

Distance Oracles

Christian Wulff-Nilsen

Algorithmic Techniques for Modern Data Models

DTU

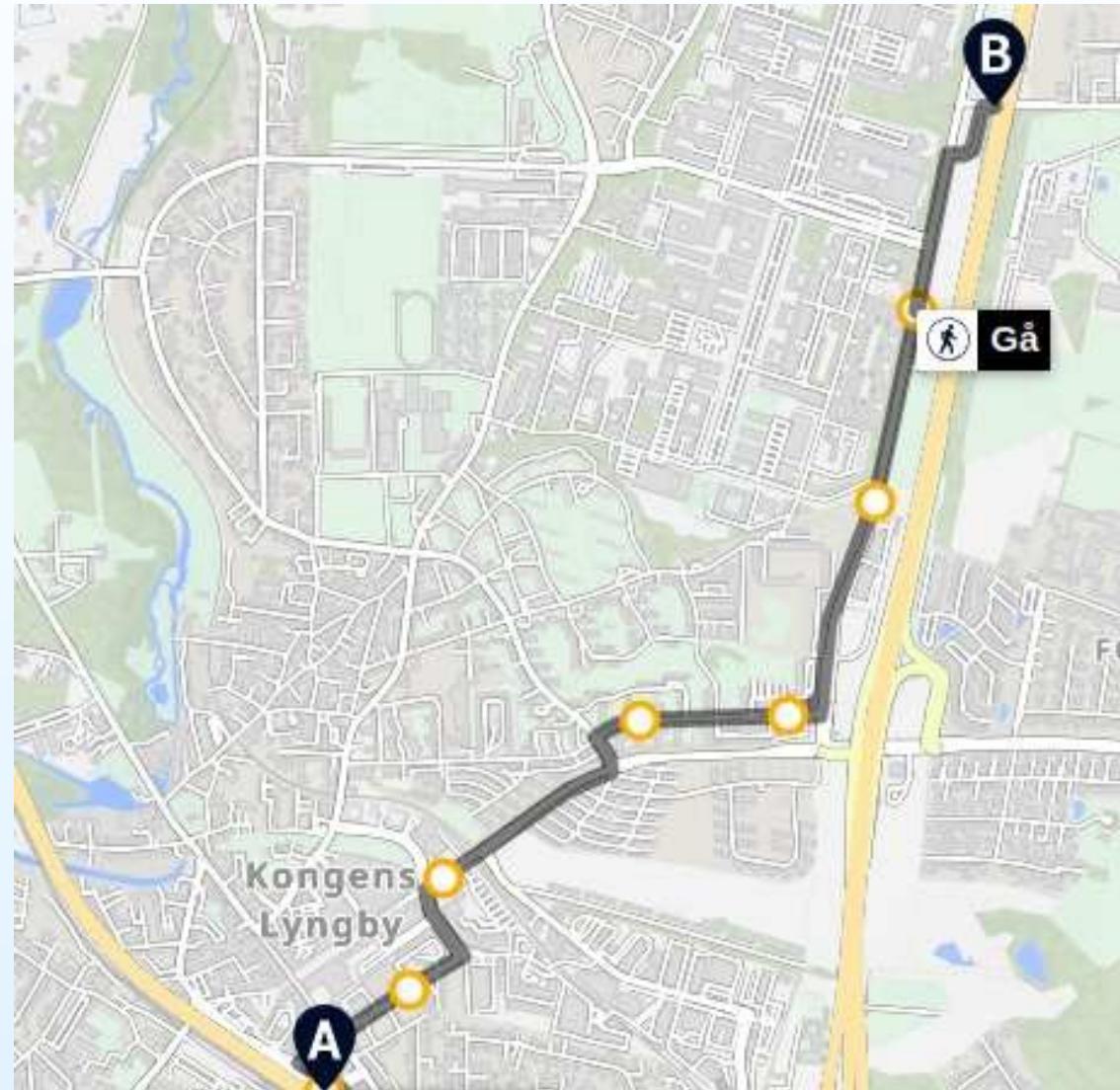
September 12, 2025

Overview for today

- Shortest paths and distance oracles
- Approximate distance oracles
- A special case of the Thorup-Zwick approximate distance oracle
- The general Thorup-Zwick oracle:
 - Bounding space
 - Bounding stretch

The shortest path problem

- The shortest path problem:



Our problem

- For an edge-weighted graph $G = (V, E)$, let $\delta(s, t)$ be the shortest path distance in G from $s \in V$ to $t \in V$
- We want a data structure for G that can answer queries of the form “What is $\delta(s, t)$?” for any $s, t \in V$

Approximate distances

- Let $n = |V|$
- It can be shown that in the worst case, $\Theta(n^2)$ space is the best we can hope for if we want exact distances
- We therefore consider *approximate* shortest path distances

Approximate distance oracle

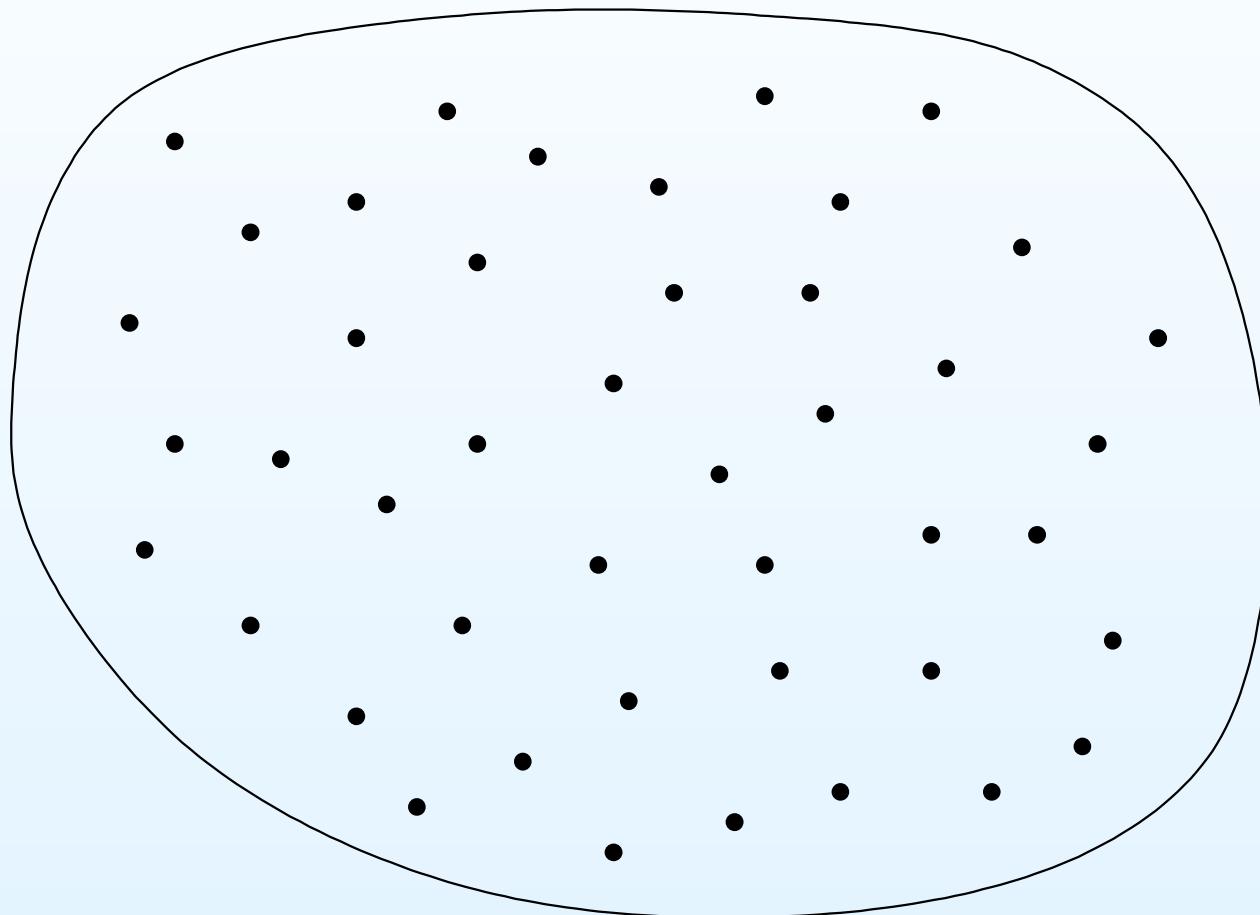
- For any $u, v \in V$, $\tilde{\delta}(u, v)$ is an estimate of $\delta(u, v)$ of *stretch* $t \geq 1$ if

$$\delta(u, v) \leq \tilde{\delta}(u, v) \leq t \cdot \delta(u, v)$$

- t -approximate distance oracle:
 - Answers queries with estimates of stretch t
 - Should preferably use a small amount of space
- From now on, we consider only *undirected* edge-weighted graphs since for any stretch t , $\Theta(n^2)$ space is the best we can hope for for directed graphs

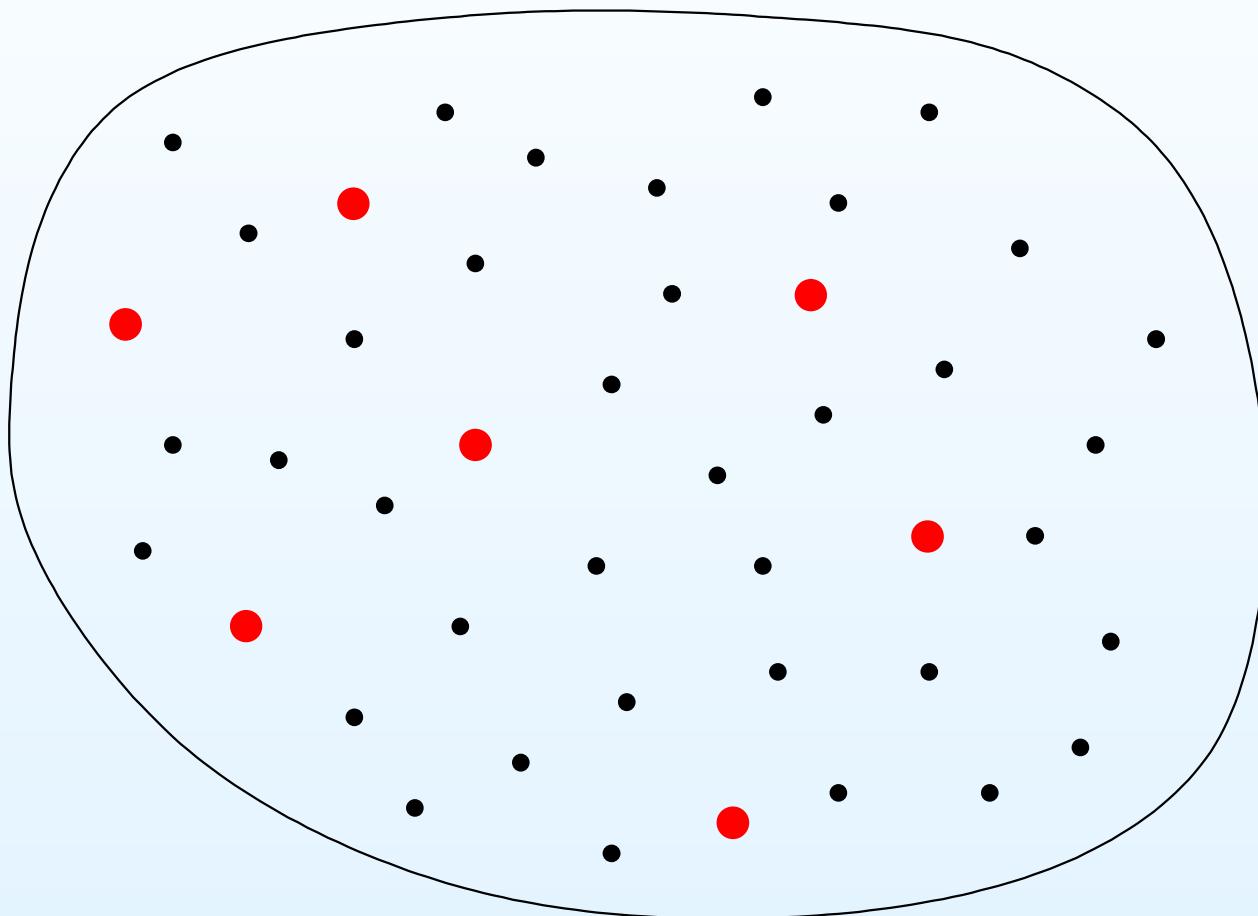
A simple approximate distance oracle

- Sample each vertex independently with some probability p :



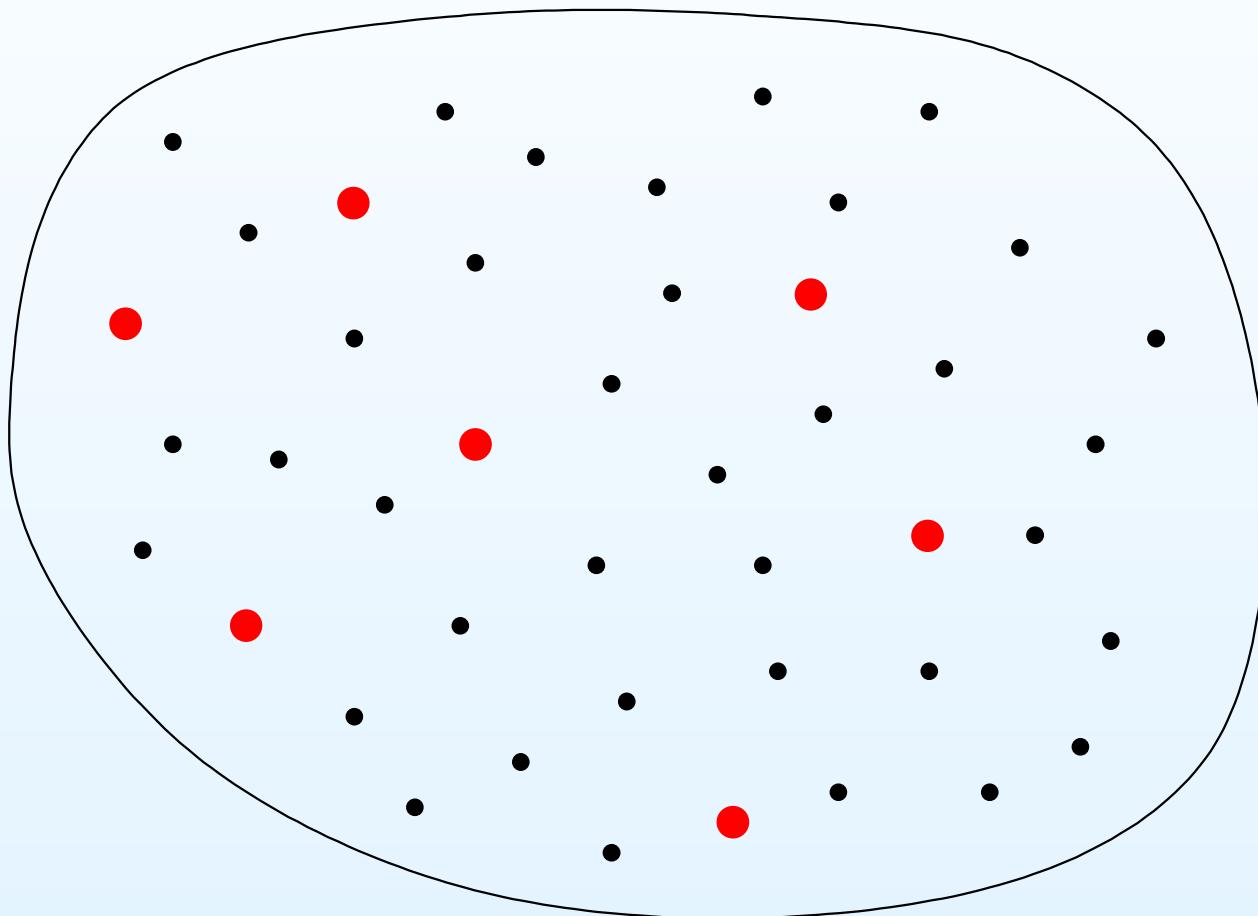
A simple approximate distance oracle

- Sample each vertex independently with some probability p :



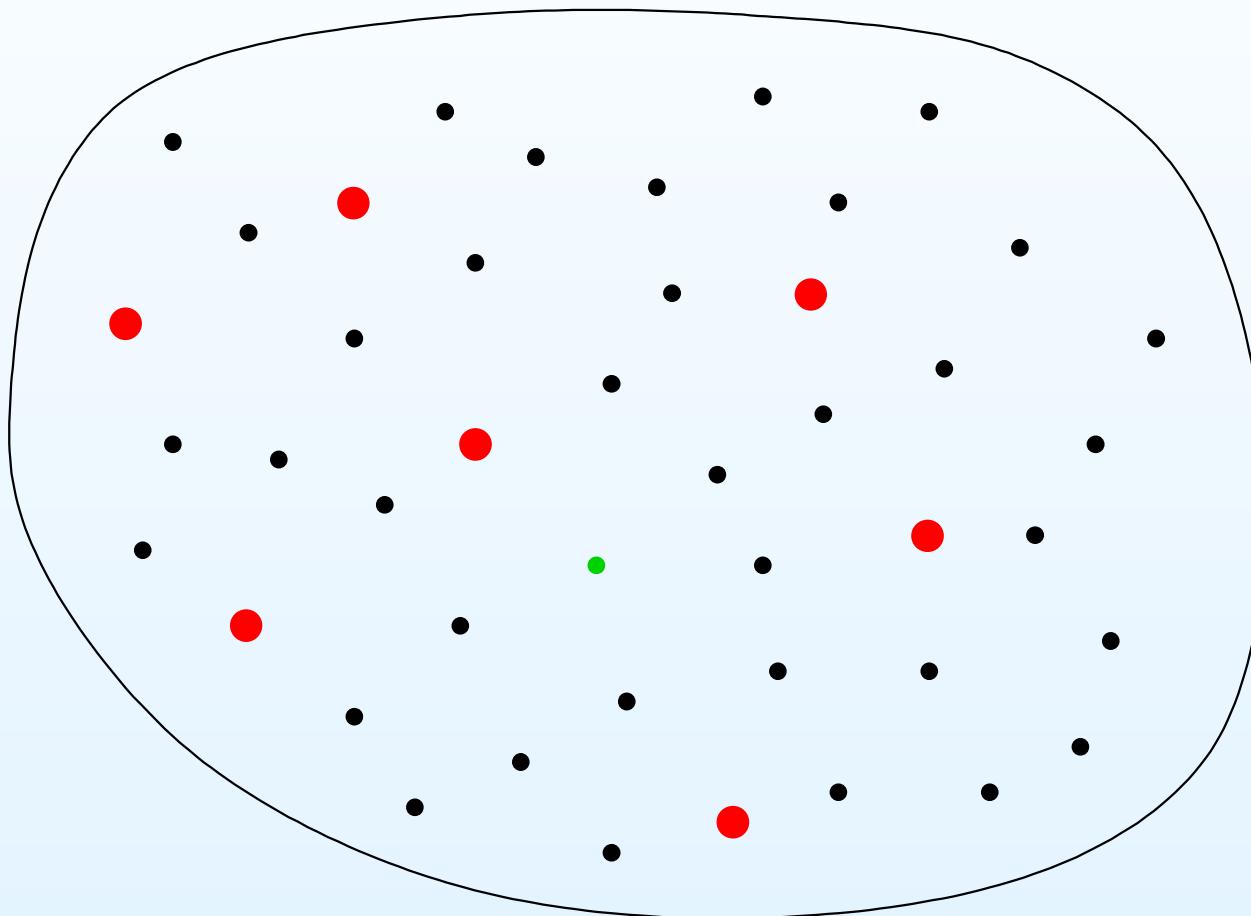
A simple approximate distance oracle

- For each vertex:



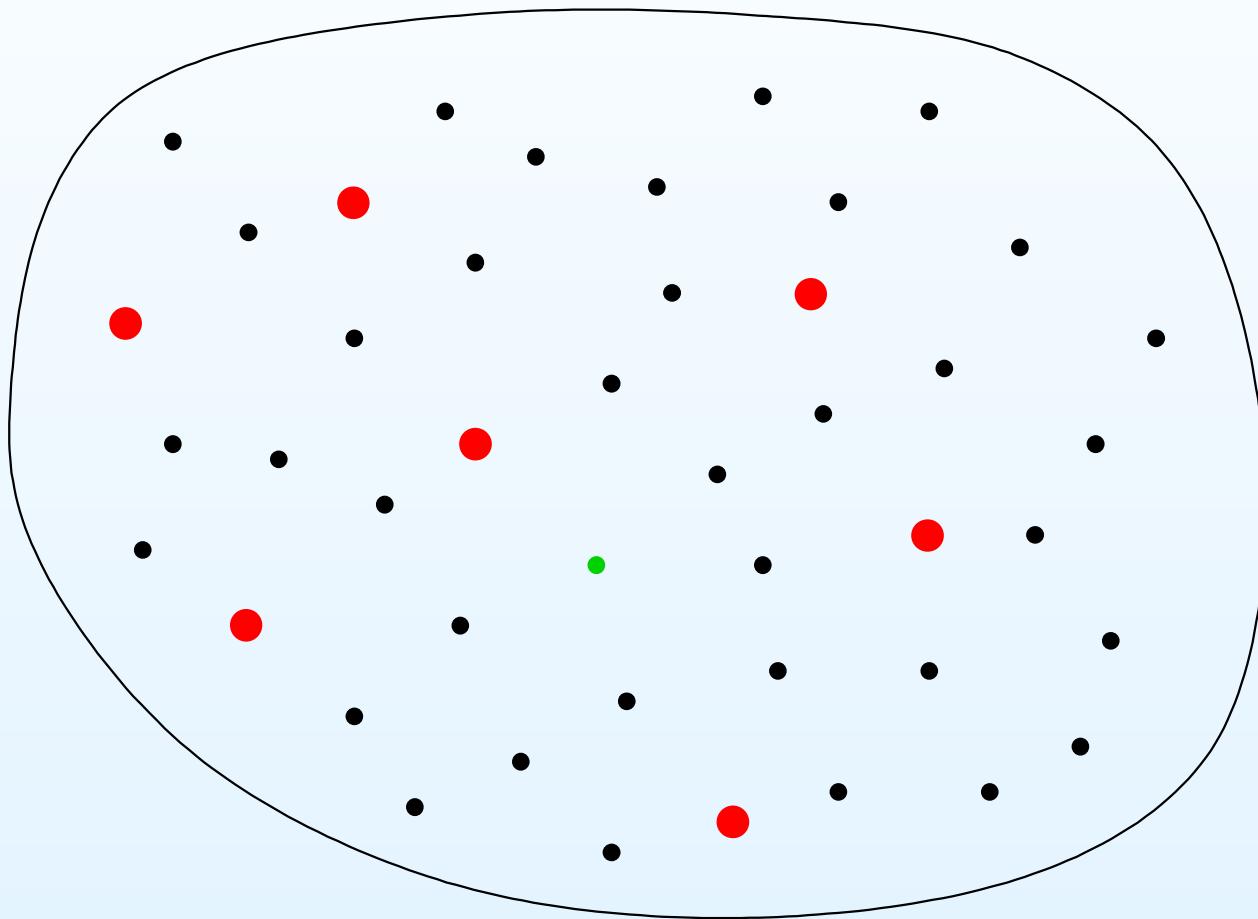
A simple approximate distance oracle

- For each vertex:



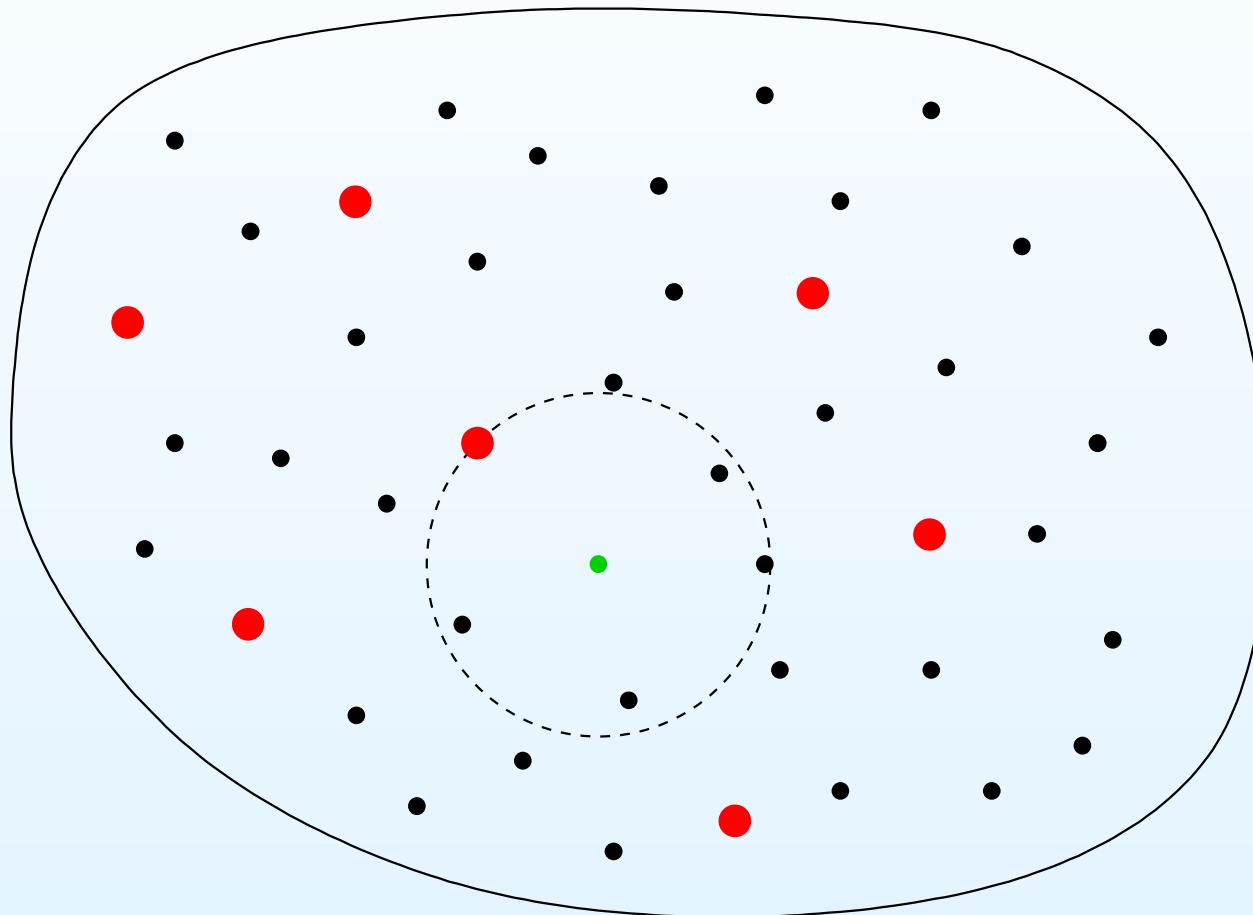
A simple approximate distance oracle

- Consider vertices closer than nearest sampled vertex:



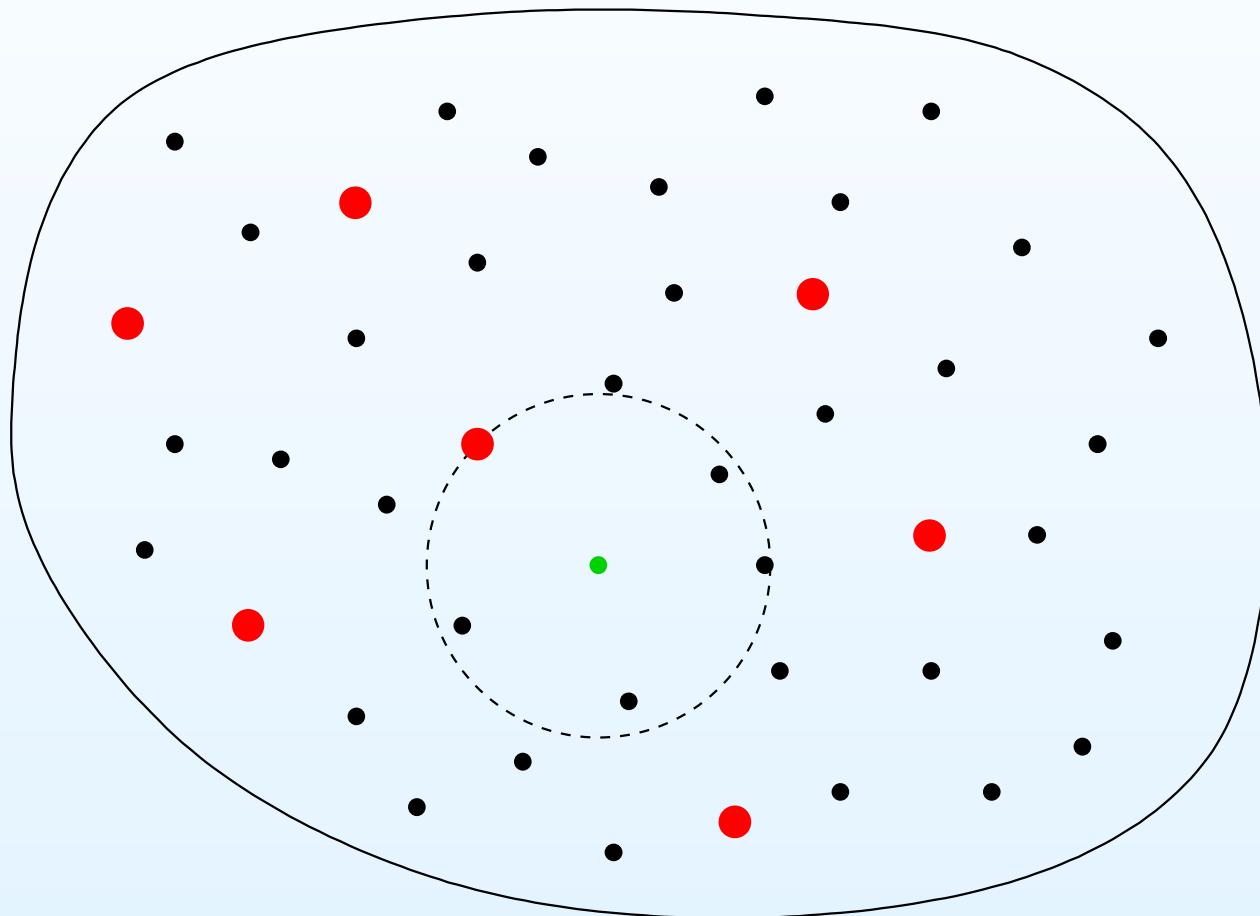
A simple approximate distance oracle

- Consider vertices closer than nearest sampled vertex:



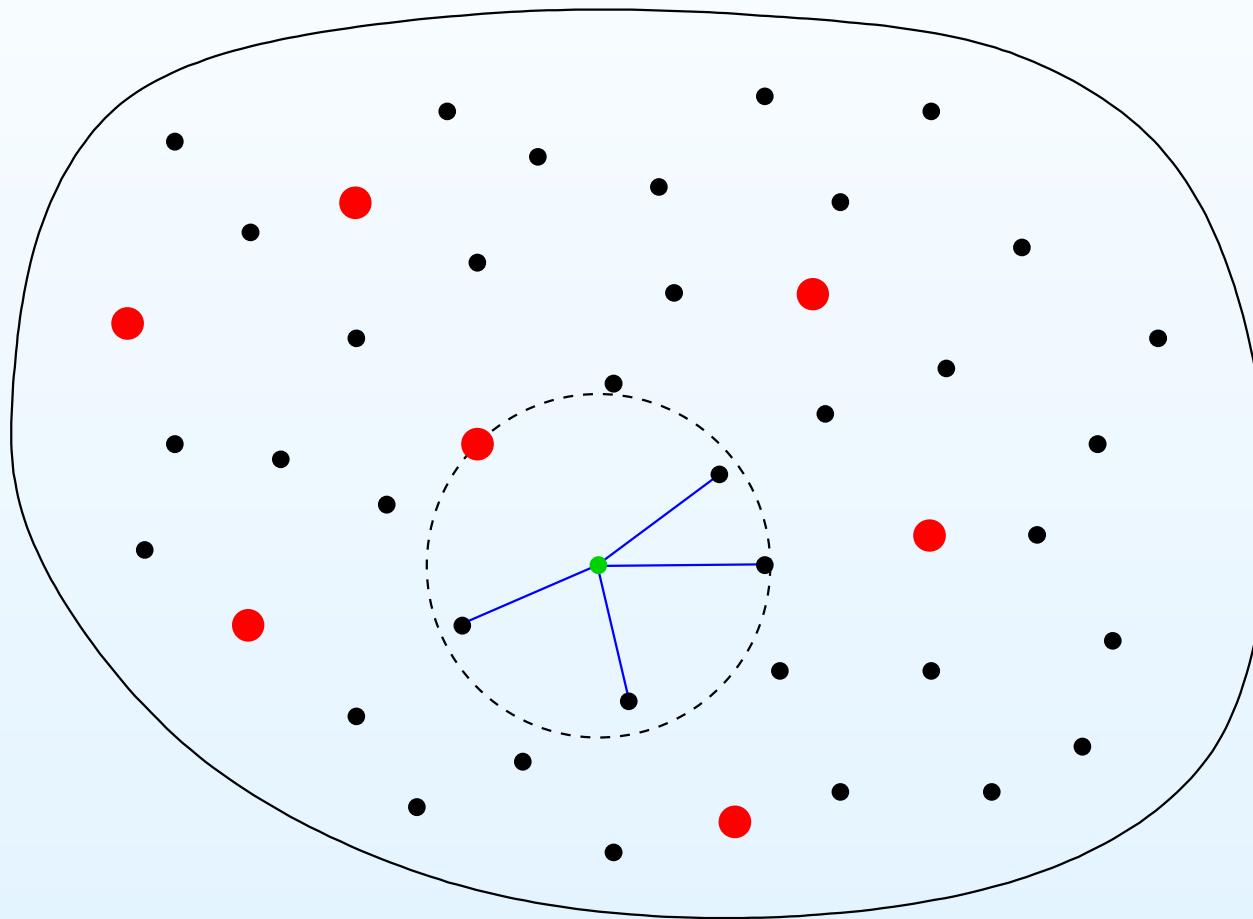
A simple approximate distance oracle

- Compute and store distances to these vertices:



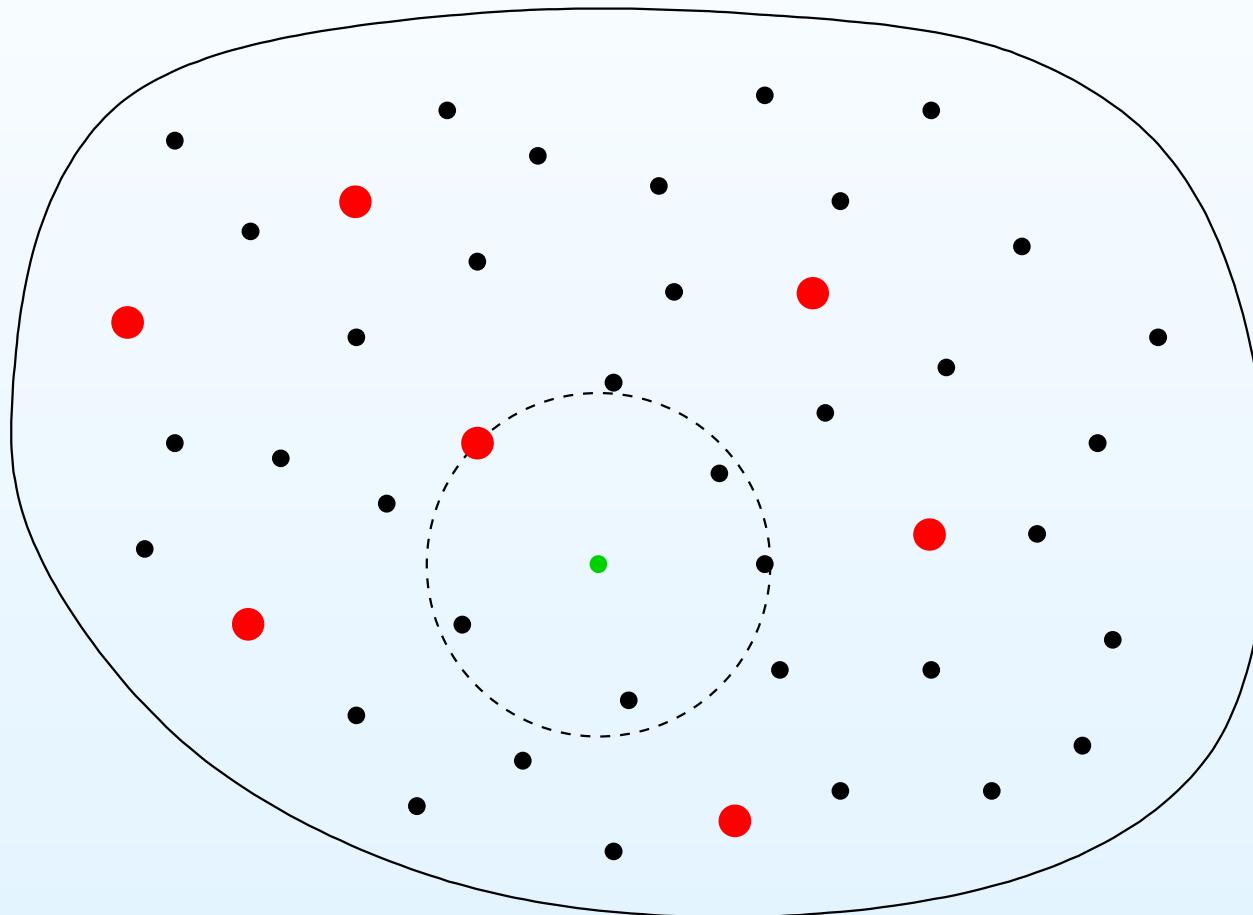
A simple approximate distance oracle

- Compute and store distances to these vertices:



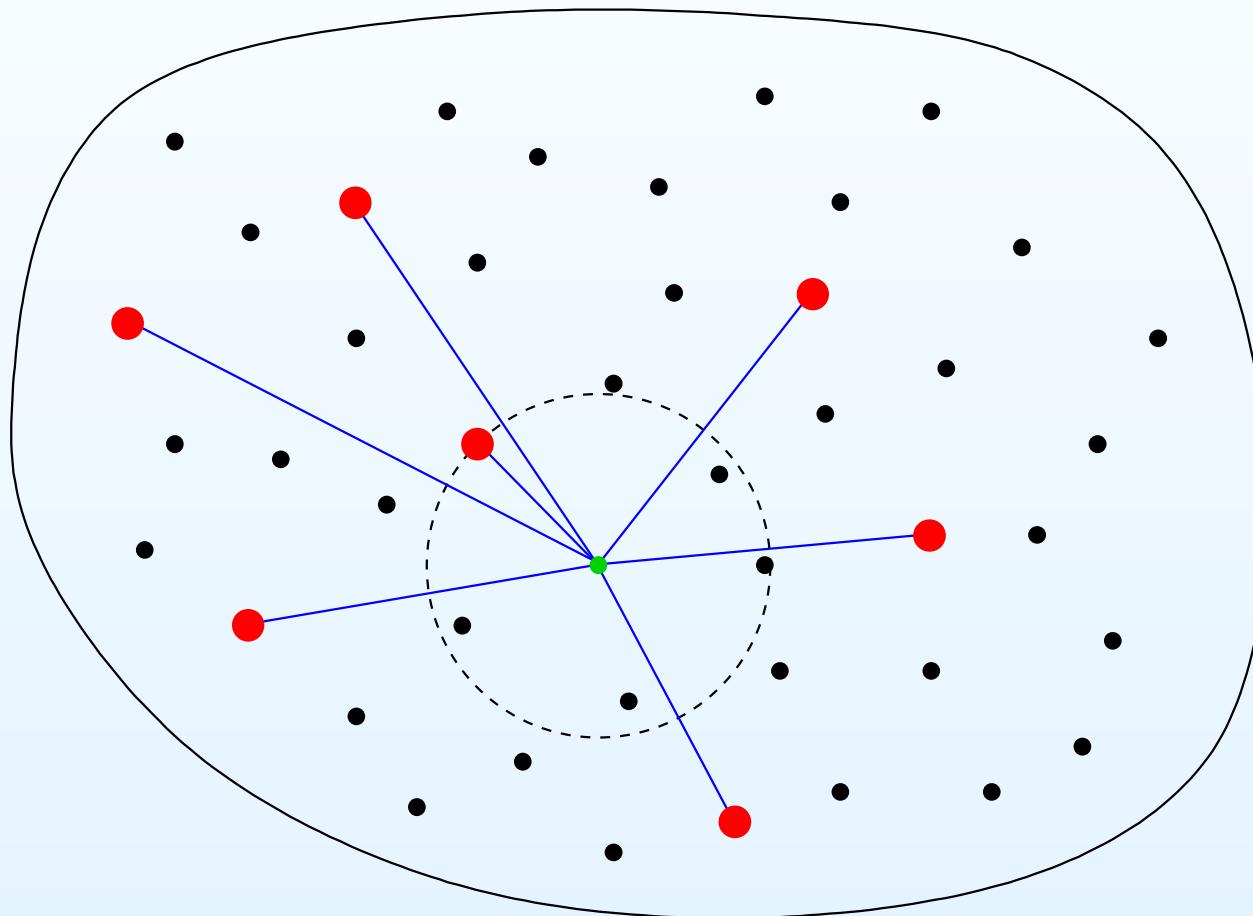
A simple approximate distance oracle

- Compute and store distances to all sampled vertices:



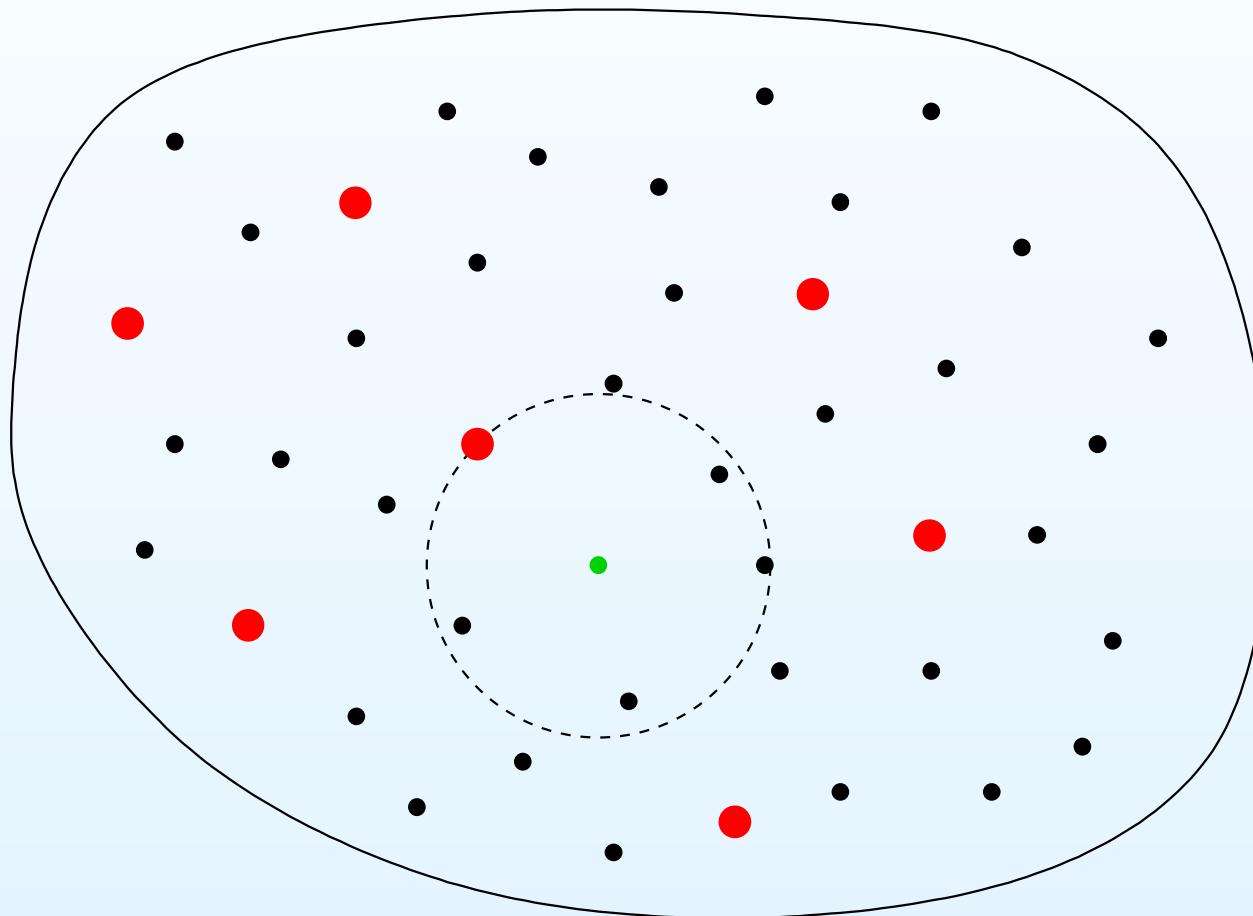
A simple approximate distance oracle

- Compute and store distances to all sampled vertices:



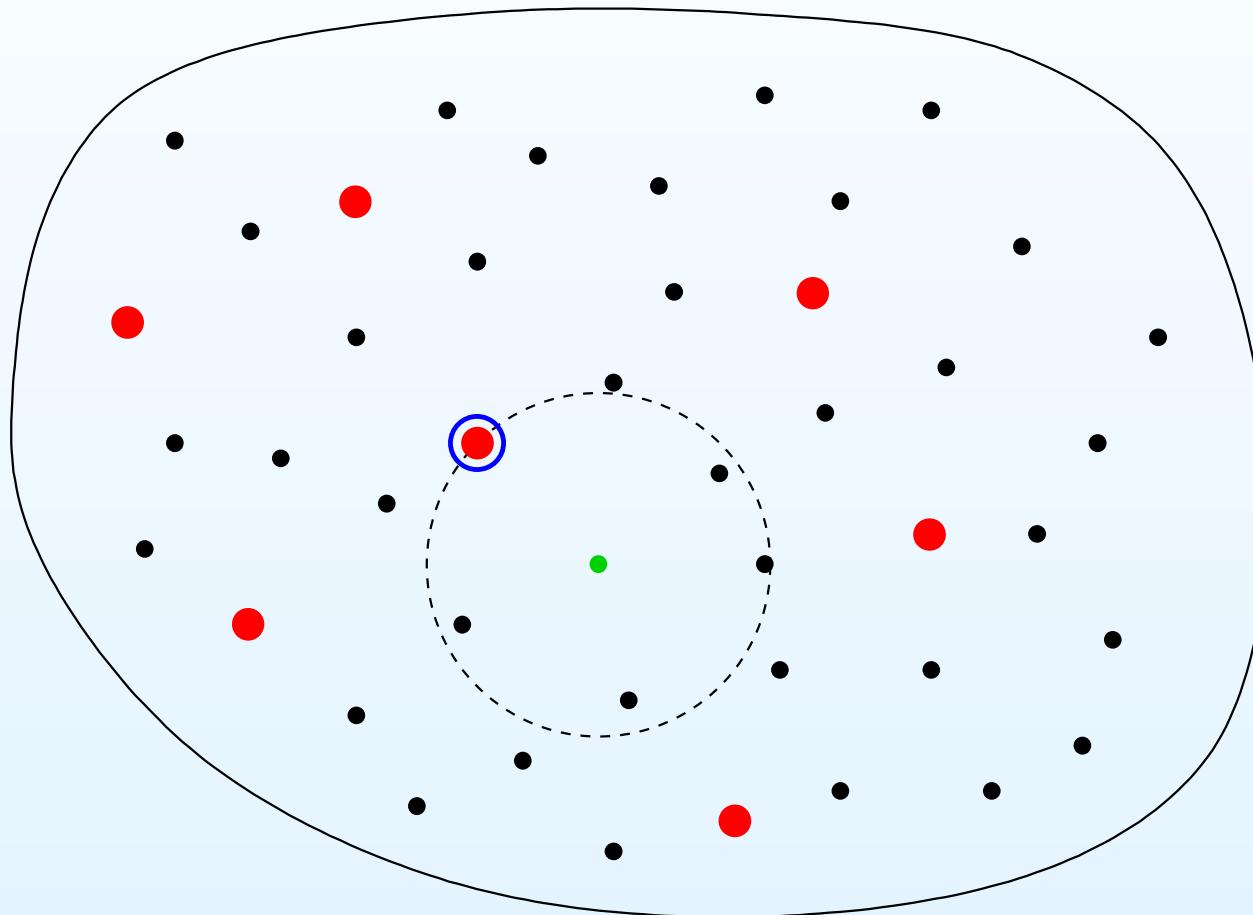
A simple approximate distance oracle

- Store the nearest sampled vertex:



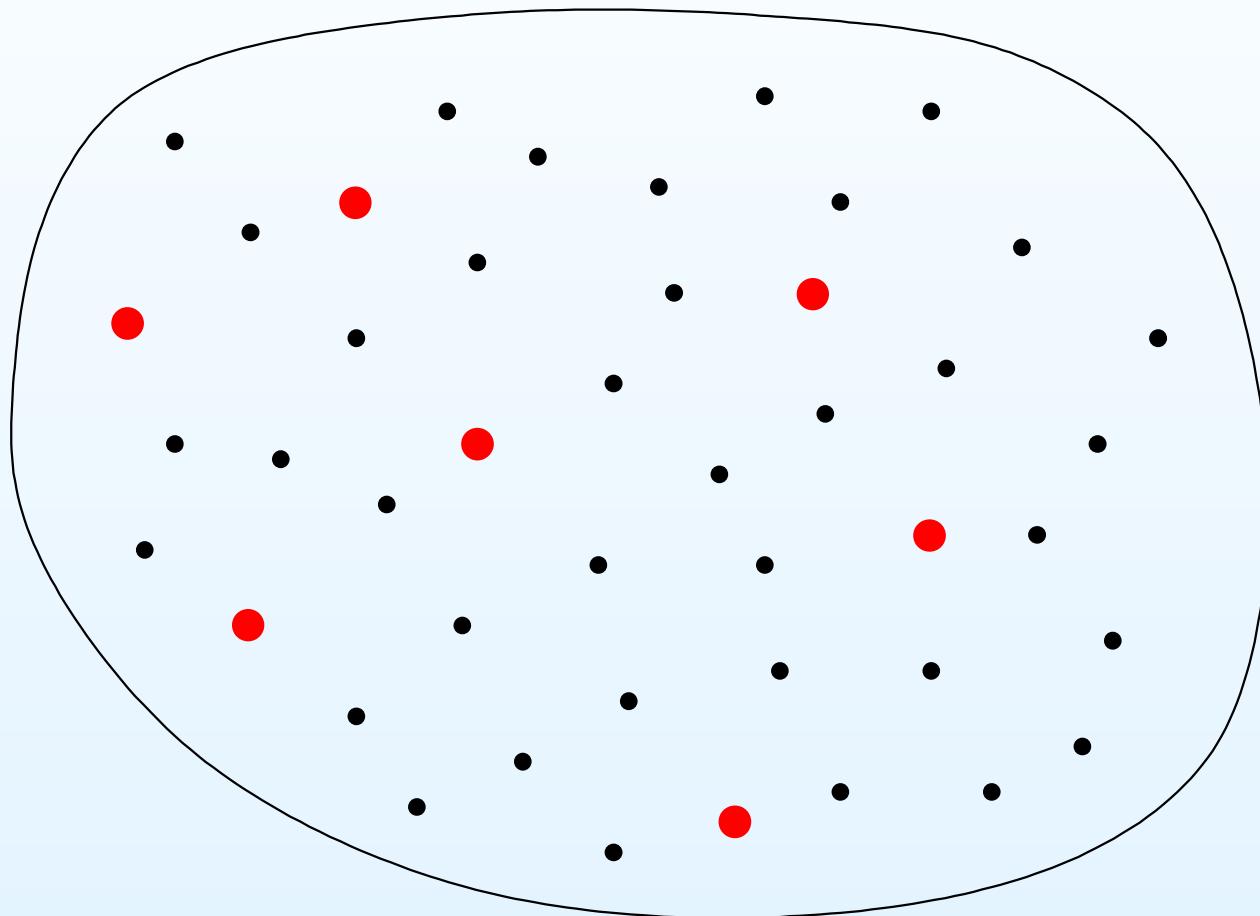
A simple approximate distance oracle

- Store the nearest sampled vertex:



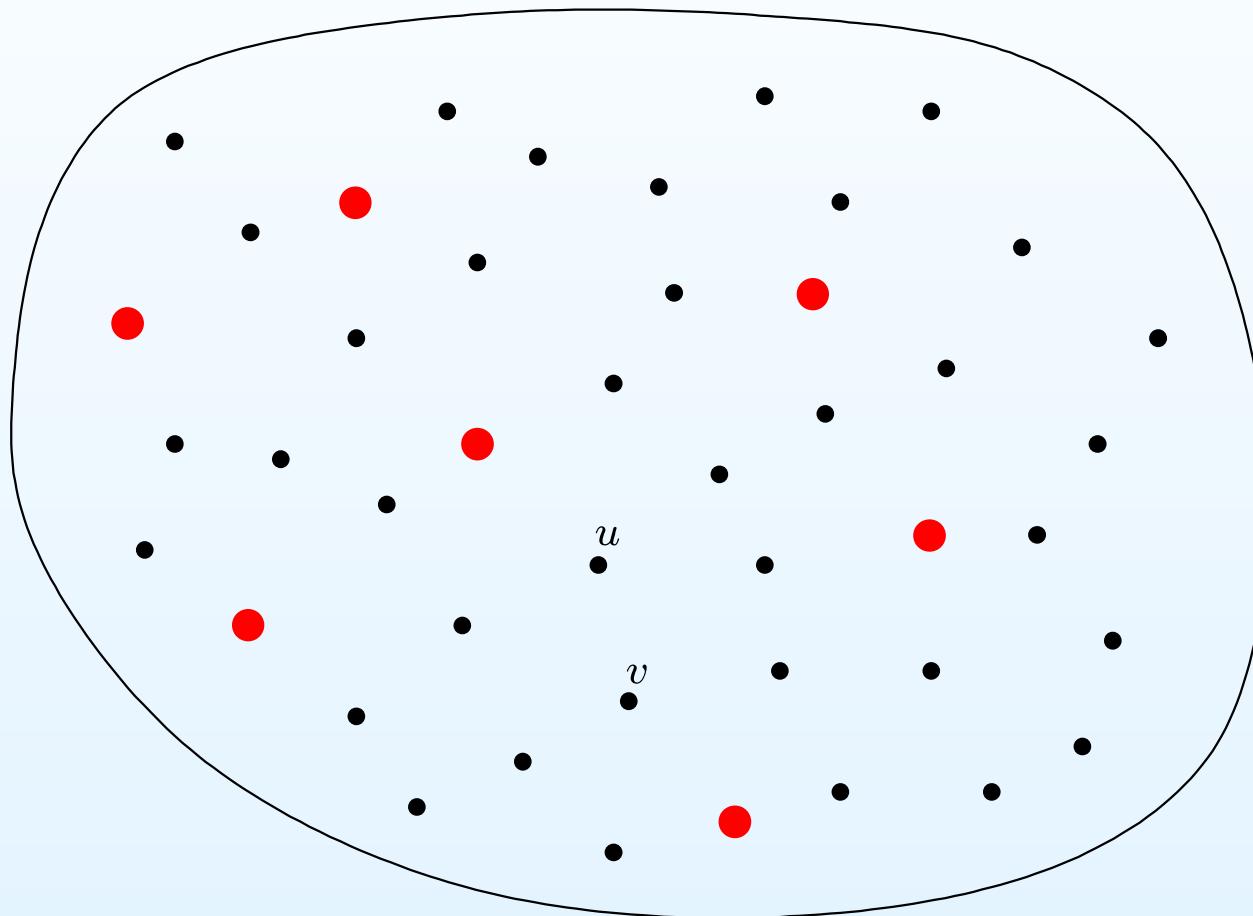
A simple approximate distance oracle

- To answer a query for the distance between u and v :



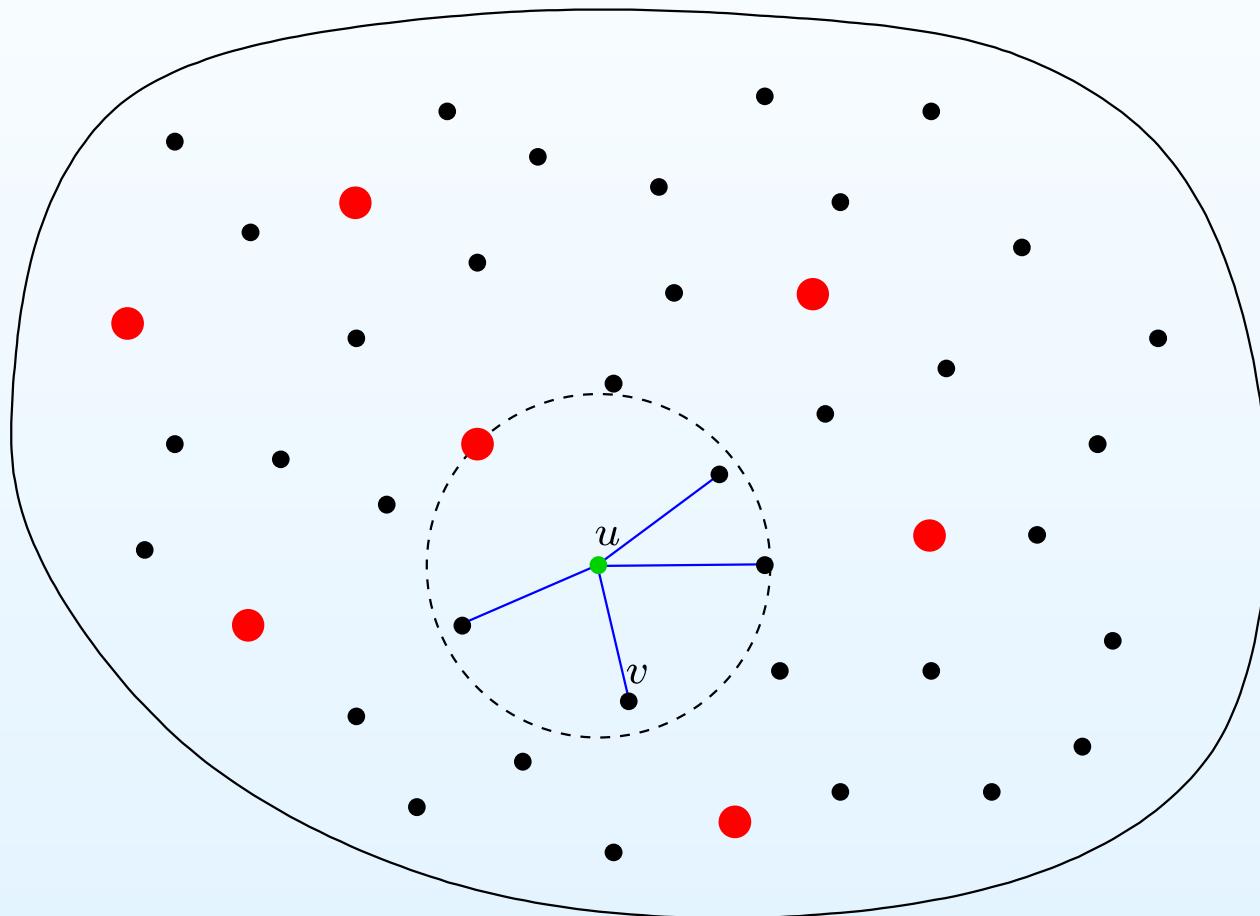
A simple approximate distance oracle

- To answer a query for the distance between u and v :



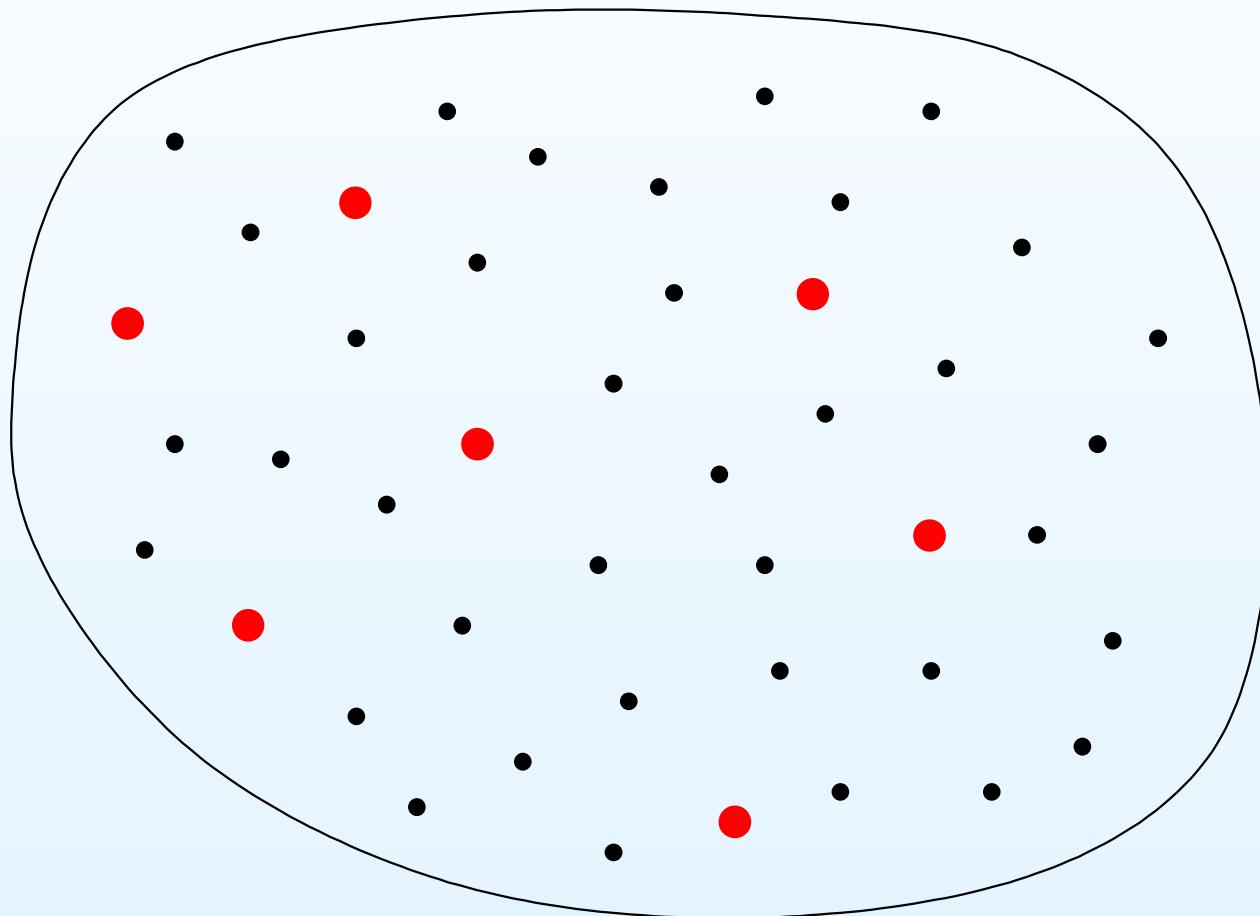
A simple approximate distance oracle

- Simple look-up if v is closer to u than nearest sampled vertex:



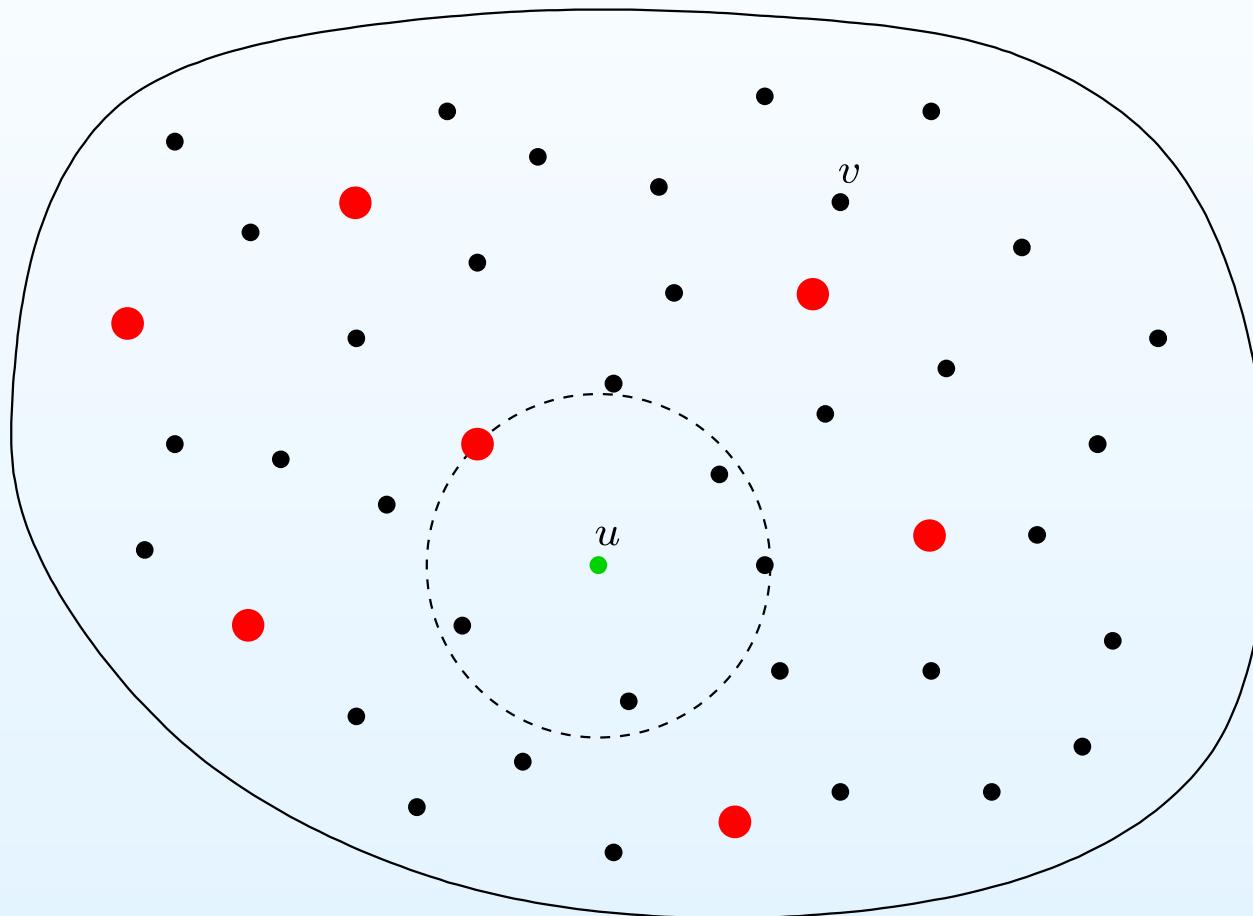
A simple approximate distance oracle

- Simple look-up if v is closer to u than nearest sampled vertex:



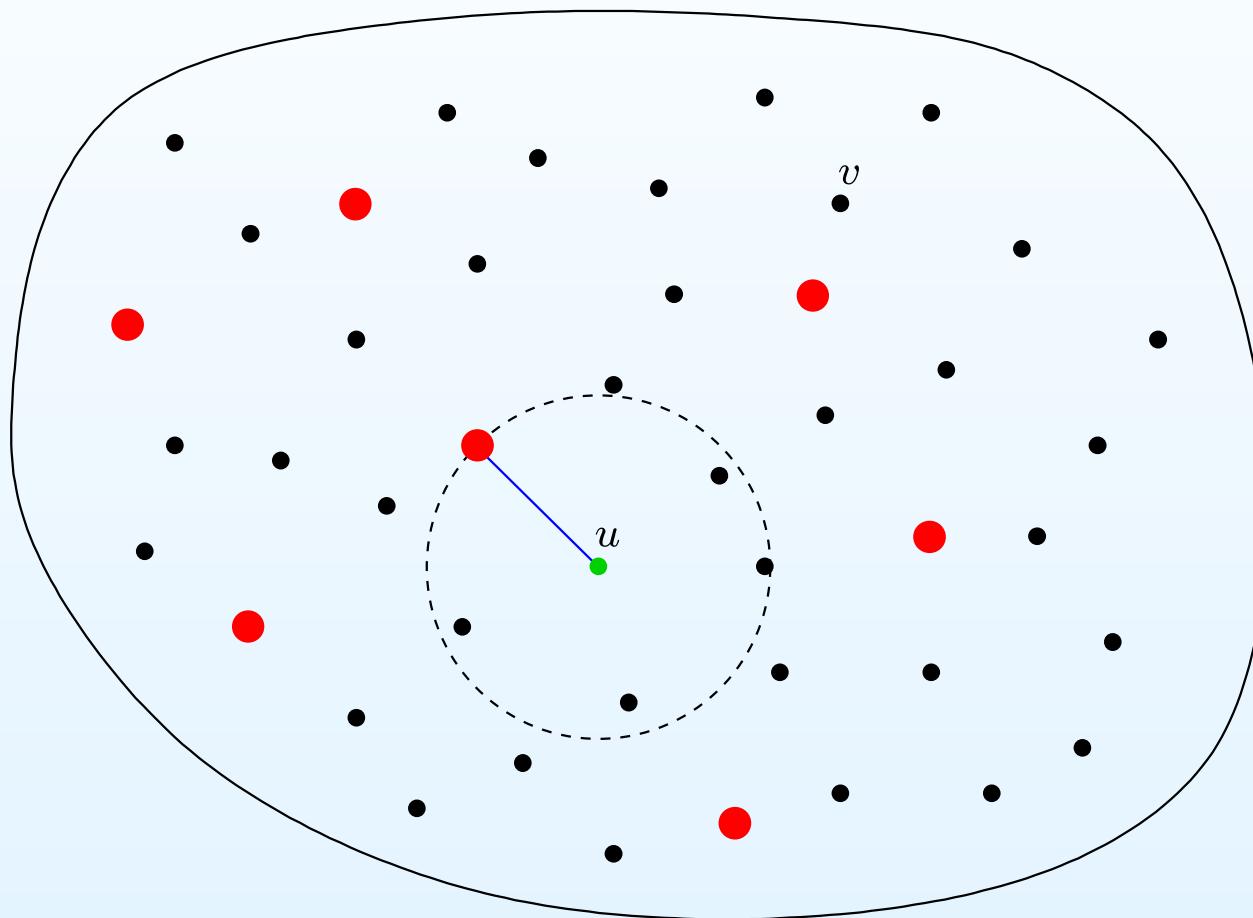
A simple approximate distance oracle

- Consider the opposite case:



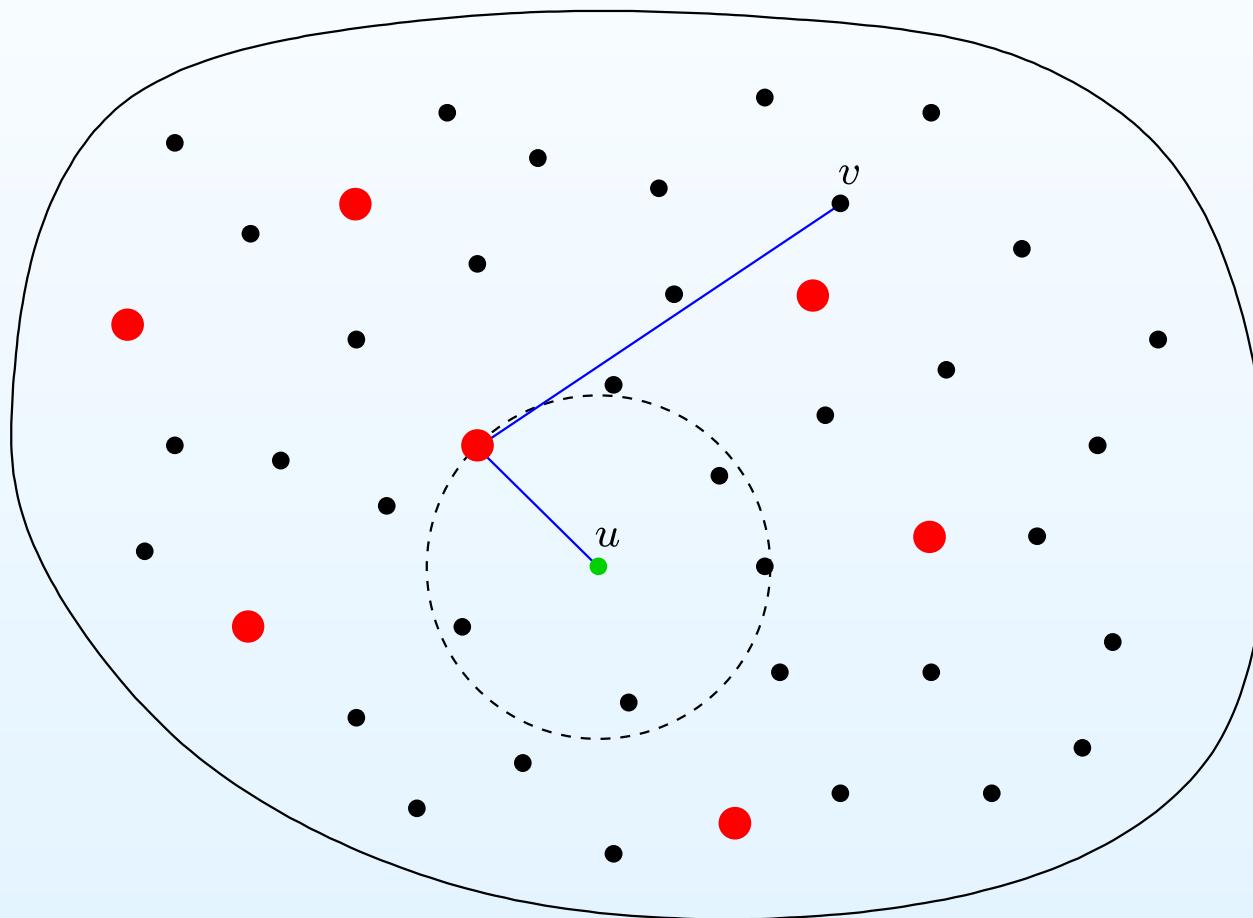
A simple approximate distance oracle

- We stored the nearest sample to u and the distance between them:



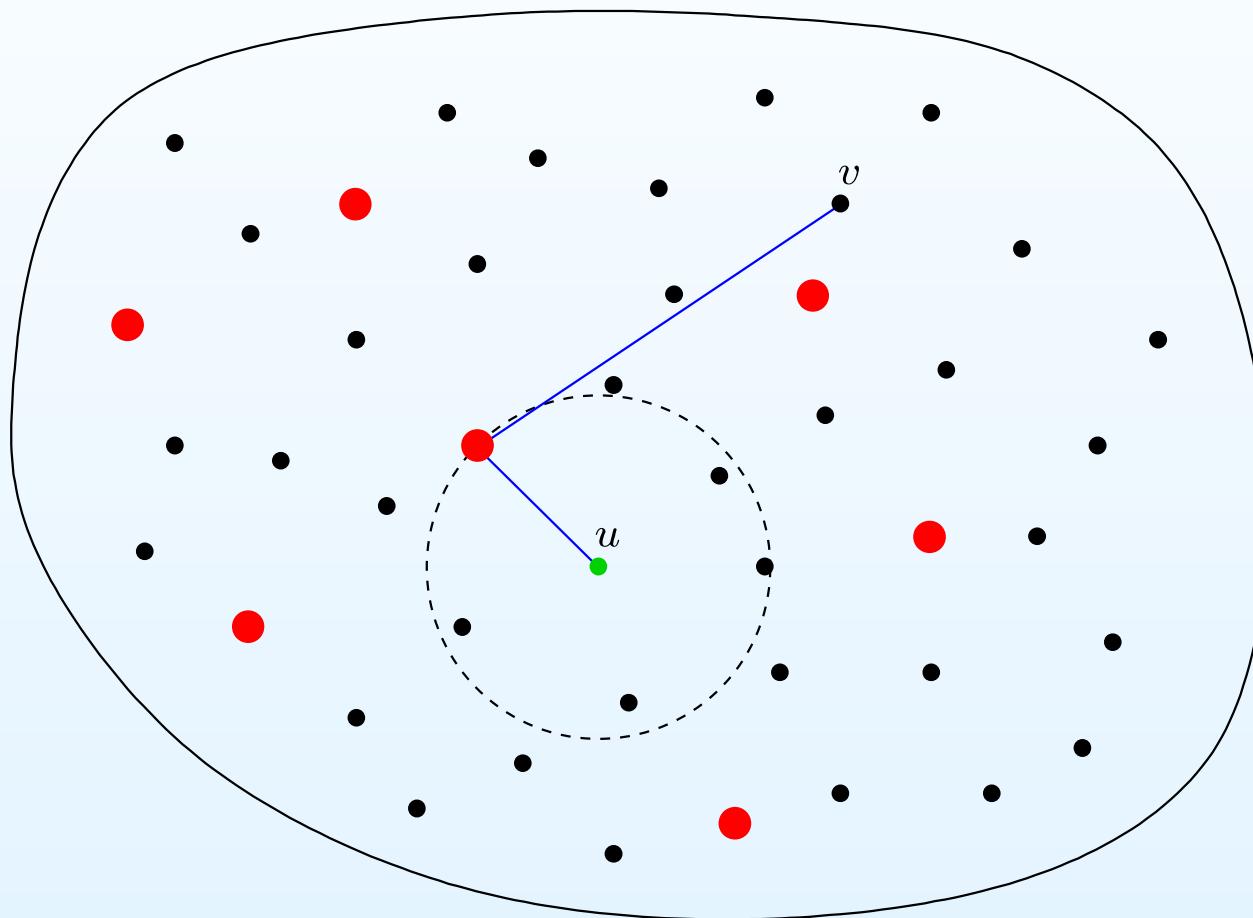
A simple approximate distance oracle

- And we stored the distance between v and this sampled vertex:



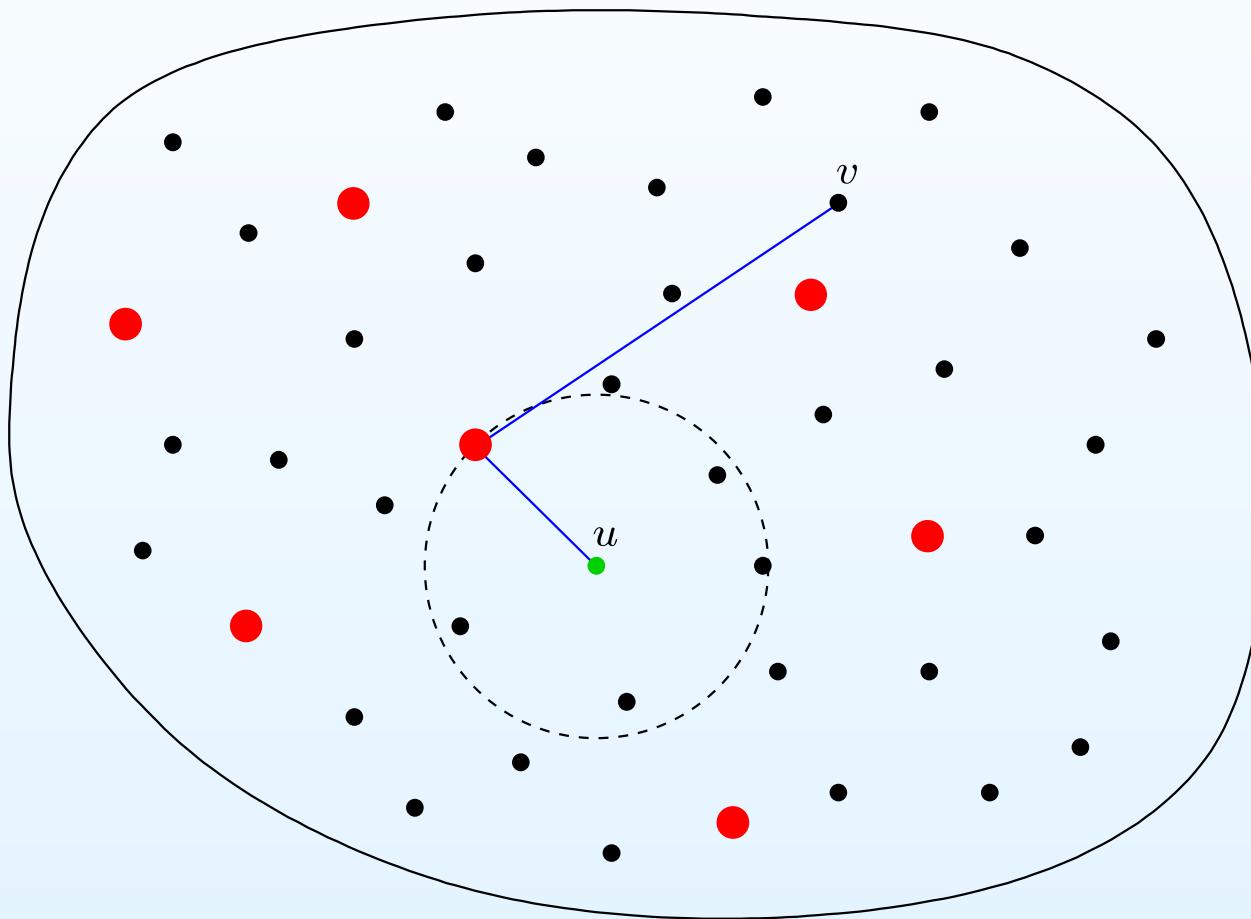
A simple approximate distance oracle

- The oracle returns the sum of these two distances:



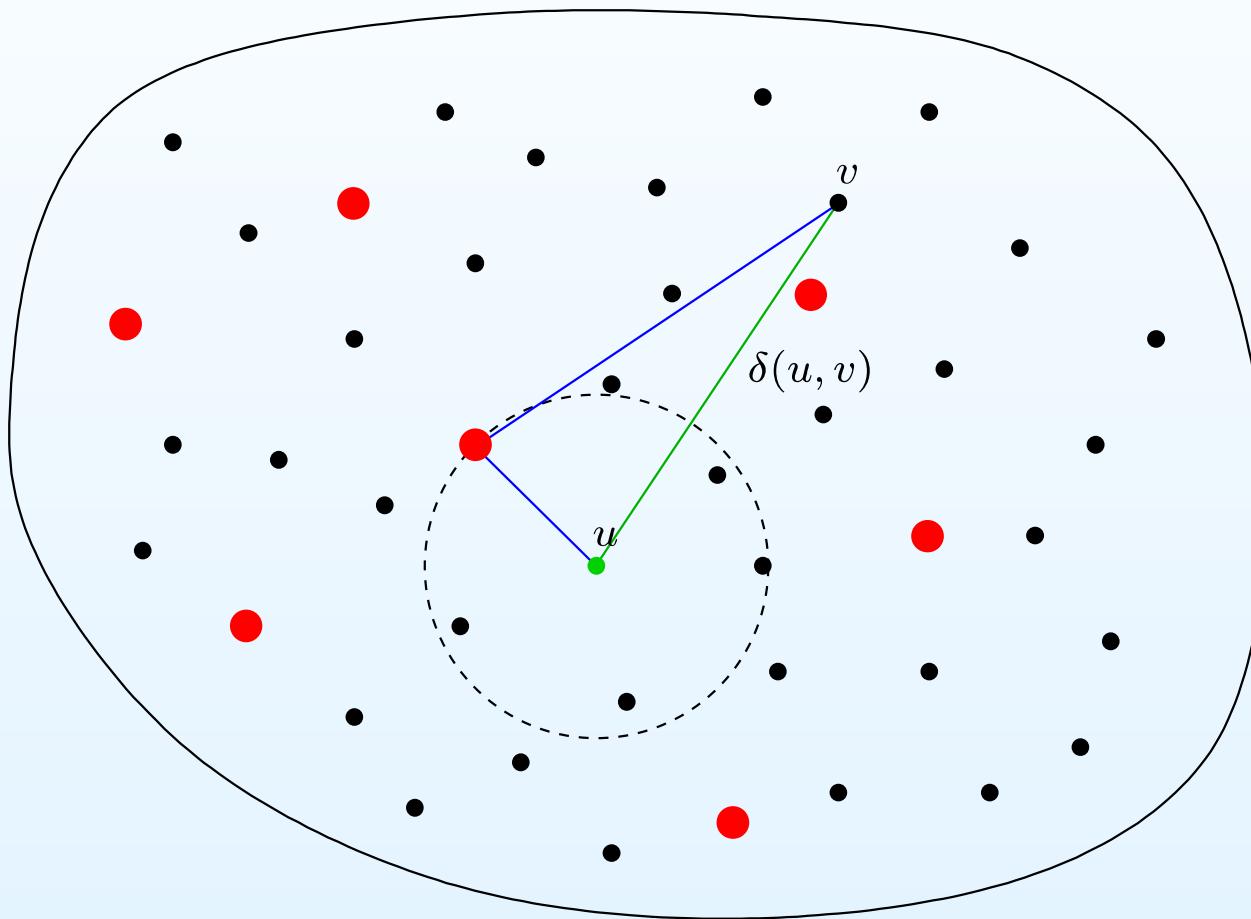
A simple approximate distance oracle

- This estimate has stretch 3 (exercise)



A simple approximate distance oracle

- This estimate has stretch 3 (exercise)



Analyzing space of the approximate distance oracle

- Letting S be the sampled vertices, the oracle stores:

Part I: $\delta(s, v)$ for all $s \in S$ and all $v \in V$

Part II: $\delta(u, v)$ for all $u \in V$ and all v closer to u than u 's nearest vertex in S

Part I: $\delta(s, v)$ for all $s \in S$ and all $v \in V$

- Since each $v \in V$ is sampled independently with probability p ,

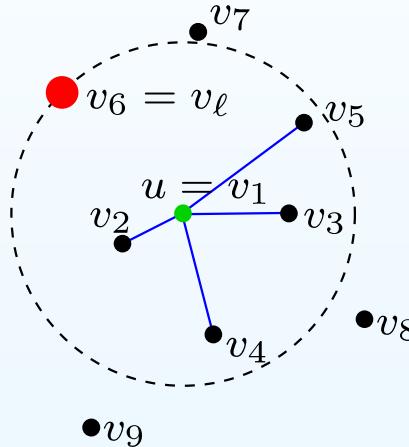
$$E[|S|] = np$$

- Expected space for Part I is thus

$$E[|V||S|] = E[n|S|] = nE[|S|] = O(n^2p)$$

Part II: $\delta(u, v)$, all $u \in V$, v closer to u than u 's nearest S -vertex

- For $u \in V$, consider all $v \in V$ in non-decreasing distance to u :
 v_1, v_2, \dots, v_n where $v_1 = u$
- Let ℓ be the smallest index such that $v_\ell \in S$ (assume ℓ exists)



- The number of distances stored from u is $\ell - 1 < \ell$
- Need to calculate the expected space which is less than $E[\ell]$
- Consider a coin with probability p of heads
- Finding ℓ corresponds to:
 - Flipping the coin for v_1 , then for v_2 , etc.
 - Stopping once it lands on heads; ℓ is the number of coin tosses
- ℓ has the geometric distribution with parameter p so $E[\ell] = 1/p$
- Expected space for Part II: $O(n \cdot E[\ell]) = O(n/p)$

Total space of oracle

- We have shown that Parts I and II of the oracle require a total expected space of

$$O\left(n^2 p + \frac{n}{p}\right)$$

$$n^2 p = \frac{n}{p} \Leftrightarrow p^2 = \frac{1}{n} \Leftrightarrow p = \frac{1}{\sqrt{n}}$$

- This optimal choice of p gives space $O(n^{3/2})$
- Is this space bound measured in words or bits?

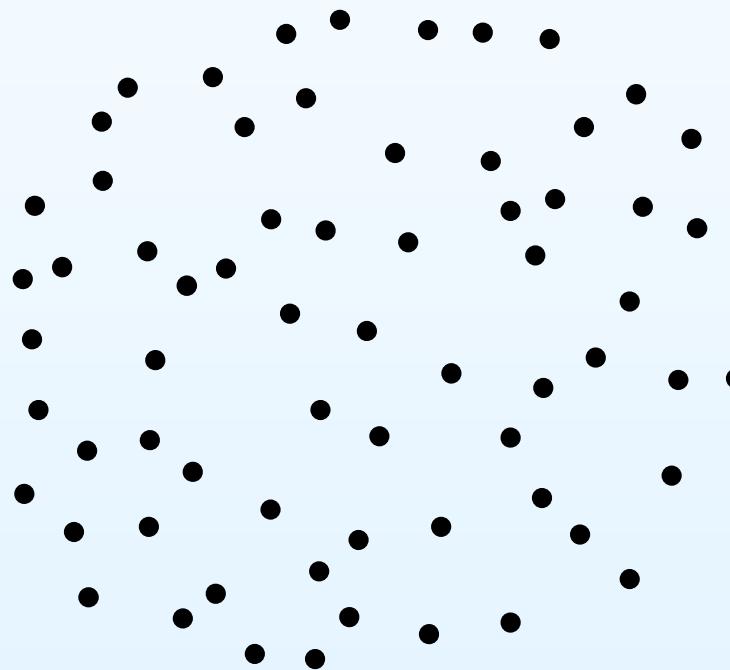
The Thorup-Zwick Oracle

- We have seen a 3-approximate distance oracle requiring $O(n^{3/2})$ space and $O(1)$ query time
- Thorup, Zwick, 2001: for any $k \in \mathbb{N}$, there is an oracle with
 - $O(kn^{1+1/k})$ space,
 - $O(k)$ query time,
 - stretch $2k - 1$
 - $O(kmn^{1/k})$ construction time (not covered in the course)
- Examples:
 - $k = 1$: $O(n^2)$ space and stretch 1 (exercise)
 - $k = 2$: $O(n^{3/2})$ space and stretch 3 (previous slides)
 - $k = 3$: $O(n^{4/3})$ space and stretch 5
- Trade-off between space and stretch is conjectured to be essentially optimal
- We now present and analyze this oracle

Sampling on multiple levels

- Define sampled subsets A_0, \dots, A_k of V as follows:
 - $A_0 = V$
 - For $i = 1, \dots, k - 1$, A_i is obtained from A_{i-1} by sampling each of its vertices independently with some probability p
 - $A_k = \emptyset$
- Example with $k = 3$:

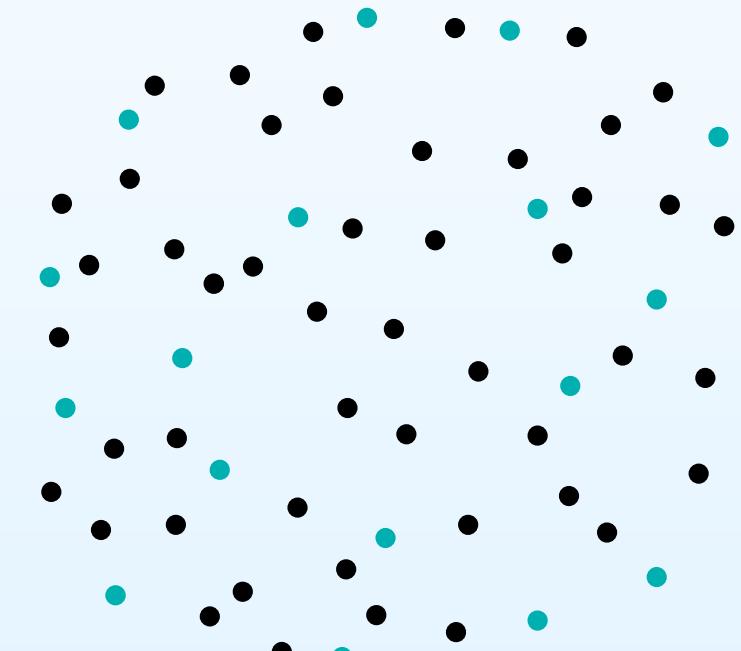
- $\in A_0$
- $\in A_1$
- $\in A_2 = A_{k-1}$



Sampling on multiple levels

- Define sampled subsets A_0, \dots, A_k of V as follows:
 - $A_0 = V$
 - For $i = 1, \dots, k - 1$, A_i is obtained from A_{i-1} by sampling each of its vertices independently with some probability p
 - $A_k = \emptyset$
- Example with $k = 3$:

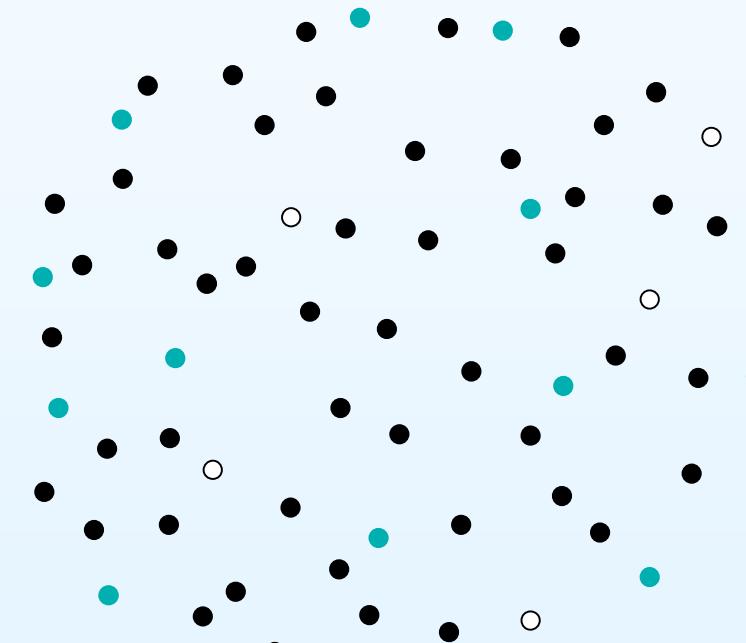
$\bullet \in A_0$
 $\textcolor{teal}{\bullet} \in A_1$
 $\circ \in A_2 = A_{k-1}$



Sampling on multiple levels

- Define sampled subsets A_0, \dots, A_k of V as follows:
 - $A_0 = V$
 - For $i = 1, \dots, k - 1$, A_i is obtained from A_{i-1} by sampling each of its vertices independently with some probability p
 - $A_k = \emptyset$
- Example with $k = 3$:

$\bullet \in A_0$
 $\textcolor{teal}{\bullet} \in A_1$
 $\circ \in A_2 = A_{k-1}$

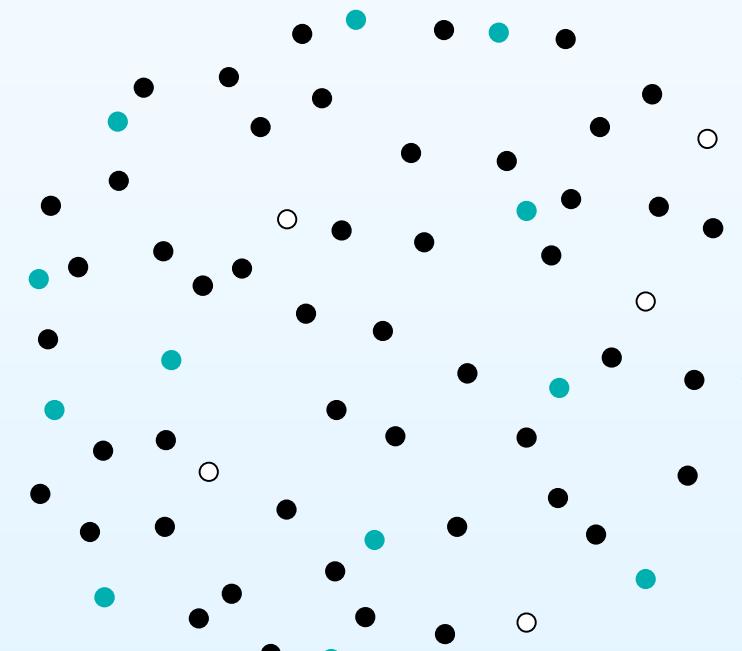


Sampling on multiple levels

- Define sampled subsets A_0, \dots, A_k of V as follows:
 - $A_0 = V$
 - For $i = 1, \dots, k - 1$, A_i is obtained from A_{i-1} by sampling each of its vertices independently with some probability p
 - $A_k = \emptyset$

- Example with $k = 3$:

$\bullet \in A_0$
 $\bullet \in A_1$
 $\circ \in A_2 = A_{k-1}$



- Note that this matches what we did in the 3-approximate distance oracle where $k = 2$

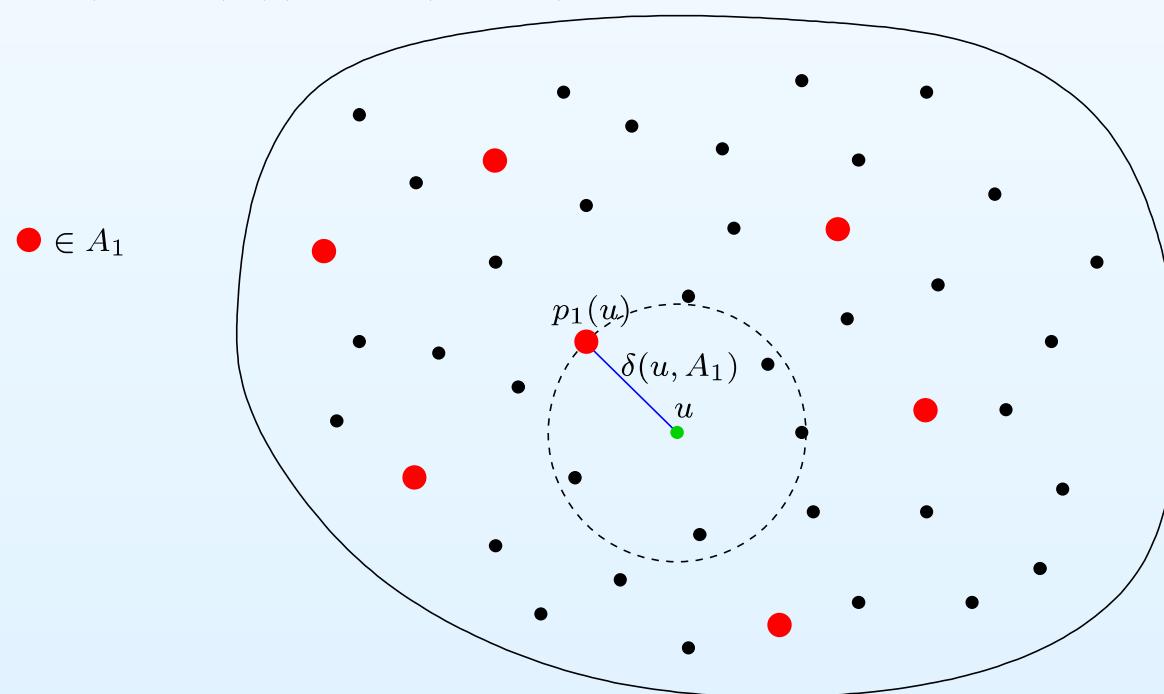
Distance to a Set

- For $u \in V$ and $i = 0, \dots, k$, define

$$\delta(u, A_i) = \min_{v \in A_i} \delta(u, v)$$

where $\delta(u, A_i) = \infty$ if $A_i = \emptyset$

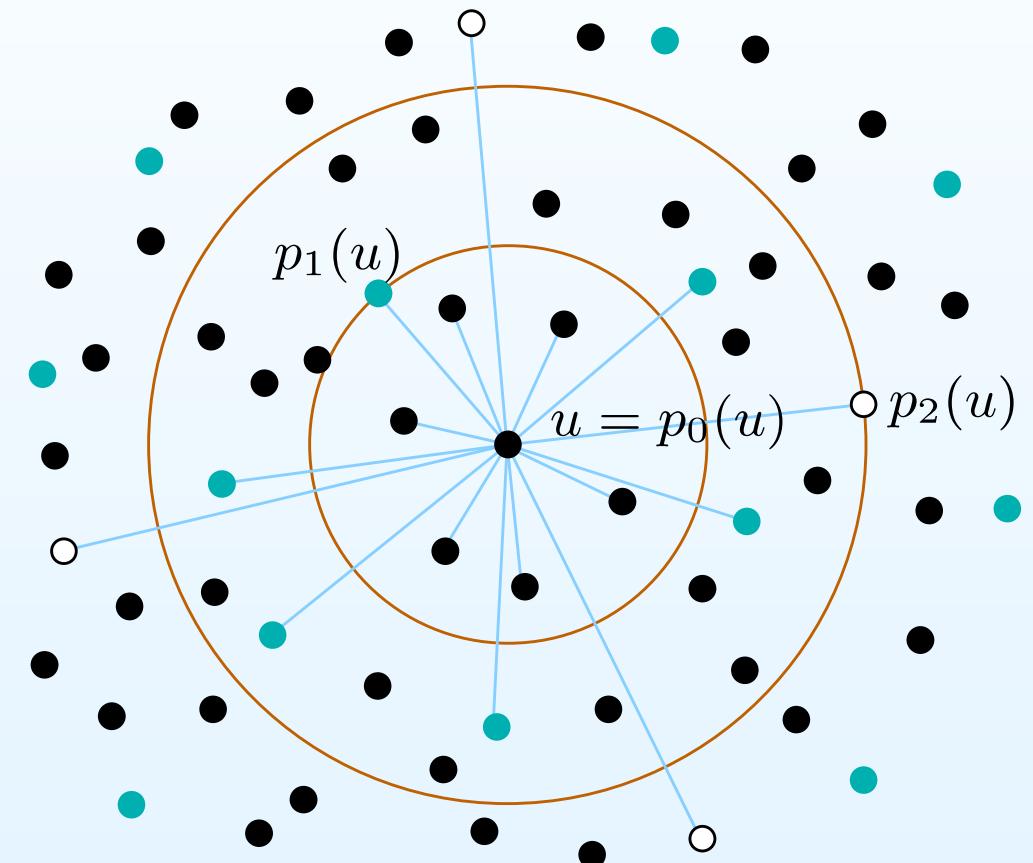
- If $A_i \neq \emptyset$, let $p_i(u)$ be a closest vertex to u in A_i
- Note that $\delta(u, p_i(u)) = \delta(u, A_i)$



Bunches

- *Bunch $B(u)$* contains the vertices in A_i that are closer to u than $\delta(u, A_{i+1})$, $i = 0, \dots, k - 1$
- Example with $k = 3$:

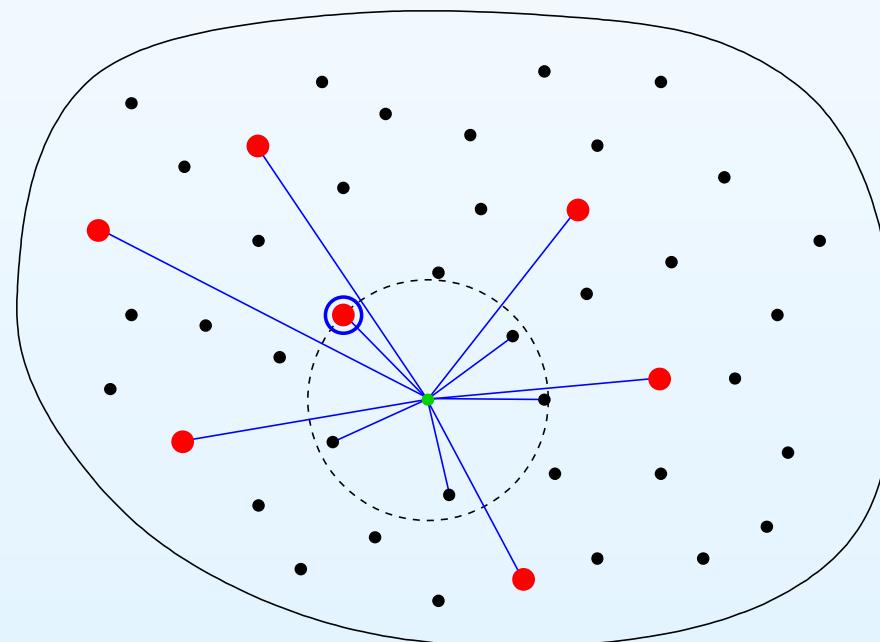
- $\bullet \in A_0$
- $\textcolor{teal}{\bullet} \in A_1$
- $\circ \in A_2 = A_{k-1}$



- Observe $A_k = \emptyset \Rightarrow \delta(u, A_k) = \infty \Rightarrow A_{k-1} \subseteq B(u)$

Information stored by the $(2k - 1)$ -approximate distance oracle

- For each $u \in V$, the oracle stores
 - $B(u)$ (hash table),
 - $p_i(u)$ for $i = 1, 2, \dots, k - 1$,
 - $\delta(u, v)$ for each $v \in B(u)$
- This matches what we store for our 3-approximate distance oracle where $k = 2$:



- Space for oracle is bounded by total space for all bunches

Space used

- We will show that for each $u \in V$

$$E[|B(u)|] = O(k/p + np^{k-1})$$

- Picking $p = n^{-1/k}$ then gives

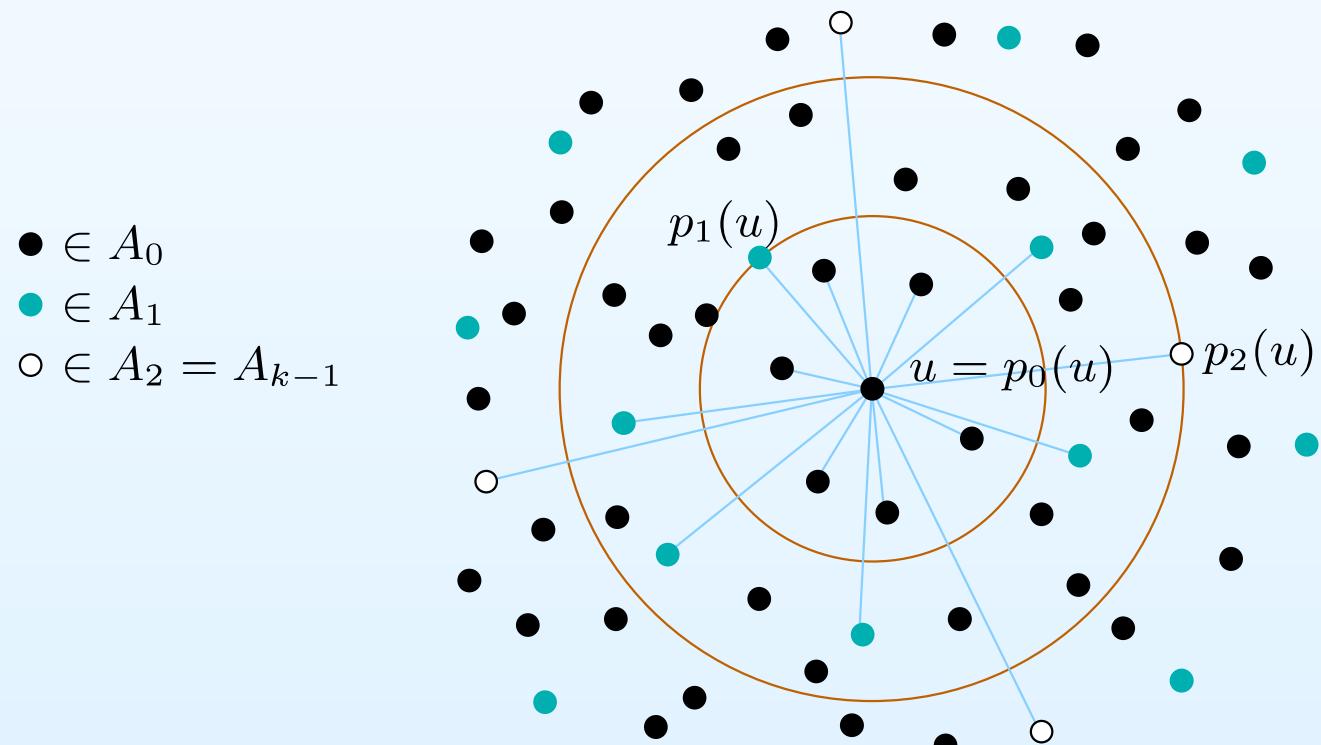
$$\begin{aligned} E[|B(u)|] &= O(k/n^{-1/k} + n \cdot (n^{-1/k})^{k-1}) \\ &= O(kn^{1/k} + n^{1/k}) \\ &= O(kn^{1/k}) \end{aligned}$$

- By linearity of expectation, the total expected space is

$$E \left[\sum_{u \in V} |B(u)| \right] = \sum_{u \in V} E[|B(u)|] = O(kn^{1+1/k})$$

Showing $E[|B(u)|] = O(k/p + np^{k-1})$

- Recall: p is the probability that a vertex in A_i is sampled and included in A_{i+1} , $i = 0, \dots, k - 2$
- Analysis for 3-approximate oracle: the expected number of vertices in A_0 closer to u than $\delta(u, A_1)$ is $O(1/p)$
- Generalizing to $i = 0, \dots, k - 2$, the expected number of vertices in A_i closer to u than $\delta(u, p_{i+1}(u))$ is $O(1/p)$



Showing $E[|B(u)|] = O(k/p + np^{k-1})$

- Recall: p is the probability that a vertex in A_i is sampled and included in A_{i+1} , $i = 0, \dots, k - 2$
- Analysis for 3-approximate oracle: the expected number of vertices in A_0 closer to u than $\delta(u, A_1)$ is $O(1/p)$
- Generalizing to $i = 0, \dots, k - 2$, the expected number of vertices in A_i closer to u than $\delta(u, p_{i+1}(u))$ is $O(1/p)$
- This is a total of $(k - 1)O(1/p) = O(k/p)$
- Bunch $B(u)$ also contains A_{k-1} whose expected size is

$$E[|A_{k-1}|] = np^{k-1}$$

since a vertex of V is in A_{k-1} if and only if it is sampled independently $k - 1$ times in a row with probability p

- Thus, $E[|B(u)|] = O(k/p + np^{k-1})$ as desired

Answering queries

- Pseudo-code for query algorithm $\text{dist}_k(u, v)$:

```
distk(u, v)
1 w ← u, i ← 0
2 while w ∉ B(v)
3   i ← i + 1
4   (u, v) ← (v, u)
5   w ← pi(u)
6 return δ(w, u) + δ(w, v)
```

Answering queries

- Pseudo-code for query algorithm $\text{dist}_k(u, v)$:

```
distk(u, v)
1 w ← u, i ← 0
2 while w ∉ B(v)
3   i ← i + 1
4   (u, v) ← (v, u)
5   w ← pi(u)
6 return δ(w, u) + δ(w, v)
```

$p_4(u)$ •

• $p_3(v)$

$p_2(u)$ •

• $p_1(v)$

$u = p_0(u)$ •

Answering queries

- Pseudo-code for query algorithm $\text{dist}_k(u, v)$:

```
distk(u, v)
1 w ← u, i ← 0
2 while w ∉ B(v)
3   i ← i + 1
4   (u, v) ← (v, u)
5   w ← pi(u)
6 return δ(w, u) + δ(w, v)
```

$p_4(u)$ •

• $p_3(v)$

$p_2(u)$ •

• $p_1(v)$

$u = p_0(u) \circ w$

Answering queries

- Pseudo-code for query algorithm $\text{dist}_k(u, v)$:

```
distk(u, v)
1 w ← u, i ← 0
2 while w ∉ B(v)
3   i ← i + 1
4   (u, v) ← (v, u)
5   w ← pi(u)
6 return δ(w, u) + δ(w, v)
```

$p_4(u)$ ●

● $p_3(v)$

$p_2(u)$ ●

$w \circ p_1(v)$

$u = p_0(u) \circ$

Answering queries

- Pseudo-code for query algorithm $\text{dist}_k(u, v)$:

```
distk(u, v)
1 w ← u, i ← 0
2 while w ∉ B(v)
3   i ← i + 1
4   (u, v) ← (v, u)
5   w ← pi(u)
6 return δ(w, u) + δ(w, v)
```

$p_4(u)$ ●

● $p_3(v)$

$p_2(u) \circ w$

○ $p_1(v)$

$u = p_0(u) \circ$

Answering queries

- Pseudo-code for query algorithm $\text{dist}_k(u, v)$:

```
distk(u, v)
1 w ← u, i ← 0
2 while w ∉ B(v)
3   i ← i + 1
4   (u, v) ← (v, u)
5   w ← pi(u)
6 return δ(w, u) + δ(w, v)
```

$$p_4(u) \bullet$$

$$w \circ p_3(v)$$

$$p_2(u) \circ$$

$$\circ p_1(v)$$

$$u = p_0(u) \circ$$

Answering queries

- Pseudo-code for query algorithm $\text{dist}_k(u, v)$:

```
distk(u, v)
1 w ← u, i ← 0
2 while w ∉ B(v)
3   i ← i + 1
4   (u, v) ← (v, u)
5   w ← pi(u)
6 return δ(w, u) + δ(w, v)
```

$$p_4(u) \circ w \\ \circ p_3(v)$$

$$p_2(u) \circ \\ \circ p_1(v)$$

$$u = p_0(u) \circ$$

Answering queries

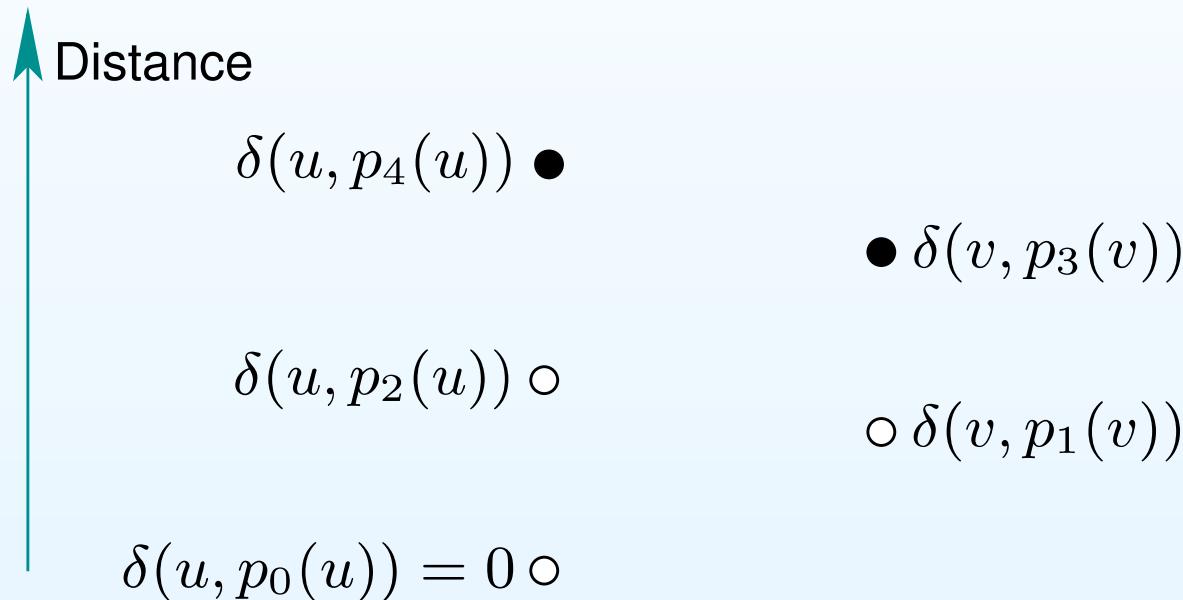
- Pseudo-code for query algorithm $\text{dist}_k(u, v)$:

```
distk(u, v)
1 w ← u, i ← 0
2 while w ∉ B(v)
3   i ← i + 1
4   (u, v) ← (v, u)
5   w ← pi(u)
6 return δ(w, u) + δ(w, v)
```

- Exercises:
 - Show that the query algorithm must terminate no later than at index $i = k - 1$
 - Show that in Line 6, both distances $\delta(w, u)$ and $\delta(w, v)$ are stored by the data structure

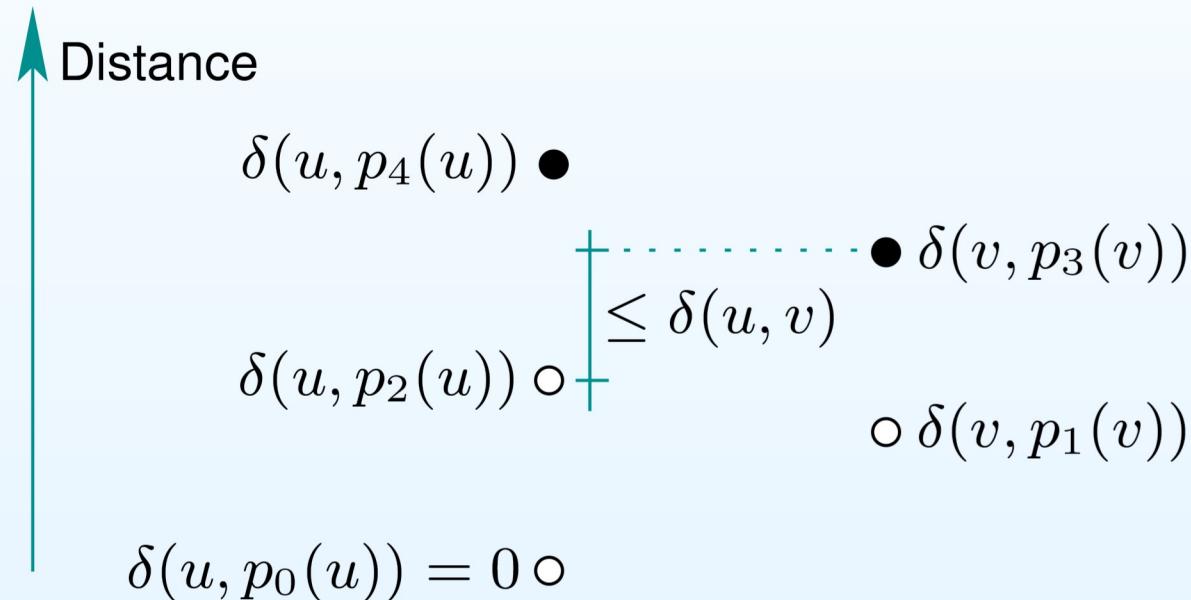
Bounding the increase in stretch per iteration

- We will show $\delta(v, p_{i+1}(v)) \leq \delta(u, p_i(u)) + \delta(u, v)$ for each even i that satisfies the condition $w \notin B(v)$ in the while loop



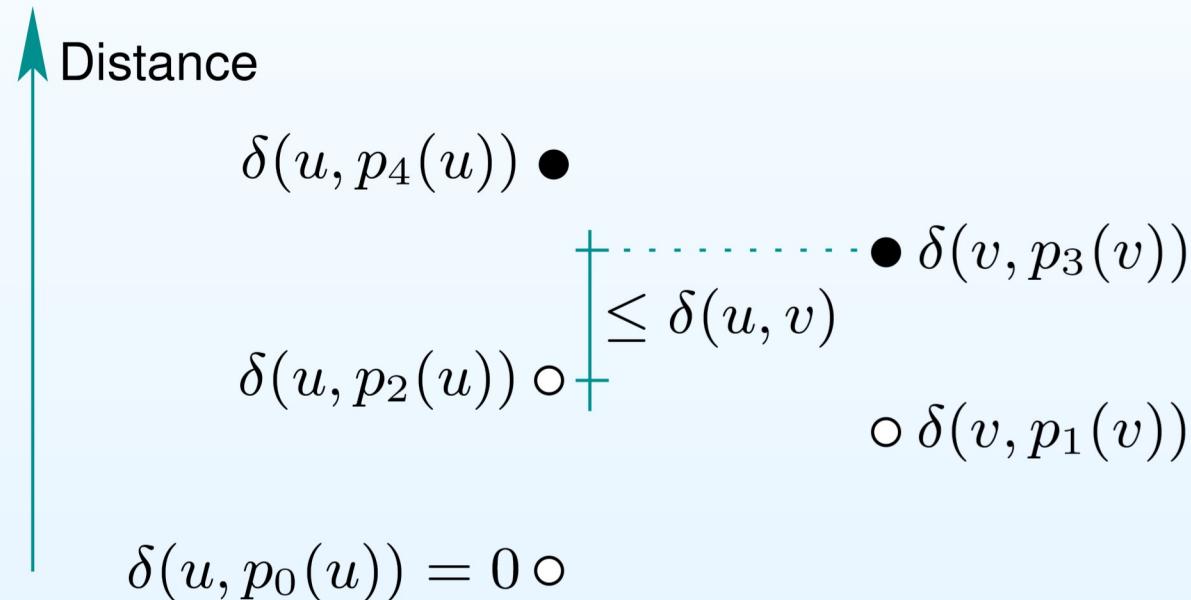
Bounding the increase in stretch per iteration

- We will show $\delta(v, p_{i+1}(v)) \leq \delta(u, p_i(u)) + \delta(u, v)$ for each even i that satisfies the condition $w \notin B(v)$ in the while loop



Bounding the increase in stretch per iteration

- We will show $\delta(v, p_{i+1}(v)) \leq \delta(u, p_i(u)) + \delta(u, v)$ for each even i that satisfies the condition $w \notin B(v)$ in the while loop



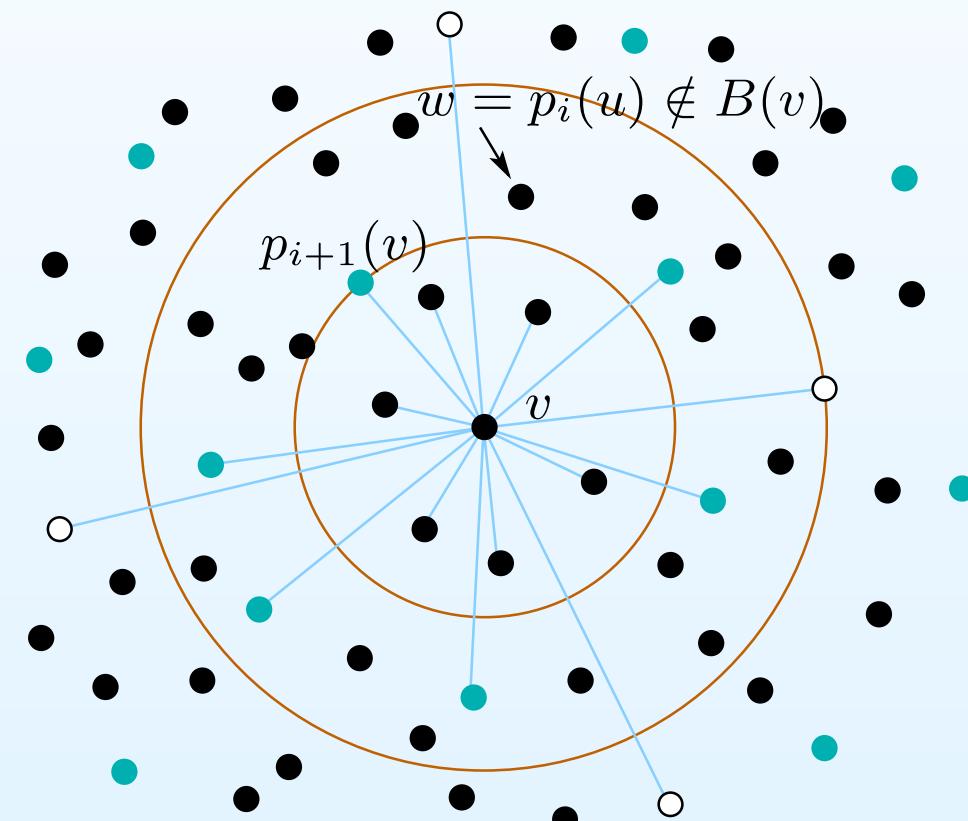
- By a symmetric argument, $\delta(u, p_{i+1}(u)) \leq \delta(v, p_i(v)) + \delta(u, v)$ for each odd i satisfying the condition in the while loop
- Every gap in the figure satisfying the condition is thus $\leq \delta(u, v)$

Showing $\delta(v, p_{i+1}(v)) \leq \delta(u, p_i(u)) + \delta(u, v)$

- While loop condition: $w = p_i(u) \notin B(v)$
- Since $B(v)$ contains all vertices of A_i closer to v than $p_{i+1}(v)$,

$$\delta(v, p_{i+1}(v)) \leq \delta(v, p_i(u))$$

- $\in A_0$
- $\in A_1$
- $\in A_2 = A_{k-1}$



Showing $\delta(v, p_{i+1}(v)) \leq \delta(u, p_i(u)) + \delta(u, v)$

- While loop condition: $w = p_i(u) \notin B(v)$
- Since $B(v)$ contains all vertices of A_i closer to v than $p_{i+1}(v)$,

$$\delta(v, p_{i+1}(v)) \leq \delta(v, p_i(u))$$

- By the triangle inequality,

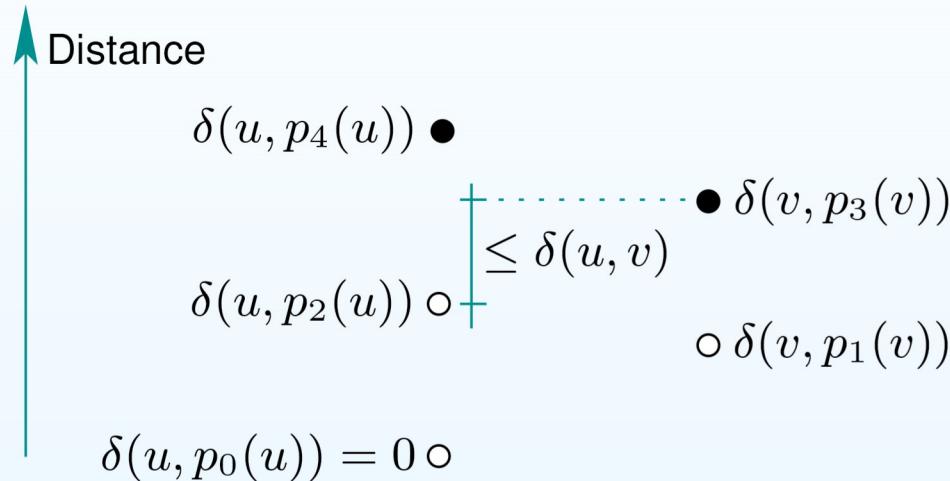
$$\delta(v, p_i(u)) \leq \delta(v, u) + \delta(u, p_i(u))$$

- Combining these inequalities gives the desired:

$$\delta(v, p_{i+1}(v)) \leq \delta(u, p_i(u)) + \delta(u, v)$$

Showing the $(2k - 1)$ -stretch bound

- We have shown the following for all gaps satisfying the while-loop condition:



- If, e.g., $w = p_i(u)$ at termination, we have $i \leq k - 1$ and thus:

$$\overbrace{\delta(w, u) + \delta(w, v)}^{\text{return value}} \leq \delta(w, u) + \delta(w, v) + \delta(u, v) \quad (\text{triangle ineq.})$$
$$= 2\delta(w, u) + \delta(u, v)$$
$$\leq 2i\delta(u, v) + \delta(u, v) \quad (\text{gap bounds above})$$
$$\leq 2(k - 1)\delta(u, v) + \delta(u, v) \quad (i \leq k - 1)$$
$$= (2k - 1)\delta(u, v)$$

Summary and Conclusion

- We have shown that the approximate distance oracle of Thorup and Zwick has
 - $2k - 1$ stretch
 - $O(k)$ query time
 - $O(kn^{1+1/k})$ space
- The space bound is measured in words, not in bits
- For stretch $2k - 1$, there is a conjectured lower bound of $\Omega(n^{1+1/k})$ bits (girth conjecture)
- In practice:
 - there are much better data structures for road networks
 - Distances returned are exact (stretch 1)
 - These data structure leverage special properties of road networks that do not generalize to arbitrary graphs