Fast Approximation Algorithms for the Diameter and Radius of Sparse Graphs

Preksha Mishra: 2022ES11849
Dharavath Ashvitha: 2022CS11132

Abstract

The diameter and radius of a graph are fundamental structural measures with wide applications in network analysis, communication optimization, and facility location. Computing them exactly requires solving the All-Pairs Shortest Paths (APSP) problem, which takes $\Theta(mn)$ time on sparse graphs. This work reviews and analyzes a sequence of increasingly efficient approximation algorithms, beginning with the 2-approximation from double BFS, the classical (3/2)-approximation by Aingworth et al., and the breakthrough randomized $\tilde{O}(m\sqrt{n})$ algorithm by Roditty and Vassilevska Williams. These approaches demonstrate how random sampling and combinatorial neighborhood estimation can drastically reduce running time without sacrificing approximation quality. The report also discusses conditional lower bounds under the Strong Exponential Time Hypothesis (SETH), showing that the (3/2)-approximation factor is likely optimal for sparse graphs.

1 Introduction

Let $G = (V, E)$ be an unweighted graph. For vertices $u, v \in V$, let $d(u, v)$ denote the shortest-path distance between them. The *eccentricity* of a vertex v is defined as

$$\mathrm{ecc}(v) = \max_{u \in V} d(v, u).$$

The *diameter* of the graph is the maximum eccentricity,

$$D = \max_{v \in V} \mathrm{ecc}(v),$$

and the *radius* is the minimum eccentricity,

$$r = \min_{v \in V} \mathrm{ecc}(v).$$

These parameters arise in classical and contemporary applications, including facility location, communication latency minimization, and structural analysis of large-scale networks. Computing D and r exactly requires solving the All-Pairs Shortest Paths (APSP) problem. In unweighted sparse graphs, performing a BFS from every vertex yields a running time of $\Theta(mn)$, where $n = |V|$ and $m = |E|$. For sparse graphs with $m = \Theta(n)$, this gives $\Theta(n^2)$ time, which is prohibitive for large networks.

This motivates the central question:

Can diameter and radius be approximated strictly faster than $\Theta(mn)$?

A trivial linear-time procedure gives a 2-approximation: perform BFS from an arbitrary vertex s to find a vertex t farthest from it, then BFS from t. This yields a value $\hat{D}$ satisfying $\frac{D}{2} \leq \hat{D} \leq D$. While efficient, the factor-2 bound is often too coarse.

Aingworth, Chekuri, Indyk, and Motwani (SODA'96; SICOMP'99) achieved a stronger result: a (3/2)-approximation in time $\tilde{O}(m\sqrt{n} + n^2)$. Their method introduced the idea of bounded neighborhoods and hitting sets. Fifteen years later, Roditty and Vassilevska Williams (STOC 2013) presented a randomized improvement achieving $\tilde{O}(m\sqrt{n})$ expected time, eliminating the n^2 term.

Moreover, they derived similar bounds for radius and eccentricity approximations, and proved under the Strong Exponential Time Hypothesis (SETH) that further improvement beyond $(3/2 - \varepsilon)$ in subquadratic time is impossible.

2 Methodology and Algorithms

2.1 BFS-Based APSP (Exact)

Problem Statement

Given an unweighted graph $G = (V, E)$ (directed or undirected), compute *all-pairs shortest path* distances

$$\mathrm{dist}(u, v) \quad \forall\, u, v \in V$$

exactly, by using breadth-first search (BFS). The expected (and worst-case) running time is

$$\boxed{O(mn)} \quad \text{(equivalently } O(n(m+n))\text{)},$$

where $n = |V|$ and $m = |E|$.

Explanation

In unweighted graphs, BFS from a single source s discovers vertices in nondecreasing order of their shortest-path distance from s. The BFS tree levels correspond exactly to distance layers: the k-th layer contains precisely those vertices at distance k from s. To obtain all-pairs distances, we simply run BFS once from each vertex $s \in V$, recording the distance labels it assigns to every vertex. This yields an $n \times n$ distance matrix with exact shortest-path lengths.

Pseudo Code

```
Input: $G = (V, E)$.
Initialize: $D \in (\mathbb{Z}_{\geq 0} \cup \{\infty\})^{n \times n}$,   $D[u, v] \leftarrow \infty,\ \forall u, v \in V$;   $D[v, v] \leftarrow 0,\ \forall v$.
For each $s \in V$ :
    Initialize queue $Q$, mark all vertices unvisited; vis$[s] \leftarrow$ true.
    For all $v \in V$ : $\text{dist}_s(v) \leftarrow \infty$; $\text{dist}_s(s) \leftarrow 0$.
    $Q$.push$(s)$.
    While $Q \neq \emptyset$ :
        $u \leftarrow Q$.pop();
        For each $(u, x) \in E$ :
            if $\text{dist}_s(x) = \infty$ then $\text{dist}_s(x) \leftarrow \text{dist}_s(u) + 1$, $Q$.push$(x)$.
    For all $v \in V$ : $D[s, v] \leftarrow \text{dist}_s(v)$.
Output: $D$ (APSP distance matrix).
```

Proof of Correctness

For a fixed source s, BFS explores vertices by increasing number of edges from s. Let L_k be the k-th BFS layer from s. By induction on k: (i) every $v \in L_k$ has a path of length k from s (constructed by the tree edges), hence $\text{dist}(s, v) \leq k$; and (ii) no vertex is discovered earlier than its shortest-path length, since discovery requires an incident edge from some L_{k-1} vertex, implying any discovered v has a path of length k but none shorter. Thus $\text{dist}_s(v)$ returned by BFS equals $\text{dist}(s, v)$. Running BFS for every $s \in V$ fills D with exact all-pairs distances.

Running Time Analysis

Each BFS runs in $O(m + n)$ time. Performing BFS from every source yields

$$\sum_{s \in V} O(m + n) \;=\; O\big(n(m + n)\big) \;=\; O(mn + n^2).$$

In sparse graphs ($m = O(n)$), this is $\Theta(n^2)$. Space to store D is $O(n^2)$.

2.2 2-Approximation via Double BFS

Problem Statement

Approximate the diameter D of an unweighted graph $G = (V, E)$ within a factor of 2 using two BFS traversals:

$$\frac{D}{2} \;\leq\; \hat{D} \;\leq\; D,$$

in overall time

$$\boxed{O(m + n)}.$$

Explanation

In unweighted graphs, a breadth-first search (BFS) from a source s discovers vertices in nondecreasing order of their distance from s. If we select any starting vertex s and run BFS, some vertex a at maximum distance from s (a "peripheral" vertex relative to s) is found. Intuitively, a lies near an endpoint of a longest shortest path. Running a second BFS from a reveals distances from a to all vertices; the maximum of these distances is our estimate $\hat{D}$. This value cannot exceed the true diameter, and it is guaranteed to be at least half of it. The procedure is extremely fast—just two BFS runs—and often gives tight results in practice (exact for trees).

Pseudo Code

```
Input: $G = (V, E)$.
Pick any $s \in V$.
Run BFS($s$) $\Rightarrow \mathrm{dist}_s(\cdot)$.
$a = \arg\max_{v \in V} \mathrm{dist}_s(v)$.
Run BFS($a$) $\Rightarrow \mathrm{dist}_a(\cdot)$.
$\hat{D} = \max_{v \in V} \mathrm{dist}_a(v)$.
Output: $\hat{D}$.
```

Proof of Correctness

Let $x, y \in V$ be a pair with $d(x, y) = D$. For any $s \in V$, at least one of x or y has distance $\geq D/2$ from s; otherwise the triangle inequality would give $d(x, y) < D$. Thus $\max_v d(s, v) \geq D/2$. Since a is chosen to maximize $d(s, \cdot)$, we have $d(s, a) \geq D/2$. The second BFS yields $\hat{D} = \max_v d(a, v) \leq D$ by definition of diameter, and $\hat{D} \geq d(a, \cdot) \geq D/2$ by the previous argument. Therefore $D/2 \leq \hat{D} \leq D$.

Running Time Analysis

Each BFS takes $O(m + n)$ time; the algorithm runs exactly two BFS traversals. Hence the total time is $O(m + n)$ and the extra space is $O(n)$.

2.3 Aingworth et al. Diameter Approximation Algorithm

Problem Statement

Compute an estimate $\hat{D}$ such that:

$$\left\lfloor \frac{2D}{3} \right\rfloor \;\leq\; \hat{D} \;\leq\; D,$$

where D is the true diameter of the graph. The deterministic running time is:

$$\boxed{\tilde{O}(m\sqrt{n} + n^2)}.$$

Explanation

The algorithm is based on identifying "representative" vertices that collectively cover all large-distance neighborhoods in the graph. For a parameter s, let $N_s(v)$ denote the set of the s closest vertices to v in a BFS from v. If we could select a small set H of vertices such that H intersects $N_s(v)$ for every vertex v, then BFS from all vertices in H would be sufficient to approximate the diameter.

The key is to construct such a set H, called a **hitting set**. To do this deterministically, we: 1. Compute $N_s(v)$ for every v. 2. View the collection $\{N_s(v) : v \in V\}$ as a set system. 3. Apply a greedy set-cover procedure to select H of size $O(n/s)$.

Then, we run BFS from every vertex in H. To refine the estimate further, we identify a vertex w that is farthest from H, compute $N_s(w)$, and also run BFS from all vertices in $N_s(w)$. The largest BFS depth observed is the diameter estimate $\hat{D}$.

Pseudo Code

```
**Input:** $G = (V, E),\ s = \Theta(\sqrt{n})$.
For each $v \in V$ : run BFS($v$) until $s$ vertices are reached; $N_s(v)$ = first $s$ vertices.
$\mathcal{S} = \{N_s(v) : v \in V\}$.
$H$ = greedy hitting set for $\mathcal{S}$, $|H| = O(n/s)$.
For each $u \in H$ : run BFS($u$).
$w = \arg\max_{v \in V} \min_{u \in H} d(v, u)$.
Run BFS($w$) to obtain $N_s(w)$.
For each $x \in N_s(w)$ : run BFS($x$).
$\hat{D} = \max_{\text{BFS roots } z} \max_{v \in V} d(z, v)$.
**Output:** $\hat{D}$.
```

Proof of Correctness

For every $v \in V$, H intersects $N_s(v)$ by construction of the hitting set. Thus there exists $h \in H$ such that $d(v, h) \leq$ radius of $N_s(v)$, which is at most $\frac{D}{3}$ for appropriately chosen s. This implies that BFS from h reaches all vertices within distance at most $D - \frac{D}{3}$ from the true diameter endpoints. Similarly, the BFS from $N_s(w)$ covers the complementary side of any longest shortest-path. Therefore, the maximum BFS depth observed lies between $\frac{2D}{3}$ and D.

Running Time Analysis

- Computing all $N_s(v)$ sets costs $O(ns^2)$.
- Greedy hitting set construction on n sets of size s costs $O(ns)$.
- $|H| = O(n/s)$ BFS calls cost $O((n/s) \cdot m)$.
- BFS calls from $N_s(w)$ cost $O(s \cdot m)$.

Setting $s = \Theta(\sqrt{n})$ gives:

$$O(ns^2) + O((n/s)m) + O(sm) = \tilde{O}(n^2 + m\sqrt{n}).$$

Thus the total running time is:

$$\boxed{\tilde{O}(m\sqrt{n} + n^2)}.$$

2.4 Randomized Diameter Approximation Algorithm (Main Result)

Problem Statement

Approximate the diameter D of an undirected, unweighted graph $G = (V, E)$ such that:

$$\left\lfloor \frac{2D}{3} \right\rfloor \leq \hat{D} \leq D,$$

in expected time:

$$\boxed{\tilde{O}(m\sqrt{n})}.$$

Explanation

This algorithm improves the running time of Aingworth et al. by avoiding the explicit computation of all $N_s(v)$ sets and the deterministic hitting set. Instead, we replace these with a *random sample* S of size $\Theta(\sqrt{n}\log n)$, which acts as an approximate hitting set with high probability.

We run BFS from all vertices in S to obtain distance information. Then, we select a vertex w that is farthest from S; this often lies near one endpoint of a diameter. We run a BFS from w, and consider the first $\sqrt{n}$ vertices encountered in increasing BFS distance from w. These vertices form a set $N_s(w)$ which approximates the growth behavior of BFS layers. Running BFS from all vertices in $S \cup N_s(w)$ is enough to capture one endpoint of a longest shortest-path with high probability. The maximum BFS depth observed among these searches is returned as the diameter estimate.

Pseudo Code

```
Input: $G = (V, E),\ s = \Theta(\sqrt{n})$.
Choose random sample $S \subseteq V,\ |S| = \Theta(\sqrt{n}\log n)$.
For each $q \in S:$ run BFS($q$).
$p_S(v) = \arg\min_{q \in S} d(v, q), \quad \forall v \in V$.
$w = \arg\max_{v \in V} d(v, p_S(v))$.
Run BFS($w$) and let $N_s(w)$ be the first $s$ vertices visited.
For each $x \in N_s(w):$ run BFS($x$).
$\hat{D} = \max_{z \in S \cup N_s(w)} \max_{v \in V} d(z, v)$.
Output: $\hat{D}$.
```

Proof of Correctness

With high probability, the random sample S intersects a $(2D/3)$-radius neighborhood around at least one diameter endpoint. Then w lies near the opposite endpoint of a longest shortest-path. The first $s = \Theta(\sqrt{n})$ layers of BFS(w) include a vertex close to that endpoint, ensuring that BFS from some $x \in N_s(w)$ reaches the full diameter length. Thus:

$$\frac{2D}{3} \le \hat{D} \le D.$$

Running Time Analysis

- $|S| = \Theta(\sqrt{n})$ BFS calls: $O(m\sqrt{n})$.
- BFS(w): $O(m)$.
- $|N_s(w)| = \Theta(\sqrt{n})$ BFS calls: $O(m\sqrt{n})$.

Hence the total expected time is:

$$\boxed{\tilde{O}(m\sqrt{n})}.$$

2.5 Deterministic Diameter Approximation via Hitting Set

Problem Statement

Compute an estimate $\hat{D}$ satisfying:

$$\left\lfloor \frac{2D}{3} \right\rfloor \le \hat{D} \le D$$

deterministically, without random sampling. The running time is:

$$\boxed{\tilde{O}(m\sqrt{n} + n^2)}.$$

Explanation

The randomized algorithm uses a random sample S of size $\Theta(\sqrt{n}\log n)$ which acts, with high probability, like a *hitting set* that intersects neighborhoods $N_s(v)$ for all vertices v. To make the algorithm deterministic, we must explicitly construct such a hitting set.

For a parameter $s = \Theta(\sqrt{n})$, define:

$$N_s(v) = \text{the first } s \text{ vertices discovered in BFS}(v).$$

These sets represent small neighborhoods around each vertex. We compute $N_s(v)$ for every $v \in V$ and then apply a greedy set-cover procedure to obtain a hitting set H of size $O(n/s)$ such that:

$$H \cap N_s(v) \neq \emptyset \quad \forall v \in V.$$

We then run BFS from each vertex in H. Next, we choose a vertex w maximally distant from H, compute $N_s(w)$, and run BFS from each vertex in $N_s(w)$. The maximum BFS depth observed yields the diameter estimate.

Pseudo Code

```
**Input:** $G = (V, E),\ s = \Theta(\sqrt{n})$.
For each $v \in V$ : run BFS$(v)$ until $s$ vertices are discovered; $N_s(v) \leftarrow$ first $s$ vertices.
$H \leftarrow$ greedy hitting set of $\{N_s(v) : v \in V\}$.
For each $h \in H$ : run BFS$(h)$.
$w = \arg\max_{v\in V} \min_{h\in H} d(v, h)$.
Run BFS$(w)$, let $N_s(w)$ be first $s$ vertices visited.
For each $x \in N_s(w)$ : run BFS$(x)$.
$\hat{D} = \max_{z\in H\cup N_s(w)} \max_{v\in V} d(z, v)$.
**Output:** $\hat{D}$.
```

Proof of Correctness

Since H intersects every $N_s(v)$, each vertex v has some $h \in H$ with $d(v, h)$ small. This ensures that BFS from H reaches far enough toward one diameter endpoint. Similarly, selecting w farthest from H and expanding to $N_s(w)$ captures the opposite endpoint. Thus:

$$\frac{2D}{3} \leq \hat{D} \leq D.$$

Running Time Analysis

- Computing all $N_s(v)$ sets: $O(ns^2) = O(n^2)$ for $s = \Theta(\sqrt{n})$.
- Greedy hitting set construction: $\tilde{O}(n^2)$.
- $|H| = O(n/s) = O(\sqrt{n})$ BFS calls: $O(m\sqrt{n})$.

- $|N_s(w)| = s = \Theta(\sqrt{n})$ BFS calls: $O(m\sqrt{n})$.

Total running time:

$$\boxed{\tilde{O}(m\sqrt{n} + n^2)}.$$

2.6 Eccentricity Approximation Algorithm

Problem Statement

For every node v, compute an estimate $\hat{e}(v)$ such that:

$$\max\{r,\ \tfrac{2}{3}ecc(v)\} \ \le\ \hat{e}(v) \ \le\ \min\{D,\ \tfrac{3}{2}ecc(v)\},$$

where $ecc(v)$ is the eccentricity of v, r is the graph radius, and D is the diameter. This achieves a $(2/3, 3/2)$ multiplicative approximation and runs in:

$$\tilde{O}(m\sqrt{n}) \text{ expected time.}$$

Explanation

The algorithm begins by selecting a random sample S of $\Theta(\sqrt{n}\log n)$ vertices. Distances from all vertices to S are obtained by running BFS from each vertex in S. We also record the eccentricities $ecc(q)$ for every $q \in S$ directly from their BFS trees, since the farthest vertex reached in each BFS gives $ecc(q)$. For each vertex v, we then identify $p_S(v)$, the closest vertex in S.

We then choose a vertex w that maximizes the distance $d(w, p_S(w))$; this vertex can be thought of as lying deep in the graph.

Next, we run a BFS from w, and for each vertex v, we identify v_t, the first vertex on the shortest path from v to w that lies within the first $\sqrt{n}$ layers of the BFS tree rooted at w.

To estimate eccentricity, we first define:

$$e'(v) = \max\left(d(v, w),\ \max_{q \in S} d(v, q)\right).$$

We then compare how v lies relative to w along its shortest path. If v is "closer" to the midpoint of the path, the eccentricity can be approximated using $ecc(v_t)$. Otherwise, the estimate relies on eccentricities of vertices in S. This distinction ensures that the approximation remains within the $(2/3, 3/2)$ factor.

Pseudo Code

```
**Input:** $G = (V, E)$
**Sample:** $S \subseteq V,\ |S| = \Theta(\sqrt{n}\log n)$.
For each $q \in S$ : run BFS($q$).
$p_S(v) = \arg\min_{q \in S} d(v, q),\ \forall v \in V$.
$w = \arg\max_{v \in V} d(v, p_S(v))$.
Run BFS($w$).
$v_t$ = first vertex on shortest path($v \to w$) in first $\sqrt{n}$ BFS layers.
$e'(v) = \max\big(d(v, w),\ \max_{q \in S} d(v, q)\big)$.
$\hat{e}(v) = \begin{cases} \max\{e'(v),\ ecc(v_t)\}, & \text{if } d(v, v_t) \le d(v_t, w), \\ \max\{e'(v),\ \min_{q \in S} ecc(q)\}, & \text{otherwise.} \end{cases}$
```

Proof of Correctness

In the case $d(v, v_t) \le d(v_t, w)$, the node v_t lies close to v along a shortest path to w, implying:

$$ecc(v_t) \le ecc(v) + d(v, v_t) \le \tfrac{3}{2} ecc(v).$$

In the complementary case, the structure of the BFS layers ensures that the distance from v to w satisfies:

$$d(v, w) \ge \tfrac{2}{3} ecc(v),$$

giving the lower bound. Therefore:

$$\tfrac{2}{3} ecc(v) \le \hat{e}(v) \le \tfrac{3}{2} ecc(v).$$

Running Time Analysis

BFS from all vertices in S costs $O(m\sqrt{n})$, BFS from w costs $O(m)$, and all remaining computation is linear in n. Thus, the overall running time is:

$$\boxed{\tilde{O}(m\sqrt{n})}.$$

Radius Approximation

Since we have already computed approximate eccentricities $\hat{e}(v)$ for all $v \in V$, the radius can be approximated directly by:

$$\hat{r} = \min_{v \in V} \hat{e}(v).$$

This requires only a linear scan over all vertices.

2.7 Constant-Diameter Deterministic Algorithm

Problem Statement

Let the diameter of the graph be $D = 3h+z$, where h is a fixed constant and $z \in \{0,1,2\}$. The goal is to compute a $\frac{2D}{3}$-approximation to the diameter deterministically in:

$$\tilde{O}\left(m^{2-\frac{1}{2h+3}}\right) \text{ time.}$$

Explanation

When the diameter is small, the BFS layers expand slowly, and the graph has limited depth. The algorithm exploits this by selecting a set of "high-degree" vertices that act as natural structural hubs.

Let Δ be a degree threshold. Define:

$$H = \{v \in V : \deg(v) \geq \Delta\}.$$

Since h is constant, the neighborhood of any vertex can be explored in at most h BFS steps without the graph blowing up in size.

We run BFS from all vertices in H. If any of these BFS trees reaches depth at least $\frac{2D}{3}$, we obtain a diameter witness. If not, then every vertex is close to some high-degree vertex. In that case, we select a vertex w maximizing the distance to H, run BFS from w, and use this to approximate the diameter in the same manner as the randomized algorithm, but without sampling.

Choosing $\Delta = m^{1/(2h+3)}$ balances the number of BFS calls with their depth, giving the stated running time.

Pseudo Code

```
**Input:** $G = (V, E),\ D = 3h + z,\ h$ constant.
$\Delta = m^{1/(2h+3)}$.
$H = \{v \in V : \deg(v) \geq \Delta\}$.
For each $u \in H :$ run BFS($u$).
If any BFS depth $\geq \frac{2D}{3},$ return that value.
$d(v, H) = \min_{u \in H} d(v, u),\ \forall v \in V$.
$w = \arg\max_{v \in V} d(v, H)$.
Run BFS($w$).
$\hat{D}$ = depth of BFS($w$).
**Output:** $\hat{D}$.
```

Proof of Correctness

If some $u \in H$ can reach a vertex at distance at least $\frac{2D}{3}$, we immediately obtain a (2/3)-approximation.

Otherwise, every vertex is within distance $< \frac{2D}{3}$ of H. Selecting w maximizes distance to H, forcing $d(w, H)$ to be large. Then, the BFS tree rooted at w reaches a vertex at distance $\geq \frac{2D}{3}$, ensuring:

$$\frac{2D}{3} \leq \hat{D} \leq D.$$

Running Time Analysis

The set H has size at most:

$$|H| \leq \frac{2m}{\Delta} = O\left(m^{1-\frac{1}{2h+3}}\right).$$

Running BFS from each vertex in H costs:

$$|H| \cdot O(m) = \tilde{O}\left(m^{2-\frac{1}{2h+3}}\right).$$

All remaining steps are linear or smaller. Therefore the total running time is:

$$\boxed{\tilde{O}\left(m^{2-\frac{1}{2h+3}}\right)}.$$

2.8 Improved Diameter Distinguishing Procedure

Problem Statement

Given an estimate $\hat{D}$ such that:

$$\frac{2D}{3} \leq \hat{D} \leq D,$$

distinguish whether the true diameter D equals $\hat{D}$, $\hat{D}+1$, or $\hat{D}+2$, and refine $\hat{D}$ accordingly. This refinement runs in:

$$\tilde{O}(m\sqrt{n}) \text{ time.}$$

Explanation

Once we have a coarse $(2/3, 1)$-approximation of the diameter, we would like to determine whether D is exactly $\hat{D}$ or slightly larger. The key idea is to examine BFS layers carefully around candidate diameter endpoints.

Let a be a vertex whose BFS depth equals $\hat{D}$ (i.e., a diameter witness from the approximate algorithm). Vertices at depth $\hat{D}$ from a are potential endpoints of longest shortest-paths.

If any such vertex b has no incoming or outgoing edges from nodes that lie within distance $\hat{D}-1$, then the shortest-path structure forces the diameter to be at least $\hat{D}+1$. By repeating this one more step, we can test whether the diameter is $\hat{D}$, $\hat{D}+1$, or $\hat{D}+2$.

This procedure works because BFS layers act as combinatorial "certificates" of distance in unweighted graphs.

Pseudo Code

```
**Input:** $G = (V, E),\ \hat{D},\ a$ s.t. BFS($a$) gives depth $\hat{D}$.
Run BFS($a$).
$L = \{v : d(a, v) = \hat{D}\}$.
For each $b \in L$ : run BFS($b$).
If $\exists b \in L : \max_v d(b, v) = \hat{D} :\quad D = \hat{D}$.
Else if $\exists b \in L : \max_v d(b, v) = \hat{D} + 1 :\quad D = \hat{D} + 1$.
Else: $D = \hat{D} + 2$.
**Output:** $D$.
```

Proof of Correctness

Vertices in L are the farthest possible from a under $\hat{D} \leq D$. If one such vertex b achieves depth $\hat{D}$, then no pair of vertices is farther apart than $\hat{D}$, so $D = \hat{D}$. If not, but b reaches $\hat{D}+1$, then $D = \hat{D}+1$. Otherwise, distances grow at most by one more BFS layer, giving $D = \hat{D} + 2$. Thus, this check correctly resolves the diameter to within 0, +1, or +2.

Running Time Analysis

The BFS from a costs $O(m)$. The layer L has at most $O(\sqrt{n})$ vertices due to the sampling structure used earlier, so BFS from all $b \in L$ costs:

$$O(\sqrt{n} \cdot m) = \tilde{O}(m\sqrt{n}).$$

This matches the running time of the diameter approximation step and does not increase overall complexity.

3 SETH-Based Hardness of Improving the Approximation Factor

Statement of Result

The approximation factor $\frac{3}{2}$ for the diameter in sparse graphs is unlikely to be improved in truly subquadratic time. Formally, for any $\varepsilon > 0$, if there exists an algorithm that computes a $(\frac{3}{2} - \varepsilon)$-approximation to the diameter of an unweighted sparse graph in time:

$$O(m^{2-\varepsilon}),$$

then the **Strong Exponential Time Hypothesis (SETH)** is false.

Explanation

The Strong Exponential Time Hypothesis states that for every $\delta > 0$, there exists a k such that k-SAT on n variables cannot be solved in time $O(2^{(1-\delta)n})$. It is widely believed to be true and used as a fine-grained analogue of $\mathbf{P} \neq \mathbf{NP}$.

The reduction in the paper constructs, in near-linear time, a sparse graph whose diameter encodes whether a given CNF-SAT formula is satisfiable. The key insight is that distances between carefully arranged vertex sets reflect clause satisfaction structure:

$$\text{diameter} = 3 \quad \text{if the formula is satisfiable,}$$

$$\text{diameter} = 4 \quad \text{if the formula is not satisfiable.}$$

Thus, distinguishing between diameter 3 and 4 in $O(m^{2-\varepsilon})$ time would yield a faster-than-2^n algorithm for SAT, contradicting SETH.

Consequence

This result explains why all known diameter approximation algorithms for sparse graphs stop at the $\frac{3}{2}$ threshold:

$$\boxed{\textbf{Improving the approximation beyond } \tfrac{3}{2} \textbf{ requires breaking SETH.}}$$

Therefore, the $(2/3, 3/2)$ approximation factors for eccentricity, radius, and diameter produced by the algorithms studied in this paper are conditionally optimal under standard complexity assumptions.

4 Discussion

Algorithm trade-offs: The choice of method depends on the required accuracy and computational constraints:

- **BFS-Based APSP (Exact)** provides the true diameter and radius but requires $\Theta(mn)$ time, which becomes infeasible for large sparse graphs.
- **2-Approximation via Double BFS** runs in $O(m+n)$ time and gives a simple and fast estimate, though the worst-case factor of 2 can be loose.
- **Aingworth et al. Diameter Approximation Algorithm** improves the approximation to a (3/2)-factor using bounded neighborhoods and a hitting set, but requires $\tilde{O}(m\sqrt{n} + n^2)$ time.
- The **Randomized Diameter Approximation Algorithm (Main Result)** achieves the same (3/2) approximation in expected $\tilde{O}(m\sqrt{n})$ time by replacing the explicit hitting set with a random sample, eliminating the n^2 neighborhood computation.
- **Deterministic Diameter Approximation via Hitting Set** restores determinism but reintroduces the $\tilde{O}(n^2)$ preprocessing cost.
- The **Eccentricity Approximation Algorithm** extends the sampling-based idea to approximate all vertex eccentricities within $(2/3, 3/2)$ in $\tilde{O}(m\sqrt{n})$ time, which directly yields a radius approximation.
- When the diameter is small and of the form $D = 3h + z$ for constant h, the **Constant-Diameter Deterministic Algorithm** achieves a $\frac{2D}{3}$-approximation in $\tilde{O}(m^{2-\frac{1}{2h+3}})$ time.

- The **Improved Diameter Distinguishing Procedure** refines an existing approximation to distinguish whether the true diameter differs by +0, +1, or +2.

Limits of approximation: The **SETH-Based Hardness Result** suggests that improving the approximation factor beyond 3/2 in truly subquadratic time is unlikely. Therefore, the (3/2) barrier observed in the deterministic and randomized diameter approximation algorithms is believed to be essentially optimal for sparse graphs.

5 Conclusion

This report surveyed several increasingly efficient methods for estimating diameter, radius, and eccentricity in sparse unweighted graphs. Beginning with the exact BFS-Based APSP (Exact) algorithm, we moved to the 2-Approximation via Double BFS, which provides a fast but coarse estimate. The Aingworth et al. Diameter Approximation Algorithm achieves a stronger (3/2) guarantee through bounded neighborhoods and hitting sets, and its randomized refinement eliminates the n^2 term to obtain a $\tilde{O}(m\sqrt{n})$ running time. A deterministic variant recovers reproducibility at the cost of the additional neighborhood computation. We also considered the Constant-Diameter Deterministic Algorithm for graphs with small diameter, the Eccentricity Approximation Algorithm for approximating all vertex eccentricities (and hence the radius), and the Improved Diameter Distinguishing Procedure to refine coarse diameter estimates.

Together, these algorithms illustrate how small sets of strategically chosen BFS explorations can capture the global structure of a graph and reduce computation from $\Theta(mn)$ to $\tilde{O}(m\sqrt{n})$ while maintaining a (3/2)-style approximation. Finally, SETH-based hardness provides evidence that this approximation factor is near-optimal for sparse graphs, explaining the persistent nature of the (3/2) threshold.