CE2101/ CZ2101: Algorithm Design and Analysis

Week 6: Review Lecture

Ke Yiping, Kelly

Content

- Dynamic equivalence relations
- Union-find programs
- Kruskal’s algorithm, its correctness and time complexity

Dynamic Equivalence Relations

Can Change

- Dynamic equivalence relation with basic operations
  - Initialize
  - union(p, q)
  - connected(p, q)
- Useful in modelling dynamic relationships and processing connectivity queries

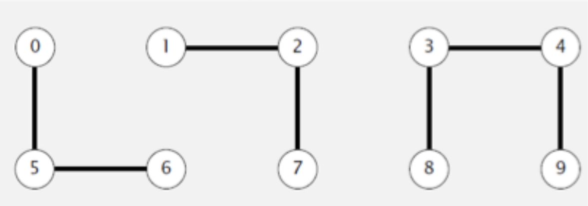
reflexivity
symmetry
transitivity

Union-Find

- Three operations:
  - find(p)
  - union(p, q)
  - connected(p, q)
- Efficient implementation is needed to support huge problems

QuickFind Algorithm

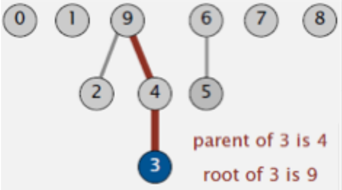
- Use array id[ ] to store the component IDs
- union() is too expensive: O(N)



	0	1	2	3	4	5	6	7	8	9
id[]	0	1	1	8	8	0	0	1	8	8

QuickUnion Algorithm

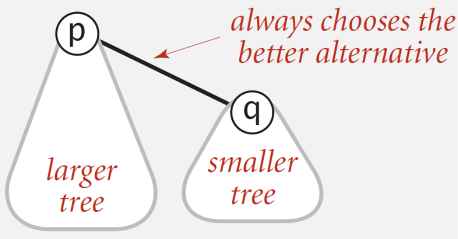
- Use array id[ ] to store the parent IDs
- Trees can grow tall, resulting in O(N) worst case in find()



	0	1	2	3	4	5	6	7	8	9
id[]	0	1	9	4	9	6	6	7	8	9

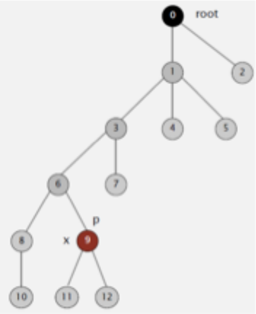
Weighted QuickUnion

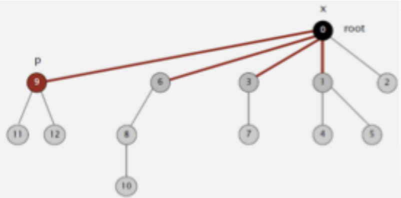
- Weighting:
  - Use extra array sz[ ] to store the size of each component
  - Utilize sz[ ] to decide which tree points to which



Weighted QuickUnion with Path Compression

- Path Compression:
  - In each call of find(), let each node on the examined path point to the root (2-pass)
  - Or let every other node point to its grandparent (1-pass)





Summary: Union-Find

- Complexity for M union-find operations on N objects

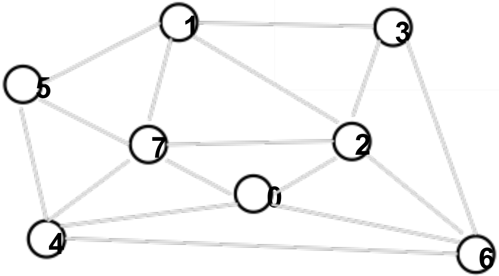
Algorithms	Worst-Case Time
QuickFind	M N
QuickUnion	M N
Weighted QuickUnion	N + M logN
Weighted QuickUnion with Path Compression	N + M log*N

Kruskal's Algorithm

- To solve MST problem
- Uses greedy strategy

Graph edges in weight increasing order

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58



Kruskal's Algorithm: Correctness

- **Proposition.** [Kruskal 1956] Kruskal's algorithm correctly computes the MST.
- Prove by contradiction and the MST property.

Kruskal's Algorithm: Complexity

- It uses the union-find data structure to
  - Implement the dynamically changing resultant tree *T* and
  - For efficient checking of cycles
- Time complexity: O(|E| log|E|)

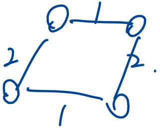
Exercise

Depending on how algo breaks a tie

No. (Same tree, different method)

- Do the Prim's algorithm and the Kruskal's algorithm always obtain the same minimum spanning tree (MST) on a given input graph? If yes, provide a proof. Otherwise, describe when they generate different MSTs. **[AY2021S2]**

Different MST:

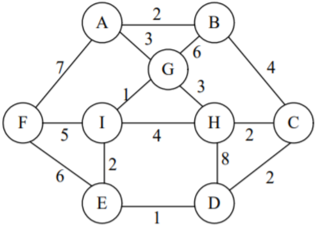


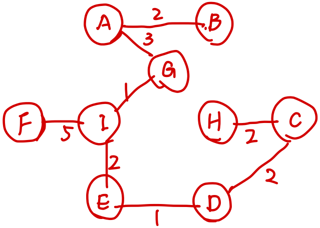
- 2 edge in a cycle
- 2 edge carries the same wt
- 2 edge is the maximum wt

三条件缺一不可

Exercise

- Execute Kruskal's algorithm on G in the figure. Draw the minimum spanning tree obtained.





Edge	Weight	Add to MST?
E D, I G	1	✓
E I, D C, H C, A B	2	✓
A G	3	✓
G H		X cycle
B C	4	X cycle
I H		X cycle
F I	5	✓

Stop. # edge = 8

NANYANG TECHNOLOGICAL UNIVERSITY

Exercise

- In the union(p, q) implementation of the weighted QuickUnion, suppose we set id[root(p)] to q instead of id[root(q)]. Would the resulting algorithm be correct? Yes.

But tree would be big.

Original:

compare p, q size
set smaller to bigger.