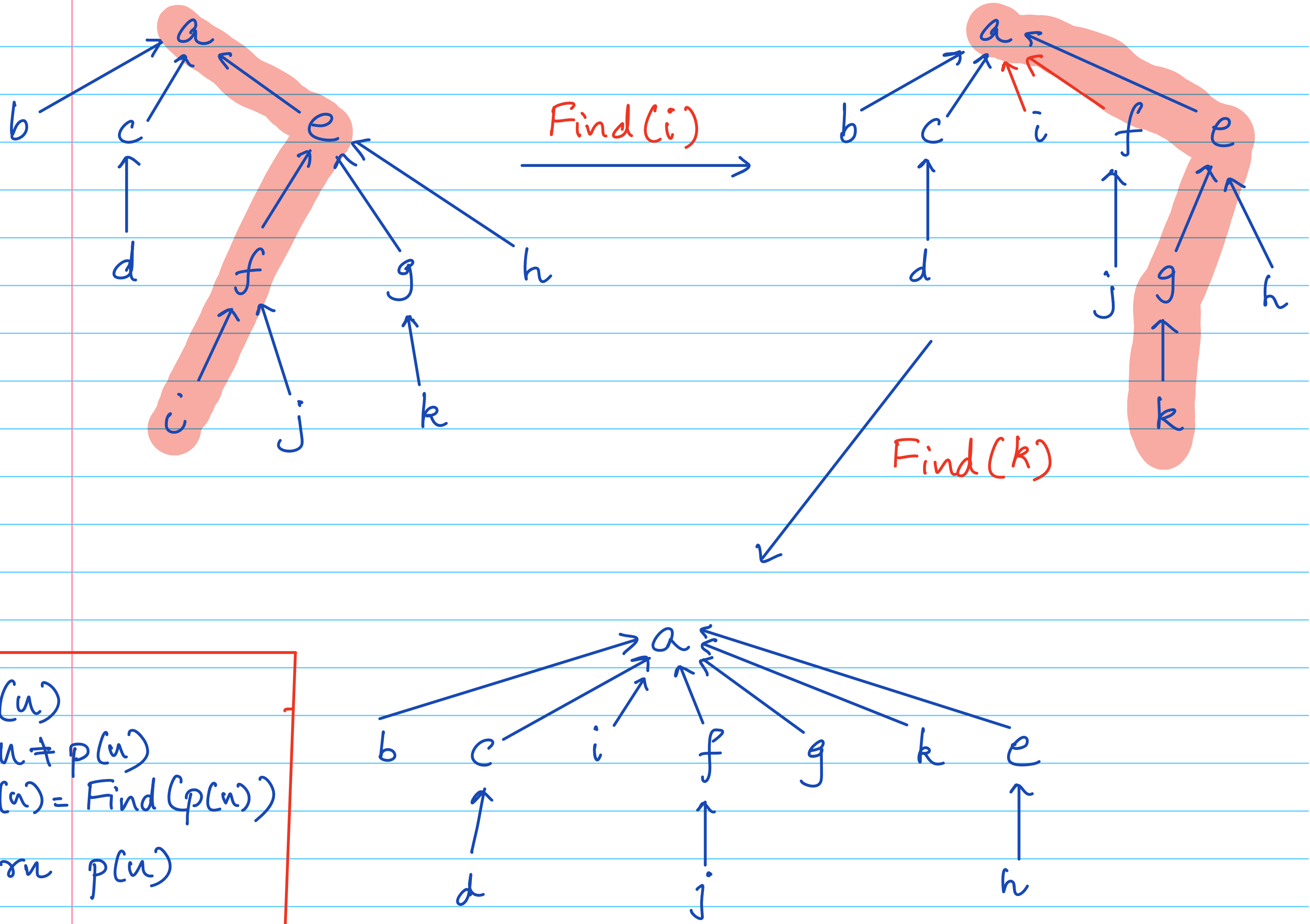
Union-by-rank with path compression

* Perform book-keeping during each find operation to reduce the depth of the tree.
  - For each node in the path from u to the root, connect the node directly to the root.



```
Find(u)
  If u ≠ p(u)
    p(u) = Find(p(u))
  return p(u)
```

Amortized analysis: Accounting method

* Maintaining Dynamic tables

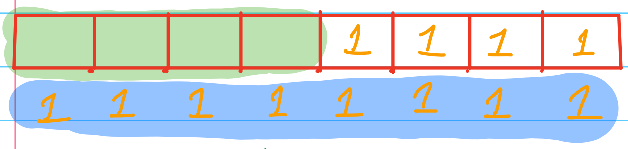
- How do we implement the vector class in STL efficiently
    - Lists of arbitrary sizes
    - O(1) - access times
    - Linked lists does not allow O(1)-access times
    - Resizing arrays seem expensive

Algorithmic idea:

- Initialize an array of some size m
- Whenever the array is full, create a new array of size 2m, copy the elements into the new array, deleting the old array

Can take O(n) in the worst-case for one insertion!

- Pay extra cost / operation
- Use the extra cost for a more expensive operation later



accounts for the resizing operation

For each item to be inserted, pay a cost of 3

- 1 for the insertion
- 1 for a later copy
- 1 for copying another elt

Total cost = $3n$