ma
the
ma
tisch

cen
trum

AFDELING ZUIVERE WISKUNDE
(DEPARTMENT OF PURE MATHEMATICS)

ZN 68/76 DECEMBER

T.P. VAN DER WEIDE

A NOTE ON THE PATH-COMPRESSION-RULE

amsterdam 1976

stichting
mathematisch
centrum

AFDELING ZUIVERE WISKUNDE ZN 68/76 DECEMBER
(DEPARTMENT OF PURE MATHEMATICS)

T.P. VAN DER WEIDE

A NOTE ON THE PATH-COMPRESSION-RULE

2e boerhaavestraat 49 amsterdam

Printed at the Mathematical Centre, 49, 2e Boerhaavestraat, Amsterdam.

The Mathematical Centre, founded the 11-th of February 1946, is a non-profit institution aiming at the promotion of pure mathematics and its applications. It is sponsored by the Netherlands Government through the Netherlands Organization for the Advancement of Pure Research (Z.W.O).

AMS(MOS) subject classification scheme (1970): 68A20, 05C05

A note on the path-compression-rule

by

T.P. van der Weide*)

ABSTRACT

Two variations of the path-compression-rule are considered. It turns out that the original path-compression-rule yields a better running time.

KEY WORDS & PHRASES: *Algorithm, complexity, equivalence, partition, set union, tree, path-compression-rule.*

*TWI Rijks Universiteit Leiden.

1. INTRODUCTION

Consider the following operations on sets:

- UNION (A,B,C) equivalent with: $C := A \cup B$, provided: $A \cap B = \emptyset$
- FIND (x) delivers the set currently containing x

Let U be a finite, non-empty set, where $|U| = n$. We consider a process, consisting of (n-1) UNION's and m FIND's ($m \geq n$). The process starts with the set singletons in U. Note that during this process U will remain partitioned in a number of subsets. We will represent a member of such a partition by a rooted tree (the arrow pointing to the root). We assume that UNION has been implemented using the weighted-union-rule, and FIND using the path-compression-rule [1]. The maximum computing time for all such processes, consisting of (n-1) UNION's and m FIND's will be called t(m,n).
Tarjan has shown [1]:

$$t(m,n) \in \Theta(m\alpha(m,n))$$

where α is the functional inverse of the Ackermann-function (see [1])
The set $\Theta(f)$ is the set of all functions g satisfying:

$$\exists_{c_1, c_2 > 0} [c_1 . f \leq g \leq c_2 . f]$$

This notation has been suggested by Knuth ([2]).

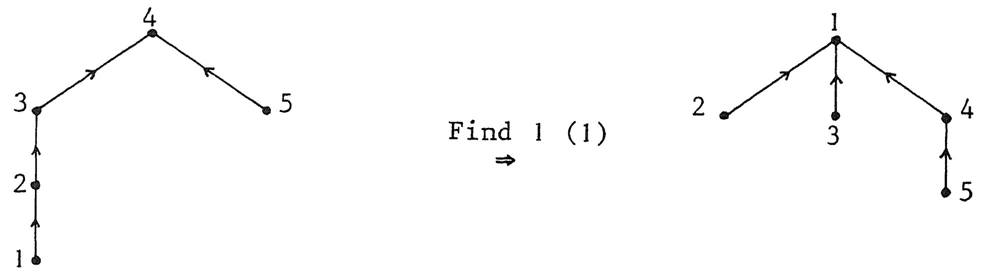
2. THE PATH-COMPRESSION-RULE

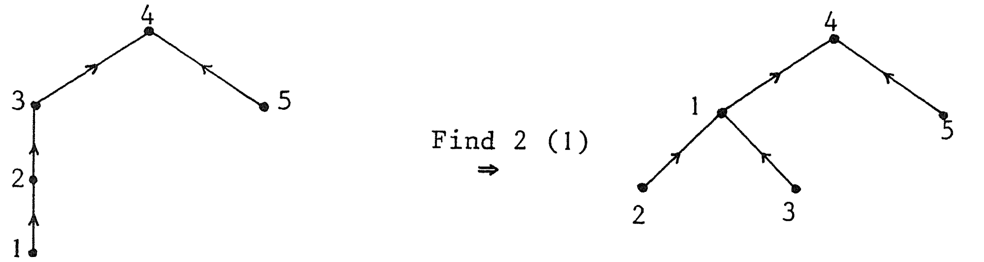
A possible disadvantage of the path-compression-rule is, that the path from the node in question to the parent root should be travelled twice. An amended algorithm with no backtracking could be implemented thus ([3]):

(i)
```
proc  FIND1 (x: node);
      u, v: node;
      u:= x; v:= u. father;
      while  v ≠  nill  do  u.father:= v.father; v.father:= u;
                            v:= u.father
                        od
corp
```

```
(ii) proc FIND2 (x: node);
          u, v, w: node;
          u:= x; v:= u.father; w:= v.father;
          while w≠ nill do u.father:= w; v.father:= u;
                           v:= u.father: w:= v.father
                        od
     corp
```

EXAMPLE: we will give an example of how FIND1 and FIND2 work:





(End of example)

However, in both cases we can show

$$t(m,n) \in \Omega(m \log n)$$

REMARK. $\Omega(f)$ is the set of all functions g satisfying

$$\exists_{c>0}\ [g \geq c.f] \qquad \text{(see [2])}$$

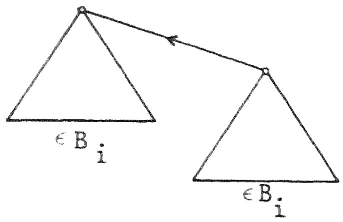
3. THE ANALYSIS

We shall construct a UNION-FIND-process, which has a computingtime in $\Theta(m \log n)$.

We introduce the following classes of trees B_i ([4], exercise 4.14):

(o) each tree of class B_0 will consist of a single node

(i) each tree of class B_{i+1} will be built up from two trees out of the class B_i in the following way ($i \geq 0$):



$\} \in B_{i+1}$

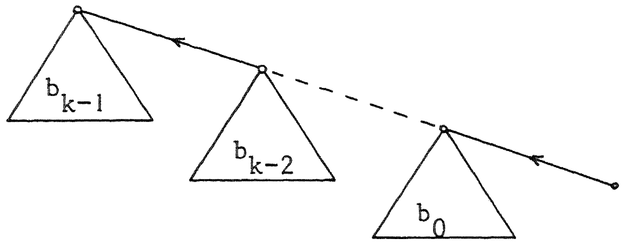
We have the following propositions:

<u>LEMMA 1.</u> *For each* $k > 0$ and $b_k \in B_k$: $- \; |b_k| = 2^k$

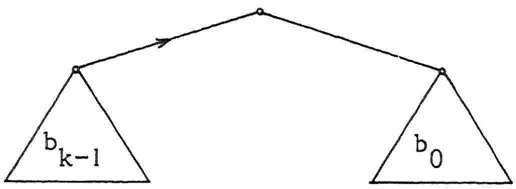
$- \; \text{depth}(b_k) = k$

<u>LEMMA 2.</u> *For each* $k \geq 0$ *and* $b_k \in B_k$ *there exist* $b_0 \in B_0$,---, $b_k \in B_{k-1}$ *such that*:

$b_k =$



<u>LEMMA 3.</u> *For each* $k \geq 0$ *and* $b_k \in B_k$ *there exist* $b_0 \in B_0$,---, $b_{k-1} \in B_{k-1}$ *such that*:



These three lemma's can easily be proved by induction on k.
It will be clear that any tree from B_k $(k \geq 0)$ may be constructed with a series of UNION-instructions.

<u>LEMMA 4.</u> *Let* $k \geq 0$ *and* $b_k \in B_k$,
and let x *be the unique node in* b_k *with depth* k
then: FIND1(x) *and* FIND2(x) *transform* b_k *into a tree from* B_K

<u>PROOF</u>. Follows directly from lemma's 1-3

As a consequence we can construct a UNION-FIND-process with computingtime in $\Theta(m \log n)$

REFERENCES

1. TARJAN, R.E., *Efficiency of a good but not linear set union algorithm*, J. Assoc. Comput. Mach., 22 (1975), 215-225.

2. KNUTH, D.E., *Big omicron and big omega and big theta*, Sigact news, 8 (1976), 18-24.

3. LEEUWEN, J.v., *Private communication.*

4. AHO, A.V., J.E. HOPCROFT & J.D. ULLMANN, *The Design and Analysis of computer Algorithms*, Addison-Wesley, Reading, Mass., 1974.