

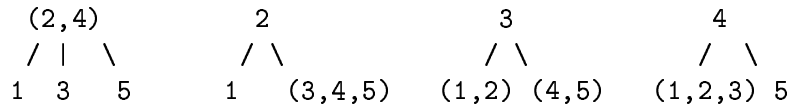
HW7 Solution Sketches

April 1, 1999

15.1-5 Given x , we just need to size for the element y such that $size[y] = size[x] + i$. Using algorithm OS-SELECT, this takes time $O(\lg n)$.

[Grader: Jason Howes]

19.1-3 If $t = 2$, every node root has outdegree 2, 3, or 4 (this is a 2-3-4 tree) and each node has 1–3 keys. Since every leaf has the same depth this means that the root has 2 keys and three children, each of which has 1 key, or the root has 1 key and its children have the remaining four keys. This gives us the following four possible trees:



[Grader: Joy Alamgir]

19.1-4 If the minimum degree is t , we can store at most $2t - 1$ keys at every node and each node has outdegree at most $2t$. Thus, there are at most $(2t)^d$ nodes at depth d . This means that the total number of keys is at most

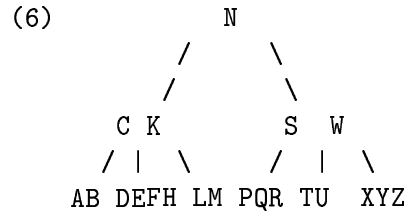
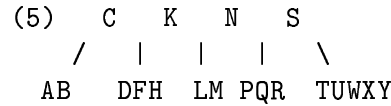
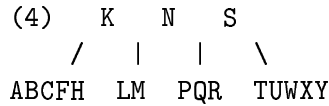
$$(2t - 1) \sum_{d=0}^h (2t)^d = (2t - 1) \frac{(2t)^{d+1} - 1}{2t - 1} = 2t^{d+1} - 1.$$

This bound is achievable if we in fact have $2t - 1$ keys at each node and each internal node has outdegree $2t$.

[Grader: Joy Alamgir]

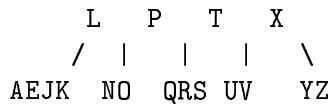
19.2-1 Since $t = 3$, a node must split if it has 5 keys and we are about to insert a new key into it. Here is what the trees look like just before a node must split and the final configuration:



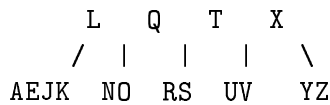


[Grading: 2 points for the final tree; 1 for each of the preceding ones.]
 [Grader: Rami Baalbaki]

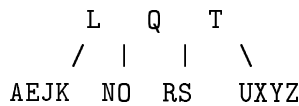
19.3-1 (a) After deleting C:



(b) After deleting P:



(c) After deleting V:

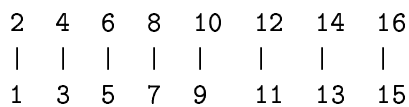


[Grading: 2 points for each case. If (b) is consistent given (a), then full credit is given. Likewise if (c) is consistent given (b).]

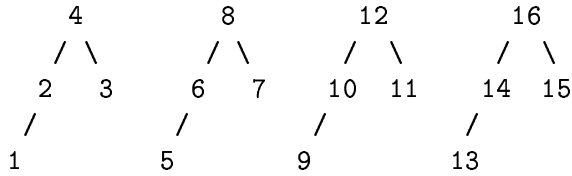
[Grader: David Welte]

22.3-1 As pointed out on netnews, we have to assume that the UNION here is the one used in the text (or, equivalently, we have to do FINDs on the arguments of the UNION, since in general, the arguments are not the roots of trees. Sorry about the confusion here.

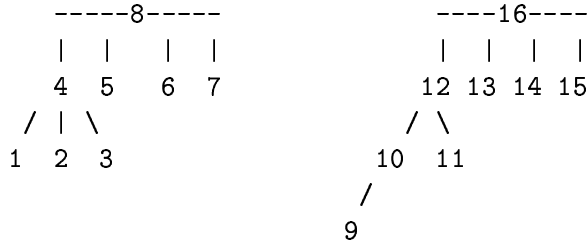
After line 4, we have the 8 trees:



After line 6, we have the 4 trees:

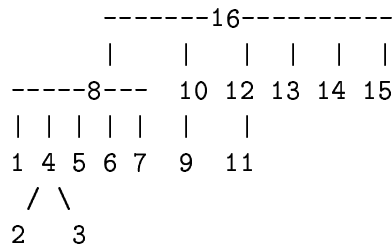


After line 8, we have the 2 trees:

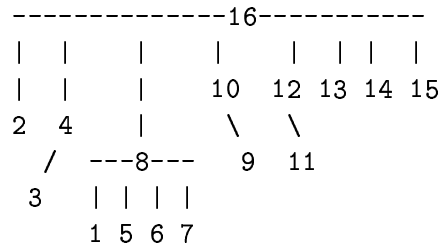


Note that 1 is the child of 4 because in the course of doing UNION(1,5), we had to do a FIND on 1, which resulted in path compression. For similar reasons, 5 is the child of 8, and 13 is the child of 16.

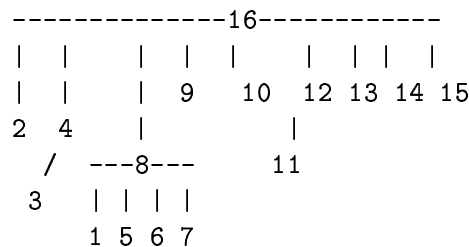
After line 9 we have 1 tree:



After line 10, we do path compression for 2 and 4 after finding 2, so we get the following hard-to-draw tree:



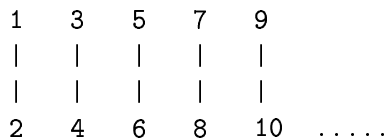
After line 11, we do path compression for 9, so we get the following even harder-to-draw tree:



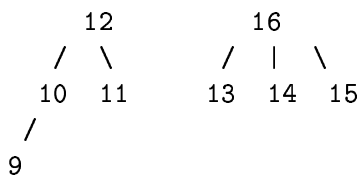
[Grading: 1 each for the trees that arise after lines 4, 6, 8, and 9. 2 each for the results of path compression after lines 10 and 11.]

Common mistakes:

1. A lot students don't do the path compression when executing $\text{FIND}(x)$.
2. Some students didn't notice that the default root of $\text{UNION}(x, y)$ is y if $\text{rank}(x) = \text{rank}(y)$, and they end up with trees like these:



3. When executing $\text{UNION}(x_{11}, x_{13})$ (where this version of UNION is the one in the text), we must really do $\text{UNION}(\text{FIND}(x_{11}), \text{FIND}(x_{13}))$, using the UNION presented in class. Doing $\text{FIND}(x_{11})$ and $\text{FIND}(x_{13})$ results in the following trees (after compression):



We observe that the left tree is deeper. But FIND does not doesn't change the rank, so these two trees still have the same rank (since they had the same rank before we did the FIND). That's why we obtain the tree with 16 as its root. Several students made 12 the root.

[Grader: Lantian Zheng]

Extra problem 1: Suppose we are given a sequence σ of K MAKE-SETS + M FINDS + N UNIONS, where all the FINDS are performed at the end. Let σ' be the prefix of σ consisting of the MAKE-SETS and UNIONS. Thus, $\sigma = \sigma' \text{FIND}(v_1) \dots \text{FIND}(v_k)$. Clearly it takes time $O(K + M)$ to perform all the operations in σ' . After this is done, we have a forest (collection of trees), say T_1, \dots, T_k , where the trees have K nodes altogether. Let r_1, \dots, r_k be the roots of these trees. Since a tree with n nodes has $n - 1$ edges (one for every node but the root, going to its parent), the total number of edges in these trees is $K - k$.

Now consider what happens to these trees after we perform $\text{FIND}(v_1), \dots, \text{FIND}(v_j)$, for $j \leq M$. We still have k trees, T_1^j, \dots, T_k^j , where T_i^j has the same nodes and the same root as T_i . For any node $v \in T_i$, either it has the same parent in T_i and T_i^j , or its parent in T_i^j is r_i (if v was involved in some compression). To do the accounting, as we do $\text{FIND}(v_1), \text{FIND}(v_2), \dots, \text{FIND}(v_j)$, mark the edges in the original tree that get compressed. (That is, if v is in T_i and, as a result of compression following a FIND, the parent of v is r_i in T_i^j , we mark the edge from v to its parent in T_i .) Note if an edge in T_i is marked, so are all the edges from there on up to r_i , because of the way path compression works.

If v_{j+1} is in tree T_i , then the cost of $\text{FIND}(v_{j+1})$ is proportional to the cost of the path from v to r_i in T_i^j . The length of this path is 1 + the number of currently unmarked edges on the path from T_i . Thus, the total cost of all the M FINDS is $O(K + M)$ (since there are only $K - k$ edges to mark). That means the total cost of σ is $O(K + M + N)$. and FIND are $O(1)$ operations.

Grading: 1 point each for mentioning that UNION and FIND are $O(1)$ operations. 4 points for the proof that M FINDS take $O(K + M)$ (or $O(N + M)$).

Common mistakes:

1. Many students did an “average” case analysis of the running time of FIND. For some M number of FIND’s, where M might even be a small number, this does not make sense at all. This is probably a hangover from the probabilistic “expected” analysis of data-structures like skip lists. The homework question does not mention an average case analysis – by default, that means you should do a *worst case analysis*.
2. Many students proved that M finds take $O((K + M)lg^*K)$ (or something similar), following up with the assertion that the lg^*K is constant for all practical purposes. It might be, but not for this proof. The question did not say anything about making such an assumption, so you can’t (neither do any of the proofs given in class).
3. Some students made the mistake of blindly following the proof given in class for the general case (where the FIND’s are not at the end) - this was unnecessary (and leads you to the wrong assumption about lg^*K); the proof to the homework problem is much simpler.

[Grader: Indranil Gupta]

Extra problem 2: SKIPDELETE(S, k)

```
1  $y \leftarrow$  SKIP-SEARCH( $S, k$ )
2 if  $y \neq x$ 
3   then return “ $x$  is not in list”
4   else while  $y \neq$  NIL
5     do after[before[ $y$ ]]  $\leftarrow$  after[ $y$ ]
6       before[after[ $y$ ]]  $\leftarrow$  before[ $y$ ]
7        $y \leftarrow$  above[ $y$ ]
```

[Grader: Jason Howes]