CS 61B Fall 2020 — Disjoint Sets and Asymptotics

Exam Prep Discussion 6: September 28, 2020

1 Disjoint Sets

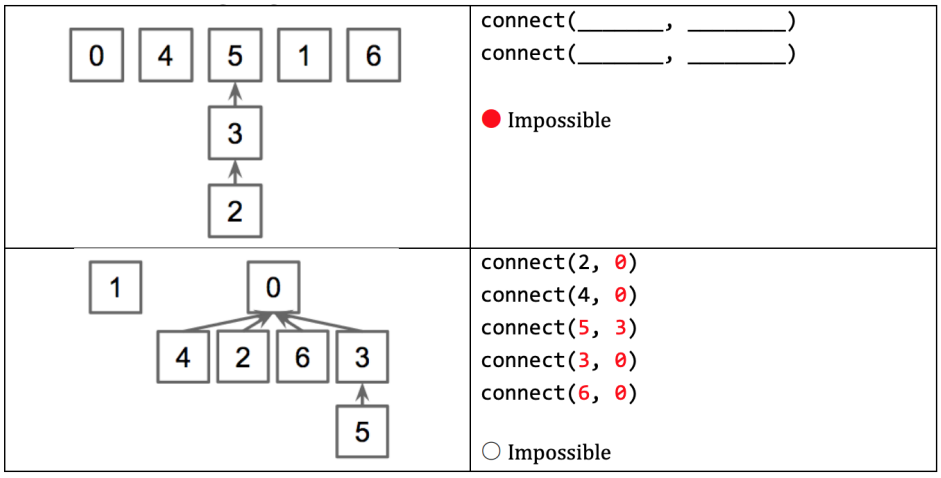
For each of the arrays below, write whether this could represent a weighted quick union with path compression and explain your reasoning.

```
      i:   0   1   2   3   4   5   6   7   8   9
      ---------------------------------------
A. a[i]:   1   2   3   0   1   1   1   4   4   5
B. a[i]:   9   0   0   0   0   0   9   9   9 -10
C. a[i]:   1   2   3   4   5   6   7   8   9 -10
D. a[i]: -10   0   0   0   0   1   1   1   6   2
E. a[i]: -10   0   0   0   0   1   1   1   6   8
F. a[i]:  -7   0   0   1   1   3   3  -3   7   7
```

(a) Impossible: has a cycle 0-1, 1-2, 2-3, and 3-0 in the parent-link representation.
(b) Impossible: the nodes 1, 2, 3, 4, and 5 must link to 0 when 0 is a root; hence, 0 would not link to 9 because 0 is the root of the larger tree.
(c) Impossible: tree rooted at 9 has height 9 >lg 10.
(d) Possible: 8-6, 6-1, 7-1, 5-1, 9-2, 3-0, 4-0, 2-0, 1-0.
(e) Impossible: tree rooted at 0 has height 4 >lg 10.
(f) Impossible: tree rooted at 0 has height 3 >lg 7.

2 Weighted Quick Union (Spring 2017, MT2)

Suppose we have a **weighted quick union** object. What calls to `connect(a, b)` produce the following trees? Assume that each **WQU** starts with all items disconnected. Fill in the "Impossible" option if the given tree is impossible. To tie break, the root of the left argument is placed below the root of the right argument.



Note that in the bottom example above, there is a little wiggle room in the order for when you do 'connect(6,0)' as long as 'connect(5,3)' comes before 'connect(3,0)'.

Suppose we add a new operation undo(a, b) that undoes an earlier Disjoint Sets connect operation. If connect(a, b) has never been called, then this method has no effect. For each of the implementations of Disjoint Sets, mark the corresponding box if it is impossible to add the undo operation without adding additional data structures (i.e. instance variables) to that implementation.

■Quick Union ■Quick Find ■Weighted Quick Union(WQU) ■WQU with Path Compression

For QuickFind, we have a series of sets that contain all the nodes within that set. Calling `undo(a,b)` on that would be impossible because we wouldn't know which node was in the set first and which other nodes were associated with each subset. For QuickUnion, we won't be able to tell whether the call to connect was `connect(a,b)` or `connect(a,some_child_of_b)`. For WQU with and without path compression, we can't tell whether the call was `connect(a,b)` or `connect(b,a)`.

3 It Begins (Spring 2017, MT2)

For each code block below, fill in the blank(s) so that the function has the desired runtime. Do not use any commas. If the answer is impossible, just write "impossible" in the blank.

```
public static void f1(int N) {      //Desired Runtime: Θ(N)
    for (int i = 1; i < N; i += 1) {System.out.println("hi");}
}
```

```
public static void f2(int N) {      //Desired Runtime: Θ(logN)
    for (int i = 1; i < N; i *= 2) {System.out.println("hi");}
}
```

```
public static void f3(int N) {      //Desired Runtime: Θ(1)
    for (int i = 1; i < N; i += N) {System.out.println("hi");}
}
```

4 Slightly Harder (Spring 2017, MT2)

Give the runtime of the following functions in Θ or O notation as requested. Your answer should be as simple as possible with no unnecessary leading constatns or lower order terms. For f5, your bound should be as tight as possible (so don't just put $O(N^{NM!})$ or similar for the second answer).

Θ(N^2logN)

```
public static void f4(int N) {
    if (N == 0) {return;}
    f4(N / 2);
    f4(N / 2);
    f4(N / 2);
    f4(N / 2);
    g(N); // runs in Θ(N^2) time
}
```

We will try a sample input, N = 4.

```
                                    f4(4)

        f4(2)                f4(2)                f4(2)                f4(2)

   f4(1)    f4(1)     f4(1)    f4(1)     f4(1)    f4(1)     f4(1)    f4(1)

f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0)
```

For the first layer, the time needed is dominated by `g(N)`, which runs in $\Theta(N^2)$ time. Therefore, the time taken at this level is 4^2.

At the second level, each call is 2^2. But there are four calls to `f4(2)`, so the total time is $(4)(2)^2 = 4^2$.

In general, at the i-th level, the total time is $(4^i)(N/2^i)^2$, which is equal to exactly N^2.

Therefore:

```
                                    f4(N)                                N^2

      f4(N/2)            f4(N/2)           f4(N/2)           f4(N/2)        (4*N^2)/4

   f4(1)    f4(1)     f4(1)    f4(1)     f4(1)    f4(1)     f4(1)    f4(1)     (8*N^2)/8

f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0) f4(0)f4(0)  (16*N^2)/16
```

Each layer takes total time N^2, and the number of layers is $log_2 N$ (one layer when $N = 2$, three layers when $N = 8$, etc.). The total time is $\sum_{i=0}^{\log N} N^2 = \Theta(N^2 \log N)$.

```
1  O(N)  public static void f5(int N, int M) {
2                  if (N < 10) {return;}
3                  for (int i = 0; i <= N % 10; i++) {
4                      f5(N / 10, M / 10);
5                      System.out.println(M);
6                  }
7              }
```

Again, we can think of this as a tree. Each call to `f5` does N%10 work. We can consider this to be contant, since even as `N` gets massive, `N\%10` will always be between 0 and 10. So let's assume the worst case, which means we assume `N\%10 = 9` all the time. This means we make 9 calls to `f5` each time. Now, how many levels does our tree have before it ends? We are dividing `N` by 10 each call and we end when we hit `N < 10`. So, there are `logN` levels to our tree. We can now sum up the work done at each level:

$$1 + 9 + 9^2 + ... + 9^{\log N} = \sum_{i=0}^{N} 9^i = \frac{1 - 9^{\log N}}{1 - 9}$$

Remember that asymptotics don't care about constant coefficients! So we can get rid of all those to get:

$$9^{\log N} \approx O(N)$$