Distance Oracles

Goal

- A compact data structure
- Can answer queries of the form dist?(u,v)

(preprocessing time is also interesting)

The trivial extreme cases

- Pre-compute all answers

Size: $O(n^2)$, Query: $O(1)$

- Run Dijkstra/BFS at query time

Size: $O(m)$, Query: $O(m + n \log(n))$

An immediate lower bound

$$\#\, graphs\, on\, \{1,2,\ldots,n\} = 2^{\binom{n}{2}}$$

Data structures for different graphs must be different:

If $(u,v) \in G_1$ but $(u,v) \notin G_2$ then

$\delta_{G_1}(u,v) = 1$ and $\delta_{G_2}(u,v) \neq 1$

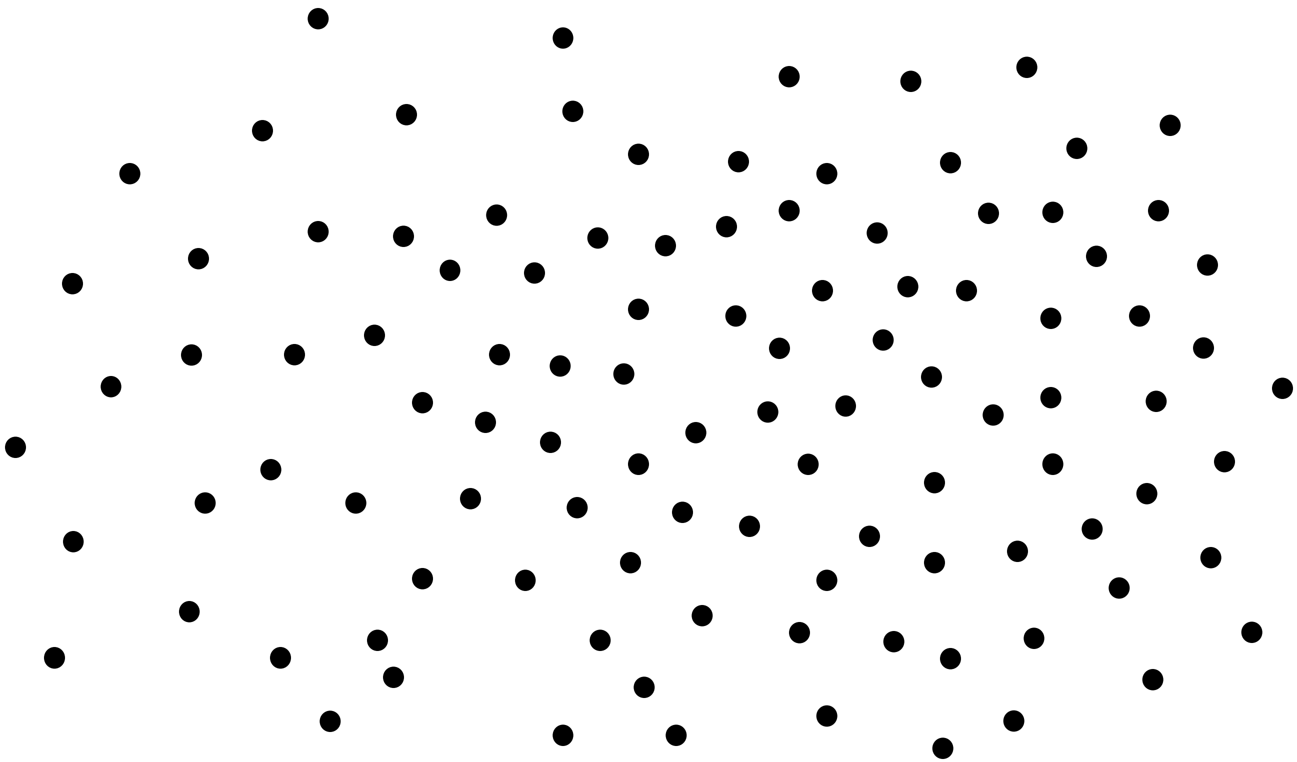
➔ Some graphs must have a data structure of $\Omega(n^2)$ bits

Allow approximate answers

- Stretch (2k-1) for some k > 1
- Lets start with k=2, so we want distance estimates such that:

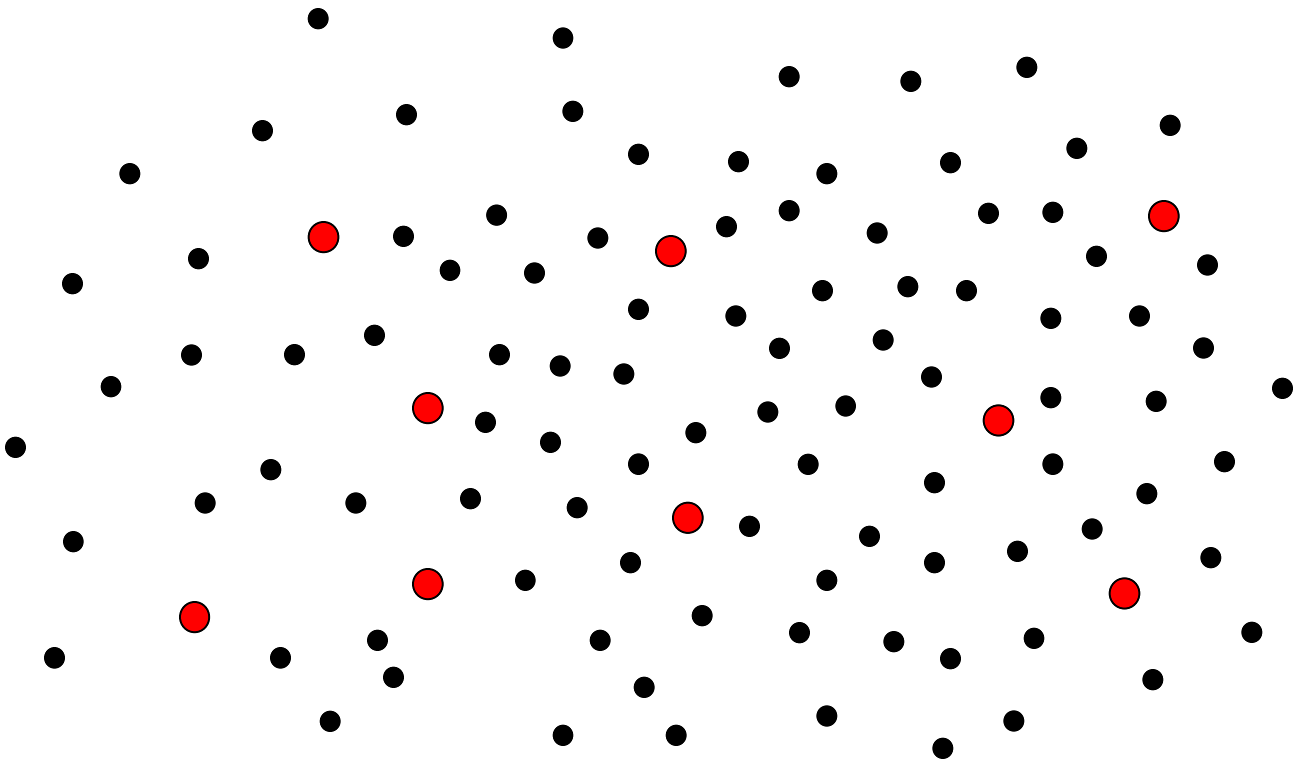
$$\boldsymbol{\delta(u,v) \leq \mathrm{dist?}(u,v) \leq 3\delta(u,v)}$$

Sample centers



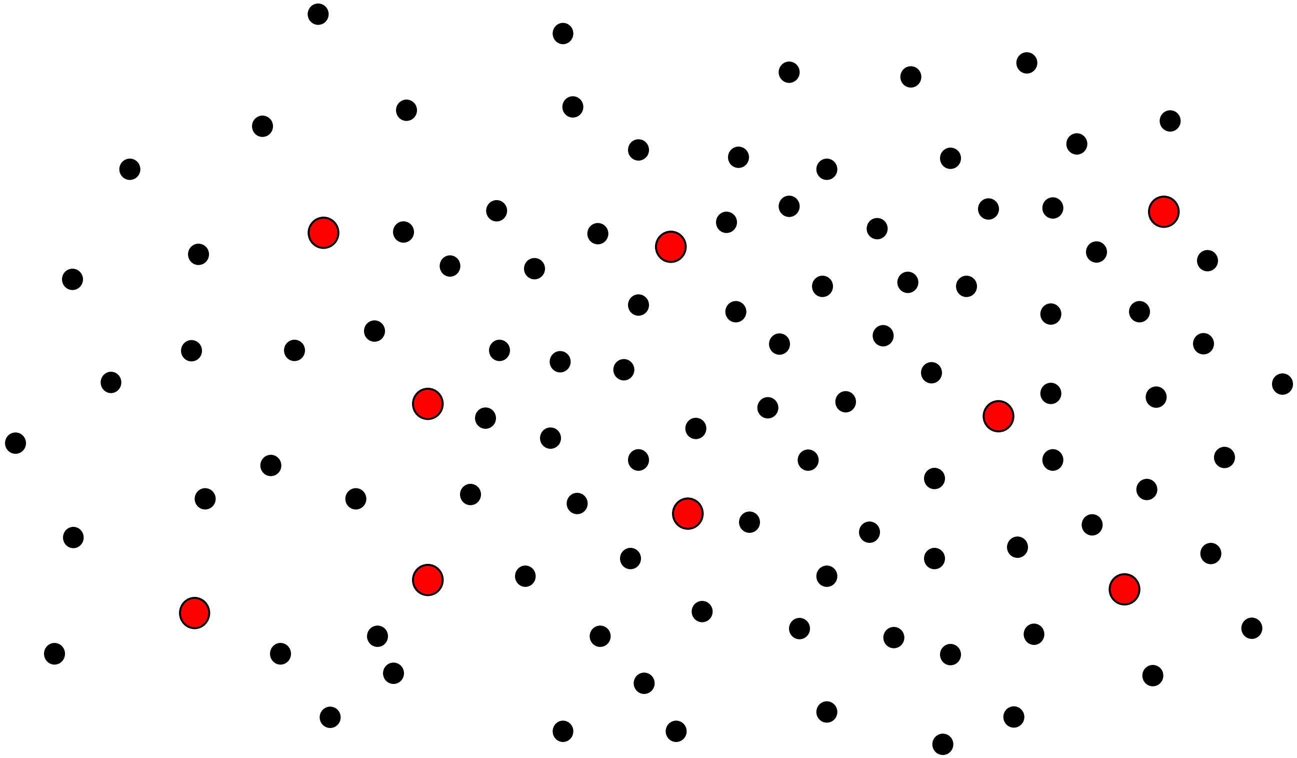
$A_0 \leftarrow V$; $A_1 \leftarrow sample(A_0, n^{-1/2})$;

Sample centers



$A_0 \leftarrow V$; $A_1 \leftarrow \text{sample}(A_0, n^{-1/2})$;

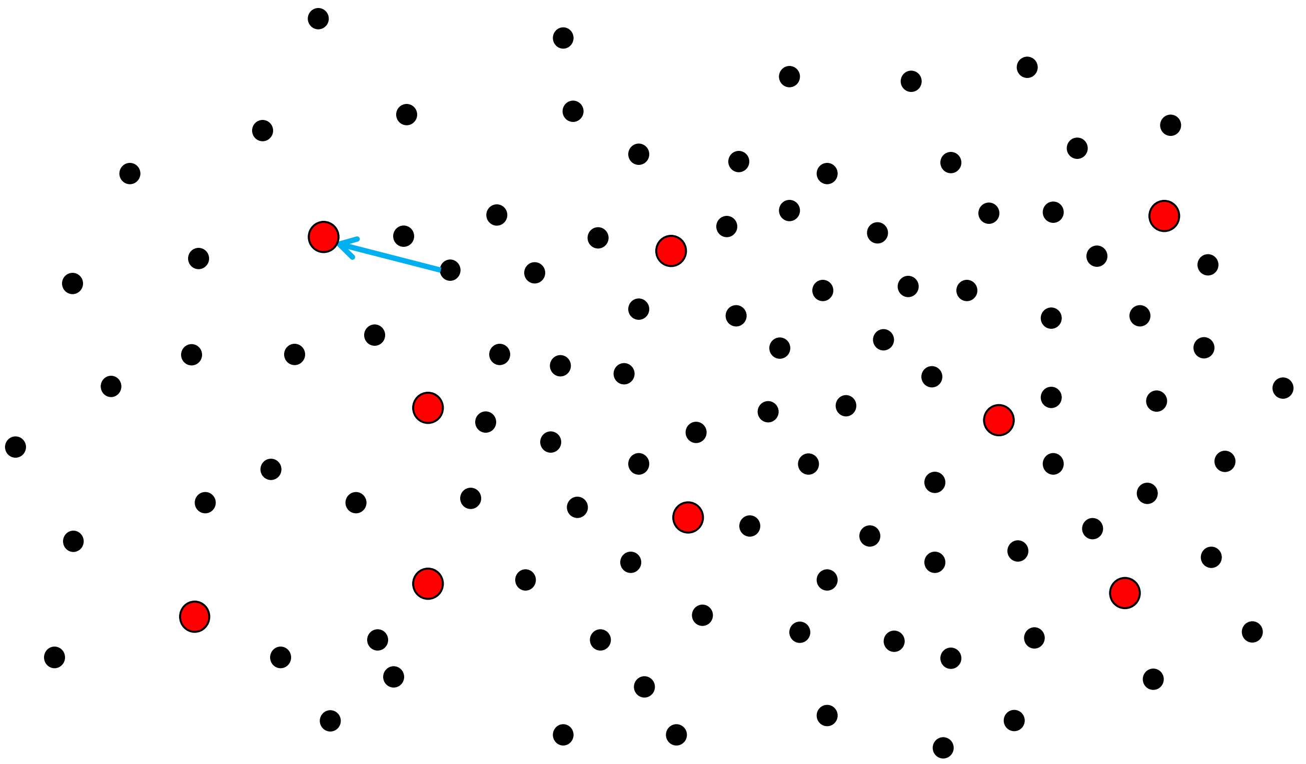
Sample centers



$A_1 \leftarrow \text{sample}(V, n^{-1/2})$;

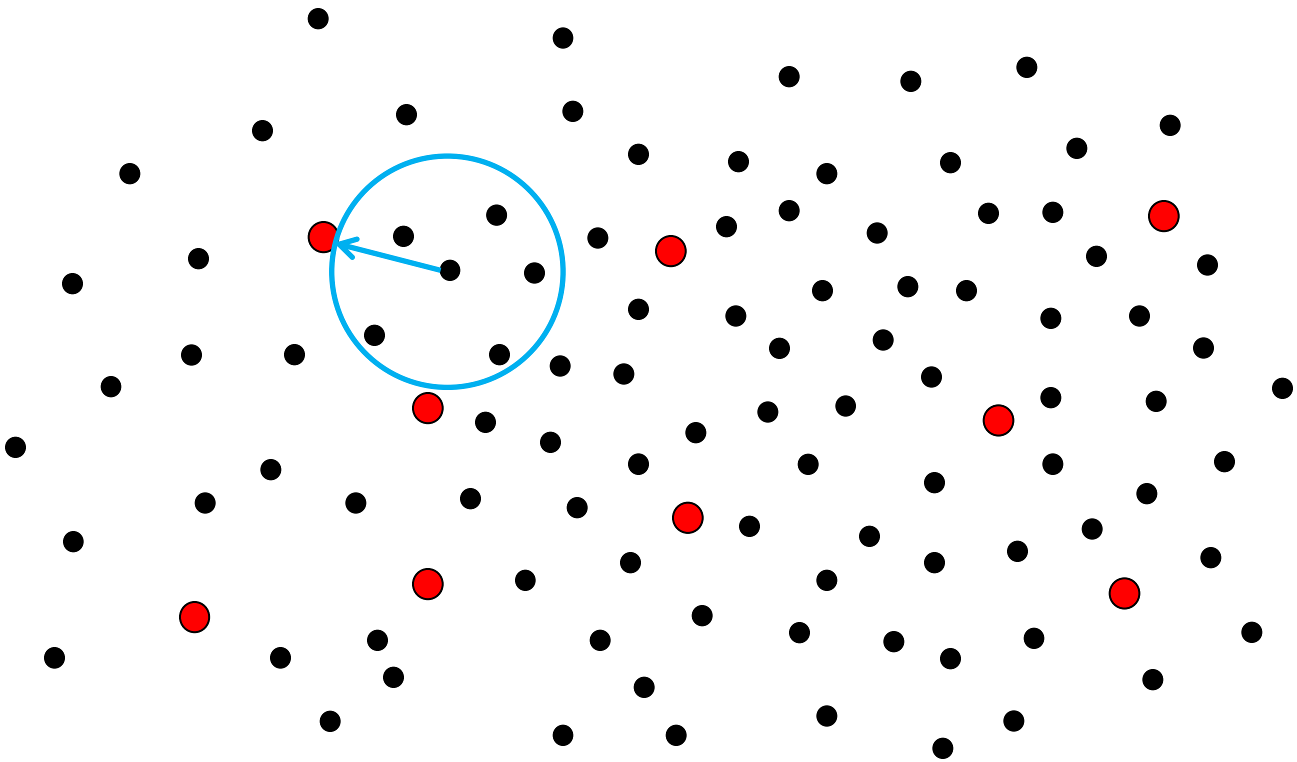
Note that $E(|A_1|) = \sqrt{n}$

Information per vertex v



$p_1(v)$ = The closest center to v

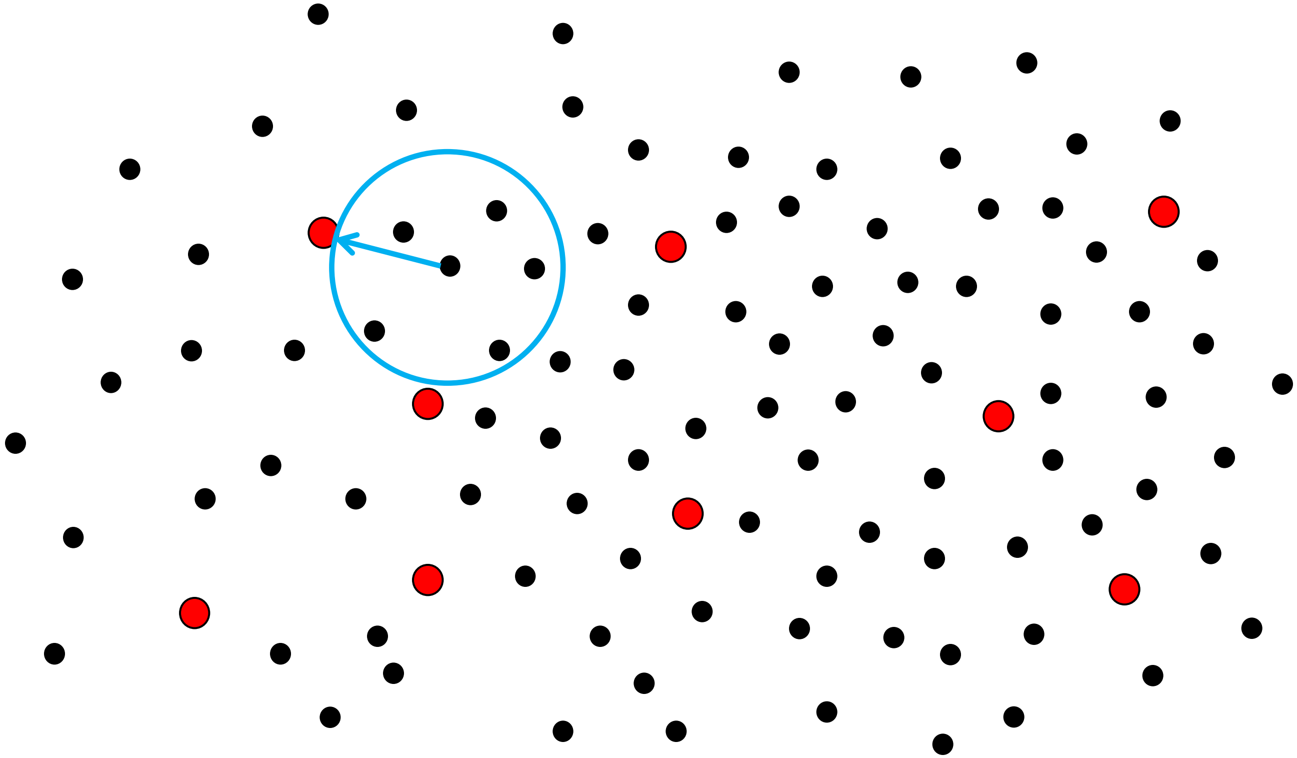
Information per vertex v



$$B_0(v) \leftarrow \{ w \in V - A_1 \mid \delta(v,w) < \delta(v, p_1(v)) \}$$

Note that $E(|B_0(v)|) \leq \sqrt{n}$

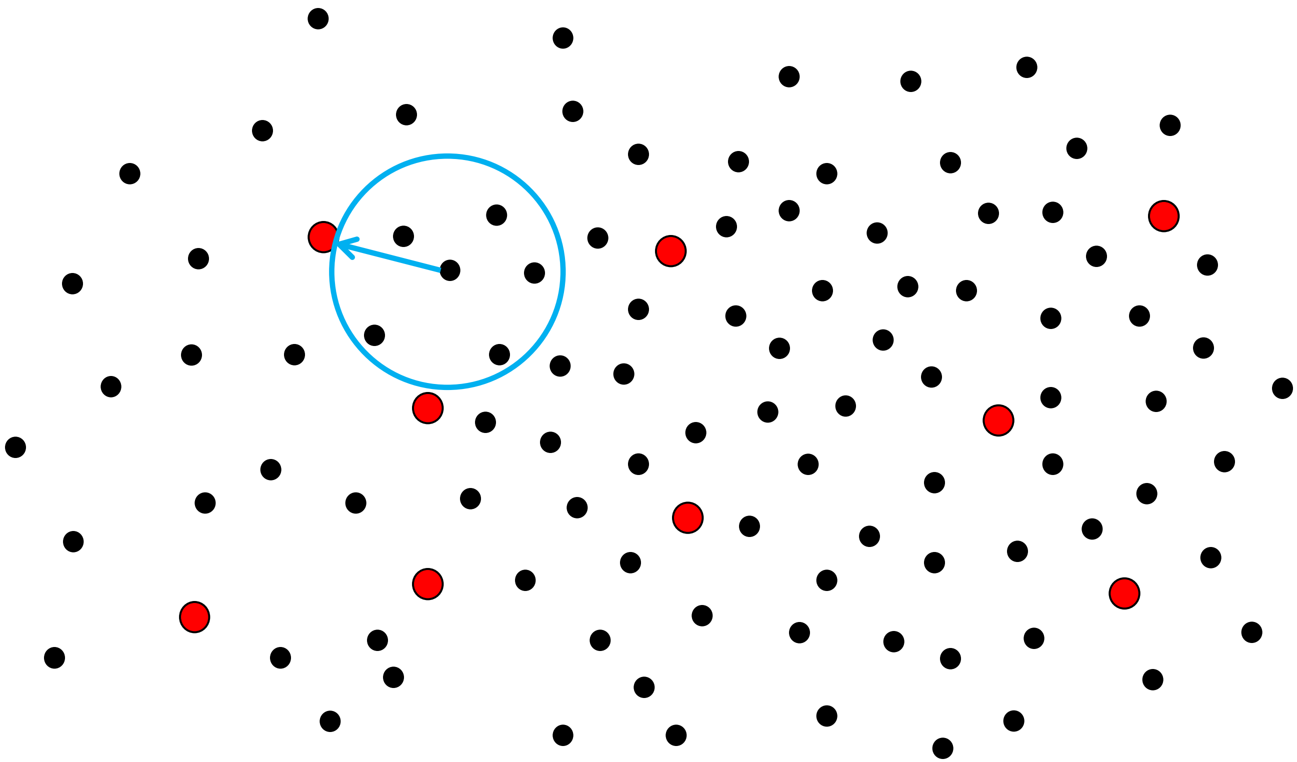
Information per vertex v



$$B_0(v) \leftarrow \{w \in V - A_1 \mid \delta(v,w) < \delta(v, p_1(v))\}$$

$$B_1(v) \leftarrow A_1 \qquad B(v) \leftarrow B_0(v) \cup B_1(v)$$

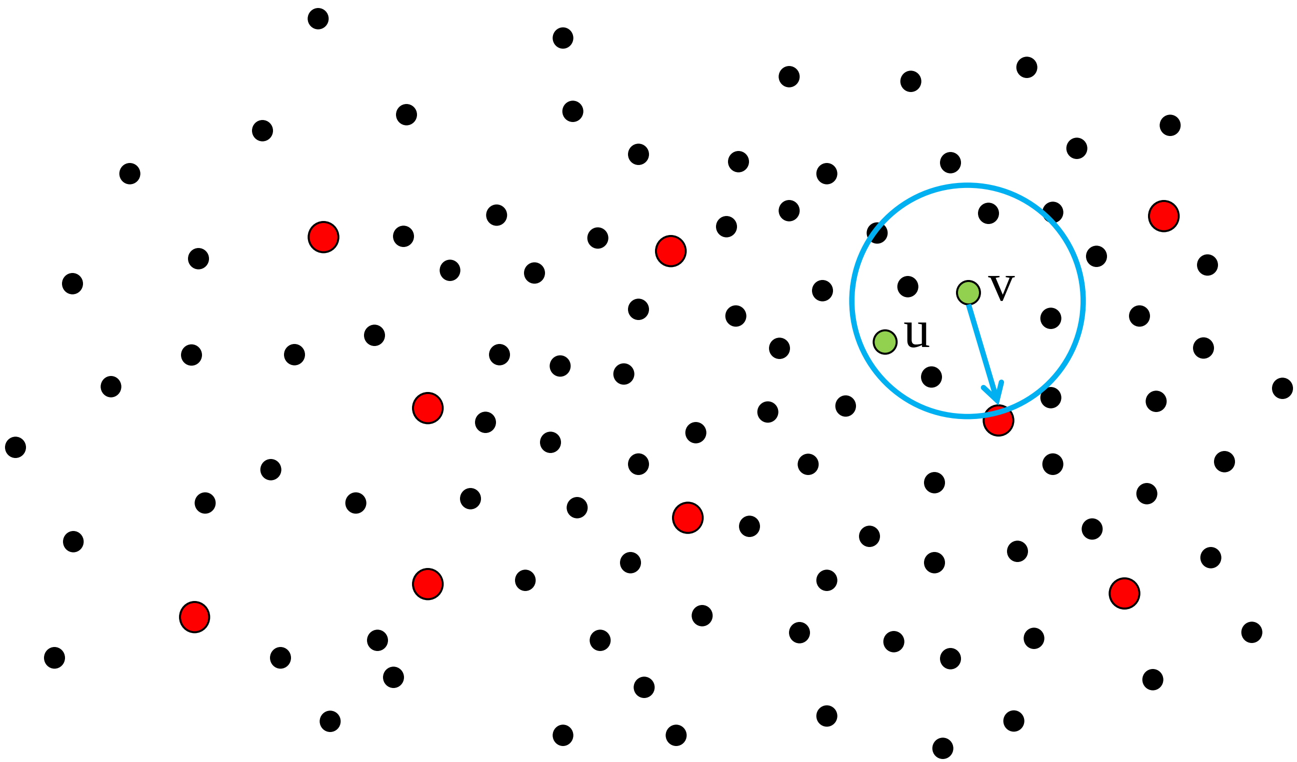
Information per vertex v



Save B(v) in a hash table together with δ(v,u) for every u ∈ B(v)

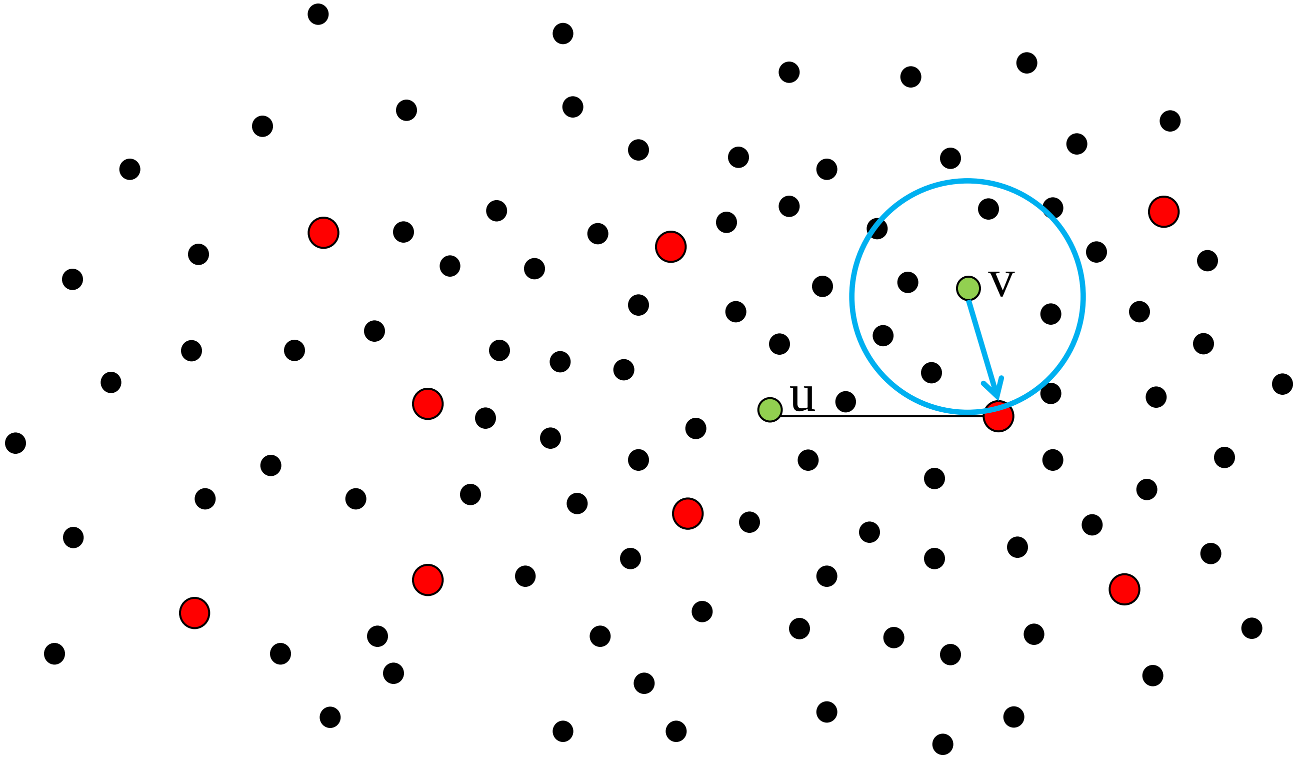
Save $p_1(v)$ and $\delta(v,p_1(v))$ $O\left(n\sqrt{n}\right)$ space

Query



if u ∈ B(v) then return δ(v,u)

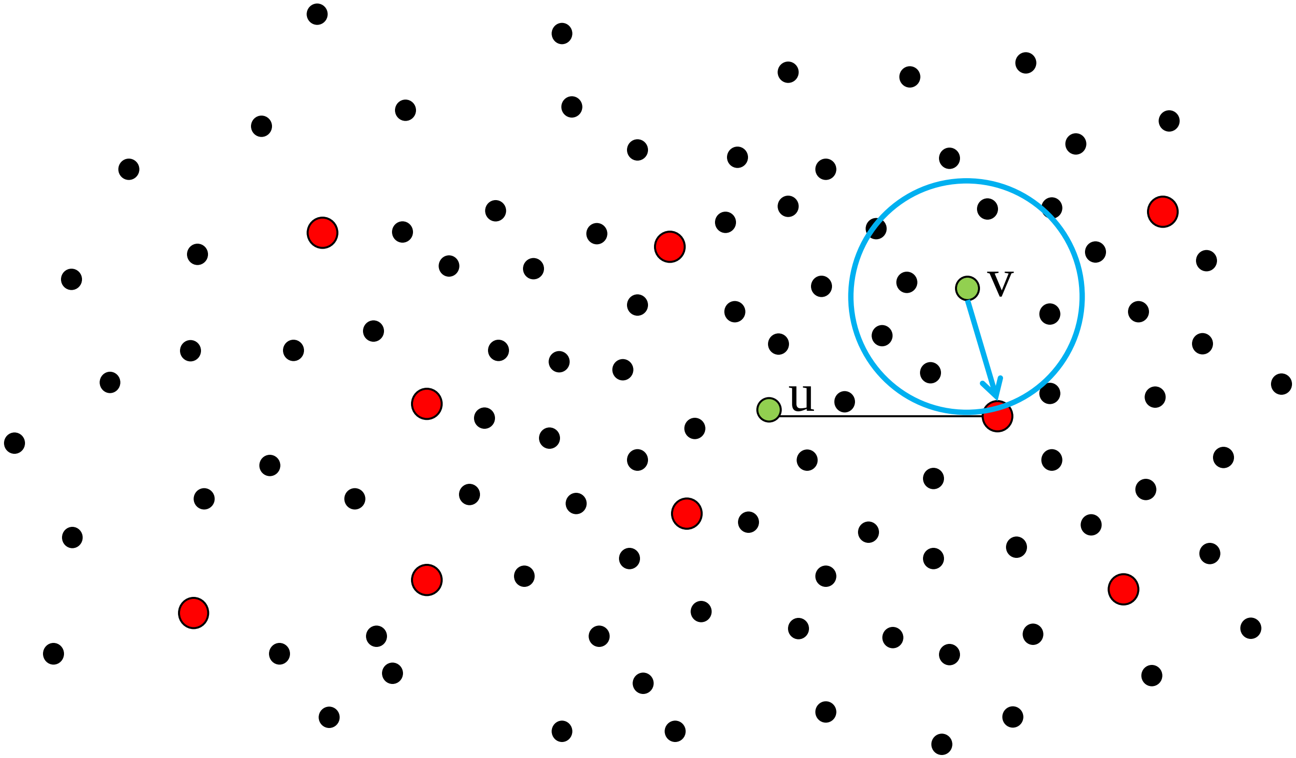
Query



if $u \in B(v)$ then return $\delta(v,u)$

else return $\delta(v,p_1(v)) + \delta(u,p_1(v))$

Query



$$\color{red}{\delta(v,p_1(v)) + \delta(u,p_1(v))} \le \delta(v,u) + \delta(u,p_1(v))$$

$$\le \delta(v,u) + \delta(u,v) + \delta(v,p_1(v))$$

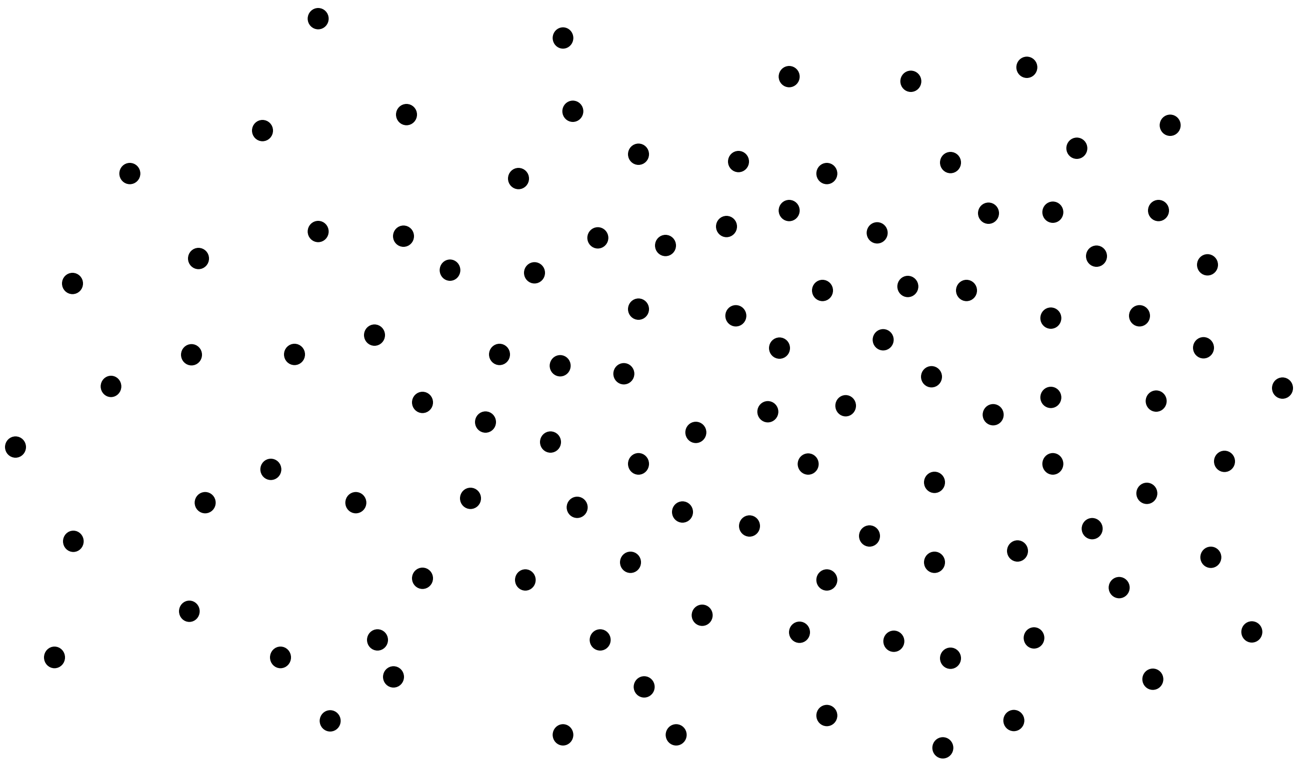
$$\color{red}{\le \delta(v,u) + \delta(u,v) + \delta(v,u)}$$

Larger k

- Stretch (2k-1) for some $k \geq 1$

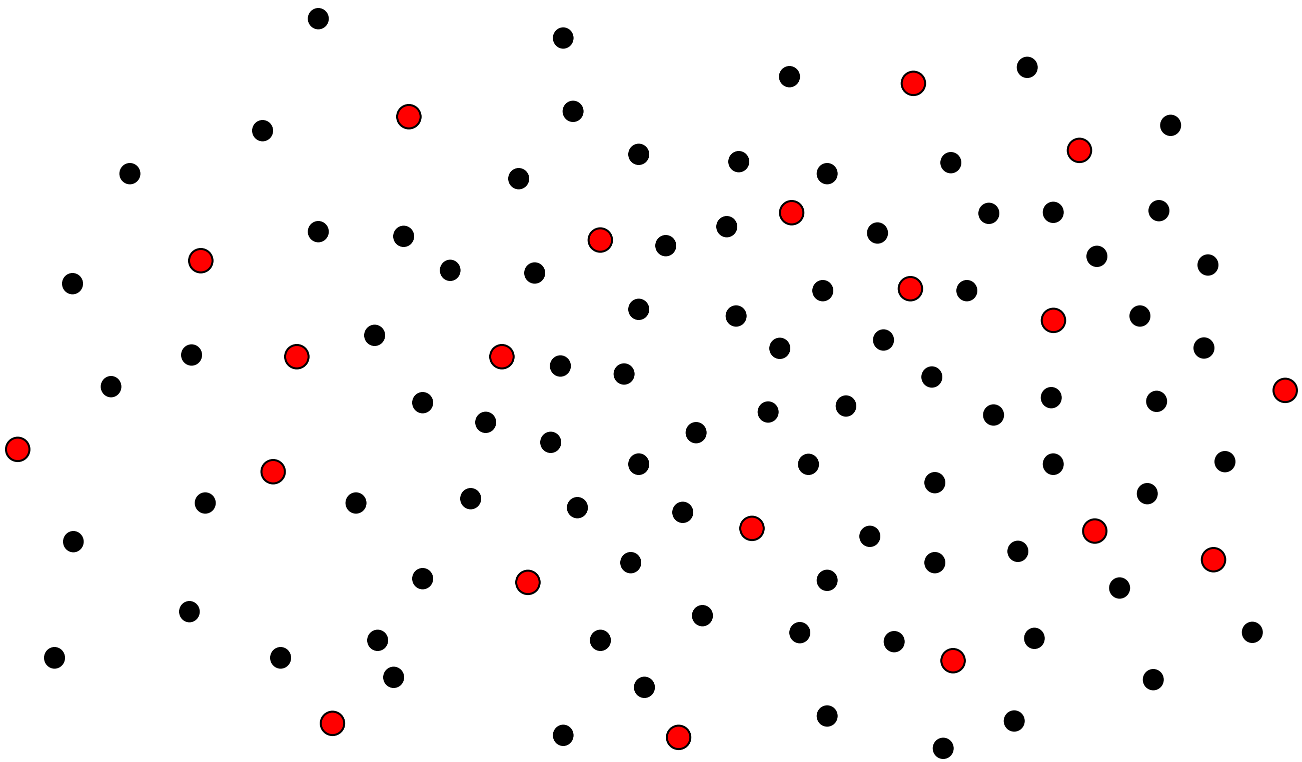
$\delta(u,v) \leq$ dist?$(u,v) \leq (2k-1)\delta(u,v)$

A hierarchy of centers



$A_0 \leftarrow V$; $A_i \leftarrow \text{sample}(A_{i-1}, n^{-1/k}),\ 1 \le i \le (k-1)$;

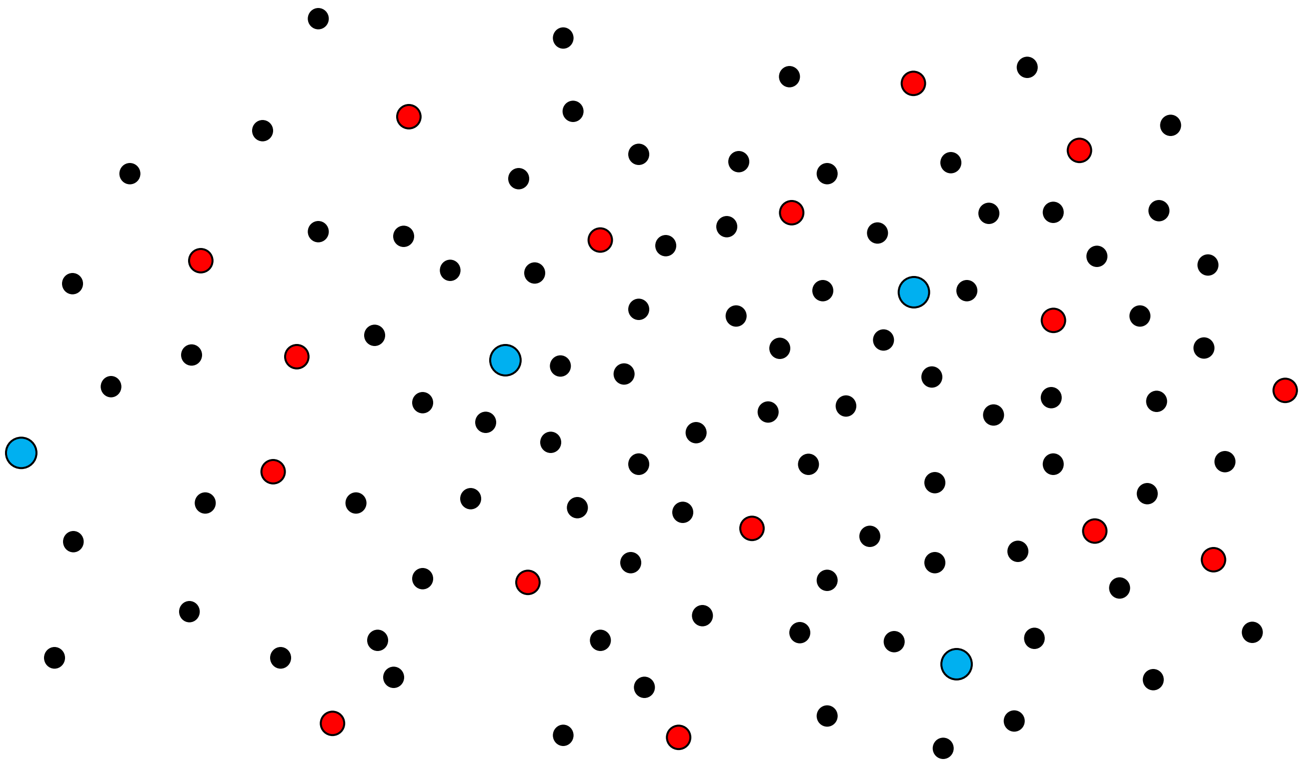
k=3



$A_0 \leftarrow V$;

$A_1 \leftarrow \text{sample}(A_0, n^{-1/3})$; $E(|A_1|) = n^{2/3}$

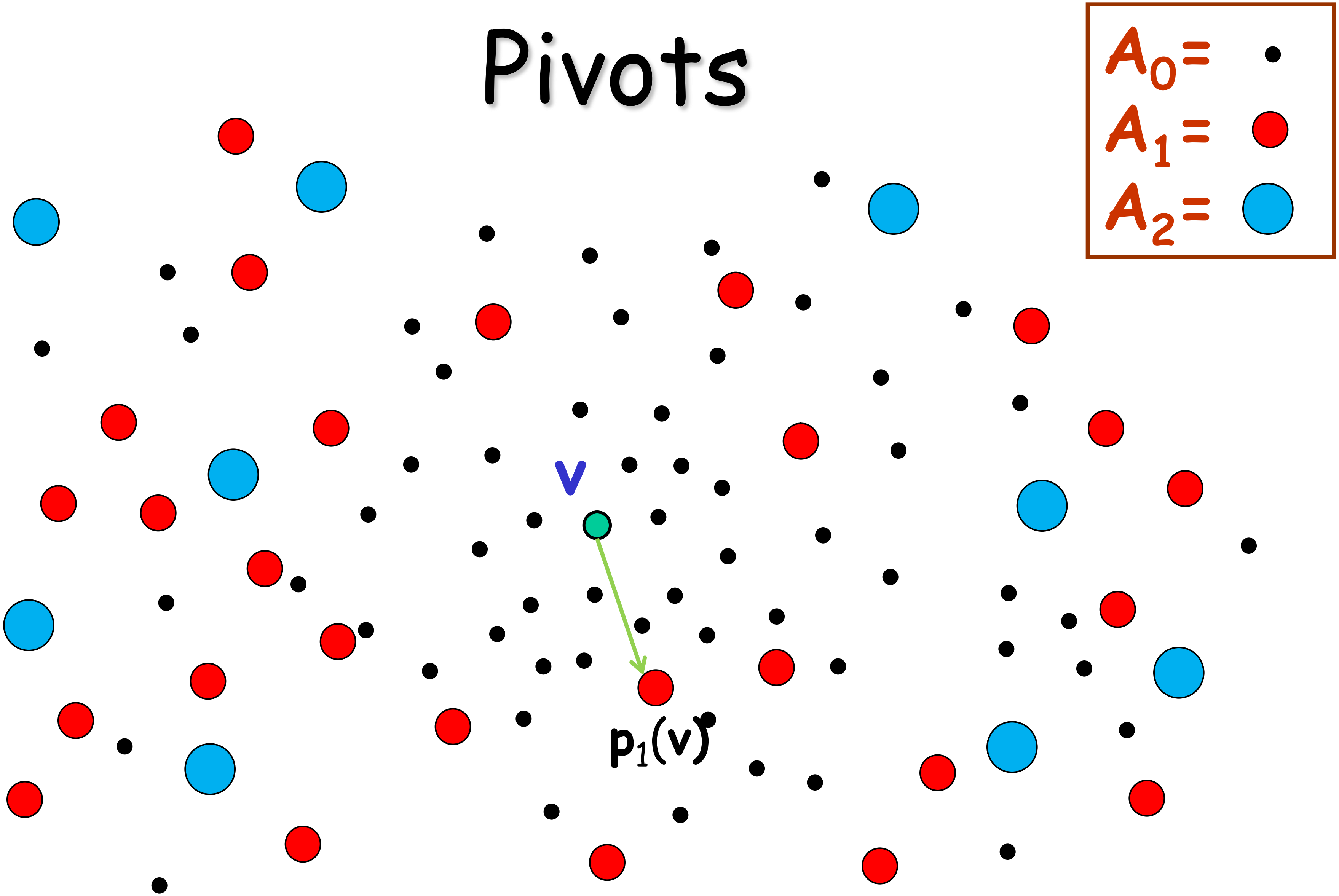
k=3



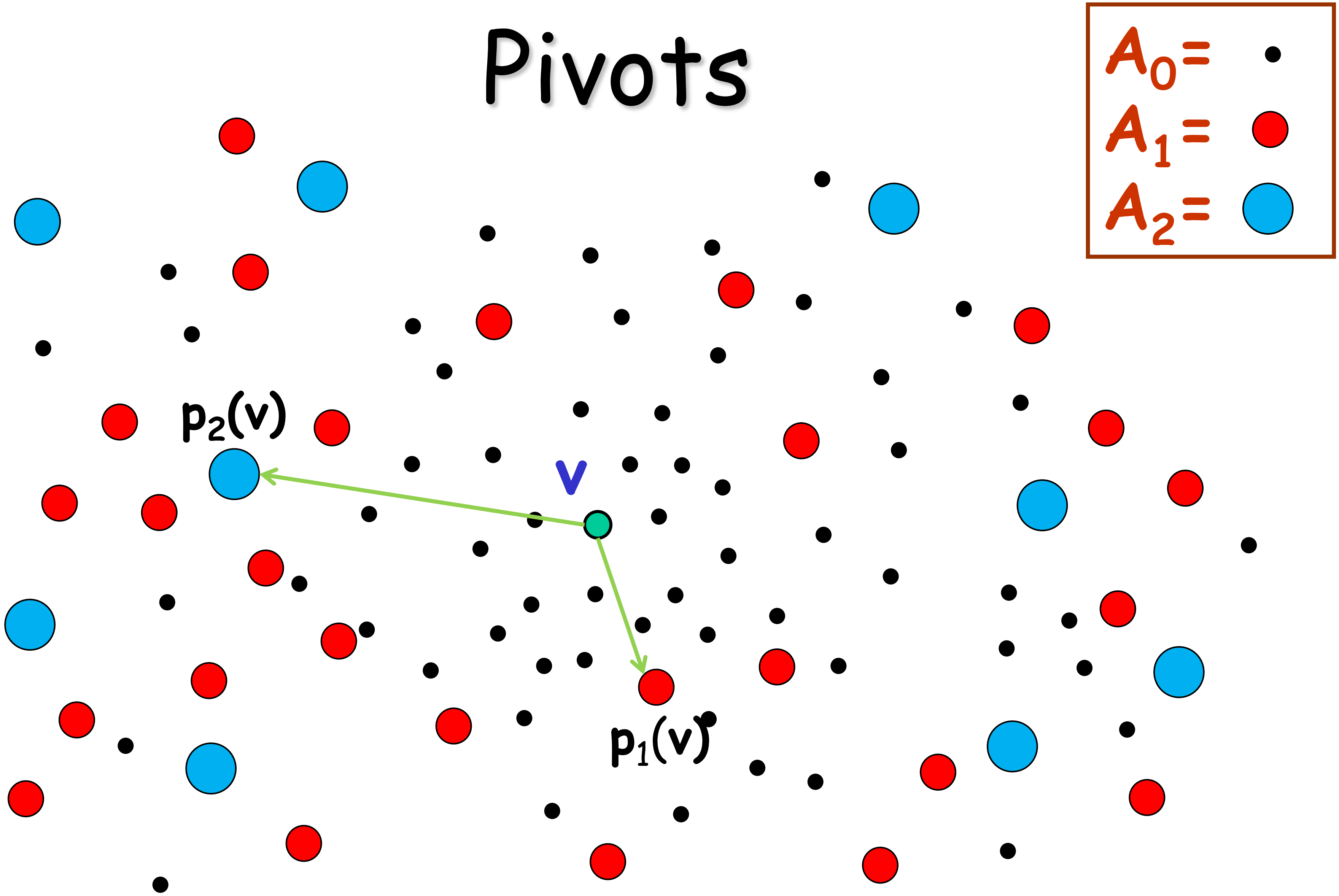
$A_0 \leftarrow V$;

$A_2 \leftarrow \text{sample}(A_1, n^{-1/3})$; $E(|A_2|) = n^{1/3}$

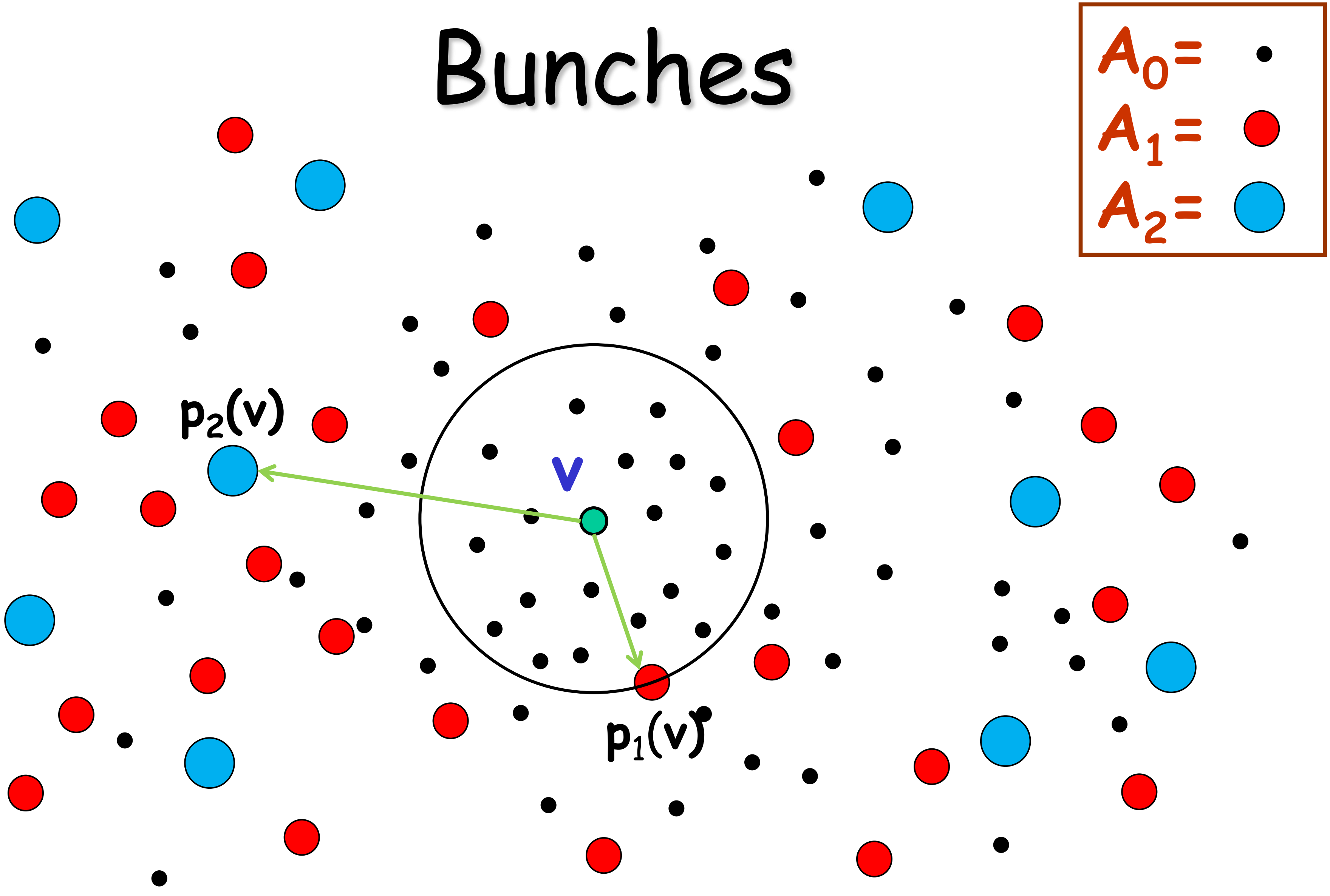
Pivots



Pivots

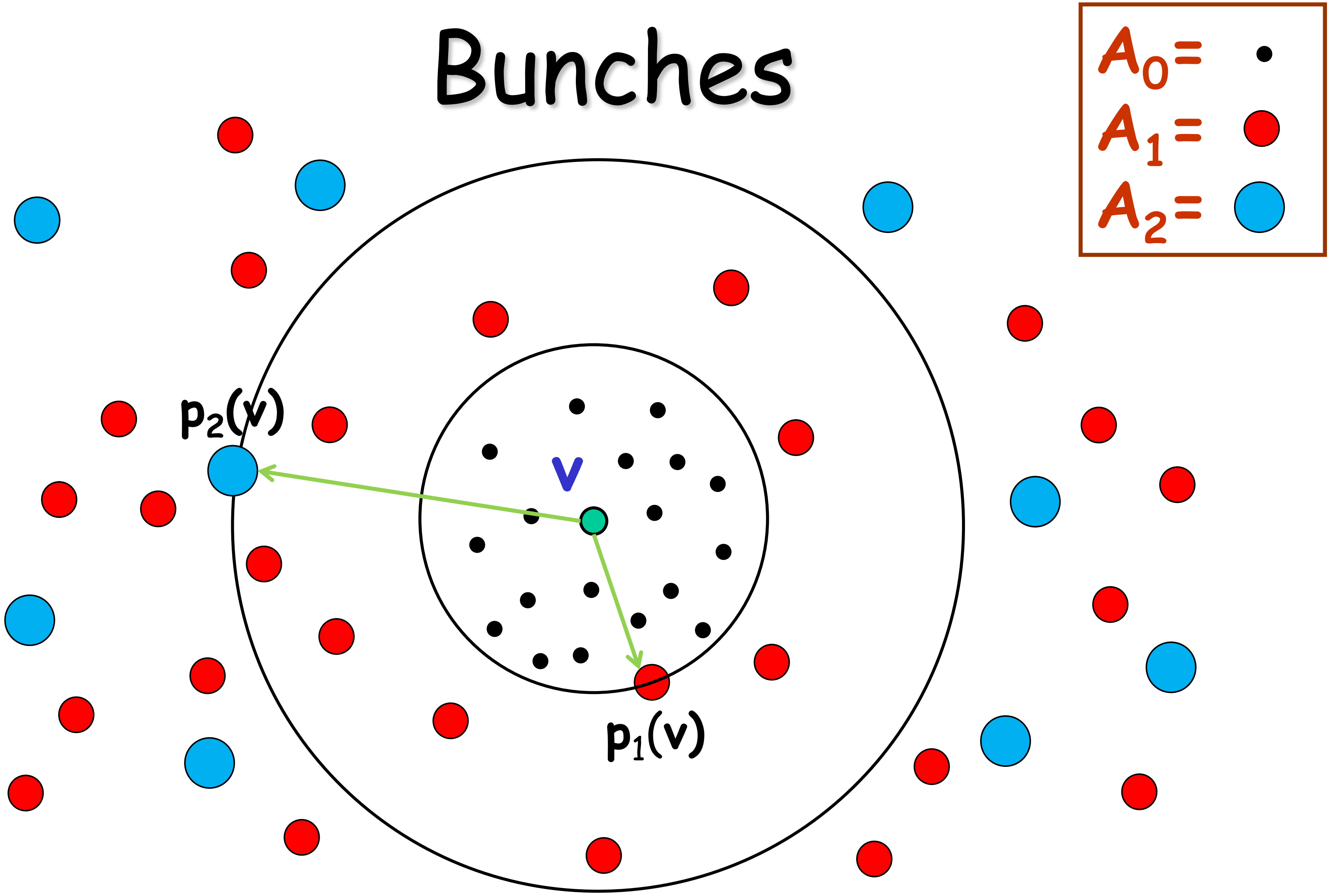


Bunches



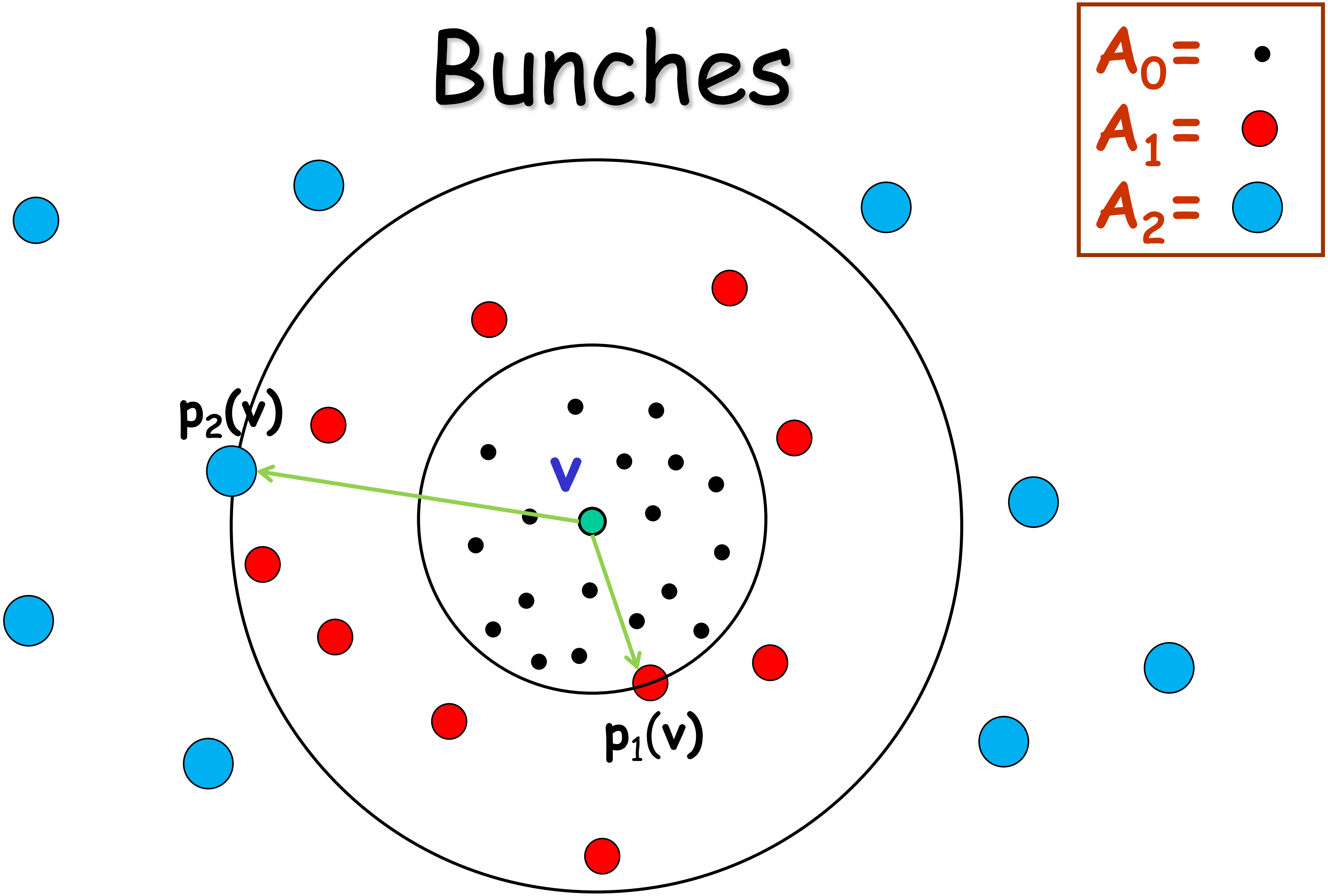
$$B_0(v) \leftarrow \{w \in A_0 - A_1 \mid \delta(v,w) < \delta(v, p_1(v))\}$$

Bunches



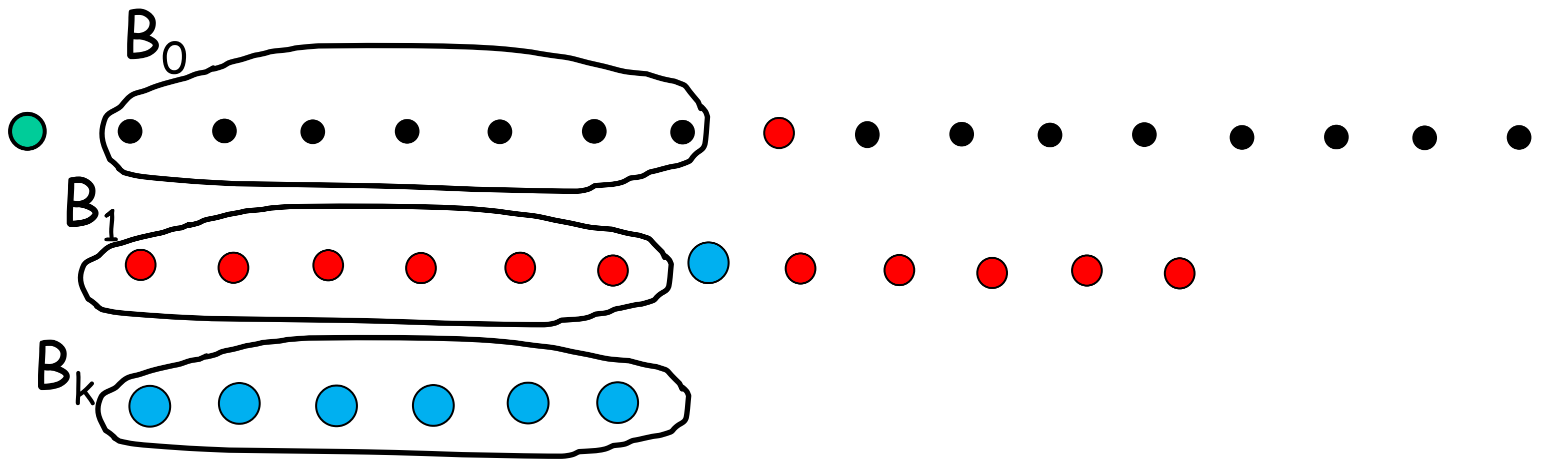
$$B_1(v) \leftarrow \{w \in A_1 - A_2 \mid \delta(v,w) < \delta(v, p_2(v))\}$$

Bunches



$B_2(v) \leftarrow A_2 \qquad B(v) \leftarrow B_0(v) \cup B_1(v) \cup B_2(v)$

The size of the bunch



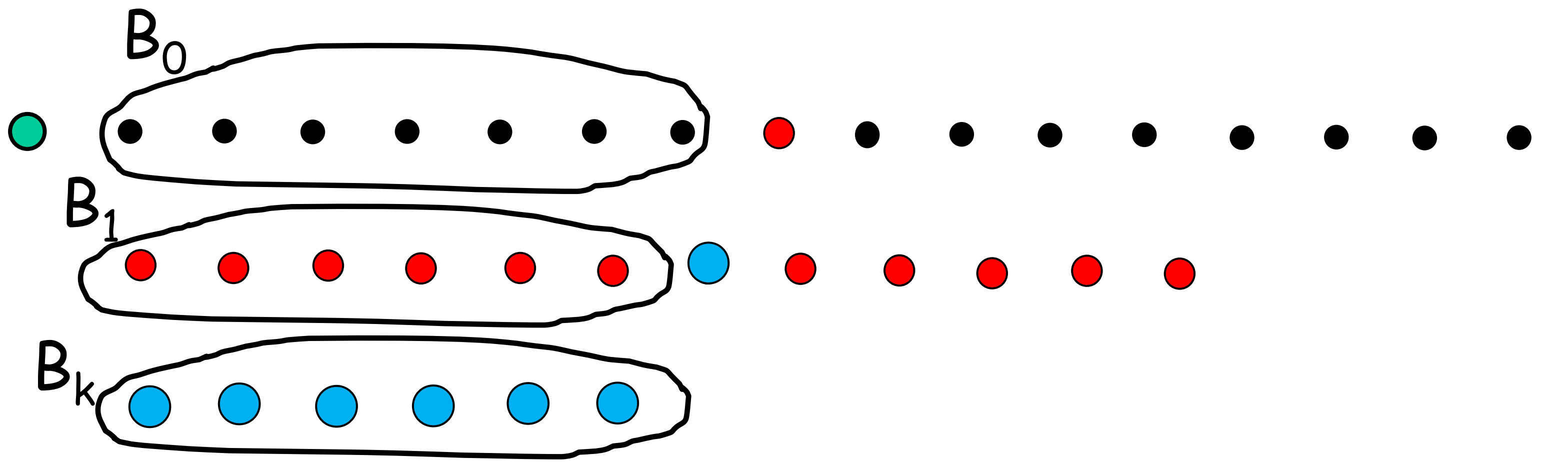
Order the vertices by their distance from v, the first one sampled into A_1 shows up after $\leq n^{1/k}$ on average.
➔ $E(|B_0(v)|) \leq n^{1/k}$

Order the vertices of A_1 by their distance from v, the first one sampled into A_2 appears after $\leq n^{1/k}$ on average. ➔ $E(|B_1(v)|) \leq n^{1/k}$

……….

The expected size of $A_k=B_k(v)$ is $n^{1/k}$

The size of the bunch



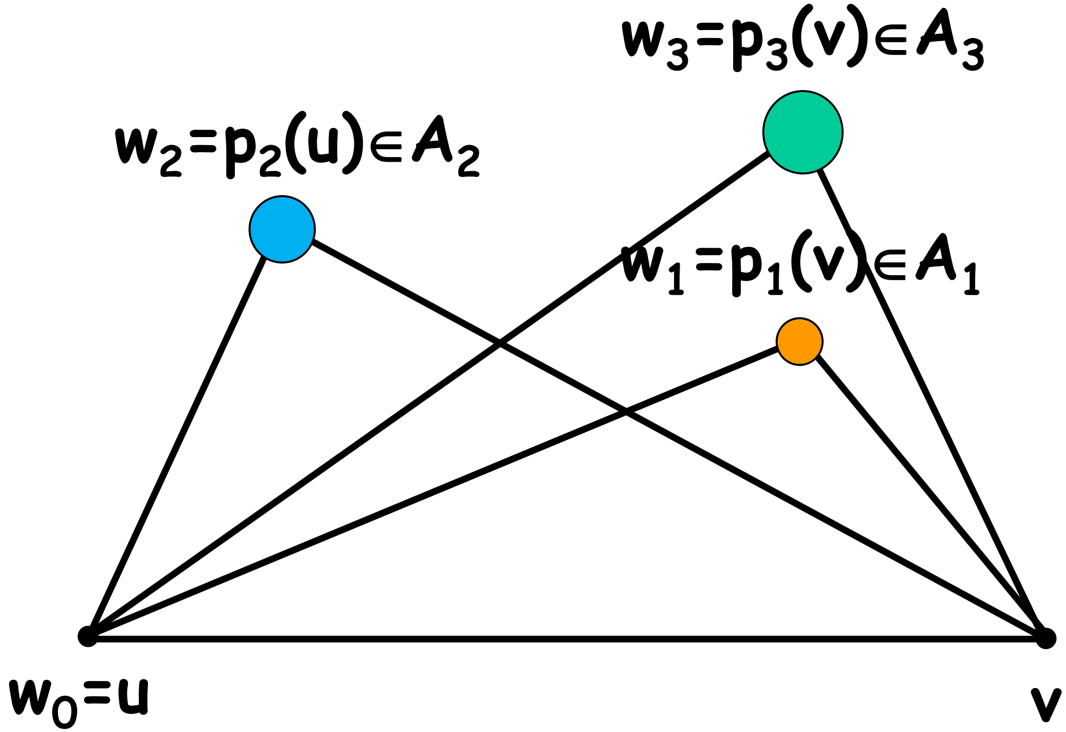
$E(|B(v)|) = O(kn^{1/k})$

Save $B(v)$ in a hash table together with $\delta(v,u)$ for every $u \in B(v)$

Save $p_i(v)$ and $\delta(v,p_i(v))$, $1 \le i \le (k-1)$

$$O\left(kn^{1+\frac{1}{k}}\right) \text{ space}$$

Query answering algorithm



Query answering algorithm

Algorithm $\mathrm{dist}_k(u,v)$

$w \leftarrow u$, $i \leftarrow 0$

while $w \notin B(v)$

{ $i \leftarrow i+1$

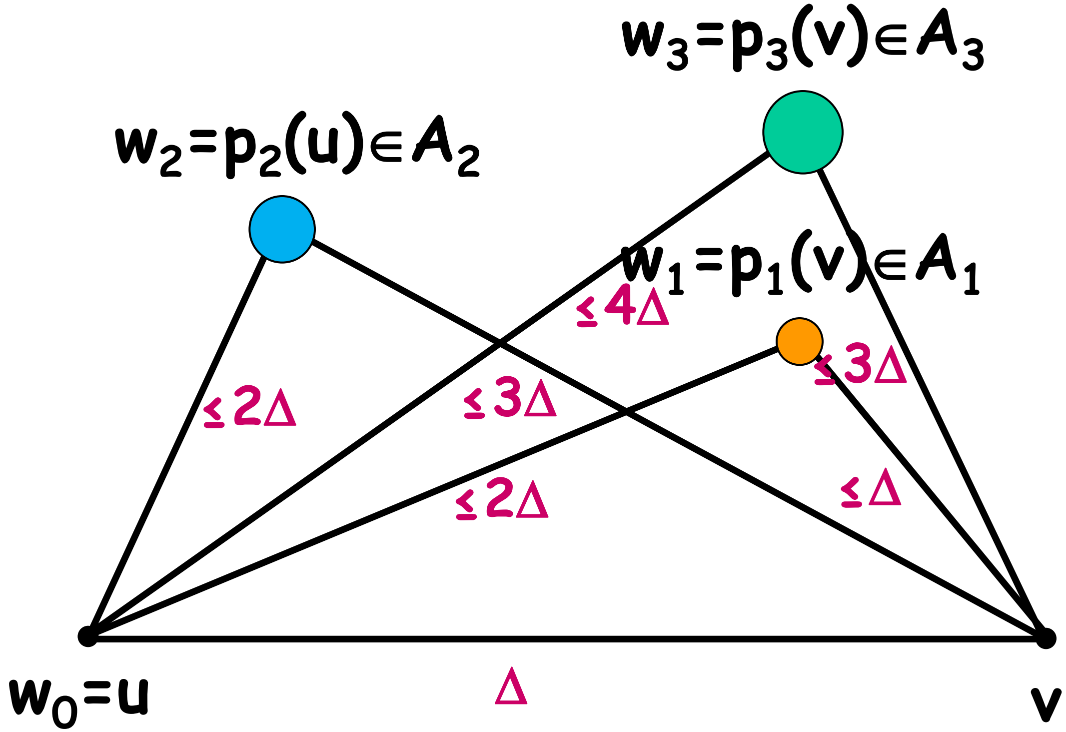
$(u,v) \leftarrow (v,u)$

$w \leftarrow p_i(u)$ }

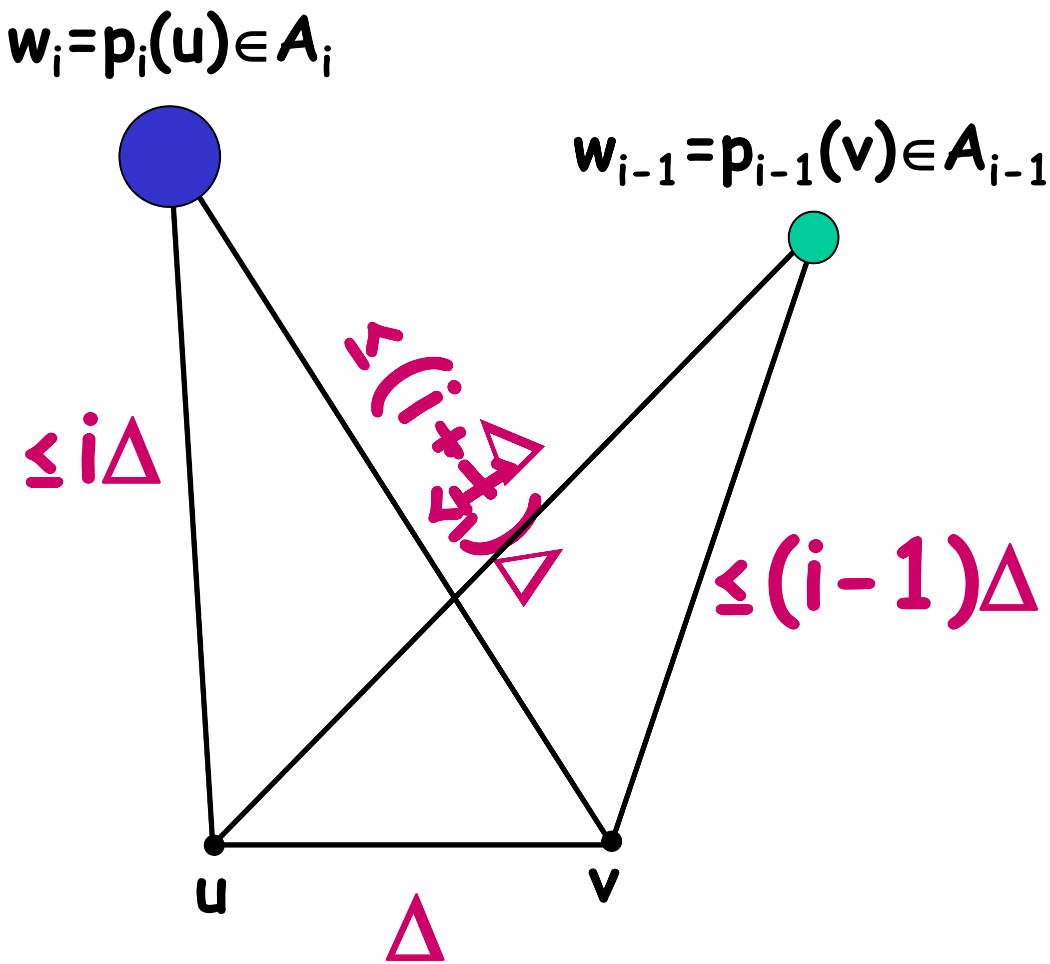
return $\delta(u,w) + \delta(w,v)$

Takes $O(k)$ time

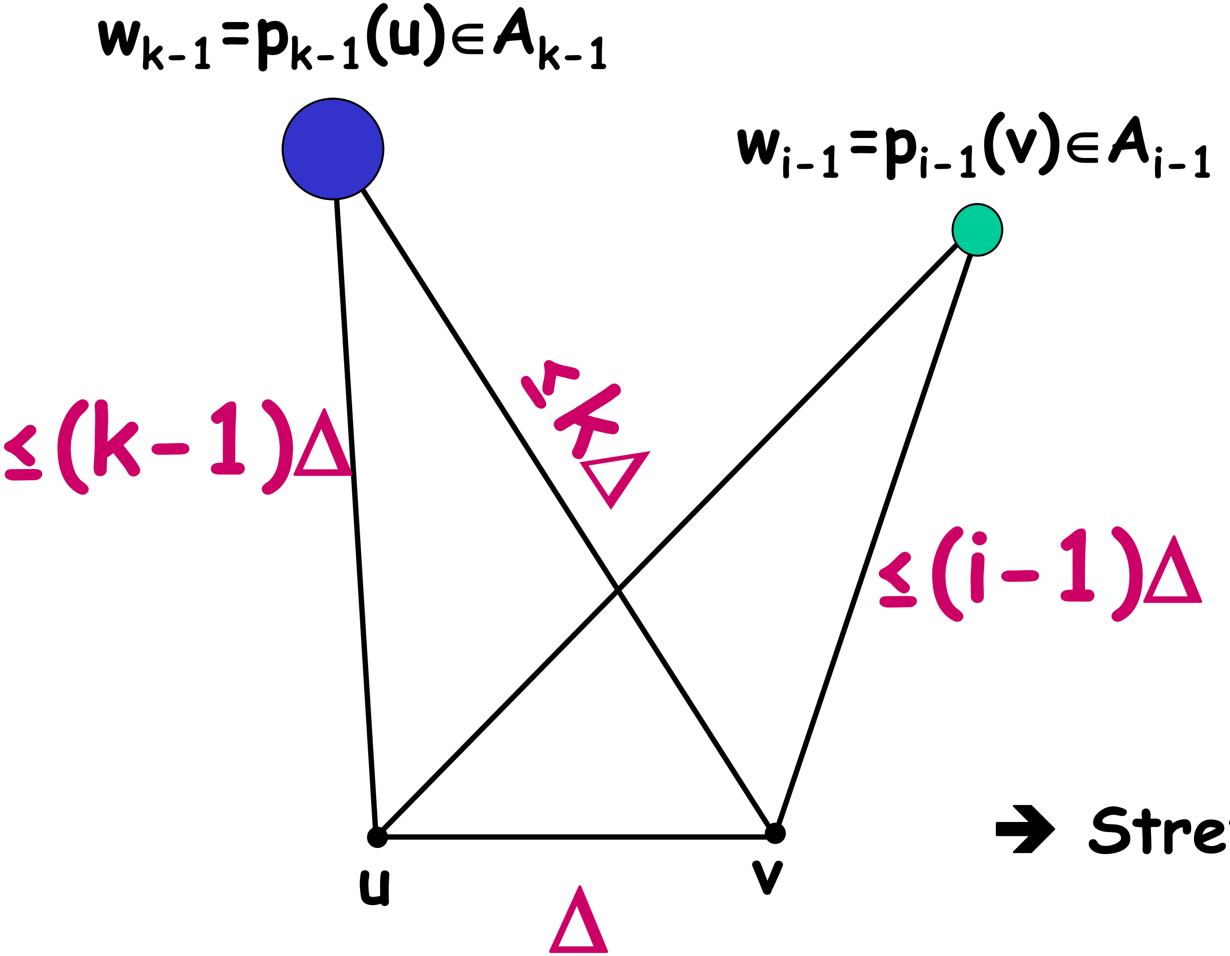
Bound the strech



Bound the stretch

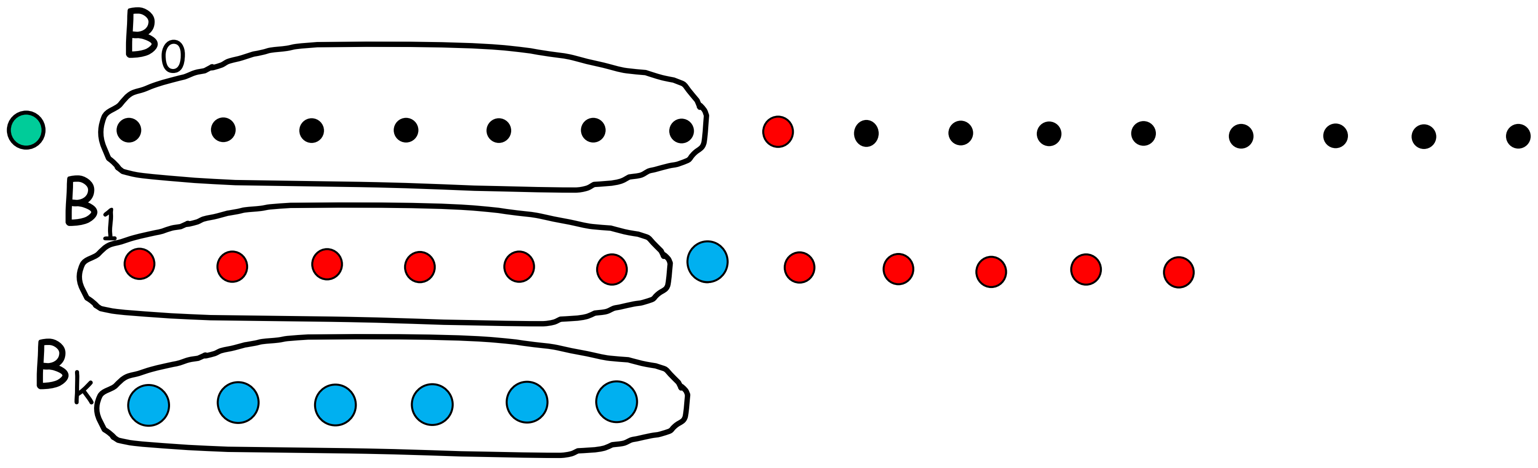


Bound the stretch



➔ Stretch $\leq (2k-1)$

Preprocessing

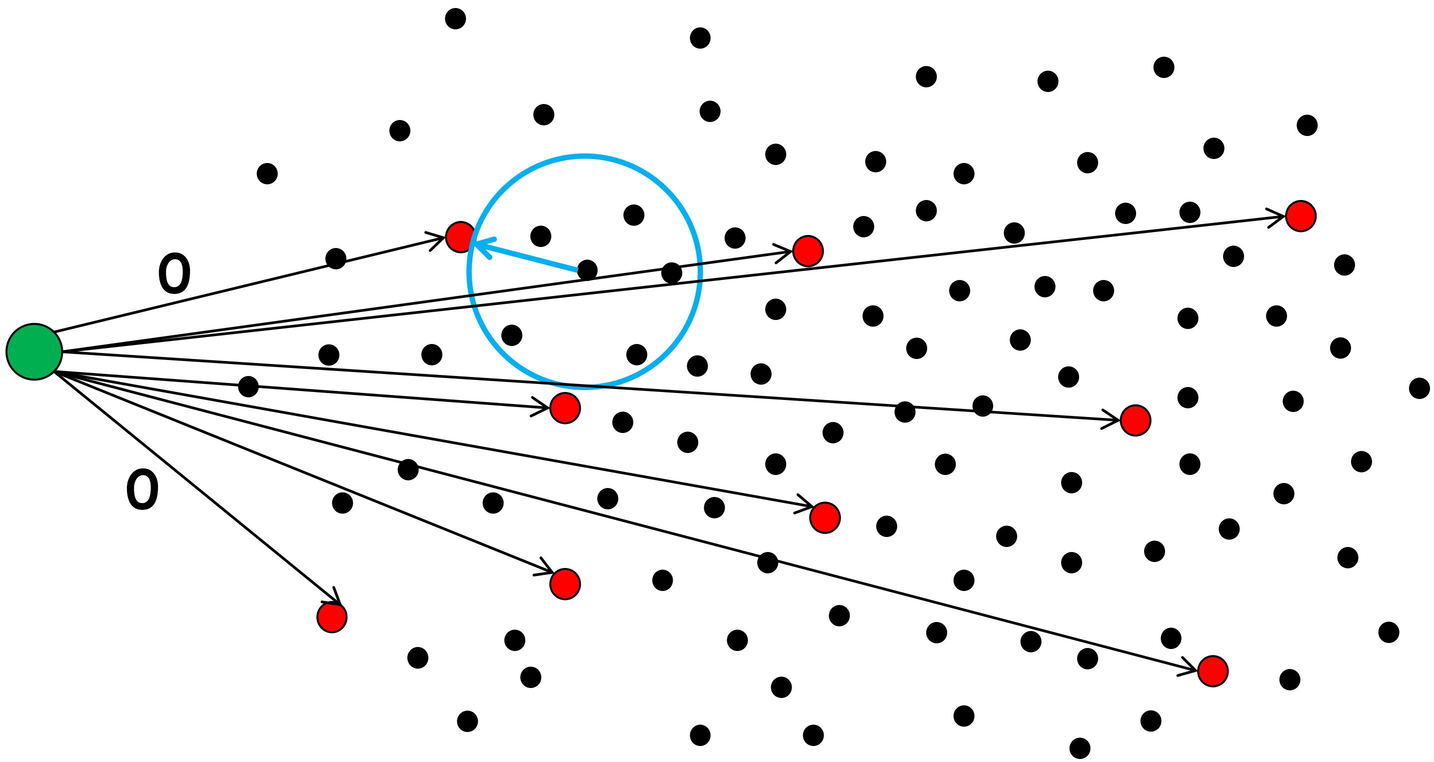


Easy, if we know all distances from v (run Dijkstra from v)

But this takes Ω(m) time per vertex → Ω(nm) total time

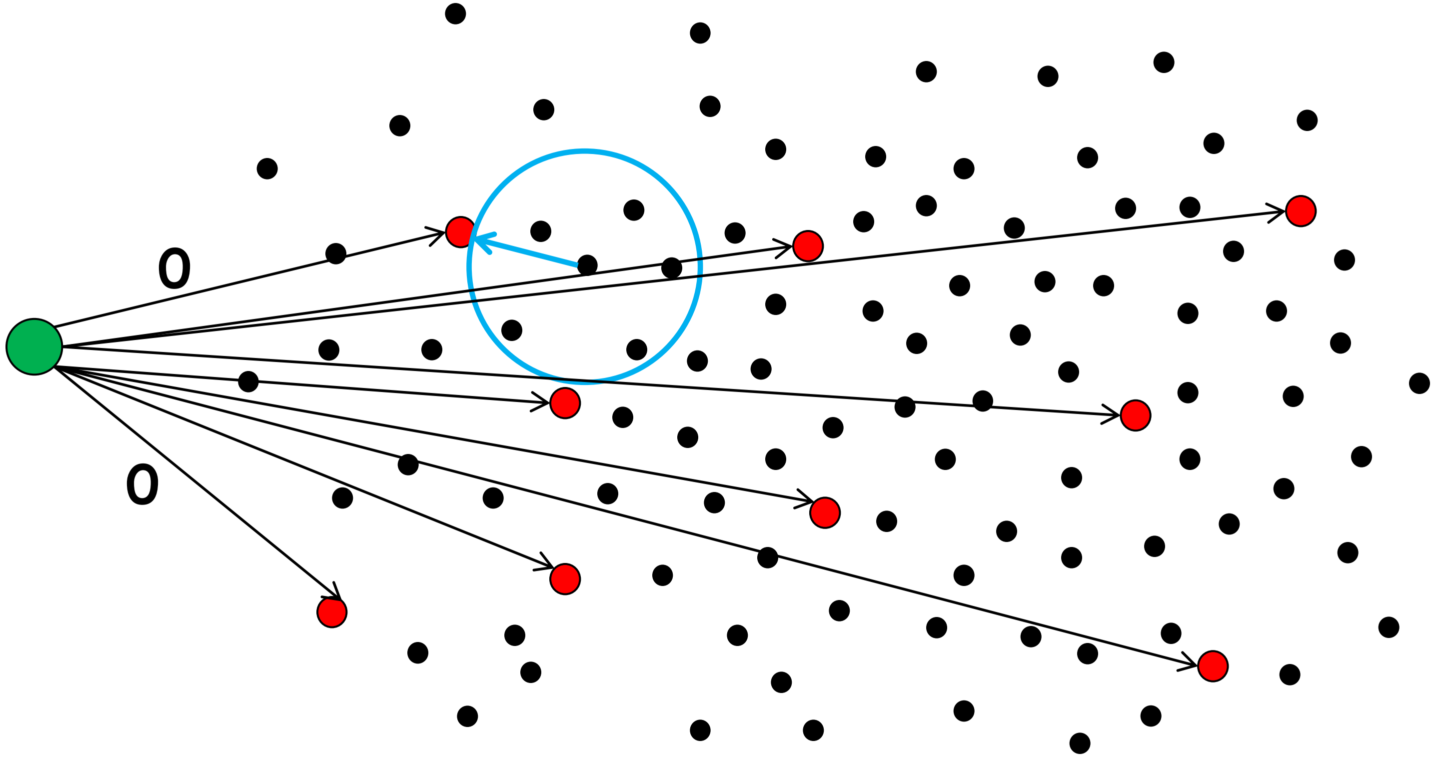
More efficient preprocessing

Computing Pivots



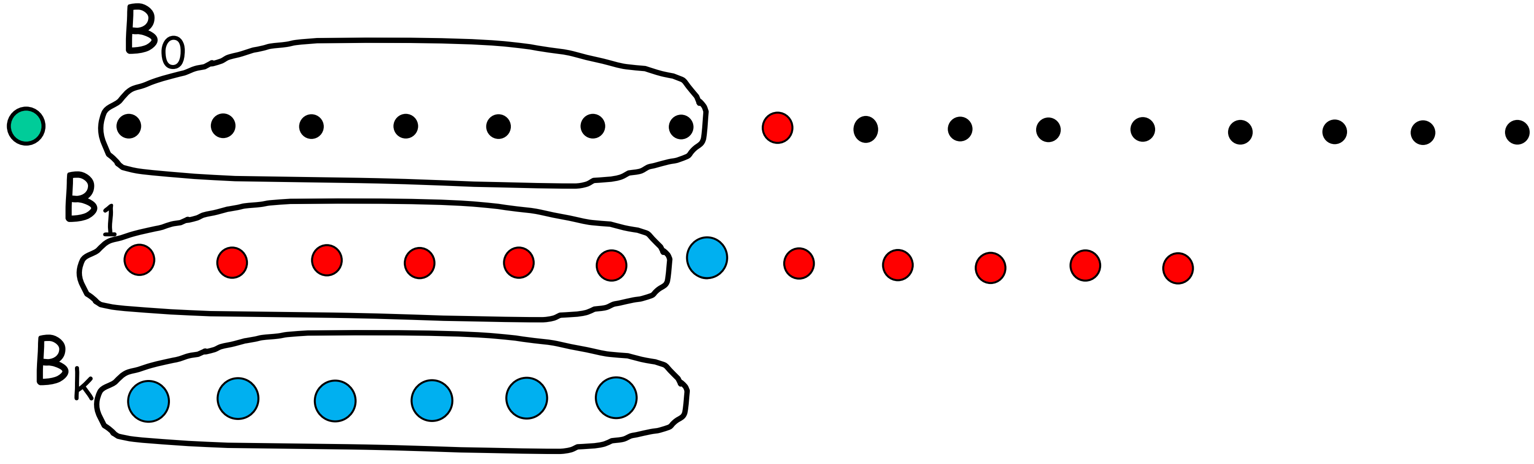
Do a shortest path computation from the **super-source**, the **red** ancestor of v in the tree is its pivot

Computing Pivots



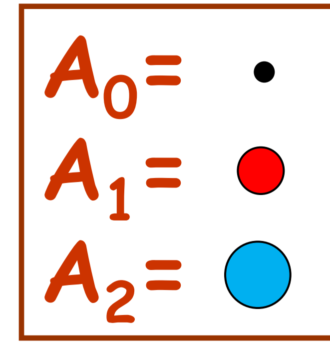
➔ In O(k) applications of Dijkstra's algorithm we know all pivots

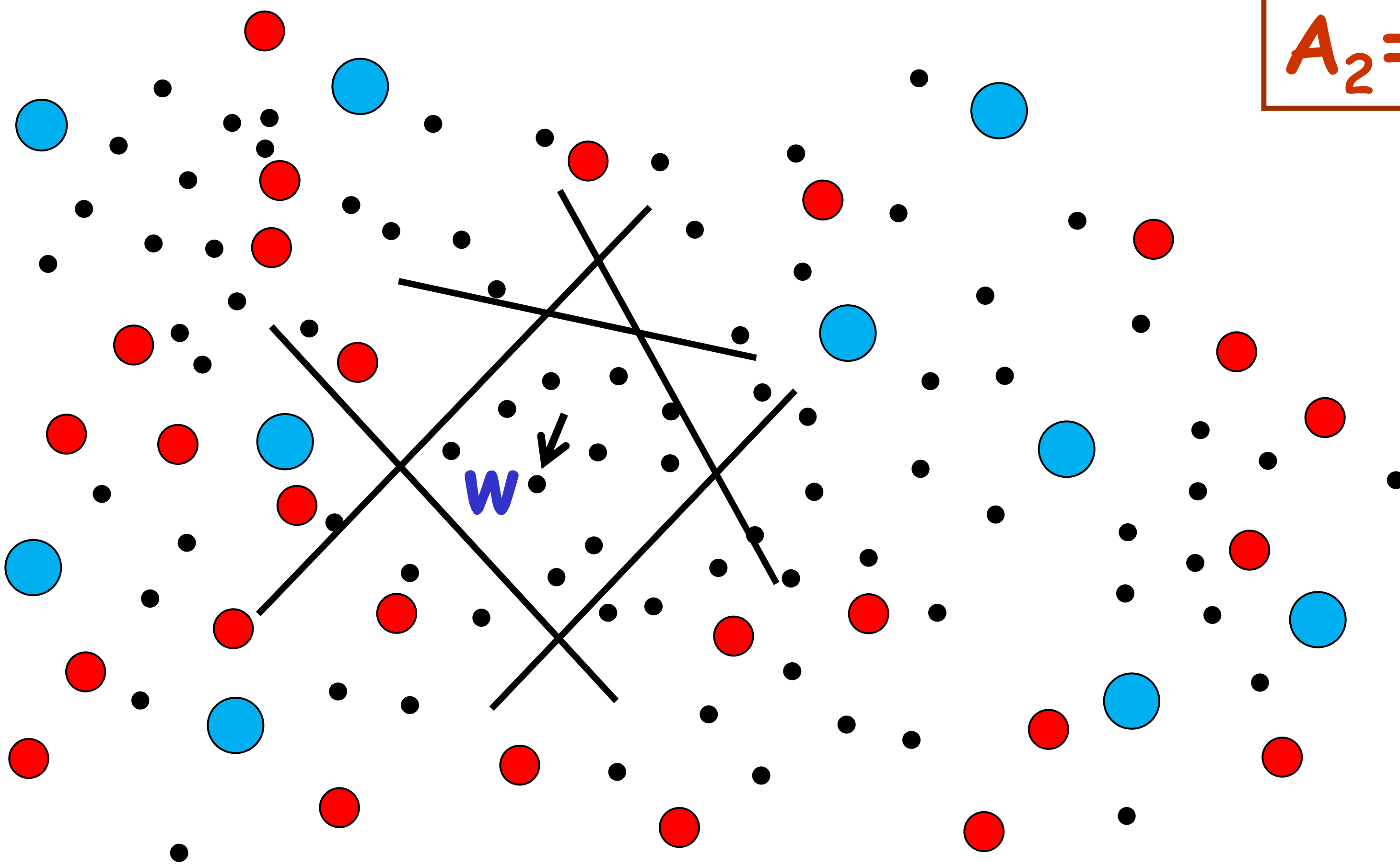
More efficient preprocessing



A vertex w will collect all vertices v such that $w \in B(v)$

Clusters





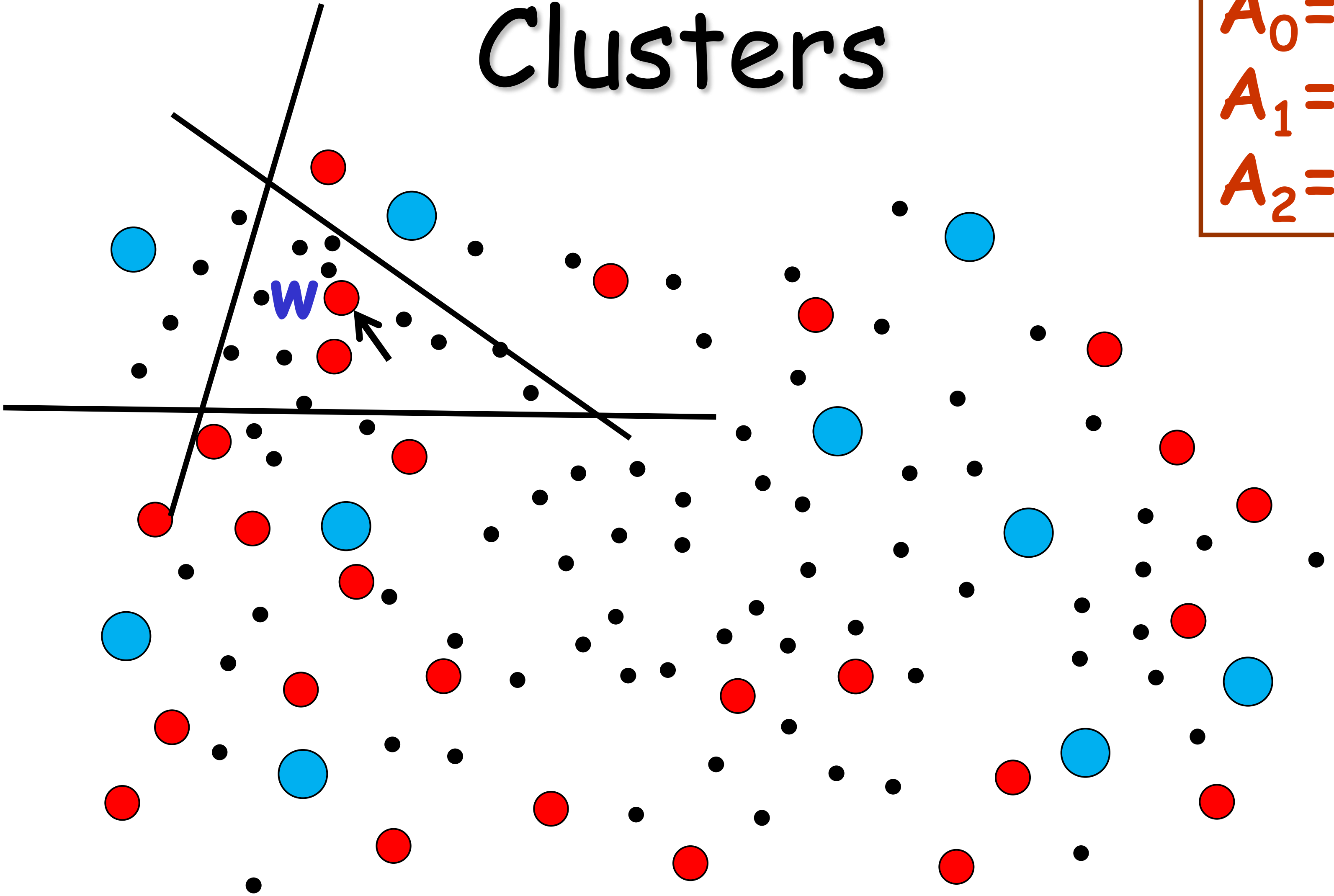
$$C(w) \leftarrow \{v \in V \mid \delta(w,v) < \delta(v, A_1)\} \qquad w \in A_0 - A_1$$

Bunches are inverse clusters

Consider $w \in A_0 - A_1$

$$w \in B_0(v) \quad \Leftrightarrow \quad v \in C(w)$$

Clusters



$A_0 = \bullet$

$A_1 = \text{red circle}$

$A_2 = \text{blue circle}$

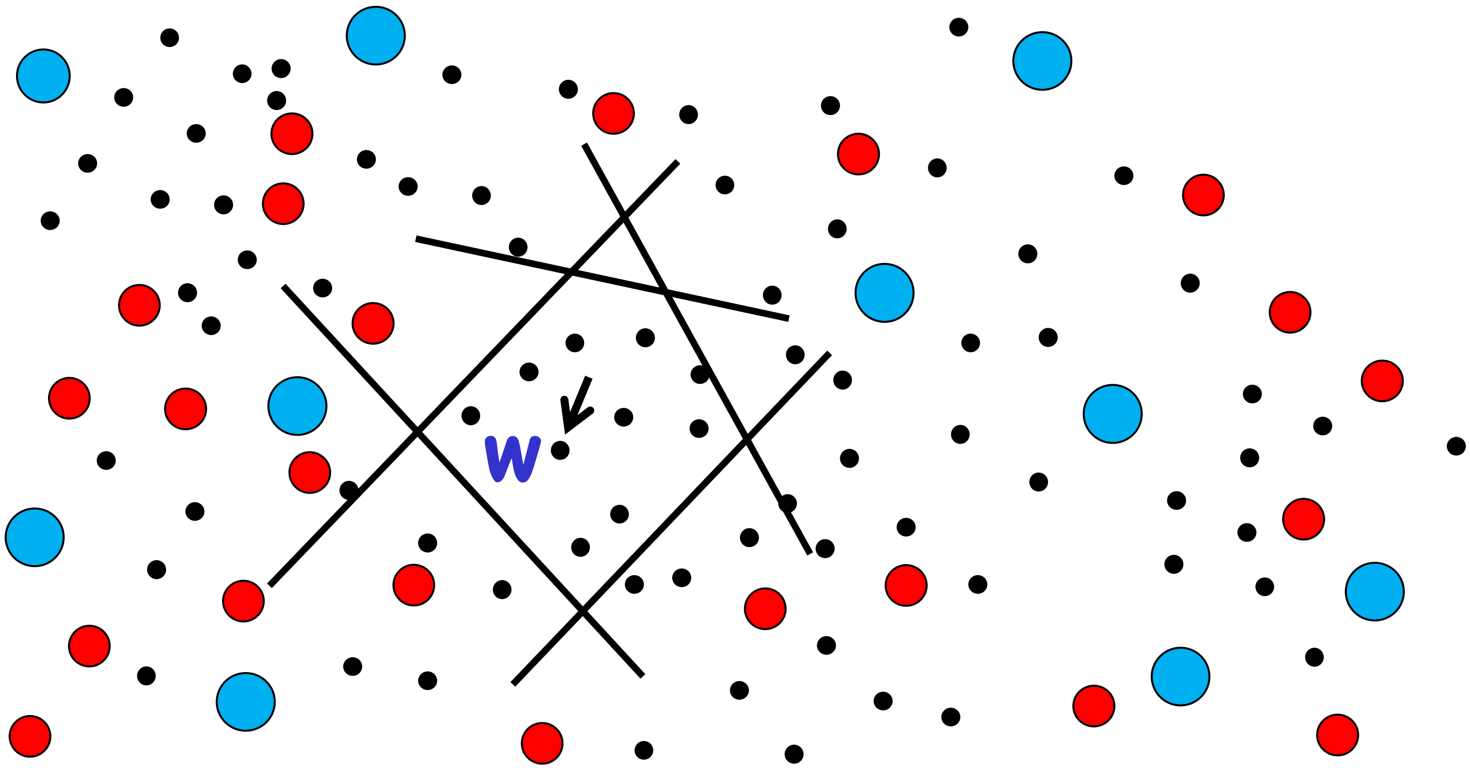
$$C(w) \leftarrow \{v \in V \mid \delta(w,v) < \delta(v, A_2)\} \qquad w \in A_1 - A_2$$

Bunches are inverse clusters

Consider $w \in A_1 - A_2$

$$w \in B_1(v) \quad \Leftrightarrow \quad v \in C(w)$$

Computing clusters



Run Dijkstra from w **but**

relax an edge (u,v) only if $d(u) + \ell(u,v) < \delta(v,A_1) = \delta(v,p_1(v))$

Computing clusters

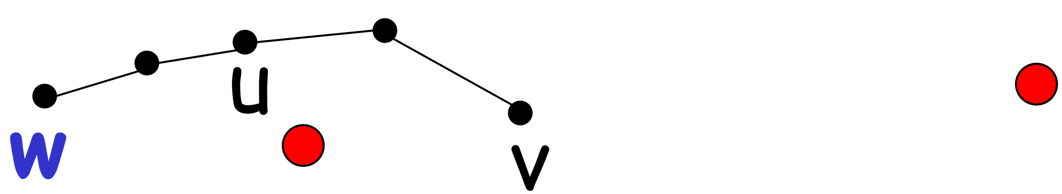
Lemma: This application of Dijkstra's alg. scans exactly the vertices of $C(w)$ giving them correct distance labels

Proof: (By the def of the alg.) A vertex $v \notin C(w)$ does not get a finite distance label so we do not visit v.

Computing clusters

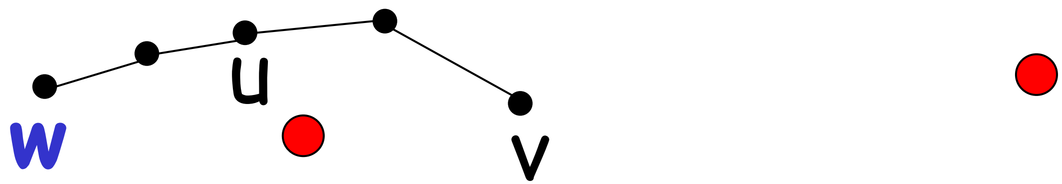
Lemma: This application of Dijkstra's alg. scans exactly the vertices of $C(w)$ giving them correct distance labels

Proof: If $v \in C(w)$ then every u on the shortest path from w to v is in $C(w)$



Computing clusters

Proof: If $v \in C(w)$ then every u on the shortest path from w to v is in $C(w)$



If $\delta(u,p_1(u)) < \delta(w,u)$ then
$\delta(u,p_1(u)) + \delta(u,v) < \delta(w,u) + \delta(u,v) = \delta(w,v)$

We finish showing that the algorithm scans and give correct labels to all vertices in $C(w)$ by induction on the hop-distance from w.

Analysis (preprocessing)

Neglecting logarithmic factors the running time is

$O(\Sigma_w \text{edges}(C(w)))$

A vertex v appears in |B(v)| clusters and contributes its degree to each so

$=O(\Sigma_v |B(v)||\text{edges}(v)|) = O(kn^{1/k}m)$

Stretch/space tradeoff

Let $G=(V,E)$ be a graph with $|V|=n$ and $girth(G)\geq 2k+2$.

Any subgraph $G'=(V,E')$ of G must have a distinct data structure for stretch $t < 2k+1$

If $(u,v)\in E'$, then $\delta_{G'}(u,v)=1$. Otherwise $\delta_{G'}(u,v) \geq 2k+1$.

As there are $2^{|E|}$ different subgraphs of G, some subgraphs must have data structures of at least $|E|$ bits.

Conjecture: (Erdös '65) For every $k\geq 1$, there are infinitely many n-vertex graphs with $\Omega(n^{1+1/k})$ edges that have girth$\geq 2k+2$.

Applications

- Routing
- Distance labels
- Spanners