank[$u$] = rank[$v$] then
      rank[$u$] $\leftarrow $ rank [$u$] + 1
    end
  end
```

¿Nº mínimo de nodos si rank $= r$ de la raíz? /* colocar los dibujos asociados para cada valor de r */

- $r = 0$
- $r = 1$
- $r = 2$
- $r = 3$

Si el rank $= r$, entonces se observa que el algoritmo demora 2^r (y puede lograrse, ← son *árboles binomiales*). Si B_0 es un nodo, entonces B_r es unir dos B_{r-1}. Por lo tanto, camino a la raíz de un árbol de rank r es a lo más r, si tiene n nodos es a lo más $\log_2 n$.

Algoritmo de Kruskal: A lo más $2|E|$ find $\rightsquigarrow |V|-1$.

$$O\big(|V| + 2|E|\log_2|V|\big) = O\big(|E|\log|V|\big) \tag{1.1}$$

1.1. Path Compression

/* Dibujo: después de hacer un find hacemos un path compression */

Algoritmo path compression:

```
find ($v$)
  $u \leftarrow v$;
  while $u \ne $parent[$u$] do
    $u \leftarrow$ parent[$u$]
  end
  while $v \ne$ parent[$v$] do
```

```
    $x \leftarrow$ parent[$v$]
    parent[$v$] $\leftarrow u$
    $v \leftarrow x$
  end
```

Un programa un poco más simple:

```
while $u \ne parent[$u$] do$
  parent[$u$] $\leftarrow$ parent[parent[$u$]]
  $u \leftarrow$ parent[$u$]
end
```