

Undirected 3-Fault Replacement Path in Nearly Cubic Time

Shucheng Chi ^{*1}, Ran Duan ^{†1}, Benyu Wang ^{‡2}, and Tianle Xie ^{§1}

¹Institute for Interdisciplinary Information Sciences, Tsinghua University

²University of Michigan, Ann Arbor

September 14, 2025

Abstract

Given a graph $G = (V, E)$ and two vertices $s, t \in V$, the f -fault replacement path (f FRP) problem computes for every set of edges F where $|F| \leq f$, the distance from s to t when edges in F fail. A recent result shows that 2FRP in directed graphs can be solved in $\tilde{O}(n^3)$ time [Vassilevska Williams, Woldeghebriel, Xu 2022]. In this paper, we show a 3FRP algorithm in deterministic $\tilde{O}(n^3)$ time for undirected weighted graphs, which almost matches the size of the output. This implies that f FRP in undirected graphs can be solved in almost optimal $\tilde{O}(n^f)$ time for all $f \geq 3$.

To construct our 3FRP algorithm, we introduce an incremental distance sensitivity oracle (DSO) with $\tilde{O}(n^2)$ worst-case update time, while preprocessing time, space, and query time are still $\tilde{O}(n^3)$, $\tilde{O}(n^2)$ and $\tilde{O}(1)$, respectively, which match the static DSO [Bernstein and Karger 2009]. Here in a DSO, we can preprocess a graph so that the distance between any pair of vertices given any failed edge can be answered efficiently. From the recent result in [Peng and Rubinstein 2023], we can obtain an offline dynamic DSO from the incremental worst-case DSO, which makes the construction of our 3FRP algorithm more convenient. By the offline dynamic DSO, we can also construct a 2-fault single-source replacement path (2-fault SSRP) algorithm in $\tilde{O}(n^3)$ time, that is, from a given vertex s , we want to find the distance to any vertex t when any pair of edges fail. Thus the $\tilde{O}(n^3)$ time complexity for 2-fault SSRP is also almost optimal.

Now we know that in undirected graphs 1FRP can be solved in $\tilde{O}(m)$ time [Nardelli, Proietti, Widmayer 2001], and 2FRP and 3FRP in undirected graphs can be solved in $\tilde{O}(n^3)$ time. In this paper, we also show that a truly subcubic algorithm for 2FRP in undirected graphs does not exist under APSP-hardness conjecture.

1 Introduction

The shortest path problem is one of the most fundamental problems in computer science, and the single-source shortest path problem is known to have an almost linear $\tilde{O}(m)$ time algorithm [14, 15]. In a failure-prone graph $G = (V, E)$ ($n = |V|$, $m = |E|$), some edges may fail on the shortest path, so we want to find a *replacement path*. In the classical replacement path (RP) problem, given source s and destination t , we want

^{*}chisc21@mails.tsinghua.edu.cn

[†]duanran@mail.tsinghua.edu.cn

[‡]benyuw@umich.edu

[§]xtl21@mails.tsinghua.edu.cn

to find the s - t shortest path when edge e fails for every edge $e \in E$. Of course, if e is not on the original s - t shortest path in G , the replacement path is still the original shortest path, so in fact we just need to find an s - t replacement path for every edge e on the s - t shortest path in G , which counts for $O(n)$ replacement paths.

We can generalize the RP problem to any number of failed edges f , that is, finding s - t shortest paths for every failed edge set F which has $|F| \leq f$. This problem is called f -fault replacement path (f FRP) [30], therefore 1FRP is just the original RP problem. As before, given the first $(f - 1)$ failed edges and the corresponding s - t replacement shortest path, we still only need to consider the cases when the f -th failed edge is on it. Because there are $O(n)$ edges on each shortest path, the total number of replacement paths we need to find for f FRP problem is $O(n^f)$. The current well-known results of f FRP algorithms for *real-weighted* directed and undirected graphs are summarized as follows.

- RP problem in directed graphs can be solved in $O(mn + n^2 \log \log n) = O(n^3)$ time [20].
- Under APSP-hardness conjecture¹, RP in directed graphs does not have $O(n^{3-\epsilon})$ time algorithm for any constant $\epsilon > 0$ [31]. So the $O(n^3)$ -time algorithm is (conditionally) nearly optimal.
- RP problem in undirected graphs can be solved in $\tilde{O}(m)$ time² [25], thus also almost optimal.
- Recently, Vassilevska Williams, Woldeghebriel and Xu [30] gave a 2FRP algorithm with $\tilde{O}(n^3)$ running time for directed graphs, almost matching the 1FRP case. (Although not formulated, it also works for undirected graphs with slight modifications.)

Thus, 1FRP and 2FRP problems both have $\tilde{O}(n^3)$ -time algorithms, so it is natural to ask whether 3FRP still has a $\tilde{O}(n^3)$ -time algorithm. In this paper, we give such a 3FRP algorithm for undirected graphs: (Note that all algorithms in this paper are deterministic.)

Theorem 1.1. *The 3FRP problem in undirected real-weighted graphs can be solved in $\tilde{O}(n^3)$ time.*

Denote the shortest path between s and t in graph G by $\pi_G(s, t)$. Then in $\tilde{O}(n^3)$ time, for every edge $d_1 \in \pi_G(s, t)$, we can find all edges of $\pi_{G-\{d_1\}}(s, t)$, then for every edge $d_2 \in \pi_{G-\{d_1\}}(s, t)$, we can find all edges of $\pi_{G-\{d_1, d_2\}}(s, t)$, then for every edge $d_3 \in \pi_{G-\{d_1, d_2\}}(s, t)$, we can find $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$, so the algorithm outputs $O(n^3)$ distances in total. One can see the difficulty of this problem since every answer only takes $\tilde{O}(1)$ time on average.

For any $f \geq 3$, by Theorem 1.1, the f FRP problem can be solved in $\tilde{O}(n^f)$ time in undirected graphs (see the reduction in [30]). Since the size of the output of f FRP can be $\Theta(n^f)$, the running time is almost optimal. As in [30], we also take the 1-failure distance sensitivity oracle as an important subroutine, since it only takes $\tilde{O}(1)$ query time. Here an f -failure distance sensitivity oracle (DSO) is a data structure that supports distance queries between any pair of $u, v \in V$ given any f failed edges. It is widely known that 1-failure DSO takes $\tilde{O}(n^2)$ space, $O(1)$ query time, and $\tilde{O}(mn)$ construction time [11, 5]. Since current 2-failure DSOs are still hard to construct efficiently [17], we instead construct an incremental 1-failure DSO: (In the following DSO means 1-failure DSO for convenience.)

Theorem 1.2. *For a given undirected graph G , there is an incremental DSO that can be constructed in $\tilde{O}(n^3)$ time, so that when we insert an edge e into G , the DSO can be maintained in worst-case $\tilde{O}(n^2)$ time. The DSO also has $\tilde{O}(n^2)$ space and $\tilde{O}(1)$ query time.*

¹APSP is an abbreviation for all-pair shortest path problem, and APSP-hardness conjecture suggests that APSP cannot be solved in truly subcubic $O(n^{3-\epsilon})$ time for any constant $\epsilon > 0$.

²Here $\tilde{O}(\cdot)$ hides polylogarithmic factors.

Recently Peng and Rubinfeld [26] gave a reduction from offline fully dynamic structure to worst-case incremental structure, where “offline” means all updates are known ahead. So we can obtain an offline dynamic efficient DSO. (We also give a simple proof of the reduction we need by a different method as [26], see Theorem 2.5.)

Theorem 1.3. ([26]) *Let $T \geq 1$ be the total number of updates. If there exists an incremental dynamic algorithm with query time Γ_q and worst-case update time Γ_u , then there is an offline dynamic algorithm with query time Γ_q and worst-case update time $O(\Gamma_u \cdot \log^2(T))$.*

Corollary 1.4. *Given an undirected graph G and a sequence of $O(n)$ edge insertions and deletions, there is an offline dynamic DSO which can be constructed in $\tilde{O}(n^3)$ time, and the total update time is also $\tilde{O}(n^3)$. It can answer the distance between any pair of vertices when any edge fails in $\tilde{O}(1)$ time.*

We have known 1FRP is as hard as APSP in directed graphs [31], but in undirected graphs, 1FRP can be solved in $\tilde{O}(m)$ time [25], and 2FRP in undirected graphs can be solved in $\tilde{O}(n^3)$ time. One may wonder whether 2FRP in undirected graphs has a truly subcubic time algorithm. However, we show that it is not possible under the APSP-hardness conjecture.

Theorem 1.5. *Assuming the APSP-hardness conjecture that APSP cannot be solved in truly subcubic $O(n^{3-\epsilon})$ time for any constant $\epsilon > 0$, then 2FRP problem in undirected weighted graphs cannot be solved in truly subcubic time.*

We can also apply the offline dynamic DSO to get a 2-fault single-source replacement path (2-fault SSRP) algorithm for undirected graphs, that is, given a source s , for every other vertex $t \in V$, we can find $\pi_{G-\{d_1, d_2\}}(s, t)$ for all edges d_1, d_2 . The reduction is very simple: we can remove an edge d_1 from the shortest path tree from s each time, and then put it back. By the offline dynamic DSO, we can answer the distance between s and t avoiding d_2 in the current graph without d_1 .

Theorem 1.6. *The 2-fault single-source replacement path problem can be solved in $\tilde{O}(n^3)$ time for undirected graphs.*

Note that Theorem 1.6 can be extended to undirected f -fault SSRP algorithm in $\tilde{O}(n^{f+1})$ time for all $f \geq 2$, which is almost optimal. Previously there are 1-fault single-source replacement path algorithms of running time $\tilde{O}(m\sqrt{n} + n^2)$ for unweighted undirected graphs [8, 12] and unweighted directed graphs [9].

Note that the 2FRP, 3FRP, and single-source 2FRP algorithms in this paper obtain every distance from DSOs, which are essentially similar to the one in [5], or the APSP table. By the properties of the DSOs and the construction of our algorithms, in these algorithms in fact we can obtain an oracle of size $\tilde{O}(n^3)$ in which we can retrieve a shortest path under failures in $O(1)$ time per edge.

Other related work. The current fastest running time for the directed all-pair shortest path (APSP) problem is $n^3/2^{\Omega(\sqrt{\log n})}$ [29]. Finding truly subcubic time algorithms for APSP with arbitrary weights is considered a major open problem in algorithm research, and its hardness is one of the major assumptions in proving conditional lower bounds.

For replacement path (RP) and single-source replacement path (SSRP) problems, there are also many subcubic time results for graphs with small integer edge weights. In unweighted directed graphs, the RP problem can be solved in $\tilde{O}(m\sqrt{n})$ time [2]. If the edge weights are integers in $[-M, M]$, Vassilevska Williams gave a $\tilde{O}(Mn^\omega)$ time RP algorithm [29], where $\omega < 2.371339$ is the exponent of the complexity of matrix multiplication [1, 28, 16]. Moreover, [30] gave a 2FRP algorithm in small edge weight directed

graphs in $\tilde{O}(M^{2/3}n^{2.9153})$ time. For SSRP problem, there is also a $\tilde{O}(Mn^\omega)$ time algorithm for graphs with integer weights in $[1, M]$ and algorithms with running time $\tilde{O}(M^{0.7519}n^{2.5286})$ and $\tilde{O}(M^{0.8043}n^{2.4957})$ for graphs with integer weights in $[-M, M]$. [22, 21, 24].

After the breakthrough result of efficient 1-failure DSO by Demetrescu, Thorup, Chowdhury and Ramachandran [11], there are many efforts to improve the preprocessing time [4, 5, 21, 22, 8, 27, 23], space [19], and extend to more failures. However, keeping query time $\tilde{O}(1)$ and space $\tilde{O}(n^2)$ is difficult when generalized to any constant number of f failures. The 2-failure DSO of $\tilde{O}(n^2)$ space and $\tilde{O}(1)$ query time [17] is much more complicated than the 1-failure case, so it seems impossible to generalize it to f -failure DSO in arbitrary graphs. For undirected graphs, recently Dey and Gupta gave an f -failure DSO with $O(f^4n^2 \log^2(nM))$ space and $O((f \log(nM))^{O(f^2)})$ query time [13], improving the result in [18], but their construction time is still large, thus still not suitable for efficient f FRP algorithms.

Organization. In Section 2 we introduce basic notations, assumptions and concepts. In Section 3 we give a brief description of the 2FRP algorithm for undirected graphs, which is a little simpler than the one for directed graphs in [30], then give an overview on how to extend it to 3FRP algorithm. In Section 4 the incremental DSO is discussed, which is the crucial part of our 3FRP algorithm. The algorithms for the cases that 1 edge, 2 edges, and 3 edges of the set of failed edges is/are on the original shortest path $\pi_G(s, t)$ be given in Section 5, 6, 7, respectively. In Section 8 we prove the hardness of undirected 2FRP under APSP-hardness conjecture, and in Section 9 the 2-fault single-source replacement path algorithm is discussed.

2 Preliminaries

2.1 Notations and Assumptions

We consider an undirected weighted graph $G = (V, E)$ and $n = |V|, m = |E|$. For any two vertices $u, v \in V$, we use $\pi_G(u, v)$ to denote the shortest path from u to v in G . If the graph is clear in the context, then we use uv to be short for $\pi_G(u, v)$. For a path P , we use $|P|, \|P\|$ to denote the length of P and the number of edges in P , respectively.

If the context is clear that a vertex z is on the shortest path uv from u to v in a graph G , we use $z \oplus i$ to represent the vertex that is i vertices after z in the direction from u to v . Similarly, we define $z \ominus i$ to represent the vertex that is i vertices before z . If x and y are two vertices on the shortest path from u to v in graph G , we use the notation $x > y$ to say that the number of edges between u and x is larger than that between u and y , and the vertices u, v should be clear in the context. Similarly $x < y$ means $\|ux\| < \|uy\|$, $x \leq y$ means $\|ux\| \leq \|uy\|$, $x \geq y$ means $\|ux\| \geq \|uy\|$.

Also when $x, y \in \pi_G(u, v)$ we say $\pi_G(x, y)$ is a subpath of $\pi_G(u, v)$. If the context is clear we use the interval $[x, y]$ for $x \leq y$ to denote the subpath xy of uv in graph G . Thus the interval between i -th vertex and j -th vertex after u on uv is denoted as $R = [u \oplus i, u \oplus j]$.

Let P_1, P_2 be two paths. If they share a same endpoint, we use $P_1 \circ P_2$ to denote the concatenation of them. If they do not share any edge, we say P_1 avoids P_2 . Sometimes for brevity, we use $\min\{P_1, P_2\}$ to denote the shorter path of P_1, P_2 , which means if $|P_1| \leq |P_2|$, $\min\{P_1, P_2\} = P_1$ and otherwise $\min\{P_1, P_2\} = P_2$.

In this paper, for a path P in G , $G - P$ denotes the graph obtained by removing **edges** of P from G . (We do not need to remove any vertex in this paper.)

As in many distance oracle papers, we also use the **unique shortest path assumption**, which means the shortest path between any two vertices in any subgraph of G is unique. For the incremental DSO, we also make the unique shortest path assumption at any time. If this is not the case, we can break the ties by adding

small random variables to the edge weights. (See [10].) So for example, we can check whether an edge $e = (x, y)$ is on st or not by checking whether $|sx \circ (x, y) \circ yt| = |st|$ or $|sy \circ (y, x) \circ xt| = |st|$.

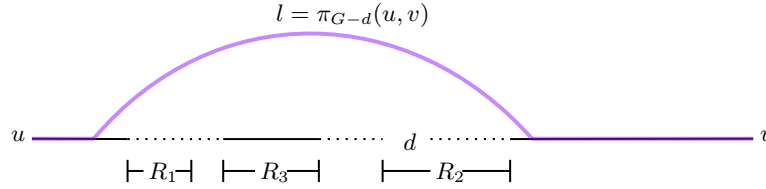
2.2 Replacement Paths

We first consider the divergence and convergence points of two paths: (As before, although our graph is undirected, we may assume a path P starts from a vertex u and ends at v .)

- For two paths P, Q both starting from u , we say they diverge at vertex $a \in P \cap Q$ if Q first gets into an edge out of P from a , that is, the subpaths of P and Q before a coincide, and we denote this divergence point a by $\Delta(P, Q)$. If P, Q are both shortest paths, their divergence point is unique.
- Symmetrically, for P, Q both ends at v , we say they converges at vertex $b \in P \cap Q$ if Q gets into an edge in P from b , and the subpaths of P and Q coincide after b . The convergence point b is denoted by $\nabla(P, Q)$.
- For two shortest paths P, Q which do not share endpoints, if they intersect, the intersection part is also a shortest path, that is, they must first converge and then diverge.

The following theorem is well known for replacement paths.

Theorem 2.1. ([11]) *Let l be a 1-fault replacement path in G , i.e. $l = \pi_{G-d}(u, v)$ for an edge d . Let R_1, R_2 be two subpaths in uv with no intersection, and R_3 be the subpath between them. If l avoids both R_1 and R_2 , then it avoids R_3 .*



From this theorem, we can see for any f -fault replacement path $\pi_{G-F}(u, v)$ where only one failed edge in F is on the original shortest path uv , then $\pi_{G-F}(u, v)$ only diverges and converges once on uv . And if two failed edges d_1, d_2 in F are on uv , they will cut uv into three intervals D_1, D_2, D_3 . Because D_1, D_2, D_3 are all shortest paths in $G - F$, $\pi_{G-F}(u, v)$ maybe:

- diverges at D_1 and converges at D_3
- diverges at D_1 , converges at some point in D_2 , diverges from D_2 , then converges at D_3

In undirected weighted graphs, a notable theorem on the structure of replacement paths says that a 1-fault replacement path $\pi_{G-f}(u, v)$ is a concatenation of a shortest path ux , an edge $e = (x, y)$, and a shortest path yv , for some vertices x and y . (Any part can degenerate to a point.) More generally,

Theorem 2.2. ([7]) *For any undirected weighted graph G and any set of failed edges F with $|F| = f$, an f -fault replacement path can be partitioned into a concatenation interleaving at most $f + 1$ subpaths and f edges, where each subpath is a shortest path in the graph G .*

From this theorem, we have:

Lemma 2.3. ([6]) *The union of all 1-fault u - v replacement paths has $O(n)$ edges in total in undirected graphs.*

Proof. **TOPROVE 0**

□

2.3 DSOs

The paper uses the 1-fault DSO given by Bernstein and Karger ([5]) to initialize the incremental DSO. While [5] gives a 1-fault DSO in directed weighted graphs, it also works for undirected weighted graphs by a simple reduction. Suppose G is an undirected weighted graph, we introduce a directed graph G' by replacing each edge in G by a pair of directed edges with the same weight. Suppose that in graph G edge (x, y) on the shortest path st is removed (x is closer to s), then we remove the directed edge (x, y) on the shortest path from s to t in G' . We can see that the 1-fault replacement path $\pi_{G'-(x,y)}(s, t)$ cannot go through edge (y, x) . This is because that no edge on $\pi_{G'}(s, x)$ is removed, so $\pi_{G'-(x,y)}(s, x) = \pi_{G'}(s, x)$ does not go through (y, x) . Therefore, $\pi_{G-(x,y)}(s, t) = \pi_{G'-(x,y)}(s, t)$.

There is a theorem in [26] to view incremental structures as offline fully dynamic structures.

Theorem 2.4. ([26]) *Let $T \geq 1$ be the total number of updates. Suppose there exists an incremental dynamic algorithm with query time Γ_q and worst-case update time Γ_u , then there is a dynamic algorithm for deletions-look-ahead setting with query time Γ_q and worst-case update time $O(\Gamma_u \cdot \log^2(T))$.*

This theorem can be used to utilize our incremental DSO in Section 4 as an offline fully dynamic DSO, and here we also prove the reduction we need in a self-contained way:

Theorem 2.5. *Starting from a graph of $O(n)$ vertices, if we have an incremental DSO of $\tilde{O}(n^2)$ space with preprocessing time $\tilde{O}(n^3)$, worst case update time $\tilde{O}(n^2)$ and query time $\tilde{O}(1)$, then it can be transformed into an offline fully dynamic DSO, such that if the total number of updates is $T = O(n)$, the total preprocessing and update time is $\tilde{O}(n^3)$ and the query time is still $\tilde{O}(1)$.*

Proof. **TOPROVE 1** □

3 Technical Overview

In this section, we first give a brief description of 1FRP and 2FRP algorithms for undirected graphs, which can list all edges of each path in $\tilde{O}(n^3)$ time, so that we can find the third failed edge in our 3FRP algorithm. Then the high-level ideas for the 3FRP algorithm and incremental DSO are discussed.

3.1 2FRP algorithm between s, t

All 1-fault replacement paths are easy to find in $\tilde{O}(mn)$ time since we can delete each edge in st and then run Dijkstra algorithm [14] from s . Then for edges $d_1 \in st$ and $d_2 \in \pi_{G-d_1}(s, t)$, consider whether d_2 is on st or not.

3.1.1 Only d_1 is on st

By the methods in [30], we create a graph H from $G - st$ by adding new vertices as follows. Let N be a number that is large enough. (Namely, N can be the sum of all edge weights in G .)

- First let H be a copy of $G - st$, that is, remove all edges on $\pi_G(s, t)$ from G .
- For every edge $d = (x, y) \in st$ and x is before y on st , introduce two new vertices d^- and d^+ .
 - For every vertex x' on sx , the node d^- is connected with x' by an edge with weight $|sx'| + N$.
 - For every vertex y' on yt , the node d^+ is connected with y' by an edge with weight $|y't| + N$.

The number of vertices in H is still $O(n)$, so we construct a 1-failure DSO on H with $\tilde{O}(n^3)$ construction time, $\tilde{O}(n^2)$ space and $O(1)$ query time by [5]. We say the edge (d^-, x') corresponds to the path sx' since their length differs by a fixed number N , similarly (y', d^+) corresponds to the path $y't$. (Here N guarantees that any path between new vertices will not travel through other new vertices.) Since d_2 is not on st , we can obtain $\pi_{G-\{d_1, d_2\}}(s, t)$ by querying the DSO. We have:

Lemma 3.1. *Given G and $d_1 \in st$, $d_2 \notin st$, in the graph H we defined,*

$$|\pi_{G-\{d_1, d_2\}}(s, t)| = |\pi_{H-d_2}(d_1^-, d_1^+)| - 2N$$

Proof. **TOPROVE 2** □

Since the 1-failure DSO supports path retrieving [11, 5] in $O(1)$ time per edge, we can restore all paths $\pi_{G-\{d_1, d_2\}}(s, t)$ in $\tilde{O}(n^3)$ time.

3.1.2 Only d_1, d_2 are on st

Our structure in this case is also similar to [30], but is simpler since G is undirected here. If we use the graph H before, the distance $\pi_H(d_1^-, d_2^+)$ cannot capture $\pi_{G-\{d_1, d_2\}}(s, t)$ since it only formulates the subpaths before divergence point and after convergence point on st , but $\pi_{G-\{d_1, d_2\}}(s, t)$ may go through some edges between d_1 and d_2 on st , which are not included in H .

W.l.o.g. assume $d_1 = (x_1, y_1)$ is before $d_2 = (x_2, y_2)$ on st , so $s \leq x_1 < y_1 \leq x_2 < y_2 \leq t$ on the shortest path st . For all pairs of d_1, d_2 , we want to find the paths: (w, a, b, z are all on st .)

$$P(d_1, d_2) = \min_{w \leq x_1, y_1 \leq a \leq b \leq x_2, z \geq y_2} \{sw \circ \pi_{G-st}(w, a) \circ ab \circ \pi_{G-st}(b, z) \circ zt\}$$

and also:

$$P'(d_1, d_2) = \min_{w \leq x_1, y_1 \leq b \leq a \leq x_2, z \geq y_2} \{sw \circ \pi_{G-st}(w, a) \circ ab \circ \pi_{G-st}(b, z) \circ zt\}$$

First we run the APSP algorithm on the graph $G - st$, then find the paths $U(d_1, a) = \min_{w \leq x_1} \{sw \circ \pi_{G-st}(w, a)\}$ for all edge $d_1 = (x_1, y_1)$ and vertex $a \geq y_1$ on st in $O(n^3)$ time, since we need $O(n)$ time to enumerate w for every pair of d_1 and a . Symmetrically also find $U'(d_2, b) = \min_{z \geq y_2} \{\pi_{G-st}(b, z) \circ zt\}$ for all edge $d_2 = (x_2, y_2)$ and vertex $b \leq x_2$ on st in $O(n^3)$ time. Then for every pair of $d_1 = (x_1, y_1), d_2 = (x_2, y_2)$ and $y_1 \leq b \leq x_2$, we can find:

$$W(d_1, d_2, b) = \min_{w \leq x_1, b \leq a \leq x_2} \{sw \circ \pi_{G-st}(w, a) \circ ab\}$$

That is, the path diverges before d_1 , converges with st at some point a between b and x_2 , then goes through subpath of st to b . If $b = x_2$, it is just $U(d_1, x_2)$. Then we can compute $W(d_1, d_2, b)$ for b from x_2 backward to y_1 . If (b, b') with $b < b'$ is an edge on st , the path $W(d_1, d_2, b)$ can either first go through $W(d_1, d_2, b')$ or directly go to b , so $W(d_1, d_2, b) = \min\{U(d_1, b), W(d_1, d_2, b') \circ (b, b')\}$. So the total time to get all of $W(d_1, d_2, b)$ is still $O(n^3)$.

Thus, given d_1, d_2 , we can get $P'(d_1, d_2)$ from $W(d_1, d_2, b) \circ U'(d_2, b)$ by enumerating all possible b between d_1, d_2 in $O(n)$ time. $P(d_1, d_2)$ can be obtained similarly. So the total time is $O(n^3)$.

3.2 Ideas of 3FRP algorithm

As the 2FRP algorithm in Section 3.1, we also consider how many of d_1, d_2, d_3 are on the original shortest path st , and use different methods to solve them.

Only d_1 is on st . (Although there is 2-failure DSO of $\tilde{O}(n^2)$ space and $\tilde{O}(1)$ query time [17], it is hard to utilize it here since its construction time is too large. Here we build an incremental 1-failure DSO instead.) To solve the case that only d_1 is on st but d_2, d_3 are not, note that if $d_2 \in \pi_{G-d_1}(s, t)$, from Lemma 2.3, the number of possible “second failure” d_2 is $O(n)$. If we have a dynamic 1-failure DSO with $\tilde{O}(n^2)$ update time when updating an edge, then for every possible d_2 we can delete it from H temporarily and query $|\pi_{(H-d_2)-d_3}(d_1^-, d_1^+)|$ for every d_1 and d_3 , which equals $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)| + 2N$. After that add d_2 back to H . Although fully dynamic DSO may be difficult, in our algorithm we only need an offline dynamic version, and by the reduction from offline dynamic structure to worst-case incremental structure in [26] and Theorem 2.5, we construct an incremental DSO with worst-case $\tilde{O}(n^2)$ update time instead.

Only d_1, d_2 are on st . Even with the incremental DSO, it is not very easy to solve the case that d_1, d_2 are on st but d_3 is not. So we apply the incremental DSO on the binary partition structure as in the 2-failure DSO of [17], that is, we build a binary range tree on the path st , and define two graphs $H_{i,0}$ and $H_{i,1}$ (like the graph H in Section 3.1.1) on each level i of the range tree, where in $H_{i,0}$ we remove all odd ranges on level i and in $H_{i,1}$ we remove all even ranges. Given d_1 and d_2 , we find the first level on which d_1 and d_2 are in different ranges, then d_1 is in an even range and d_2 is in an adjacent odd range, and let m be the point separating these two ranges. Consider the following cases on the intersection of $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ and the edges between d_1 and d_2 on st .

- If $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ does not travel through edges between d_1 and d_2 , we can use DSO on H to give the answer.
- If $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ only goes through edges between d_1 and m , we can use $H_{i,0}$ which does not contain edges between m and d_2 . So the distance between d_1^- and d_2^+ in $H_{i,0} - d_1$ avoiding d_3 can give the answer. (Note that edges between s and d_1 in $H_{i,0}$ will not affect the answer, similarly for edges between d_2 and t .)
- If $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ only goes through edges between m and d_2 , symmetrically we can use $H_{i,1} - d_2$.
- The only case left is that $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ travels through m , we can also use the offline dynamic DSO on $H_{i,0} - d_1$ and $H_{i,1} - d_2$ to answer $|\pi_{G-\{d_1, d_2, d_3\}}(s, m)| + |\pi_{G-\{d_1, d_2, d_3\}}(m, t)|$.

So we need to build offline dynamic DSOs on $H_{i,0}$ and $H_{i,1}$ in each level i . Since there are $O(\log n)$ levels, the total time is still $\tilde{O}(n^3)$.

All of d_1, d_2, d_3 are on st . We assume the order of them on st is just d_1, d_2, d_3 , then they will cut st into four ranges D_1, D_2, D_3, D_4 in order. The difficulty will come from the fact that the path $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ can go through both of D_2 and D_3 , in any order. To solve this case, we first build a structure in $\tilde{O}(n^3)$ time which can answer the following query in $\tilde{O}(1)$ time:

- Given two vertex-disjoint ranges R_1, R_2 of consecutive edges on the original shortest path st (R_1, R_2 can be single vertices), answer the shortest path that starts from the leftmost (rightmost) vertex of R_1 , diverges from R_1 , converges in R_2 , and finally ends at the leftmost (rightmost) vertex of R_2 .
- Given an edge d_1 on st and two vertex-disjoint ranges R_1, R_2 of consecutive edges on st after d_1 , answer the shortest path that starts from s , diverges before d_1 , converges in R_1 , and diverges from R_1 , then converges in R_2 , and finally ends at the leftmost (rightmost) point of R_2 .

Note that in the structure we store the paths for all ranges R_1, R_2 in the binary range tree, then given any R_1, R_2 in the query, they can be split into $O(\log n)$ ranges in the binary range tree.

Similar to Section 3.1.2, by this structure, when given d_1, d_2, d_3 and a vertex b on st , it takes $\tilde{O}(1)$ time to find the shortest path goes through b avoiding d_1, d_2, d_3 . So it will take $\tilde{O}(n)$ time to find the shortest path goes through both of D_2 and D_3 avoiding d_1, d_2, d_3 , but we need the time bound to be $\tilde{O}(1)$ on average. For every pair of d_1 and d_3 , the main ideas are as follows.

- First find the middle edge e_1 on the range between d_1 and d_3 , and use $\tilde{O}(n)$ time to find the shortest path $\pi_{G-\{d_1, d_3, e_1\}}(s, t)$.
- For other edges d_2 between d_1 and d_3 , if d_2 is not on $\pi_{G-\{d_1, d_3, e_1\}}(s, t)$, then either $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ is equal to $\pi_{G-\{d_1, d_3, e_1\}}(s, t)$ or it goes through e_1 , so this case can be solved in $\tilde{O}(1)$ time.
- The intersection of $\pi_{G-\{d_1, d_3, e_1\}}(s, t)$ and st between d_1 and d_3 consists of at most two ranges, then find the middle edge e_2 of the larger range, and find the shortest path avoiding d_1, d_3, e_1, e_2 and goes through two ranges between them. This can also be done in $\tilde{O}(n)$ time.
- As before, for d_2 not on the new path we can solve it by querying the path goes through e_1 and e_2 , so next we only need to consider the edges in the intersection of all paths so far, and we can argue that the size of the intersection between d_1 and d_3 will shrink by a constant factor in every two iterations, so only $O(\log n)$ iterations are needed. Thus the time for every pair of d_1 and d_3 is only $\tilde{O}(n)$.

3.3 Ideas of incremental DSO

To maintain APSP incrementally, when inserting an edge $e = (x, y)$, for every pair of vertices u, v we just need to check whether e can make their distance shorter, that is, the new distance will be $\min\{|uv|, |ux| + |e| + |yv|, |uy| + |e| + |xv|\}$, thus update time is $O(n^2)$. Similar to previous works [11], we want to make a DSO structure that maintains the following for all pairs of vertices u, v .

- Shortest path tree from every vertex.
- $\pi_{G-(u \oplus i)(v \oplus j)}(u, v)$ for all $i, j \in \mathbb{N}$ that are 0 or integer powers of 2 such that $i + j < \|uv\|$.

(Remind that $\|uv\|$ stands for the number of edges in uv .) Note that it is a little different from previous DSOs [11, 5] since we want to query $\pi_{G-ab}(u, v)$ for any subpath ab in uv , which is needed in computing new paths after inserting an edge e into G .

As in [5], we can also use $\max_{f \in (u \oplus i)(v \oplus j)} \pi_{G-f}(u, v)$ to replace $\pi_{G-(u \oplus i)(v \oplus j)}(u, v)$. When inserting new edges into the graph it is very difficult to maintain the path avoiding the whole range or the maximum shortest detour of the range. However, in $\max_{f \in (u \oplus i)(v \oplus j)} \pi_{G-f}(u, v)$, we only need the path $\pi_{G-f}(u, v)$ for some f which avoids the whole range between $(u \oplus i)$ and $(v \oplus j)$. When such an f does not exist, that is, even the maximum one still intersects with the range between $(u \oplus i)$ and $(v \oplus j)$, we do not need to care about it. This makes the incremental DSO possible.

4 Incremental DSO

In this section, we give the construction of the incremental distance sensitive oracle. That is, given an undirected weighted graph G , we can construct an oracle of space $\tilde{O}(n^2)$ in $\tilde{O}(n^3)$ preprocessing time, then for any pair of vertices u, v , and an edge e , we can query the shortest $u-v$ path in G that avoids e in time $\tilde{O}(1)$.

Further, the oracle supports incremental update: when a weighted edge is added to the graph, we can maintain the oracle in worst-case update time $\tilde{O}(n^2)$. By Theorem 2.5, it can be transformed into an offline fully dynamic DSO, such that if the total number of updates is $O(n)$, the total preprocessing and update time is $\tilde{O}(n^3)$ and the query time is still $\tilde{O}(1)$.

4.1 Notations

Below we say a subpath $ab \subseteq uv$ is **weak** under uv if there exists an edge $f \in ab$, $\pi_{G-f}(u, v) = \pi_{G-ab}(u, v)$, and f is called a **weak point** of ab (under uv). (Recall that $G - ab$ means removing edges of ab from G .) In other words, if ab is weak under uv , the shortest $u - v$ path avoiding the weak point also avoids the whole subpath ab . By definition, on uv all weak points of ab correspond to the same replacement path $\pi_{G-ab}(u, v)$.

Definition 4.1. If a path P between u and v in a graph G is represented by the concatenation of a shortest path ux , an edge (x, y) and a shortest path yv in G , then we say P is in the **proper form** in G . Here the edge (x, y) and shortest path yv can be empty.

By [7] and Theorem 2.2 in this paper, when ab is weak under uv , $\pi_{G-ab}(u, v)$ can be represented in the proper form $ux \circ (x, y) \circ yv$. We further define $\varpi_{G-ab}(u, v)$ to be a null path, or of the proper form, such that,

- If ab is weak, $\varpi_{G-ab}(u, v) = \pi_{G-ab}(u, v)$
- If ab is not weak, $\varpi_{G-ab}(u, v)$ is any path of the proper form $ux \circ (x, y) \circ yv$ avoiding ab , or is a null path with weight $+\infty$.

Note that in this structure we generally can hardly say whether a given ab is weak or not under uv , even if we know the value of $\varpi_{G-ab}(u, v)$. We also point out that even if ab is weak under uv we do not record the position of the weak point.

In our incremental structure, the graph before introducing edge e is called G , and the graph $G \cup \{e\}$ is called G' . For example, $\pi_{G'}(u, v)$ is the shortest $u - v$ path in G' , which is possibly different from uv , the shortest $u - v$ path in G , as it may go through e .

We sometimes need to check whether a path is in proper form, and if so, transform it to its proper form representation. From Lemma 4.3 and 4.4, by constructing a least common ancestor (LCA) structure with $O(n)$ preprocessing time and $O(1)$ query time [3] on the shortest path tree from every vertex, in our incremental DSO structure we can transform every path to its proper form if possible in $\tilde{O}(1)$ time.

Definition 4.2. Let P be a path from u to v in G and let R be a subpath of uv in G . We define a transform $T_{G,R}(P)$ on P to be

- the path P of the proper form, if P can be transformed to the proper form and it avoids edges of R .
- a null path with weight $+\infty$, otherwise.

When this transform is used in this section, P is given by a concatenation of several shortest paths and edges in G , and we want to transform it into the proper form in G or the new graph G' . (Then R is also a shortest path in G' .) By Lemma 4.3 and 4.4, such transform $T_{G,R}$ can be done in $\tilde{O}(1)$ time.

Lemma 4.3. Given a path P explicitly or implicitly, suppose the list of vertices on P is v_0, v_1, \dots, v_r , such that when given an index i , we can find v_i and the length of the subpath of P from v_0 to v_i in $\tilde{O}(1)$ time. Then we can transform P into the proper form if it can, or otherwise point out that it cannot, in $\tilde{O}(1)$ time.

Proof. **TOPROVE 3** □

Lemma 4.4. Let P be a path in the proper form $ux \circ e \circ yv$ in G , and let R be a subpath in the shortest path between u and v in G , then we can check whether P and R share any edge in $O(1)$ time.

Proof. **TOPROVE 4** □

4.2 Preprocessing

In our structure for all pairs of vertices u, v , we maintain

- Shortest path tree from every vertex, and the LCA structure [3] on them.
- $\varpi_{G-(u \oplus i)(v \ominus j)}(u, v)$ for all $i, j \in \{0\} \cup \{2^k\}_{k \in \mathbb{N}}$ such that $i + j < \|uv\|$.

(Recall that $\|uv\|$ stands for the number of edges in uv .) Note that all the detours in this structure are maintained in the proper form. We can see the space of this structure is $\tilde{O}(n^2)$. Although our structure is a little different from the DSO in [5], we can use their DSO to construct our structure:

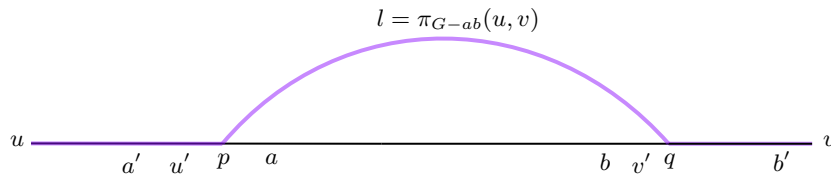
Lemma 4.5. We can construct this initial structure for any graph G in $\tilde{O}(n^3)$ time.

Proof. **TOPROVE 5** □

Using our structure we can answer the shortest path between u, v avoiding any subpath ab of uv if ab is weak under uv .

Theorem 4.6. Let ab be a subpath of uv . By the information we maintain in the DSO, we can calculate the value of $\varpi_{G-ab}(u, v)$ in $\tilde{O}(1)$ time. W.l.o.g., suppose a is closer to u than b is. Let i be the largest integer power of 2 s.t. $u \oplus i$ falls in ua . Similarly j is the largest integer power of 2 s.t. $v \ominus j$ falls in bv . Let $a' = a \ominus i, u' = u \oplus i, v' = v \ominus j, b' = b \oplus j$. (Note that if $a = u$, $a' = u' = u$ and if $b = v$, $b' = v' = v$.) Then

$$\varpi_{G-ab}(u, v) = \min \left\{ T_{G,ab}(ua' \circ \varpi_{G-ab}(a', b') \circ b'v), \varpi_{G-u'v'}(u, v), T_{G,ab}(ua' \circ \varpi_{G-av'}(a', v)), T_{G,ab}(\varpi_{G-u'b}(u, b') \circ b'v) \right\}. \quad (1)$$



Proof. **TOPROVE 6** □

We remark that when (a, b) is an edge, the path ab is always weak by definition, so we can get $\pi_{G-ab}(u, v)$ by Theorem 4.6. Therefore, we can answer any DSO query $\pi_{G-f}(u, v)$ in $\tilde{O}(1)$ time.

4.3 Incremental Update

Consider the current DSO structure for G and an edge $e = (x, y)$ to be added. Recall that we denote $G' = G \cup \{e\}$, so we want to construct the DSO structure for G' using current structure.

- For every pair of vertices u, v , we first maintain the new information of their shortest path as:

$$\pi_{G'}(u, v) = \min\{uv, ux \circ e \circ yv, uy \circ e \circ xv\}.$$

Since if the new shortest path does not go through e , it will be the original uv in G . Otherwise, there are two cases that it goes through e from x to y or from y to x , which correspond to the other two cases. We also construct the shortest path tree from every vertex u and the LCA structure on it. [3]

- For the detours on intervals $\varpi_{G-(u \oplus i)(v \ominus j)}(u, v)$, where i, j are zero or integer powers of 2, we consider two cases separately as below: either $\pi_{G'}(u, v) \neq uv$ (see 4.3.1), or $\pi_{G'}(u, v) = uv$ (see 4.3.2).

In the remaining part of this section, when showing the correctness of the incremental algorithm, by default we assume that $R = \pi_{G'}(u \oplus i, v \ominus j)$ is weak under $\pi_{G'}(u, v)$. This is because we only care about the value $\pi_{G'-R}(u, v)$ when R is weak. If R is not weak, because of the transform of $T_{G',R}$, we can always make sure that the path given by the algorithm is a proper form for the ϖ function (or a null path).

4.3.1 $\pi_G(u, v) \neq \pi_{G'}(u, v)$

We know in this case $e = (x, y)$ is on $\pi_{G'}(u, v)$. W.l.o.g., suppose that x is nearer to u than y . We can see that uv is a shortest $u-v$ path in $G = G' - e$, i.e. $uv = \pi_{G'-e}(u, v)$. Let $p = \Delta(uv, \pi_{G'}(u, v))$ be the divergence point of uv and $\pi_{G'}(u, v)$, and $q = \nabla(uv, \pi_{G'}(u, v))$ be the convergence point. Let $a = u \oplus i, b = v \ominus j$, then $R = \pi_{G'}(a, b)$ and we want to find $\varpi_{G'-R}(u, v)$. So the points a, b, x, y and p, q are all on the new shortest path $\pi_{G'}(u, v)$.

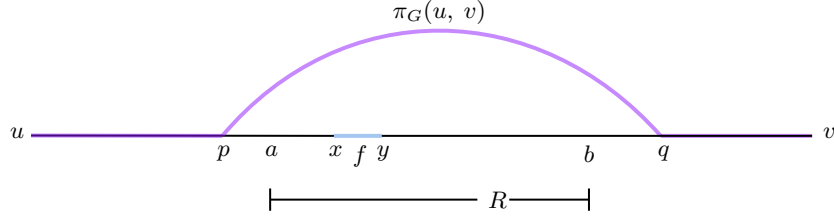
There are six possible relative positions of R, p, q, e :

- **CASE 1:** $p, q \notin R, e \in R$
- **CASE 2:** $p, q \notin R, e \notin R, pq \cap R = \emptyset$
- **CASE 3:** $p, q \notin R, e \notin R, pq \cap R \neq \emptyset$
- **CASE 4:** One of p, q is in $R, e \in R$
- **CASE 5:** One of p, q is in $R, e \notin R$
- **CASE 6:** $p, q \in R, e \in R$

Note that it is impossible to have both $p, q \in R$ but $e \notin R$. This is because $e \in \pi_{G'}(u, v)$ but $e \notin \pi_G(u, v)$, meaning that $e \in \pi_{G'}(p, q)$. Therefore, the relative positions of p, q, x, y, a, b fall into one of the following six cases or their symmetric cases. (Recall that we only consider the case that R is weak under $\pi_{G'}(u, v)$.)

CASE 1: $p \leq a \leq x < y \leq b \leq q$,

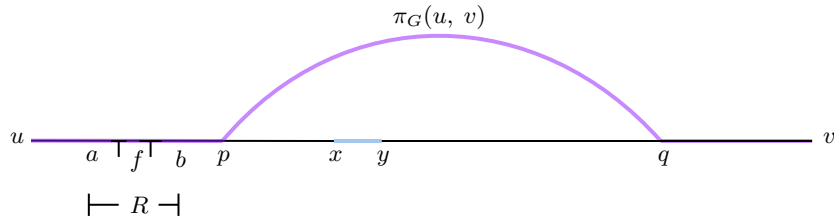
$$\varpi_{G'-R}(u, v) = T_{G',R}(uv).$$



In this case, since $uv = \pi_{G'-e}(u, v)$, $e \in R$, $R = \pi_{G'}(a, b)$, and uv avoids R , so R is weak with a weak point $f = e$, and $\pi_{G'-e}(u, v) = uv$, which must pass the transform $T_{G', R}$.

CASE 2: $a \leq b \leq p \leq x < y \leq q$,

$$\varpi_{G'-R}(u, v) = \min\{T_{G', R}(\varpi_{G-R}(u, v)), T_{G', R}(\varpi_{G-R}(u, x) \circ e \circ yv)\}.$$



- If $\pi_{G'-R}(u, v)$ does not go through e ,

similar as **CASE 1**, it is in $G' - e$, which equals G . Since R is weak under $\pi_{G'}(u, v)$, it is also weak under uv , so $\pi_{G'-R}(u, v) = \varpi_{G-R}(u, v)$.

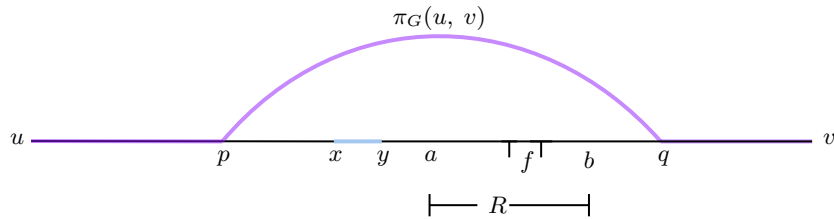
- If $\pi_{G'-R}(u, v)$ goes through e ,

by Theorem 2.2 and 2.1, it takes a shortest $u - x$ path that avoids R , then goes along $\pi_{G'}(x, v)$, i.e. it equals $\pi_{G-R}(u, x) \circ e \circ yv$. Suppose $f \in ab$ is a weak point of R under $\pi_{G'}(u, v)$, $\pi_{G-f}(u, x) \circ e \circ yv = \pi_{G'-f}(u, v) = \pi_{G'-R}(u, v) = \pi_{G-R}(u, x) \circ e \circ yv$, so R is also weak under ux with a weak point f , so $\varpi_{G-R}(u, x) = \pi_{G-R}(u, x)$.

This means if $\varpi_{G-R}(u, x) \circ e \circ yv$ is not a proper form, $\pi_{G'-R}(u, v)$ does not go through e , so $\varpi_{G'-R}(u, v) = uv$. Otherwise we should choose the minimum of both. Note that if $\varpi_{G-R}(u, x)$ intersects with yv , it cannot be the shortest one and also cannot pass the transform $T_{G', R}$.

CASE 3: $p \leq x < y \leq a \leq b \leq q$,

$$\varpi_{G'-R}(u, v) = \min\{T_{G', R}(uv), T_{G', R}(ux \circ e \circ \varpi_{G-R}(y, v))\}.$$



The discussion is very similar to **CASE 2**. If $\pi_{G'-R}(u, v)$ does not go through e , then $\pi_{G'-R}(u, v)$ is in $G' - e$, which equals G , so we have $\pi_{G'-R}(u, v) = uv$. If $\pi_{G'-R}(u, v)$ goes through e , it is the same as **CASE 2**, so $\pi_{G'-R}(u, v) = ux \circ e \circ \varpi_{G-R}(y, v)$.

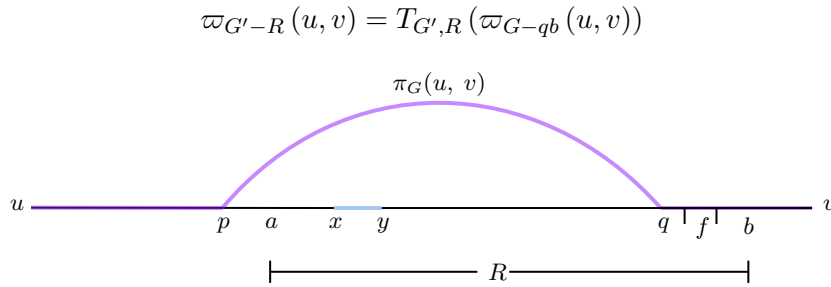
Now we introduce Lemma 4.7 to be used in the following cases.

Lemma 4.7. *Under the settings of Section 4.3.1. suppose that $e \in R = \pi_{G'}(a, b)$, and R is weak under $\pi_{G'}(u, v)$ with weak point f . Assuming that $uv \cap R \neq \emptyset$, then:*

- (1) $f \in uv$,
- (2) $\pi_{G-f}(u, v) = \pi_{G'-f}(u, v) = \pi_{G'-R}(u, v)$.

Proof. **TOPROVE 7** □

CASE 4: $p \leq a \leq x < y \leq q \leq b$,

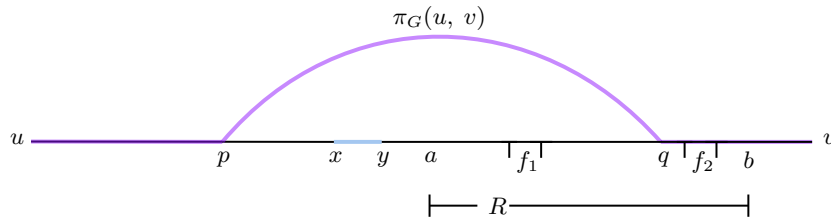


By Lemma 4.7, if R is weak under $\pi_{G'}(u, v)$, the weak point f is in qb , and $\pi_{G-f}(u, v) = \pi_{G'-f}(u, v) = \pi_{G'-R}(u, v)$. By the unique shortest path assumption, this means $\pi_{G-f}(u, v)$ also avoids R , so it also avoids qb thus qb is weak under uv and f is a weak point of it. Now $\pi_{G'-R}(u, v) = \pi_{G-f}(u, v) = \varpi_{G-qb}(u, v)$.

Also, if $\varpi_{G-qb}(u, v)$ does not avoid R , R is not weak under $\pi_{G'}(u, v)$, so any path passing the transform $T_{G',R}$ or the null path is legal for $\varpi_{G'-R}(u, v)$. (Note that it is possible that $\varpi_{G-qb}(u, v)$ intersects with ax in R .)

CASE 5: $p \leq x < y \leq a \leq q \leq b$,

$$\varpi_{G'-R}(u, v) = \min\{T_{G',R}(\varpi_{G-qb}(u, v)), T_{G',R}(ux \circ e \circ \varpi_{G-ab}(y, v))\}.$$



Assume R is weak and has a weak point f , so $\pi_{G'-f}(u, v) = \pi_{G'-R}(u, v)$.

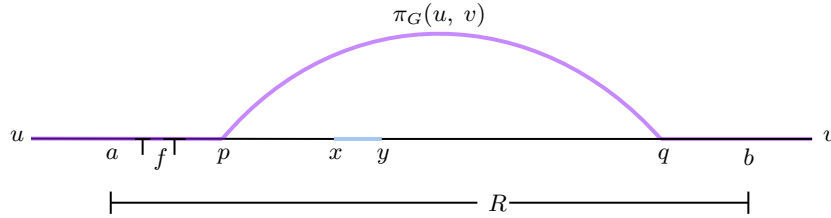
- If $\pi_{G'-f}(u, v)$ goes through e , similar as the discussion in **CASE 2** we know it equals $ux \circ e \circ \pi_{G-f}(y, v)$. Since f is a weak point of R , this path avoids R , so $\pi_{G-f}(y, v) = \pi_{G-ab}(y, v)$, thus ab is weak under yv with weak point f . So $\pi_{G'-f}(u, v) = ux \circ e \circ \varpi_{G-ab}(y, v)$.

- If $\pi_{G'-f}(u, v)$ does not go through e , so $\pi_{G'-f}(u, v) = \pi_{G-f}(u, v)$ and they avoids R .
 - If $f \in aq$, then f is not in uv , $\pi_{G-f}(u, v) = uv$ which intersects with R , making a contradiction.
 - Thus $f \in qb$. Since $\pi_{G-f}(u, v)$ avoids R , it avoids qb , so $\pi_{G-qb}(u, v) = \pi_{G-f}(u, v)$, and qb is weak under uv with weak point f . So in this case $\varpi_{G'-R}(u, v) = \varpi_{G-qb}(u, v)$.

Thus, if R is weak, the minimum one of $\varpi_{G-qb}(u, v)$ and $ux \circ e \circ \varpi_{G-ab}(y, v)$ which pass the transform $T_{G',R}$ must be $\varpi_{G'-R}(u, v)$. If R is not weak, any path passing the transform or the null path is legal.

CASE 6: $a \leq p \leq x < y \leq q \leq b$,

$$\varpi_{G'-R}(u, v) = T_{G',R}(\varpi_{G-ab}(u, v)).$$



As we have assumed, R is weak. Suppose $f \in R$ is a weak point of R . By Lemma 4.7, $f \in ap \cup qb$, and $\pi_{G-f}(u, v) = \pi_{G'-f}(u, v) = \pi_{G'-R}(u, v)$. By the unique shortest path assumption, $\pi_{G-f}(u, v)$ also avoids R , so it avoids ap and qb . If $\pi_{G-f}(u, v)$ intersects $ab = \pi_G(a, b)$, it can only intersects $pq = \pi_G(p, q)$. By Theorem 2.1, it is impossible that $\pi_{G-f}(u, v)$ intersects pq but avoids ap and qb , so $\pi_{G-f}(u, v)$ avoids ab , and ab is weak under uv with weak point f , thus $\varpi_{G'-R}(u, v) = \pi_{G-f}(u, v) = \varpi_{G-ab}(u, v)$.

Also, if $\varpi_{G-ab}(u, v)$ does not pass the transform $T_{G',R}$, then R is not weak under $\pi_{G'}(u, v)$, so any path passing the transform $T_{G',R}$ or the null path is legal for $\varpi_{G'-R}(u, v)$.

4.3.2 $\pi_G(u, v) = \pi_{G'}(u, v)$

As before, let $a = u \oplus i$, $b = v \ominus j$, $G' = G \cup \{e\}$, $R = \pi_{G'}(a, b)$, and we want to find $\varpi_{G'-R}(u, v)$. Also suppose that R is weak under $\pi_{G'}(u, v)$ in graph G' , and f is a weak point of it. The algorithm is as follows.

- Set $\varpi_{G'-R}(u, v) \leftarrow T_{G',R}(\varpi_{G-R}(u, v))$
- Execute the following algorithm, then swap x, y and repeat again
- Let $p = \Delta(ux, uv)$ be the last point that $\pi_G(u, x)$ and $\pi_G(u, v)$ shares, and similarly $q = \nabla(yv, uv)$ be the first point that $\pi_G(y, v)$ and $\pi_G(u, v)$ shares, consider the following cases
 - If $p \leq a < b \leq q$, set $\varpi_{G'-R}(u, v) \leftarrow \min\{\varpi_{G'-R}(u, v), T_{G',R}(ux \circ e \circ yv)\}$
 - If $a < p \leq b \leq q$, set $\varpi_{G'-R}(u, v) \leftarrow \min\{\varpi_{G'-R}(u, v), T_{G',R}(\varpi_{G-ap}(u, x) \circ e \circ yv)\}$
 - If $p \leq a \leq q < b$, set $\varpi_{G'-R}(u, v) \leftarrow \min\{\varpi_{G'-R}(u, v), T_{G',R}(ux \circ e \circ \varpi_{G-qb}(y, v))\}$
 - If $a < p \leq q < b$, do nothing
 - If $a < b \leq p < q$, set $\varpi_{G'-R}(u, v) \leftarrow \min\{\varpi_{G'-R}(u, v), T_{G',R}(\varpi_{G-R}(u, x) \circ e \circ yv)\}$
 - if $p < q \leq a < b$, set $\varpi_{G'-R}(u, v) \leftarrow \min\{\varpi_{G'-R}(u, v), T_{G',R}(ux \circ e \circ \varpi_{G-R}(y, v))\}$

We first prove some lemmas we need.

Lemma 4.8. For any edge f , if $\pi_{G'-f}(u, v)$ does not go through $e = (x, y)$, then $\pi_{G'-f}(u, v) = \pi_{G-f}(u, v)$.

Proof. **TOPROVE 8** □

Lemma 4.9. For any edge $f \in pq$, if $\pi_{G'-f}(u, v)$ goes through $e = (x, y)$ and u is closer to x than to y in G' , then $\pi_{G'-f}(u, v) = ux \circ e \circ yv$.

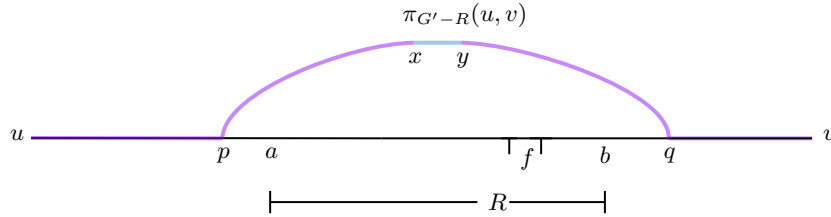
Proof. **TOPROVE 9** □

Lemma 4.10. If R is weak under $\pi_{G'}(u, v)$ in G' and $\pi_{G'-R}(u, v)$ does not go through e , then $\varpi_{G'-R}(u, v) = \varpi_{G-R}(u, v)$.

Proof. **TOPROVE 10** □

In Lemma 4.10 we can see if $\varpi_{G-R}(u, v)$ does not pass the transform $T_{G',R}$, R is not weak or $\varpi_{G'-R}(u, v)$ must pass e . So in the algorithm we first set $\varpi_{G'-R}(u, v)$ to be $T_{G',R}(\varpi_{G-R}(u, v))$. Then we consider the cases that $\varpi_{G'-R}(u, v)$ goes through $e = (x, y)$ (u is closer to x than to y):

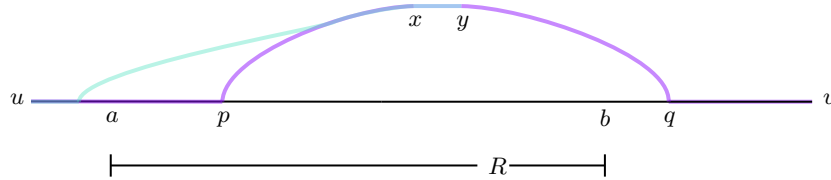
CASE 1: $p \leq a < b \leq q$,



By Lemma 4.9, we set $\varpi_{G'-R}(u, v) = T_{G',R}(ux \circ e \circ yv)$.

(Note that when $\varpi_{G'-R}(u, v) = ux \circ e \circ yv$, we do not know whether R is weak, and even if it is weak we cannot find the weak point. We only know that if R is weak then it is correct. That is the main reason why ϖ is defined in this way.)

CASE 2: $a < p \leq b \leq q$,



If $\pi_{G'-R}(u, v)$ goes through e , from the assumption that R is weak, if its weak point $f \in pb$, by Lemma 4.9, $\pi_{G'-f}(u, v) = ux \circ e \circ yv$, so it intersects with R , thus the weak point f of R must be in ap . Then $\pi_{G'-f}(u, v) = \pi_{G'-R}(u, v)$ which goes through e , so $\pi_{G'-f}(u, v) = \pi_{G'-f}(u, x) \circ e \circ \pi_{G'-f}(y, v) = \pi_{G-f}(u, x) \circ e \circ yv$. By the unique shortest path assumption, this path also does not go through R , thus $\pi_{G-f}(u, x)$ does not intersect with ap , so ap is weak under ux with weak point f . Thus, we get:

$$\varpi_{G'-R}(u, v) = T_{G',R}(\varpi_{G-ap}(u, x) \circ e \circ yv)$$

Thus if $\varpi_{G-ap}(u, x) \circ e \circ yv$ does not pass the transform $T_{G',R}$, either $\pi_{G'-R}(u, v)$ does not go through e , or R is not weak in $\pi_{G'}(u, v)$.

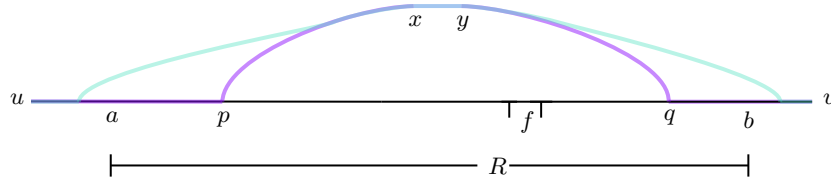
CASE 3: $p \leq a \leq q < b$,

This is symmetric to **CASE 2**.

$$\varpi_{G'-R}(u, v) = T_{G',R}(ux \circ e \circ \varpi_{G-qb}(y, v)).$$

CASE 4: $a < p \leq q < b$,

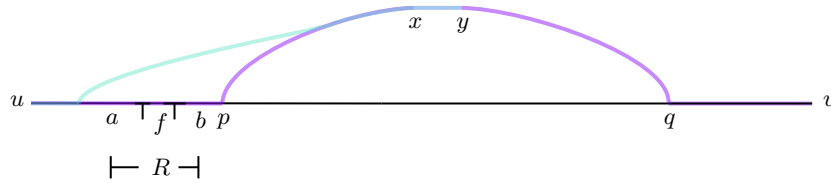
We can prove that in this case if R is weak then $\pi_{G'-R}(u, v)$ cannot go through e .



Prove by contradiction. Suppose R is weak under $\pi_{G'}(u, v)$ with weak point f and $\pi_{G'-R}(u, v)$ goes through e (first x then y), then $\pi_{G'}(u, x) = ux$ and $\pi_{G'}(y, v) = yv$ since they cannot go through e . We know $\pi_{G'-f}(u, v)$ is a 1-replacement-path in G' , so by Theorem 2.2, it equals a concatenation of a shortest path, an edge, and a shortest path in G' . However, $a < p$ means that $\pi_{G'-R}(u, x)$ is different from ux , so it is a concatenation of at least two shortest paths in G' . Same for $\pi_{G'-R}(y, v)$. Therefore, $\pi_{G'-R}(u, v)$ equals the shortest $u - x$ path avoiding R plus e plus the shortest $y - v$ path avoiding R , which will be a concatenation of at least three shortest paths in G , contradicting to R is weak.

CASE 5: $a < b \leq p < q$,

$$\varpi_{G'-R}(u, v) = T_{G',R}(\varpi_{G-R}(u, x) \circ e \circ yv).$$



Suppose that R is weak under uv with a weak point f . By assumption, $\pi_{G'-f}(u, v)$ goes through e , so it equals $\pi_{G'-f}(u, x) \circ e \circ \pi_{G'-f}(y, v)$. By Lemma 4.8, $\pi_{G'-f}(y, v) = \pi_{G-f}(y, v) = yv$ and $\pi_{G'-f}(u, x) = \pi_{G-f}(u, x)$. We can see that R is also weak under ux with weak point f in G' , because $\pi_{G'-f}(u, x)$ is a part of $\pi_{G'-f}(u, v)$, which avoids R . Therefore, by Lemma 4.10, $\pi_{G'-f}(u, x) = \pi_{G'-R}(u, x) = \varpi_{G-R}(u, x)$.

CASE 6: $p < q \leq a < b$,

This is symmetric to **CASE 5**.

$$\varpi_{G'-R}(u, v) = T_{G',R}(ux \circ e \circ \varpi_{G-R}(y, v)).$$

5 One Failed Edge on Original Shortest Path

With the incremental DSO, we can proceed to solve the 3FRP problem for given vertices s and t . Here, by Theorem 2.5, we transform our incremental DSO in Section 4 to an offline fully dynamic DSO. In this section, we suppose only one failed edge, named d_1 , is on the original shortest path st . If d_2, d_3 are both not on $\pi_{G-d_1}(s, t)$, then the replacement path will be $\pi_{G-d_1}(s, t)$. Therefore, we suppose that d_2 is on $\pi_{G-d_1}(s, t)$, and we can restore all edges $d_3 \in \pi_{G-\{d_1, d_2\}}(s, t)$ based on Section 3.1.1.

Theorem 5.1. *There exists an algorithm that, given G with two vertices $s, t \in V(G)$, for all edges $d_1 \in st$, $d_2, d_3 \notin st$, $d_2 \in \pi_{G-d_1}(s, t)$ and $d_3 \in \pi_{G-\{d_1, d_2\}}(s, t)$, answers $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$ in $\tilde{O}(n^3)$ time.*

5.1 The Graph H

As in Section 3.1.1, we construct a graph H as follows: (Here let N be the sum of all edge weights in G .)

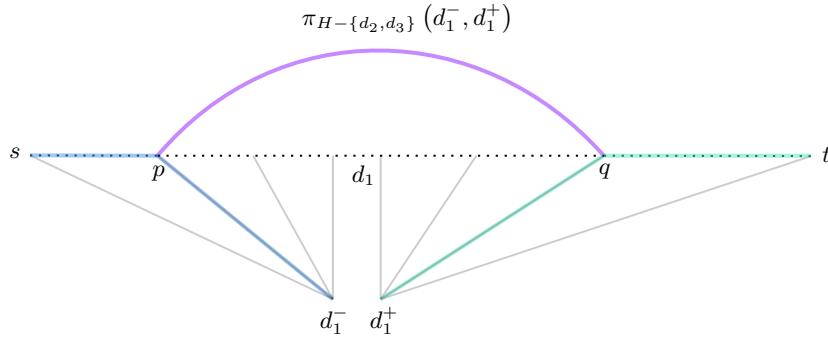
- First let H be a copy of $G - st$, that is, remove all edges on $\pi_G(s, t)$.
- For every edge $d = (x, y) \in st$ and x is before y on st , introduce two new vertices d^- and d^+ .
 - For every vertex x' on sx , the vertex d^- is connected with x' by an edge with weight $|sx'| + N$.
 - For every vertex y' on yt , the vertex d^+ is connected with y' by an edge with weight $|y't| + N$.

Here, since N is large enough, we can easily observe that for any shortest path $\pi_H(u, v)$ in H , if $u, v \in G$, it does not use the new edges; and if one or both of them are new vertices, new edges can only appear at both ends in $\pi_H(u, v)$.

We proceed to our lemma for this case:

Lemma 5.2. *Given G and one failure d_1 on st , for any d_2, d_3 not on st , in the graph H we defined,*

$$|\pi_{H-\{d_2, d_3\}}(d_1^-, d_1^+)| = |\pi_{G-\{d_1, d_2, d_3\}}(s, t)| + 2N.$$



Proof. **TOPROVE 11**

□

5.2 The Algorithm

We consider every d_2 that is on $\pi_{G-d_1}(s, t)$ for some d_1 on st . By Lemma 2.3, the number of possible d_2 is $O(n)$. We start from the graph H and consider every d_2 in any predetermined order and use the offline dynamic DSO from Section 4 and Theorem 2.5.

For every d_2 we considered, we first delete d_2 from the graph to get $H - d_2$. Then for every d_1 on the original shortest path st , and every d_3 on $\pi_{G-\{d_1, d_2\}}(s, t)$, we calculate the value $|\pi_{H-\{d_2, d_3\}}(d_1^-, d_1^+)| - 2N$ in this graph $H - d_2$ using our oracle. Finally, we add d_2 back to proceed.

By Lemma 5.2, the value $|\pi_{H-\{d_2, d_3\}}(d_1^-, d_1^+)| - 2N$ we computed equals $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$. Therefore, we have got the correct replacement path lengths for d_1, d_2, d_3 .

We have deleted and added $O(n)$ edges in total and queried the oracle once for every triple (d_1, d_2, d_3) . By Section 4 and Theorem 2.5, the total update time is in $\tilde{O}(n^3)$, and each query can be performed in $\tilde{O}(1)$ time. Therefore, the total time we use is $\tilde{O}(n^3)$.

6 Two Failed Edges on Original Shortest Path

In this section, we consider the case that exactly two failures d_1, d_2 are on the original shortest path st . Our approach follows the framework and adapts some techniques from [17].

In this section, without loss of generality, we assume there are 2^k edges on st for some integer $k = O(\log n)$. Otherwise, for any $\|st\|$ let $k = \lceil \log \|st\| \rceil = O(\log n)$, we can add an extra path from s' to s with $2^k - \|st\|$ edges with weight 0 and consider the 3FRP problem for (s', t) to solve the original 3FRP problem for (s, t) . With this assumption, we have $t = s \oplus 2^k$ on the original shortest path st .

Theorem 6.1. *There exists an algorithm that, given G with two points $s, t \in V(G)$, for all edges $d_1, d_2 \in st$, $d_3 \notin st$ and $d_3 \in \pi_{G-\{d_1, d_2\}}(s, t)$, answers $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$ in $\tilde{O}(n^3)$ time.*

6.1 The Graph H

We use the same graph H that is obtained by adding new vertices d^- and d^+ and their edges to $G - st$, as defined in Section 5.1. Also, we use the interval notation $[x, y]$ for $x, y \in st$ and x is closer to s , to denote the subpath xy on st in the original graph G .

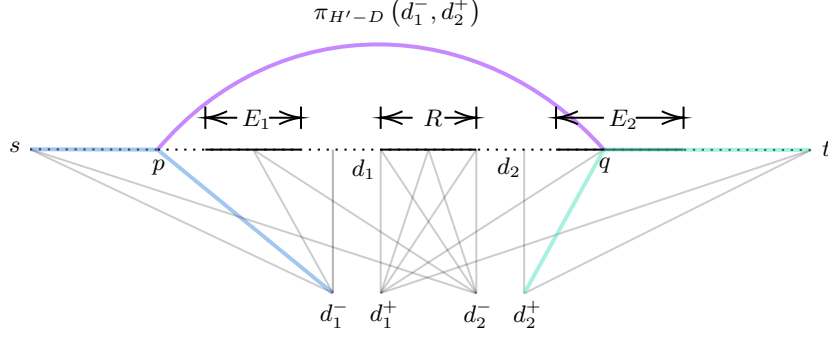
Consider two edge failures $d_1 = (s \oplus (l-1), s \oplus l)$, $d_2 = (s \oplus r, s \oplus (r+1))$ on st , $0 < l \leq r < 2^k$. Define $R = [s \oplus l, s \oplus r]$, that is, the interval between d_1 and d_2 , but not including them. Remind that in H , edges from d_1^- may replace the subpaths in the interval $[s, s \oplus (l-1)]$, and edges from d_2^+ may replace the subpaths in the interval $[s \oplus (r+1), t]$, but the detour can still go through edges in R . Before introducing our algorithm, we first prove some properties on H .

Consider any failure set D that is in $(G - st) \cup R$. Similarly, we may use the shortest path between d_1^- and d_2^+ that avoids D in $H \cup R$ to represent $\pi_{G-(\{d_1, d_2\} \cup D)}(s, t)$ in the original graph G .

Moreover, consider edges in $[s, s \oplus (l-1)]$ and $[s \oplus (r+1), t]$. Adding an arbitrary subset of them to the graph $H \cup R$ will not influence the distance between d_1^- and d_2^+ , as the edges from d_1^- to any vertex in $[s, s \oplus (l-1)]$ and (similarly) from d_2^+ to any vertex in $[s \oplus (r+1), t]$ will not be shortcut by adding these edges. We demonstrate the results in the following lemma:

Lemma 6.2. *Given G and two failures d_1, d_2 on st , consider arbitrary edge subsets $E_1 \subseteq [s, s \oplus (l-1)]$ and $E_2 \subseteq [s \oplus (r+1), t]$. Define $H' = (H \cup R \cup E_1 \cup E_2)$ to be the graph obtained by adding the whole $R = [s \oplus l, s \oplus r]$ and E_1, E_2 to H . Then for any edge set $D \subseteq (G - st) \cup R$,*

$$|\pi_{(H'-D)}(d_1^-, d_2^+)| = |\pi_{G-(\{d_1, d_2\} \cup D)}(s, t)| + 2N.$$



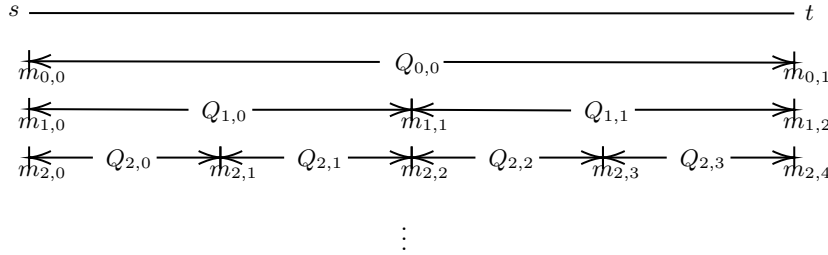
Proof. **TOPROVE 12** □

6.2 The Binary Partition Structure

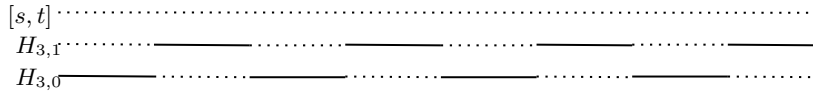
We define a binary partition structure similar to [17] as follows. Recall that we assume there are 2^k edges on st , so $t = s \oplus 2^k$. Consider a set $\{m_{i,j} | 0 \leq i \leq k, 0 \leq j \leq 2^i\}$ to be:

- $m_{0,0} = s, m_{0,1} = t$.
- For any $1 \leq i \leq k$, $0 \leq j \leq 2^i$, if j is even, we define $m_{i,j} = m_{i-1,j/2}$; if j is odd, we define $m_{i,j} = m_{i-1,(j-1)/2} \oplus 2^{k-i}$.

Each vertex may be represented by multiple elements in $\{m_{i,j}\}$. Moreover, we define intervals $Q_{i,j} = [m_{i,j}, m_{i,j+1}]$ for all $0 \leq i \leq k$ and $0 \leq j < 2^i$. Thus, in level i , we will have 2^i edge-disjoint ranges $Q_{i,0}, Q_{i,1}, \dots, Q_{i,2^i-1}$, each with 2^{k-i} edges, and their union is the whole original shortest path st .



We further define $2k$ graphs as follows. For all $1 \leq i \leq k$, let $H_{i,0} = H \cup_{\text{even } j} Q_{i,j}$ and let $H_{i,1} = H \cup_{\text{odd } j} Q_{i,j}$. These graphs are obtained from H , with $H_{i,0}$ adding the ranges with even orders in level i , and $H_{i,1}$ adding the ranges with odd orders. (Note that in H the original shortest path st is removed, and we now add half of the ranges back to get $H_{i,0}$ and $H_{i,1}$.) A demonstration of the graphs $H_{i,0}$ and $H_{i,1}$ is as below.



Consider two failures $d_1 = (s \oplus (l-1), s \oplus l)$, $d_2 = (s \oplus r, s \oplus (r+1))$ on st , with $0 < l \leq r < 2^k$. In the partition structure, we can find a vertex $m_{i,j}$ (where i is minimized) separating them, so d_1 is on $Q_{i,j-1}$ and d_2 is on $Q_{i,j}$. We know j should be odd since otherwise the vertex $m_{i,j}$ will also appear as $m_{i-1,j/2}$. Let the whole interval between d_1 and d_2 be $R = [s \oplus l, s \oplus r]$, the left part of R be $R_l = [s \oplus l, m_{i,j}]$ and the right part be $R_r = [m_{i,j}, s \oplus r]$. We can see that $H_{i,0}$ contains R_l but not R_r , and $H_{i,1}$ contains R_r but not R_l .

6.3 The Algorithm

Let's consider what we need to compute $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$. Here, we claim the following theorem for this case with two edge failures on st :

Theorem 6.3. *The length of the replacement path $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$ equals the minimum one in the following four values:*

- $|\pi_{H-d_3}(d_1^-, d_2^+)| - 2N$ (as in Lemma 6.4);
- $|\pi_{H_{i,0}-d_1-d_3}(d_1^-, d_2^+)| - 2N$ (as in Lemma 6.5);
- $|\pi_{H_{i,1}-d_2-d_3}(d_1^-, d_2^+)| - 2N$ (as in Lemma 6.6);
- $\min\{|\pi_{H_{i,0}-d_1-d_3}(d_1^-, m_{i,j})|, |\pi_{H_{i,1}-d_2-d_3}(d_1^-, m_{i,j})|\}$
 $+ \min\{|\pi_{H_{i,0}-d_1-d_3}(m_{i,j}, d_2^+)|, |\pi_{H_{i,1}-d_2-d_3}(m_{i,j}, d_2^+)|\} - 2N$. (as in Lemma 6.12)

The proof will be shown in Section 6.4.

Now let's consider what do we need to compute these values. For $|\pi_{H-d_3}(d_1^-, d_2^+)| - 2N$, we can obtain it by one DSO query with failure d_3 in H . For the other values, we notice that all these shortest path values are either shortest paths in $\pi_{H_{i,0}-d_1-d_3}$, or shortest paths in $\pi_{H_{i,1}-d_2-d_3}$. These can be obtained by one DSO query with failure d_3 in either $H_{i,0} - d_1$ or $H_{i,1} - d_2$.

Now note that there are only $O(\log n)$ graphs in $\{H_{i,j}\}$. By the offline dynamic DSO from Section 4 and Theorem 2.5, we can compute all DSOs for graphs like $\{H_{i,j} - d$ for every $d \in st$ by deleting and adding back d one by one. There are $O(n)$ operations done for each graph $\{H_{i,j}\}$, therefore, the total update time is $\tilde{O}(n^3)$.

When we obtained a DSO for $\{H_{i,j} - d$ at some time in the offline dynamic oracle, we can operate all queries related to it. There are $O(n^3)$ queries in total (constant number of queries need to be made for any set $\{d_1, d_2, d_3\}$), and each query can be done in time $\tilde{O}(1)$. Therefore, the total query time is also in $\tilde{O}(n^3)$. In conclusion, we can run the algorithm for all sets $\{d_1, d_2, d_3\}$ satisfying only d_1, d_2 are on st , in $\tilde{O}(n^3)$ time.

6.4 Resolving Possible Path Cases

We observe that the replacement shortest path $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ has the following structure:

- It starts from s , goes along st , and diverges before d_1 .
- It may converges and diverges in st between d_1 and d_2 (possibly in either direction).
- It converges in st after d_2 , goes along st , and ends at t .

Now consider the part $R = [s \oplus l, s \oplus r]$ between d_1 and d_2 on st . For the replacement shortest path $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$, regarding the vertex $m_{i,j}$, there are four cases we need to consider:

- There is no intersection with $R = [s \oplus l, s \oplus r]$.
- The edge intersection is fully in $R_l = [s \oplus l, m_{i,j}]$.
- The edge intersection is fully in $R_r = [m_{i,j}, s \oplus r]$.

- The edge intersection is an interval containing the vertex $m_{i,j}$.

First, consider the case that the replacement path has no intersection with $R = [s \oplus l, s \oplus r]$. Here, we can assume that the whole interval R is also avoided and we consider simulating it using the graph H (with everything in R deleted). We get the following lemma:

Lemma 6.4. *Starting from the graph H , we have:*

$$|\pi_{H-d_3}(d_1^-, d_2^+)| = |\pi_{G-(\{d_1, d_2, d_3\} \cup R)}(s, t)| + 2N \geq |\pi_{G-\{d_1, d_2, d_3\}}(s, t)| + 2N.$$

Moreover, if $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ has no edge intersection with $R = [s \oplus l, s \oplus r]$, then:

$$|\pi_{H-d_3}(d_1^-, d_2^+)| = |\pi_{G-\{d_1, d_2, d_3\}}(s, t)| + 2N.$$

Proof. **TOPROVE 13** □

Now suppose the replacement path has an intersection with R that is only on a subinterval of R_l . Here we can assume that the interval R_r is also avoided and we consider simulating it using the graph $H_{i,0}$ (with everything in R_r deleted but everything in R_l still included). We get the following lemma:

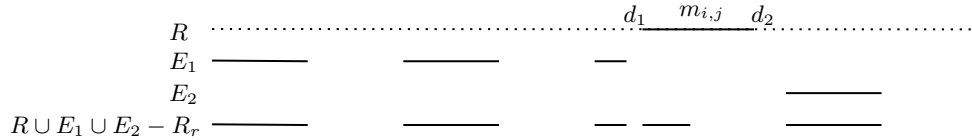
Lemma 6.5. *Starting from the graph $H_{i,0} - d_1$, we have:*

$$|\pi_{H_{i,0}-d_1-d_3}(d_1^-, d_2^+)| = |\pi_{G-(\{d_1, d_2, d_3\} \cup R_r)}(s, t)| + 2N \geq |\pi_{G-\{d_1, d_2, d_3\}}(s, t)| + 2N.$$

Moreover, if $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ intersects $R = [s \oplus l, s \oplus r]$ only on $R_l = [s \oplus l, m_{i,j}]$, then we have:

$$|\pi_{H_{i,0}-d_1-d_3}(d_1^-, d_2^+)| = |\pi_{G-\{d_1, d_2, d_3\}}(s, t)| + 2N.$$

(Note that $H_{i,0}$ includes $\{d_1\} \cup R_l$ and does not include $R_r \cup \{d_2\}$.)



Proof. **TOPROVE 14** □

Now suppose the replacement path has an intersection with R that is only on a subinterval of R_r . This is exactly the symmetric case and we demonstrate it in the following lemma:

Lemma 6.6. *Starting from the graph $H_{i,1} - d_2$, we have:*

$$|\pi_{H_{i,1}-d_2-d_3}(d_1^-, d_2^+)| = |\pi_{G-(\{d_1, d_2, d_3\} \cup R_l)}(s, t)| + 2N \geq |\pi_{G-\{d_1, d_2, d_3\}}(s, t)| + 2N.$$

If $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ intersects $R = [s \oplus l, s \oplus r]$ only on $R_r = [m_{i,j}, s \oplus r]$, then we have:

$$|\pi_{H_{i,1}-d_2-d_3}(d_1^-, d_2^+)| = |\pi_{G-\{d_1, d_2, d_3\}}(s, t)| + 2N.$$

(Note that $H_{i,1}$ includes $R_r \cup \{d_2\}$ and does not include $\{d_1\} \cup R_l$.)

Proof. **TOPROVE 15** □

Finally we consider the case that $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ goes through the vertex $m_{i,j}$. Here, we consider breaking the whole replacement shortest path into two parts: One from s to $m_{i,j}$, and one from $m_{i,j}$ to t , in the graph $G - \{d_1, d_2, d_3\}$. We can immediately obtain the following lemma:

Lemma 6.7. *We have:*

$$|\pi_{G-\{d_1, d_2, d_3\}}(s, t)| \leq |\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| + |\pi_{G-\{d_1, d_2, d_3\}}(m_{i,j}, t)|.$$

If $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ contains the vertex $m_{i,j}$, then:

$$|\pi_{G-\{d_1, d_2, d_3\}}(s, t)| = |\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| + |\pi_{G-\{d_1, d_2, d_3\}}(m_{i,j}, t)|.$$

Proof. **TOPROVE 16** □

Finally, we consider how to compute these two parts when $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ contains the vertex $m_{i,j}$ as in Lemma 6.7. Here, we will show in the following lemmas the approach to calculate $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$, and after that, we can calculate $\pi_{G-\{d_1, d_2, d_3\}}(m_{i,j}, t)$ symmetrically.

Consider the path $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$. We observe that the path cannot intersect the interval R on edges both in R_l and R_r . If so, then there must be a shorter path from s to $m_{i,j}$ by taking the shortest path after it first intersects R .

Moreover, if $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$ goes through $m_{i,j}$, then we show that the first part $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$ cannot go through any vertex in $[s \oplus (r+1), t]$ as well. Otherwise, if it goes through some vertex $q \in [s \oplus (r+1), t]$ before $m_{i,j}$, however, $\pi_{G-\{d_1, d_2, d_3\}}(s, q) \circ qt$ will be a shorter path that does not go through $m_{i,j}$, which comes to contradiction.

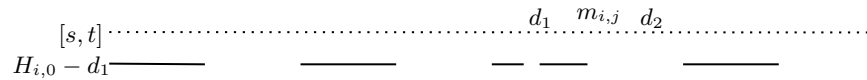
In the following lemmas, we make use of the graphs $H_{i,0} - d_1$ and $H_{i,1} - d_2$ to capture both cases that $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$ intersects R only on R_l or only on R_r . We obtain the following lemmas:

Lemma 6.8. *Consider the graph $H_{i,0} - d_1$, we have:*

$$|\pi_{H_{i,0}-d_1-d_3}(d_1^-, m_{i,j})| \geq |\pi_{G-\{d_1, d_2, d_3\} \cup R_r}(s, m_{i,j})| + N \geq |\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| + N.$$

Moreover, if $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$ does not go through any vertex in $[s \oplus (r+1), t]$ and intersects R only on R_l , then:

$$|\pi_{H_{i,0}-d_1-d_3}(d_1^-, m_{i,j})| = |\pi_{G-\{d_1, d_2, d_3\} \cup R_r}(s, m_{i,j})| + N = |\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| + N.$$



Proof. **TOPROVE 17** □

Following we state the symmetric case that $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$ intersects R only on R_r . Essentially the same proof works for this lemma.

Lemma 6.9. *Consider the graph $H_{i,1} - d_2$, we have:*

$$|\pi_{H_{i,1}-d_2-d_3}(d_1^-, m_{i,j})| \geq |\pi_{G-\{d_1, d_2, d_3\} \cup R_l}(s, m_{i,j})| + N \geq |\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| + N.$$

Moreover, if $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$ does not go through any vertex in $[s \oplus (r+1), t]$ and intersects R only on R_r , then:

$$|\pi_{H_{i,1}-d_2-d_3}(d_1^-, m_{i,j})| = |\pi_{G-\{d_1, d_2, d_3\} \cup R_l}(s, m_{i,j})| + N = |\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| + N.$$

Proof. **TOPROVE 18** □

Lemma 6.10. *We can bound the length of $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$ by:*

$$|\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| \leq \min\{|\pi_{H_{i,0}-d_1-d_3}(d_1^-, m_{i,j})|, |\pi_{H_{i,1}-d_2-d_3}(d_1^-, m_{i,j})|\} - N.$$

Moreover, if $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$ does not go through any node in $[s \oplus (r+1), t]$, then it will become equation:

$$|\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| = \min\{|\pi_{H_{i,0}-d_1-d_3}(d_1^-, m_{i,j})|, |\pi_{H_{i,1}-d_2-d_3}(d_1^-, m_{i,j})|\} - N.$$

Proof. **TOPROVE 19** □

Symmetrically, we can obtain the length of the second part of the shortest path $|\pi_{G-\{d_1, d_2, d_3\}}(m_{i,j}, t)|$. We state the following lemma here and the proof is necessarily the same as Lemma 6.10.

Lemma 6.11. *We can bound the length of $\pi_{G-\{d_1, d_2, d_3\}}(m_{i,j}, t)$ by:*

$$|\pi_{G-\{d_1, d_2, d_3\}}(m_{i,j}, t)| \leq \min\{|\pi_{H_{i,0}-d_1-d_3}(m_{i,j}, d_2^+)|, |\pi_{H_{i,1}-d_2-d_3}(m_{i,j}, d_2^+)|\} - N.$$

Moreover, if $\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})$ does not go through any node in $[s, s \oplus (l-1)]$, then it will become equation:

$$|\pi_{G-\{d_1, d_2, d_3\}}(m_{i,j}, t)| = \min\{|\pi_{H_{i,0}-d_1-d_3}(m_{i,j}, d_2^+)|, |\pi_{H_{i,1}-d_2-d_3}(m_{i,j}, d_2^+)|\} - N.$$

Now consider the case that $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ goes through $m_{i,j}$. We claim below that we have an equation for the last value as in Lemma 6.7.

Lemma 6.12. *If $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$ goes through $m_{i,j}$, then:*

- $|\pi_{G-\{d_1, d_2, d_3\}}(s, m_{i,j})| = \min\{|\pi_{H_{i,0}-d_1-d_3}(d_1^-, m_{i,j})|, |\pi_{H_{i,1}-d_2-d_3}(d_1^-, m_{i,j})|\} - N;$
- $|\pi_{G-\{d_1, d_2, d_3\}}(m_{i,j}, t)| = \min\{|\pi_{H_{i,0}-d_1-d_3}(m_{i,j}, d_2^+)|, |\pi_{H_{i,1}-d_2-d_3}(m_{i,j}, d_2^+)|\} - N.$

Proof. **TOPROVE 20** □

Now all cases are evaluated in the lemmas and we begin our final analysis for $\pi_{G-\{d_1, d_2, d_3\}}(s, t)$. We can get it by a minimization among all the four cases above in this section.

Theorem 6.13. (cf. Theorem 6.3) *The length of the replacement path $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$ equals the minimum one in the following four values:*

- $|\pi_{H-d_3}(d_1^-, d_2^+)| - 2N$ (as in Lemma 6.4);
- $|\pi_{H_{i,0}-d_1-d_3}(d_1^-, d_2^+)| - 2N$ (as in Lemma 6.5);
- $|\pi_{H_{i,1}-d_2-d_3}(d_1^-, d_2^+)| - 2N$ (as in Lemma 6.6);
- $\min\{|\pi_{H_{i,0}-d_1-d_3}(d_1^-, m_{i,j})|, |\pi_{H_{i,1}-d_2-d_3}(d_1^-, m_{i,j})|\}$
 $+ \min\{|\pi_{H_{i,0}-d_1-d_3}(m_{i,j}, d_2^+)|, |\pi_{H_{i,1}-d_2-d_3}(m_{i,j}, d_2^+)|\} - 2N$ (adding two equations in Lemma 6.12).

Proof. **TOPROVE 21** □

This ends our proof of Theorem 6.3 and finishes our analysis of the case.

7 Three Failed Edges on Original Shortest Path

In this section, we consider the last case that all three failures d_1, d_2, d_3 are on the original shortest path st from left (i.e. close to s) to right (i.e. close to t), which cut st into four intervals D_1, D_2, D_3, D_4 , also from left to right. We prove the following theorem:

Theorem 7.1. *There exists an algorithm that, given undirected G with two vertices $s, t \in V(G)$, for all edges $d_1, d_2, d_3 \in st$, answers $|\pi_{G-\{d_1, d_2, d_3\}}(s, t)|$ in $\tilde{O}(n^3)$ time.*

In this section we still assume that there are 2^k edges on st for some integer $k = O(\log n)$. Let the replacement path be $\pi' = \pi_{G-\{d_1, d_2, d_3\}}(s, t)$. It is straightforward to observe the followings:

Observation 7.2 (informal). *The replacement path π' satisfies:*

- It diverges at D_1 at first, and converges at D_4 at last.
- It may converges and diverges at D_2, D_3 , or both of them (in any order) in the middle.
- Between a convergence point and next divergence point on st , it will take the shortest path on st .
- Between a divergence point and next convergence point on st , it will take the shortest path in $G - st$.

With this observation, we can first compute all-pairs shortest paths in $G - st$, which makes it possible to simplify our analysis to purely interval queries for the nodes on the original shortest path st .

Therefore, in this section, we use a **range tree structure** from Section 6.2, Namely:

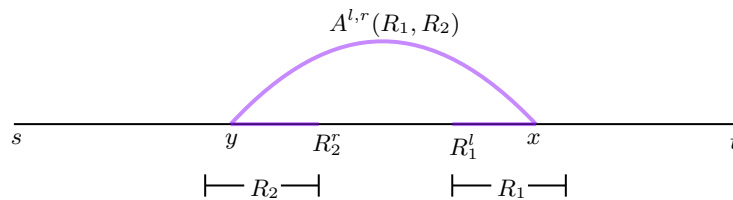
- The ranges contained in the range tree are all $\{Q_{i,j}\}$ in the binary partition structure.
- For any range $\{Q_{i,j}\}$ with $i < k$, let its childs to be $Q_{i+1,2j}$ and $Q_{i+1,2j+1}$, which divide it into two subranges.

Note that any interval on st can be represented as a union of at most $O(\log n)$ ranges in the range tree. In this section, for any interval R on st , we use R^l to denote the leftmost vertex of R , and use R^r to denote the rightmost vertex of R .

Below, we build an oracle based on the range tree structure. Moreover, we also show how to use these oracles for 3-fault queries.

7.1 Oracle Build-up

Theorem 7.3. *For a graph G and vertices s, t , we can construct an oracle A in $\tilde{O}(n^3)$ preprocessing time that answers the following type of query in $\tilde{O}(1)$ time: given two vertex-disjoint intervals R_1 and R_2 of consecutive edges on the original shortest path st (R_1, R_2 can be single vertices), answer the shortest path that starts from the leftmost (rightmost) vertex of R_1 , diverges from R_1 , converges in R_2 , and finally ends at the leftmost (rightmost) vertex of R_2 . Denote these by: $A^{l,l}(R_1, R_2) = \min_{x \in R_1, y \in R_2} \{R_1^l x \circ \pi_{G-st}(x, y) \circ y R_2^l\}$ and $A^{l,r}, A^{r,l}, A^{r,r}$ are defined accordingly.*



Note that we can store the path by storing the positions of x and y . First, we prove the following lemma.

Lemma 7.4. *If the interval R_1 on st is the union of intervals R_1^l and R_1^r ($R_1^r = R_1^{l'}$ or there is an edge connecting R_1^r and $R_1^{l'}$), and we already have $A^{l,l}(R_1^l, R_2)$ and $A^{l,l}(R_1^r, R_2)$, then we can obtain $A^{l,l}(R_1, R_2)$ in $O(1)$ time. Symmetrically, we can get $A^{l,l}(R_1, R_2)$ from $A^{l,l}(R_1, R_2^l)$ and $A^{l,l}(R_1, R_2^r)$ if R_2 is the union of R_2^l and R_2^r . This holds for all other $A^{l,r}$, $A^{r,l}$ and $A^{r,r}$ cases as well.*

Proof. **TOPROVE 22** □

Thus, the data structure of Theorem 7.3 stores the paths for ranges R_1 and R_2 in the range tree structure or single vertices. The preprocessing and query algorithms for $A^{l,l}$ are as follows, and $A^{l,r}$, $A^{r,l}$, $A^{r,r}$ are similar.

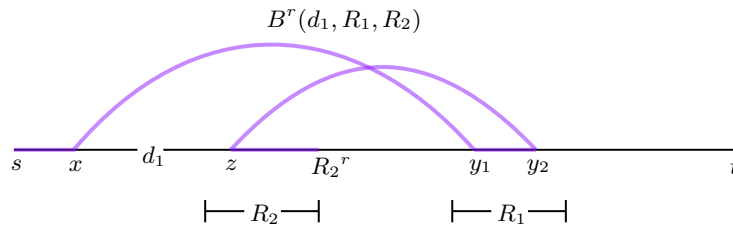
- **Preprocessing:** First compute $A^{l,l}(a, b)$ for all vertices a, b on st , which can be done by APSP in $G - st$ in $O(n^3)$ time. Then by Lemma 7.4, we can compute $A^{l,l}(R_1, R_2)$ for all vertex-disjoint ranges R_1 and R_2 in the range tree structure in $O(n^2)$ time.
- **Query:** Since every interval is the union of $O(\log n)$ ranges in the range tree structure (or a single vertex), by Lemma 7.4, we can get $A^{l,l}(R_1, R_2)$ for any R_1 and R_2 in $\tilde{O}(1)$ time.

This completes the proof of Theorem 7.3.

Theorem 7.5. *For a graph G and vertices s, t , we can construct an oracle B in $\tilde{O}(n^3)$ preprocessing time, and answer the following type of query in $\tilde{O}(1)$ time: given an edge $d_1 = (a, b)$ on st and two vertex-disjoint intervals R_1 and R_2 of consecutive edges on st after d_1 (R_1 can be on the left or right of R_2), answer the shortest path that starts from s , diverges before d_1 , converges in R_1 , and diverges from R_1 , then converges in R_2 , and finally ends at the leftmost (rightmost) point of R_2 . Denote them by*

$$B^l(d_1, R_1, R_2) = \min_{x \leq a, y_1, y_2 \in R_1, z \in R_2} \{sx \circ \pi_{G-st}(x, y_1) \circ y_1 y_2 \circ \pi_{G-st}(y_2, z) \circ z R_2^l\}$$

$$B^r(d_1, R_1, R_2) = \min_{x \leq a, y_1, y_2 \in R_1, z \in R_2} \{sx \circ \pi_{G-st}(x, y_1) \circ y_1 y_2 \circ \pi_{G-st}(y_2, z) \circ z R_2^r\}$$



Also we can store the path by storing the positions of x, y_1, y_2, z . We also apply the idea of Lemma 7.4 to this theorem.

Lemma 7.6. *If the interval R_1 on st is the union of intervals R_1^l and R_1^r ($R_1^r = R_1^{l'}$ or there is an edge connecting R_1^r and $R_1^{l'}$), and we already have $B^l(d_1, R_1^l, R_2)$, $B^l(d_1, R_1^r, R_2)$ and oracle A , then we can get $B^l(d_1, R_1, R_2)$ in $\tilde{O}(1)$ time. Also, we can get $B^l(d_1, R_1, R_2)$ from $B^l(d_1, R_1, R_2^l)$ and $B^l(d_1, R_1, R_2^r)$ in $O(1)$ time if R_2 is the union of R_2^l and R_2^r . This also holds for B^r .*

Proof. **TOPROVE 23** □

So the data structure of Theorem 7.5 stores the paths for all d_1 on s, t and ranges R_1 and R_2 in the range tree structure or single vertices, so the total space is $O(n^3)$. The preprocessing and query algorithm for B^l is as follows, and B^r is similar.

- Preprocessing: For all d_1 on s, t and R_2 in the range tree structure or single vertex, consider R_1 in the range tree structure which is disjoint with R_2 .
 - First compute $B^l(d_1, x, R_2)$ for all single vertex x after d_1 and not in R_2 , which we can get from oracle A in $\tilde{O}(n^3)$ time. Let R_3 be the interval from s to the left vertex of d_1 , then $B^l(d_1, x, R_2) = A^{l,l}(R_3, x) \circ A^{l,l}(x, R_2)$.
 - Then by Lemma 7.6, we can compute $B^l(d_1, R_1, R_2)$ for all vertex-disjoint ranges R_1 and R_2 in the range tree structure in $\tilde{O}(n^3)$ time.
- Query: Since every interval on st is the union of $O(\log n)$ ranges in the range tree structure (or a single vertex), by Lemma 7.6, we can get $B^l(d_1, R_1, R_2)$ for any d_1, R_1, R_2 in $\tilde{O}(1)$ time.

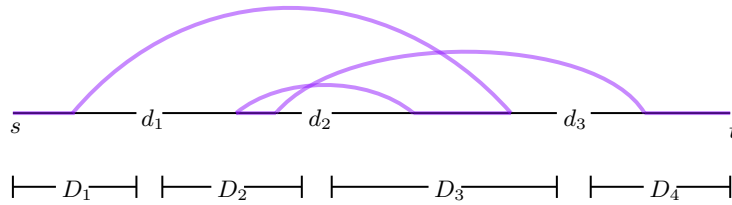
This completes the proof of Theorem 7.5. Note that Theorem 7.5 can be extended to the symmetric case that the shortest path that starts at the leftmost (rightmost) point of R_1 , diverges from R_1 , converges in R_2 , and diverges from R_2 , then converges after d_3 , and finally ends at t , for d_3 on s, t and vertex-disjoint intervals R_1 and R_2 on st before d_3 .

7.2 Answering 3-Fault Queries

Now we are ready to answer queries for the replacement path $\pi' = \pi_{G-\{d_1, d_2, d_3\}}(s, t)$ given three failed edges d_1, d_2, d_3 on st . We suppose the original shortest path st is cut into four intervals D_1, D_2, D_3, D_4 . Here we first state a more detailed version of Observation 7.2:

Observation 7.7 (cf. 7.2, formal). *The 3-fault replacement path π' falls in one of the cases:*

- **Type 1:** Diverges at D_1 , converges at D_4 .
- **Type 2:** Diverges at D_1 , converges and diverges at D_2 , and finally converges at D_4 .
- **Type 3:** Diverges at D_1 , converges and diverges at D_3 , and finally converges at D_4 .
- **Type 4:** Diverges at D_1 , converges and diverges at D_2 , then converges and diverges at D_3 , and finally converges at D_4 .
- **Type 5:** Diverges at D_1 , converges and diverges at D_3 , then converges and diverges at D_2 , and finally converges at D_4 .



Proof. **TOPROVE 24**

□

Therefore, in the following part we consider the shortest candidates in all 5 types, and a minimization of them will be the correct replacement path that we want.

When shortest path π' is of type 1, we can solve this case by straightly calling $A^{l,r}(D_1, D_4)$ from Theorem 7.3, which captures type 1 paths.

When shortest path π' is of type 2 or type 3, similarly, by Theorem 7.5, we can call $B^r(d_1, D_2, D_4)$ and $B^r(d_1, D_3, D_4)$ to capture type 2 and type 3 paths, respectively.

When shortest path π' is of type 4 or type 5, the case is harder. For example, suppose we want to compute the path with type 5 below (and similarly for type 4). Here, as illustrated in the above figure, the path can be formally expressed as:

$$\min_{s_0 \in D_1, x_1, x_2 \in D_3, y_1, y_2 \in D_2, t_0 \in D_4} \{ss_0 \circ \pi_{G-st}(s_0, x_1) \circ x_1x_2 \circ \pi_{G-st}(x_2, y_1) \circ y_1y_2 \circ \pi_{G-st}(y_2, t_0) \circ t_0t\}.$$

Lemma 7.8. *Let d_1, d_2, d_3 be three edges on $\pi(s, t)$, and x be a vertex (or an edge) on $\pi(s, t)$, which is after d_1 and before d_3 . Calculating a shortest path of type 4 or 5 from s to t in $G - \{d_1, d_2, d_3\}$ that goes through x requires time $\tilde{O}(1)$.*

Proof. **TOPROVE 25** □

Below we generalize type 4 and 5 paths as **snake paths** when there are $w = \tilde{O}(1)$ edges removed.

Definition 7.9. Suppose $w = \tilde{O}(1)$, and there are w edges d_1, d_2, \dots, d_w on st removed. Suppose they cut st into $w + 1$ intervals D_1, \dots, D_{w+1} , a **snake path** is defined as a path that only converges and then diverges in exactly two intervals in D_2, \dots, D_w in the middle.

With this definition, we similarly extend the lemma as follows:

Lemma 7.10. *If x is a vertex (or an edge) after d_1 and before d_w , then calculating a shortest snake path that goes through x avoiding $\{d_1, d_2, \dots, d_w\}$ still requires time $\tilde{O}(1)$. Moreover, we can compute a shortest snake path in $G - \{d_1, d_2, \dots, d_w\}$ in time $\tilde{O}(n)$.*

Proof. **TOPROVE 26** □

We are ready to close this section by proving the following lemma.

Lemma 7.11. *Let d_1, d_3 be two edges on $\pi(s, t)$. Calculating a shortest snake path from s to t in $G - \{d_1, d_2, d_3\}$ for all possible d_2 between d_1 and d_3 requires time $\tilde{O}(n)$.*

We can easily see that, if Lemma 7.11 is true, then we can enumerate all possible d_1 and d_3 and then get answers for all d_2 . Therefore, we can answer the replacement path for all triples $\{d_1, d_2, d_3\}$, and the total time is in $\tilde{O}(n^3)$. Therefore, to prove Theorem 7.1, it remains for us to prove this lemma.

Proof. **TOPROVE 27** □

8 Hardness of 2FRP

In this section we show that the 2FRP problem in undirected graphs is hard, assuming the APSP conjecture. We note that in directed graphs, 1FRP is as hard as APSP [31]. However, in undirected graphs, 1FRP can be solved in almost-optimal $\tilde{O}(m)$ time [25].

Theorem 8.1. *If 2FRP in undirected weighted graphs can be solved in $O(n^{3-\epsilon})$ time for any $\epsilon > 0$, then in undirected weighted graphs, the all-pairs shortest path problem can also be solved in $O(n^{3-\epsilon})$ time.*

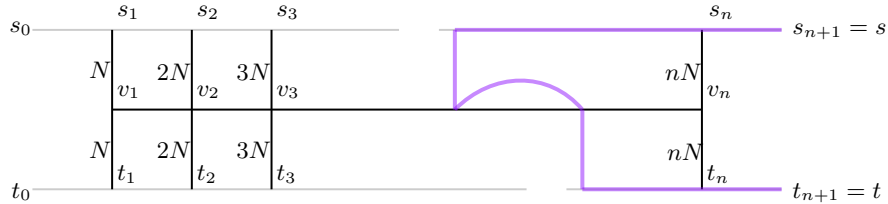
We remark that by [31], the hardness of the all-pairs shortest path problem in undirected weighted graphs is equivalent to the hardness of APSP. Therefore, any algorithm in $O(n^{3-\epsilon})$ time for 2FRP will refute the APSP conjecture. So we immediately get the corollary:

Corollary 8.2. *Assuming the APSP conjecture that APSP cannot be solved in $O(n^{3-\epsilon})$ time for any $\epsilon > 0$, 2FRP problem in undirected weighted graphs cannot be solved in $O(n^{3-\epsilon})$ time for any $\epsilon > 0$.*

Suppose toward contradiction that we can solve 2FRP in $O(n^{3-\epsilon})$ time. Consider any undirected weighted all-pairs shortest path instance G with n nodes v_1, v_2, \dots, v_n , we will construct an undirected weighted 2FRP instance H , solving which will result in solving all-pairs shortest path in G .

Let N be a number that is larger than the sum of all weights in G . Let S be a path with $(n+2)$ nodes $(s_0, s_1, s_2, \dots, s_n, s_{n+1} = s)$ with weight 0 edges (s_i, s_{i+1}) for all $0 \leq i \leq n$. Similarly, let T be a path with $(n+2)$ nodes $(t_0, t_1, t_2, \dots, t_n, t_{n+1} = t)$ with weight 0 edges (t_i, t_{i+1}) for all $0 \leq i \leq n$. Moreover, we construct two matchings $E_1 = \{(s_i, v_i)\}$ with weight $w(s_i, v_i) = iN$ for all $1 \leq i \leq n$, and $E_2 = \{(v_i, t_i)\}$ with weight $w(v_i, t_i) = iN$ for all $1 \leq i \leq n$.

Let $H = S \cup E_1 \cup G \cup E_2 \cup T$. By inspection, the s to t shortest path in H is precisely the path goes from s to s_1 in S , from s_1 to v_1 to t_1 , and then from t_1 to t in T . Consider any pair of failed edges, one on $\pi_H(s, s_1)$ and the other on $\pi_H(t_1, t)$. We will show the following relationship for 2FRP in H and all-pairs shortest paths in G :



Lemma 8.3. *For any $1 \leq i, j \leq n$, let $D = \{(s_{i-1}, s_i), (t_{j-1}, t_j)\}$ be a set of two edge failures with one in S and one in T . Then regarding the failure set D :*

$$|\pi_{H-D}(s, t)| = |\pi_G(v_i, v_j)| + (i+j)N.$$

Proof. **TOPROVE 28** □

With this lemma, we can prove Theorem 8.1. Considering any undirected weighted all-pairs shortest path instance G , we construct an undirected weighted graph H as above. Now we check all failure set $D = \{(s_{i-1}, s_i), (t_{j-1}, t_j)\}$ for all pairs (i, j) , $1 \leq i, j \leq n$. By Lemma 8.3, we can obtain $|\pi_G(v_i, v_j)|$ from $|\pi_{H-D}(s, t)|$, which means by solving 2FRP in H we can solve all-pairs shortest paths in G .

Since we know $n' = 2(n+2) + n = O(n)$ nodes in H , if we have a truly subcubic time algorithm that solving 2FRP in time $O((n')^{3-\epsilon}) = O(n^{3-\epsilon})$ for H , then using this algorithm we can solve all-pairs shortest paths in $O(n^{3-\epsilon})$ time for G . Therefore, Theorem 8.1 is true.

9 Almost Optimal Single-Source Replacement Path under Multiple Edge Failures

For a graph G and a vertex s , the 2-fault single-source replacement path problem is that for every tuple of two edges d_1, d_2 and a vertex t , answer the distance between s and t in G after removing edges d_1 and d_2 . We show how to solve it in almost optimal time $\tilde{O}(n^3)$ for undirected weighted graphs, via the incremental DSO.

Let T be the shortest path tree of s in G , with edges e_1, e_2, \dots, e_{n-1} . Our algorithm goes as follows: we maintain a dynamic DSO on G . For each $1 \leq i < n$, we remove e_i in G , maintain the dynamic DSO. For every vertex t_j in the subtree of e_i in T and every edge d_k on the replacement path $\pi_{G-e_i}(s, t_j)$, save the distance $\pi_{(G-e_i)-d_k}(s, t_j)$ in $G - e_i$ via querying the dynamic DSO. At the end of each step, add e_i back into G .

Correctness. For each tuple (d_1, d_2, t) , if none of d_1, d_2 is in T , we know the shortest path st is the same after removing d_1, d_2 . W.l.o.g., suppose $d_1 \in T$, and t is in the subtree of d_1 in T . If d_2 is not in $\pi_{G-d_1}(s, t)$, we know that $\pi_{G-d_1-d_2}(s, t) = \pi_{G-d_1}(s, t)$, and this value can be queried in the static DSO. If $d_2 \in \pi_{G-d_1}(s, t)$, we know the value equals $\pi_{(G-d_1)-d_2}(s, t)$ from the dynamic DSO in graph $G - d_1$, which is obtained in our algorithm.

Time Analysis. The incremental DSO in Section 4 can be viewed as an offline fully dynamic DSO by Theorem 2.4. Therefore, the update time is $\tilde{O}(n^2)$ for each edge e_i . The number of edges on a shortest path tree is $O(n)$, so the total update time is $\tilde{O}(n^3)$. The number of distances we query is $O(n^3)$, each with query time $\tilde{O}(1)$, so the total query time is also $\tilde{O}(n^3)$. We also retrieve all 1-fault single-source replacement paths $\pi_{G-e_i}(s, t_j)$ in the static DSO, and the time is also bounded by $\tilde{O}(n^3)$.

Extend to f -fault SSRP for $f \geq 2$. As the f FRP reduction in [30], when we have an f -fault SSRP algorithm in $O(n^s)$ time, we can construct an $(f+1)$ -fault SSRP algorithm in $O(n^{s+1})$ time, by computing the f -fault SSRP in graph $G - e$ for every e in the shortest path tree from s . Thus, the undirected 2-fault SSRP algorithm in $\tilde{O}(n^3)$ time can be extended to undirected f -fault SSRP algorithm in $\tilde{O}(n^{f+1})$ time for all $f \geq 2$, which is almost optimal.

References

- [1] Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication, 2024.
- [2] Noga Alon, Shiri Chechik, and Sarel Cohen. Deterministic combinatorial replacement paths and distance sensitivity oracles. In *International Colloquium on Automata, Languages and Programming*, 2019.
- [3] Michael A. Bender and Martín Farach-Colton. The LCA problem revisited. In Gaston H. Gonnet and Alfredo Viola, editors, *LATIN 2000: Theoretical Informatics*, pages 88–94, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

- [4] Aaron Bernstein and David Karger. Improved distance sensitivity oracles via random sampling. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, page 34–43, USA, 2008. Society for Industrial and Applied Mathematics.
- [5] Aaron Bernstein and David Karger. A nearly optimal oracle for avoiding failed vertices and edges. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 101–110, New York, NY, USA, 2009. Association for Computing Machinery.
- [6] Greg Bodwin, Fabrizio Grandoni, Merav Parter, and Virginia Vassilevska Williams. Preserving distances in very faulty graphs. *CoRR*, abs/1703.10293, 2017.
- [7] Anat Bremler-Barr, Yehuda Afek, Haim Kaplan, Edith Cohen, and Michael John Merritt. Restoration by path concatenation: fast recovery of MPLS paths. *ACM*, 2001.
- [8] Shiri Chechik and Sarel Cohen. Near optimal algorithms for the single source replacement paths problem. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '19, page 2090–2109, USA, 2019. Society for Industrial and Applied Mathematics.
- [9] Shiri Chechik and Ofer Magen. Near Optimal Algorithm for the Directed Single Source Replacement Paths Problem. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 81:1–81:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [10] Camil Demetrescu and Giuseppe F. Italiano. A new approach to dynamic all pairs shortest paths. *J. ACM*, 51(6):968–992, nov 2004.
- [11] Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM J. Comput.*, 37:1299–1318, 2008.
- [12] Dipan Dey and Manoj Gupta. Near Optimal Algorithm for Fault Tolerant Distance Oracle and Single Source Replacement Path Problem. In Shiri Chechik, Gonzalo Navarro, Eva Rotenberg, and Grzegorz Herman, editors, *30th Annual European Symposium on Algorithms (ESA 2022)*, volume 244 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 42:1–42:18, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [13] Dipan Dey and Manoj Gupta. Nearly optimal fault tolerant distance oracle. In *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*, STOC 2024, page 944–955, New York, NY, USA, 2024. Association for Computing Machinery.
- [14] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
- [15] R. Duan, J. Mao, X. Shu, and L. Yin. A randomized algorithm for single-source shortest path on undirected real-weighted graphs. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 484–492, Los Alamitos, CA, USA, nov 2023. IEEE Computer Society.
- [16] R. Duan, H. Wu, and R. Zhou. Faster matrix multiplication via asymmetric hashing. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2129–2138, Los Alamitos, CA, USA, nov 2023. IEEE Computer Society.

- [17] Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '09, page 506–515, USA, 2009. Society for Industrial and Applied Mathematics.
- [18] Ran Duan and Hanlin Ren. Maintaining exact distances under multiple edge failures. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2022, page 1093–1101, New York, NY, USA, 2022. Association for Computing Machinery.
- [19] Ran Duan and Tianyi Zhang. Improved distance sensitivity oracles via tree partitioning. In Faith Ellen, Antonina Kolokolova, and Jörg-Rüdiger Sack, editors, *Algorithms and Data Structures*, pages 349–360, Cham, 2017. Springer International Publishing.
- [20] Zvi Gotthilf and Moshe Lewenstein. Improved algorithms for the k simple shortest paths and the replacement paths problems. *Information Processing Letters*, 109(7):352–355, 2009.
- [21] Fabrizio Grandoni and Virginia Vassilevska Williams. Improved distance sensitivity oracles via fast single-source replacement paths. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*, pages 748–757, 2012.
- [22] Fabrizio Grandoni and Virginia Vassilevska Williams. Faster replacement paths and distance sensitivity oracles. *ACM Trans. Algorithms*, 16(1), dec 2019.
- [23] Yong Gu and Hanlin Ren. Constructing a Distance Sensitivity Oracle in $O(n^{2.5794}M)$ Time. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 76:1–76:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [24] Yuzhou Gu, Adam Polak, Virginia Vassilevska Williams, and Yinzhan Xu. Faster Monotone Min-Plus Product, Range Mode, and Single Source Replacement Paths. In Nikhil Bansal, Emanuela Merelli, and James Worrell, editors, *48th International Colloquium on Automata, Languages, and Programming (ICALP 2021)*, volume 198 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 75:1–75:20, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [25] Enrico Nardelli, Guido Proietti, and Peter Widmayer. A faster computation of the most vital edge of a shortest path. *Information Processing Letters*, 79(2):81–85, 2001.
- [26] Binghui Peng and Aviad Rubinfeld. Fully-dynamic-to-incremental reductions with known deletion order (e.g. sliding window). In *Symposium on Simplicity in Algorithms (SOSA)*, pages 261–271. SIAM, 2023.
- [27] Hanlin Ren. Improved distance sensitivity oracles with subcubic preprocessing time. *Journal of Computer and System Sciences*, 123:159–170, 2022.
- [28] Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. New bounds for matrix multiplication: from alpha to omega. In *Proceedings of SODA'24*, pages 3792–3835, 01 2024.
- [29] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the Forty-Sixth Annual ACM Symposium on Theory of Computing*, STOC '14, page 664–673, New York, NY, USA, 2014. Association for Computing Machinery.

- [30] V. Williams, E. Woldeghebriel, and Y. Xu. Algorithms and lower bounds for replacement paths under multiple edge failure. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 907–918, Los Alamitos, CA, USA, nov 2022. IEEE Computer Society.
- [31] Virginia Vassilevska Williams and R. Ryan Williams. Subcubic equivalences between path, matrix, and triangle problems. *J. ACM*, 65(5), aug 2018.