SIAM J. COMPUT.
Vol. 14, No. 1, February 1985


010



# ON THE EXPECTED PERFORMANCE OF PATH COMPRESSION ALGORITHMS*

ANDREW C. YAO†

**Abstract.** We consider the expected running time of an equivalence algorithm using the path compression rule (but not the weighting rule). An $O(n)$ expected running time is proved for the execution of a random equivalence program in the Spanning Tree Model.

**Key words.** equivalence program, expected running time, path compression, set merging, spanning tree model

**1. Introduction.** Let $S$ be a set of $n$ elements. An *equivalence program* $\sigma$ on $S$ is a sequence of *equivalence instructions* $(x[1] \equiv y[1], x[2] \equiv y[2], \cdots, x[m] \equiv y[m])$ with each $x[i], y[i] \in S$. Starting with $n$ equivalent classes each containing one element, an equivalence instruction $x[i] \equiv y[i]$ asks whether $x[i]$ and $y[i]$ currently belong to different equivalent classes, and if so requests that the two classes be merged. Equivalence programs have many applications, such as the processing of EQUIVALENCE statements in FORTRAN [4]. A common method to implement an equivalence program is by using a *set merging scheme.* A set merging scheme (see AHU [1], Tarjan [7]) maintains the equivalence classes as sets and processes commands of the forms FIND $(x)$ and UNION $(A, B)$. The command FIND $(x)$ requires that the name of the set containing $x$ be returned, and the command UNION $(A, B)$ asks that the two sets with names $A$ and $B$ be merged into one. To implement an equivalence program using a given set merging scheme, one need only replace each equivalence instruction $x[i] \equiv y[i]$ by the sequence FIND $(x[i])$, FIND $(y[i])$, UNION $(A, B)$, where $A$, $B$ are the names of the sets containing $x[i]$, $y[i]$ (omit the UNION if $A = B$). In this paper, we are interested in the expected running time of a random equivalence program, when a particular set merging scheme is used. This set merging scheme uses a forest data structure, and employs a path compression rule [1], [7]; we will refer to this scheme as *quick merge with path compression* (or, QMP). The expected performance of equivalence algorithms using other set merging schemes has been extensively studied in Knuth and Schönage [5], Yao [9]. It seems reasonable to regard the expected performance on equivalence programs as a benchmark for the average-case behavior of a set merging scheme (Doyle and Rivest [2] discussed the expected cost of a set merging scheme by considering a sequence of random FINDs and UNIONs directly, however).

In the QMP set merging scheme, the family of subsets (equivalence classes) are represented by a forest of disjoint rooted trees. Each tree corresponds to a subset, with the name of that subset stored at the root. Command FIND $(x)$ accesses the node $v$ representing $x$ and triggers a traversal up the tree to its root $r$. In addition to returning the name of the subset, FIND $(x)$ also performs a *path compression* from $v$ to $r$, i.e., connecting every node on the path directly to $r$. Command UNION$(A, B)$ is implemented by attaching the root for subset $A$ to that for subset $B$. For definiteness, we charge 1 time unit for UNION and $l$ time units for FIND, where $l$ is the number of

* Received by the editors July 31, 1981, and in final revised form November, 1983. This research was done while the author was visiting the Computer Science Department, IBM San Jose Research Center, 5600 Cottle Road, San Jose, California. This work was supported in part by the National Science Foundation under grant MCS-77-05313-A01.

† Computer Science Department, Stanford University, Stanford, California 94305.

nodes on the traversed path. It is known that, with QMP, the worst-case time for performing a sequence of $O(n)$ UNIONs and FINDs is $\Theta(n \log n)$, where the lower bound proof is due to Fischer [3] and the upper bound is due to Paterson [6].

We now describe the randomness assumption we will use on the equivalence program. Let $\mathcal{T}_n$ be the set of all equivalence programs $\sigma = (x[1] \equiv y[1],\ x[2] \equiv y[2], \cdots, x[n-1] \equiv y(n-1])$ such that the set of edges $\{x[i], y[i]\}$, $1 \leqq i < n$, forms a spanning tree for the set $S$. Clearly, $|\mathcal{T}_n| = n^{n-2}(n-1)!2^{n-1}$, where the last factor $2^{n-1}$ accounts for the fact that each edge $\{a, b\}$ can appear as either $a \equiv b$ or $b \equiv a$. Let us consider the *spanning tree model* [5] [9], in which each equivalence program $\sigma \in \mathcal{T}_n$ is equally likely. Let $C_n^{\mathrm{QMP}}$ be the expected running time of a random $\sigma$ when QMP is used.

Our main result is the following theorem.

THEOREM 1. $C_n^{\mathrm{QMP}} = O(n)$.

As an intermediate step, we will prove a result of some independent interest (Theorem 2 below) that applies to the expected running time under any randomness assumption belonging to a general class.

Consider a model $\tau$ for random equivalence programs, specified by a probability distribution $p_\tau(\sigma)$. We call $\tau$ a *canonical model* if $p_\tau(\sigma) = 0$ for all $\sigma \notin \mathcal{T}_n$. For each $\sigma$, let $C_\sigma^{\mathrm{QMP}}$ be the running time of $\sigma$ when QMP is used. The expected running time is then $C_\tau^{\mathrm{QMP}} = \sum_\sigma p_\tau(\sigma) C_\sigma^{\mathrm{QMP}}$. Note that the Spanning Tree Model is a canonical model $\tau_0^{(n)}$ such that $P_{\tau_0^{(n)}}(\sigma) = 1/|\mathcal{T}_n|$ for all $\sigma \in \mathcal{T}_n$.

For any $\sigma$, let $W_i(\sigma)$ be the new equivalence class obtained from the merge of the two equivalence classes containing $x[i]$ and $y[i]$, when the $i$th instruction $x[i] \equiv y[i]$ of $\sigma$ is performed. Let $\alpha_\sigma = \sum_i \log_2 |W_i(\sigma)|$. For a model $\tau$, define the *potential* of $\tau$ as $H_\tau = \sum_\sigma p_\tau(\sigma)\alpha_\sigma$.

THEOREM 2. *For any canonical model* $\tau$,

$$C_\tau^{\mathrm{QMP}} \leqq 2H_\tau + 5(n-1).$$

In § 2 we briefly review Paterson's proof [6] for the worst-case upper bound on the QMP running time. In § 3 we establish Theorem 2, which involves a refinement of Paterson's analysis. We then prove Theorem 1 in § 4 by using Theorem 2. Some remarks and open problems are given in § 5.

**2. Paterson's entropy.** In this section we review Paterson's proof [6] for the QMP worst-case upper bound.

Let $T$ be a rooted forest with only internal nodes. For each $v \in T$, let $w_T(v)$ be the number of descendants of $v$ (including itself). The *entropy* of $T$ is defined to be $H_0(T) = \sum_{v \in T} \log_2 (w_T(v))$. Clearly $H_0(T) \leqq n \log_2 n$, if $T$ has $n$ nodes.

LEMMA 1 (Paterson [6]). *Suppose a path compression of length* $t+2$ *is performed along a path* $v_0, v_1, \cdots, v_t, v_{t+1}$ *in* $T$. *Let* $T'$ *be the new resultant tree. Then there exists a* $1 < \beta \leqq (w_T(v_t))^{1/t}$ *such that*

$$H(T) - H(T') \geqq t \log_2 \frac{\beta}{\beta - 1}.$$

*Proof* (sketch). The expression

$$H(T) - H(T') = \sum_{1 \leqq i \leqq t} (\log_2 w_T(v_i) - \log_2 (w_T(v_i) - w_T(v_{i-1})))$$

(under the constraint $1 \leqq w_T(v_0) < w_T(v_1) < \cdots < w_T(v_t)$) is minimized when $\{w_T(v_i)\}$ form a geometric progression $\{\alpha\beta^i\}$ with $\alpha \geqq 1$, $\alpha\beta^t = w_T(v_t)$. The lemma follows by an explicit evaluation. □

For any sequence $\sigma$ of $O(n)$ UNIONs and FINDs, one can equivalently first carry out all the UNIONs, followed by the FINDs (which are now "partial" FINDs as the path compressions may end at nodes other than the roots of the forest) [1]. Let $T_\sigma$ be the forest obtained after all the UNIONs are performed, but before any FIND is. The subsequent (partial) FINDs will modify the forest and decrease its entropy. Each FIND with cost $t+2>\log_2 n+2$ will decrease the entropy by at least $t$, according to Lemma 1. Thus, the total cost for FINDs is bounded by $H_0(T_\sigma)+O(n)$, plus the costs due to the FINDs with individual cost $\leqq \log_2 n+2$. It follows that the total cost for the FINDs is $O(n \log n)$; the UNIONs, of course, only cost $O(n)$. This finishes Paterson's proof of an $O(n \log n)$ upper bound for QMP.

**3. Proof of Theorem 2.** An equivalence program $\sigma$ induces a sequence $\sigma'$ of UNIONs and FINDs that the QMP algorithm actually executes. We will use $T_\sigma$ to stand for $T_{\sigma'}$.

LEMMA 2. *For any* $\sigma \in \mathcal{T}_n$,

$$C_\sigma^{\mathrm{QMP}} \leqq H_0(T_\sigma)+\alpha_\sigma+5(n-1).$$

*Proof.* Let $\sigma=(x[1]\equiv y[1],\ x[2]\equiv y[2], \cdots, x[n-1]\equiv y[n-1])$, and $S_{2i-1}\subseteq S$, $S_{2i}\subseteq S$ be the components containing $x[i]$, $y[i]$ just before the $i$th equivalence instruction is executed. Consider the sequence of path compressions $\xi_1, \xi_2, \cdots, \xi_{2n-2}$ that QMP carries out on $T_\sigma$, where $\xi_{2i-1}$ is induced by FIND $(x[i])$ and $\xi_{2i}$ by FIND $(y[i])$. Let $l_i$ be the length of $\xi_i$; define $J_1=\{i \mid l_i>\log_2|S_i|+2\}$ and $J_2=\{i \mid l_i \leqq \log_2|S_i|+2\}$. Let $A_1=\sum_{i\in J_1} t_i$ and $A_2=\sum_{i\in J_2} t_i$, where $t_i=l_i-2$. As each UNION only costs one unit time, we have

$$C_\sigma^{\mathrm{QMP}}=n-1+\sum_i l_i \leqq 5(n-1)+A_1+A_2.$$

We first prove $A_1 \leqq H_0(T_\sigma)$. Each $\xi_i$ will successively modify the forest $T_\sigma$ and decrease the entropy of the forest by at least $t_i \log_2(\beta/(\beta-1))$ according to Lemma 1. It is easy to see that the quantity $w_T(v_{t_i})$ in Lemma 1 is at most $|S_i|$. It follows that $1<\beta \leqq (w_T(v_{t_i}))^{1/t_i} \leqq 2$ (since $t_i>\log_2|S_i|$). The entropy decrease is thus at least $t_i$. This implies $H_0(T_\sigma) \geqq \sum_{i\in J_1} t_i = A_1$.

To finish the proof of Lemma 2, we need only prove $A_2 \leqq \alpha_\sigma$. By definition,

$$A_2 \leqq \sum_{i\in J_2} \log_2|S_i| \leqq \sum_{1\leqq i\leqq 2n-2} \log_2|S_i|.$$

Observe that, except for the $n-1$ $S_i$ with $|S_i|=1$, the $S_i$ are in one-to-one correspondence with the $W_j(\sigma)$ in the expression $\alpha_\sigma=\sum_{1\leqq j\leqq n-1}\log_2|W_j(\sigma)|$. This proves $A_2 \leqq \alpha_\sigma$. □

LEMMA 3. *For any* $\sigma \in \mathcal{T}_n$,

$$H_0(T_\sigma) \leqq \alpha_\sigma.$$

*Proof.* Let $v \in T_\sigma$ and let $D_v$ be the subtree rooted at $v$. In the formation of $T_\sigma$, there is a unique UNION instruction that makes $D_v$ a component of the forest; let $x[i_v]\equiv y[i_v]$ denote the equivalence instruction that induces this UNION. Clearly, $|W_{i_v}(\sigma)|=w_{T_\sigma}(v)$. It is also evident that distinct $v$ give distinct $i_v$. Hence

$$H_0(T_\sigma) \leqq \sum_{1\leqq j<n} \log_2|W_j(\sigma)|=\alpha_\sigma. \qquad \square$$

It follows from Lemmas 2 and 3 that $C_\sigma^{\mathrm{QMP}} \leqq 5(n-1)+2\alpha_\sigma$. Taking the average with weight $p_\tau(\sigma)$, we obtain Theorem 2.

**4. Proof of Theorem 1.** Because of Theorem 2, it suffices to prove $H_\tau = O(n)$ for $\tau = \tau_0^{(n)}$. Take a random $\sigma$ in the Spanning Tree Model. Let $p_{nk}$ be the probability that $|V_x| = k$ and $|V_y| = n-k$, where $V_x$ and $V_y$ are the two components containing $x[n-1]$ and $y[n-1]$ just before the last instruction $x[n-1] \equiv y[n-1]$ in $\sigma$. It is easy to verify the following facts:

(A) $$p_{nk} = \frac{1}{2(n-1)}\binom{n}{k}\left(\frac{k}{n}\right)^{k-1}\left(\frac{n-k}{n}\right)^{n-k-1}.$$

(B) Let $\sigma_x$ be the subsequence of $\sigma$ acting on $V_x$, and $\sigma_y$ be the subsequence of $\sigma$ acting on $V_y$; then $\sigma_x$ is a random equivalence program in the spanning tree model $\tau_0^{(k-1)}$, and similarly $\sigma_y$ is a random equivalence program in the model $\tau_0^{(n-k)}$.

Fact (B) is immediate from the definition of a random equivalence program. Fact (A) follows from a simplification of the equation

$$p_{nk} = \frac{\binom{n}{k}k(n-k)\binom{n-2}{k-1}k^{k-2}(k-1)!\,2^{k-1}(n-k)^{n-k-2}(n-k-1)!\,2^{n-k-1}}{n^{n-2}(n-1)!\,2^{n-1}}.$$

In the above expression, the factor $\binom{n}{k}$ comes from enumerating the choice of $V_x$ and $V_y$, $k(n-k)$ comes from enumerating the choice of $x[n-1]$ and $y[n-1]$ within $V_x$ and $V_y$; $\binom{n-2}{k-1}$ is the number of ways to interleave $\sigma_x$ and $\sigma_y$, and the remaining numerators give the number of possible $\sigma_x$ and $\sigma_y$. (Facts (A) and (B) were also shown in [5, § 9], although the spanning tree model there was phrased in a slightly different language.)

Let $r_n = H_\tau$ where $\tau = \tau_0^{(n)}$. It follows from Fact (B) that

$$r_n = \log_2 n + \sum_{0<k<n} p_{nk}(r_k + r_{n-k}) \quad \text{for } n > 1,$$

and

$$r_1 = 0.$$

A recurrence relation of this form with $p_{nk}$ as in (A) was studied in Knuth and Schönhage [5, eq. (12.8)], where it was shown that the solution satisfies $r_n = O(n)$.

We have proved Theorem 1.

**5. Remarks.** For any equivalence program $\sigma = (x[1] \equiv y[1],\ x[2] \equiv y[2], \cdots, x[n-1] \equiv y[n-1]) \in \mathcal{T}_n$, consider the *union tree* $Y_\sigma$ defined in Knuth and Schönhage [5, § 13] as follows: For $1 \leqq i < n$, construct a new node whose left subtree is the union tree for the current component containing $x[i]$ and whose right subtree is the union tree for the current component containing $y[i]$ ("current" means just before performing the instruction $x[i] \equiv y[i]$); the union tree for a single element is a leaf. (Note that $Y_\sigma$ is different from the tree $T_\sigma$ considered before, as can be seen from the fact that $Y_\sigma$ is always a binary tree.) We can regard $\alpha_\sigma$ as the "potential" of the tree $Y_\sigma$, defined by $\sum_{v \in Y_\sigma} \log_2 w(v)$, where $w(v)$ is the number of leaf-descendants of $v$. Theorem 2 can then be described as "the expected running time of QMP in a canonical model $\tau$ is bounded by the average potential of a random union tree in $\tau$".

In the Spanning Tree Model, we have shown that the equivalence algorithm using path compression has an expected $O(n)$ running time for carrying out $n-1$ equivalence instructions. It is easy to show by a similar argument that the expected running time of the first $l$ instructions is $O(l)$. However, it is not known if the expected running time for performing the $l$th instruction is $O(1)$ for every $1 \leqq l < n$. An interesting related open problem is the determination of the average rank of elements in the final forest data structure. In passing, we remark that there are algorithms that run in

worst-case $O(n)$ time on the spanning tree model, and in fact on any canonical model (Tarjan [8]).

Although our motivation for studying this model is mainly theoretical, the result may be relevant in some situations involving sparse graphs. Consider the processing of equivalence instructions in Kruskal's minimum spanning tree algorithm for random weighted input graph $G_{en}$, such that each connected graph with $e$ edges on $n$ vertices is equally likely to occur, and each of the $e!$ different permutations of edge weights is equally likely to happen. When $e = n-1$, the distribution of the sequence of equivalence instructions is the same as in the spanning tree model considered in this paper. It is even plausible that as long as $e = O(n)$, the result obtained in the spanning tree model may give a better estimate of the cost than in other models, say, the random graph model [5], [9], as connectivity is a severe constraint (a random graph in that model does not become connected until $e = O(n \log n)$). It is an interesting open problem to confirm this conjecture, and more generally, to analyze the compression algorithm in this "random $G_{e,n}$" model with general $e$, $n$.

Two other randomness models for equivalence programs have been discussed in the literature. In the *Doyle–Rivest model* [2][1] any pair of equivalence classes is equally likely to be joined. It is not hard to show that $H_\tau = O(n)$ in this case; from the discussions in [5, § 13], one can obtain the recurrence $r_n = \log_2 n + 1/(n-1) \times \sum_k (r_k + r_{n-k})$, where $r_n$ stands for $H_\tau$ with $n$ elements. This implies an $O(n)$ expected running time for QMP. In the random graph model [5], [9], the expected time for QMP is an unresolved question. The present approach yields only a trivial $O(n \log n)$ bound, since a component of size $\Omega(n)$ is involved with probability $\Omega(1)$ in the $l$th equivalence instruction for $l > (\frac{1}{2} + \varepsilon)n$, which gives $H_\tau = \Omega(n \log n)$. Bob Sedgewick (private communication, 1979) has done an extensive simulation up to 100,000 nodes. The running time appears definitely nonlinear in $n$, and is consistent with an $n \log n$ growth. A theoretical resolution of this case is a major remaining open problem in the analysis of set merging algorithms.

## REFERENCES


[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[2] J. DOYLE AND R. L. RIVEST, *Linear expected time of a simple union–find algorithm*, Inform. Processing Lett., 5 (1976), pp. 146–148.

[3] M. J. FISCHER, *Efficiency of equivalence algorithms*, in Complexity of Computer Computations, R. E. Miller and J. W. Thatcher, eds., Plenum Press, New York, 1972.

[4] B. A. GALLER AND M. J. FISCHER, *An improved equivalence algorithm*, Comm. ACM, 7 (1964), pp. 301–303.

[5] D. E. KNUTH AND A. SCHÖNHAGE, *The expected linearity of a simple equivalence algorithm*, Theoret. Comput. Sci., 5 (1978), pp. 281–315.

[6] M. S. PATERSON, 1972, unpublished; a description of Paterson's proof was given in MIT class notes (Course 6.851J) by A. R. Meyer and M. J. Fischer, 1973.

[7] R. E. TARJAN, *Complexity of combinatorial algorithms*, SIAM Rev., 20 (1978), pp. 457–491.

[8] ———, *Worst-case analysis of set union algorithms*, to appear.

[9] A. C. YAO, *On the average behavior of set merging algorithms*, (extended abstract), Proc. 8th Annual ACM Symposium on Theory of Computing, 1976, pp. 192–195.


[1] The original model in [2] is defined in terms of FINDs and UNIONs; we modify it here by replacing each UNION with an equivalence instruction and omitting the FINDs.