CS3383 Unit 2 Lecture 3: Union Find / Disjoint Set

David Bremner

March 17, 2024

Outline

Union Find

- Motivation: MST
- Forest representation for disjoint sets
- Bounding the height of trees

Union Find

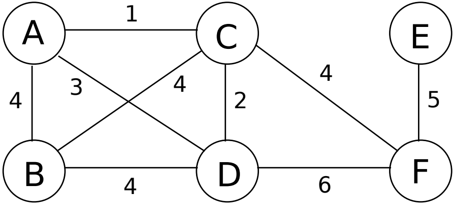
- Motivation: MST
- Forest representation for disjoint sets
- Bounding the height of trees

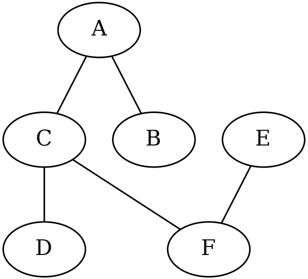
Kruskal's MST algorithm

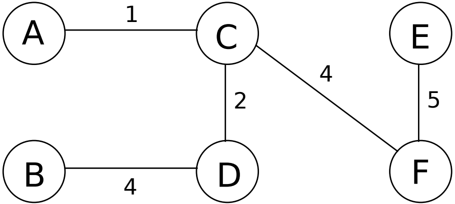
```
def kruskal(n,E):
    P=Partition(n);  X=[]
    E.sort()
    for (weight,u,v) in E:
        if P.find(u) != P.find(v):
            X.append((u,v))
            P.union(u,v)
    return X
```

- How does crossing property apply? What is S?

Kruskal example







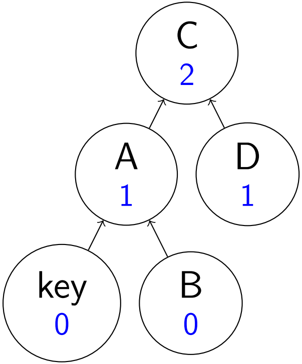
Union Find

- Motivation: MST
- Forest representation for disjoint sets
- Bounding the height of trees

Init and Find

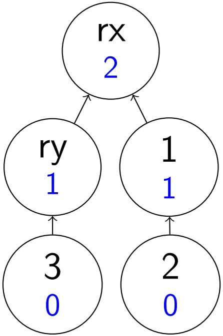
```
def __init__(P,n):
  # sometimes called makeset(j)
  P.parent = [j for j in range(n)]
  P.rank = [0] * n
```

```
def find(P, key):
  while P.parent[key] != key:
    key = P.parent[key]
  return key
```



Union operation

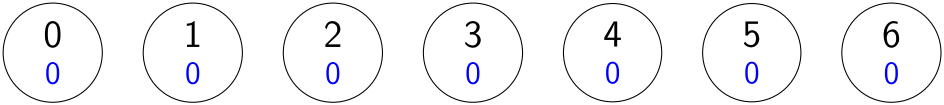
```
def union(P,x,y):
  rx = P.find(x)
  ry = P.find(y)
  if rx != ry:
    if P.rank[rx] > P.rank[ry]:
      P.parent[ry] = rx
    else:
      P.parent[rx] = ry
      if P.rank[rx] == P.rank[ry]:
        P.rank[ry] += 1
```



Case 1 of main if

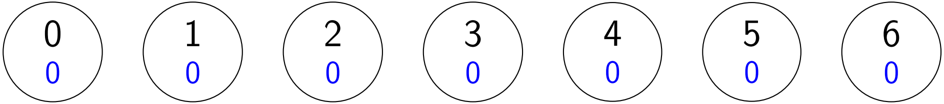
Union Find Example 1/3

- initial partition

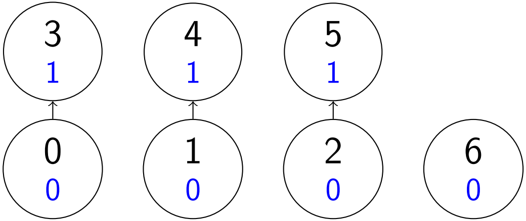


Union Find Example 1/3

- initial partition

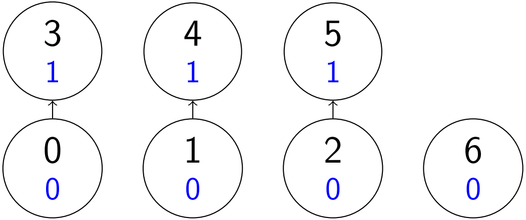


- after union(0,3), union(1,4), union(2,5)

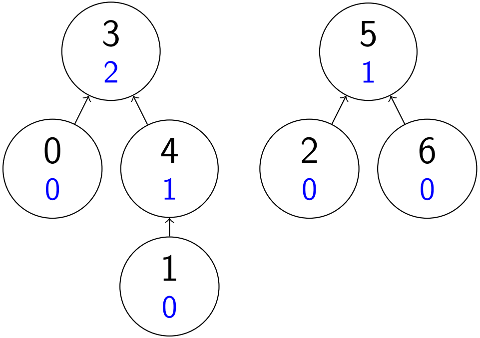


Union Find Example 2/3

after union(0,3), union(1,4), union(2,5)

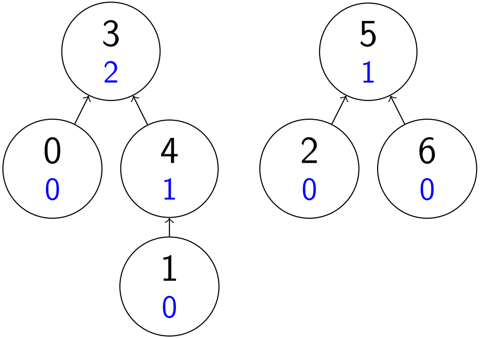


after union(2,6), union(4,0)

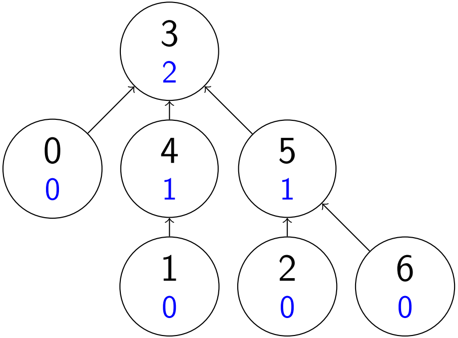


Union Find Example 3/3

after union(2,6), union(4,0)



after union(1,6)



Union Find

- Motivation: MST
- Forest representation for disjoint sets
- Bounding the height of trees

Properties of Union Find trees

Property 1

For any x such that $\mathrm{parent}(x) \neq x$,
$\mathrm{rank}(x) < \mathrm{rank}(\mathrm{parent}(x))$

Properties of Union Find trees

Property 1

For any x such that $\text{parent}(x) \neq x$,
$\text{rank}(x) < \text{rank}(\text{parent}(x))$

Property 2

Any node of rank k has at least 2^k nodes in its subtree.

Properties of Union Find trees

Property 1

For any x such that $\text{parent}(x) \neq x$,
$\text{rank}(x) < \text{rank}(\text{parent}(x))$

Property 2

Any node of rank k has at least 2^k nodes in its subtree.

Property 3

If there are n elements, there are at most $\lfloor n/2^k \rfloor$ nodes of rank k.

Properties of Union Find trees

Property 1

For any x such that $\text{parent}(x) \neq x$, $\text{rank}(x) < \text{rank}(\text{parent}(x))$

Property 2

Any node of rank k has at least 2^k nodes in its subtree.

Property 3

If there are n elements, there are at most $\lfloor n/2^k \rfloor$ nodes of rank k.

Conclusion

$\therefore$ Trees are height at most $\log_2 n$

Proof of Property 1

Property 1

For any x such that $\mathrm{parent}(x) \neq x$,
$\mathrm{rank}(x) < \mathrm{rank}(\mathrm{parent}(x))$

Proof of Property 1

Property 1

For any x such that $\text{parent}(x) \neq x$, $\text{rank}(x) < \text{rank}(\text{parent}(x))$

Base Case

Initially every node has $\text{parent}(x) = x$.

Proof of Property 1

Property 1

For any x such that $\mathrm{parent}(x) \neq x$,
$\mathrm{rank}(x) < \mathrm{rank}(\mathrm{parent}(x))$

induction

```
if P.rank[rx] > P.rank[ry]:
  P.parent[ry] = rx
else:
  P.parent[rx] = ry
  if P.rank[rx] == P.rank[ry]:
    P.rank[ry] += 1
```

Base Case

Initially every node has $\mathrm{parent}(x) = x$.

Proof of Property 2

Property 2

Any node of rank k has at least 2^k nodes in its subtree.

Proof of Property 2

Property 2

Any node of rank k has at least 2^k nodes in its subtree.

base case

true for $k = 0$.

Proof of Property 2

Property 2

Any node of rank k has at least 2^k nodes in its subtree.

Induction

- Rank $k+1$ is created only when joining two trees of rank k.

```
if P.rank[rx] == P.rank[ry]:
  P.rank[ry] += 1
```

- by induction, each of these subtrees has at least 2^k nodes

base case

true for $k = 0$.

Proof of property 3

Property 3

If there are n elements, are at most $\lfloor n/2^k \rfloor$ nodes of rank k.

Proof of property 3

Property 3

If there are n elements, are at most $\lfloor n/2^k \rfloor$ nodes of rank k.

Proof

- By Property 1 any element has at most one ancestor of rank k.
- Therefore the children of two rank k nodes are distinct.
- Apply property 2.