Parallel Cluster-BFS and Applications to Shortest Paths

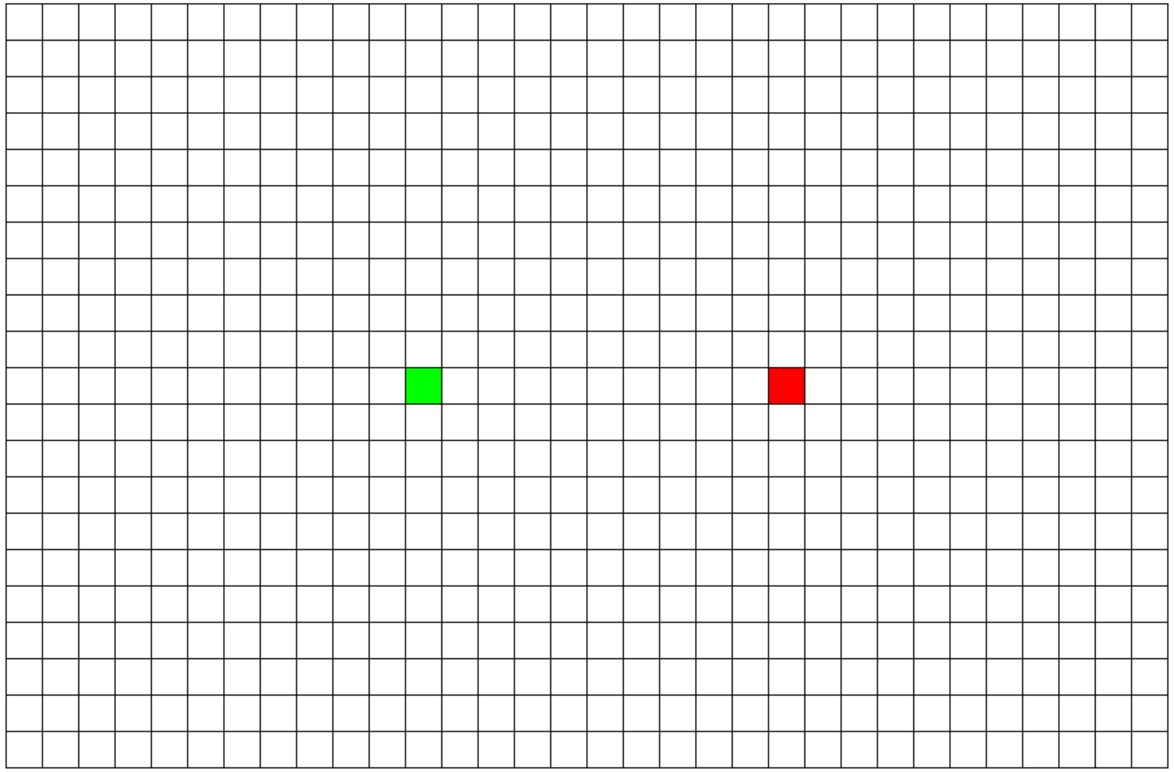
Letong Wang*

Joint work with Guy Blelloch[†], Yan Gu*, Yihan Sun*

*: UC Riverside   †: Carnegie Mellon University

Definition of Cluster-BFS

- **Breadth-first search (BFS) is one of the most important graph processing subroutine**
  - Computing the unweighted distance



source: https://www.jvruo.com/archives/1877/

Definition of Cluster-BFS

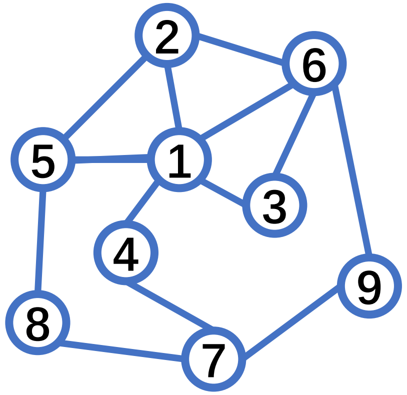
- **Breadth-first search (BFS) is one of the most important graph processing subroutine**
  - Computing the unweighted distance
- **Many applications may require running BFS from multiple sources**
  - Computing all pairs shortest paths (APSP)
  - Constructing distance oracles for fast shortest distance query

Definition of Cluster-BFS

- **Breadth-first search (BFS) is one of the most important graph processing subroutine**
  - Computing the unweighted distance
- **Many applications may require running BFS from multiple sources**
  - Computing all pairs shortest paths (APSP)
  - Constructing distance oracles for fast shortest distance query
- **One special variant of multi-source BFS is Cluster-BFS (C-BFS)**

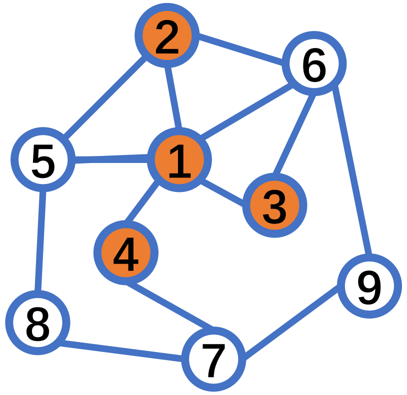
Definition of Cluster-BFS

- Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously



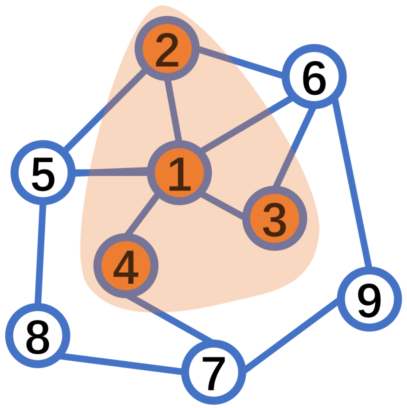
Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close



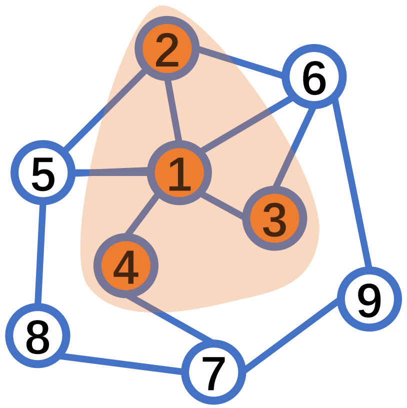
Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close



Definition of Cluster-BFS

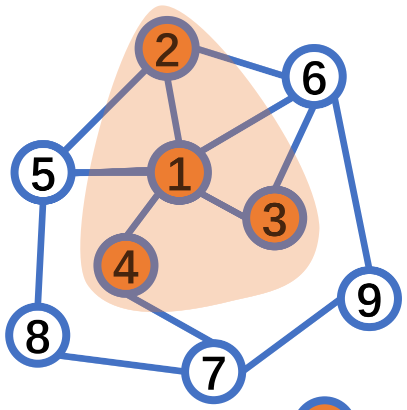
- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close



$$k = 4$$
$$d = 2$$

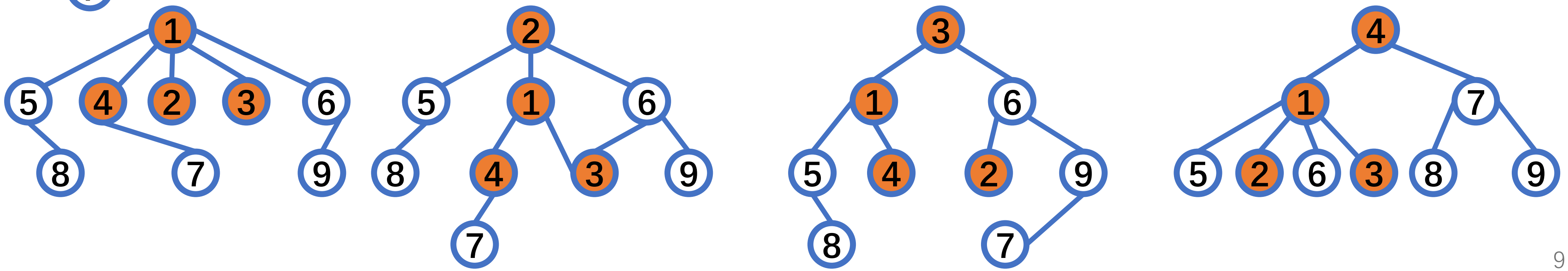
Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close



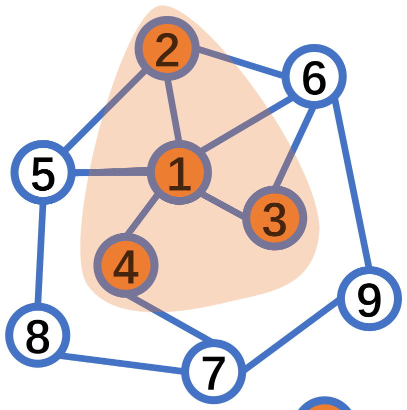
$k = 4$

$d = 2$



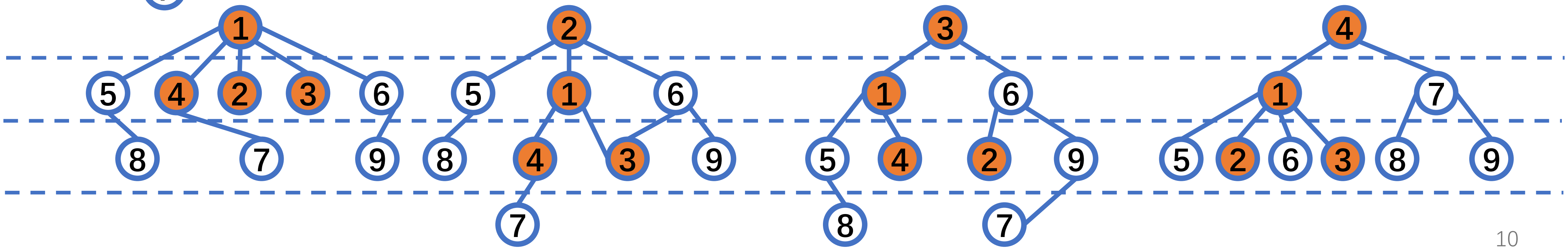
Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close



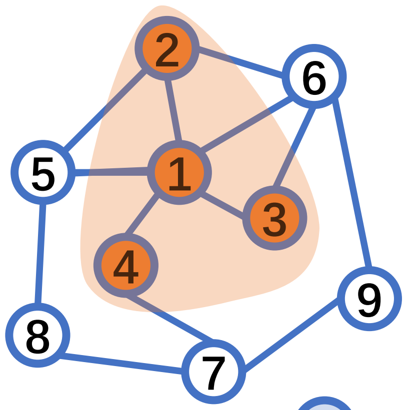
$k = 4$

$d = 2$

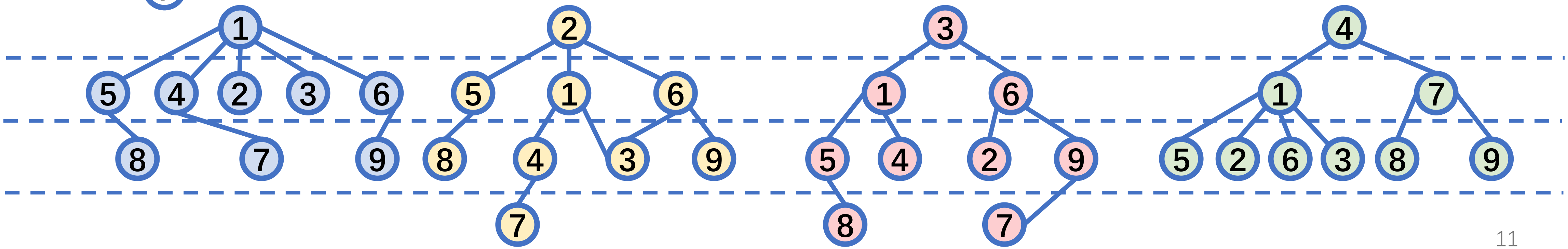


Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close

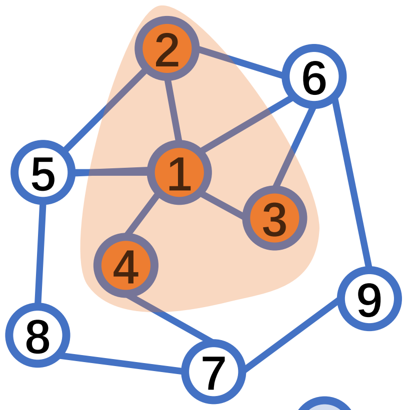


$k = 4$
$d = 2$

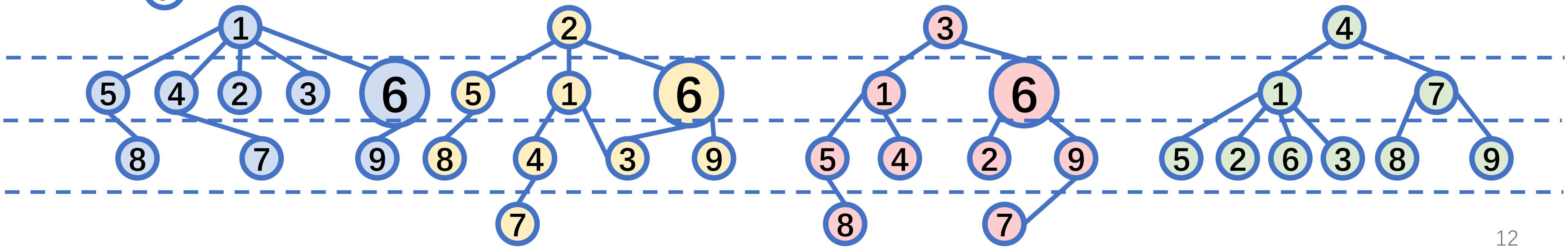


Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close

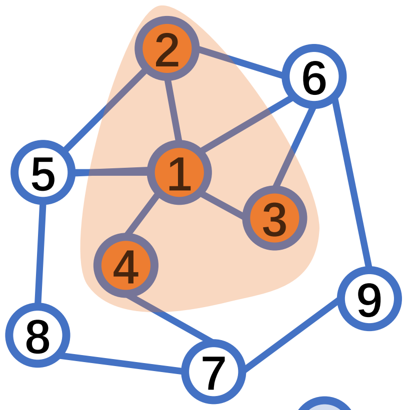


$k = 4$
$d = 2$

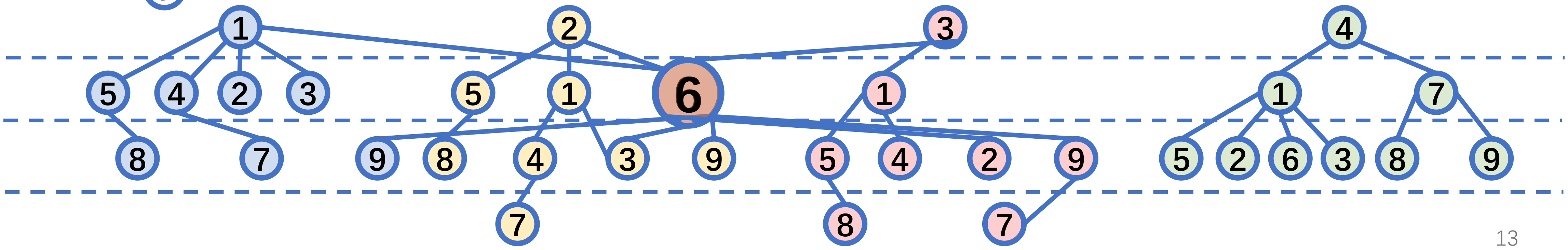


Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close

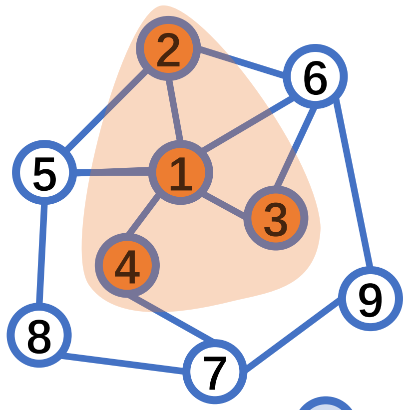


$k = 4$
$d = 2$

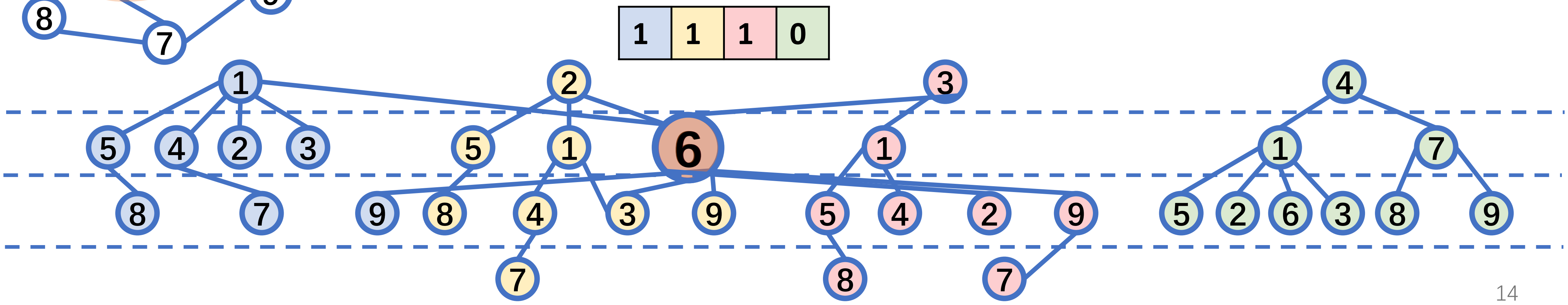


Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close

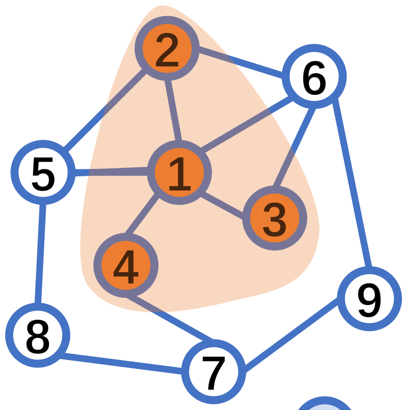


$k = 4$
$d = 2$



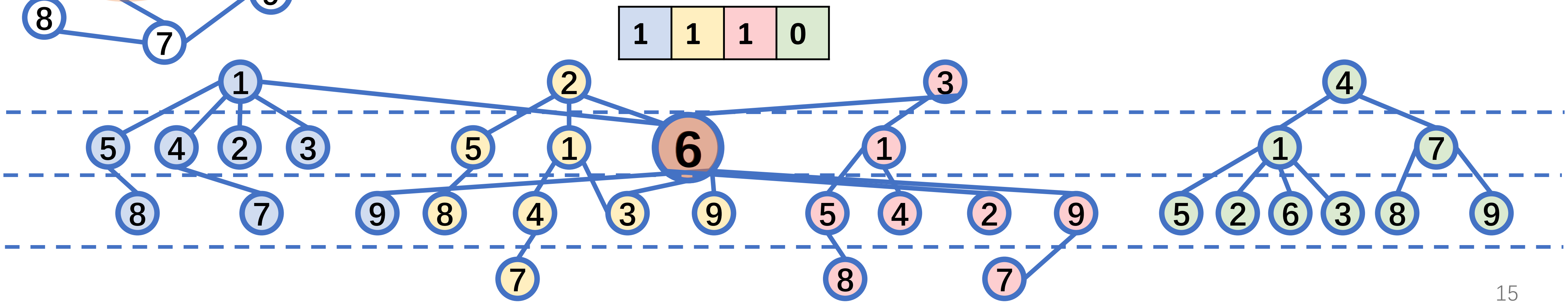
Definition of Cluster-BFS

- **Cluster-BFS (C-BFS) conducts BFSs from a cluster simultaneously**
- **A cluster is a subset of vertices that are close to each other**
  - Cluster size is represented by $\boldsymbol{k}$ -- how large
  - Cluster diameter is represented by $\boldsymbol{d}$ -- how close



$$k = 4$$
$$d = 2$$

Bitwise-parallelism



Bitwise-parallelism shows efficiency in C-BFS

Algorithms	Sources	Contributions	Parallelism
Chan's BFS [Chan'12]			
Sequential C-BFS [AIY'12]			
Ligra BFS [ShunBlelloch'13]			

Bitwise-parallelism is efficient in C-BFS

Algorithms	Sources	Contributions	Parallelism
Chan's BFS [Chan'12]	A general cluster	**Theory Improvement** All pair shortest paths algorithm with $O(mn/\log n)$ work	Bitwise
Sequential C-BFS [AIY'12]			Bitwise
Ligra BFS [ShunBlelloch'13]			

Bitwise-parallelism is efficient in C-BFS

Algorithms	Sources	Contributions	Parallelism
Chan's BFS [Chan'12]	A general cluster	**Theory Improvement** All pair shortest paths algorithm with $O(mn/\log n)$ work	Bitwise
Sequential C-BFS [AIY'12]	A star-shaped cluster	**Efficient Implementation** Exact two-hop distance oracle	Bitwise
Ligra BFS [ShunBlelloch'13]			

Multi-threaded single-source BFS is highly optimized

Algorithms	Sources	Contributions	Parallelism
Chan's BFS [Chan'12]	A general cluster	**Theory Improvement** All pair shortest paths algorithm with $O(mn/\log n)$ work	Bitwise
Sequential C-BFS [AIY'12]	A star-shaped cluster	**Efficient Implementation** Exact two-hop distance oracle	Bitwise
Ligra BFS [ShunBlelloch'13]	A single vertex	**Highly-Optimized Building Block** Forward-backward optimization	Thread

How do bit-parallelism and thread-level parallelism perform together?

Algorithms	Sources	Contributions	Parallelism
Chan's BFS [Chan'12]	A general cluster	**Theory Improvement** All pair shortest paths algorithm with $O(mn/\log n)$ work	Bitwise
Sequential C-BFS [AIY'12]	A star-shaped cluster	**Efficient Implementation** Exact two-hop distance oracle	Bitwise
Ligra BFS [ShunBlelloch'13]	A single vertex	**Highly-Optimized Building Block** Forward-backward optimization	Thread
Our algorithm			**Bitwise + Thread**

How do bit-parallelism and thread-level parallelism perform together?

Algorithms	Sources	Contributions	Parallelism
Chan's BFS [Chan'12]	A general cluster	**Theory Improvement** All pair shortest paths algorithm with $O(mn/\log n)$ work	Bitwise
Sequential C-BFS [AIY'12]	A star-shaped cluster	**Efficient Implementation** Exact two-hop distance oracle	Bitwise
Ligra BFS [ShunBlelloch'13]	A single vertex	**Highly-Optimized Building Block** Forward-backward optimization	Thread
Our algorithm	**A general cluster**		**Bitwise + Thread**

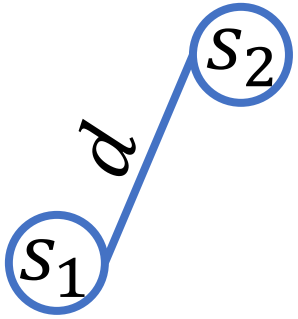
How do bit-parallelism and thread-level parallelism perform together?

Algorithms	Sources	Contributions	Parallelism
Chan's BFS [Chan'12]	A general cluster	**Theory Improvement** All pair shortest paths algorithm with $O(mn/\log n)$ work	Bitwise
Sequential C-BFS [AIY'12]	A star-shaped cluster	**Efficient Implementation** Exact two-hop distance oracle	Bitwise
Ligra BFS [ShunBlelloch'13]	A single vertex	**Highly-Optimized Building Block** Forward-backward optimization	Thread
Our algorithm	**A general cluster**	**Theory Guarantee** **Efficient Implementation** **Highly-Optimized** **Applications to Shortest Paths**	**Bitwise + Thread**

Representation: Cluster Distances

The property of 'clusters'

- **Fact: On an unweighted graph $G = (V, E)$, if the distance between vertex s_1 and s_2 is d, then for any vertex $v \in V$, $|\delta(s_1, v) - \delta(s_2, v)| \le d$.**

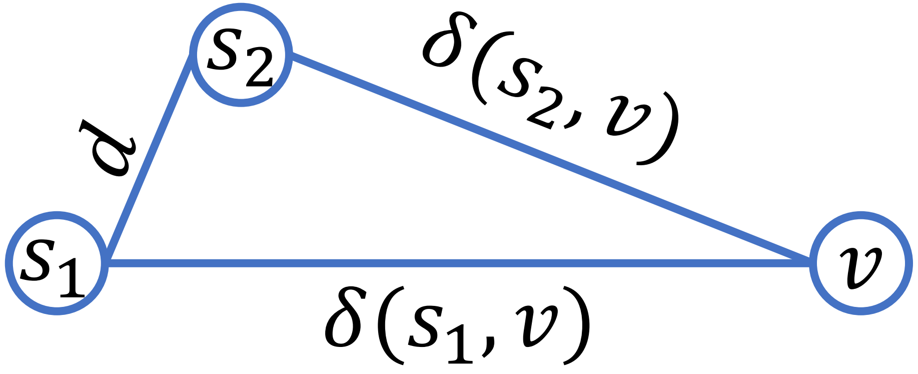


The property of 'clusters'

- Fact: On an unweighted graph $G = (V, E)$, if the distance between vertex s_1 and s_2 is d, then for any vertex $v \in V$, $|\delta(s_1, v) - \delta(s_2, v)| \leq d$.

The property of 'clusters'

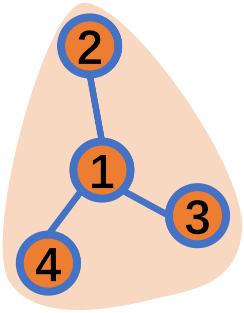
- **Fact: On an unweighted graph $G = (V, E)$, if the distance between vertex s_1 and s_2 is d, then for any vertex $v \in V$, $|\delta(s_1, v) - \delta(s_2, v)| \le d$.**



The property of 'clusters'

- Fact: On an unweighted graph $G = (V, E)$, if the distance between vertex s_1 and s_2 is d, then for any vertex $v \in V$, $|\delta(s_1, v) - \delta(s_2, v)| \le d$.

$d = 2$

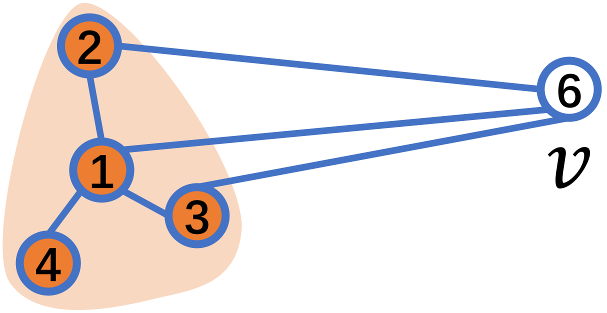


S

The property of ‘clusters’

- **Fact: On an unweighted graph $G = (V, E)$, if the distance between vertex s_1 and s_2 is d, then for any vertex $v \in V$, $|\delta(s_1, v) - \delta(s_2, v)| \le d$.**

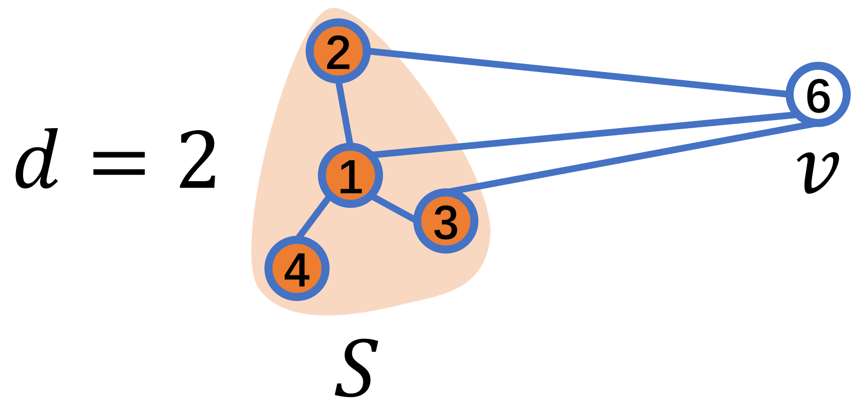
$d = 2$



S

The property of 'clusters'

- Fact: On an unweighted graph $G = (V, E)$, if the distance between vertex s_1 and s_2 is d, then for any vertex $v \in V$, $|\delta(s_1, v) - \delta(s_2, v)| \leq d$.



- Corollary: On an unweighted graph $G = (V, E)$, given a set S of vertices with diameter no more than d, for any vertex $v \in V$,

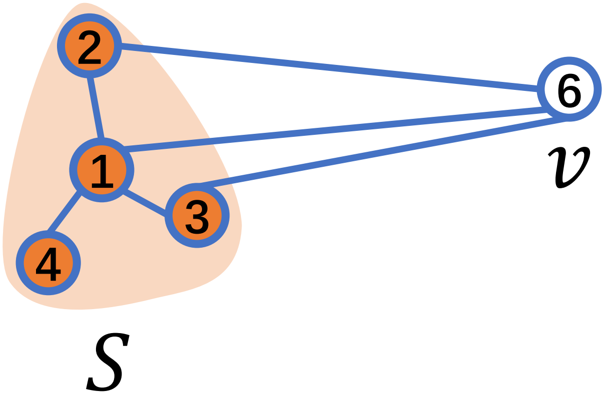
$$\max_{s \in S} \delta(s, v) - \min_{s \in S} \delta(s, v) \leq d$$

The property of 'clusters'

- Corollary: On an unweighted graph $G = (V, E)$, given a set S of vertices with diameter no more than d, for any vertex $v \in V$,

$$\max_{s \in S} \delta(s, v) - \min_{s \in S} \delta(s, v) \leq d$$

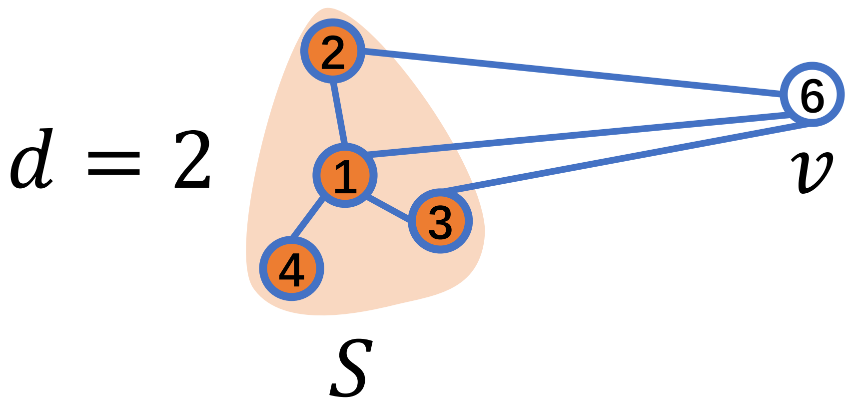
$d = 2$



The property of 'clusters'

- Corollary: On an unweighted graph $G = (V, E)$, given a set S of vertices with diameter no more than d, for any vertex $v \in V$,

$$\max_{s \in S} \delta(s, v) - \min_{s \in S} \delta(s, v) \le d$$

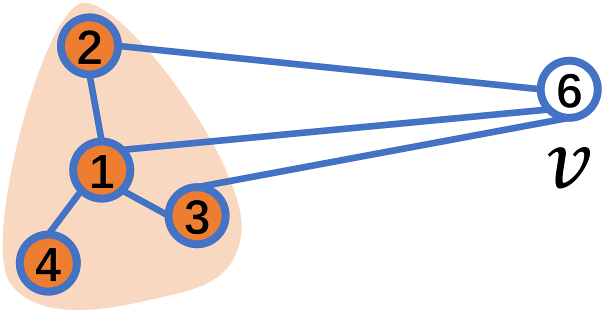


- Let $\Delta_v = \min_{s \in S} \delta(s, v)$, the distance between v and any $s \in S$ is in range $[\Delta_\mathrm{v}, \Delta_\mathrm{v} + d]$

The property of ‘clusters’

- Let $\Delta_v = \min_{s \in S} \delta(s, v)$, the distance between v and any $s \in S$ is in range $[\Delta_\mathrm{v}, \Delta_\mathrm{v} + d]$

$d = 2$



S

- Let $S_v[i]$ to be the subset of S that has distances to v as $\Delta_v + i$

$$S_v[i] = \{s \in S \mid \delta(s, v) = \Delta_v + i\}$$

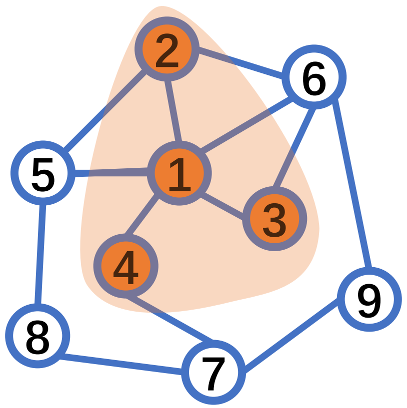
- The distance from cluster S to vertex v can be represented by tuple <Δ_v, $S_v[0, \ldots, \mathrm{d}]$>

Cluster Distance representations

- Let $S_v[i]$ to be the subset of S that has distances to v as $\Delta_v + i$

$$S_v[i] = \{s \in S \mid \delta(s, v) = \Delta_v + i\}$$

- The distance from cluster S to vertex v can be represented by tuple <Δ_v, $S_v[0, \dots, \mathrm{d}]$>



$k = 4$
$d = 2$

	$\delta(1,v)$	$\delta(2,v)$	$\delta(3,v)$	$\delta(4,v)$
7	2	3	3	1

k words

1 word

	Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
7	1	0b0001	0b1000	0b0110

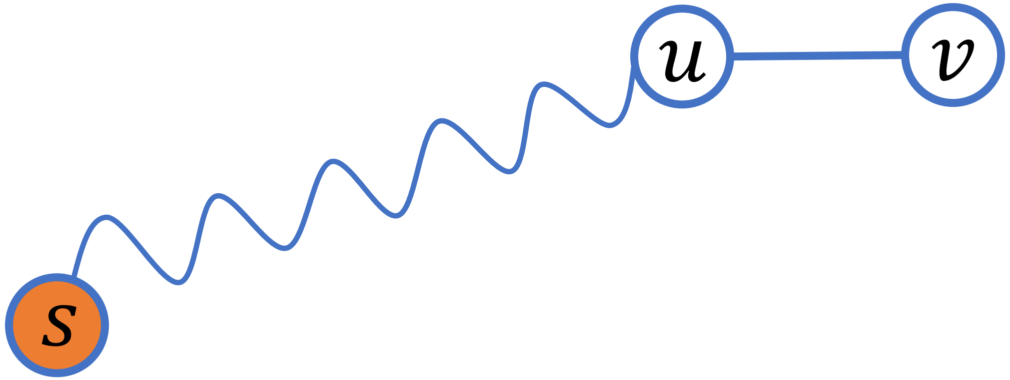
$d + 2$ words

1 word

Our Parallel Cluster-BFS

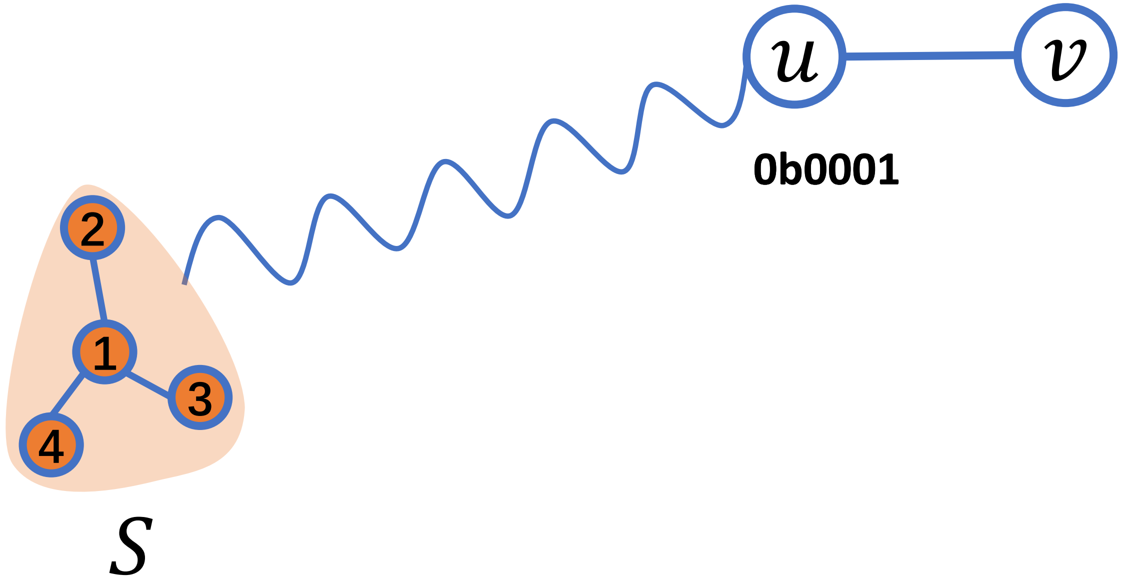
How to propagate BFS information?

- If u and v are neighbors, and there exists a path $s \rightsquigarrow u$ with length i-1
- then, there exists a path $s \rightsquigarrow v$ with length i



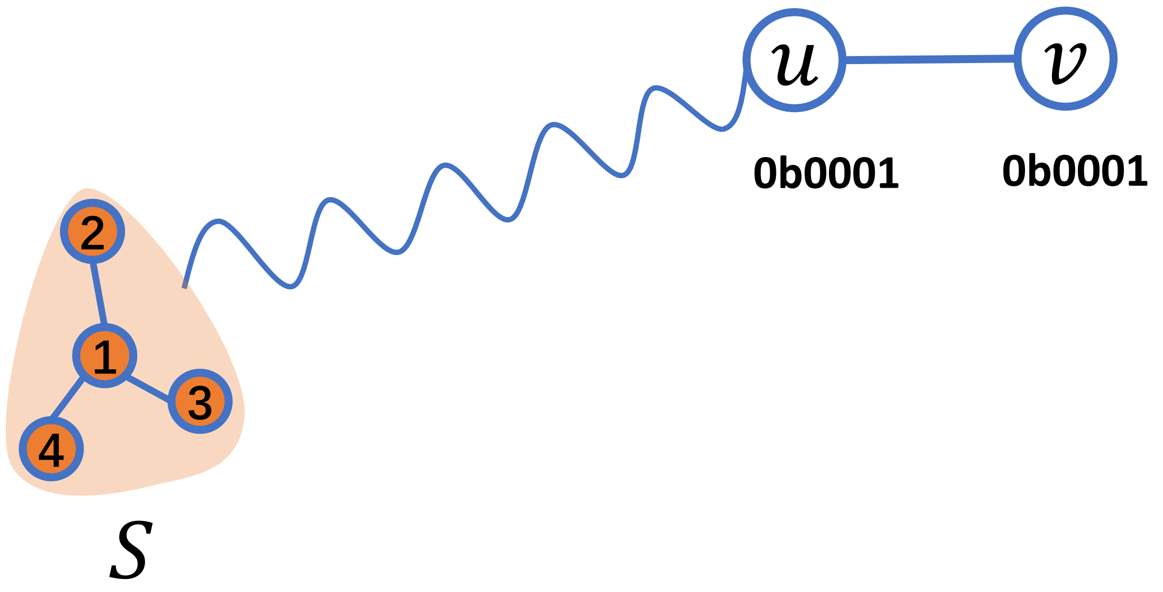
How to propagate BFS information?

- If u and v are neighbors, and there exists a path $s \rightsquigarrow u$ with length i-1
- then, there exists a path $s \rightsquigarrow v$ with length i



How to propagate BFS information?

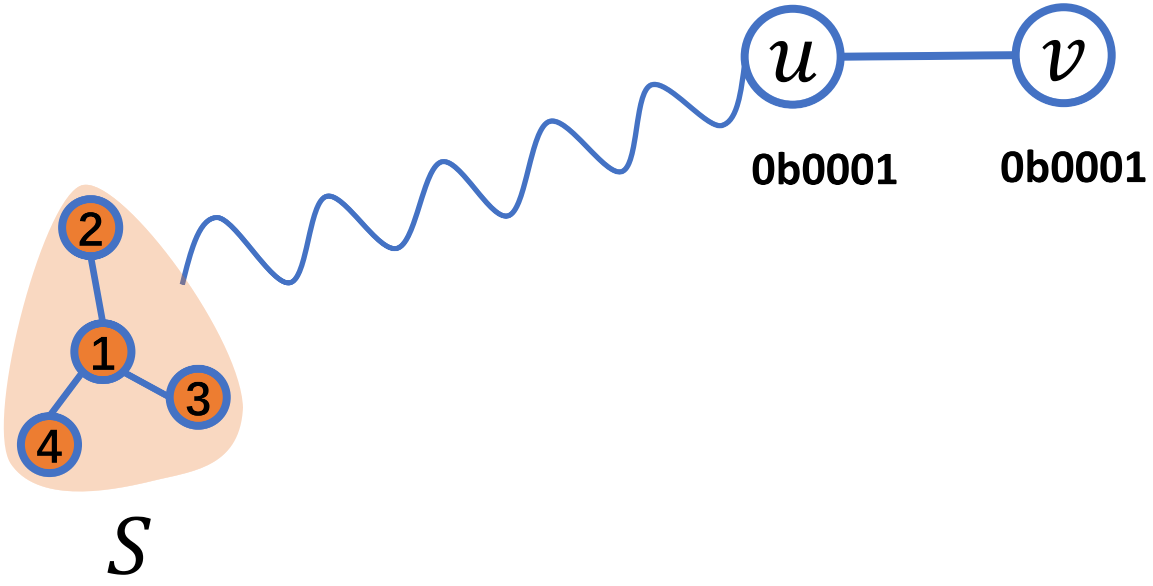
- If u and v are neighbors, and there exists a path $s \rightsquigarrow u$ with length i-1
- then, there exists a path $s \rightsquigarrow v$ with length i



How to propagate BFS information?

In round i:

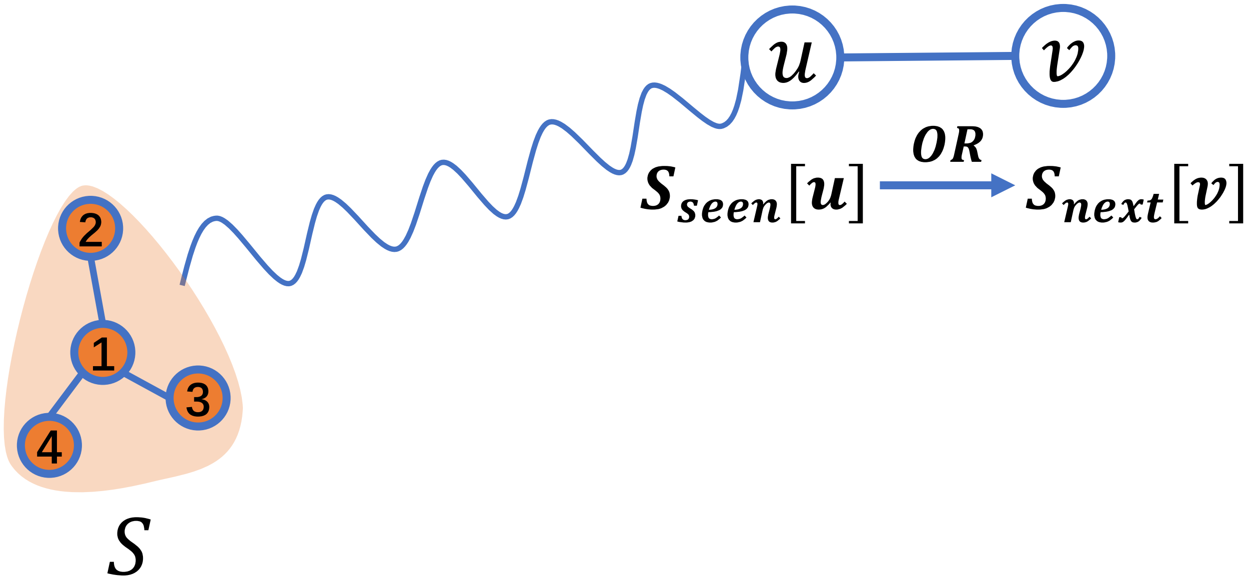
- $S_{seen}[v]$: subset of S whose distances to v is smaller than i
- $S_{next}[v]$: subset of S whose distances to v is smaller or equal to i



How to propagate BFS information?

In round i:

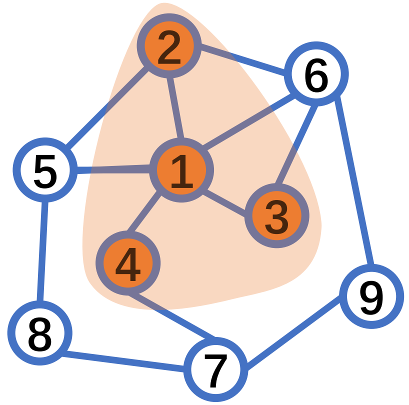
- $S_{seen}[v]$: subset of S whose distances to v is smaller than i
- $S_{next}[v]$: subset of S whose distances to v is smaller or equal to i



Our Cluster-BFS

- **Step 1: Initialization**
- **Step 2: Traversing**
  - Put the sources to the first frontier.
  - While the frontier is not empty:
    - Step 2.1: Vertex Mapping
      - Update cluster-distance
    - Step 2.2: Edge Traversing
      - Propagate BFS information and put updated vertices to the next frontier

Our Cluster-BFS: 1. Initialization



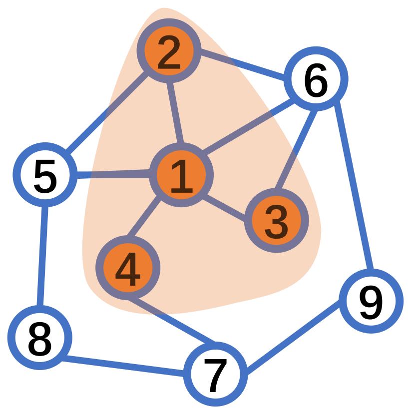
- $\Delta_v \leftarrow \infty$
- $S_{seen}[v] \leftarrow \emptyset$
- For $s \in S$, $S_{next}[s] \leftarrow \{s\}$; $v \in V \backslash S$, $S_{next}[v] \leftarrow \emptyset$

$$\mathcal{F}_0 = \{1\ 2\ 3\ 4\}$$

	$S_{seen}[v]$	$S_{next}[v]$
1	0b0000	0b0000
2	0b0000	0b0000
3	0b0000	0b0000
4	0b0000	0b0000
5	0b0000	0b0000
6	0b0000	0b0000
7	0b0000	0b0000
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
∞			
∞			
∞			
∞			
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: 1. Initialization



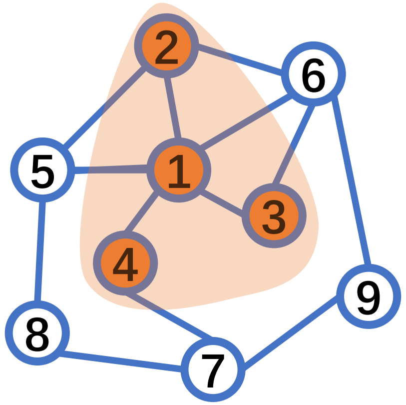
- $\Delta_v \leftarrow \infty$
- $S_{seen}[v] \leftarrow \emptyset$
- For $s \in S$, $S_{next}[s] \leftarrow \{s\}$; $v \in V \backslash \mathrm{S}$, $S_{next}[v] \leftarrow \emptyset$

$$\mathcal{F}_0 = \{1\ 2\ 3\ 4\}$$

	$S_{seen}[v]$	$S_{next}[v]$
1	0b0000	0b1000
2	0b0000	0b0100
3	0b0000	0b0010
4	0b0000	0b0001
5	0b0000	0b0000
6	0b0000	0b0000
7	0b0000	0b0000
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
∞			
∞			
∞			
∞			
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: 2.1 Vertices Mapping



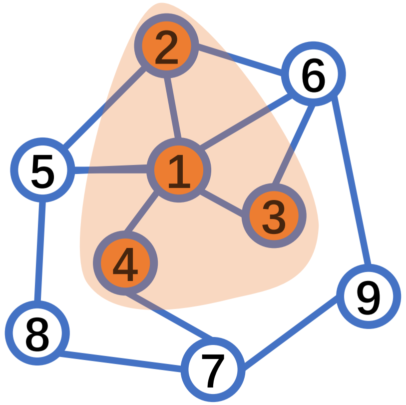
$\mathcal{F}_0 = \{1\ 2\ 3\ 4\}$

```
ParallelForEach u ∈ F_i do{
   if Δ_u = ∞ then {Δ_u = i}
   S_v[i − Δ_u] ← S_next[u]/S_seen[u]
   S_seen[u] ← S_seen[u] ∪ S_next[u]
}
```

	$S_{seen}[v]$	$S_{next}[v]$
1	0b0000	0b1000
2	0b0000	0b0100
3	0b0000	0b0010
4	0b0000	0b0001
5	0b0000	0b0000
6	0b0000	0b0000
7	0b0000	0b0000
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
∞			
∞			
∞			
∞			
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: 2.1 Vertices Mapping



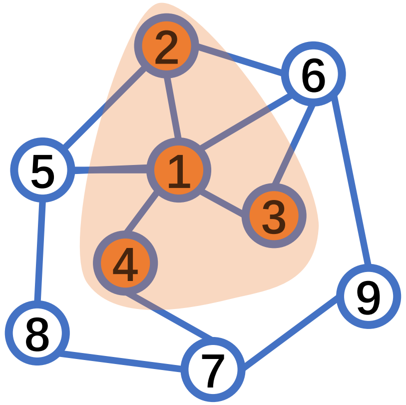
$\mathcal{F}_0 = \{1\ 2\ 3\ 4\}$

ParallelForEach $u \in \mathcal{F}_i$ do{
  if $\Delta_u = \infty$ then $\{\Delta_u = i\}$
  $S_v[i - \Delta_u] \leftarrow S_{next}[u]/S_{seen}[u]$
  $S_{seen}[u] \leftarrow S_{seen}[u] \cup S_{next}[u]$
}

	$S_{seen}[v]$	$S_{next}[v]$
1	0b0000	0b1000
2	0b0000	0b0100
3	0b0000	0b0010
4	0b0000	0b0001
5	0b0000	0b0000
6	0b0000	0b0000
7	0b0000	0b0000
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
0			
0			
0			
0			
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: 2.1 Vertices Mapping



$$\mathcal{F}_0 = \{ ①\ ②\ ③\ ④ \}$$

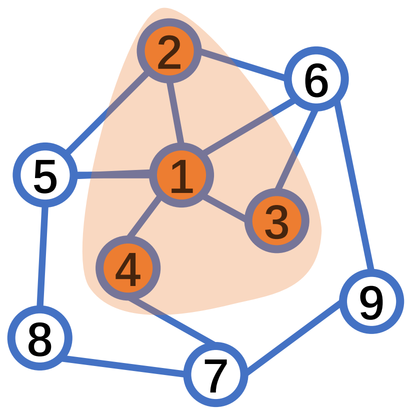
```
ParallelForEach u ∈ F_i do{
  if Δ_u = ∞ then {Δ_u = i}
  S_v[i − Δ_u] ← S_next[u]/S_seen[u]
  S_seen[u] ← S_seen[u] ∪ S_next[u]
}
```

$S_{next}[v] \setminus S_{seen}[v]$: subset of $\boldsymbol{S}$ whose distance to $\boldsymbol{v}$ is $\boldsymbol{i}$

	$S_{seen}[v]$	$S_{next}[v]$
1	0b0000	0b1000
2	0b0000	0b0100
3	0b0000	0b0010
4	0b0000	0b0001
5	0b0000	0b0000
6	0b0000	0b0000
7	0b0000	0b0000
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
0	0b1000		
0	0b0100		
0	0b0010		
0	0b0001		
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: 2.1 Vertices Mapping



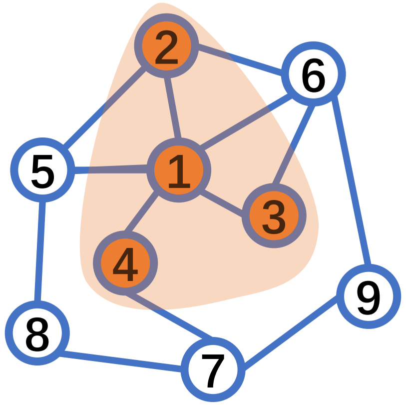
$\mathcal{F}_0 = \{1\ 2\ 3\ 4\}$

ParallelForEach $u \in \mathcal{F}_i$ do{
  if $\Delta_u = \infty$ then $\{\Delta_u = i\}$
  $S_v[i - \Delta_u] \leftarrow S_{next}[u] / S_{seen}[u]$
  $S_{seen}[u] \leftarrow S_{seen}[u] \cup S_{next}[u]$
}

	$S_{seen}[v]$	$S_{next}[v]$
1	0b1000	0b1000
2	0b0100	0b0100
3	0b0010	0b0010
4	0b0001	0b0001
5	0b0000	0b0000
6	0b0000	0b0000
7	0b0000	0b0000
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
0	0b1000		
0	0b0100		
0	0b0010		
0	0b0001		
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: 2.2 Edge Traversing



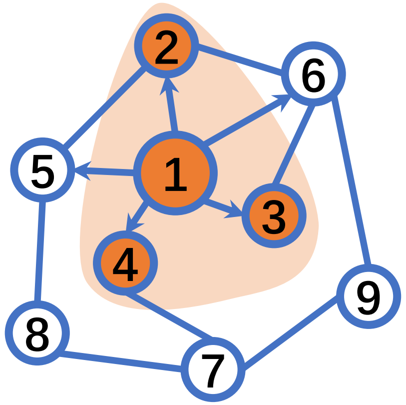
$$\mathcal{F}_0 = \{1\ 2\ 3\ 4\}$$

```
ParallelForEach u ∈ F_i do{
  ParallelForEach v ∈ N(u) do{
    if FetchAndOR(S_next[v],S_seen[u]){
      F_{i+1} = F_{i+1} ∪ {v}
    }
  }
}
```

	$S_{seen}[v]$	$S_{next}[v]$
1	0b1000	0b1000
2	0b0100	0b0100
3	0b0010	0b0010
4	0b0001	0b0001
5	0b0000	0b0000
6	0b0000	0b0000
7	0b0000	0b0000
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
0	0b1000		
0	0b0100		
0	0b0010		
0	0b0001		
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: 2.2 Edge Traversing



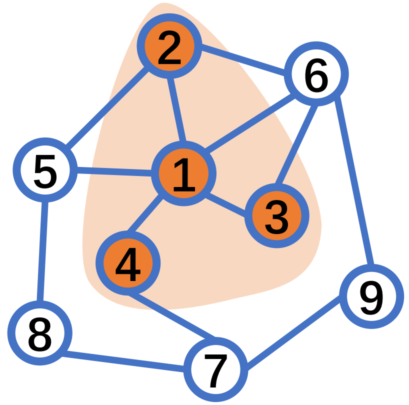
$$\mathcal{F}_0 = \{1\ 2\ 3\ 4\}$$

ParallelForEach $u \in \mathcal{F}_i$ do{
  ParallelForEach $v \in N(u)$ do{
    if FetchAndOR($S_{next}[v]$,$S_{seen}[u]$){
      $\mathcal{F}_{i+1} = \mathcal{F}_{i+1} \cup \{v\}$
    }
  }
}

	$S_{seen}[v]$	$S_{next}[v]$
1	0b1000	0b1000
2	0b0100	0b1100
3	0b0010	0b1010
4	0b0001	0b1001
5	0b0000	0b1000
6	0b0000	0b1000
7	0b0000	0b0000
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
0	0b1000		
0	0b0100		
0	0b0010		
0	0b0001		
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: 2.2 Edge Traversing



$$\mathcal{F}_0 = \{1\ 2\ 3\ 4\}$$

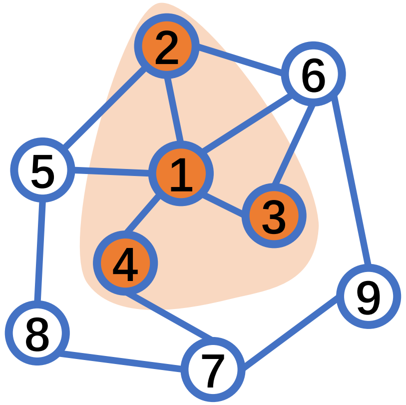
$$\mathcal{F}_1 = \{2\ 3\ 4\ 5\ 6\ 1\ 7\}$$

```
ParallelForEach u ∈ F_i do{
  ParallelForEach v ∈ N(u) do{
    if FetchAndOR(S_next[v],S_seen[u]){
      F_{i+1} = F_{i+1} ∪ {v}
    }
  }
}
```

	$S_{seen}[v]$	$S_{next}[v]$
1	0b1000	0b1111
2	0b0100	0b1100
3	0b0010	0b1010
4	0b0001	0b1001
5	0b0000	0b1100
6	0b0000	0b1110
7	0b0000	0b0001
8	0b0000	0b0000
9	0b0000	0b0000

Δ_v	$S_v[0]$	$S_v[1]$	$S_v[2]$
0	0b1000		
0	0b0100		
0	0b0010		
0	0b0001		
∞			
∞			
∞			
∞			
∞			

Our Cluster-BFS: Traversing Optimization



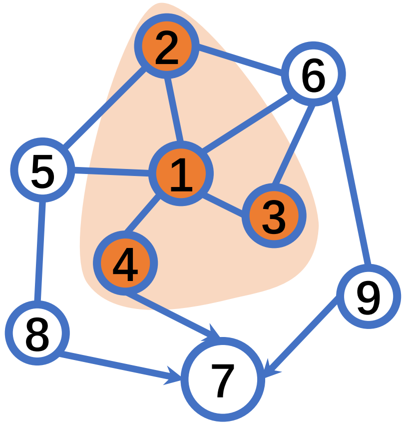
Forward Traversing

ParallelForEach $u \in \mathcal{F}_i$ do{
  ParallelForEach $v \in N(u)$ do{
    if FetchAndOR($S_{next}[v]$,$S_{seen}[u]$){
      $\mathcal{F}_{i+1} = \mathcal{F}_{i+1} \cup \{v\}$
    }
  }
}

Backward Traversing

ParallelForEach $v \in V$ do{
  ParallelForEach $u \in N(v)$ and $u \in \mathcal{F}_i$ do{
    if $S_{next}[v] \leftarrow S_{next}[v] \cup S_{seen}[u]${
      $\mathcal{F}_{i+1} = \mathcal{F}_{i+1} \cup \{v\}$
    }
  }
}

Our Cluster-BFS: Traversing Optimization



Forward Traversing

ParallelForEach $u \in \mathcal{F}_i$ do{
  ParallelForEach $v \in N(u)$ do{
    if $\boxed{\text{FetchAndOR}(S_{next}[v], S_{seen}[u])}${
      $\mathcal{F}_{i+1} = \mathcal{F}_{i+1} \cup \{v\}$
    }
  }
}

Atomic Operation
1 random read+write: $S_{next}[v]$

Backward Traversing

ParallelForEach $v \in V$ do{
  ParallelForEach $u \in N(v)$ and $u \in \mathcal{F}_i$ do{
    if $\boxed{S_{next}[v] \leftarrow S_{next}[v] \cup S_{seen}[u]}${
      $\mathcal{F}_{i+1} = \mathcal{F}_{i+1} \cup \{v\}$
    }
  }
}

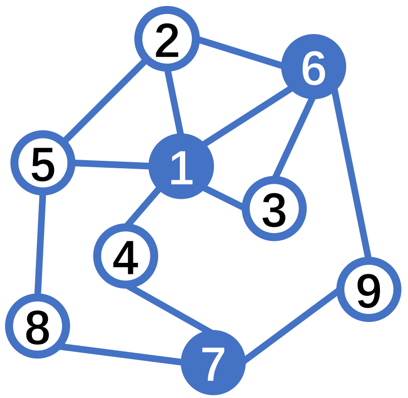
Normal Operation
1 random read: $S_{seen}[u]$

Applications: Approximate Distance Oracles (ADO) for Unweighted Graphs

ADO based on Landmark Labeling

- **Landmark Labeling (LL) is one of the most widely-used and probably the simplest ADO**
- **The basic idea is to select a subset H of vertices as landmarks, and compute the distance from landmark to all the vertices**
  - i.e. compute $\boldsymbol{\delta(h, v)}$, $\forall h \in H$ and $\forall v \in V$
- **Answering $\boldsymbol{query(u, v)}$:**
  - $\min_{h\in H} \delta(h, v) + \delta(h, u)$

$query(2,9) =$

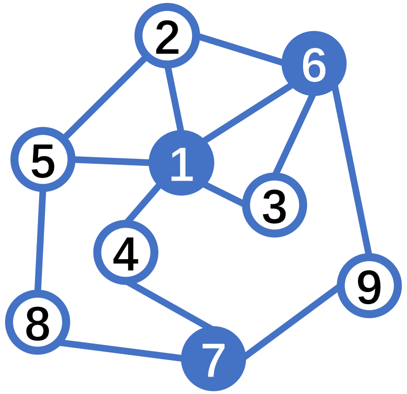


v	1	2	3	4	5	6	7	8	9
1	0	1	1	1	1	1	2	2	2
6	1	1	1	2	2	0	2	3	1
7	2	3	3	1	3	2	0	1	1

ADO based on Landmark Labeling

- **Landmark Labeling (LL) is one of the most widely-used and probably the simplest ADO**
- **The basic idea is to select a subset H of vertices as landmarks, and compute the distance from landmark to all the vertices**
  - i.e. compute $\boldsymbol{\delta(h, v)}$, $\forall h \in H$ and $\forall v \in V$
- **Answering $\boldsymbol{query(u, v)}$:**
  - $\min_{h \in H} \delta(h, v) + \delta(h, u)$

$$query(2,9) = \min\{3,2,4\}$$

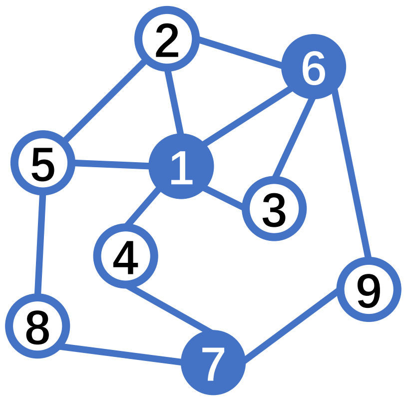


v	1	2	3	4	5	6	7	8	9	
1	0	1	1	1	1	1	2	2	2	3
6	1	1	1	2	2	0	2	3	1	2
7	2	3	3	1	3	2	0	1	1	4

ADO based on Landmark Labeling

- **Landmark Labeling (LL) is one of the most widely-used and probably the simplest ADO**
- **The basic idea is to select a subset H of vertices as landmarks, and compute the distance from landmark to all the vertices**
  - i.e. compute $\boldsymbol{\delta(h, v)}$, $\forall h \in H$ and $\forall v \in V$
- **Answering $\boldsymbol{query(u, v)}$:**
  - $\min_{h \in H} \delta(h, v) + \delta(h, u)$

$$query(2,9) = \min\{3,2,4\}$$



v	1	2	3	4	5	6	7	8	9	
1	0	1	1	1	1	1	2	2	2	3
6	1	1	1	2	2	0	2	3	1	2
7	2	3	3	1	3	2	0	1	1	4

- **The more landmarks selected, the more accurate the answers will be**

Our ADO based on Cluster Landmark Labeling

- **Landmark Labeling (LL) is one of the most widely-used and probably the simplest ADO**
- **We select a cluster as a landmark instead of a vertex, and store the cluster-distances to all vertices $v \in V$**
- **Answering query(u,v):**
  - First, find minimum distance between u and v through a cluster
  - Then, find the minimum value among all clusters

cluster	1	2	3	4	5	6	7	8	9
S_a	$\Delta_1, S_1[1,..d]$	$\Delta_2, S_2[1,..d]$	$\Delta_3, S_3[1,..d]$	$\ldots$	$\ldots$				
S_b	$\Delta_1, S_1[1,..d]$	$\Delta_2, S_2[1,..d]$	$\Delta_3, S_3[1,..d]$	$\ldots$	$\ldots$				

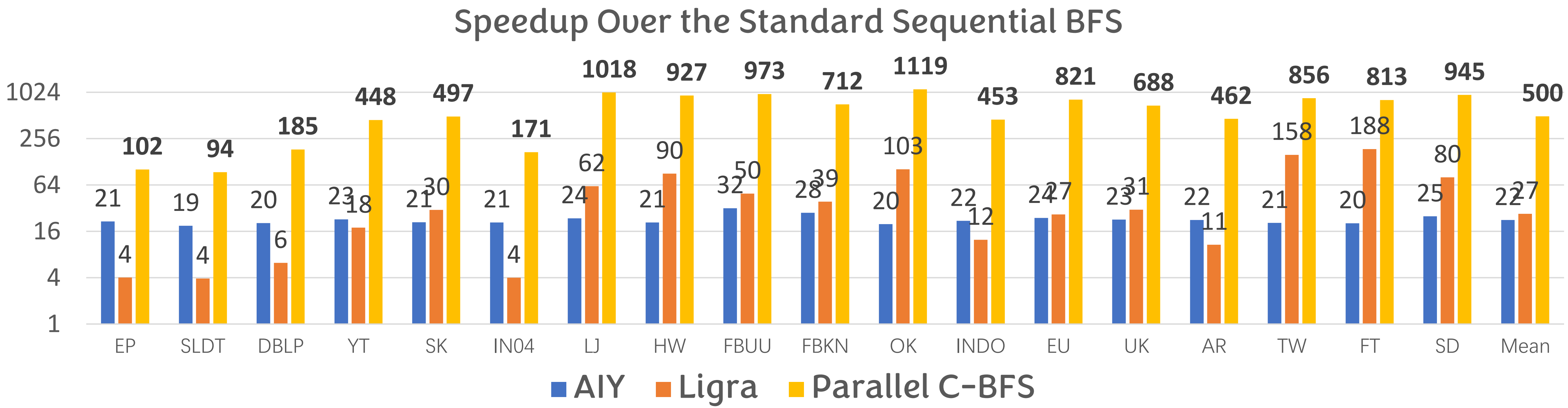
Experiments

Setup

- **Machine:**
  - 96 cores
  - 1.5 TB memory
- **Graphs:**
  - 18 undirected graphs, which are either social or web graphs with low diameters

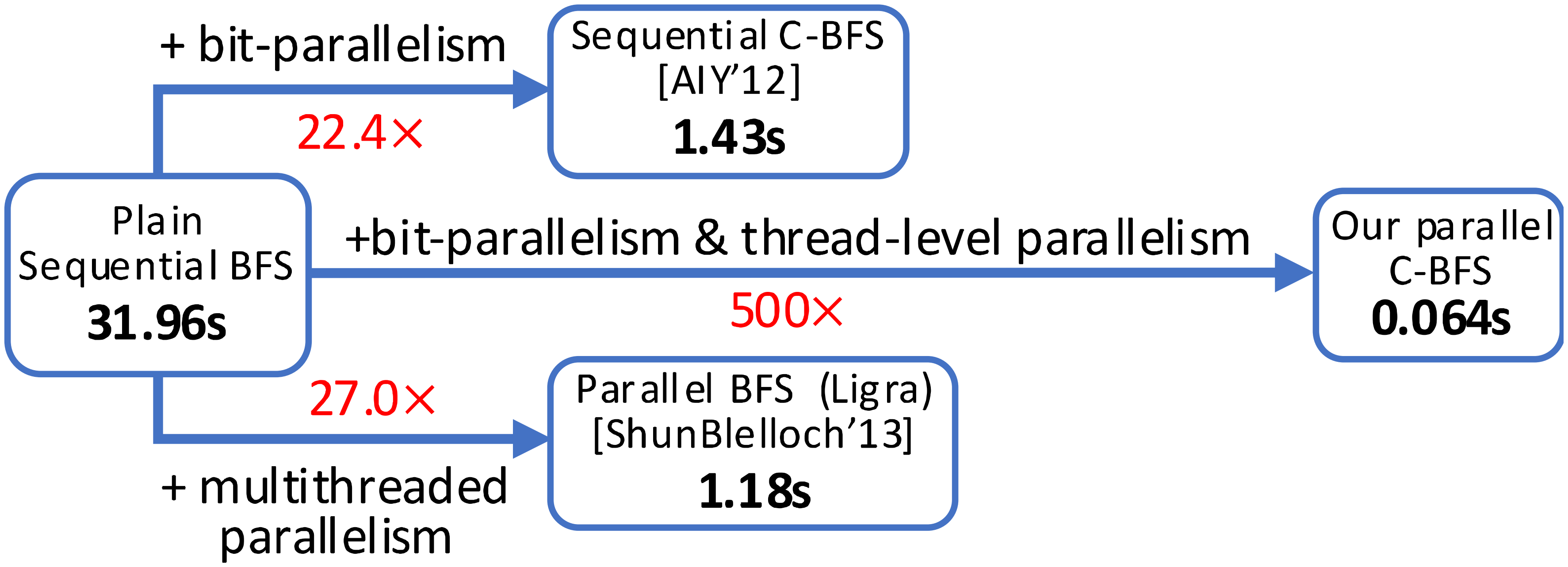
Microbenchmarks for Cluster-BFS

- **Tested on 10 random clusters of size k=64 and d=2 and took the average**
- **Baselines:**
  - Plain: the standard sequential single source BFS
  - AIY[AIY'12]: sequential cluster-BFS – only with bitwise-parallelism
  - Ligra[ShunBlelloch'13]: parallel single-source BFS – only with thread-parallelism



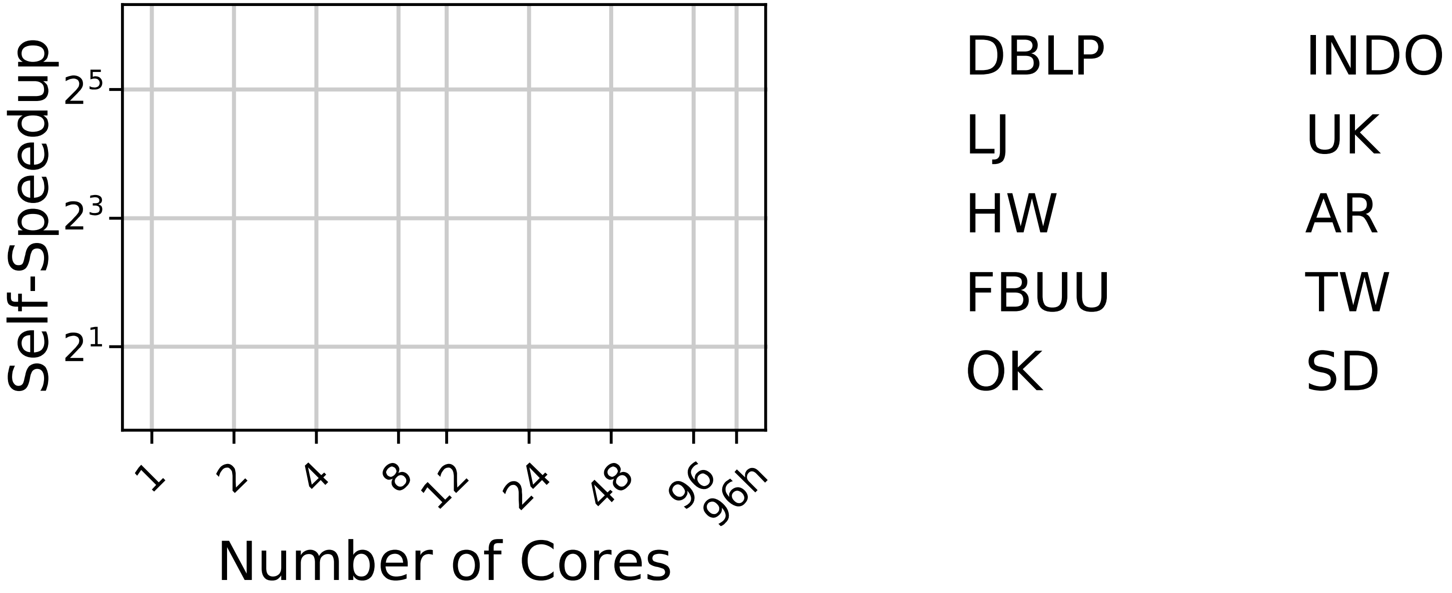
Microbenchmarks for Cluster-BFS

- **Tested on 10 random clusters with size k=64 and d=2 and took the average**
- **Baselines:**
  - Plain: the standard sequential single source BFS
  - AIY[AIY'12]: sequential cluster-BFS – only with bitwise-parallelism
  - Ligra[ShunBlelloch'13]: parallel single-source BFS – only with thread-parallelism



Microbenchmarks for Cluster-BFS

- **Scalability**



Approximate Landmark Labeling

- **Comparison between cluster-based landmark labeling with Plain**
  - Control the **index size** to be same (1024 bytes/vertex), and compare **preprocessing time** and **accuracy**

Data	Index Time (s)			ϵ (%)		
	Plain	$k=64$	$k=8$	Plain	$k=64$	$k=8$
EP	1.26	0.02	0.08	0.4	0.1	0.1
SLDT	1.15	0.02	0.07	0.7	0.1	0.1
DBLP	3.57	0.08	0.25	2.5	2.2	1.0
YT	9.22	0.23	0.59	0.3	0.3	0.1
SK	13.4	0.55	1.77	1.4	0.7	0.4
IN04	20.0	0.96	3.88	2.1	1.9	0.9
LJ	36.2	1.72	5.63	5.0	4.3	3.5
HW	12.4	0.93	4.10	10.6	5.6	7.1
FBUU	138	11.3	27.0	6.2	11.9	6.9
FBKN	127	10.5	24.9	6.2	11.9	6.9
OK	26.3	2.87	10.1	8.7	7.7	7.3
INDO	83.2	5.44	29.8	3.1	1.5	1.3
EU	87.3	7.01	34.9	2.6	1.3	1.7
UK	80.4	8.28	38.8	3.9	4.9	3.1
AR	148	17.6	86.8	2.6	4.0	2.2
TW	112	31.0	99.3	1.5	1.4	1.1
FT	251	61.1	193	16.8	12.4	12.8
SD	318	75.6	255	0.6	0.3	0.3

$$\epsilon = \frac{query(u,v)}{\delta(u,v)} - 1,$$

smaller is better

Approximate Landmark Labeling

- **The Performance of Different d on Landmark Labeling**
  - Control the index size to be same, compare the preprocessing time and accuracy.

Data	Construction Time (s)				Distortion ϵ (%)			
	Plain	$d=2$	$d=3$	$d=4$	Plain	$d=2$	$d=3$	$d=4$
EP	1.26	0.03	0.02	0.02	0.4	0.1	0.2	0.2
SLDT	1.15	0.02	0.02	0.02	0.7	0.1	0.4	0.6
DBLP	3.57	0.09	0.07	0.06	2.5	2.2	3.4	4.0
YT	9.22	0.23	0.19	0.17	0.3	0.3	0.6	0.8
SK	13.4	0.58	0.41	0.37	1.4	0.7	1.5	2.1
IN04	20.0	1.06	0.70	0.58	2.1	1.9	2.4	2.6
LJ	36.2	1.79	1.41	1.26	5.0	4.3	5.5	6.0
HW	12.4	0.99	0.75	0.63	10.6	5.6	6.5	7.0
FBUU	138	11.7	9.31	8.26	6.2	11.9	13.4	15.9
FBKN	127	10.8	8.82	7.52	6.2	11.9	13.5	16.0
OK	26.3	3.05	2.65	2.32	8.7	7.7	7.5	7.6
INDO	83.2	6.09	3.92	3.40	3.1	1.5	2.2	2.6
EU	87.3	8.00	6.33	5.55	2.6	1.3	1.9	2.2
UK	80.4	9.06	6.63	5.90	3.9	4.9	5.3	5.6
AR	148	19.4	13.4	11.7	2.6	4.0	6.8	7.1
TW	112	31.0	27.8	23.9	1.5	1.4	1.5	1.8
FT	251	61.1	52.0	45.8	16.8	12.4	13.7	14.6
SD	318	75.6	61.2	53.1	0.6	0.3	0.8	0.9

Summary

- **Present a parallel implementation for cluster-BFS**
  - The first one that combines the bitwise-parallelism and thread-parallelism
  - The implementation is highly efficient that both techniques still fully contribute to the performance
- **Applications: Distance Oracles**
  - By applying cluster-BFS, the landmark labeling-based ADO improves preprocessing time and accuracy
- **Our cluster-BFS is the first implementation support clusters with general $\boldsymbol{d}$**
  - We tested the performance of cluster BFS-based ADO under different d