Comments on "Rem's algorithm for the recording of equivalence classes"
Maarten M. Fokkinga, 1979 jan. 31.

Abstract. It is assumed that the reader is familiar with chapter 23 of A Discipline of Programming (Dijkstra 1976). We give a different motivation for the property $\exists x. x \geq f(x)$, and obtain Rem's algorithm quite naturally, meanwhile lifting out its "compelling beauty".

Having seen Dijkstra's algorithms, pp. 161-164, we may continue the story as follows. It is not nice that both for testing whether $p$ and $q$ are equivalent and for processing the edge $\{p,q\}$, all the vertices up to the identifying ones have to be traversed. In both cases we need not go beyond the first common element, if any, of the sequences $(p, f_p, f(f(p)), \ldots)$ and $(q, f_q, f(f(q)), \ldots)$.

Let us denote that element, if existent, by $fce(p, q)$.

So let us develop a better algorithm for the equivalence of $p$ and $q$. We introduce two variables $p_0$ and $q_0$ and choose the following invariant relation and variant function. The second term of $P$ will express that $fce(p, q)$, if existent, has not been passed.

$P : eqv(p, q) = eqv(p_0, q_0)$ and (not $eqv(p_0, q_0)$) for $fce(p_0, q_0) = fce(p, q)$,

$T : (\min k. f^k(p) = f^{k+1}(p)) + (\min k. f^k(q) = f^{k+1}(q))$.

Obviously $p := f(p_0)$ is a candidate command: $p_0 \neq f(p_0)$ guarantees effective decrease of $T$. In order that $fce(p_0, q_0) = fce(p, q)$ is kept invariant in case that $eqv(p, q)$ and $p_0 \neq f(p_0)$ hold true, we need to check

\[ p_0 \forall\{ q_0, f(q_0), f(f(q_0)), \ldots \} \]

It is at this point that we propose to exploit the ordering between the vertex numbers: if $x \geq f(x)$ holds for all $x$, then the condition is implied by either $p_0 > q_0$ (whence $p_0 > q_0 \geq f(q_0) \geq \ldots$),

or $f(p_0) > f(q_0)$ (whence $p_0 \neq q_0$ and $p_0 > f(p) > f(q_0) \geq \ldots$)

or $f^2(p_0) > f^2(q_0)$ (whence $p_0 \neq q_0$ and $p_0 \neq f(q_0)$ and $p_0 > f^2(p_0) > f^2(q_0) \geq \ldots$

and $f^i(p_0) > f^i(q_0)$ in general, for any $i \geq 0$ \(^*)\). (Proof: for $0 \leq j \leq i$

\(^*)\) Note that neither of these inequalities implies another one.
we have \( f^{i-j}(p0) \geq f^j f^{i-j}(p0) = f^i(p0) > f^i(q0) = f^{i-j} f^3(q0) \) so 
\( f^{i-j}(p) > f^{i-j}(f^3(q0)) \) hence \( p0 \# f^3(q0) \). Further, for \( j \geq i \) we have 
\( p0 \geq f^i(p0) > f^i(q0) \geq f^3(q0) \) so \( p0 \# f^3(q0) \). End of proof."

Without further problems, we arrive at the following algorithm;

i may be any value \( \geq 0 \).

\[
\begin{align*}
  & p0 \textbf{vir int, } q0 \textbf{vir int:=p,q; } p1 \textbf{vir int, } q1 \textbf{vir int:=f(p0),f(q0);} \\
  & \textbf{do } p0 \neq p1 \text{ and } f^i(p0) > f^i(q0) \rightarrow p0, p1:= p1, f(p1) \\
  & \quad \textbf{if } q0 \neq q1 \text{ and } f^i(q0) > f^i(p0) \rightarrow q0, q1:= q1, f(q1) \\
  & \quad \textbf{od;} \\
  & \quad \{ p0=f^i(p0) \textbf{ or } f^i(p0)=f^i(q0) \textbf{ or } f^i(p0)<f^i(q0)=q0 \} \\
  & \quad \textbf{if } f^i(p0)=f^i(q0) \rightarrow \textbf{eqv:=true} \\
  & \quad \quad \textbf{if } f^i(p0) \neq f^i(q0) \rightarrow \textbf{eqv:=false} \\
  & \quad \textbf{fi} \\
\end{align*}
\]

Note that, even when \( p \) and \( q \) are not equivalent, not all the vertices up to the identifying ones have been traversed!!

Clearly, the edge \{p,q\} can be processed by the above algorithm, if the alternative construct has been replaced by e.g.

\[
\begin{align*}
  & \textbf{if } f^i(q0) \neq f^i(q0) \rightarrow f:(p0)=f^i(p0) \\
  & \quad f^i(q0) \neq f^i(p0) \rightarrow f:(p0)=f^i(q0) \\
  & \quad \textbf{fi}, \\
\end{align*}
\]

possibly followed by a second scan compressing the pathes. Suppose however that we are not allowed to do a second scan. So there is no other choice than to compress the traversed path, if at all, within the repetition as follows

\[
\begin{align*}
  & \textbf{do } p0 \neq p1 \text{ and } f^i(p0) > f^i(q0) \rightarrow f:(p0)="new f(p0)"; p0, p1:= p1, f(p1) \\
  & \quad q0 \neq q1 \text{ and } f^i(q0) > f^i(p0) \rightarrow f:(q0)="new f(q0)"; q0, q1:= q1, f(q1) \\
  & \quad \textbf{od} \\
  & \textbf{..} \\
\end{align*}
\]

In thinking about choices for "new f(p0)" and "new f(q0)", it seems quite natural, on account of \( \forall x. x \geq f(x) \), to take \( x \) or \( f(x) \) of \( f(f(x)) \) or, in general, \( f^j(x) \) as a heuristic measure for the average length of the pathes

\( (x, f(x), f[f(x)], ...) \)

which will be traversed in applications of the equivalence testing or edge processing algorithm. Let us take \( i = j \). Then, the refinements

"new f(p0)" \( : f^i(q0) \)

"new f(q0)" \( : f^i(p0) \)

effectuate the largest possible decrease of that measure, while leaving
part(f)$p_0,q_0$ constant and \( \forall x \geq f(x) \) invariant.

Rather surprisingly we may now delete the terms \( p_0 \neq p_1 \) and \( q_0 \neq q_1 \) from the guards. Indeed, they were fully caused by the requirement of effective decrease of \( T \).

But if we now choose, instead of \( T \),
\[ T': f(p_0)+f(q_0) \]
then, thanks to the updating of \( f \) at \( p_0 \) and \( q_0 \), effective decrease of \( T' \) is guaranteed even if \( p_0 = p_1 \) and \( q_0 = q_1 \)!!

Due to this deletion, the repetition terminates with \( f^i(p_0) = f^i(q_0) \), so no further statement is needed. Choosing \( i = 1 \) yields Rem's algorithm.

**Conclusion.** We have given a quite different, but in our opinion more fundamental, motivation for the property \( \forall x \geq f(x) \) than Dijkstra has done. With this property the algorithms for testing the equivalence of \( p \) and \( q \) and for processing the edge \( \{p,q\} \), become fully symmetrical in \( p \) and \( q \). With regard to the number of evaluations of \( f \), they seem at least as efficient as the algorithms without that property: avoidable traversals and corresponding compressions of path-parts are merely postponed as long as possible.

Given that property of \( f \), and given the constraint of a single scan algorithm, Rem's algorithm comes then quite naturally. The only trick is the deletion of the conditions \( p_0 \neq p_1 \) and \( q_0 \neq q_1 \). As a surprising and beautiful consequence that simple repetition alone updates \( f \) completely, even for the last values of \( p_0 \) and \( q_0 \).