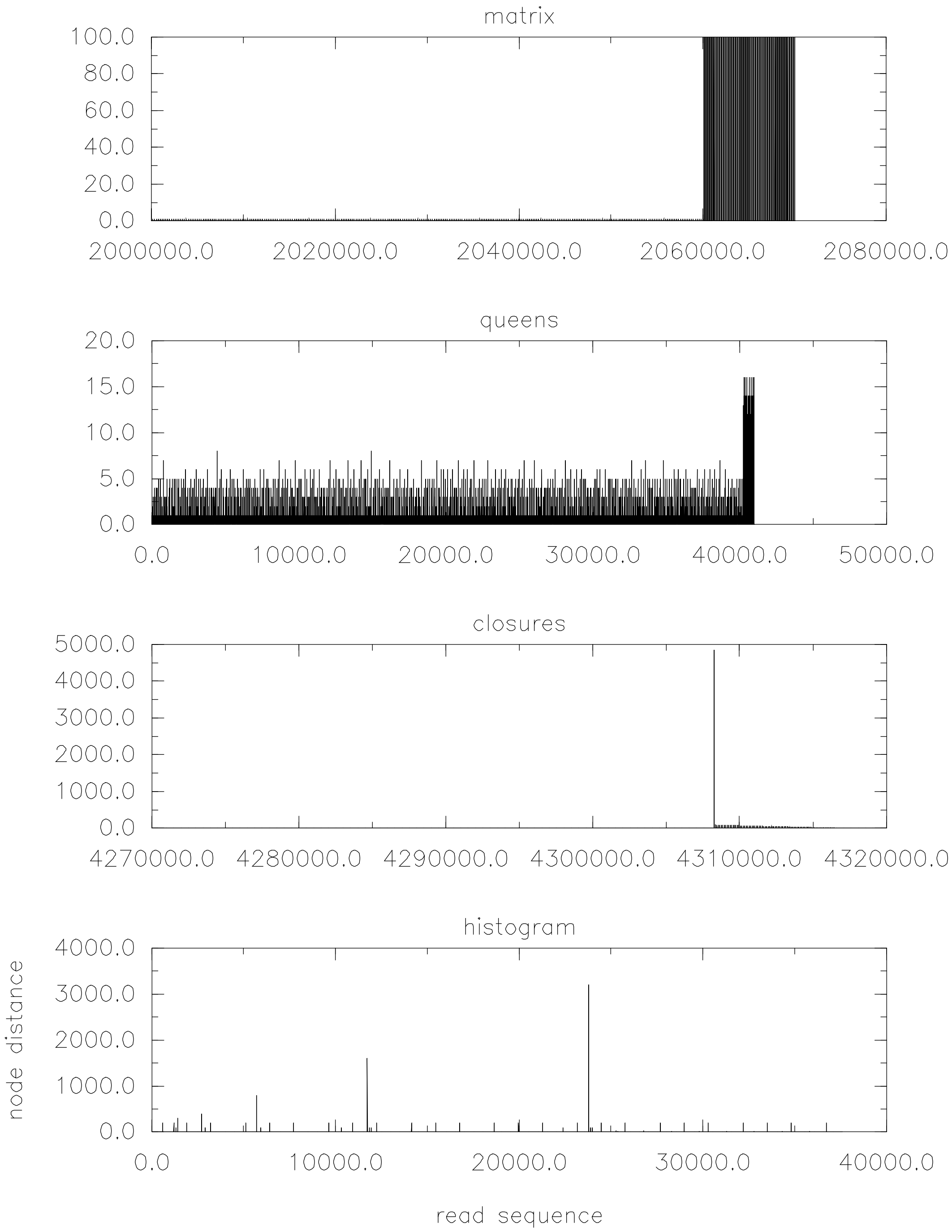

Figure 5.5: Trace graphs of the read sequences of four programs.

Note. The X–axis is for the step of the read sequence, and the Y–axis is for the distance between the current root node and the node to be read. Due to the large number of reads in the matrix and the histogram programs, only parts of their traces are shown. Note that each graph has its own scale.

	create	update	read	root node (cache size)	non–root node	rerooting	rotation
matrix	202	40200	2070102	202 (100)	40200	20202	40200
queens	1	2056	41018	1 (8)	2056	2056	5020
closures	1	9803	5328354	49 (100)	9803	14851	27415
histogram	2	20101	37625	60 (100)	20001	25070	37713

Table 5.1: Operation counts of the four programs.

	destructive	fragmented shallow	shallow	copying
matrix	43.08 (1.47)	50.38 (5.91)	50.32 (5.09)	84.73 (10.34)
queens	not applicable	1.12 (0.07)	1.10 (0.05)	1.10 (0.05)
closures	not applicable	58.99 (2.19)	59.07 (2.22)	66.00 (5.54)
histogram	not applicable	4.23 (1.95)	19.26 (12.06)	1455.45 (603.03)

Table 5.2: Execution time of the four programs.

which is the total space used, is about the magnitude of the number of updates. The number of rotations, which is the total time spent in rerooting nodes, is about the magnitude of the number of read operations too. This shows that, in an amortized sense, an update operation consumes constant space and a read operation costs constant time (see Corollary 5.4.55). Also note that the total number of reads being performed is far greater than the total number of updates being performed in each of the four programs. These statistics are very likely the result of incremental updates/voluminous reads execution patterns, which make the fragmented shallow binding scheme very much applicable. Figure 5.5 and figure 5.6 show the traces of the four programs' read sequences. The traces show that most of the entire read sequence in each of the four programs is voluminous. This can be observed from the fact that after a read operation causing a long rerooting, which records a large node distance between the new root node and the old root node in the trace graph, the subsequent read operations often need few rotations to make the accessed nodes the root nodes.

Table 5.2 shows the execution times of the four programs. The programs are implemented in Standard ML of New Jersey, version 0.66, and run on a Sparc workstation (Sun 4/290, with 32 MB of physical memory) Each entry in the table is of the format **total time (garbage collection time)**, as provided by the `System.Timer` module of SML of New Jersey. Each datum in the table is the average of the data from three runs. The SML/NJ implementation of fully persistent arrays by the fragmented shallow binding scheme is included