Case study: Union-find

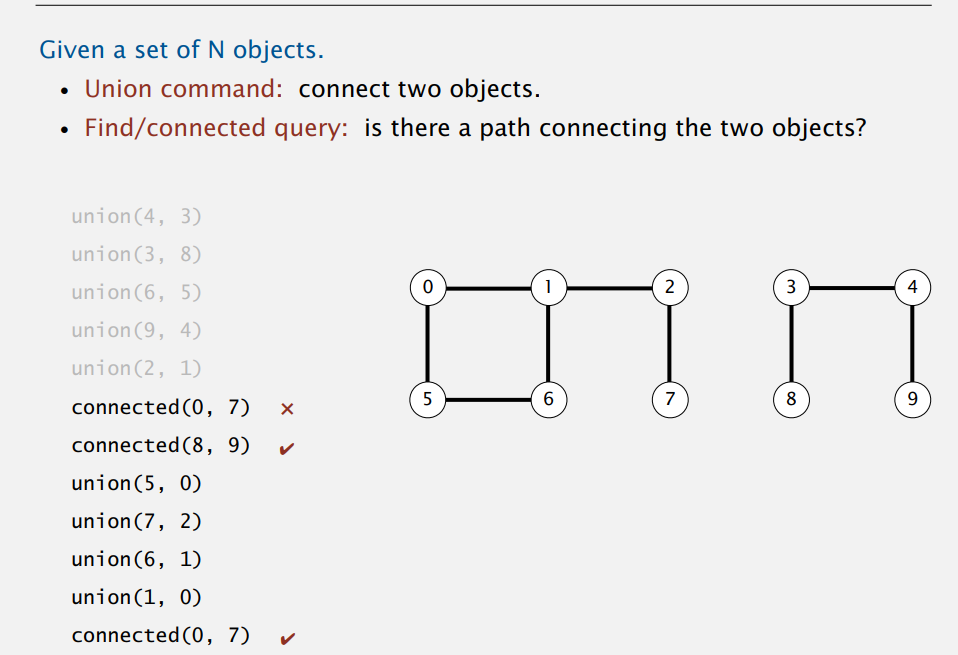
example: dynamic connectivity

description: the input is a sequence of pairs of integers, where each integer represents an object of some type and we are to interpret the pair p and q

as meaning "p is connected to q". we need to establish a network consists of such sequence.

task: according to the network, we need to write a program to make sure whether node p is connected to q or not.

pics:



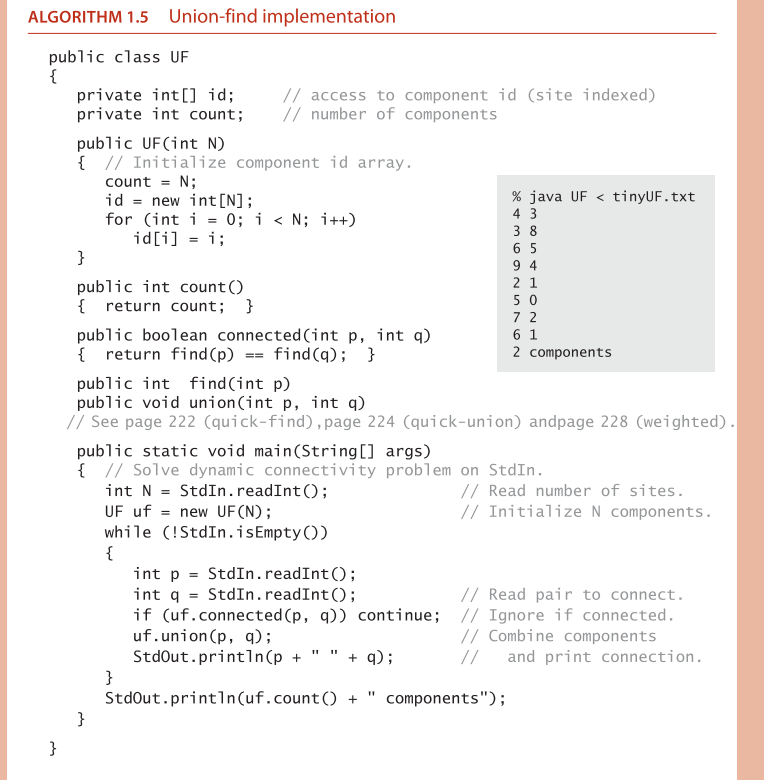
Step1: we just consider the number of integers is small.

We articulate the following API:

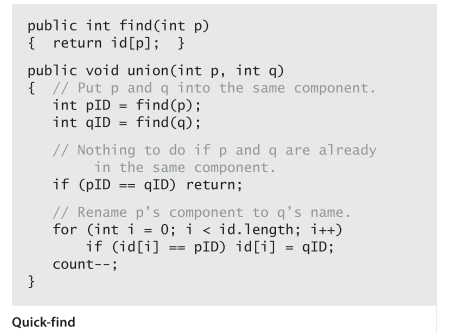
Public class UF:

|  |  |
| --- | --- |
| UF(int N) | Initialize N sites with integer names(0 to N-1) |
| void union(int p, int q) | Add connection between p and q |
| Int find(int p) | Component identifier for p(0 to N-1) |
| Boolean connected(int p, int q) | Return true if p and q are in the same component |
| Int count() | Number of components |

Programming:



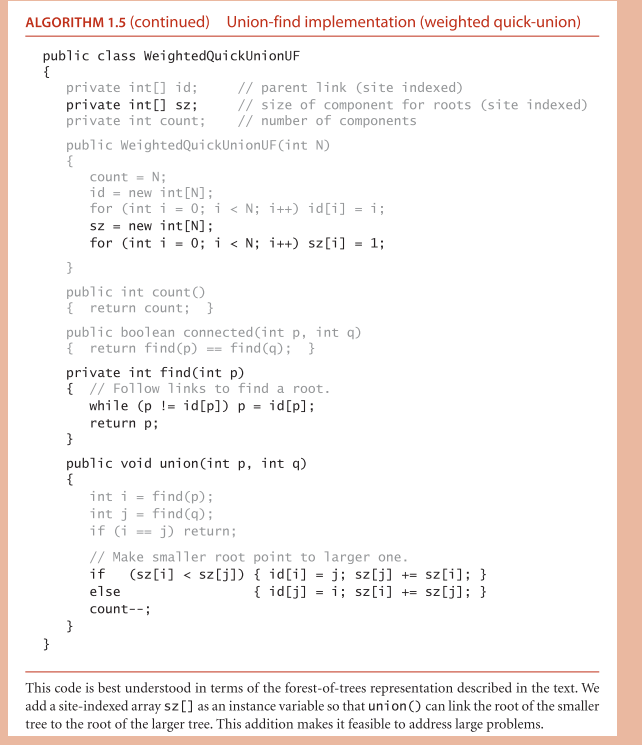
And the method of find and union is:



However, when the number of integers is too large, the efficiency is too slow.

We just consider Weighted quick-union.

The details of the algorithm can be found in the book. I just print some codes for tests:



And the process is as following pics:

