Graph Streaming

Christian Wulff-Nilsen
Algorithmic Techniques for Modern Data Models
DTU

November $14, 2025$

Overview for today

- Graph streaming model
- Graph connectivity
- Bipartiteness testing
- Distance estimation and spanners

Graph Streaming

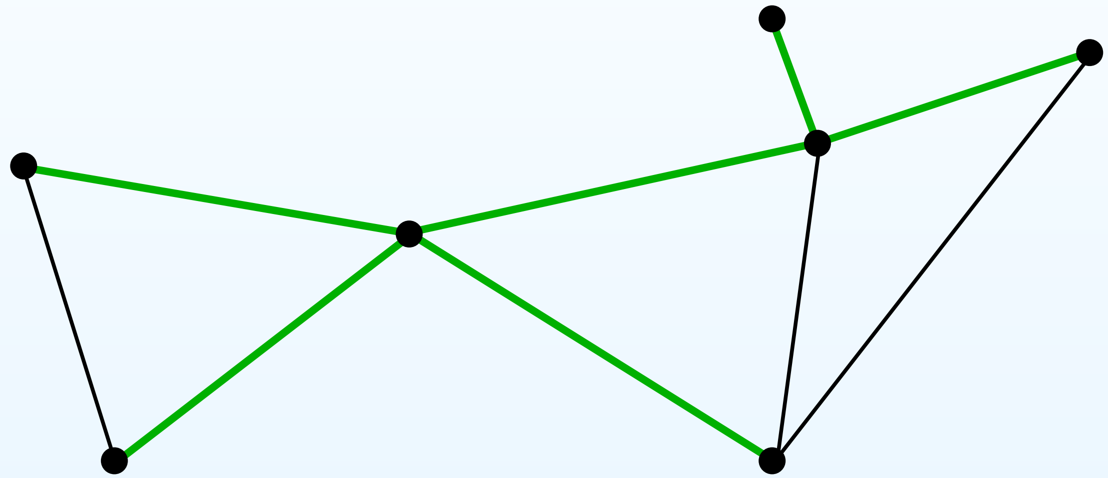
- Stream consists of the edges of E for a graph $G = (V, E)$
- Each edge of E occurs exactly once in the stream
- $V = [n]$ and n is known by the algorithm
- This is known as a *vanilla* or *insertion-only* graph stream
- We let $m = |E|$ be the stream length
- We allow space $O(n(\log n)^c)$, constant c ("semi-streaming" algorithm)
- One exception is our spanner algorithm at the end of the lecture

Graph Connectivity

- G is undirected
- Problem: determine if G is connected
- Easy with $O(m + n)$ words of space
- We give an algorithm using only $O(n \log n)$ *bits* of space
- This is close to a lower bound of $\Omega(n)$ bits

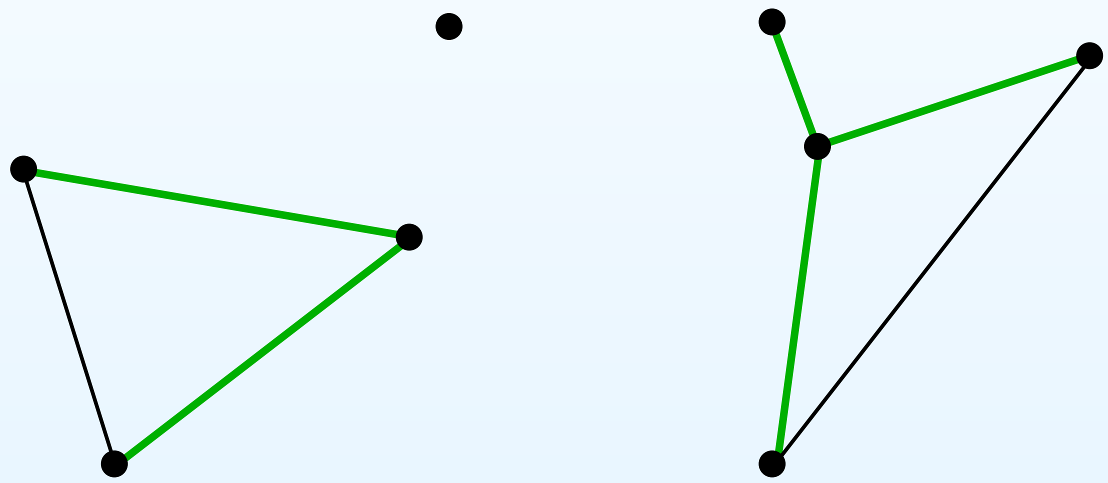
Spanning Tree and Spanning Forest

- *Spanning tree* of a connected graph $G = (V, E)$: a tree in G with vertex set V



Spanning Tree and Spanning Forest

- *Spanning tree* of a connected graph $G = (V, E)$: a tree in G with vertex set V
- *Spanning forest* of a graph $G = (V, E)$: consists of a spanning tree for each connected component of G



Spanning Tree and Spanning Forest

- *Spanning tree* of a connected graph $G = (V, E)$: a tree in G with vertex set V
- *Spanning forest* of a graph $G = (V, E)$: consists of a spanning tree for each connected component of G
- G is connected $\Leftrightarrow$ any spanning forest of G is a spanning tree of G
- Any spanning tree of a connected k-vertex graph has $k - 1$ edges
- Thus, G is connected $\Leftrightarrow$ any spanning forest of G has $n - 1$ edges

Graph Connectivity Algorithm: Correctness

- Pseudo-code (assume G has more than one vertex):

Graph Connectivity Algorithm

Initialize: $F \leftarrow \emptyset$, connected $\leftarrow$ false

Process(token $\{u, v\}$):
  if $F \cup \{\{u, v\}\}$ has no cycle then
    $F \leftarrow F \cup \{\{u, v\}\}$
    if $|F| = n - 1$ then connected $\leftarrow$ true

Output: connected

- Correctness:
  - F is the edges of a spanning forest of the part of G seen so far (exercise)
  - At termination: $|F| = n - 1 \Leftrightarrow G$ is connected (previous slide)

Graph Connectivity Algorithm: Space Analysis

- Pseudo-code (assume G has more than one vertex):

Graph Connectivity Algorithm

Initialize: $F \leftarrow \emptyset$, connected $\leftarrow$ false

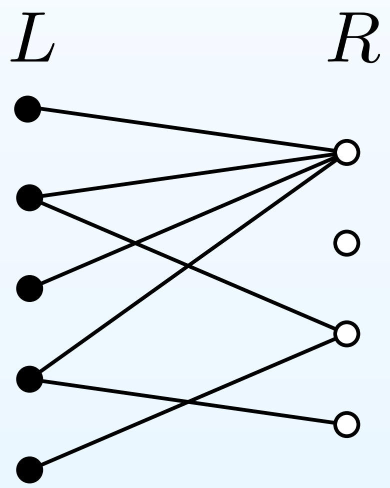
Process(token $\{u, v\}$):
 if $F \cup \{\{u, v\}\}$ has no cycle then
  $F \leftarrow F \cup \{\{u, v\}\}$
  if $|F| = n - 1$ then connected $\leftarrow$ true

Output: connected

- Space used:
  - Dominated by $|F|$
  - Since $|F| \leq n - 1$, the algorithm uses $O(n)$ words of space
  - This is $O(n \log n)$ bits

Bipartiteness Testing

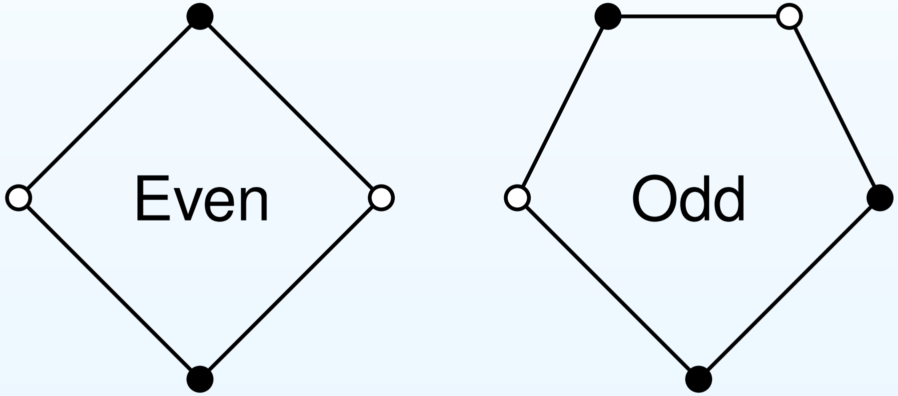
- An undirected graph $G = (V, E)$ is *bipartite* if there is a partition (L, R) of V where each edge has one endpoint in L and one in R
- Equivalent to saying that G is 2-colorable (exercise)



- Bipartiteness testing: determine if G is bipartite
- We give an algorithm using $O(n \log n)$ bits of space

Odd and Even Cycles

- A cycle is *odd* if it has an odd number of edges
- Otherwise, it is *even*
- An even cycle can be 2-colored; an odd cycle cannot:



Bipartiteness Testing Algorithm

- Pseudo-code:

Bipartiteness Testing Algorithm

Initialize: $F \leftarrow \emptyset$, is_bipartite $\leftarrow$ true

Process(token $\{u, v\}$):
if $F \cup \{\{u, v\}\}$ has no cycle then
  $F \leftarrow F \cup \{\{u, v\}\}$
else if $F \cup \{\{u, v\}\}$ has an odd cycle then
  is_bipartite $\leftarrow$ false

Output: is_bipartite

Bipartiteness Testing Algorithm: Space Analysis

- Pseudo-code:

Bipartiteness Testing Algorithm

Initialize: $F \leftarrow \emptyset$, is_bipartite $\leftarrow$ true

Process(token $\{u, v\}$):
  if $F \cup \{\{u, v\}\}$ has no cycle then
    $F \leftarrow F \cup \{\{u, v\}\}$
  else if $F \cup \{\{u, v\}\}$ has an odd cycle then
    is_bipartite $\leftarrow$ false

Output: is_bipartite

- Space is dominated by $|F|$
- F is a forest at any point in the algorithm
- Space in bits is thus $O(n \log n)$

Bipartiteness Testing Algorithm: Correctness

- Pseudo-code:

Bipartiteness Testing Algorithm

Initialize: $F \leftarrow \emptyset$, is_bipartite $\leftarrow$ true

Process(token $\{u, v\}$):
  if $F \cup \{\{u, v\}\}$ has no cycle then
    $F \leftarrow F \cup \{\{u, v\}\}$
  else if $F \cup \{\{u, v\}\}$ has an odd cycle then
    is_bipartite $\leftarrow$ false

Output: is_bipartite

- Assume first that the algorithm returns false
- Want to show that G is not bipartite
- When processing some $\{u, v\} \in E$, $F \cup \{\{u, v\}\}$ contained an odd cycle which cannot be 2-colored
- This odd cycle is in G so G is not bipartite

Bipartiteness Testing Algorithm: Correctness

- Pseudo-code:

Bipartiteness Testing Algorithm

Initialize: $F \leftarrow \emptyset$, is_bipartite $\leftarrow$ true

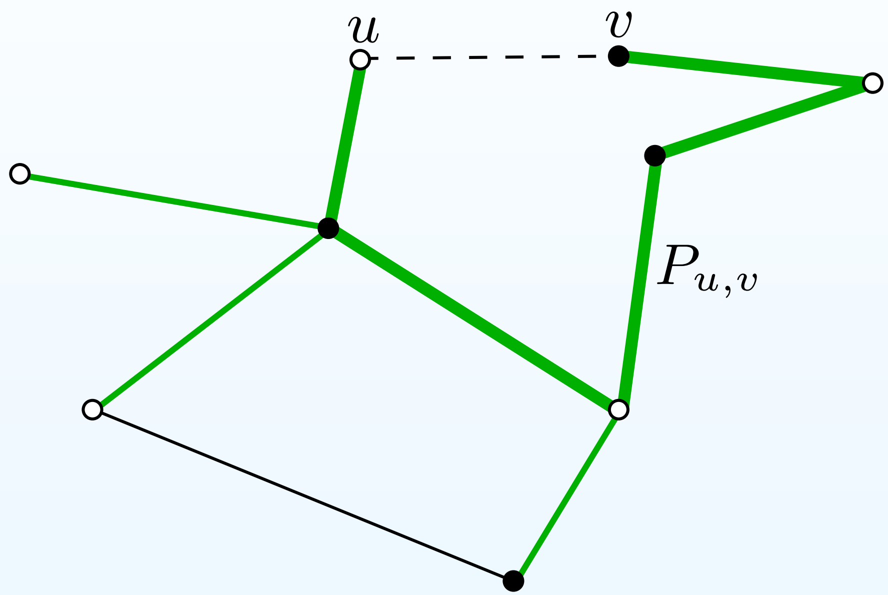
Process(token $\{u, v\}$):
 if $F \cup \{\{u, v\}\}$ has no cycle then
  $F \leftarrow F \cup \{\{u, v\}\}$
 else if $F \cup \{\{u, v\}\}$ has an odd cycle then
  is_bipartite $\leftarrow$ false

Output: is_bipartite

- Now, assume true is returned; want to show that G is 2-colorable
- Let $K : [n] \rightarrow \{0, 1\}$ be a 2-coloring of forest F (exercise)
- Will show that for any $\{u, v\} \in E$, $K(u) \neq K(v)$
- This is clear if $\{u, v\} \in F$ so assume $\{u, v\} \notin F$
- Let P_{uv} be the path from u to v in F (exercise)
- $C_{uv} = P_{uv} \cup \{\{u, v\}\}$ is a cycle in G
- When $\{u, v\}$ was processed, it reached the else case
- Algorithm returns true $\Rightarrow C_{uv}$ is even $\Rightarrow K(u) \neq K(v)$

Bipartiteness Testing Algorithm: Correctness

- Illustration:



- Now, assume true is returned; want to show that G is 2-colorable
- Let $K : [n] \to \{0, 1\}$ be a 2-coloring of forest F (exercise)
- Will show that for any $\{u, v\} \in E$, $K(u) \neq K(v)$
- This is clear if $\{u, v\} \in F$ so assume $\{u, v\} \notin F$
- Let P_{uv} be the path from u to v in F (exercise)
- $C_{uv} = P_{uv} \cup \{\{u, v\}\}$ is a cycle in G
- When $\{u, v\}$ was processed, it reached the else case
- Algorithm returns true $\Rightarrow$ C_{uv} is even $\Rightarrow$ $K(u) \neq K(v)$

Distance Estimation

- Given undirected unweighted graph $G = (V, E)$, parameter $t \geq 1$
- $\delta_G(u, v)$: shortest path distance between u and v in G
- Distance estimation:
  - Given query vertex pair (u, v)
  - Return stretch t estimate $\hat{\delta}_G(u, v)$ of $\delta_G(u, v)$:

$$\delta_G(u, v) \leq \hat{\delta}_G(u, v) \leq t \cdot \delta_G(u, v)$$

- t-*spanner* of G: subgraph H such that for all $u, v \in V$,

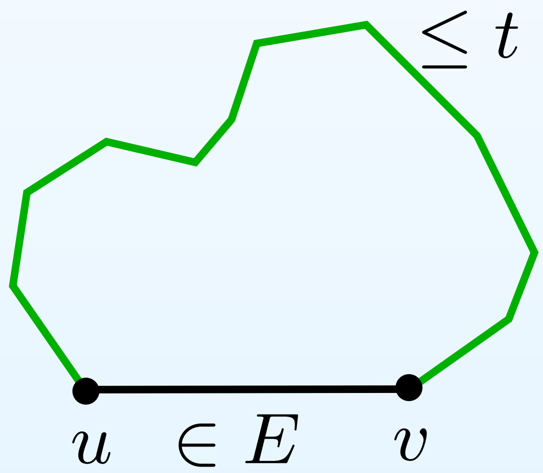
$$\delta_H(u, v) \leq t \cdot \delta_G(u, v)$$

- Since H is a subgraph of G, we also have $\delta_G(u, v) \leq \delta_H(u, v)$
- Our distance estimation algorithm maintains a t-spanner
- Queries are answered by simply running BFS in the t-spanner

t-Spanner Edge Property

- Let H be a subgraph of $G = (V, E)$ and let $t \geq 1$
- H has the t-*spanner edge property* if for all $\{u, v\} \in E$,

$$\delta_H(u, v) \leq t$$



t-Spanner Edge Property

- Claim: H is a t-spanner of $G \Leftrightarrow H$ has the t-spanner edge property
- Proof: exercise (and later in the lecture)

Spanner Algorithm

- Pseudo-code:

t-spanner Algorithm

Initialize: $H \leftarrow \emptyset$

Process(token $\{u, v\}$):
if $\delta_H(u, v) \geq t + 1$ then
$H \leftarrow H \cup \{\{u, v\}\}$

Output(x, y): $\delta_H(x, y)$

Spanner Algorithm: Correctness

- Pseudo-code:

t-spanner Algorithm

Initialize: $H \leftarrow \emptyset$

Process(token $\{u, v\}$):
  if $\delta_H(u, v) \geq t + 1$ then
    $H \leftarrow H \cup \{\{u, v\}\}$

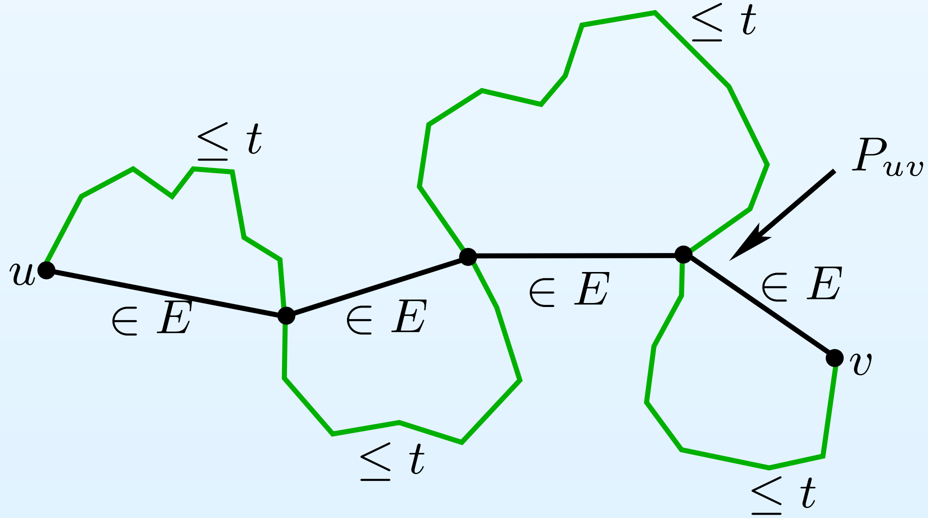
Output(x, y): $\delta_H(x, y)$

- When an edge $e = \{u, v\}$ is processed, either:
  - e was added to H, or
  - $\delta_H(u, v) \leq t$
- In both cases, $\delta_H(u, v) \leq t$ after e is processed
- This also holds at any later point as edges are not removed from H
- t-spanner edge property $\Rightarrow$ H is a t-spanner of processed edges
- At termination, H is a t-spanner of G

Proving the t-Spanner Edge Property

- Claim: H is a t-spanner of $G \Leftrightarrow H$ has the t-spanner edge property
- Proof of "$\Rightarrow$": $\{u, v\} \in E \Rightarrow \delta_G(u, v) = 1 \Rightarrow \delta_H(u, v) \leq t$
- Proof of "$\Leftarrow$":
  - Assume H has the t-spanner edge property and let $u, v \in V$
  - Let P_{uv} be a shortest path in G (trivial if P_{uv} does not exist)
  - For any edge $\{a, b\}$ of P_{uv}, $\delta_H(a, b) \leq t$
  - Thus, H has a path from a to b of length at most

$$t \cdot |P_{uv}| = t \cdot \delta_G(u, v)$$



Girth of a Graph

- *Girth* of a graph H: length of its shortest cycle
- We denote it by $\gamma(H)$
- If H is acyclic, we define $\gamma(H) = \infty$

Spanner Algorithm: Space Analysis

- Pseudo-code:

t-spanner Algorithm

Initialize: $H \leftarrow \emptyset$

Process(token $\{u, v\}$):
  if $\delta_H(u, v) \geq t + 1$ then
    $H \leftarrow H \cup \{\{u, v\}\}$

Output(x, y): $\delta_H(x, y)$

- Space is $O(|E(H)| \log n)$ bits
- It can be shown that $\gamma(H) \geq t + 2$ (exercise)
- We will use this to bound $|E(H)|$

Girth and Max Number of Edges

- Theorem:
  - Given undirected graph H with m edges and n vertices
  - Let $k \geq 2$ be an integer such that $\gamma(H) \geq k$
  - Then:

$$m \leq n + n^{1+1/\lfloor (k-1)/2 \rfloor}$$

- For the graph H in our algorithm, $\gamma(H) \geq t+2$
- Theorem with $k = t+2$: the number of edges in this graph is

$$O\left(n^{1+1/\lfloor (k-1)/2 \rfloor}\right) = O\left(n^{1+1/\lfloor (t+1)/2 \rfloor}\right) = O\left(n^{1+2/t}\right)$$

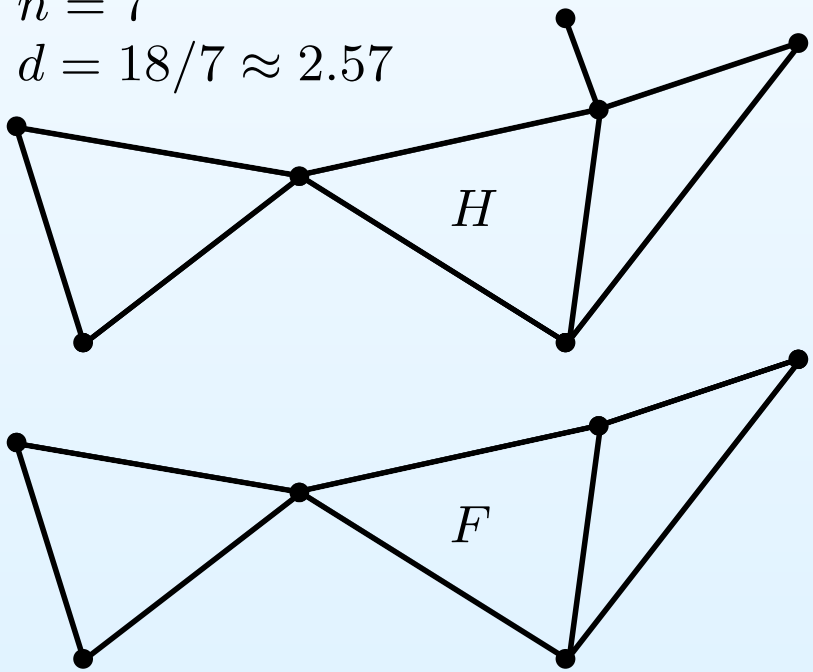
- Space (in bits) used by algorithm:

$$O\left(n^{1+2/t} \log n\right)$$

Proof of the Theorem: Removing Small-Degree Vertices

- Let d denote the average degree in H
- Note that $\sum_{v \in V} \deg_H(v) = 2m$ since each edge is counted twice
- Thus, $d = 2m/n$
- Construct subgraph F of H by removing vertices of degree $< d/2$
- By construction, F has minimum degree at least $d/2$
- Since F is a subgraph of H, $\gamma(F) \geq \gamma(H) \geq k$

$m = 9$
$n = 7$
$d = 18/7 \approx 2.57$



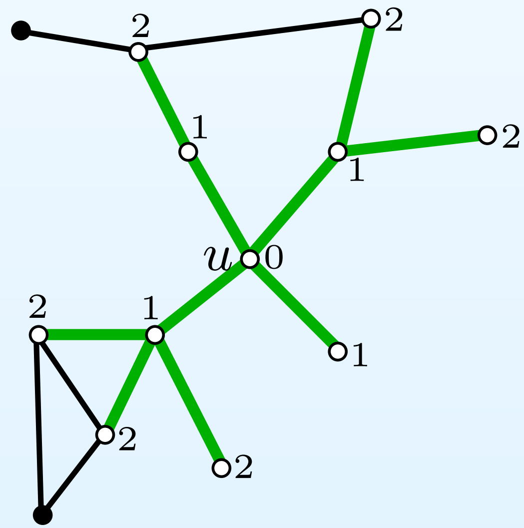
Proof of the Theorem: Tree Subgraph

- Let $\ell = \lfloor (k-1)/2 \rfloor$ and $u \in V$
- Let $B_u = (V_u, E_u)$ be subgraph of F visited by a BFS search from u up to distance ℓ:

$$V_u = \{v \in V \mid \delta_F(u, v) \leq \ell\}$$
$$E_u = \{(v, w) \in E(F) \mid \min\{\delta_F(u, v), \delta_F(u, w)\} \leq \ell - 1\}$$

- Example with white vertices in V_u and green edges in E_u:

$\ell = 2$:

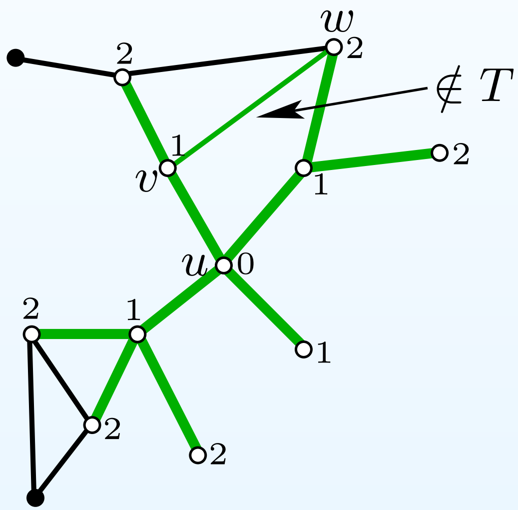


- We will show that B_u is a tree (exercise)

Proof that B_u is a Tree

- B_u is connected as it contains a BFS tree T
- Want to show $B_u = T$
- Assume for contradiction that B_u has an edge (v, w) not in T

$\ell = 2$:



- Recall:
  $E_u = \{(v, w) \in E(F) \mid \min\{\delta_F(u, v), \delta_F(u, w)\} \leq \ell - 1\}$
- Since $\ell = \lfloor (k-1)/2 \rfloor$, B_u has a cycle of length at most

$$\delta_T(u, v) + \underbrace{\delta_{B_u}(v, w)}_{=1} + \delta_T(w, u) \leq (\ell - 1) + 1 + \ell = 2\ell < k$$

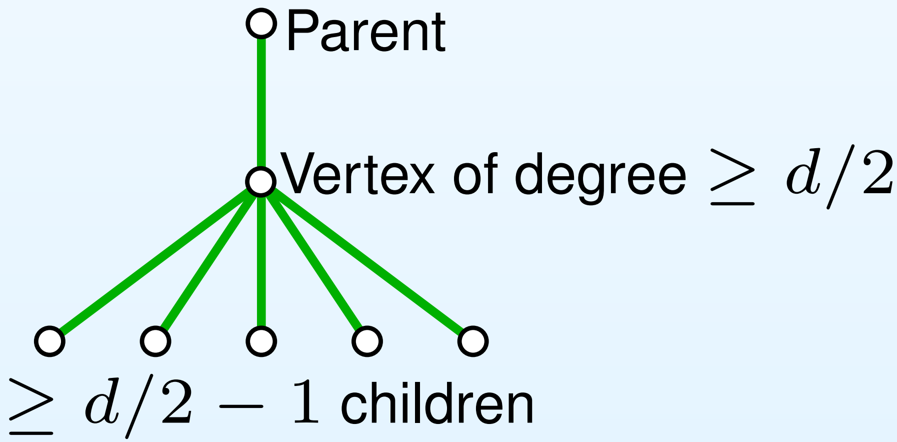
- But then $\gamma(B_u) < k$, contradicting that $\gamma(B_u) \geq \gamma(F) \geq k$

Proof of the Theorem: Showing that $m \le n + n^{1+1/\ell}$

- Have shown that $B_u = (V_u, E_u)$ is a BFS tree from u where

$$V_u = \{v \in V \mid \delta_F(u,v) \le \ell\}$$
$$E_u = \{(v,w) \in E(F) \mid \min\{\delta_F(u,v), \delta_F(u,w)\} \le \ell - 1\}$$

- F has min degree at least $d/2 \Rightarrow$ each non-leaf vertex in B_u has at least $d/2 - 1$ children



Proof of the Theorem: Showing that $m \leq n + n^{1+1/\ell}$

- Have shown that $B_u = (V_u, E_u)$ is a BFS tree from u where

$$V_u = \{v \in V \mid \delta_F(u,v) \leq \ell\}$$
$$E_u = \{(v,w) \in E(F) \mid \min\{\delta_F(u,v), \delta_F(u,w)\} \leq \ell - 1\}$$

- F has min degree at least $d/2 \Rightarrow$ each non-leaf vertex in B_u has at least $d/2 - 1$ children
- If n' is the number of vertices at distance ℓ from u in B_u,

$$n \geq n' = \left(\frac{d}{2} - 1\right)^\ell = \left(\frac{2m/n}{2} - 1\right)^\ell = \left(\frac{m}{n} - 1\right)^\ell$$

- Isolating m shows the theorem:

$$n \geq \left(\frac{m}{n} - 1\right)^\ell \Leftrightarrow n^{1/\ell} \geq \frac{m}{n} - 1 \Leftrightarrow n^{1+1/\ell} \geq m - n$$