SIAM J. DISCRETE MATH.
Vol. 28, No. 1, pp. 206–225

REPLACEMENT PATHS VIA ROW MINIMA OF CONCISE MATRICES*

CHENG-WEI LEE† AND HSUEH-I LU‡

Abstract. Matrix M is *k-concise* if the finite entries of each column of M consist of k or fewer intervals of identical numbers. We give an $O(n+m)$-time algorithm to compute the row minima of any $O(1)$-concise $n \times m$ matrix. Our algorithm yields the first $O(n+m)$-time reductions from the replacement-paths problem on an n-node m-edge undirected graph (respectively, directed acyclic graph) to the single-source shortest-paths problem on an $O(n)$-node $O(m)$-edge undirected graph (respectively, directed acyclic graph). That is, we prove that the replacement-paths problem is no harder than the single-source shortest-paths problem on undirected graphs and directed acyclic graphs. Moreover, our linear-time reductions lead to the first $O(n+m)$-time algorithms for the replacement-paths problem on the following classes of n-node m-edge graphs: (1) undirected graphs in the word-RAM model of computation, (2) undirected planar graphs, (3) undirected minor-closed graphs, and (4) directed acyclic graphs.

Key words. paths and cycles, planar graphs, graph algorithms, data structures

AMS subject classifications. 05C38, 05C10, 05C85, 68P05

DOI. 10.1137/120897146

1. Introduction.

Computing a shortest path between two nodes in a graph is one of the most fundamental algorithmic problems in computer science. The variant of the shortest-path problem which asks for a shortest path between two nodes that avoids a failed node or edge has also been extensively studied in the last few decades. Let G be a graph. For any node v of G, let $G - v$ denote the graph obtained from G by deleting v and its incident edges. For any edge e of G, let $G - e$ denote the graph obtained from G by deleting e. For any subgraph G' of G, let $w(G')$ be the sum of edge weights of G'. An *rs-path* is a path from node r to node s. The *distance* $d_G(r, s)$ from r to s in G is the minimum of $w(P)$ over all rs-paths P of G. A *shortest* rs-path P of G satisfies $w(P) = d_G(r, s)$. We study the following two versions of the *replacement-paths problem* on G with respect to a given shortest rs-path P of G:

- The *edge-avoiding version* computes $d_{G-e}(r, s)$ for all edges e of P.
- The *node-avoiding version* computes $d_{G-v}(r, s)$ for all nodes v of P other than r and s.

The edge-avoiding version can be reduced in linear time to the node-avoiding version: Let G' be the graph obtained from G by subdividing each edge xy of P into two edges xv and vy with $w(xv) = w(vy) = w(xy)/2$. We have $d_{G-xy}(r, s) = d_{G'-v}(r, s)$. No linear-time reduction for the other direction is known. See, e.g., [21, 9, 31] for applications of the problem. Extensive surveys of the long history of algorithms and applications of this problem can be found in [14, 35]. We show that the replacement-paths

*Received by the editors October 31, 2012; accepted for publication (in revised form) October 28, 2013; published electronically February 6, 2014.
http://www.siam.org/journals/sidma/28-1/89714.html
†Department of Computer Science and Information Engineering, National Taiwan University, 106 Taipei, Taiwan (r99922035@ntu.edu.tw).
‡Department of Computer Science and Information Engineering, Graduate Institute of Networking and Multimedia, and Graduate Institute of Biomedical Electronics and Bioinformatics, National Taiwan University, 106, Taipei, Taiwan (hil@csie.ntu.edu.tw, www.csie.ntu.edu.tw/~hil). This author's research was supported in part by NSC grant 101–2221–E–002–062–MY3.

problem on an n-node m-edge undirected graph can be reduced in $O(n+m)$ time to the single-source shortest-paths problem on an $O(n)$-node $O(m)$-edge undirected graph.

THEOREM 1.1. *Let G be an n-node m-edge undirected graph. Let P be a given shortest rs-path of G, where r and s are two distinct nodes of G. Given distances $d_G(r,v)$ and $d_G(v,s)$ for all nodes v of G, we have the following statements:*

1. *It takes $O(n+m)$ time to solve the edge-avoiding replacement-paths problem on G with respect to P.*
2. *The node-avoiding replacement-paths problem on G with respect to P can be reduced in $O(n+m)$ time to the problem of computing distances $d_{G_0}(r_0,v)$ for some node r_0 and all nodes v of an $O(n)$-node $O(m)$-edge undirected graph G_0.*

Combining with Dijkstra's single-source shortest-paths algorithm (see, e.g., [12]), Theorem 1.1 solves the replacement-paths problem in $O(m+n\log n)$ time, matching the best known result for the edge-avoiding version of Malik, Mittal, and Gupta [26] and that for the node-avoiding version of Nardelli, Proietti, and Widmayer [30]. Combining with the algorithm of Henzinger et al. [19], Theorem 1.1 yields an $O(n+m)$-time algorithm for both versions of the problem on planar graphs, while $O(n+m)$-time algorithms on planar graphs were known only for the edge-avoiding version (see Bhosle [7]). Combining with the algorithm of Tazari and Müller-Hannemann [36], Theorem 1.1 leads to the first $O(n+m)$-time algorithm on minor-closed graphs. Combining with the algorithms of Thorup [38, 37], Theorem 1.1 solves both versions of the problem in $O(n+m)$ time in the word-RAM model of computation, improving upon the $O(m\cdot\alpha(m,n))$-time transmuter-based algorithm of Nardelli, Proietti, and Widmayer [29], which works only for the edge-avoiding version. See [32] for more results of the single-source shortest-paths problem that can be combined with our reductions to yield efficient algorithms for the replacement-paths problem.

Our proof of Theorem 1.1 also holds for directed acyclic graphs. Since the single-source shortest-paths problem can be solved in linear time on directed acyclic graphs (see, e.g., [12]), we solve both versions of the replacement-paths problem on any n-node m-edge directed acyclic graph in $O(n+m)$ time, improving upon the algorithm of Bhosle [7] for the edge-avoiding version, which runs in $O(m+n\cdot\alpha(2n,n))$ time in the word-RAM model of computation and runs in $O(m\cdot\alpha(m,n))$ time in general.

THEOREM 1.2. *For any two nodes r and s of an n-node m-edge directed acyclic graph G, it takes $O(n+m)$ time to solve the replacement-paths problem on G with respect to any given shortest rs-path of G.*

Table 1.1 compares our results with previous work.

TABLE 1.1
Previous work and our results on the replacement-paths problem.

	Edge-avoiding version	Node-avoiding version	Ours
Directed graph	$O(mn+n^2\log\log n)$ [17]	$O(mn+n^2\log n)$ [12]	
Directed acyclic graph	$O(m+n\cdot\alpha(m,n))$ [7]		$O(m+n)$
Directed acyclic graph (RAM)	$O(m+n\cdot\alpha(2n,n))$ [7]		$O(m+n)$
Undirected graph	$O(m+n\log n)$ [26]	$O(m+n\log n)$ [30]	$O(m+n\log n)$
Undirected graph (RAM)	$O(m\cdot\alpha(m,n))$ [29]		$O(m+n)$
Undirected planar graph	$O(n)$ [7]		$O(n)$
Undirected minor-closed graph			$O(n)$

1.1. Technical overview. A matrix M is *k-concise* if the finite entries of each column of M consist of k or fewer intervals of identical numbers. A 1-concise matrix is *concise*. Figure 1.1(a) shows a concise matrix. Figure 1.1(b) shows a 2-concise matrix. A k-concise matrix may not be sparse, but each column of a k-concise matrix can be concisely represented by $O(k)$ numbers, i.e., three numbers for each of the k or fewer intervals of identical finite numbers: (a) the starting row index, (b) the ending row index, and (c) the identical number of the interval. For instance, the columns with indices v_6v_5, v_7v_4, and v_9v_5 of the 2-concise matrix in Figure 1.1(b) can be represented by $\langle 1,1,13;2,4,12\rangle$, $\langle 2,2,20;3,3,16\rangle$, and $\langle 3,3,19;4,4,9\rangle$, respectively. Throughout the paper, all matrices are in this *concise representation*. The *row-minima problem* on a matrix M is to compute the minimum of each row of M. We show that the replacement-paths problem on an n-node m-edge undirected (respectively, directed acyclic) graph can be reduced in $O(n+m)$ time to the row-minima problem on a 2-concise $n \times m$ matrix obtainable from the solution to the single-source shortest-paths problem on an $O(n)$-node $O(m)$-edge undirected (respectively, directed acyclic) graph. (See Lemma 2.1 in section 2.1 for the edge-avoiding version and Lemma 2.2 in section 2.2 for the node-avoiding version.) Our reductions exploit the structure properties of replacement paths studied by, e.g., Malik, Mittal, and Gupta [26], Nardelli, Proietti, and Widmayer [30, 29], and Bhosle [7]. To show that the replacement-paths problem is no harder than the single-source shortest-paths problem, we give the first $O(n+m)$-time algorithm for the row-minima problem on any $O(1)$-concise $n \times m$ matrix (see Lemma 3.1 in section 3). As illustrated by Figure 1.2, for any k-concise $n \times m$ matrix N with $k = O(1)$, it takes $O(m)$ time to derive concise $n \times m$ matrices $N_1, N_2, \ldots, N_k$ whose entrywise minimum is N. Thus, the main technical challenge lies in computing the row minima of an $n \times m$ concise matrix M in $O(n+m)$ time. The rest of the overview elaborates on our $O(n+m)$-time algorithm for the row-minima problem on any concisely represented $n \times m$ concise matrix M.

The *thickness* θ of M is the length of a longest interval of identical finite entries over all columns of M. For instance, the thickness of the matrix in Figure 1.1(a) (respectively, Figures 1.2(a) and 1.2(b)) is 4 (respectively, 2 and 3). The *broadness*

M	v_0v_6	v_0v_8	v_6v_7	v_6v_5	v_7v_4	v_9v_5
1	13	15				
2		15	18	12		
3		15		12	16	
4				12	16	9
5				12		9

(a)

N	v_0v_8	v_6v_7	v_6v_5	v_7v_4	v_9v_5
1	15	19	13		
2	15		12	20	
3			12	16	19
4			12		9

(b)

FIG. 1.1. (a) *A concise* 5×6 *matrix* M. (b) *A* 2*-concise* 4×5 *matrix* N. *The* ∞*-entries in* M *and* N *are left out.*

N_1	v_0v_8	v_6v_7	v_6v_5	v_7v_4	v_9v_5
1	15	19	13		
2	15			20	
3					19
4					

(a)

N_2	v_0v_8	v_6v_7	v_6v_5	v_7v_4	v_9v_5
1					
2			12		
3			12	16	
4			12		9

(b)

FIG. 1.2. *Two concise* 4×5 *matrices* N_1 *and* N_2 *whose entry-wise minimum is the* 2*-concise* 4×5 *matrix* N *of Figure* 1.1(b). *The* ∞*-entries of* N_1 *and* N_2 *are left out.*

β of M is the minimum of (i) the number of distinct starting row indices for the intervals of finite entries over all columns of M, and (ii) the number of distinct ending row indices for the intervals of finite entries over all columns of M. For instance, the broadness of the matrix in Figure 1.1(a) (respectively, Figures 1.2(a) and 1.2(b)) is 4 (respectively, 3 and 2). The row minima of M can be computed in $O(n + m + \theta \cdot \beta)$ time by Lemma 3.4 in section 3.1. The thickness and broadness of M can both be as large as n, so applying Lemma 3.4 on M may require $\Omega(n^2)$ time. Our $O(n+m)$-time algorithm is based upon the technique of deriving matrices with smaller thickness or broadness whose row minima yield the row minima of M. (Details are in the proofs of Lemma 3.1 in section 3.3 and Lemma 3.5 in section 3.1.) Specifically, we derive four n-row matrices M_0, M_1, M_2, M_3 from M according to some positive integral *brush factor* h such that the row minima of M is the entrywise minima of the row minima of the four matrices. A column of M is h-*brushed* if it contains at least one finite entry in rows $h, 2h, \ldots, \lfloor \frac{n}{h} \rfloor \cdot h$. For instance, all columns of the matrix in Figure 1.3(a) are 3-brushed. Matrix M_0 is the submatrix of M induced by the non-h-brushed columns. See Figure 1.4(a) for a matrix M_0 that has no 3-brushed columns. Matrices M_1, M_2, and M_3 represent the h-brushed columns of M: Matrix M_1 takes over the first h or less finite entries of each h-brushed column of M up to the first row with a finite entry whose index is an integral multiple of h; matrix M_3 takes over the last $h - 1$ or less finite entries of each h-brushed column of M starting from the row with a finite entry that immediately succeeds the last row whose index is an integral multiple of h; and matrix M_2 takes over the finite entries of each h-brushed column in between. The entrywise minimum of matrices M_1, M_2, and M_3 is the submatrix of M induced by the h-brushed columns. See Figures 1.3(b)–1.3(d) for M_1, M_2, and M_3 obtained from M in Figure 1.3(a) with brush factor $h = 3$. Matrices M_1 and M_3 have thickness $O(h)$ and broadness $O(\frac{n}{h})$, so the row minima of M_1 and M_3 can be computed in $O(n + m)$ time by Lemma 3.4 for any choice of h. In order to compute the row minima of M_0 and M_2 in $O(n + m)$ time, we let $h = \Theta(\log \log n)$ and resort to two intermediate algorithms for row-minima problem. As to be explained in the next two paragraphs, we (1) apply the first intermediate algorithm on an $O(\frac{n}{h})$-row $O(m)$-column matrix obtained from M_2 by condensing its identical rows and (2) apply the second intermediate algorithm on $O(h)$-

M	1	2	3	4	5	μ	q	z
1	9					9	9	1
2	9	7				7	7	1
3	9	7	5			5	5	1
4		7	5			5		0
5		7	5	6		5		0
6		7	5	6		5		0
7		7	5	6	8	5		0
8		7	5	6	8	5		0
9		7		6	8	6	6	1
10				6	8	6		0
11					8	8	8	1

(a)

M_1	1	2	3	4	5
1	9				
2	9	7			
3	9	7	5		
4					
5				6	
6				6	
7					8
8					8
9					8
10					
11					

(b)

M_2	2	3	4
1			
2			
3			
4	7	5	
5	7	5	
6	7	5	
7	7		6
8	7		6
9	7		6
10			
11			

(c)

M_3	3	4	5
1			
2			
3			
4			
5			
6			
7	5		
8	5		
9			
10		6	8
11			8

(d)

FIG. 1.3. *Each column of matrix M is 3-brushed. The minima array μ of M and its corresponding query array q and auxiliary binary string z are displayed to the right of M. The entries of q that do not matter are left out. Matrix M_1 has thickness 3 and broadness 3. Every three consecutive rows of M_2 are identical. Matrix M_3 has thickness 2 and broadness 2. The ∞-entries in these four matrices are left out. Matrix M_1 has no all-∞ columns. The all-∞ columns of M_2 and M_3 are omitted.*

M_0	1	2	3	4	5
1	9				
2	9				
3					
4		7	5		
5			5		
6					
7				6	
8					
9					
10					8
11					8

(a)

M_1	1
1	9
2	9

(b)

M_2	2	3
4	7	5
5		5

(c)

M_3	4
7	6
8	

(d)

M_4	5
10	8
11	8

(e)

FIG. 1.4. *M_0 has no 3-brushed columns. The row minima of M_0 can be obtained from combining the row minima of M_1, M_2, M_3, and M_4. The ∞-entries in the matrices are left out.*

row matrices derived from M_0 whose overall number of rows (respectively, columns) is $O(n)$ (respectively, $O(m)$).

The broadness of the matrix M_2 obtained in the previous paragraph is $O(\frac{n}{h})$. Although the thickness of M_2 could be $\Omega(n)$, every h consecutive rows of M_2 are identical. See Figure 1.3(c) for an example of M_2 with $h = 3$. We condense matrix M_2 into an $O(\frac{n}{h})$-row $O(m)$-column matrix M^*. By $h = \Theta(\log\log n)$, it takes $O(n + m)$ time to compute the row minima of M_2 by applying our first intermediate algorithm (see Lemma 3.2 in section 3.1) on the condensed matrix M^*. For the rest of the paragraph, let M (with slight abuse of notation) be the input $n \times m$ matrix of this $O(m + n\log\log n)$-time intermediate algorithm, which is based upon the above technique of reducing thickness and broadness in a more complicated manner. We first partition M into submatrices $M_1, M_2, \ldots, M_\ell$ with $\ell = O(\log\log n)$ in $O(m + n\log\log n)$ time. Specifically, let $h_0, h_1, \ldots, h_\ell$ be a decreasing sequence of positive integers such that $h_0 \geq n$, $h_1 < n$, $h_\ell = 1$, and $h_{k-1} = \Theta(h_k^2)$ holds for each $k = 1, 2, \ldots, \ell$. Let M_k be the submatrix of M induced by the h_k-brushed columns that are not h_{k-1}-brushed, implying that M_k has thickness $O(h_{k-1}) = O(h_k^2)$. For each $n \times m_k$ matrix $N = M_k$ with $1 \leq k \leq \ell$, we derive three $n \times m_k$ matrices N_1, N_2, and N_3 with brush factor $h = h_k$ (again, as in the proof of Lemma 3.5 in section 3.1 and as illustrated by Figure 1.3). Both N_1 and N_3 have thickness $O(h_k)$ and broadness $O(\frac{n}{h_k})$. Since every h_k consecutive rows of N_2 are identical and N_2 are not h_{k-1}-brushed, we condense N_2 into an $O(\frac{n}{h_k})$-row m_k-column matrix N^* with thickness $O(h_k)$ and broadness $O(\frac{n}{h_k})$. The row minima of N_1, N^*, and N_3 can be computed in $O(n + m_k)$ time by Lemma 3.4. The row minima of $M_k = N$ can be obtained from those of N_1, N^*, and N_3 in $O(n)$ time. Taking entrywise minima on the row minima of $M_1, M_2, \ldots, M_\ell$, we have the row minima of M in time $\sum_{1\leq k\leq \ell} O(m_k + n) = O(m + n\log\log n)$.

The thickness of the matrix M_0 obtained in the paragraph preceding the previous paragraph is $O(h)$. Since M_0 has no h-brushed columns, one can partition the finite entries of M_0 into $O(h)$-row matrices $M_1, M_2, \ldots, M_\ell$ with $\ell = O(\frac{n}{h})$ whose overall number of columns is $O(m)$. See Figure 1.4 for an illustration. (Details are in the proof of Lemma 3.1 in section 3.3.) Recursively applying the procedure described in the previous two paragraphs on $M_1, \ldots, M_\ell$ would only lead to an $O((m+n)\log^* n)$-time algorithm. Instead, by $h = \Theta(\log\log n)$, the row minima of each $O(h)$-row

m_k-column matrix M_k can be computed in $O(m_k + \log\log n)$ time by our second intermediate algorithm (i.e., Algorithm 1 in Figure 3.2 in the proof of Lemma 3.6 in section 3.2). Putting together the row minima of $M_1, M_2, \ldots, M_\ell$, we solve the row-minima problem on M_0 in time $\sum_{1\le k\le \ell} O(m_k + \log\log n) = O(m+n)$. This $O(m_k + \log\log n)$-time intermediate algorithm for the row-minima problem on any $O(\log\log n)$-row m_k-column matrix M_k proceeds iteratively with the help of two data structures. For each $j = 1, 2, \ldots, m_k$, at the end of the jth iteration, the first data structure keeps the minimum of the first j columns of each row in a concise manner such that the minima of consecutive rows can be efficiently updated. Specifically, let $\mu(i)$ be the minimum of the first j entries of row i. An array q and a binary string z satisfying $q(\mathit{pred}(z,i)) = \mu(i)$ for all row indices i are used to represent array μ, where $\mathit{pred}(z,i)$ denotes the largest index i_1 with $i_1 \le i$ and $z(i_1) = 1$. The value of $\mu(i)$ can be obtained from $q(\mathit{pred}(z,i))$. Updating $\mu(i)$ for all indices i with $\mathit{pred}(z,i) = i_1$ to a smaller value can be done by decreasing $q(i_1)$. See Figure 1.3(a) for an example of μ, q, and z with $j = 5$. If the index $\mathit{pred}(z,i)$ for each i were $O(1)$-time computable and the value of $z(i)$ for each i were $O(1)$-time updatable, then our Algorithm 1 in Figure 3.2 in section 3.2 would have been an $O(n+m)$-time algorithm for the row-minima problem on any $n \times m$ matrix. However, it is impossible in general to come up with a polynomial-sized dynamic data structure for binary string z that supports both $O(1)$-time update on $z(i)$ and $O(1)$-time query $\mathit{pred}(z,i)$ [4]. Fortunately, the binary string z needed to represent the minima array μ of the $O(h)$-row matrix M_k has only $O(h) = O(\log\log n)$ bits. Thus, one can precompute all possible updates and queries on z in $o(n)$ time and organize all the precomputed information in an $o(n)$-space table capable of supporting each query and update on z in $O(1)$ time. With the help of this second data structure, our second intermediate algorithm computes the row minima of each M_k with $1 \le k \le \ell$ in $O(m_k + \log\log n)$ time.

1.2. Related work. On directed graphs with nonnegative weights, Gotthilf and Lewenstein [17] gave the best known algorithm, running in $O(mn + n^2 \log\log n)$ time, for the edge-avoiding version of the replacement-paths problem. The $O(mn+n^2\log n)$-time algorithm of running Dijkstra's $O(m + n\log n)$-time algorithm for $O(n)$ times remains the best known algorithm for the node-avoiding version. Bernstein [5] gave an algorithm to output $(1+\epsilon)$-approximate solutions for both versions of the problem for any positive parameter ϵ. Hershberger, Suri, and Bhosle [22] showed a lower bound $\Omega(m\sqrt{n})$ on the time complexity of the problem in the path-comparison model of Karger, Koller, and Phillips [24]. The randomized algorithm of Roditty and Zwick [35] on unweighted directed graphs runs in $\tilde{O}(m\sqrt{n})$ time. On directed graphs with integral weights in $\{-W, \ldots, W\}$, Weimann and Yuster [41, 42] gave an $\tilde{O}(Wn^\omega + W^{2/3}n^{1+2\omega/3})$-time randomized algorithm for both versions of the problem, where ω is the infimum of all numbers such that multiplying two $n \times n$ matrices takes $\tilde{O}(n^\omega)$ time. The running time was improved to $\tilde{O}(Wn^\omega)$ by Vassilevska Williams [39], who [40] recently reduced the long-standing upper bound on ω of Coppersmith and Winograd [11] from $\omega < 2.376$ to $\omega < 2.3727$. Recently, Grandoni and Vassilevska Williams [18] addressed the single-source version of the problem. On directed planar graphs with nonnegative weights, the algorithm of Wulff-Nilsen [43] runs in $O(n\log n)$ time, improving on the $O(n\log^3 n)$-time algorithm of Emek, Peleg, and Roditty [14] and the $O(n\log^2 n)$-time algorithm of Klein, Mozes, and Weimann [25]. Erickson and Nayyeri [16] extended Wulff-Nilsen's result on bounded-genus graphs.

Bernstein and Karger [6] addressed the all-pairs replacement-paths problem by giving an $\tilde{O}(n^2)$-space $\tilde{O}(mn)$-time data structure capable of answering $d_{G-v}(r, s)$ for any nodes r, s, and v of directed graph G in $O(1)$ time. Baswana, Lath, and Mehta [3] studied the single-source and all-pairs replacement-paths problems on directed planar graphs. Malik, Mittal, and Gupta [26] studied replacement paths that avoid multiple failed edges. Duan and Pettie [13] studied replacement paths that avoid two failed nodes or edges. Weimann and Yoster [42] studied replacement paths that avoid multiple failed nodes and edges. Chechik et al. [10] studied near optimal replacement paths that avoid multiple failed edges.

For the closely related problem of finding k shortest rs-paths for any given nodes r and s of directed graph G with nonnegative edge weights, Eppstein [15] gave an $O(m+n \log n + k)$-time algorithm, which may output nonsimple paths. If the output paths are required to be simple, the best currently known algorithm, also due to Gotthilf and Lewenstein [17], uses replacement paths. Specifically, Roditty and Zwick [35] showed that the problem can be reduced to $O(k)$ computations of the second shortest simple rs-path. Therefore, the replacement-paths algorithm of Gotthilf and Lewenstein yields an $O(kmn + kn^2 \log\log n)$-time algorithm for the problem of finding k shortest simple paths. See [34, 5, 20] for more results on this related problem. See [25, 2, 1, 28, 27, 8, 33, 23] for results involving the row-minima problem on matrices with special structures.

1.3. Road map. The rest of the paper is organized as follows. Section 2 gives the preliminaries, including our $O(n + m)$-time reductions for both versions of the replacement-paths problem on an n-node m-edge undirected graph to (1) the row-minima problem on $O(1)$-concise $n \times m$ matrices and (2) the single-source shortest-paths problem on $O(n)$-node $O(m)$-edge undirected graphs. Both reductions also work for directed acyclic graphs. Section 3 gives our $O(n + m)$-time algorithm for the row-minima problem on any $O(1)$-concise $n \times m$ matrix and proves Theorems 1.1 and 1.2. Section 4 concludes the paper.

2. Preliminaries. Let $|S|$ denote the cardinality of set S. A row (respectively, column) of a matrix is *dummy* if all its entries are ∞. Given distances $d_G(r, v)$ for all nodes v of an n-node m-edge graph G, a shortest-paths tree T in G rooted at r that contains the given shortest rs-path P can be obtained in $O(m + n)$ time. Let p be the number of edges in P. Let $v_0, v_1, \ldots, v_p$ be the nodes of P from $r = v_0$ to $s = v_p$. For each $i = 1, 2, \ldots, p$, let e_i be edge $v_{i-1}v_i$. See Figures 2.1(a) and 2.1(b) for an example of G, T, and P.

Subsection 2.1 gives our reduction for the edge-avoiding version. Subsection 2.2 gives our reduction for the node-avoiding version. Our reductions are presented in a way that also works for directed acyclic graphs. The reductions for directed acyclic graphs hold even with the existence of negative-weighted edges, while the reductions for undirected graphs assume nonnegative edge weights. We comment on handling negative weights for undirected graphs in section 4.

2.1. A reduction for the edge-avoiding version. For each node v of G, let *level* $\lambda(v)$ of v in T be the largest index i such that v_i is on the path of T from r to v. Levels $\lambda(v)$ for all nodes v of G can be computed from T in $O(n)$ time. For each $i = 1, 2, \ldots, p$,

- let R_i consist of the nodes x with $\lambda(x) \leq i - 1$ and
- let $\bar{R}_i$ consist of the nodes y with $\lambda(y) \geq i$.

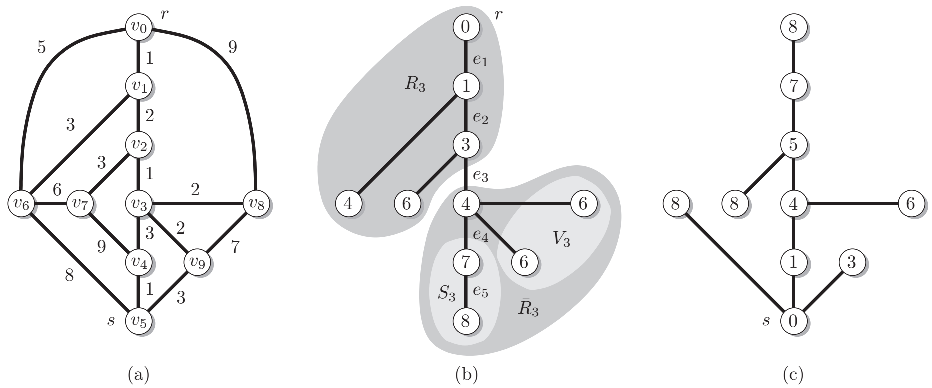

FIG. 2.1. (a) *Graph G in which $(v_0, v_1, \ldots, v_5)$ is a shortest rs-path P.* (b) *A shortest-paths tree T of G rooted at r, in which P consists of edges $e_1, e_2, \ldots, e_5$. The number in each node is its distance from r in G.* (c) *A shortest-paths tree T' of G rooted at s. The number in each node is its distance to s in G.*

That is, R_i (respectively, $\bar{R}_i$) consists of the nodes v that are reachable (respectively, unreachable) from r in $T - e_i$. See Figure 2.1(b) for an illustration of R_i and $\bar{R}_i$. For any edge xy of G with $\lambda(x) < \lambda(y)$, define

$$\textit{replacement-cost}_1(x, y) = d_G(r, x) + w(xy) + d_G(y, s).$$

Since R_i and $\bar{R}_i$ define a cut between nodes r and s, any rs-path of G contains some edge xy with $x \in R_i$ and $y \in \bar{R}_i$. We have

$$d_{G-e_i}(r, s) = \min\{\textit{replacement-cost}_1(x, y) \mid x \in R_i,\ y \in \bar{R}_i, \text{ and } xy \in G - e_i\} \tag{2.1}$$

for each $i = 1, 2, \ldots, p$ (see also, e.g., [29, 26]). The *edge-replacement matrix* of G with respect to T and P is the $p \times m$ matrix M defined by

$$M(i, xy) = \begin{cases} \textit{replacement-cost}_1(x, y) & \text{if } \lambda(x) < i \leq \lambda(y) \text{ and } e_i \neq xy, \\ \infty & \text{otherwise} \end{cases}$$

for each $i = 1, 2, \ldots, p$ and each edge xy of G with $\lambda(x) < \lambda(y)$. For instance, the matrix in Figure 1.1(a) is the edge-replacement matrix of the graph G in Figure 2.1(a) with respect to the tree T and path P in Figure 2.1(b), where the dummy columns are omitted. Let G' be the graph obtained from G by reversing the direction of each edge of G. (This statement handles the case that G is a directed acyclic graph. For the undirected case, we simply have $G = G'$.) The distances $d_G(v, s)$ for all nodes v of G and a shortest-paths tree T' in G' rooted at s can be obtained from each other in $O(m + n)$ time. See Figure 2.1(c) for an example of T'.

LEMMA 2.1. *The edge-replacement matrix M of G with respect to T and P is a concise matrix whose concise representation can be obtained from G, P, T, and T' in $O(n + m)$ time. Moreover, for each $i = 1, 2, \ldots, p$, the minimum of the ith row of M equals $d_{G-e_i}(r, s)$.*

Proof. By definition of M, if the xyth column of M is not dummy, then $\lambda(x) \neq \lambda(y)$. Let x and y be the endpoints of such an edge with $\lambda(x) < \lambda(y)$. The entries of the xyth column in rows $\lambda(x) + 1, \lambda(x) + 2, \ldots, \lambda(y)$ are all *replacement-cost*$_1(x, y)$. The other entries are all ∞. Since each column of M consists of at most one interval of identical finite numbers, M is concise. Given G, P, T, and T', values *replacement-cost*$_1(x, y)$ for all edges xy of G with $\lambda(x) < \lambda(y)$ can be obtained in overall $O(n+m)$ time. Matrix M can be obtained from G, P, T, and T' in $O(n+m)$ time. The minimum of the ith row is the minimum of *replacement-cost*$_1(x, y)$ over all edges xy of G with $\lambda(x) < i \leq \lambda(y)$ and $e_i \neq xy$. By definition of R_i and $\bar{R}_i$, edge xy satisfies $\lambda(x) < i \leq \lambda(y)$ if and only if $x \in R_i$ and $y \in \bar{R}_i$. By (2.1), the minimum of the ith row of M is indeed $d_{G-e_i}(r, s)$. The lemma is proved. ∎

2.2. A reduction for the node-avoiding version. Observe that the level $\lambda(v)$ of node v in T is also the smallest index i such that v is reachable from r in $T - v_{i+1}$. For each $i = 1, \ldots, p-1$, let the nodes of $G - v_i$ be partitioned into R_i, V_i, and S_i, where

- R_i, as defined in section 2.1, consists of the nodes x with $\lambda(x) \leq i - 1$,
- V_i consists of the nodes $x \neq v_i$ with $\lambda(x) = i$, and
- S_i consists of the nodes y with $\lambda(y) > i$.

See Figure 2.1(b) for an illustration of R_i, V_i, and S_i, where V_i and S_i are depicted by lighter shaded regions. Since $R_i \cup V_i$ and S_i define a cut for nodes r and s in $G - v_i$, each rs-path of $G - v_i$ contains some edge xy with $x \in R_i \cup V_i$ and $y \in S_i$. For any node subset U of G, let $G[U]$ denote the subgraph of G induced by U. We have

(2.2)
$$\begin{aligned} d_{G-v_i}(r,s) &= \min\{d_{G[R_i \cup V_i]}(r,x) + w(xy) + d_G(y,s) \mid x \in R_i \cup V_i,\, y \in S_i,\, xy \in G\} \\ &= \min\{\min\{d_G(r,x) + w(xy) + d_G(y,s) \mid x \in R_i,\, y \in S_i,\, xy \in G\}, \\ &\quad \times \min\{d_{G[R_i \cup V_i]}(r,x) + w(xy) + d_G(y,s) \mid x \in V_i,\, y \in S_i,\, xy \in G\}\}, \end{aligned}$$

where the first equality is proved by Nardelli, Proietti, and Widmayer [30, Lemma 3] and the second equality follows from the observation that $d_{G[R_i \cup V_i]}(r, x) = d_G(r, x)$ holds for each node $x \in R_i$.

We now define a graph G_0 and specify a node r_0 of G_0 such that

$$d_{G[R_i \cup V_i]}(r, x) = d_{G_0}(r_0, x) \tag{2.3}$$

holds for each $i = 1, 2, \ldots, p-1$ and each node $x \in V_i$. For each $i = 1, 2, \ldots, p-1$, let G_i be $G[V_i]$ plus one new node r_i and $|V_i|$ new edges, where for each node $x \in V_i$ the xth new edge is $r_i x$ with weight

$$w(r_i x) = \min\{d_G(r, u) + w(ux) \mid u \in R_i, ux \in G\}.$$

Let graph G_0 be $G_1 \cup G_2 \cup \cdots \cup G_{p-1}$ plus a new node r_0 and $p-1$ zero-weighted edges $r_0 r_1, r_0 r_2, \ldots, r_0 r_{p-1}$. G_0 is the disjoint union of $p-1$ induced subgraphs of G plus a tree with internal nodes $r_0, r_1, \ldots, r_{p-1}$. For the case that G is a directed acyclic graph, all edges of the tree are outgoing toward the disjoint union of the $p-1$ induced subgraphs of G, which is acyclic. G_0 has to be a directed acyclic graph. For the case that G is planar, the disjoint union of the $p-1$ induced subgraphs of G is planar. If edge $r_i x$ for some node $x \in V_i$ has finite edge weight, x has at least one neighbor of G in R_i. Although G_0 may not be planar, the subgraph of G_0 induced by the edges with finite edge weights has to be planar. Let T_0 be a shortest-paths tree of G_0 rooted at r_0. See Figure 2.2 for an example. Observe that G_0 is an $O(n)$-node $O(m)$-edge graph, obtainable in $O(n+m)$ time from G and T, such that (2.3) holds for each $i = 1, 2, \ldots, p-1$. For any edge xy of G with $\lambda(x) < \lambda(y)$, define

$$\textit{replacement-cost}_2(x, y) = d_{G_0}(r, x) + w(xy) + d_G(y, s).$$

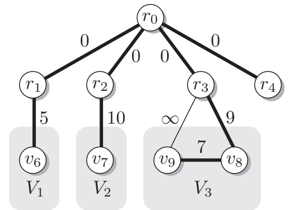

FIG. 2.2. *The graph G_0 obtained from the graph G in Figure* 2.1(a) *and the tree T and path P in Figure* 2.1(b). *The edges in thick lines form a shortest-paths tree T_0 of G_0 rooted at r_0.*

The *node-replacement matrix* of G with respect to T and P is the $(p-1) \times m$ matrix N defined by

$$N(i, xy) \begin{cases} \textit{replacement-cost}_2(x,y) & \text{if } \lambda(x) = i < \lambda(y) \text{ and } x \neq v_i, \\ \textit{replacement-cost}_1(x,y) & \text{if } \lambda(x) < i < \lambda(y) \text{ and } x \neq v_i, \\ \infty & \text{otherwise} \end{cases}$$

for each $i = 1, 2, \ldots, p-1$ and each edge xy of G with $\lambda(x) < \lambda(y)$. For instance, the matrix in Figure 1.1(b) is the node-replacement matrix of the graph G in Figure 2.1(a) with respect to the tree T and path P in Figure 2.1(b), where the dummy columns are omitted.

LEMMA 2.2. *The node-replacement matrix N of G with respect to T and P is a* 2-*concise matrix whose concise representation can be obtained from G, P, T, T', and T_0 in $O(n+m)$ time. Moreover, for each $i = 1, 2, \ldots, p-1$, the minimum of the ith row of N equals $d_{G-v_i}(r, s)$.*

Proof. By definition of N, if the xyth column of N with $\lambda(x) \leq \lambda(y)$ is not dummy, then $\lambda(x) + 1 \leq \lambda(y)$. The entry of the xyth column in row $\lambda(x)$ is *replacement-cost*$_2(x, y)$. If $\lambda(x) + 2 \leq \lambda(y)$, the entries of the xyth column in rows $\lambda(x)+1, \lambda(x)+2, \ldots, \lambda(y)-1$ are all *replacement-cost*$_1(x, y)$. The other entries of the xyth column are all ∞. Since the finite entries of each column of N consist of at most two intervals of identical numbers, N is 2-concise. Given G, P, T, T', and T_0, values *replacement-cost*$_1(x, y)$ and *replacement-cost*$_2(x, y)$ for all edges xy of G with $\lambda(x) < \lambda(y)$ can be obtained in overall $O(n+m)$ time. Matrix N can be obtained from G, P, T, T', and T_0 in $O(n+m)$ time. By (2.2) and (2.3), we have

$$\begin{aligned} d_{G-v_i}(r,s) = \min\{&\min\{\textit{replacement-cost}_1(x,y) \mid x \in R_i,\ y \in S_i,\ xy \in G\}, \\ &\min\{\textit{replacement-cost}_2(x,y) \mid x \in V_i,\ y \in S_i,\ xy \in G\}\}. \end{aligned}$$

For each $i = 1, \ldots, p-1$, the minimum of the ith row of N is indeed $d_{G-v_i}(r, s)$. The lemma is proved. □

3. The row minima of an $O(1)$-concise matrix in linear time. This section proves Lemma 3.1. Theorem 1.1 follows immediately from Lemmas 2.1, 2.2, and 3.1. Theorem 1.2 follows immediately from Lemma 3.1 and the analogous versions of Lemmas 2.1 and 2.2 for directed acyclic graphs.

LEMMA 3.1. *It takes $O(n+m)$ time to compute the row minima of a concisely represented $O(1)$-concise $n \times m$ matrix.*

As illustrated in Figure 1.2, a k-concise $n \times m$ matrix M with $k = O(1)$ can be decomposed in $O(m)$ time into k concise $n \times m$ matrices whose entry-wise minimum is M. To prove Lemma 3.1, it suffices to solve the row-minima problem on any $n \times m$ concise matrix in $O(n+m)$ time. For the rest of the section, all matrices are concise.

Each matrix M is concisely represented by arrays a_M, b_M, and c_M such that for each $i = 1, 2, \ldots, n$ and each $j = 1, 2, \ldots, m$, the (i, j)-entry of M can be determined in $O(1)$ time by

$$M(i,j) = \begin{cases} c_M(j) & \text{if } a_M(j) \le i \le b_M(j), \\ \infty & \text{otherwise.} \end{cases}$$

For instance, if M is the matrix in Figure 1.3(a), then $a_M = (1, 2, 3, 5, 7)$, $b_M = (3, 9, 8, 10, 11)$, and $c_M = (9, 7, 5, 6, 8)$. Subscripts M in a_M, b_M, and c_M can be omitted if matrix M is clear from the context.

Subsection 3.1 proves Lemma 3.2, which states an $O(m + n \log\log n)$-time algorithm for solving the row-minima problem on any $n \times m$ matrix. Subsection 3.2 proves Lemma 3.6, which states an $O(m + \log\log n)$-time algorithm for solving the row-minima problem on any $O(\log\log n) \times m$ matrix, with the help of an $O(n)$-time precomputable $O(n)$-space data structure that supports $O(1)$-time queries and updates on any $O(\log\log n)$-bit binary string. Subsection 3.3 proves Lemma 3.1 using Lemmas 3.2 and 3.6.

3.1. A near-linear-time intermediate algorithm. This subsection proves Lemma 3.2, which requires Lemmas 3.3, 3.4, and 3.5.

LEMMA 3.2. *It takes $O(m + n \log\log n)$ time to compute the row minima of an $n \times m$ matrix.*

An $n \times m$ matrix M is *sorted* if the following properties hold, where (a) M_i is the submatrix of M induced by the columns whose indices j satisfy $a_M(j) = i$, and (b) m_i is the number of columns in M_i.

Property S1: $a_M(1) \le a_M(2) \le \cdots \le a_M(m)$.

Property S2: $b_{M_i}(1) \le b_{M_i}(2) \le \cdots \le b_{M_i}(m_i)$ holds for each $i = 1, \ldots, n$.

That is, if M is sorted, then

$$(a_M(1), b_M(1)), (a_M(2), b_M(2)), \ldots, (a_M(m), b_M(m))$$

are in lexicographically nondecreasing order. For instance, the matrices M in Figures 1.1(a) and 1.3(a), the matrix M_0 in Figure 1.4(a), and the matrix M_9 in Figure 3.1 are sorted. The matrix N_1 in Figure 1.2 is not sorted, since the column with index v_0v_8 is not the third column.

LEMMA 3.3. *It takes $O(n + m)$ time to reorder the columns of an $n \times m$ matrix such that the resulting matrix is sorted.*

Proof. Since $a(j)$ and $b(j)$ for all indices $j = 1, 2, \ldots, m$ are positive integers in $\{1, 2, \ldots, n\}$, the lemma is straightforward by counting sort (see, e.g., [12]). ∎

M_9	1	2	3	4	5	6	7	8	9	10	row minimum
9	*3*	95	25	66	32	76	51	88	76	81	3
10		*95*	25	66	32	76	51	88	76	81	25
11			*25*	66	32	76	51	88	76	81	25
12				*66*	32	76	51	88	76	81	32
13					*32*	76	51	88	76	81	32
14					*32*	76	51	88	76	81	32
15						*76*	*51*	88	76	81	51
16								*88*	76	81	76
17									*76*	*81*	76

FIG. 3.1. *A sorted n-row m_i-column thickness-θ matrix M_i with $n = 17$, $i = 9$, $m_i = 10$, and $\theta = 9$. The dummy rows of M_i are omitted. The ∞-entries are left out. The italic entries form the lower-left boundary of the finite entries.*

Define

$$\begin{aligned} \textit{thickness}(M) &= \max\{b_M(j) - a_M(j) + 1 \mid 1 \le j \le m\}; \\ \textit{broadness}(M) &= \min\{|\{a_M(1), a_M(2), \dots, a_M(m)\}|, |\{b_M(1), b_M(2), \dots, b_M(m)\}|\}. \end{aligned}$$

We have *thickness*(M) = *broadness*$(M) = 4$ for the matrix M in Figure 1.1(a) and *thickness*$(M_9) = 9$ and *broadness*$(M_9) = 1$ for the matrix M_9 in Figure 3.1.

LEMMA 3.4. *It takes* $O(n + m + \textit{thickness}(M) \cdot \textit{broadness}(M))$ *time to compute the row minima of an* $n \times m$ *matrix* M.

Proof. Let $\theta = \textit{thickness}(M)$ and $\beta = \textit{broadness}(M)$. Subscripts M of a_M and b_M in the proof are omitted. We prove the lemma for the case with $\beta = |\{a(1), a(2), \dots, a(m)\}|$. The case with $\beta = |\{b(1), b(2), \dots, b(m)\}|$ can be proved by reversing the row order of M. We first apply Lemma 3.3 to have M sorted in $O(n + m)$ time. For each $i = 1, 2, \dots, n$, let M_i be the submatrix of M induced by columns whose indices j satisfy $a(j) = i$. Let m_i be the number of columns in M_i. For each of the β indices i with $m_i \ge 1$, the nondummy rows of submatrix M_i are all in rows $i, i + 1, \dots, i + \theta - 1$. Since $a(j) = i$ holds for all column indices j of M_i, the sequence of minima of rows $i, i + 1, \dots, i + \theta - 1$ of M_i is nondecreasing. By Property S2 of M, the minima of the θ or fewer nondummy rows of M_i can be computed in $O(m_i + \theta)$ time by a right-to-left and bottom-up traversal of the lower-left boundary of the finite entries. See Figure 3.1 for an illustration. The row minima of M can be obtained from the row minima of the nondummy rows of the β matrices M_i with $m_i \ge 1$ in $O(n + \theta \cdot \beta)$ time. The row-minima problem on M can thus be solved in $O(n + m + \theta \cdot \beta)$ time. The lemma is proved. ∎

For any positive integer h, we say that the jth column of M is *h-brushed* if interval $[a_M(j), b_M(j)]$ contains at least one integral multiple of h. It takes $O(1)$ time to determine from $a_M(j)$ and $b_M(j)$ whether the jth column of M is h-brushed or not.

LEMMA 3.5. *If* M *is an* $n \times m$ *matrix whose columns are all* h*-brushed, then the row-minima problem on* M *can be reduced in* $O(n + m)$ *time to the row-minima problem on an* $O(\frac{n}{h}) \times m$ *matrix* M^* *with* $\textit{thickness}(M^*) = O(\frac{1}{h} \cdot \textit{thickness}(M))$ *and* $\textit{broadness}(M^*) = O(\frac{n}{h})$.

Proof. Let M_1, M_2, and M_3 be the following three $n \times m$ matrices, obtainable from M in $O(m)$ time, whose entrywise minimum is M. For each $i = 1, 2, \dots, n$ and each $j = 1, 2, \dots, m$, let

$$M_1(i, j) = \begin{cases} M(i, j) & \text{if } a_M(j) \le i \le h \cdot \left\lceil \frac{a_M(j)}{h} \right\rceil, \\ \infty & \text{otherwise,} \end{cases}$$

$$M_2(i, j) = \begin{cases} M(i, j) & \text{if } h \cdot \left\lceil \frac{a_M(j)}{h} \right\rceil + 1 \le i \le h \cdot \left\lfloor \frac{b_M(j)}{h} \right\rfloor, \\ \infty & \text{otherwise,} \end{cases}$$

$$M_3(i, j) = \begin{cases} M(i, j) & \text{if } h \cdot \left\lfloor \frac{b_M(j)}{h} \right\rfloor + 1 \le i \le b_M(j), \\ \infty & \text{otherwise.} \end{cases}$$

See Figure 1.3 for an example. Since each $b_{M_1}(j)$ with $1 \le j \le m$ is an integral multiple of h, we have *broadness*$(M_1) = O(\frac{n}{h})$. Since each $a_{M_3}(j) - 1$ with $1 \le j \le m$ is an integral multiple of h, we have *broadness*$(M_3) = O(\frac{n}{h})$. By Lemma 3.4 with *thickness*$(M_1) = O(h)$ and *thickness*$(M_3) = O(h)$, the row-minima problems on M_1 and M_3 can be solved in $O(n+m)$ time. Every h consecutive rows of M_2 are identical. Specifically, for each positive index t, rows $(t - 1) \cdot h + 1, (t - 1) \cdot h + 2, \dots, t \cdot h$ of

M_2 are identical. Let M_2 be condensed into an $O(\frac{n}{h}) \times m$ matrix M^* by merging every h consecutive rows of M_2 into a single row. We have *thickness*$(M^*) = O(\frac{1}{h} \cdot$ *thickness*$(M))$ and *broadness*$(M^*) = O(\frac{n}{h})$. The row minima of M_2 can be obtained from those of M^* in $O(n)$ time. The lemma is proved. □

We are ready to prove Lemma 3.2.

Proof of Lemma 3.2. Let M be the input $n \times m$ matrix. We first apply Lemma 3.3 to have M sorted in $O(n+m)$ time. Let $\ell = 1 + \lceil \log_2 \log_2 n \rceil$. Assume $n \geq 2$ without loss of generality, so $\ell \geq 1$. Define a decreasing sequence $h_0, h_1, \ldots, h_\ell$ of positive integers as follows:

$$h_k = \begin{cases} 2^{2^{\ell-k-1}} & \text{if } 0 \leq k \leq \ell - 1, \\ 1 & \text{if } k = \ell. \end{cases}$$

Each h_k is a power of two. One can verify that $h_0 \geq n$, $h_1 < n$, $h_{\ell-1} = 2$, and $h_{k-1} = h_k^2$ holds for each $k = 1, 2, \ldots, \ell - 1$. For each $k = 1, 2, \ldots, \ell$, if k is the smallest positive integer such that the jth column of M is h_k-brushed, then let $j \in J_k$. By $h_\ell = 1$, sets $J_1, J_2, \ldots, J_\ell$ form a disjoint partition of the indices of the nondummy columns of M. For the matrix in Figure 3.1 with $n = 17$, we have $\ell = 4$, $h_0 = 256$, $h_1 = 16$, $h_2 = 4$, $h_3 = 2$, $h_4 = 1$, $J_4 = \{1\}$, $J_3 = \{2, 3\}$, $J_2 = \{4, 5, 6, 7\}$, and $J_1 = \{8, 9, 10\}$. For each $k = 1, 2, \ldots, \ell$, let $j_k = |J_k|$. By $j_1 + j_2 + \cdots + j_\ell = m$, the lemma follows immediately from the following two statements:

Statement 1: Sets $J_1, J_2, \ldots, J_\ell$ can be obtained from M in $O(m + n \cdot \ell)$ time.

Statement 2: For each $k = 1, 2, \ldots, \ell$, the row-minima problem on the submatrix of M induced by the columns with indices in J_k can be solved in $O(n + j_k)$ time.

Statement 1. For each $i = 1, 2, \ldots, n$, let M_i be the submatrix of M induced by the columns whose indices j satisfy $a_M(j) = i$. Let m_i be the number of columns in M_i. For each $i = 1, 2, \ldots, n$ and each $j = 1, 2, \ldots, m_i$, let $\kappa(i, j)$ be the index k such that J_k contains the index of the column of M that is the jth column of M_i. Let $\kappa(i, 0) = \ell$. Since $h_1, h_2, \ldots, h_k$ are all integral multiples of h_k for each $k = 1, 2, \ldots, \ell$, Property S2 of M implies $\kappa(i, 0) \geq \kappa(i, 1) \geq \cdots \geq \kappa(i, m_i) \geq 1$. For each $j = 1, 2, \ldots, m_i$, to determine $\kappa(i, j)$, it suffices to look for the first integer k starting from $\kappa(i, j-1)$ down to 1 such that the jth column of M_i is h_k-brushed but not h_{k-1}-brushed. Therefore, it takes overall $O(m_i + \ell)$ time to compute indices $\kappa(i, 1), \kappa(i, 2), \ldots, \kappa(i, m_i)$. Sets $J_1, J_2, \ldots, J_\ell$ can thus be obtained in $O(m + n \cdot \ell)$ time. Statement 1 is proved.

Statement 2. Let M_k be the submatrix of M induced by the columns with indices in J_k. If $j \in J_k$, then the jth column of M is not h_{k-1}-brushed, implying *thickness*$(M_k) < h_{k-1} = O(h_k^2)$. By Lemma 3.5, the row-minima problem on M_k can be reduced in $O(n + j_k)$ time to the row-minima problem on an $O(\frac{n}{h_k}) \times j_k$ matrix M_k^* with *thickness*$(M_k^*) = O($*thickness*$(M_k) \cdot \frac{1}{h_k}) = O(h_k)$ and *broadness*$(M_k^*) = O(\frac{n}{h_k})$. By Lemma 3.4, the row minima of M_k^* can be computed in time $O(\frac{n}{h_k} + j_k + h_k \cdot \frac{n}{h_k}) = O(n + j_k)$. Therefore, the row minima of M_k can be computed in $O(n + j_k)$ time. Statement 2 is proved. The lemma is proved. □

3.2. A linear-time intermediate algorithm for matrices with very few rows. This subsection proves the following lemma.

LEMMA 3.6. *Let n be a given positive integer. Let $h = \max(1, \lceil \log_2 \log_2 n \rceil)$. It takes $O(n)$ time to compute an $O(n)$-space data structure, with which the row minima of any $h \times m$ matrix can be computed in $O(h + m)$ time.*

Initialization:	Let $q(0) = \infty$, $z(0) = 1$, and $z(1) = z(2) = \cdots = z(h+1) = 0$.
For-loop:	For each $j = 1, 2, \ldots, m$, execute the following steps.
Step 1:	Let $i_0 = a(j)$, $i_2 = b(j) + 1$, and $i_1 = pred(z, i_2 - 1)$.
Step 2:	If $c(j) \geq q(i_1)$, then proceed to the next iteration of the for-loop.
Step 3:	If $z(i_2) = 0$, then let $z(i_2) = 1$ and $q(i_2) = q(i_1)$.
Step 4:	While $i_0 \leq i_1$ and $c(j) < q(i_1)$, execute the following substep.
Substep 4*a*:	Let $z(i_1) = 0$, $i_2 = i_1$, and $i_1 = pred(z, i_2 - 1)$.
Step 5:	If $c(j) < q(i_1)$, then let $z(i_0) = 1$ and $q(i_0) = c(j)$.
Step 6:	If $c(j) > q(i_1)$, then let $z(i_2) = 1$ and $q(i_2) = c(j)$.

FIG. 3.2. *Algorithm* 1*: Computing the row minima for an $h \times m$ sorted concise matrix concisely represented by arrays a, b, and c.*

Proof. Let z be a binary string. For each index $i \geq 1$, let $z(i)$ denote the ith bit of z. Let $pred(z, i_2)$ be the largest index i_1 with $i_1 \leq i_2$ and $z(i_1) = 1$. Let Z consist of all h-bit binary strings. By $|Z| = 2^h = O(\log n)$, it takes $o(n)$ time to construct an $o(n)$-space data structure capable of supporting each update to $z(i)$ and each query $pred(z, i)$ in $O(1)$ time.

Let M be the input $h \times m$ matrix. Subscripts M of a_M, b_M, and c_M are omitted in the proof. To avoid boundary conditions, let there be two additional dummy rows 0 and $h+1$ in M. We first apply Lemma 3.3 to have M sorted in $O(h+m)$ time. The proof needs only Property S1 of M, though. The algorithm proceeds iteratively, one iteration per column of M, obtaining $\mu(i) = \min\{M(i,1), M(i,2), \ldots, M(i,j)\}$ for all row indices $i = 1, 2, \ldots, h$ at the end of the jth iteration. As a result, at the end of the algorithm, we have the minimum of each row of M computed in the *minima array* μ. To support efficient dynamic updates and queries, we cannot afford to explicitly store each element of μ. Instead, we use an h-element *query array* q together with an *auxiliary binary string* z for q to represent μ such that $\mu(i) = q(pred(z, i))$ holds for each row index $i = 1, 2, \ldots, h$. Observe that if $z(i) = 0$, then the value of $q(i)$ does not matter. See Figures 1.3(a) and 3.3(a) for examples of μ, q, and z.

The algorithm is as shown in Figure 3.2. The initial binary string z has exactly one 1-bit. Each iteration of the for-loop increases the number of 1-bits in z by at most three via Steps 3, 5, and 6. Each iteration of the while-loop of Step 4 decreases the number of 1-bits in z by exactly one. Therefore, the overall number of times executing Substep 4a throughout all m iterations of the for-loop is $O(m)$. Since the initialization takes $O(h)$ time, Algorithm 1 runs in $O(m+h)$ time. The rest of the proof ensures the correctness of Algorithm 1.

For each $j = 0, 1, \ldots, m$, let μ_j, z_j, and q_j be the μ, z, and q at the end of the jth iteration, respectively. See Figure 3.3(b) for the query array q_j at the end of the jth iteration for each $j = 0, 1, \ldots, 7$ on the matrix M in Figure 3.3(a). By induction on the column index j, we prove

$$q_j(pred(z_j, i)) = \mu_j(i) \text{ for all indices } i \text{ with } 1 \leq i \leq h. \tag{3.1}$$

Equation (3.1) with $j = 0$ for all indices i with $1 \leq i \leq h$ follows immediately from the initialization of Algorithm 1. Assuming

$$q_{j-1}(pred(z_{j-1}, i)) = \mu_{j-1}(i) \text{ for all indices } i \text{ with } 1 \leq i \leq h \tag{3.2}$$

M	1	2	3	4	5	6	7	μ	q	z
1	8							8	8	1
2		3	7					3	3	1
3		3	7	6	6			3		0
4		3	7	6	6	3		3		0
5			7	6	6	3	7	3		0
6			7	6	6	3	7	3		0
7					6		7	6	6	1
8					6		7	6		0

(a)

	q_0	q_1	q_2	q_3	q_4	q_5	q_6	q_7
0	∞	∞	∞	∞	∞	∞	∞	∞
1		*8*	8	8	8	8	8	8
2		∞	*3*	3	3	3	*3*	3
3								
4								
5			∞	*7*	*6*	*6*		
6								
7				∞	∞		6	6
8								
9						∞	∞	∞

(b)

FIG. 3.3. (a) *A sorted* 8×7 *concise matrix* M*, the final minima array* μ *of* M*, the final query array* q *of* μ*, and the final auxiliary binary string* z*. The* ∞*-entries of* M *and the entries of* q *that do not matter are left out.* (b) *For each* $j = 0, 1, \ldots, 7$*, the query array* q_j *at the end of the* j*th iteration of the for-loop. The entries that do not matter are left out. The shaded cells of the* j*th column with* $1 \le j \le 6$ *indicate the indices* i_1 *and* i_2 *in the* j*th iteration. The italic cell of the* j*th column indicates the index* i^* *of the* j*th column. For instance, we have* $i_1 = 0$*,* $i^* = 1$*, and* $i_2 = 2$ *in the first iteration and* $i_1 = i^* = 2$ *and* $i_2 = 5$ *in the sixth iteration.*

holds with $j \ge 1$, we show (3.1) by the following analysis on the jth iteration of the for-loop. By Property S1 of M, we have $a(j) \ge \max\{a(1), a(2), \ldots, a(j-1)\}$, implying

$$\mu_{j-1}(a(j)) \le \mu_{j-1}(a(j)+1) \le \mu_{j-1}(a(j)+2) \le \cdots \le \mu_{j-1}(b(j)). \tag{3.3}$$

We first consider the case with $\mu_{j-1}(b(j)) \le c(j)$. See iteration 7 of the example in Figure 3.3 for an instance of this situation. By (3.3), the jth column of M does not affect the content of the minima array, i.e., $\mu_j = \mu_{j-1}$. By (3.2), at the end of Step 1, we have $q(i_1) = q_{j-1}(\mathit{pred}(z_{j-1}, b(j))) = \mu_{j-1}(b(j)) \le c(j)$. Therefore, Step 2 proceeds to the next iteration without altering the content of q and z. By $\mu_j = \mu_{j-1}$, $z_j = z_{j-1}$, and $q_j = q_{j-1}$, (3.1) follows from (3.2). The rest of the proof assumes $c(j) < \mu_{j-1}(b(j))$, implying that Steps 4, 5, and 6 are executed in the jth iteration.

To prove (3.1) for indices i with $b(j) < i \le h$, we first show that Steps 4, 5, and 6 do not alter the values of $z(i)$ and $q(i)$ for indices i with $b(j) < i \le h$. At the end of Step 3, condition $c(j) < q(i_1)$ holds. Step 6 sets $z(i_2) = 1$ and $q(i_2) = c(j)$ only if $c(j) > q(i_1)$, implying that Substep 4a executes at least once. We have $i_2 \le b(j)$ when Step 6 alters the values of $z(i_2)$ and $q(i_2)$. Observe that $\max(i_0, i_1) \le b(j)$ holds throughout the jth iteration. Therefore, Steps 4, 5, and 6 do not alter the values of $q(i)$ and $z(i)$ for indices i with $b(j) < i \le h$. By (3.2) and Step 3, we have $z_j(b(j)+1) = 1$ and $q_j(b(j)+1) = \mu_{j-1}(b(j)+1) = \mu_j(b(j)+1)$. Since $\mu_j(i) = \mu_{j-1}(i)$ holds for indices i with $b(j) < i \le h$, (3.1) for indices i with $b(j) < i \le h$ follows from (3.2) for indices i with $b(j) < i \le h$. See iterations 1–6 of the example in Figure 3.3 for instances of this situation: Step 3 alters the content of q and z in iterations 1–3 and 5–6; Step 3 does not alter the content of q and z in iteration 4.

It remains to prove (3.1) for indices i with $1 \le i \le b(j)$. After Step 1, we have $i_0 = a(j)$ for the rest of the jth iteration. Step 4 sets $z(i) = 0$ for each index i with $i_0 \le i \le b(j)$, $z_{j-1}(i) = 1$, and $c(j) < q_{j-1}(i)$. The following equations hold for the fixed values of indices i_1 and i_2 after Step 4 (i.e., during the execution of Steps 5 and 6):

$$i_0 > i_1 \text{ or } c(j) \geq \mu_{j-1}(i_1), \tag{3.4}$$

$$\mu_{j-1}(i) = q_{j-1}(i_1) \text{ for all indices } i \text{ with } i_1 \leq i \leq i_2 - 1. \tag{3.5}$$

Equation (3.4) is due to the fact that the condition of while-loop of Step 4 does not hold. Equation (3.5) follows from (3.2) and $i_1 = \mathit{pred}(z_{j-1}, i_2 - 1)$, as ensured by Step 1 and Substep 4a. By $c(j) < \mu_{j-1}(b(j))$, we have $\mu_j(b(j)) = c(j)$. Moreover, if $i_2 \leq b(j)$ (i.e., Substep 4a being executed at least once in the jth iteration), then (3.3) implies

$$\mu_j(i) = c(j) \text{ for all indices } i \text{ with } i_2 \leq i \leq b(j). \tag{3.6}$$

Let i^* be the smallest index with $i_1 \leq i^*$ and $\mu_j(i^*) = \mu_j(i^* + 1) = \cdots = \mu_j(b(j)) = c(j)$. In iterations 1–6 of the example in Figure 3.3, for each $j = 1, 2, \ldots, 6$, the i^*th entry of q_j is italic and the i_1th and i_2th entries of q_j with $i_1 < i_2$ are shaded in Figure 3.3(b). For instance, we have $(i_1, i_2, i^*) = (0, 2, 1)$ in iteration 1 and $(i_1, i_2, i^*) = (2, 5, 2)$ in iteration 6. One can verify

$$\mu_j(i) = \mu_{j-1}(i) \text{ for all indices } i \text{ with } 1 \leq i < i^* \tag{3.7}$$

as follows. For each index i with $1 \leq i < i_0$, we already have $\mu_j(i) = \mu_{j-1}(i)$, since the (i, j)-entry of M is ∞. Therefore, it remains to consider the case with $i_0 \leq i^* - 1$ and verify (3.7) for indices i with $i_0 \leq i \leq i^* - 1$. By (3.3), it suffices to ensure $\mu_j(i^* - 1) = \mu_{j-1}(i^* - 1)$. Assume $\mu_j(i^* - 1) \neq \mu_{j-1}(i^* - 1)$ for a contradiction. We have $\mu_{j-1}(i^* - 1) > \mu_j(i^* - 1) = c(j)$. By $\mu_j(i^* - 1) = c(j)$ and the definition of i^*, we have $i^* = i_1$, which implies $i_0 < i_1$. By $i_0 < i_1 = i^*$ and (3.4), we have $c(j) \geq \mu_{j-1}(i_1) = \mu_{j-1}(i^*)$, implying $\mu_{j-1}(i^* - 1) > c(j) \geq \mu_{j-1}(i^*)$. By definition of i^*, we have $i^* \leq b(j)$. However, $\mu_{j-1}(i^* - 1) > \mu_{j-1}(i^*)$ and $i_0 \leq i^* - 1 < b(j)$ contradict (3.3).

Assume $i_2 < i^*$ for a contradiction. By the definition of i^*, we have $i_2 \leq b(j)$, implying that Step 4a is executed at least once. By (3.6), $\mu_j(i) = c(j)$ holds for all indices i with $i_2 \leq i \leq b(j)$, which contradicts the definition of i^*. By $i^* \leq i_2$, we have $q(i) = q_{j-1}(i)$ and $z(i) = z_{j-1}(i)$ for all indices i with $1 \leq i < i^*$ at the end of Step 4. By $i_1 \leq i^*$, we have $z(i) = 0$ for all indices i with $i^* < i \leq b(j)$ at the end of Step 4. Combining with (3.7), in order to satisfy (3.1) for all indices i with $1 \leq i \leq b(j)$, it suffices for Steps 5 and 6 to additionally ensure $z(i^*) = 1$ and $q(i^*) = c(j)$. By the following case analysis, ensuring $z_j(i^*) = 1$ and $q_j(i^*) = c(j)$ is exactly what Steps 5 and 6 do:

- Case 0: $c(j) < q_{j-1}(i_1)$. We show $i^* = i_0$. By $c(j) < q_{j-1}(i_1) = \mu_{j-1}(i_1)$ and (3.4), we have $i_1 < i_0$. Before executing Step 4, we have $i_0 < i_2$. Each time Substep 4a is executed, the current value of i_2 equals the value of i_1 in the previous iteration of the while-loop, when condition $i_0 \leq i_1$ of the while-loop must hold. Regardless of whether Step 4a is executed, we have $i_0 \leq i_2$ at the end of Step 4. If $i_0 < i_2$, then $i_1 < i_0 < i_2$ and (3.5) imply $\mu_{j-1}(i_0) = q_{j-1}(i_1) > c(j)$. If $i_0 = i_2$, then i_0 equals the value of i_1 at the execution of Substep 4a for the last time, when condition $c(j) < q(i_1)$ of the while-loop must hold. Thus, we have $\mu_{j-1}(i_0) = q_{j-1}(i_0) > c(j)$. Either way, we have $\mu_{j-1}(i_0) > c(j)$. By $\mu_{j-1}(i_0) > c(j)$, and (3.3), we have $\mu_j(i) = c(j)$ for all indices i with $i_0 \leq i \leq b(j)$. By $i_1 < i_0$, we have $i^* \leq i_0$. By

$i_1 \le i_0 - 1 < i_2$ and (3.5), we have $\mu_j(i_0-1) = \mu_{j-1}(i_0-1) = q_{j-1}(i_1) > c(j)$, implying $i^* = i_0$.

- Case 1: $c(j) = q_{j-1}(i_1)$. We show $i^* = i_1$. By $c(j) = q_{j-1}(i_1)$ and the fact that condition $c(j) < q(i_1)$ holds at the end of Step 3, we know that Step 4a is executed at least once, implying $i_2 \le b(j)$ and (3.6). By $c(j) = q_{j-1}(i_1)$ and (3.5), we have $\mu_{j-1}(i) = c(j)$ and thus $\mu_j(i) = c(j)$ for all indices i with $i_1 \le i < i_2$. Therefore, $i^* = i_1$.
- Case 2: $c(j) > q_{j-1}(i_1)$. We show $i^* = i_2$. By $c(j) > q_{j-1}(i_1)$ and the fact that condition $c(j) < q(i_1)$ holds at the end of Step 3, we know that Step 4a is executed at least once. By (3.6), we have $i^* \le i_2$. By (3.5) and $c(j) > q_{j-1}(i_1)$, we have $c(j) > \mu_{j-1}(i_2 - 1)$, implying $\mu_j(i_2 - 1) < c(j)$. Therefore, $i^* = i_2$.

For Case 0, i.e., $i^* = i_0$, as illustrated by iterations 1 and 2 of the example in Figure 3.3, Step 5 correctly sets $z_j(i^*) = 1$ and $q_j(i^*) = c(j)$. For Case 2, i.e., $i^* = i_2$, as illustrated by iterations 3 and 4 of the example in Figure 3.3, Step 6 correctly sets $z_j(i^*) = 1$ and $q_j(i^*) = c(j)$. For Case 1, we have $i^* = i_1$, as illustrated by iterations 5 and 6 of the example in Figure 3.3. At the end of Step 4, we already have $z(i^*) = 1$ and $q(i^*) = c(j)$. Since Steps 5 and 6 do not alter the content of q and z, we also have $z_j(i^*) = 1$ and $q_j(i^*) = c(j)$. The lemma is proved. ∎

3.3. Proving Lemma 3.1. We are ready to prove the lemma of the section.

Proof of Lemma 3.1. It suffices to prove the lemma for the case that the input $n \times m$ matrix is concise. Let $h = \max(1, \lceil \log_2 \log_2 n \rceil)$. Let M be the submatrix of the input matrix induced by the h-brushed columns. By Lemma 3.5, the row-minima problem on M can be reduced in $O(n+m)$ time to the row-minima problem on an $O(\frac{n}{h}) \times O(m)$ matrix M^*. By Lemma 3.2, the row minima of M^* can be computed in time $O(\frac{n}{h} \log\log n + m) = O(n+m)$, which yield the row minima of M in $O(n+m)$ time.

Let M_0 be the submatrix of the input matrix induced by the columns that are not h-brushed. Let $\ell = \lceil \frac{n}{h} \rceil$. For each $k = 1, 2, \ldots, \ell$, let M_k be the submatrix of M_0 induced by the columns whose indices j satisfy $(k-1) \cdot h < a_{M_0}(j) \le b_{M_0}(j) < k \cdot h$ and the rows with indices $(k-1) \cdot h + 1, (k-1) \cdot h + 2, \ldots, k \cdot h - 1$. See Figure 1.4 for an illustration. Let m_k be the number of columns in M_k. By Lemma 3.6, the row minima of M_k can be computed in $O(h + m_k)$ time with the help of an $O(n)$-time precomputable data structure. As a result, the row-minima problems on all matrices M_k with $1 \le k \le \ell$ can be solved in overall time $O(n) + \sum_{1 \le k \le \ell} O(h + m_k) = O(n+m)$. The row minima of M_0 can be obtained from combining the row minima of $M_1, M_2, \ldots, M_\ell$ in $O(n+m)$ time. The lemma is proved. ∎

4. Concluding remarks. For directed acyclic graphs and undirected graphs, we give linear-time reductions for the replacement-paths problem to the single-source shortest-paths problem. The reductions are based upon our $O(n+m)$-time algorithm for the row-minima problem on an $O(1)$-concise $n \times m$ matrix, which is allowed to have negative entries. On the one hand, our reductions for directed acyclic graphs in sections 2.1 and 2.2 work even if there are negative-weighted edges. Therefore, we have shown that the replacement-paths problem on directed acyclic graphs with general weights is no harder than the single-source shortest-paths problem on directed acyclic graphs with general weights. On the other hand, our reductions for undirected graphs in sections 2.1 and 2.2 do assume nonnegativity of edge weights. However, it is not difficult to accommodate negative-weighted edges in undirected graphs for the replacement-paths problem, as is briefly explained in the next two paragraphs.

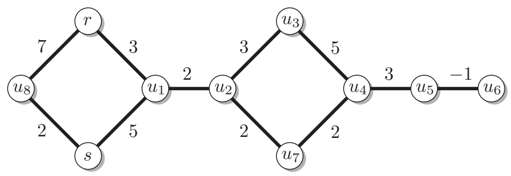

FIG. 4.1. *Two undirected connected graphs G with $d_G(r, s) = -\infty$.*

Let r and s be two nodes of the input connected undirected n-node m-edge graph G with negative-weighted edges. See Figure 4.1 for examples. We have $d_G(r, s) = -\infty$. G has no shortest rs-path. The input rs-path P must pass some negative-weighted edge an infinite number of times. For each edge $e \in P$, let G_e denote the connected component of $G - e$ that contains r. It takes overall $O(n + m)$ time to classify all edges e of P into the following three sets:

- Set 1: $s \notin G_e$. We have $d_{G-e}(r, s) = \infty$.
- Set 2: $s \in G_e$ and G_e has negative-weighted edges. We have $d_{G-e}(r, s) = -\infty$.
- Set 3: $s \in G_e$ and G_e has no negative-weighted edges. We have $d_{G-e}(r, s) = d_{G_e}(r, s)$.

It can be verified that if Set 3 is nonempty, then distances $d_{G_e}(r, s)$ are identical for all edges e of Set 3. See Figure 4.1(a) for an example. The edges in Set 3 are u_1u_2, u_4u_5, and u_5u_6. We have $d_{G-u_1u_2}(r, s) = d_{G-u_4u_5}(r, s) = d_{G-u_5u_6}(r, s) = 8$. Therefore, the replacement-paths problem on G with respect to P can be reduced in $O(n+m)$ time to the single-source shortest-paths problem on G_e for an arbitrary edge e in Set 3. As a result, the edge-avoiding version of the replacement-paths problem on undirected graphs with general weights is no harder than the single-source shortest-paths problem on undirected graphs with *nonnegative* weights.

The node-avoiding version of the replacement-paths problem is slightly more complicated. For each node $v \in P$ other than r and s, let G_v denote the connected component of $G - v$ that contains r. It takes overall $O(n + m)$ time to classify all nodes v of P other than r and s into the following three sets:

- Set 1′: $s \notin G_v$. We have $d_{G-v}(r, s) = \infty$.
- Set 2′: $s \in G_v$ and G_v has negative-weighted edges. We have $d_{G-v}(r, s) = -\infty$.
- Set 3′: $s \in G_v$ and G_v has no negative-weighted edges. We have $d_{G-v}(r, s) = d_{G_v}(r, s)$.

If Set 3′ is nonempty, then $d_{G_v}(r, s)$ are not necessarily identical for all nodes v of Set 3′. See Figure 4.1(b) for an example. The nodes in Set 3′ are u_1 and u_2. We have $d_{G-u_1}(r, s) = 9$ and $d_{G-u_2}(r, s) = 8$. However, one can show that there are at most two distinct values of $d_{G_v}(r, s)$ for all nodes v of Set 3′. Therefore, the node-avoiding version of the replacement-paths problem on undirected graphs with general weights is also no harder than the single-source shortest-paths problem on undirected graphs with *nonnegative* weights.

Our presentation focuses on computing the edge-avoiding and node-avoiding distances. It is not difficult to additionally report their corresponding edge-avoiding and node-avoiding shortest paths in $O(1)$ time per edge. For instance, given a shortest-paths tree T of G rooted at r and a shortest-paths tree T' of G' rooted at s as defined in section 2.1, if the xyth column of the edge-replacement matrix M contains the minimum of the ith row, then the union of (a) the rx-path in T, (b) the edge xy, and (c) the ys-path in T' is a shortest rs-path in $G - e_i$. The node-avoiding shortest

rs-path can be similarly obtained from T, T', and a shortest-paths tree T_0 of G_0 rooted at r_0 as defined in section 2.2.

It would be of interest to see results for the single-source, all-pairs, or near-optimal version of the problem of finding replacement paths in undirected graphs or directed acyclic graphs that avoid multiple failed nodes or edges.

Acknowledgment. We thank the anonymous reviewers for their helpful comments.

REFERENCES

[1] A. Aggarwal, M. M. Klawe, S. Moran, P. W. Shor, and R. E. Wilber, *Geometric applications of a matrix-searching algorithm*, Algorithmica, 2 (1987), pp. 195–208.

[2] M. J. Atallah and S. R. Kosaraju, *An efficient parallel algorithm for the row minima of a totally monotone matrix*, J. Algorithms, 13 (1992), pp. 394–413.

[3] S. Baswana, U. Lath, and A. S. Mehta, *Single source distance oracle for planar digraphs avoiding a failed node or link*, in Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, 2012, pp. 223–232.

[4] P. Beame and F. E. Fich, *Optimal bounds for the predecessor problem and related problems*, J. Comput. System Sci., 65 (2002), pp. 38–72.

[5] A. Bernstein, *A nearly optimal algorithm for approximating replacement paths and k shortest simple paths in general graphs*, in Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, 2010, pp. 742–755.

[6] A. Bernstein and D. Karger, *A nearly optimal oracle for avoiding failed vertices and edges*, in Proceedings of the 41st Annual ACM Symposium on Theory of Computing, 2009, pp. 101–110.

[7] A. M. Bhosle, *Improved algorithms for replacement paths problems in restricted graphs*, Oper. Res. Lett., 33 (2005), pp. 459–466.

[8] P. G. Bradford and K. Reinert, *Lower bounds for row minima searching*, in Proceedings of the 23rd International Colloquium on Automata, Languages and Programming, F. Meyer auf der Heide and B. Monien, eds., Lecture Notes in Comput. Sci. 1099, Springer, New York, 1996, pp. 454–465.

[9] T. H. Byers and M. S. Waterman, *Determining all optimal and near-optimal solutions when solving shortest path problems by dynamic programming*, Oper. Res., 32 (1984), pp. 1381–1384.

[10] S. Chechik, M. Langberg, D. Peleg, and L. Roditty, *f-sensitivity distance oracles and routing schemes*, Algorithmica, 63 (2012), pp. 861–882.

[11] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J. Symbolic Comput., 9 (1990), pp. 251–280.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed., MIT Press, Cambridge, MA, 2009.

[13] R. Duan and S. Pettie, *Dual-failure distance and connectivity oracles*, in Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms, 2009, pp. 384–391.

[14] Y. Emek, D. Peleg, and L. Roditty, *A near-linear-time algorithm for computing replacement paths in planar directed graphs*, ACM Trans. Algorithms, 6 (2010), pp. 64.1–64.13.

[15] D. Eppstein, *Finding the k shortest paths*, SIAM J. Comput., 28 (1998), pp. 652–673.

[16] J. Erickson and A. Nayyeri, *Computing replacement paths in surface embedded graphs*, in Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, 2011, pp. 1347–1354.

[17] Z. Gotthilf and M. Lewenstein, *Improved algorithms for the k simple shortest paths and the replacement paths problems*, Inform. Process. Lett., 109 (2009), pp. 352–355.

[18] F. Grandoni and V. Vassilevska Williams, *Improved distance sensitivity oracles via fast single-source replacement paths*, in Proceedings of the 53rd Annual IEEE Symposium on Foundations of Computer Science, 2012, pp. 748–757.

[19] M. R. Henzinger, P. Klein, S. Rao, and S. Subramanian, *Faster shortest-path algorithms for planar graphs*, J. Comput. System Sci., 55 (1997), pp. 3–23.

[20] J. Hershberger, M. Maxel, and S. Suri, *Finding the k shortest simple paths: A new algorithm and its implementation*, ACM Trans. Algorithms, 3 (2007), pp. 45.1–45.19.

[21] J. Hershberger and S. Suri, *Vickrey prices and shortest paths: What is an edge worth?* in Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science, 2001, pp. 252–259.
[22] J. Hershberger, S. Suri, and A. M. Bhosle, *On the difficulty of some shortest path problems*, ACM Trans. Algorithms, 3 (2007), pp. 5.1–5.15.
[23] H. Kaplan, S. Mozes, Y. Nussbaum, and M. Sharir, *Submatrix maximum queries in monge matrices and monge partial matrices, and their applications*, in Proceedings of the 23rd Annual ACM-SIAM Symposium on Discrete Algorithms, 2012, pp. 338–355.
[24] D. R. Karger, D. Koller, and S. J. Phillips, *Finding the hidden path: Time bounds for all-pairs shortest paths*, SIAM J. Comput., 22 (1993), pp. 1199–1217.
[25] P. N. Klein, S. Mozes, and O. Weimann, *Shortest paths in directed planar graphs with negative lengths: A linear-space $O(n \log^2 n)$-time algorithm*, ACM Trans. Algorithms, 6 (2010), pp. 30.1–30.18.
[26] K. Malik, A. K. Mittal, and S. K. Gupta, *The k most vital arcs in the shortest path problem*, Oper. Res. Lett., 8 (1989), pp. 223–227.
[27] S. Mozes and C. Wulff-Nilsen, *Shortest paths in planar graphs with real lengths in $O(n \log^2 n / \log \log n)$ time*, in Proceedings of the 18th Annual European Symposium on Algorithms, M. de Berg and U. Meyer, eds., Lecture Notes in Comput. Sci. 6347, Springer, New York, 2010, pp. 206–217.
[28] K. Nakano and S. Olariu, *An efficient algorithm for row minima computations on basic reconfigurable meshes*, IEEE Trans. Parallel Distributed Systems, 9 (1998), pp. 561–569.
[29] E. Nardelli, G. Proietti, and P. Widmayer, *A faster computation of the most vital edge of a shortest path*, Inform. Process. Lett., 79 (2001), pp. 81–85.
[30] E. Nardelli, G. Proietti, and P. Widmayer, *Finding the most vital node of a shortest path*, Theoret. Comput. Sci., 296 (2003), pp. 167–177.
[31] N. Nisan and A. Ronen, *Algorithmic mechanism design*, in Proceedings of the 31st Annual ACM Symposium on Theory of Computing, 1999, pp. 129–140.
[32] S. Pettie, *Single-source shortest paths*, in Encyclopedia of Algorithms, M.-Y. Kao, ed., Springer, New York, 2008, pp. 1–99.
[33] R. Raman and U. Vishkin, *Optimal randomized parallel algorithms for computing the row maxima of a totally monotone matrix*, in Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, 1994, pp. 613–621.
[34] L. Roditty, *On the k-simple shortest paths problem in weighted directed graphs*, in Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms, 2007, pp. 920–928.
[35] L. Roditty and U. Zwick, *Replacement paths and k simple shortest paths in unweighted directed graphs*, in Proceedings of the 32nd International Colloquium on Automata, Languages and Programming, L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, eds., Lecture Notes in Comput. Sci. 3580, Springer, New York, 2005, pp. 249–260.
[36] S. Tazari and M. Müller-Hannemann, *Shortest paths in linear time on minor-closed graph classes, with an application to Steiner tree approximation*, Discrete Appl. Math., 157 (2009), pp. 673–684.
[37] M. Thorup, *Undirected single-source shortest paths with positive integer weights in linear time*, J. ACM, 46 (1999), pp. 362–394.
[38] M. Thorup, *Floats, integers, and single source shortest paths*, J. Algorithms, 35 (2000), pp. 189–201.
[39] V. Vassilevska Williams, *Faster replacement paths*, in Proceedings of the 22nd Annual ACM-SIAM Symposium on Discrete Algorithms, 2011, pp. 1337–1346.
[40] V. Vassilevska Williams, *Multiplying matrices faster than Coppersmith-Winograd*, in Proceedings of the 44th Annual ACM Symposium on Theory of Computing, 2012, pp. 887–898.
[41] O. Weimann and R. Yuster, *Replacement paths via fast matrix multiplication*, in Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science, 2010, pp. 655–662.
[42] O. Weimann and R. Yuster, *Replacement paths and distance sensitivity oracles via fast matrix multiplication*, ACM Trans. Algorithms, 9 (2013), pp. 14.1–14.13.
[43] C. Wulff-Nilsen, *Solving the replacement paths problem for planar directed graphs in $O(n \log n)$ time*, in Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, 2010, pp. 756–765.