Table of Contents

Section outline

1. **Disjoint sets**
   - Notion of disjoint sets
   - Kruskal's algorithm with DSUF
   - Disjoint set data structure
   - The findSet operation
   - The unionSet operation
   - Size guided unionSet operation
   - Rank guided unionSet operation
   - Properties of linking by rank
   - Union with path compression
   - Grouping of ranks by the iterated logarithm
   - Disbursing credits to drive findSet
   - Amortised costing of find with path compression
   - Complexity of Kruskal's algorithm with DSUF

Notion of disjoint sets

- Consider a universe $U = \{S_1, S_2, \ldots, S_n\}$
- Always $S_i, S_j \in U \Rightarrow S_i \cap S_j = \phi$
- If union of $S_i \in U$ and $S_j \in U$ is carried out, U is updated as $U \leftarrow (U \setminus \{S_1, S_2\}) \cup \{S_1 \cup S_2\}$
- This ensures that all members of U are still disjoint
- A set S in U may be identified by some $x \in S$, the cannonical element

  Eg. $S = \{a, b, c\}$ may be identified by a
- Operations on disjoint sets (DSUF) are as follows:

  makeSet(x) creates singleton set $\{x\}$ and adds it to U
  unionSet(x, y) Let $x \in S_i$ and $y \in S_j$; $U \leftarrow \left(U \setminus \{S_i, S_j\}\right) \cup \{S_i \cup S_j\}$
  findSet(x) returns y, $x, y \in S$ and y is the cannonical member of S
- The sets S_i are a partition of $\bigcup_i S_i$ and are induced equivalent classes
- DSUF data structure introduced by Bernard A Gallaer and Michael J Fisher in 1964

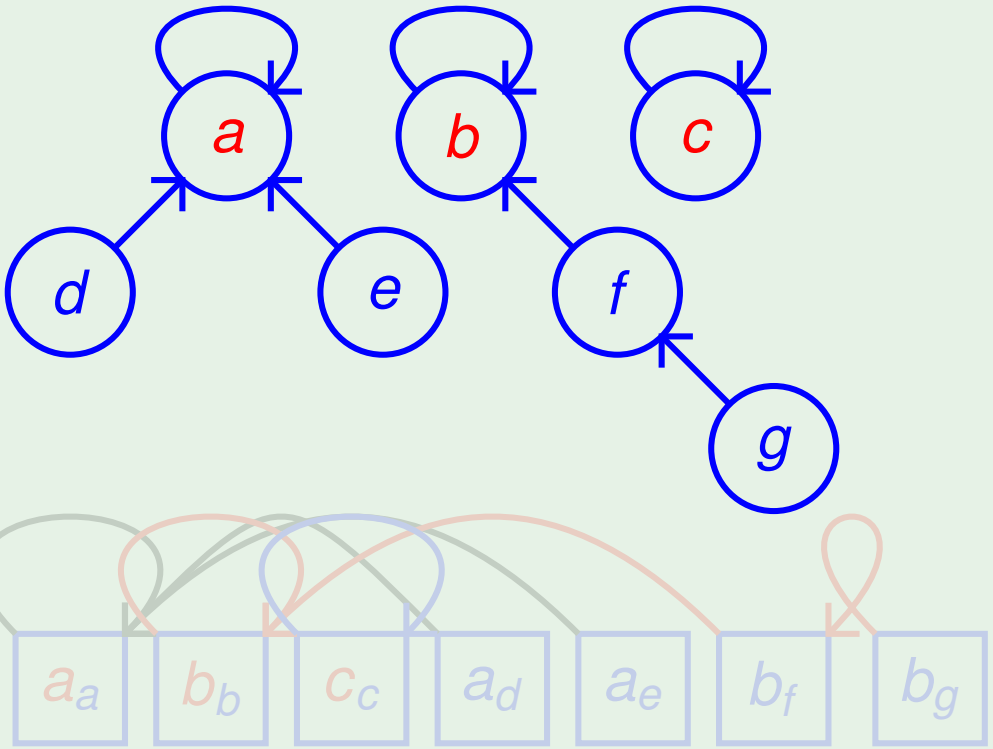
Kruskal's algorithm with DSUF

1. build a min heap H of the edges
2. initialise $T \leftarrow \phi$; edge count $c \leftarrow 0$
3. foreach $v \in V$ makeSet(v)
4. while ($c < |V| - 1$) do
5. extract min cost edge $e = \langle v_i, v_j \rangle \in H$
6. if (findSet(v_i)$\neq$findSet(v_j))
   // no cycle formed by adding e to T
7. add e to T; $c \leftarrow c + 1$
8. unionSet(v_i, v_j)
   // track the two components joining
9. done

Disjoint set data structure

- Array dsuf[] holds the index of the parent element: pI = dsuf[xI]
- For the cannonical element: dsuf[xI] == xI

Example (Tree representation of sets)

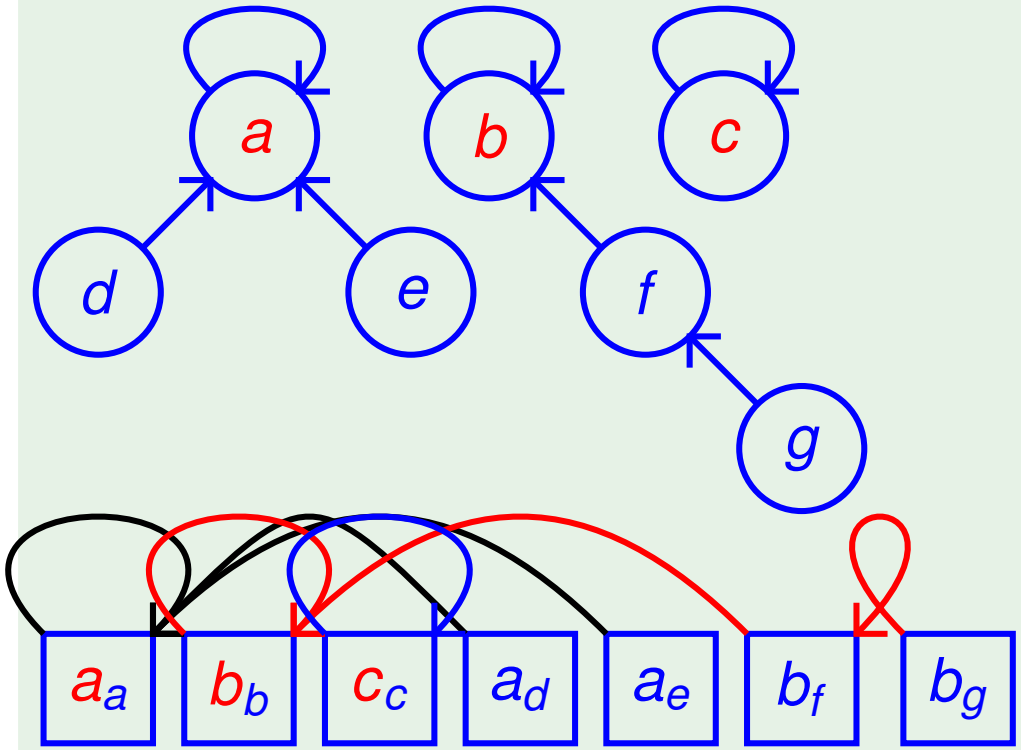


- $U = \{\{a, d, e\}, \{b, f\}, \{c\}\}$ may be represented as shown
- Elements in a set form chains leading to the representative element of the set
- The cannonical element of a set points to itself
- This structure may be implemented through arrays

Disjoint set data structure

- Array dsuf[] holds the index of the parent element: pI = dsuf[xI]
- For the cannonical element: dsuf[xI] == xI

Example (Tree representation of sets)



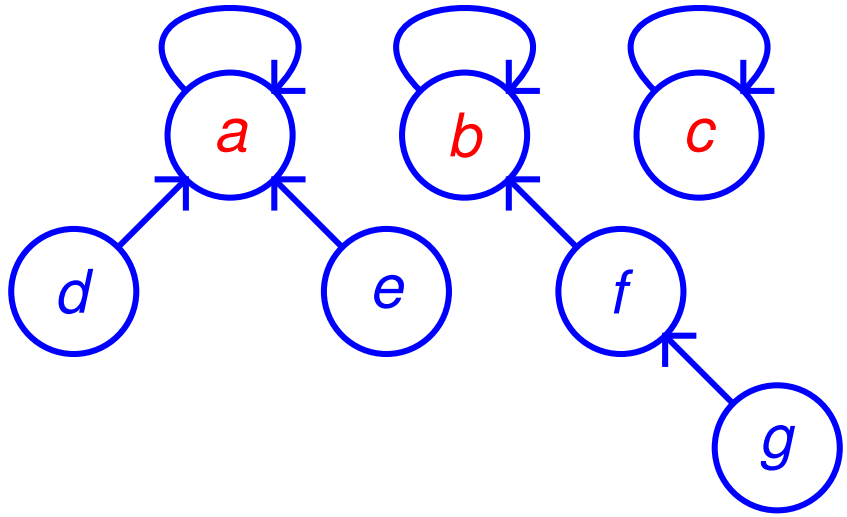
- $U = \{\{a, d, e\}, \{b, f\}, \{c\}\}$ may be represented as shown
- Elements in a set form chains leading to the representative element of the set
- The cannonical element of a set points to itself
- This structure may be implemented through arrays

The findSet operation

Basic mechanism

findSet(xl, dsuf[])

1. pl = dsuf[xl]
2. while (pl≠xl) do
3. xl=pl
4. pl=dsuf[xl]
5. done



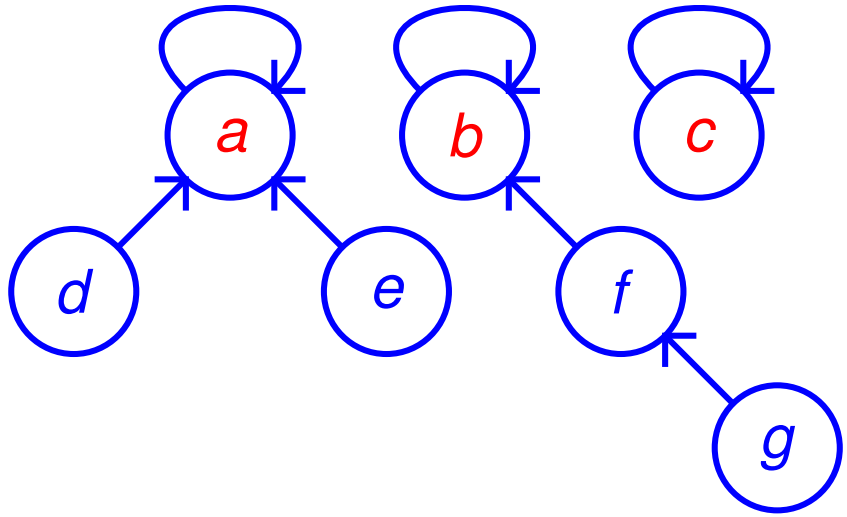
- Worst case running time is determined by length of the longest chain
- Chains can grow long, making findSet() inefficient
- Long chain can be collapsed during a search

The findSet operation

Basic mechanism

findSet(xl, dsuf[])

1. pl = dsuf[xl]
2. while (pl≠xl) do
3. xl=pl
4. pl=dsuf[xl]
5. done



- Worst case running time is determined by length of the longest chain
- Chains can grow long, making findSet() inefficient
- Long chain can be collapsed during a search

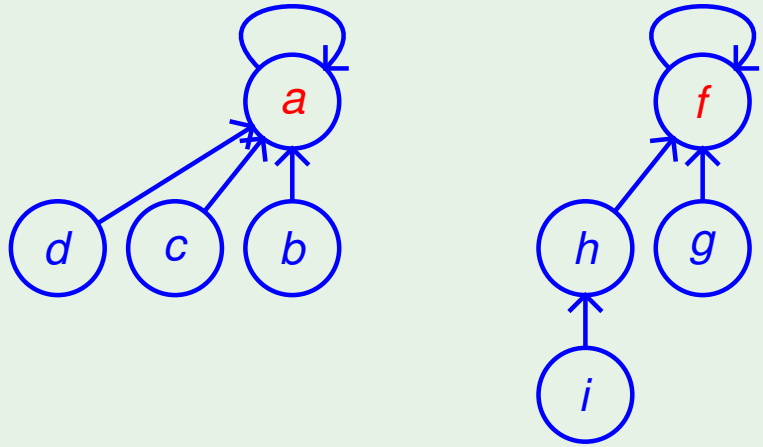
The unionSet operation

Basic mechanism

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. dsuf[pxI]=pyI

Example (Two sets)



Example (Union of sets)

Option-1

Option-2

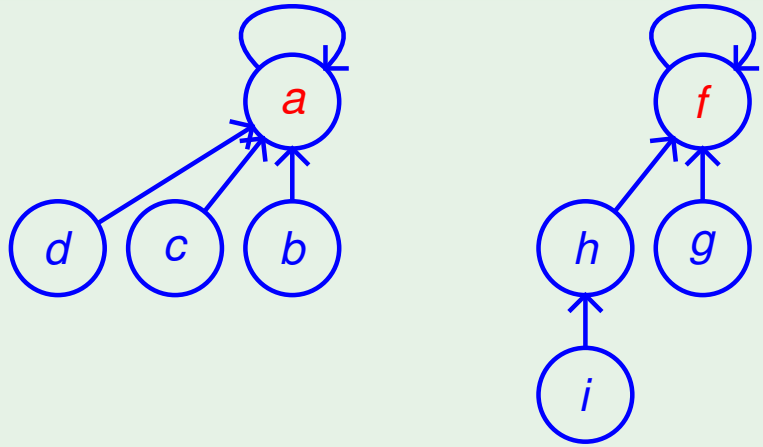
The unionSet operation

Basic mechanism

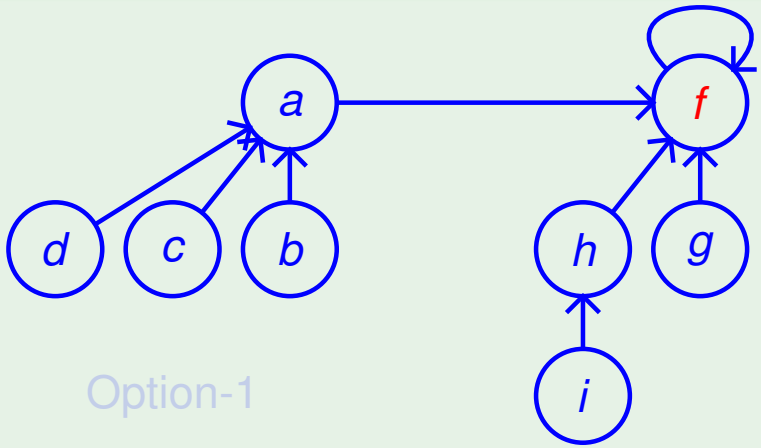
unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. dsuf[pxI]=pyI

Example (Two sets)



Example (Union of sets)



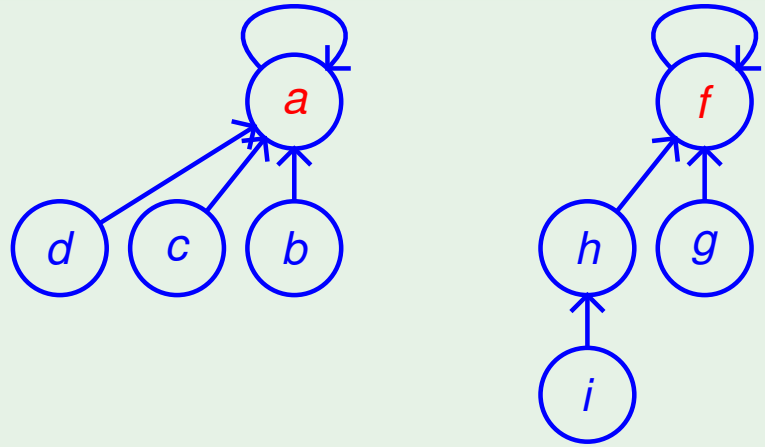
The unionSet operation

Basic mechanism

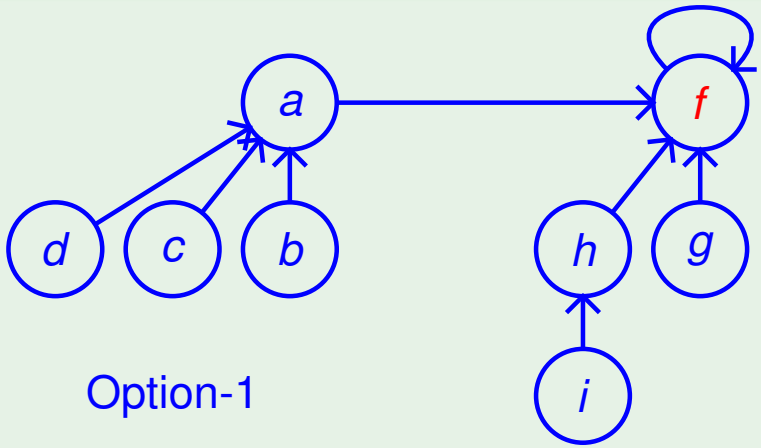
unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. dsuf[pxI]=pyI

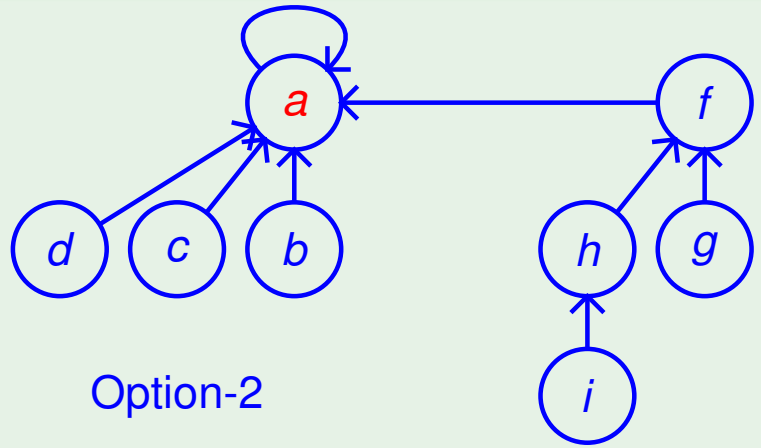
Example (Two sets)



Example (Union of sets)



Option-1



Option-2

Size guided unionSet operation

unionSet by size

unionSet(xl, yl, dsuf[])

1. pxl = findSet(xl, dsuf)
2. pyl = findSet(yl, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyl].p = pxl
5. dsuf[pxi].sz += dsuf[pyl].sz
6. else
7. dsuf[pxl].p = pyl
8. dsuf[pyl].sz += dsuf[pxl].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Proof.

Inductive hypothesis: Let a tree formed by up to i links satisfy $|S| \geq 2^{h_r}$

Base case: Tree for singleton set has $|\{r\}| = 1$ and $h_r = 0$, thus $|\{r\}| \geq 2^{h_r}$

Inductive step: Trees for S_1 and S_2 with cannonical elements r and s and heights h_r and h_s, respectively are linked; $S_1 \geq S_2$ to form S with height h'_r; $|S| = |S_1| + |S_2|$

Case 1 ($h_r > h_s$)**:** Now $h'_r = h_r$

$|S| \geq |S_1| \geq 2^{h_r} = 2^{h'_r}$

Case 2 ($h_r \leq h_s$)**:** Now $h'_r = h_s + 1$

$|S| = |S_1| + |S_2| \geq |S_2|$

Time bound for unionSet by size

Tree height for findSet or unionSet is $\leq \lfloor \lg |S| \rfloor$ – follows by corollary

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Proof.

Inductive hypothesis: Let a tree formed by up to i links satisfy $|S| \geq 2^{h_r}$
Base case: Tree for singleton set has $|\{r\}| = 1$ and $h_r = 0$, thus $|\{r\}| \geq 2^{h_r}$
Inductive step: Trees for S_1 and S_2 with cannonical elements r and s and heights h_r and h_s, respectively are linked; $S_1 \geq S_2$ to form S with height h'_r; $|S| = |S_1| + |S_2|$
Case 1 ($h_r > h_s$)**:** Now $h'_r = h_r$
$|S| \geq |S_1| \geq 2^{h_r} = 2^{h'_r}$
Case 2 ($h_r \leq h_s$)**:** Now $h'_r = h_s + 1$
$|S| = |S_1| + |S_2| \geq |S_2|$

Time bound for unionSet by size

Tree height for findSet or unionSet is $\leq \lfloor \lg |S| \rfloor$ – follows by corollary

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Proof.

Inductive hypothesis: Let a tree formed by up to i links satisfy $|S| \geq 2^{h_r}$
Base case: Tree for singleton set has $|\{r\}| = 1$ and $h_r = 0$, thus $|\{r\}| \geq 2^{h_r}$
Inductive step: Trees for S_1 and S_2 with cannonical elements r and s and heights h_r and h_s, respectively are linked; $S_1 \geq S_2$ to form S with height h'_r; $|S| = |S_1| + |S_2|$
Case 1 ($h_r > h_s$): Now $h'_r = h_r$
$|S| \geq |S_1| \geq 2^{h_r} = 2^{h'_r}$
Case 2 ($h_r \leq h_s$): Now $h'_r = h_s + 1$
$|S| = |S_1| + |S_2| \geq |S_2|$

Time bound for unionSet by size

Tree height for findSet or unionSet is $\leq \lfloor \lg |S| \rfloor$ – follows by corollary

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Proof.

Inductive hypothesis: Let a tree formed by up to i links satisfy $|S| \geq 2^{h_r}$
Base case: Tree for singleton set has $|\{r\}| = 1$ and $h_r = 0$, thus $|\{r\}| \geq 2^{h_r}$
Inductive step: Trees for S_1 and S_2 with cannonical elements r and s and heights h_r and h_s, respectively are linked; $S_1 \geq S_2$ to form S with height h'_r; $|S| = |S_1| + |S_2|$
Case 1 ($h_r > h_s$): Now $h'_r = h_r$
$|S| \geq |S_1| \geq 2^{h_r} = 2^{h'_r}$
Case 2 ($h_r \leq h_s$): Now $h'_r = h_s + 1$
$|S| = |S_1| + |S_2| \geq |S_2|$

Time bound for unionSet by size

Tree height for findSet or unionSet is $\leq \lfloor \lg |S| \rfloor$ – follows by corollary

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Proof.

Inductive hypothesis: Let a tree formed by up to i links satisfy $|S| \geq 2^{h_r}$
Base case: Tree for singleton set has $|\{r\}| = 1$ and $h_r = 0$, thus $|\{r\}| \geq 2^{h_r}$
Inductive step: Trees for S_1 and S_2 with cannonical elements r and s and heights h_r and h_s, respectively are linked; $S_1 \geq S_2$ to form S with height h'_r; $|S| = |S_1| + |S_2|$
Case 1 ($h_r > h_s$): Now $h'_r = h_r$
$|S| \geq |S_1| \geq 2^{h_r} = 2^{h'_r}$
Case 2 ($h_r \leq h_s$): Now $h'_r = h_s + 1$
$|S| = |S_1| + |S_2| \geq |S_2| \geq 2^{h_s}$

Time bound for unionSet by size

Tree height for findSet or unionSet is $\leq \lfloor \lg |S| \rfloor$ – follows by corollary

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Proof.

Inductive hypothesis: Let a tree formed by up to i links satisfy $|S| \geq 2^{h_r}$
Base case: Tree for singleton set has $|\{r\}| = 1$ and $h_r = 0$, thus $|\{r\}| \geq 2^{h_r}$
Inductive step: Trees for S_1 and S_2 with cannonical elements r and s and heights h_r and h_s, respectively are linked; $S_1 \geq S_2$ to form S with height h'_r; $|S| = |S_1| + |S_2|$
Case 1 ($h_r > h_s$): Now $h'_r = h_r$
$|S| \geq |S_1| \geq 2^{h_r} = 2^{h'_r}$
Case 2 ($h_r \leq h_s$): Now $h'_r = h_s + 1$
$|S| = |S_1| + |S_2| \geq 2|S_2| \geq 2 \cdot 2^{h_s}$

Time bound for unionSet by size

Tree height for findSet or unionSet is $\leq \lfloor \lg |S| \rfloor$ – follows by corollary

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Proof.

Inductive hypothesis: Let a tree formed by up to i links satisfy $|S| \geq 2^{h_r}$
Base case: Tree for singleton set has $|\{r\}| = 1$ and $h_r = 0$, thus $|\{r\}| \geq 2^{h_r}$
Inductive step: Trees for S_1 and S_2 with cannonical elements r and s and heights h_r and h_s, respectively are linked; $S_1 \geq S_2$ to form S with height h'_r; $|S| = |S_1| + |S_2|$
Case 1 ($h_r > h_s$): Now $h'_r = h_r$
$|S| \geq |S_1| \geq 2^{h_r} = 2^{h'_r}$
Case 2 ($h_r \leq h_s$): Now $h'_r = h_s + 1$
$|S| = |S_1| + |S_2| \geq 2|S_2| = 2^{h_s+1} = 2^{h'_r}$

Time bound for unionSet by size

Tree height for findSet or unionSet is $\leq \lfloor \lg |S| \rfloor$ – follows by corollary

Size guided unionSet operation

unionSet by size

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].sz $>$ dsuf[pyi].sz)
4. dsuf[pyI].p = pxI
5. dsuf[pxi].sz += dsuf[pyI].sz
6. else
7. dsuf[pxI].p = pyI
8. dsuf[pyI].sz += dsuf[pxI].sz

Proof.

Inductive hypothesis: Let a tree formed by up to i links satisfy $|S| \geq 2^{h_r}$
Base case: Tree for singleton set has $|\{r\}| = 1$ and $h_r = 0$, thus $|\{r\}| \geq 2^{h_r}$
Inductive step: Trees for S_1 and S_2 with cannonical elements r and s and heights h_r and h_s, respectively are linked; $S_1 \geq S_2$ to form S with height h'_r; $|S| = |S_1| + |S_2|$
Case 1 ($h_r > h_s$): Now $h'_r = h_r$
$|S| \geq |S_1| \geq 2^{h_r} = 2^{h'_r}$
Case 2 ($h_r \leq h_s$): Now $h'_r = h_s + 1$
$|S| = |S_1| + |S_2| \geq 2|S_2| = 2^{h_s+1} = 2^{h'_r}$

Max height for unionSet by size

If $r \in S$ is the cannonical member, unionSet by size ensures that $|S| \geq 2^{h_r}$

Time bound for unionSet by size

Tree height for findSet or unionSet is $\leq \lfloor \lg |S| \rfloor$ – follows by corollary

Rank guided unionSet operation

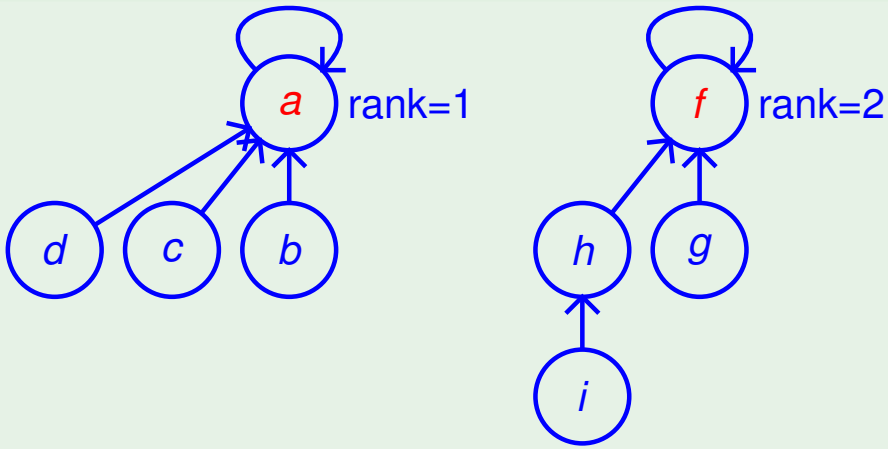
- For now let rank(x)=height(x)
- For a single node tree, rank(x)=0
- Rank changes only through union

makeSet for union by rank

makeSet(xI, dsuf[])

1. dsuf[xI].p = xI
2. dsuf[xI].rk = 0

Example (unionSet(a, f, ...) by rank)



unionSet by rank

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].rk>dsuf[pyi].rk)
4. dsuf[pyI].p = pxI
5. elsif (dsuf[pxi].rk<dsuf[pyi].rk)
6. dsuf[pxI].p = pyI
7. else
8. dsuf[pyI].p = pxI
9. dsuf[pxI].rk += 1

Rank guided unionSet operation

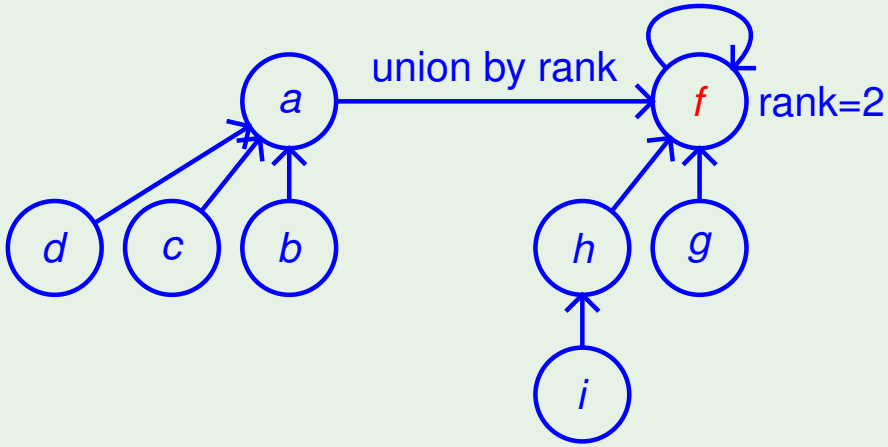
- For now let rank(x)=height(x)
- For a single node tree, rank(x)=0
- Rank changes only through union

makeSet for union by rank

makeSet(xI, dsuf[])

1. dsuf[xI].p = xI
2. dsuf[xI].rk = 0

Example (unionSet(a, f, ...) by rank)



unionSet by rank

unionSet(xI, yI, dsuf[])

1. pxI = findSet(xI, dsuf)
2. pyI = findSet(yI, dsuf)
3. if (dsuf[pxi].rk>dsuf[pyi].rk)
4. dsuf[pyI].p = pxI
5. elsif (dsuf[pxi].rk<dsuf[pyi].rk)
6. dsuf[pxI].p = pyI
7. else
8. dsuf[pyI].p = pxI
9. dsuf[pxI].rk += 1

Properties of linking by rank

1. If x is not a root node, then rank(x) $<$ rank(parent(x))
   A node of rank k is created only by merging two root nodes of rank $k-1$
2. If x is not a root node, then rank(x) will never change again
   Rank changes only for root nodes; a non-root node never becomes a root
3. If parent(x) changes, then rank(parent(x)) strictly increases
   Only a root node can link to the root node of another tree; if x is a root node, before linking parent(x) = x; after x is linked to a new root r due to union by rank, we have rank(r) $>$ rank(x)
4. If root node of tree for set S has rank k, $|S| \geq 2^k$ **(inductive hypothesis)**

   Base case: satisfied for a singleton node tree with rank $k = 0$

   Inductive step: A root node of rank $k+1$ is created only by linking a root node of rank k to another
   - By inductive hypothesis, each subtree has at least 2^k nodes
   - Resulting tree has at least $2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$ nodes

Properties of linking by rank

1. If x is not a root node, then rank(x) $<$ rank(parent(x))
   A node of rank k is created only by merging two root nodes of rank $k-1$
2. If x is not a root node, then rank(x) will never change again
   Rank changes only for root nodes; a non-root node never becomes a root
3. If parent(x) changes, then rank(parent(x)) strictly increases
   Only a root node can link to the root node of another tree; if x is a root node, before linking parent(x) = x; after x is linked to a new root r due to union by rank, we have rank(r) $>$ rank(x)
4. If root node of tree for set S has rank k, $|S| \geq 2^k$ **(inductive hypothesis)**

   Base case: satisfied for a singleton node tree with rank $k = 0$

   Inductive step: A root node of rank $k+1$ is created only by linking a root node of rank k to another
   - By inductive hypothesis, each subtree has at least 2^k nodes
   - Resulting tree has at least $2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$ nodes

Properties of linking by rank

1. If x is not a root node, then rank(x) $<$ rank(parent(x))
   A node of rank k is created only by merging two root nodes of rank $k-1$
2. If x is not a root node, then rank(x) will never change again
   Rank changes only for root nodes; a non-root node never becomes a root
3. If parent(x) changes, then rank(parent(x)) strictly increases
   Only a root node can link to the root node of another tree; if x is a root node, before linking parent(x) = x; after x is linked to a new root r due to union by rank, we have rank(r) $>$ rank(x)
4. If root node of tree for set S has rank k, $|S| \geq 2^k$ **(inductive hypothesis)**

   Base case: satisfied for a singleton node tree with rank $k = 0$

   Inductive step: A root node of rank $k+1$ is created only by linking a root node of rank k to another
   - By inductive hypothesis, each subtree has at least 2^k nodes
   - Resulting tree has at least $2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$ nodes

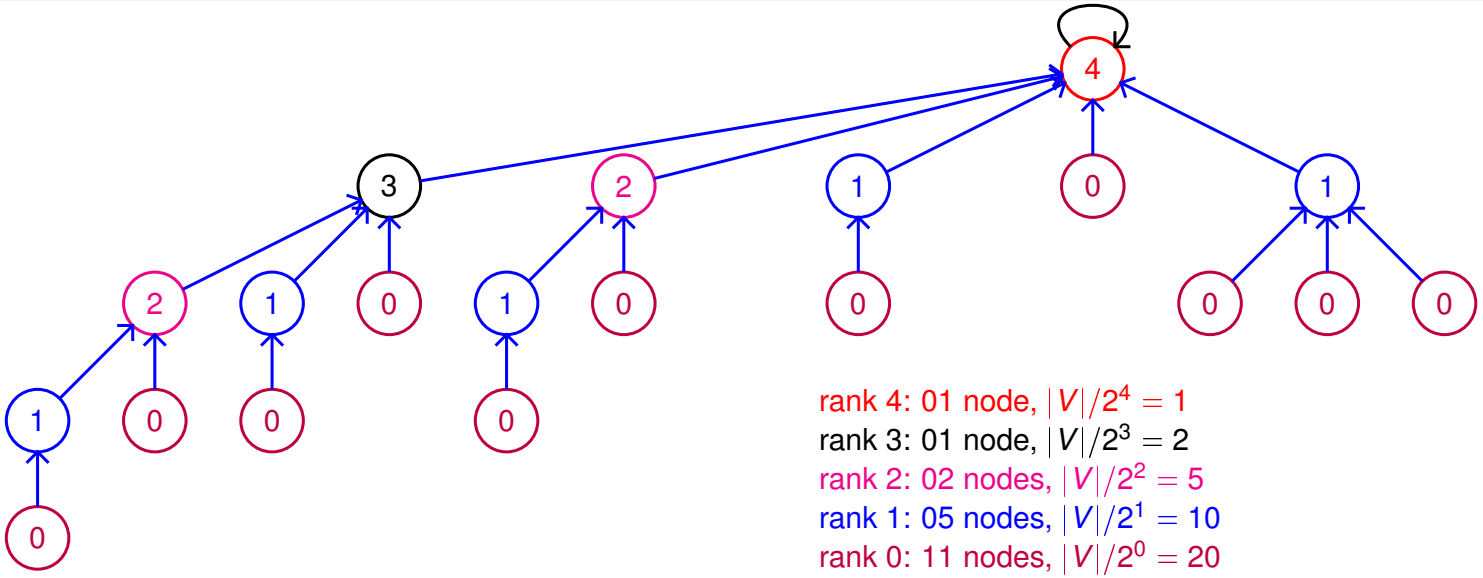
Properties of linking by rank

1. If x is not a root node, then rank(x) $<$ rank(parent(x))
   A node of rank k is created only by merging two root nodes of rank $k-1$
2. If x is not a root node, then rank(x) will never change again
   Rank changes only for root nodes; a non-root node never becomes a root
3. If parent(x) changes, then rank(parent(x)) strictly increases
   Only a root node can link to the root node of another tree; if x is a root node, before linking parent(x) = x; after x is linked to a new root r due to union by rank, we have rank(r) $>$ rank(x)
4. If root node of tree for set S has rank k, $|S| \geq 2^k$ **(inductive hypothesis)**

   Base case: satisfied for a singleton node tree with rank $k = 0$

   Inductive step: A root node of rank $k+1$ is created only by linking a root node of rank k to another
   - By inductive hypothesis, each subtree has at least 2^k nodes
   - Resulting tree has at least $2^k + 2^k = 2 \cdot 2^k = 2^{k+1}$ nodes

Properties of linking by rank (contd.)



5. For $r \geq 0, r \in \mathbb{Z}$, there are at most $n_r = |S|/2^r$ nodes with rank r in T_S
   - i. Claim is trivially satisfied for rank $r = 0$
   - ii. Let it be satisfied for rank $r = k$
   - iii. A tree with root node of rank $(k + 1)$ is formed in two ways:
     - **Case-A** linking the root of a tree with with rank k to the root of a tree with rank k, whose rank increases to $k + 1$
     - **Case-B** linking the root of a tree with with rank $\leq k$ to the root of a tree with rank $k + 1$

Properties of linking by rank (contd.)

Case-A Trees for sets S_1 and S_2 of same rank k are linked

(iv) For rank $r < k$, T_{S_1} and T_{S_2} satisfy $n_r^{(1)} \leq |S_1|/2^r$ and $n_r^{(2)} \leq |S_1|/2^r$

(v) For the composite tree, for $r < k$, $n_r = n_r^{(1)} + n_r^{(2)}$; $|S| = |S_1| + |S_2|$

(vi) $\therefore n_r \leq |S_1|/2^r + |S_2|/2^r = |S|/2^r$

(vii) For $r = k$, $n_k = 1$ satisfies $n_k \leq |S_1|/2^k \leq |S|/2^k$

(viii) For $r = k+1$, root node of rank $(k+1)$ ($n_{k+1} = 1$) has at least 2^{k+1} descendants [property-4]

(ix) $\therefore |S| \geq 2^{k+1}$ and $|S|/2^{k+1} \geq 1$

Case-B: Trees S_2 of ranks $k+1$ and S_1 of rank $k' \leq k$ are linked
Assume that the tree for set S_2 is initially formed through Case-A

(x) For rank $r \leq k$, T_{S_1} and T_{S_2} satisfy $n_r^{(1)} \leq |S_1|/2^r$ and $n_r^{(2)} \leq |S_1|/2^r$

(xi) For the composite tree, for $r \leq k$, $n_r = n_r^{(1)} + n_r^{(2)}$; $|S| = |S_1| + |S_2|$

(xii) $\therefore n_r \leq |S_1|/2^r + |S_2|/2^r = |S|/2^r$

(xiii) For $r = k+1$, $n_{k+1} = 1$ satisfies $n_{k+1} \leq |S_2|/2^{k+1} \leq |S|/2^{k+1}$

(xiv) The reasoning for this case continues to apply to trees of rank $k+1$ fromed subsequently through this case

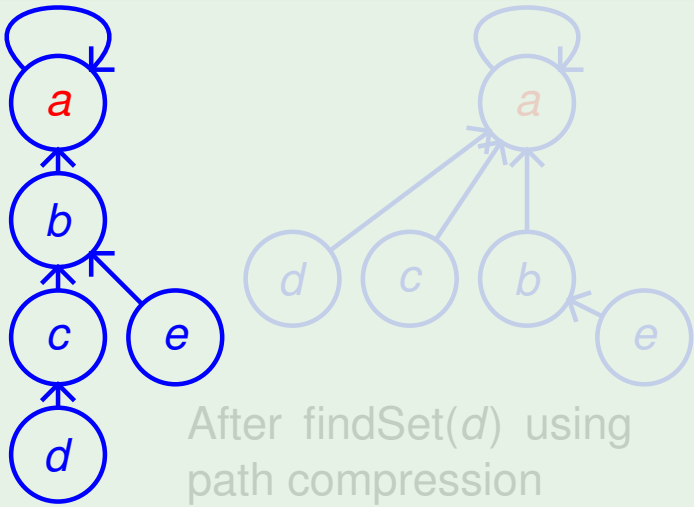
Union with path compression

Path compression mechanism

findSet(xI, dsuf[])

1. pI = dsuf[xI]
2. if (pI≠xI)
3. dsuf[xi]=findSet(pI)
4. return dsuf[xI]

Example (Path compression)



After findSet(*d*) using path compression

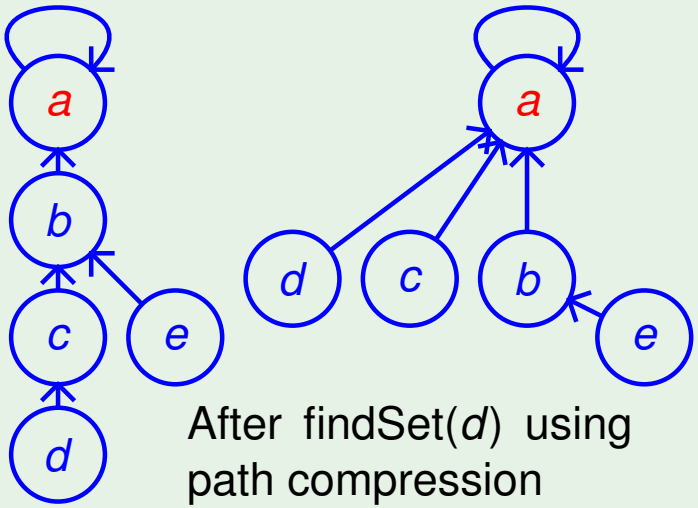
Union with path compression

Path compression mechanism

findSet(xI, dsuf[])

1. pI = dsuf[xI]
2. if (pI≠xI)
3. dsuf[xi]=findSet(pI)
4. return dsuf[xI]

Example (Path compression)



After findSet(*d*) using path compression

Properties of linking by rank and path compression

6. The highest rank of a node in a tree of a set S is at most $\lfloor \lg |S| \rfloor$
   Follows from property-4
7. Linking sets S_1, S_2 by rank for unionSet, ensures that time for any unionSet for findSet operation is $O(\lg(|S_1| + |S_2|))$
   Follows from property-6

   ..... Now also consider path compression .....

8. With path compression ranks do not change though nodes may loose height
9. In general, height(x) $\leq$ rank(x) but they are not necessarily equal
10. The rank of the parent of a node may increase – a node may be linked to a parent of a higher rank
11. If the parent changes, the rank of the new parent is strictly greater

Grouping of ranks by the iterated logarithm

$$\lg^* n = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + \lg^*(\lg n) & \text{otherwise} \end{cases}$$

Group G_k has integers in the range $(k+1)..2^k$

n	$\lg^* n$	Grp
1	0	G_0
2	1	G_1
[3, 4]	2	G_2
[5, 16]	3	G_4
[17, 65536]	4	G_{16}
[65537, 2^{65536}]	5	$G_{2^{16}}$

Grouping of ranks by the iterated logarithm

$$\lg^* n = \begin{cases} 0 & \text{if } n \leq 1 \\ 1 + \lg^*(\lg n) & \text{otherwise} \end{cases}$$

Group G_k has integers in the range
$(k+1)..2^k$

n	$\lg^* n$	Grp
1	0	G_0
2	1	G_1
[3, 4]	2	G_2
[5, 16]	3	G_4
[17, 65536]	4	G_{16}
[65537, 2^{65536}]	5	$G_{2^{16}}$

We have $\lg^* n \leq 5$ unless n exceeds the number of atoms in the universe – a "reasonable" assumption for an upper bound on the size of the sets we consider!

12. Every non-zero rank falls within one of the first $\lg^* n$ groups, assuming that the sets are of "reasonable" size

    Note that the rank of a root node for a set S is between 0 and $\lfloor \lg |S| \rfloor$ [property-6]

Disbursing credits to drive findSet

If a node ceases to be the root and its rank is in the interval of G_k it gets 2^k credits

Costing credits disbursed

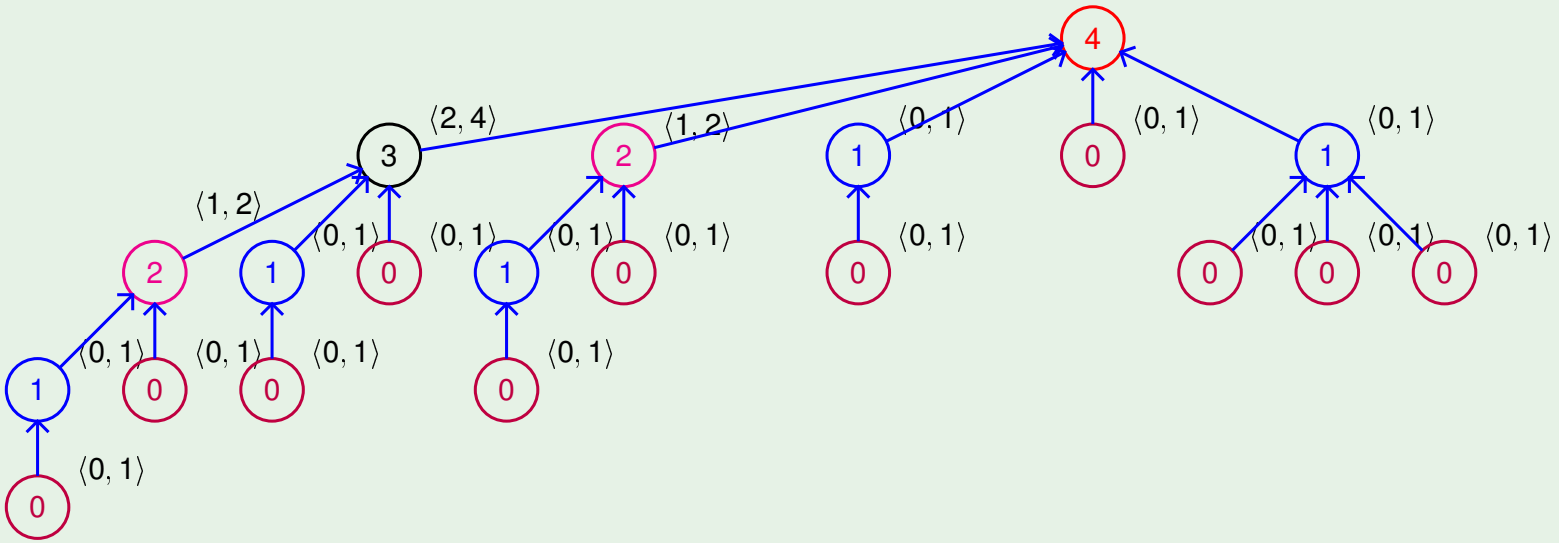
Number of credits disbursed to all nodes is $|S| \lg^* |S|$

Proof.

- By property-5, the number of nodes with ranks in G_k (the subrange $[(k+1)..2^k]$) is at most $\frac{|S|}{2^{k+1}} + \frac{|S|}{2^{k+2}} + \ldots + \frac{|S|}{2^{2^k}} \leq \frac{|S|}{2^k}$
- Thus, nodes in group G_k need at most $|S|$ credits in total @ 2^k credits per node
- As there are at most $\lg^* |S|$ groups [property-12], at most $|S| \lg^* |S|$ credits get disbursed

Amortised costing of find with path compression

Example (Showing group of rank and credits received)



- Each node in G_k has enough credits to move up to the node of highest rank in G_k – done only once over m finds
- Such link traversals within G_k are supported by disbursed credits
- Crossings from G_k to G_{k+1} (at most $\lg^* n$) charged to each find
- Total cost of m finds: $O((m+n)(\lg^* n))$
- Unions involve two finds and a linking (re-ordered)

Complexity of Kruskal's algorithm with DSUF

1. create a min heap H of the edges
2. initialise $T \leftarrow \phi$; edge count $c \leftarrow 0$
3. foreach $v \in V$ makeSet(v)
4. while ($c < |V| - 1$) do
5. extract min cost edge $e = \langle v_i, v_j \rangle \in H$
6. if (findSet(v_i)$\neq$findSet(v_j))
   // no cycle formed by adding e to T
7. add e to T; $c \leftarrow c + 1$
8. unionSet(v_i, v_j)
   // track the two components joining
9. done

- L1 takes $O(|E|)$ time
- L3 takes $O(|V|)$ time
- L5 takes $O(|E| \lg |E|) \equiv O(|E| \lg |V|)$ time overall
- L6 and L8 takes $O(|E| \lg^* |V|)$ time overall using DSUF