15-750 Algorithms (Spring 2002) **Prof. Manuel Blum**

Homework #3 **Solutions**

(problems 1 & 2 are due to Danny Sleator)

Problem 1: Path reversal in Union-Find (20 points)

Recall from the lecture on Union-Find the process of *path compression*: after finding the root v of the tree containing u in a `find(`u`)`, we traverse the path from u to v one more time and change the parent pointers of all vertices along the path to point directly to v (Figure 1). This technique (together with

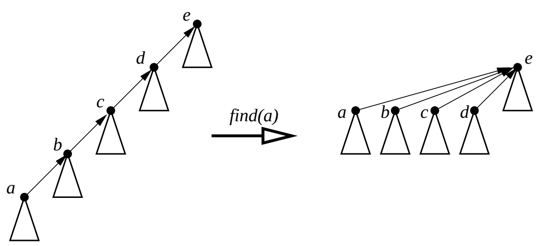


Figure 1: Path compression.

making the root of the smaller tree a child of the root of the larger in `union` operation) allowed a sequence of m `union` and `find` operations starting with n singleton sets to be performed in $\mathcal{O}((m+n)\alpha(n))$ steps, where $\alpha(n)$ is the inverse of Ackermann's function.

One drawback of such path compression is the fact, that the path from u to v is traversed twice. Consider the following variation of the algorithm, called *path reversal*: traverse the path only once, and make every node on the path point to the *first* node on the path (Figure 2). Note that this changes the root of the tree. Is the method of path reversal a worthwhile modification to Union-Find, i.e., can we still guarantee

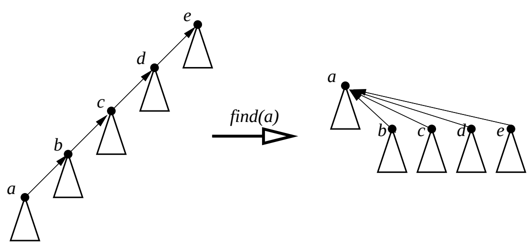


Figure 2: Path reversal.

the $\mathcal{O}((m+n)\alpha(n))$ bound? Justify your answer.

Hint: Think binomial tree.

No, the method of path reversal cannot guarantee the $\mathcal{O}((m+n)\alpha(n))$ bound. To prove it, we show an example of m operations, which starting from n singleton sets require $\Omega(n + (m-n)\log n)$ steps.

Recall the structure of a binomial tree B_k — it consists of 2 binomial trees B_{k-1}, where the root of one is attached as a child to the root of the other, and B_0 is just a single node. Wlog. assume that $n = 2^k$. When starting with n singleton sets we can obtain B_k by performing $n-1$ `union` operations: perform $n/2$ `union`'s on pairs of singleton sets to get $n/2$ B_1's, then $n/4$ `union`'s on pairs of B_1's to get $n/4$ B_2's, and so on. Now consider a `find(`v`)`, where v is the deepest node in B_k. Clearly, this operation will take at least $k = \log n$ steps (because B_k has depth k). Moreover, it easy to see that `find(`v`)` with path reversal will result in a structure which is also a binomial tree with k nodes! Hence, we can again perform `find` on the deepest node in this new tree, which will again take $\log n$ steps and yield a binomial tree with k nodes, and so on. Hence the total cost of constructing the binomial tree B_k and performing $m - n$ `find`'s on the deepest node is at least $\Omega(n + (m - n) \log n)$. □

Problem 2: Quads and Blobs (30 points)

Consider a very reactive elementary particle called a quad. Quads react with one another to form blobs of various sizes. A blob results when two quads collide, or when a quad and a blob collide. (When two blobs collide they bounce apart unaffected.) The size of a blob is the number of quads it has absorbed. (The smallest blob has size two.) A blob of size 5 is unstable and instantaneously explodes into two blobs of size 2, a gamma ray, and two new quads. The following reactions summarize the situation: (Q is a quad, and B_i is a blob of size i.)

$$\begin{aligned}
&Q + Q \to B_2 \\
&Q + B_2 \to B_3 \\
&Q + B_3 \to B_4 \\
&Q + B_4 \to 2B_2 + \gamma + 2Q
\end{aligned}$$

A quiescent configuration is one containing no quads. Suppose that into a quiescent configuration a new quad is introduced. This causes a chain reaction that produces a number of gamma rays and new quads. Eventually a new quiescent state is reached.

1. Choose a constant c and a potential function (which depends on the number of various particles present). Prove the following three statements about these choices:
   (a) The amortized number of gamma rays produced when a new quad is introduced is at most c.
   (b) Starting from an empty configuration, the introduction of k quads causes the emission of at most ck gamma rays.
   (c) Show that your value of c is the smallest one satisfying parts (a) and (b).
2. *(Extra credit)* Suppose you were given a set of n different types of particles (one of which is the quad), and m reactions involving these particles. The left hand side of each reaction is a combination of particles required for it to occur, and the right hand side is the outcome of the reaction. Each reaction also produces zero or more gamma rays. How would you approach the problem of determining if an infinite chain reaction is possible? If such a reaction is impossible, how would you determine the minimum constant c satisfying parts 1a and 1b above.

1. Let n_1 denote the number of quads, and n_i the number of B_i blobs, $i = 2, 3, 4$. Consider the potential function $\Phi(n_1, n_2, n_3, n_4) = n_1 + n_3 + 2n_4$, and take the constant $c = 1$.

   (a) The amortized number of gamma rays produced when a new quad is introduced is at most c.

   Proof. First notice, that the potential is never negative, does not increase after any of the four reactions, and decreases by 1 each time a γ ray is produced. Therefore, if a sequence of reactions produces k γ rays, the potential is reduced by at least k.

   Let n_γ denote the number of γ rays produced after introducing a new quad into a quiescent configuration (i.e. the real "cost" of introducing one quad). Then the amortized number of γ rays produced is equal $\hat{n}_\gamma = n_\gamma + \Delta\Phi$, where $\Delta\Phi$ denotes the change of potential between the initial and the final quiescent configurations.

   Introduction of a quad increases the potential by 1, and production of n_γ γ rays reduces the potential by at least n_γ, so $\Delta\Phi \leq 1 - n_\gamma$, and $\hat{n}_\gamma = n_\gamma + \Delta\Phi \leq 1$. □

   (b) Starting from an empty configuration, the introduction of k quads causes the emission of at most ck gamma rays.

   Proof. The potential of an empty configuration is equal zero, and after introduction of k quads the potential is at most k. Since each γ ray reduces the potential by 1, and since the potential is never negative, in total at most k γ rays are produced. □

   (c) Show that your value of c is the smallest one satisfying parts (a) and (b).

   Proof. It easy to see, that once a B_4 blob is present in a configuration, an introduction of a single quad can produce a γ ray and a new B_4. Hence, if we consider an empty initial configuration and introduce k quads, it is possible to produce $k - 4$ γ rays (the first 4 quads are used to create initial B_4 blob, and each next quad yields exactly one γ ray). This means, that it is possible to get $(k-4)/k$ γ rays per introduced quad, which is larger than any constant $c < 1$ as k goes to infinity. □

2. *(Extra credit)* This general setting corresponds to so called Petri Nets. For a study of its relation to data structures see *"Data Structures and Cost-bounded Petri Nets"* by D. Sleator (available at `http://www.cs.cmu.edu/~sleator/papers/reactions.ps`).

Problem 3: Variations on Splay Trees (extra extra credit)

Recall the rules of `splay(`x`, `S`)`:

(i) if x has a parent but no grandparent, just `rotate(`x`)`

(ii) if x has a parent y and a grandparent, and if x and y are either both left children or both right children, first `rotate(`y`)`, then `rotate(`x`)`

(iii) if x has a parent y and a grandparent, and if one of x, y is a left child and the other is a right child, first rotate(x), then rotate(x) again

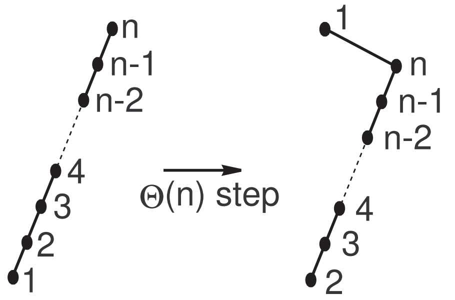
A smart programmer wishes to simplify an implementation of splay trees, and proposes three new sets of the rules:

1. replace rules (ii) and (iii) by:
   if x has a parent y and a grandparent, first rotate(x), then rotate(x) again
   (this case was shortly discussed in the lecture)
2. replace rules (ii) and (iii) by:
   if x has a parent y and a grandparent, first rotate(y), then rotate(x)
3. replace rule (iii) by:
   if x has a parent y and a grandparent, and if one of x, y is a left child and the other is a right child, just rotate(x)

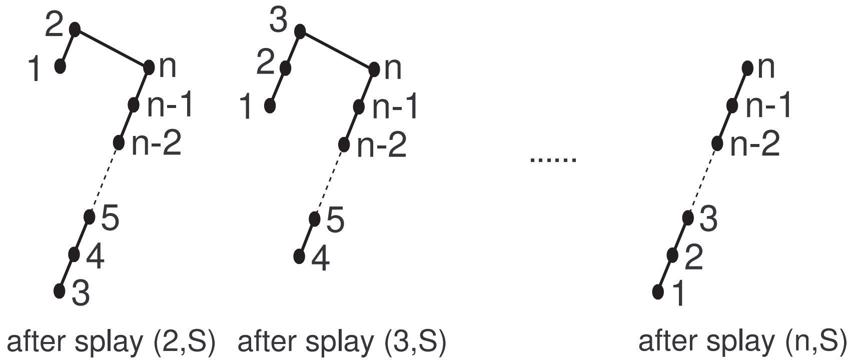
Discuss the consequences of each variation on the performance. Two performances are considered roughly "the same", if each is $\mathcal{O}(\cdot)$ of the other. Justify your claims with analytic arguments.

(solution due to Hua Zhong, Takayuki Osogami and Ting Liu)

1. The cost of the replaced rule is greater than the original one. We can prove this by just giving a counter example. Apply splay(1,S) to the following tree S.



   We can easily prove that after splay(1,S), the tree will look like the right-hand one by induction. (We will not show the proof here.) Continue doing splay(2,S), ..., splay(n,S).



The overall cost of the n splay operation is $\Theta(n + (n-1) + \cdots + 1) = \Theta(n^2)$. We continue this loop again and again; then, the time required to perform m operations is $\Theta(mn)$, which is greater than $O(m \log n)$. $\square$

2. *(no comment)*

3. We will prove that the performance is roughly "the same" as the original splay tree.

   Note: Notation and proof is similar to the textbook.

   We will first consider the following lemma.

   Lemma *Each operation splay(x,S) requires no more than*

   $$4(\mu(S) - \mu(x)) + 1$$

   credits to perform the operation and maintain the credit invariant.

   If this lemma holds, in the same way as the proof of theorem 12.3 in the textbook., we can conclude that at most $4\lfloor \log n \rfloor + 1$ credits are needed for each splay operation, and the total time required for a sequence of m such operations is $O(m \log n)$. Therefore, it is sufficient to prove the lemma.

   Proof of the Lemma:

   (i) It is immediate from the proof in the textbook that an operation (i) pays at most $4(\mu'(x) - \mu(x)) + 1$ credits.

   (ii) It is also immediate from the proof in the textbook that an operation (ii) pays at most $4(\mu'(x) - \mu(x))$ credits.

   (iii) We will consider three cases:

   (A) x has no grand-grandparent:

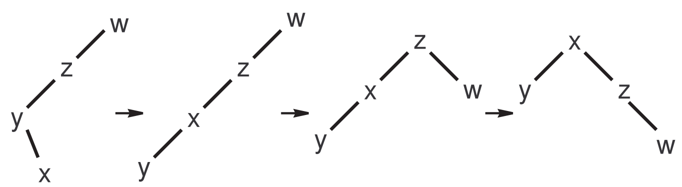
   In this case, rotate(x) is followed by another rotate (x), because after the first rotate(x), x has no grandparent. However, this two consecutive operations is equivalent to the original operation of type (iii). So, it pays at most $4(\mu'(x) - \mu(x))$ credits.

(B) x has a parent y, a grandparent z, and a grand-grandparent w, where x and z is a right child and y is a left child, or x and z is a left child and y is a right child:

In this case, rotate(x) is followed by another rotate (x), because after the first rotate(x), x has a grandparent which is a right (left, respectively) child if x is a right (left, respectively) child. However, this two consecutive operations is equivalent to the original operation of type (iii). So, it pays at most $4(\mu'(x) - \mu(x))$ credits.

(C) x has a parent y, a grandparent z, and a grand-grandparent w, where x is a right child and y and z is a left child, or x is a left child and y and z is a right child:

In this case, rotate(x) is followed by rotate(y) and rotate(x), because after the first rotate(x), x has a grandparent which is a right (left, respectively) child if x is a left (right, respectively) child.



We will prove that it pays at most $4(\mu'(x) - \mu(x))$ credits for each of the two cases:

(a) $\mu(x) = \mu(w)$:

Since y and z were ancestors of x and descendants of (w), $\mu(x) = \mu(y) = \mu(z) = \mu(w)$. Since $\mu'(x) = \mu(w)$ and both y and z becomes children of x, at least one of $\mu'(y)$ and $\mu'(z)$ is strictly less than $\mu'(x)$ the other one is at most $\mu'(x)$. Since w becomes an ancestor of x, $\mu'(w) \leq \mu'(x)$. Therefore,

$$\begin{aligned}
&\mu'(x) + \mu'(y) + \mu'(z) + \mu'(w) - \mu(x) - \mu(y) - \mu(z) - \mu(w) = \\
&= \mu'(x) + \mu'(y) + \mu'(z) + \mu'(w) - 4\mu'(x) \\
&= (\mu'(y) - \mu'(x)) + (\mu'(z) - \mu'(x)) + (\mu'(w) - \mu'(x)) \\
&\leq (\mu'(y) - \mu'(x)) + (\mu'(w) - \mu'(x)) \\
&< 0
\end{aligned}$$

Since the potential function decreases at least one, we can pay it for the cost of operation. So, it pays at most $4(\mu'(x) - \mu(x))$ credits.

(b) $\mu(x) < \mu(w)$:

Since z were ancestors of x, $\mu(z) \geq \mu(x)$. Since z and w becomes descendants of x, $\mu'(z) \leq \mu'(x)$ and $\mu'(w) \leq \mu'(x)$. Therefore,

$$\begin{aligned}
\mu'(x) + \mu'(y) + \mu'(z) + \mu'(w) - \mu(x) - \mu(y) - \mu(z) - \mu(w) = \\
= \quad & \mu'(y) + \mu'(z) + \mu'(w) - \mu(x) - \mu(y) - \mu(z) \\
\leq \quad & \mu'(x) + \mu'(x) + \mu'(x) - \mu(x) - \mu(x) - \mu(x) \\
= \quad & 3(\mu'(x) - \mu(x))
\end{aligned}$$

Since $\mu(x) < \mu(w) = \mu'(x)$, we can use $\mu'(x) - \mu(x)$ to pay for the cost of operation. So, it pays at most $4(\mu'(x) - \mu(x))$ credits.

This completes the proof of the lemma. $\square$