1976

Real Time Generation of MIN Distance Strings

Peter J. Denning

Report Number:
77-211

Real Time Generation of MIN Distance Strings

Peter J. Denning
Computer Science Department
Purdue University

CSD TR - 211

November, 1976

Real Time Generation of MIN Distance Strings*

Peter J. Denning
Purdue University**

November, 1976

Abstract: A simple proof is presented for an old result: the stack distance string for the optimal paging algorithm, MIN, can be generated in real time at the same overhead as any general stack algorithm.

Key Words and Phrases: Paging algorithm, optimal paging algorithm, MIN algorithm, MIN policy, sorting networks.

CR Categories: 8.1, 4.32

---

*Work reported herein supported in part by NSF Grant GJ-41289.

**Computer Sciences Department, West Lafayette, Indiana 47907.

Real Time Generation of MIN Distance Strings

Peter J. Denning

Introduction

Though optimal memory policies be unrealizable, their paging behaviors are of interest as the best possible performance obtainable from a given program in a given memory space. When the resident set size is held fixed, the optimal demand policy, MIN, replaces pages with the longest forward reference intervals [Bel66, MGS70]. When the resident set size is allowed to vary, the optimal demand policy, VMIN, replaces pages whose forward reference intervals exceed a given threshold [PrF76, DeS76]. Having fewer constraints than MIN, VMIN produces lower paging rates at each given value of mean resident set size. VMIN is well behaved; its proof of optimality is simple, and its paging curve can be obtained, on one pass of a program's reference string, as a subcomputation of the working set paging curves [Den75, DeS76]. In contrast, MIN's behavior is difficult to understand; its proof of optimality is tedious, and efficient procedures for computing its paging rates difficult to find.

A straightforward procedure for computing the MIN paging rates was devised by Mattson et al [MGS70]. On a forward pass over a reference string, it tags each reference with the time since prior reference; on a

reverse pass, it uses these tags as lookahead intervals. This procedure has two practical limitations. First, address trace tapes are often long; processing them twice doubles the overhead in comparison with one pass methods. Second, it cannot be employed for on-line measurement. These limitations motivate interest in one-pass MIN analyzers.

Belady's original MIN analyzer is one pass [Bel66]. It is based on deferring a given page replacement decision until enough subsequent references have been observed. It has two limitations. First, it counts page faults for one given resident set size only. Second, it does not operate in real time: its output at time t is a delayed page fault count -- that is valid at some time $t' < t$. (In fact, t' is the latest time for which all pages resident at t' are referenced again between t' and t.) To remove these limitations, Belady and Palermo devised the "multilevel MIN" analyzer, which produces the MIN paging rates on a given reference string, for all memory sizes of interest, in a single pass [BeP74].

Because it is capable of real time operation -- that is, it produces the page fault count for time t at time t -- the mulitlevel MIN procedure is the basis of a patent, issued to Belady, for on-line measurement of MIN paging rates [Bel]. In their own studies of the Belady-Palermo procedure, Lewis and Nelson discovered a simple proof that real-time analysis of the MIN algorithm is possible and a procedure for doing it [LeN74]. The purpose of this paper is giving a greatly shortened, simplified proof of these on-line procedures.

Since knowledge of stack distances suffices to compute page fault counts for stack algorithms, such as MIN [CoD73, MGS70], we will confine our attention to real time generation of MIN (stack) distance strings. After a review of stack algorithm properties, we present the intuitive basis of a one-pass MIN analyzer. By means of a "tableau procedure" we demonstrate that real time generation of MIN stack distances is possible. Analogies between the tableau procedure and sorting networks are exploited to develop a simple proof of the real time MIN analyzer.

Stack Algorithms [MGS70, CoD73]

Let $S(t) = (x_1,\ldots,x_n)$ denote a MIN stack at time t, just after the reference to page r(t), for t = 1,2,... . The resident set of size m page frames is the topmost m elements of S(t), viz. $\{x_1,\ldots,x_m\}$. The MIN distance d(t) is the position of r(t) in the stack S(t-1); a page fault occurs at time t if and only if d(t) > m. The tails of the frequency distribution of MIN distances define the page fault rate function, known as the paging curve.

Suppose D(x,t) denotes the distance of page x in the MIN stack S(t). Then d(t) = D(r(t), t-1). It is well known that

(1)
1. $D(x, t) = 1$, if $r(t) = x$;
2. $D(x, t) \geq D(x, t-1)$, if $r(t) \neq x$; and
3. $D(x, t) = D(x, t-1)$, if $d(t) < D(x, t-1)$.

The set of distances D(x,t) for some interval of t is called the trajectory of x in that interval. On a first reference to a page d(t) is infinite, a fact denoted by writing d(t) = #; the number of known pages (n) increases by 1 when d(t) = #.

MIN's replacement decisions depend on a priority list $P(t)=(p_1,\ldots,p_n)$ which orders the pages according to increasing forward distance after t -- $i < j$ implies p_i is referenced before p_j. If a page fault occurs at time t for a memory of size m, the replacement page is

$$z_m = \min(x_1,\ldots,x_m) = \min(z_{m-1}, x_m) , \quad m>1, \tag{2}$$

where $S(t-1) = (x_1,\ldots,x_m,\ldots,x_n)$ and "min" denotes smaller priority according to P(t), that is, larger forward distance. Using (1) and (2), it is not difficult to define the new stack $S(t) = (y_1,\ldots,y_n)$ in terms of the former stack $S(t-1) = (x_1,\ldots,x_n)$ and priority list P(t), when d(t)=m:

$$\begin{aligned} y_1 &= x_m, \\ y_j &= \max(z_{j-1}, x_j), \quad 1 < j < m \\ y_m &= z_{m-1} \\ y_j &= x_j, \quad m < j \leq n \end{aligned} \tag{3}$$

In case d(t)=#, S(t) contains one more page than S(t-1); the correct S(t) will result if r(t) is added to the $n+1^{st}$ position of S(t-1) and (3) is applied with m=n+1. Relations (3) can be envisaged as a sorting network as sketched in Figure 1. Let G denote the updating function (3); thus the sorting network of Figure 1 performs the transformation

$$S(t-1) \xrightarrow{G} S(t) . \tag{4}$$

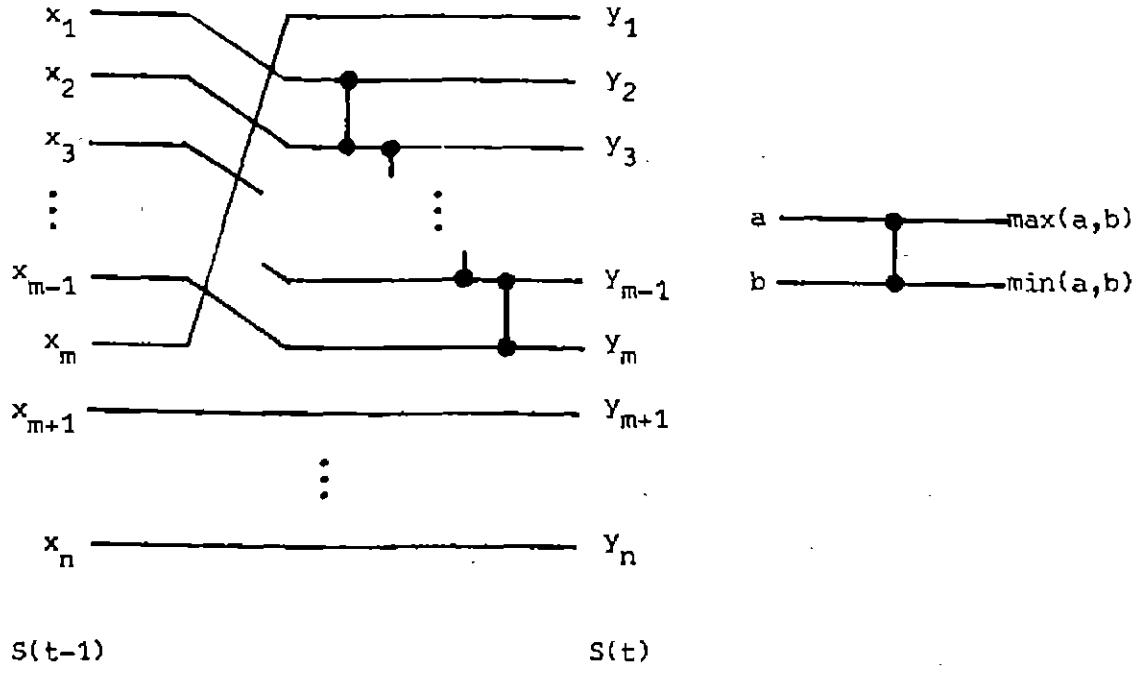

Figure 1. Sorting network for MIN stack updating procedure.

Basis of one pass MIN Anaylzer

A direct implementation of MIN is not possible since the priority list needed for updating the stack is not known in advance. However, an indirect implementation is possible. Suppose the MIN stack S(t-1) has been constructed under the hypothesized future

$$P(t-1) = (p_1,\ldots,p_{i-1},p_i,p_{i+1},\ldots,p_n) \quad (5)$$

If one then discovers that $r(t)=p_i$ for $i>1$, one must construct a corrected stack $S'(t-1)$ consistent with a corrected priority list

$$P'(t-1) = (p_i,p_1,\ldots,p_{i-1},p_{i+1},\ldots,p_n) \quad (6)$$

Since the corrected priority list is a minimal purturbation of the hypothesized one, p_i changes priority relative only to $p_1,\ldots,p_{i-1}$ while maintaining the same priority relative to $p_{i+1},\ldots,p_n$. From Figure 1, this implies that the correction of S(t-1) should involve only the permutation of $p_1,\ldots,p_i$ among the same set of positions. Denote the correction (if it exists) by a transformation F, that is

$$S(t-1) \xrightarrow{F} S'(t-1) \quad . \tag{7}$$

In the discussion of the tableau procedure, we will show that the MIN stack position of r(t) is uniquely determined by the reference string prefix r(1)...r(t); thus the position of p_i in the corrected S'(t-1) is its correct position and can be outputted as the MIN distance d(t). When the hypothesis P(t) is the <u>reversed LRU stack</u>, in which i<j implies p_i was referenced less recently than p_j, the transformation F exists and is straightforward.

What must be proved is the following. Suppose d(1)...d(t-1) have been correctly generated. The stack S(t-1) is correct for the hypothesis that $P(t-1) = (p_1,\ldots,p_n)$ is the reversed LRU stack. When it is discovered that $r(t) = p_i$, the next stack is computed from

$$S(t-1) \xrightarrow{F} S'(t-1) \xrightarrow{G} S(t) \tag{8}$$

where F rearranges $p_1,\ldots,p_i$ among their original positions and G updates according to the reversed LRU stack

$$P(t) = (p_1,\ldots,p_{i-1},p_{i+1},\ldots,p_n,p_i) \quad . \tag{9}$$

The correct MIN stack distance d(t) is the position of p_i in S'(t-1).

It is worth noting immediately that, if the above formulation is correct, the updating of S'(t-1) by G is trivial. That P(t-1) is the reversed LRU stack at time t-1 implies $r(t-1) = p_n$ is on top of both

S(t-1) and S'(t-1). Referring to the sorting network of Figure 1, one can see that moving p_i from position m to position 1 will cause p_n to move into position m, since p_n has lowest priority. In other words, G is executed simply by exchanging the referenced page with the top page of S'(t-1). If r(t) is a first reference, S(t-1)=S'(t-1); however, by placing r(t) at the $n+1^{st}$ position of S'(t-1), G is still implemented by exchanging the top and referenced pages.

A Tableau Procedure for Generating MIN Distances*

A pictorial representation of MIN behavior is useful: both for proving that real time MIN distance generation is possible, and for developing the intuition for a correcting function F. It is based on constructing a portion of the trajectory of page x, as soon as it is discovered that r(t)=x, in a two-dimensional matrix whose rows (counting down) correspond to stack positions (1,...,n) and columns (counting right) to time instants (t=1,2,...). If D(x,t)=i is finite, x will be entered in square (i,t); no entry will be shown otherwise. The tableau procedure consists in applying these steps for t =1,2,... : Let r(t)=x, then

1. Enter x in position (1,t).
2. If r(t) is not the first reference to x, locate the time t' of prior reference to x. For u = t'+1,...,t-1, enter x in position (j,u), where j is the topmost vacancy not higher than D(x, u-1).

The claim is that this procedure constructs the MIN trajectory D(x,u) for

*This discussion simplifies the presentation of Lewis and Nelson [LeN74], who observed that it reformulates the Belady-Palermo procedure [BeP74]. The example of Figure 2 also appears in [LeN74].

$t' < u \leq t$. Figure 2 illustrates a tableau constructed in stages by applying this procedure to a reference string for t = 5,6,...,12; MIN distances are indicated along the top line of each stage.

This procedure is easily proved correct with the help of relations (1)-(3). It is evidently correct when d(t) = #, and in particular for t=1. As an induction hypothesis, assume it is correct for all t<T and suppose x=r(T) is not a first reference. Suppose T' is the time of prior reference to x. Let t be given, where $T' < t < T$. Let k be the correct position of x at time t, and note from (1) that $k \geq D(x, t-1)$. Let j be the topmost vacancy in column t not higher than D(x, t-1). Any page occupying position i, where $D(x, t-1) \leq i < j$, must have been entered prior to time T and is, by hypothesis, correctly positioned; thus k<j is impossible. If it were that k>j, relations (3) require the page z that belongs in position j to be referenced earlier than x; however, this is impossible because the induction hypothesis holds that any such page z is already correctly positioned. Therefore, k=j is the only possibility for the MIN position D(x,t).

The important property demonstrated by this procedure is that the MIN trajectory for r(t) is uniquely determined by the trajectories of pages referenced prior to time t; it will not be changed by any trajectory entered subsequently. Thus it is possible to determine MIN distance d(t) as soon as r(t) is observed. The problem is to make this determination without storing the entire tableau.

t=5	# A	# B	# C	# D	# E	C	B	D	A	B	D	E
t=6	# A	# B	# C	# D C	# E C	2 C	B	D	A	B	D	E
t=7	# A	# B	# C B	# D B	# E B	2 C B	3 B	D	A	B	D	E
t=8	# A	# B	# C B	# D C B	# E C B D	2 C B D	3 B D	4 D	A	B	D	E
t=9	# A	# B A	# C B A	# D C B A	# E C B A	2 C B A	3 B D A	4 D A	5 A	B	D	E
t=10	# A	# B A	# C B A	# D C B A	# E C B A	2 C B A	3 B D A	4 D B A	5 A B D	2 B D	3 D	E
t=11	# A	# B A	# C B A	# D C B A	# E C B A	2 C B A	3 B D A	4 D B A	5 A B D	2 B D	3 D	E
t=12	# A	# B A	# C B A	# D C B A	# E C B A	2 C B A	3 B D A	4 D B A	5 A B D E	2 B D E	3 D E	4 E

Figure 2. Example showing successive stages of MIN tableau procedure. New trajectories indicated with underbars.

The Stack Correction Procedure

Figure 3(a) shows an idealization of a tableau constructed on the hypothesis that the future priority list P(t-1) is the reversed LRU stack. For each i, $p_1,\ldots,p_{i-1}$ will be called the LRU elders of p_i. The trajectories of $p_1,\ldots,p_i$ are shown, together with their resulting positions in S(t-1). These trajectories may be regarded as paths in a large sorting network (corresponding to compositions of the updating network of Figure 1); the points where trajectories cross correspond to comparisons using max/min according to P(t-1). Figure 3(b) illustrates the changes that occur when $r(t) = p_i$ is discovered. The tableau procedure will enter the new trajectory of p_i before those of its LRU elders; p_i will then follow the path of highest vacancy, which in this case was previously followed by p_2. The tableau procedure then enters new trajectories in order for $p_1,\ldots,p_{i-1}$; each will follow a path of highest vacancy (not higher than its former path) and will enter a stack position formerly occupied by some LRU elder of p_i. Figures 4(a) and 4(b) illustrate this; only A, B, and C change positions in S(16) when page C becomes the reference r(17). The following observations are important:

1. No two trajectories of $p_1,\ldots,p_i$ cross more than once en route to S(t-1), since their relative priorities do not change in this region.
2. Let $q_1,\ldots,q_m$ denote the subset of $p_1,\ldots,p_i$ appearing in S(t-1) from p_i upward; in particular, $q_m=p_i$. When p_i is promoted from lowest to highest priority among $p_1,\ldots,p_i$, only $q_1,\ldots,q_m$ are reordered, for only their trajectories cross that of p_i.

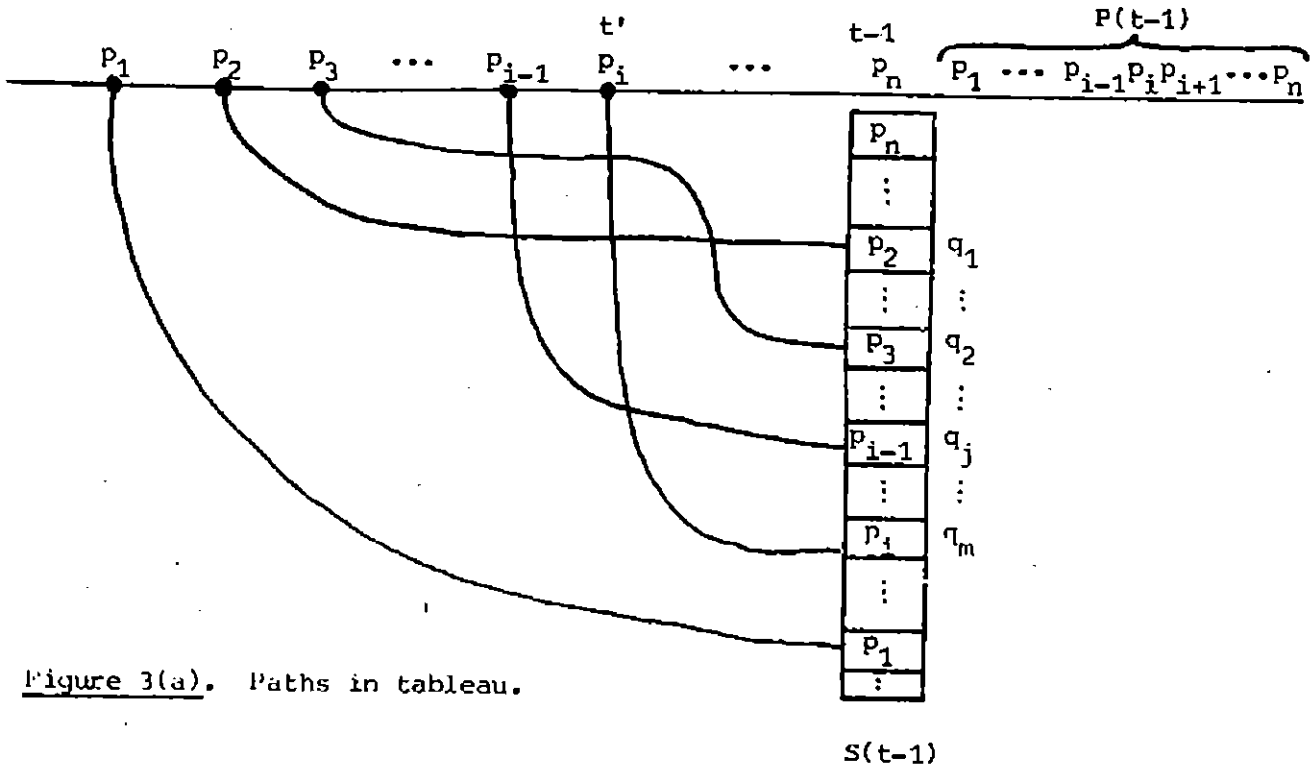

Figure 3(a). Paths in tableau.

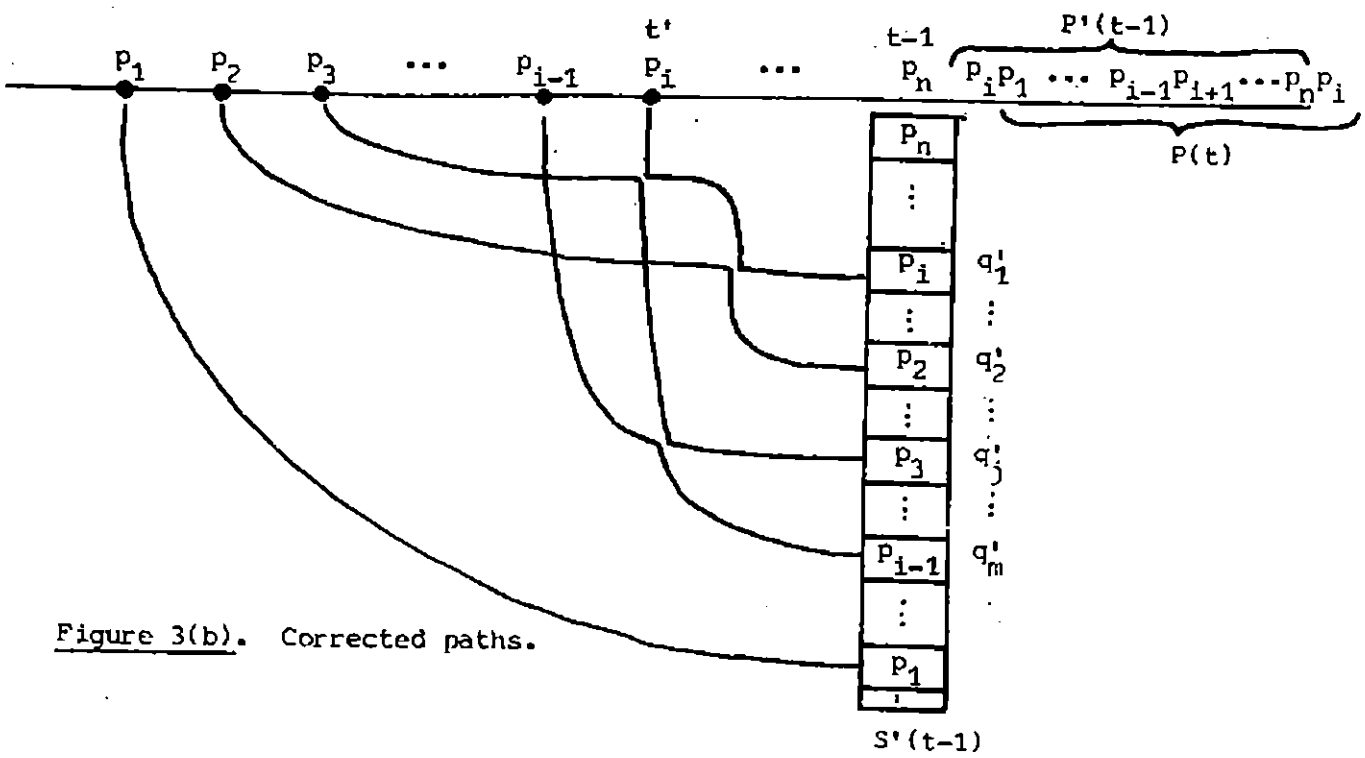

Figure 3(b). Corrected paths.

(a)

t:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
r(t):	A	V	W	X	W	V	B	V	X	C	V	Z	V	Y	Z	V	A	B	C
1	A	V	W	X	W	V	B	V	X	C	V	Z	V	Y	Z	V	A	B	C
2		A	V	W	X	X	V	B	V	V	C	V	Z	Z					
3			A	V	V		X	X	B	B	B	B	B	V	V				
4				A	A	A	A	A	A	A	A	A	A	A	A	A			
5												C	C	B	B	B	B		
6														C	C	C	C	C	

(b)

	A	V	W	X	W	V	B	V	X	C	V	Z	V	Y	Z	V	C	A	B
1	A	V	W	X	W	V	B	V	X	C	V	Z	V	Y	Z	V	C	A	B
2		A	V	W	X	X	V	B	V	V	C	V	Z	Z					
3			A	V	V		X	X	B	B	B	C	C	V	V				
4				A	A	A	A	A	A	A	A	A	A	C	C	C			
5												B	B	A	A	A	A		
6														B	B	B	B	B	

Figure 4. A MIN tableau before and after a change at t = 17.

3. The pages of $q_1,\ldots,q_m$ are reordered in the same set of stack positions, since $p_1,\ldots,p_i$ as a group continue to have the same priorities relative to the other pages between t' and t (where r(t') is the last reference to p_i).

4. In S'(t-1), p_i occupies the position held formerly by q_1 in S(t-1). Since the trajectory of $r(t)=p_i$ is uniquely determined at time t, the position of p_i in S'(t-1) is the correct MIN distance d(t).

A few moments reflection on the sorting networks of Figure 3 leads to the observation that, after q_m moves upward, $q_1,\ldots,q_{m-1}$ move downward, undergoing successive pairwise exchanges according to the max/min relations of P(t-1). In other words, the correction function F is nothing more than an application of the MIN sorting network to $q_1,\ldots,q_m$, using the reversed LRU stack as a priority list.

A proof of this can be constructed as follows. Let t' be the time of prior reference to $p_i = r(t)$. The correction, which changes only the priority of p_i relative to its LRU elders at time t, can alter only the portion of the tableau between t' and t. The stack updating procedure uses the same relative priorities for $p_1,\ldots,p_i$ everywhere in the interval (t',t); let G denote this procedure when the hypothesized priority order of these pages is $p_1p_2\ldots p_i$, and G' when the priority of these pages is $p_ip_1\ldots p_{i-1}$. Using G(u) and G'(u) to denote, respectively, instances of G and G' at time u (t'<u<t), our objective is showing that the proposed correction F completes this diagram:

(10)
$$\begin{array}{lll} S(t') & \xrightarrow{G(t'+1)\ldots G(t-1)} & S(t-1) \quad \text{[uncorrected]} \\ & & \downarrow F \\ & \xrightarrow{G'(t'+1)\ldots G'(t-1)} & S'(t-1) \quad \text{[corrected]} \end{array}$$

A few moments further reflection about F suggests that the proposed F would apply to p_i and its LRU elders everywhere in the interval between t' and t. Letting F(u) denote the instance of F used to correct the stack at time u, we have the diagram

(11)
$$\begin{array}{lll} S(u-1) & \xrightarrow{G(u)} & S(u) \quad \text{[uncorrected]} \\ \downarrow F(u-1) & & \downarrow F(u) \\ S'(u-1) & \xrightarrow{G'(u)} & S'(u) \quad \text{[corrected]} \end{array}$$

This diagram underlies an inductive proof of F. Assuming F(u-1) is valid, we can show F(u) is valid by proving

(12) $$G(u)F(u) = F(u-1)G'(u) .$$

The basis of this proof is trivial, since F(t') is an identity function because p_i is on top of S(t'), and we already know S(t') = S'(t'). A proof of the induction step, using sorting networks, is given in Appendix 1.

Conclusion

This paper has outlined a proof of the observation that the MIN stack distance d(t) can be computed in real time as soon as reference r(t) is observed. This fact rests on the observation that the MIN distance trajectory of page r(t) is determined uniquely by the trajectories of all prior references. A three stage procedure maintains a MIN stack on the hypothesis that the future reference order is the reversed LRU stack: 1) If the next reference x fails to confirm this hypothesis, the MIN stack is corrected by applying the MIN updating procedure to x and its LRU elders in the same set of stack positions. 2) The position of x in the corrected MIN stack is d(t). 3) The corrected MIN stack is updated for the corrected future hypothesis by exchanging the top and referenced pages. The procedure is repeated for the next reference. If x is a first reference, it is appended to the uncorrected stack, the MIN distance is set to #, and only Step 3 is performed. Details appear in Appendix 2.

One should not conclude that, because MIN distances are computable in real time, MIN itself is somehow realizable. Let MIN* denote a paging algorithm maintaining stacks and updating according to the F and G functions of this paper. MIN* would determine a resident set of k pages according to the hypothesized future; when the next page x is observed, the correction function F might move x upward from a position below k. This produces a page fault not produced by MIN.

Acknowledgements

I am grateful to Karl Winklmann and Kevin Kahn for criticizing an early draft of this paper; to R. A. Nelson for encouraging me to work on the problem; and to L. A. Belady for posing it.

Appendix 1 - Proof of Induction Step in Correction Procedure

We wish to prove eq. (12), the induction step in the proof that the correction function F is the MIN updating procedure applied to $p_i = r(t)$ and its LRU elders among the same set of stack positions. We do this with sorting networks. It is necessary only to specify networks that sort $p_1,\ldots,p_i$, since only the relative positions of these pages are affected by the correction. Figure 5(a) illustrates a network for F when i=10 and p_i is at the 7th position among $p_1,\ldots,p_i$ (Figure 3(a) illustrates that p_i need not be at the i^{th} position.) This network is interpreted as taking the set $p_1,\ldots,p_i$ in their order of appearance in some stack S(u), and specifying their order of appearance in the same set of positions in S'(u).

Figure 5(b) illustrates a network for G when i=10; it specifies that the set $p_1,\ldots,p_i$ in some stack S(u-1) are to be reordered and placed in S(u). It is important to realize that this G is not a full MIN updating network (Figure 1); it specifies only the relative positions of $p_1,\ldots,p_i$. (The actual positions of $p_1,\ldots,p_i$ would be determined from a full network.) The essential property of a G-network is that it contains k-1 comparators for some k, $1 \leq k \leq i$, connected across the first k inputs in the pattern shown in the figure; k is called the depth of G. That this is the only possible form of G is seen with the help of Figure 1 and the fact that $p_1,\ldots,p_i$ are adjacent in priority during every update between t' and t. Suppose $r_1,\ldots,r_i$ is the permutation of $p_1,\ldots,p_i$ appearing in a stack S(u-1). For each $j>1$, $x = \min(r_1,\ldots,r_{j-1})$ will be compared with r_j only if a) x has lower priority than all pages in S(u-1) between r_{j-1} and r_j, and b) the

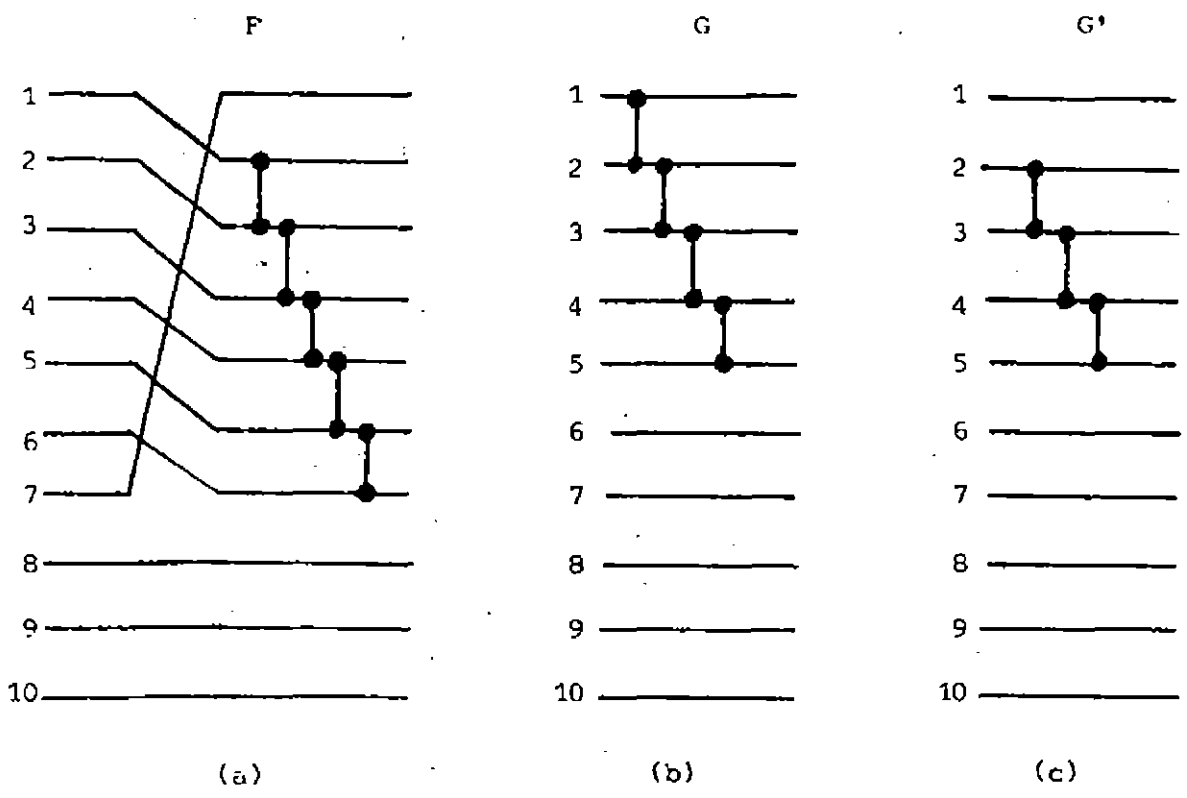

Figure 5. Network forms.

distance d(u) exceeds $D(r_j, u-1)$. Obviously, if (a) is true for all j, the depth of G is determined by d(u). Otherwise, x encounters some lower priority y in S(u-1) between r_{j-1} and r_j for some j; this y will simply pass any subsequent members of $p_1,\ldots,p_i$ with which it is compared, leaving them in the same order; in this case, the depth of G is j, being determined by D(y, u-1).

Figure 5(c) illustrates a network G' when i=10. It is identical to G except that the 1-2 comparator is deleted. The 1-2 comparator is not needed because page p_i is known (from the induction hypothesis) to occupy the highest position in S'(u-1) relative to its LRU elders, and p_i is already of highest priority among $p_1,\ldots,p_i$ in the corrected tableau.

The proof of G(u)F(u) = F(u-1)G'(u) is illustrated in Figure 6. The figures are drawn for i=10 in order to keep them simple; however, the pattern of the general case will be obvious. There are two cases to consider, depending on the relation of p_i's position in G, $D(p_i, G)$, to the depth of G.

Figure 6(a) illustrates the case that $D(p_i,G)$ does not exceed the depth of G. The important observation is that, since p_i has lowest priority among $p_1,\ldots,p_i$, it will descend to the depth of G, whereupon F(u) must bring it from this depth to the top. It is easy to see from the diagram that the outputs of the two networks are identical, since the second set of comparators (c1,...,c5) have the same inputs in each case.

Figure 6(b) illustrates the case that $D(p_i, G)$ exceeds the depth of G. In this case, F(u)=F(u-1). Since G'(u) has one less comparator that G(u),

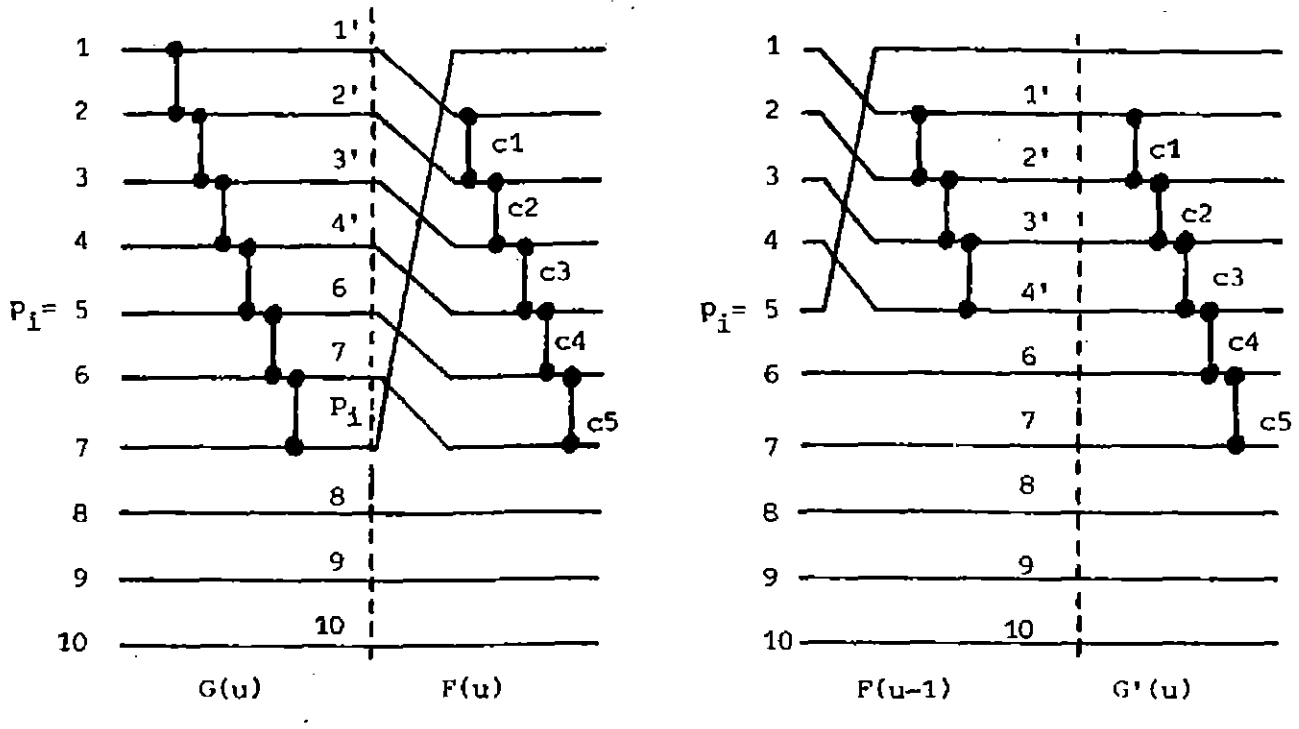

Figure 6(a). Depth of G not less than $D(p_i, G)$.

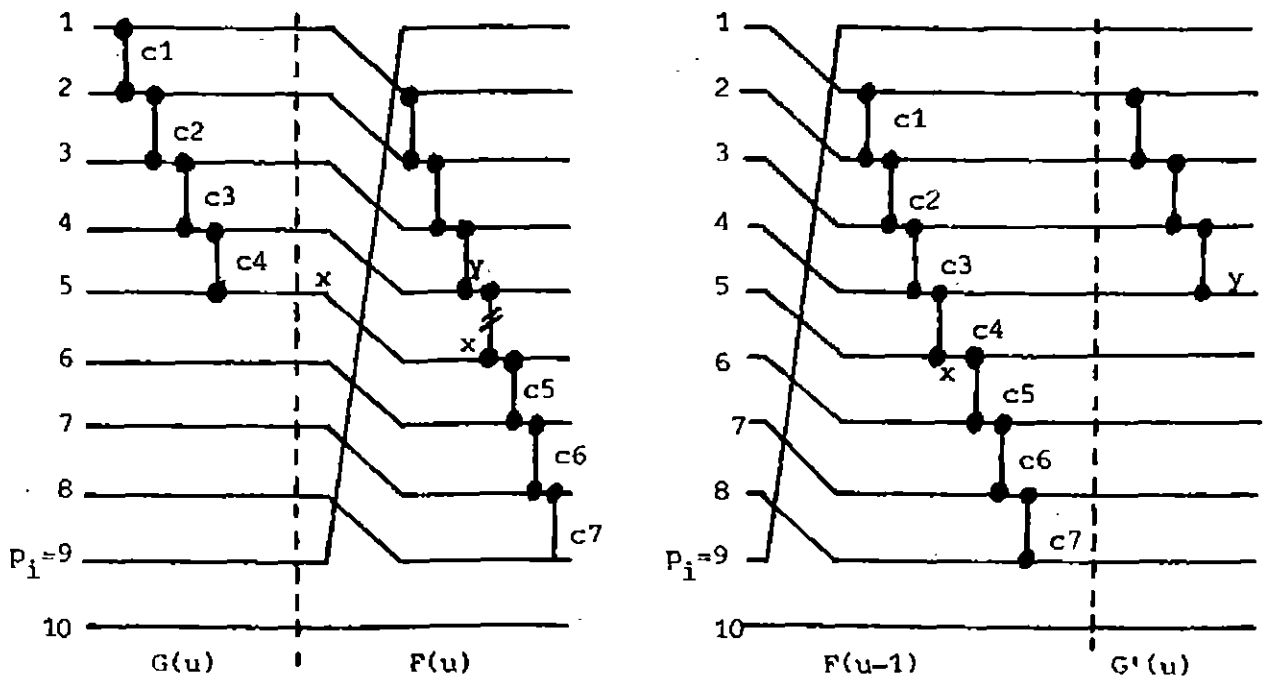

Figure 6(b). Depth of G less than $D(p_i, G)$.

the network GF has one more comparator than FG'. We will show that the comparator with the double crosshatch is redundant in the GF network, in which case the remaining networks are identical. To see this, let x denote the smallest of the first k elements, and y the second smallest, where k is the depth of G. (k=5 in the figure.) In GF, x will appear on the k^{th} output line of G, becoming the lower input of the marked comparator; and F will place y on the upper input of this comparator. Since x has lower priority than y, this comparator is redundant.

Appendix 2 - Real Time MIN Analyzer Algorithm

Suppose the linear array S[1:n] implements the MIN stack S(t) for all t, where the (variable) n is the number of pages referenced through time t. The LRU stack is implemented as a linked list using

HEAD, TAIL, LINK[1:n], DIST[1:n],

such that HEAD and TAIL are the pages at the beginning and end, respectively, of the LRU stack, LINK[i] is the successor of page i, and DIST[i] is the position of i in the LRU stack. Page i is an LRU elder of x if and only if DIST[i] > DIST[x]. Suppose r(t)=x. The algorithm to update the state of the system must perform these computations:

```
S  :=  F(S);
d(t) := position of x in S;
S  :=  G(S);
update LRU stack.
```

As noted earlier, the G function is implemented simply by exchanging the referenced page with the top page in the corrected stack. In case of a first reference there is no correction; it is necessary only to place the referenced page at the $n+1^{st}$ position of S and at the tail of the LRU stack before performing G.

Initially, all variables are zero. Once d(1)...d(t) are generated, the distance d(t+1) is generated by this procedure:

```
t := t+1
x := r(t)

{perform F}
if DIST[x] = 0
   then {first reference: add x to end of the MIN
         and LRU stacks; set m to position of x in S}
      n := n+1
      m := n
      S[n] := x
      LINK[x] := TAIL
      TAIL := x
      output(#)
   else {reference of page x is not first}
      1: "correct S, set m to position of x in S"

{perform G}
exchange( S[1], S[m] )
2: "update LRU stack"
```

The correction procedure (label 1) operates in two stages. The first moves a pointer m down until an LRU elder of the referenced page x is found; this will be the MIN position of x after correction. The second

advances a pointer p down from m until x is found, maintaining a variable y which is the eldest of the LRU elders of x so far observed. The refinement of 1 is:

```
1: {find x or first LRU elder of x}
   m := 1
   while DIST[ S[m] ] < DIST[x] do m := m+1

   {do pairwise updating of LRU
   elders of x, until x found}
   p := m
   y := S[m]
   while x ≠ S[p] do
      if DIST[y] < DIST[ S[p] ] then exchange(y, S[p])
      p := p+1
   end

   {place x in proper position}
   S[m] := x
   S[p] := y
```

The LRU stack updating proceudre locates the desired page in the LRU stack, unlinks it from its current position, and links it to the head. The LRU distances of intervening pages are increased by 1. This leads to a refinement of 2:

```
2:  if HEAD = x then return
    p := HEAD
    while p ≠ x do
       DIST[p] := DIST[p]+1
       q := p
       p := LINK[p]
    end

    {unlink p, by making successor of p be successor
    of the predecessor q of p, move p to top}
    LINK[q] := LINK[p]
    LINK[x] := HEAD
    HEAD := x
    DIST[x] := 1
    if p = TAIL then TAIL := q
```

References

[Bel66] Belady, L. A., "A study of replacement algorithms for a virtual storage computer," IBM Sys. J. 5, 2 (1966) 78-101.

[BeP74] Belady, L. A., and Palermo, F. P., "On-line measurement of paging behavior by the multivalued MIN algorithm," IBM J. of R & D 18, 1 (Jan 1974).

[CoD73] Coffman, E. G., and Denning, P. J., Operating Systems Theory, Prentice-Hall, 1973.

[Den75] Denning, P. J., "The compuation and use of optimal paging curves," Purdue Univ., Computer Sciences Dep't, TR-148, June 1975.

[DeS76] Denning, P. J. and Slutz, D. R., "Generalized working set and optimal measures for segment reference strings," Purdue Univ., Computer Sciences Dep't, TR-178, March 1976.

[LeN74] Lewis, C. H., and Nelson R. A., "Some one-pass algorithms for the generation of OPT distance strings," IBM T. J. Watson Research Center Report RC 4758 (March 1974).

[MGS70] Mattson, R. L., Gecsei, J., Slutz, D. R., and Traiger, I., "Evaluation techniques for storage hierarchies," IBM Sys J. 9, 2 (1970), 78-101.

[PrF76] Prieve, B. G., and Fabry, R. S., "VMIN - an optimal variable space page replacement algorithm," Comm. ACM 19, 5 (May 1976).

[Bel] Belady, L.A., "On-Line system for measuring the efficiency of replacement algorithms," U. S. Patent 3,577,185.