Succinct Graph Structures and Their Applications **Spring 2018**

Lecture 11: June 27

Lecturer: Merav Parter

Distance Sensitivity Oracles

A *distance oracle scheme* (Lecture 2) consists of two algorithms: a *preprocessing* algorithm, that given a graph G outputs a (small space) data structure; and a *query* algorithm that given a distance query $\langle s,t\rangle$, uses only the data structure (with no access to G) to report the distance between s and t in G. This setting has three complexity measures: the preprocessing time, the size of the data structure and the query time. In this class, we put emphasis on the latter two measures and the tradeoff between them.

In distance *sensitivity* oracles, the query consists of three vertices: source s, sink t and a failed vertex u. The output of the query algorithm is $\texttt{dist}(s,t,G\setminus\{u\})$. There are two naive schemes. One extreme is to keep in the data structure, all $O(n^3)$ possible distances explicitly, i.e., for any triplet $\langle s,t,u\rangle$, keep $\texttt{dist}(s,t,G\setminus\{u\})$. This gives space $O(n^3\log n)$ and constant query time. On the other extreme, one can keep the entire graph G, and given a query $\langle s,t,u\rangle$, apply Dijkstra in G. This gives an optimal space of $O(m)$ but quite unsatisfactory query time of $O(m)$. Today, we will see a super elegant construction that enjoys both small space of $O(n^2\log^3 n)$ and constant query time. Up to logarithmic factors, this construction is optimal.

Theorem 11.1 ([DTCR08]) *For any n-vertex graph $G=(V,E)$, there exists a construction of a distance sensitivity oracles in time $O(m\cdot n^2+n^{\cdot}\log n)$, space of $O(n^2\log^3 n)$ and constant query time.*

We will use the following fact about data structure for LCA (Lowest Common Ancestor).

fact 11.2 *[HT84] For any n-vertex tree T, there exists an $O(n)$ computable data structure which occupies $O(n)$-space and answers LCA queries ("what's the LCA of s and t in T?") in $O(1)$ time.*

Description of the oracle: A succinct and an elegant explanation of the construction appears in [DP09]. For simplicity, we consider unweighted graphs, although the construction holds for weighted graphs as well. For every $r\in V$, let T_r be a BFS tree rooted at r. The oracle $\mathcal{O}$ contains the LCA data structure of T_r for every $r\in V$. Let $\pi(x,y)$ be the $x-y$ shortest path in G. We assume uniqueness of shortest paths by breaking shortest path ties in a consistent manner. In addition, the oracle $\mathcal{O}$ contains the following:

- (1) A distance matrix B_0 where $B_0(x,y)=\texttt{dist}(x,y,G)$.
- (2) A collection of distance lists B_1 where $B_1(x,y)$ contains for every $x,y\in V$:
  - (A) For $i\in\{0,1,\ldots,\lfloor\log\texttt{dist}(x,y,G)\rfloor\}$:
    - Let u_i (resp., v_i) be the vertex at distance 2^i from x (resp., y) on $\pi(x,y)$.
    - Keep $B_1(x,y,w)=\texttt{dist}(x,y,G\setminus\{w\})$ for $w\in\{u_i,v_i\}$.
  - (B) For every $i,j\in\{0,1,\ldots,\lfloor\log\texttt{dist}(x,y,G)\rfloor\}$:
    - Keep $B_1(x,y,u_i,v_j)=\texttt{dist}(x,y,G\setminus V(\pi(u_i,v_j)))$.

It is easy to see that the total space of $\mathcal{O}$ is indeed bounded by $O(n^2\log^3 n)$ (mostly due to step (B)). The main challenge is in showing that the stored information is sufficient. In particular, note that we consider only $O(\log n)$ special vertices on each shortest path $\pi(x,y)$, even though there might be $\Omega(n)$ vertices that might fail on that path. The magic comes from the structure of replacement paths that avoids a single failed vertex. Last week, in the construction of FT-BFS, we mainly used the structure of *new-ending* replacement paths, i.e., paths whose last edge is not the edge of the shortest path in G. Here, we will use the general structure of any replacement path.

Why does it work? some intuition. Consider a pair $x, y \in V$ and a failed vertex u. Recall from last class that the replacement path $P_{x,y,u}$ consists of a detour – which bypasses the failing vertex on the x-y shortest path $\pi(x, y)$. We need the following definition. Let u_ℓ be the closest vertex to x on the $\pi(x, y)$ path such that $|\pi(u, u_\ell)|$ is a power of 2. In the same manner, let u_r be the closest vertex to y, such that $|\pi(u, u_r)|$ is a power of 2. There are three possibilities for the detour of $P_{x,y,u}$. Option (1) is where the detour goes through u_ℓ, option (2) is where the detour passes through u_r, and option (3) is where the detour does not go through u_ℓ nor u_r. In such a case, by the uniqueness of shortest paths, it must hold that the detour avoids all the vertices on the segment $\pi(u_\ell, u_r)$. See Fig. 11.1. By the triangle inequality, it always holds that:

$$\mathtt{dist}(x, y, G \setminus \{u\}) \le \mathtt{dist}(x, u_\ell, G \setminus \{u\}) + \mathtt{dist}(u_\ell, u_r, G \setminus \{u\}) + \mathtt{dist}(u_r, y, G \setminus \{u\}).$$

By the above possible configurations, we get that:

$$\mathtt{dist}(x, y, G \setminus \{u\}) = \min\{|\pi(x, u_\ell)| + \mathtt{dist}(u_\ell, y, G \setminus \{u\}), \mathtt{dist}(x, u_r, G \setminus \{u\}) + |\pi(u_r, y)|, \mathtt{dist}(x, y, G \setminus \pi(u_\ell, u_r))\}\ .$$

Note that besides the term $\mathtt{dist}(x, y, G \setminus \pi(u_\ell, u_r))$, all other distances in the above equation are explicitly kept in our data structure. That is, since u_ℓ and u_r are within power of 2 distance from u (and not from x and y respectively), we did not keep the distance $\mathtt{dist}(x, y, G \setminus \pi(u_\ell, u_r))$ explicitly. We will see how to handle it soon.

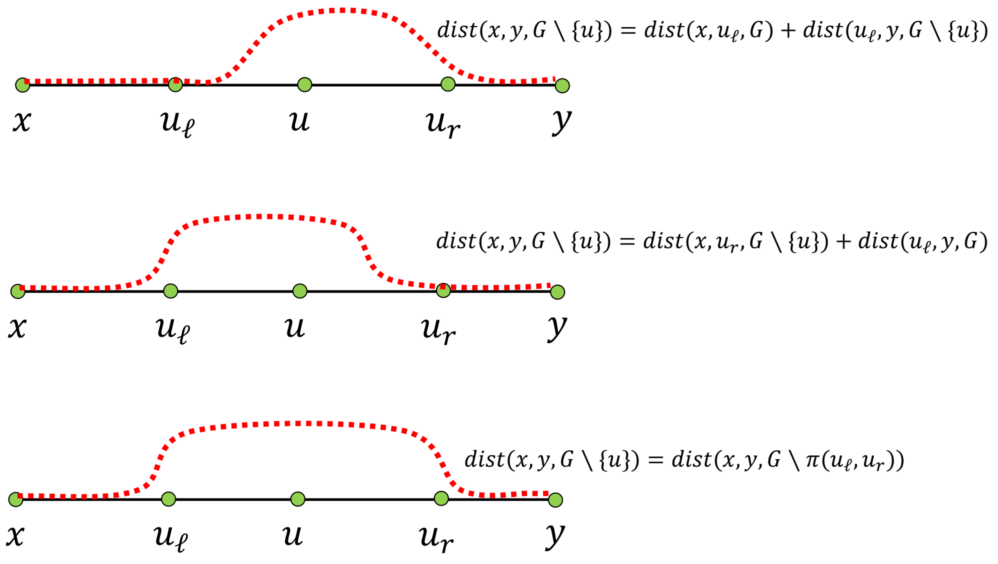

$dist(x, y, G \setminus \{u\}) = dist(x, u_\ell, G) + dist(u_\ell, y, G \setminus \{u\})$
x
u_ℓ
u
u_r
y
$dist(x, y, G \setminus \{u\}) = dist(x, u_r, G \setminus \{u\}) + dist(u_\ell, y, G)$
x
u_ℓ
u
u_r
y
$dist(x, y, G \setminus \{u\}) = dist(x, y, G \setminus \pi(u_\ell, u_r))$
x
u_ℓ
u
u_r
y

Figure 11.1: The possible configurations of a single fault detour along with the corresponding distances.

The query algorithm: Given a triplet $\langle x, y, u\rangle$, we do as follows. First, to see if $u \in \pi(x, y)$, we apply an LCA query with $\langle u, y\rangle$ in T_x. If u is the LCA of y in T_x, then we deduce that $u \in \pi(x, y)$. If u is not the LCA, then we simply return $B_0(x, y) = \mathtt{dist}(x, y, G)$. From now on, assume that $u \in \pi(x, y)$. Let $W = \{u_i, v_i \mid i \in \{0, 1, \ldots \lfloor\log(\mathtt{dist}(x, y, G))\rfloor\}\}$ be all vertices at distance 2^i from either x or y on $\pi(x, y)$.

If $u \in W$, then return $B_1(x, y, u) = \mathtt{dist}(x, y, G \setminus \{u\})$. Note that we keep a lookup table for the B_1 distances with the key x, y, w for every $w \in W$, and hence we can check in $O(1)$ time if $u \in W$ and in such a case return the corresponding distance.

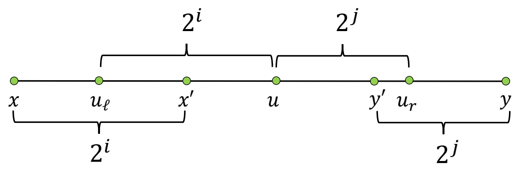
We finally get to the interesting case where $u \notin W$ and define u_ℓ and u_r as above. Since we know $\mathtt{dist}(x, y, G)$, we can compute in $O(1)$ time the vertices u_ℓ and u_r. Observe that since $u \notin W$ (i.e., its distance is not a power of two from neither x nor y), we get that $u_\ell \neq u$ and $u_r \neq y$. Let $\texttt{dist}(u, u_\ell, G) = 2^i$ and $\texttt{dist}(u, u_r, G) = 2^j$. Define x' at the vertex at distance 2^i from x on $\pi(x, y)$. In the same manner, define y' as the vertex at distance 2^j from y on $\pi(x, y)$. See Fig. for an illustration. Since x' and y' are in power of 2 distance from x and y respectively, we have that $x', y' \in W$. Thus, their computation can be done in $O(1)$ time using the lookup table of $B_1(x, y, w)$ for $w \in W$. The query algorithm returns the answer:

$$\min\{B_0(x, u_\ell) + B_1(y, u_\ell, u), B_1(x, u_r, u) + B_0(y, u_r), B_1(x, y, x', y')\}.$$

To prove the correctness of the query algorithm, it is sufficient to show:

Claim 11.3 *$x' \in \pi(u_\ell, u)$ and $y' \in \pi(u_r, u)$ and thus $u \in \pi(x', y')$.*

Proof: By the definition of u_ℓ, $|\pi(x, u_\ell)| \leq 2^i$ and thus $x' \in \pi(u_\ell, u)$. Similarly, by the definition of u_r, it holds that $|\pi(y, u_r)| \leq 2^j$ and thus $y' \in \pi(u, u_r)$. ■



Fault Tolerant Spanners

For positive integers $f, t \geq 1$, a subgraph $H \subseteq G$ is an f-edge t-Fault-Tolerant (FT) spanner if

$$\texttt{dist}(u, v, H \setminus F) \leq t \cdot \texttt{dist}(u, v, G \setminus F)$$

for every $u, v \in V$ and for every $F \subseteq E$, $|F| \leq f$. The vertex variant is defined analogously only with $F \subseteq V$. Note that this definition does not require G to be $(f+1)$-vertex connected[1]. In particular, if there is a sequence of f faults F which disconnects u and v in G, then the u-v distance in both $G \setminus F$ and $H \setminus F$ is infinity. However, as long as u and v are connected in $G \setminus F$, it is required that the u-v distance in $H \setminus F$ is a factor t approximation to the distance in $G \setminus F$. This notion of fault tolerance is known as *competitive*, see [CP10].

Before diving into the constructions of sparse fault tolerant spanners, lets try to compare them to the other fault tolerant structure we saw last class, namely to fault tolerant BFS structures.

Comparison to FT-BFS: On the one hand, it seems that the requirement of FT-spanners is stronger than that of FT-BFS: the latter only concerns the sourcewise distances $\{s\} \times V$ while the former concerns with all the pairs $V \times V$. On the other hand, FT-BFS insists on preserving the *exact* distance, while FT-spanners settle for an approximation. As we will see today, the insistence on exact distance plays the key role here which makes f FT-BFS significantly denser than f-edge FT-spanners for stretch values $t \geq 5$.

[1]A graph G is k vertex ocnnected, in the removal of any $k-1$ vertices does not disconnect G.

FT-spanners for edge failures [CLPR10]: In the setting of FT-spanners, handling edge failures is easier than vertex failures. We will present an elegant construction due to Chechik et al. [CLPR10]. This was the first work which studied the construction of sparse fault tolerant subgraphs for *general* subgraph.

Theorem 11.4 *For every $f \geq 1$ and $k \geq 1$, there exists an f-edge $(2k-1)$ FT-spanner $H \subseteq G$ with $O(f \cdot n^{1+1/k})$.*

Recall that standard $(2k-1)$ spanners have size of $O(n^{1+1/k})$ edges (First Lecture!), thus the cost of making these structures robust against f *edge* faults is linear in f!.

The algorithm of [CLPR10] is as follows:

```
Algorithm EFTSpanner($G, f, k$)
  1. Set $G' \leftarrow G$ and $H \leftarrow \emptyset$.
  2. For $i = 1, \ldots, f+1$:
       • $H_i \leftarrow$ Spanner($G', 2k-1$).
       • $H \leftarrow H \cup H_i$
       • $G' \leftarrow G' \setminus H_i$.
```

Figure 11.2: Algorithm for computing an f-edge $(2k-1)$ FT-spanners

The size of H is immediate by construction: Spanner algorithm computes a $(2k-1)$ spanner with $O(n^{1+1/k})$ edges and H contains $O(f)$ edge disjoint $(2k-1)$ spanners. For the stretch analysis, it is sufficient to show:

Claim 11.5 $\texttt{dist}(u, v, H \setminus F) \leq 2k-1$ *for every $(u,v) \in E$ and $F \subseteq E$, $|F| \leq f$ and $(u,v) \notin F$.*

Proof: If $(u,v) \in H$ then the claim holds trivially. Otherwise, since $e = (u,v)$ was not added to any of the $f+1$ spanners, it implies that H contains $f+1$ edge disjoint u-v paths of length at most $2k-1$. Since the set F contains at most f edges, one of these paths survive the failing, i.e., H contains a u-v path of length at most $2k-1$ that survives the failing of F. ■

FT-spanners for vertex failures: We now turn to discuss fault tolerant spanners that are resilient against vertex faults. Fixing the number of faults to f and the stretch value to $(2k-1)$, [CLPR10] showed a construction of f-vertex $(2k-1)$ FT-spanners with $\widetilde{O}(f^2 \cdot k^{f+1} n^{1+1/k})$ edges. This bound becomes $\widetilde{O}(n^{1+1/k})$ when both the number of faults and the stretch are constants. Dinitz and krauthgamer [DK11] improved this bound to $\widetilde{O}(f^{2-1/k} \cdot n^{1+1/k})$ edges. Finally, recently Bodwin et al. [BDPW18] provided a construction with $\widetilde{O}(f^{1-1/k} \cdot n^{1+1/k})$ for $k = O(1)$. Assuming the girth conjecture, this bound is also tight.

We will now describe the algorithm by Dinitz and krauthgamer [DK11]. The high level idea is to run several independent *experiments*. In each experiment (iteration) i, we sample many vertices into the *failing set* F_i and compute a $(2k-1)$ spanner in the remaining graph $G \setminus F_i$. Note that even though the number of faults that the spanner should handle is at most f, in each experiment we fail $(1-1/f)$ fraction of the vertices!

Claim 11.6 *H has $\widetilde{O}(f^{2-1/k} n^{1+1/k})$ edges.*

Proof: There are $O(f^3 \log n)$ iterations, in each we compute a $(2k-1)$ spanner for a graph that contains $O(n/f)$ vertices. Thus, each H_i has $O((n/f)^{1+1/k})$ edges, and the final spanner has $O(f^{2-1/k} \cdot \log n \cdot n^{1/k})$ edges, as required. ■

We now turn to show correctness and as usual consider a neighboring pair u, v and a set F of the most f vertices. The end goal is to show that $\texttt{dist}(u, v, H \setminus F) \leq 2k-1$ for every $(u,v) \in E$ and $F \subseteq V$. The analysis is based on the following definition. An experiment (or iteration) i is *good* for the triplet $\langle u, v, F \rangle$ if $u, v \notin F_i$ and $F \subseteq F_i$. We claim:

```
Algorithm VFTSpanner$(G, f, k)$
    1. Set $H \leftarrow \emptyset$.
    2. For $i = 1, \ldots, O(f^3 \log n)$:
        • Sample each $v$ into $F_i$ with probability $p$, where
              $p = \begin{cases} 1 - 1/f \text{ for } f \geq 2 \\ 1/2 \text{ for } f = 1 \end{cases}$
        • $H_i = \mathsf{Spanner}(G \setminus F_i, 2k-1)$.
        • $H \leftarrow H \cup H_i$.
```

Figure 11.3: Algorithm for computing an f-vertex $(2k-1)$ FT-spanners

Claim 11.7 *With probability of* $1 - 1/n^{cf}$ *for some constant c,* $\langle u, v, F \rangle$ *has a good experiment.*

Proof: Fix an experiment i and a triplet $\langle u, v, F \rangle$. The probability that i is good for this triplet is at least $(1-p)^2 \cdot p^f = 1/f^2 \cdot 1/e \leq 1/(2f^2)$. Since all experiments are independent of each other, the probability that *no* experiment is good is at most $(1 - 1/(2f^2))^{c' \cdot f^3 \log n} = 1/n^{\Theta(f)}$. The claim follows. ∎

We are now ready to complete the stretch argument.

Claim 11.8 $\mathtt{dist}(u, v, H \setminus F) \leq 2k - 1$ *for every* $(u, v) \in E$ *and* $F \subseteq V$.

Proof: By applying the union bound over all $O(n^{f+2})$ triplets $\langle u, v, F \rangle$, by Cl. 11.7, with probability $1 - 1/n^{c'}$, every triplet has a good experiment. Fix a triplet $\langle u, v, F \rangle$ and let i be such a good experiment for this triplet. Then H_i either contains (u, v) and we are done, or else it contains an u-v path P of length at most $(2k-1)$. Since $F \subseteq F_i$ and $H_i \subseteq G \setminus F_i$, that path P is free from any other failing vertices and thus: $\mathtt{dist}(u, v, H \setminus F) \leq \mathtt{dist}(u, v, H_i \setminus F) \leq 2k - 1$. The claim follows. ∎

(Conditional) lower bound for f-vertex $(2k-1)$ FT-spanners.

Lemma 11.9 ([BDPW18]) *Assuming Erdős' girth conjecture, for every* $k \geq 1, f \geq 1$*, there is an n-vertex graph* $G_{k,f}$ *such that any f-vertex* $(2k-1)$ *FT-spanner* $H \subseteq G_{k,f}$ *must have* $\Omega(f^{1-1/k} \cdot n^{1+1/k})$ *edges.*

Proof: Erdős' girth conjecture (see Lecture 1) states that for every $k \geq 1$ and sufficiently large n, there are n-vertex graphs G_k with $\Omega(n^{1+1/k})$ and girth[2] at least $2k+2$. The graph $G_{k,f}$ is obtained from G_k by the following operations: Each vertex u is G_k is multiplied by $f' = \lfloor f/2 \rfloor$ copies $u_1, \ldots, u_{f'}$ in $G_{k,f}$. Each edge $(u, v) \in G_k$ is replaced by a complete bipartite graph between all copies of u and all copies of v. Formally, $V(G_{k,f}) = \{u_1, \ldots, u_{f'} \mid u \in V\}$ and $E(G_{k,f}) = \{(u_i, v_j) \mid (u, v) \in E(G_k), i, j \in [1, f']\}$. The number of vertices in $G_{k,f}$ is $N = O(f \cdot n)$ and the number of edges is $O(f^2 \cdot n^{1+1/k}) = O(f^{1-1/k} N^{1+1/k})$ edges. We next show that removing any edge e from $G_{k,f}$ results in a graph $H = G_{k,f} \setminus \{e\}$ which is *not* a legal f-vertex $(2k-1)$ FT-spanner for $G_{k,f}$. This would imply that the only f-vertex $(2k-1)$ FT-spanner for $G_{k,f}$ is $G_{k,f}$, thus contains $\Omega(f^{1-1/k} N^{1+1/k})$ many edges. Let $e = (u_i, v_j)$ and let

$$F = (\{u_1, \ldots, u_{f'}\} \cup \{v_1, \ldots, v_{f'}\}) \setminus \{u_i, v_j\}.$$

Observe that $|F| \leq f$. Thus in $G_{k,f} \setminus F$, the edge $e = (u_i, v_j)$ is the only edge connecting u's copies with v's copies. Since the girth of G_k is at least $2k+2$, removing e increases the distance from u_i to v_j in $G_{k,f}$ from 1 to at least $2k+1$. We get that $\mathtt{dist}(u_i, v_j, G_{k,f} \setminus F) = 1$ but $\mathtt{dist}(u_i, v_j, H \setminus F) \geq 2k+1$. The claim follows. ∎

[2]Length of shortest cycle in the graph.

References

[BDPW18] Greg Bodwin, Michael Dinitz, Merav Parter, and Virginia Vassilevska Williams. Optimal vertex fault tolerant spanners (for fixed stretch). In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1884–1900. Society for Industrial and Applied Mathematics, 2018.

[CLPR10] Shiri Chechik, Michael Langberg, David Peleg, and Liam Roditty. Fault tolerant spanners for general graphs. *SIAM Journal on Computing*, 39(7):3403–3423, 2010.

[CP10] Shiri Chechik and David Peleg. Rigid and competitive fault tolerance for logical information structures in networks. In *Electrical and Electronics Engineers in Israel (IEEEI), 2010 IEEE 26th Convention of*, pages 000024–000025. IEEE, 2010.

[DK11] Michael Dinitz and Robert Krauthgamer. Fault-tolerant spanners: better and simpler. In *Proceedings of the 30th annual ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, pages 169–178. ACM, 2011.

[DP09] Ran Duan and Seth Pettie. Dual-failure distance and connectivity oracles. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 506–515. Society for Industrial and Applied Mathematics, 2009.

[DTCR08] Camil Demetrescu, Mikkel Thorup, Rezaul Alam Chowdhury, and Vijaya Ramachandran. Oracles for distances avoiding a failed node or link. *SIAM Journal on Computing*, 37(5):1299–1318, 2008.

[HT84] Dov Harel and Robert Endre Tarjan. Fast algorithms for finding nearest common ancestors. *siam Journal on Computing*, 13(2):338–355, 1984.