Algorithms II, CS 502

Data Structures for Disjoint Sets

Ugur Dogrusoz

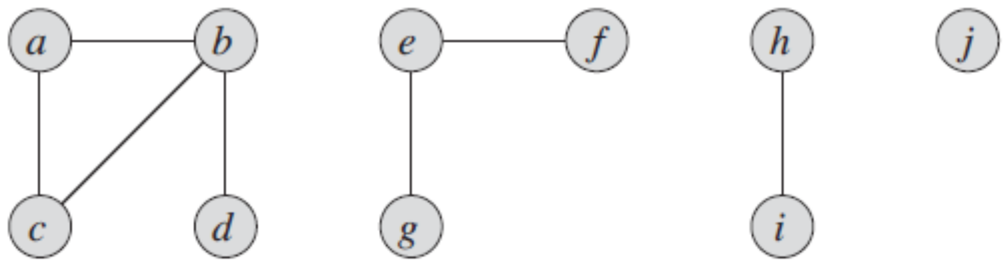
Computer Eng Dept, Bilkent Univ

Disjoint-set data structure

- Maintains a collection of disjoint dynamic sets $S = \{S_1, S_2, \ldots, S_k\}$
  - Identify each set with a representative, a member of the set
  - Supported operations:
    - MAKE-SET(x) creates a new set with member x
    - UNION(x,y) unites dynamic sets containing x and y
    - FIND-SET(x) returns a pointer to representative of set containing x
  - Analyze running time using *n*: # of MAKE-SET operations and *m*: # of total operations (MAKE-SET, UNION, and FIND-SET)

An application: connected components

- Construct and answer connected components of an undirected graph



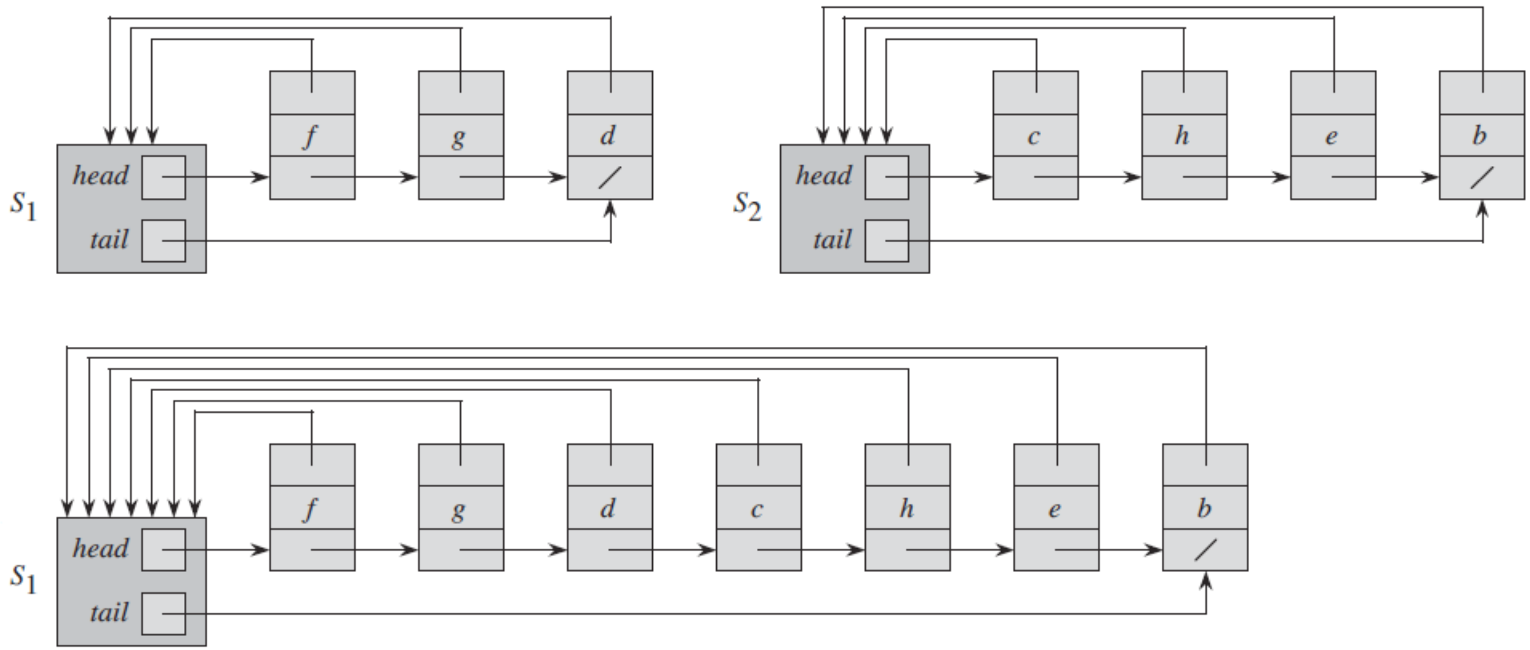
Edge processed	Collection of disjoint sets								
initial sets	{a}	{b}	{c}	{d}	{e}	{f}	{g}	{h}	{i}	{j}
(b,d)	{a}	{b,d}	{c}		{e}	{f}	{g}	{h}	{i}	{j}
(e,g)	{a}	{b,d}	{c}		{e,g}	{f}		{h}	{i}	{j}
(a,c)	{a,c}	{b,d}			{e,g}	{f}		{h}	{i}	{j}
(h,i)	{a,c}	{b,d}			{e,g}	{f}		{h,i}		{j}
(a,b)	{a,b,c,d}				{e,g}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

An application: connected components

```
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
```

Disjoint sets: linked list representation



MAKE-SET(x)	O(1)	
FIND-SET(x)	O(1)	
UNION(x,y)	O(n)	θ(n) amortized running time

Disjoint sets: linked list representation

Operation	Number of objects updated
MAKE-SET(x_1)	1
MAKE-SET(x_2)	1
$\vdots$	$\vdots$
MAKE-SET(x_n)	1
UNION(x_2, x_1)	1
UNION(x_3, x_2)	2
UNION(x_4, x_3)	3
$\vdots$	$\vdots$
UNION(x_n, x_{n-1})	$n - 1$

$\theta(n^2) / n = \theta(n)$ amortized running time

Disjoint sets: linked list representation

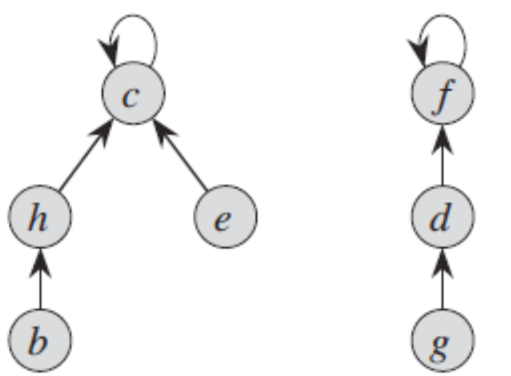
- Weighted union heuristic:
  - always append the shorter list onto the longer, breaking ties arbitrarily
  - a single union will still take $\Omega(n)$ time if both sets have $\Omega(n)$ members

Disjoint sets: linked list representation

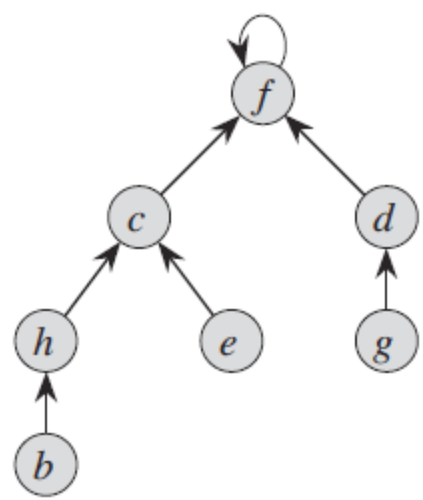
- **Theorem** Using linked-list representation of disjoint sets and weighted-union heuristic, a sequence of m MAKE-SET, UNION, and FIND-SET operations, n of which are MAKE-SET operations, takes $O(m + n \lg n)$ time
- **Proof** For $k \leq n$, after x's pointer has been updated $\lfloor \lg k \rfloor$ times, the resulting set must have at least k members.

Disjoint sets: forest representation

- Sets by rooted trees, with each node containing one member and each tree representing one set



{b,c,e,h} {d,f,g}



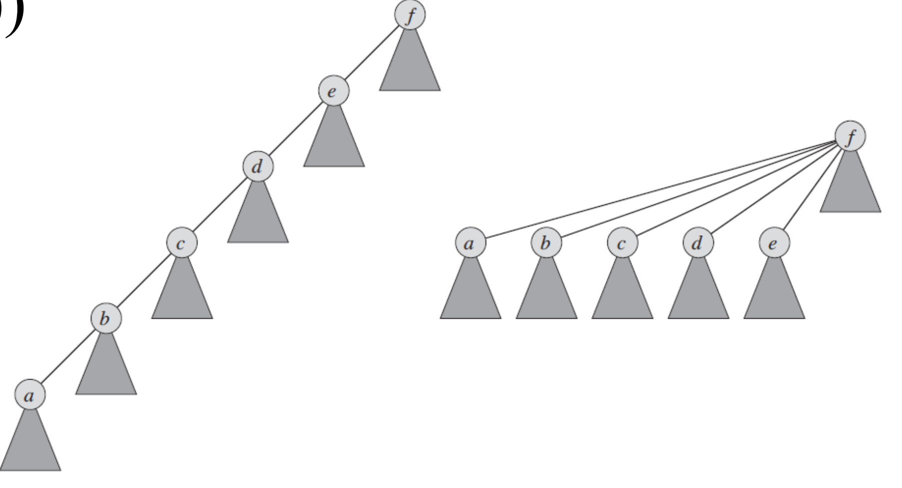
UNION(e,g)={b,c,e,h,d,f,g}

Heuristics: union by rank

- Make root with smaller rank point to root with larger rank during a UNION operation
  - rank: an upper bound on height of a node
  - only the rank of the roots may change:
    - if both roots have same rank, rank of new root increases by 1
    - otherwise, no change
  - O(m log n) (every node has rank at most $\lfloor \lg n \rfloor$)

Heuristics: path compression

- make each node on the find path point directly to root (during FIND-SET)
  - path compression does not change any ranks
  - for a sequence of n MAKE-SET operations and f FIND-SET operations, worst-case running time is $\Theta(n + f(1 + \log_{2+f/n} n))$



Disjoint set forests: operations

```
MAKE-SET(x)
x.p = x
x.rank = 0
```

```
UNION(x, y)
LINK(FIND-SET(x), FIND-SET(y))
```

```
LINK(x, y)
if x.rank > y.rank
    y.p = x
else x.p = y
    if x.rank == y.rank
        y.rank = y.rank + 1
```

```
FIND-SET(x)
if x ≠ x.p
    x.p = FIND-SET(x.p)
return x.p
```

Disjoint set forests, both heuristics

- Worst case running time is O(m $\alpha(n)$)
  - where $\alpha(n)$ is a very slowly growing function (inverse of very fast growing function called Ackermann function $A_k(n)$)

$$\alpha(n) = \begin{cases} 0, \text{ for } 0 \le n \le 2, \\ 1, \text{ for n} = 3, \\ 2, \text{ for } 4 \le n \le 7, \\ 3, \text{ for } 8 \le n \le 2047, \\ 4, \text{ for } 2048 \le n \le A_4(1) \ [A_4(1) \ge 10^{80}], \end{cases}$$