Problem Set No. 2 Solutions

1. For either fast-find or fast-union, define the *link forest* to be the forest (collection of trees) formed by making node x a child of node y if a *link* operation is performed on sets with canonical elements x and y and y is made the canonical element of the combined set. The link forest is well-defined (it is a forest) and is the same for both set union methods if, as the problem assumes, all the links are done in the same way. In the fast-find method, if node x is an ancestor of node y in the link forest, then node x will point to node y via a canonical element pointer at some time during the sequence of set operations (indeed, when y becomes the canonical element of the set containing x). On the other hand, in the fast-union method, if a node x ever points to a node y via a parent pointer, node y must be an ancestor of x in the link forest, because linking and path compression both preserve this property. Thus if x ever points to y via a parent pointer in the fast-union method, x will also point to y via a canonical element pointer at some time in the fast-find method. Hence fast-find must make as many pointer changes (or more) as fast-union, if both do linking the same.

2. (Note that $k(x)$ for a nonroot x should be defined as $\lfloor \lg (r(p(x)) - r(x)) \rfloor$, *not* $\lfloor \lg r(p(x)) - r(x) \rfloor$.) All ranks are integer and non-negative. A link increases the maximum rank by at most 1, so the maximum rank is at most $n-1$, since there are at most $n-1$ links. Also, if y is the parent of x, $r(x) < r(y)$; this is preserved by both links and path compressions. Also the rank of a node can only increase when it is a tree root. Once a node becomes a nonroot, its rank is fixed.

   Let $p(x)$ be the parent of a node x. Since $r(p(x)) > r(x)$, $k(x) = \lfloor \lg (r(p(x)) - r(x)) \rfloor \geq \lfloor \lg\ 1 \rfloor = 0$ and $k(x) \leq \lfloor \lg(n-1-0) \rfloor = \lfloor \lg(n-1) \rfloor$. Thus $\phi(x) = \lfloor \lg(n-1) \rfloor - k(x)$ is an integer between 0 and $\lfloor \lg(n-1) \rfloor$ (inclusive). Furthermore $r(x)$ remains constant and $r(p(x))$ can only increase, so $k(x)$ can only increase, and $\phi(x)$ decrease, once x is a nonroot.

   The initial potential is 0 and the final potential is non-negative. A link that makes a node y the parent of a node x has an actual time of 1 and increases the potential only of x, by an amount of at most $\lfloor \lg(n-1) \rfloor$. Thus a link takes $O(\log n)$ amortized time. Consider a find followed by a compression along a path of k nodes. Divide the nodes into two types: those that are followed by another nonroot node on the same level further along the find path, and the rest. "The rest" totals at most $\lfloor \lg(n-1) \rfloor + 2$ nodes: the root of the find path, and the last node per level. (There are at most $\lfloor \lg(n-1) \rfloor + 1$ levels.) The actual cost of the find is k. I claim that for each node among "the rest," the potential drops by at least 1 after the find, so that the amortized cost of the find is at most $k - (k - (\lfloor \lg (n-1) \rfloor + 2)) = \lfloor \lg (n-1) \rfloor + 2$. This gives the desired bound.

To verify the claim, let x be a node of level k on a find path followed by another node y of level k along the find path. Before the find, $r(p(x)) - r(x) \geq 2^k$, $r(p(y)) - r(y) \geq 2^k$, and $r(x) < r(p(x)) \leq r(y) < r(p(y))$. The parent of x after the compression, say z, has $r(z) \geq r(p(y))$. This means that $r(z) - r(x) \geq 2^k + 2^k = 2^{k+1}$, and the level of x increases by at least 1 because of the compression. Correspondingly, the potential of x decreases by at least 1.

3. (Sketchy sketch).

Divide the problem into two cases, $m \geq n$ and $m \leq n$. For the case $m \geq n$, assign a nonroot node x an extra potential of $\max\{0, \lceil m/n \rceil - r(p(x))\}$. For the case $m \leq n$, assign a nonroot node a potential of zero unless its rank is at least $\lceil n/m \rceil$; in the latter case assign its potential according to the formula in the class notes. One then needs to show that the number of nodes ever assigned non-zero potential is at most m, the total potential increase due to links is $\mathrm{O}(\mathrm{m}\,\alpha(\mathrm{n}))$, and the amortized cost of a find is $\mathrm{O}(\mathrm{n}/\mathrm{m} + \alpha(\mathrm{n}))$.