16 Oct 2019

Proofs of correctness:

- Q = our post-condition / what the algo should do
- Sequential algo:

First A, then B, then C ... ↙

code:
1: A
2: B
3: C
4: D
(no recur. no loops, no gotos)

~~eg~~ "proof" would look something like

$$A \Rightarrow B \Rightarrow C \Rightarrow E \ldots \Rightarrow Q$$
$$\Rightarrow D \Rightarrow$$

- loop / recursion ~~invar~~, use invariants L
  ↑ use this in description.

(1) Start off on the right foot

Initialization: $P \Rightarrow L$
(precondition ↑ usu. vacuously true.)

(2) Continue to be on the right path.

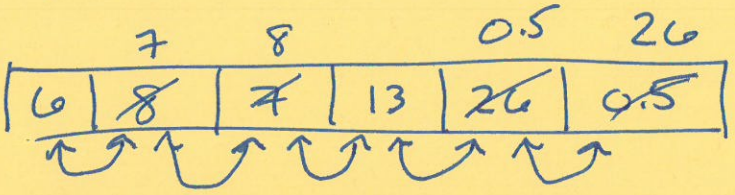
Maintenance: $L_i \wedge G \Rightarrow L_{i+1}$

(3) When it ends, we accomplished the goal.

End: $L_i \wedge \neg G \Rightarrow Q$

(T) Termination: the loop ends
(or $\neg G$ becomes true eventually)
Decrementing function.

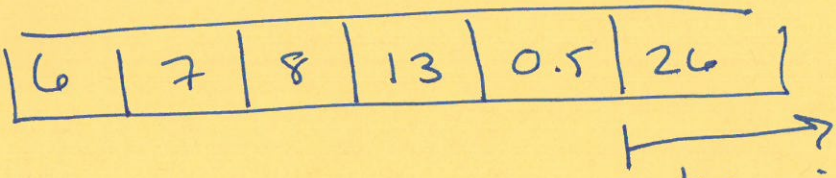
e.g., Bubble Sort




```
for j = 1 ... |A|
    for i = 1 ... |A|-1
        compare Ai with Ai+1 & swap
            if out of order
    end for
end for
```

The outer for loop:

After 1 iteration, we obtain:



this is sorted!

A_n to A_n

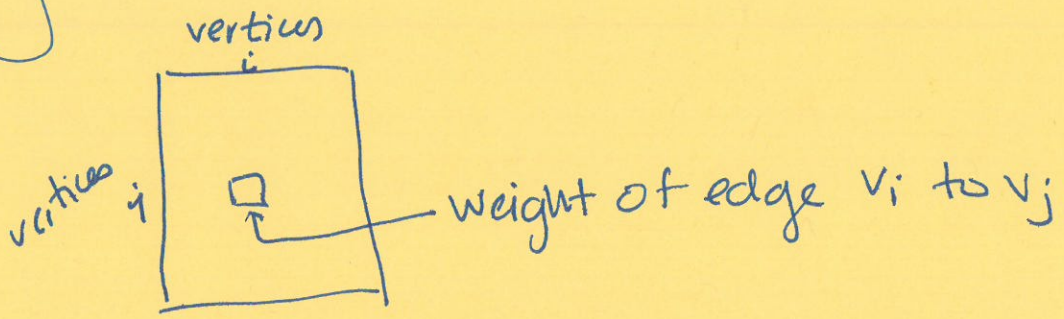
i.e., the set of size (1) at the end of the array = j

Loop invariant: the last j elements are sorted & in their final places, where $j = 0$ before the loop begins.

Challenge: Prove ①, ②, ③, & (T)

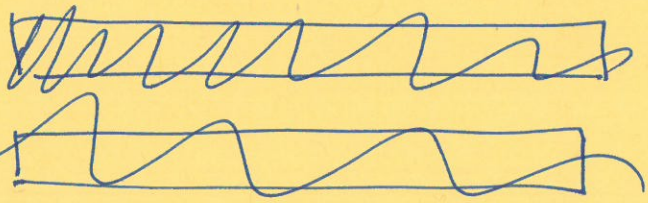
Brainstorm Ways to Represent a Graph on a Computer:

(1) Adjacency Matrix



- good if have many edges compared to vertices.
- sparse matrix could be good otherwise. (e.g., in matlab)
- might care about eigenvalues

(2) Edge List / Node List

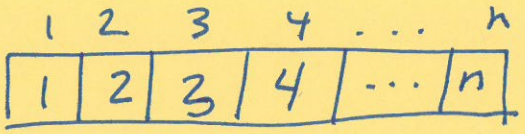


(3) Dictionary: key = id of node
value = id of nodes connected to it.

Note: A tree is a graph.

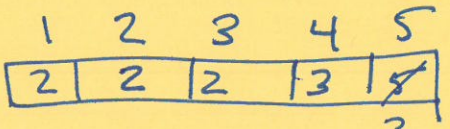
UNION/FIND : to solve connectivity questions

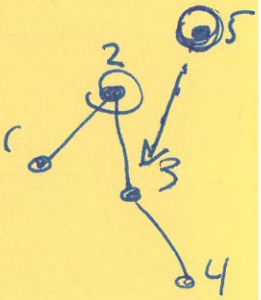
① Quick Find



- array of vertices
  storing label of root node
- Find: $\Theta(1)$
- UNION: $\Theta(n)$ b/c might need to update $\Theta(n)$ entry other node labels

② Quick Union





- array of verts
  storing label of parent node
- Find: $\Theta(n)$
- UNION: FIND-TIME + $\Theta(1)$
  ↳ First, find root of one then update to be the other node.

If I have M operations (either UNION or FIND),

① Takes $O(Mn)$ time

② Takes $\Theta(Mn)$ time.

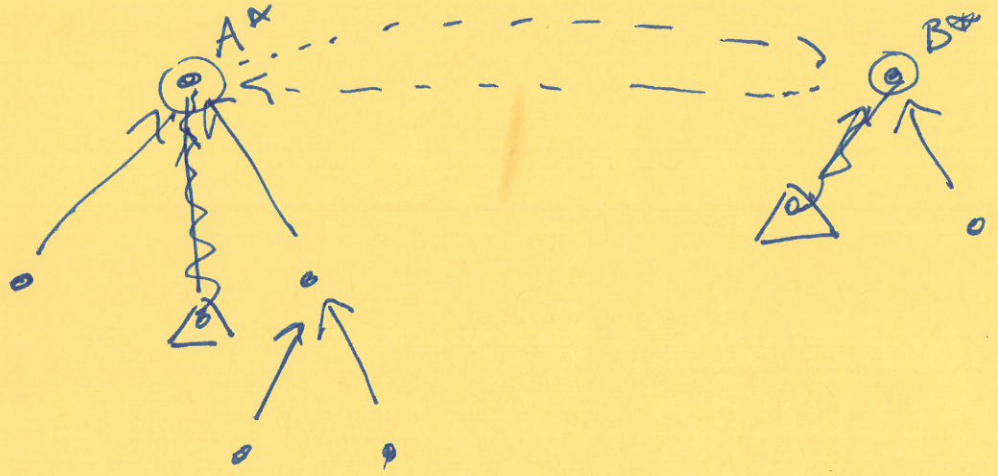
We can do better!

Let's take ② and do some improvements:

First:

Always relabel the smaller tree

①: What is the cost of M union/Find operations NOW?



tree A tree B

Depth ~~maxim~~ is at most $\log n$ with First improvement!

Second: Path compression.

Relabel all nodes you see to have new root node.