# Detecting Feedback Vertex Sets of Size k in O*(2.7^k) Time

Jason Li

**With Jesper Nederlof (Utrecht Univ., Netherlands)**

**May 7, 2020**

# Introduction

Def: given a graph G, a feedback vertex set (FVS) is a set F of vertices s.t. G-F is a forest

Equivalently, F hits all cycles of G

# Introduction

Def: given a graph G, a feedback vertex set (FVS) is a set F of vertices s.t. G-F is a forest

Equivalently, F hits all cycles of G

Parameterized by k: given a graph G, find a FVS of size k in G or determine that none exist

# Introduction

Def: given a graph G, a feedback vertex set (FVS) is a set F of vertices s.t. G-F is a forest

Equivalently, F hits all cycles of G

Parameterized by k: given a graph G, find a FVS of size k in G or determine that none exist

Want time FPT in k: f(k)*poly(n)

Goal in FPT setting: minimize function f(k).

poly(n) factor does not matter

# Prior Work

Downey and Fellows '92: $f(k) = k^{O(k)}$

Becker et al. [BBG'00]: f(k)=4^k, randomized

Cygan et al. [CNP+'11]: f(k)=3^k, randomized

- actually runs in 3^tw time, given a tree decomposition of width tw

Kociumaka and Pilipczuk '14: f(k)=3.62^k deterministic

# Prior Work

Downey and Fellows '92: $f(k) = k^{O(k)}$

Becker et al. [BBG'00]: f(k)=4^k, randomized

Cygan et al. [CNP+'11]: f(k)=3^k, randomized

- actually runs in 3^tw time, given a tree decomposition of width tw

Kociumaka and Pilipczuk '14: f(k)=3.62^k deterministic

Our result: f(k)=2.7^k randomized.

Conceptual message: 3^k barrier can be broken

Combines techniques from [BBG'00] and [CNP+11].

# Prior Work

Downey and Fellows '92: $f(k) = k^{O(k)}$

Becker et al. [BBG'00]: f(k)=4^k, randomized

Cygan et al. [CNP+'11]: f(k)=3^k, randomized

- actually runs in 3^tw time, given a tree decomposition of width tw

Kociumaka and Pilipczuk '14: f(k)=3.62^k deterministic

Our result: f(k)=2.7^k randomized.

Conceptual message: 3^k barrier can be broken

Combines techniques from [BBG'00] and [CNP+11].

This talk: (3-ε)^k, or how to break 3^k.

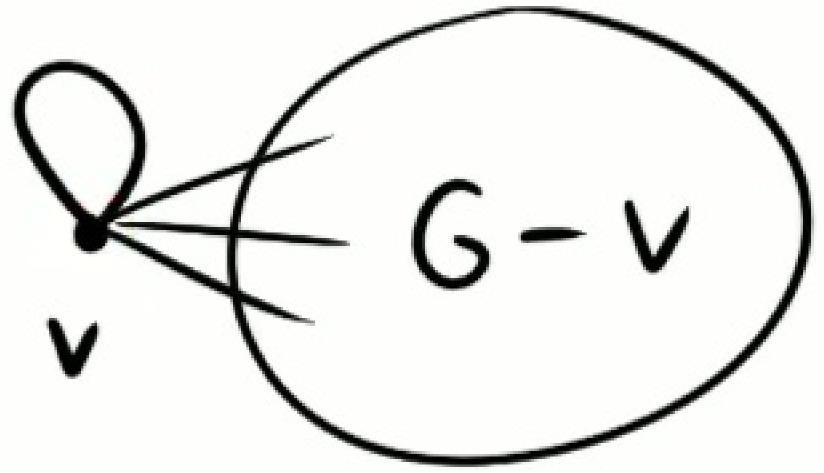
# Common Reductions

- Reduction Rule: (G, k) —> (G', k') where G has size-k FVS iff G' has size-k' FVS

# Common Reductions

- Reduction Rule: (G, k) —> (G', k') where G has size-k FVS iff G' has size-k' FVS

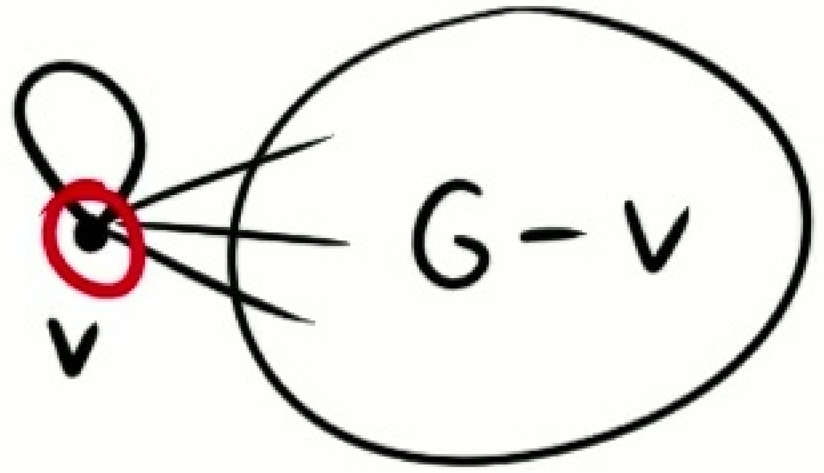
(1) Self-loop at v:



# Common Reductions

- Reduction Rule: (G, k) —> (G', k') where G has size-k FVS iff G' has size-k' FVS

(1) Self-loop at v:

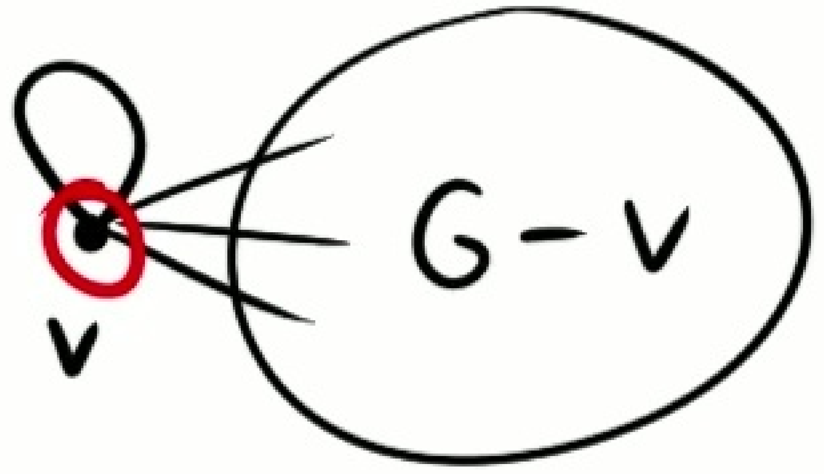



Must select v in FVS
(G, k) —> (G-v, k-1)

# Common Reductions

- Reduction Rule: (G, k) —> (G′, k′) where G has size-k FVS iff G′ has size-k′ FVS

(1) Self-loop at v:




Must select v in FVS

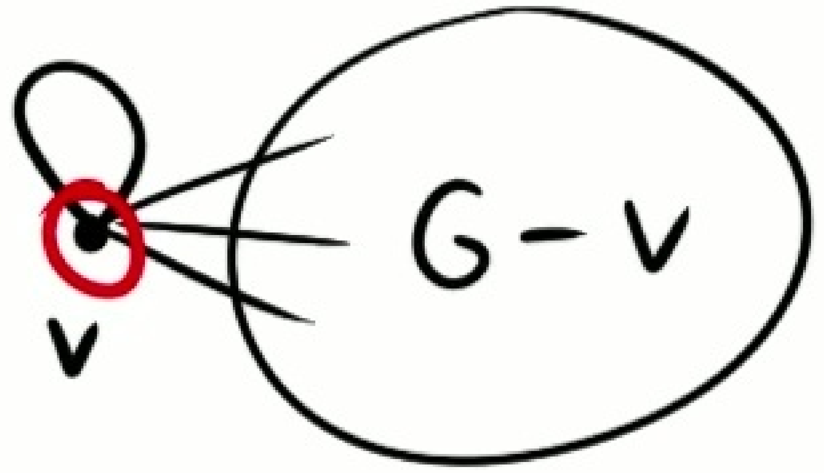
(G, k) —> (G-v, k-1)

Remove v and decrease k by 1

# Common Reductions

- Reduction Rule: (G, k) —> (G′, k′) where G has size-k FVS iff G′ has size-k′ FVS

(1) Self-loop at v:

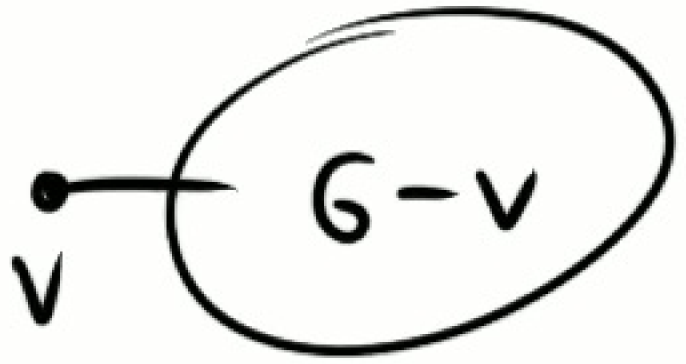



Must select v in FVS

(G, k) —> (G-v, k-1)

Remove v and decrease k by 1

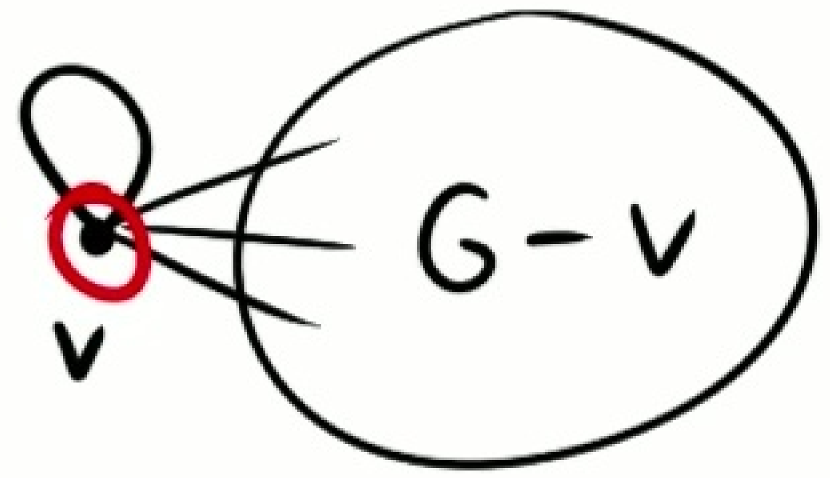
(2) Leaf vertex v (degree 1):



# Common Reductions

- Reduction Rule: (G, k) —> (G', k') where G has size-k FVS iff G' has size-k' FVS

(1) Self-loop at v:

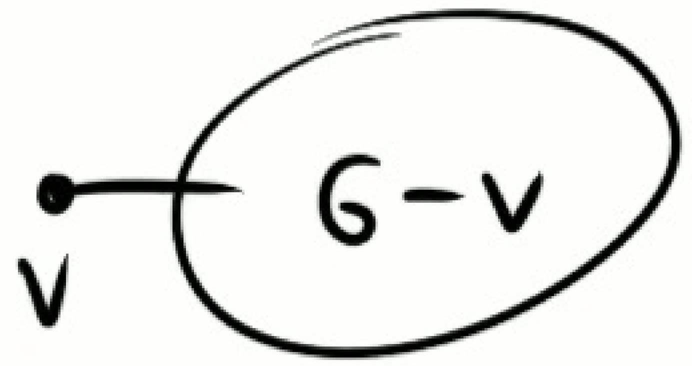



Must select v in FVS

(G, k) —> (G-v, k-1)

Remove v and decrease k by 1

(2) Leaf vertex v (degree 1):



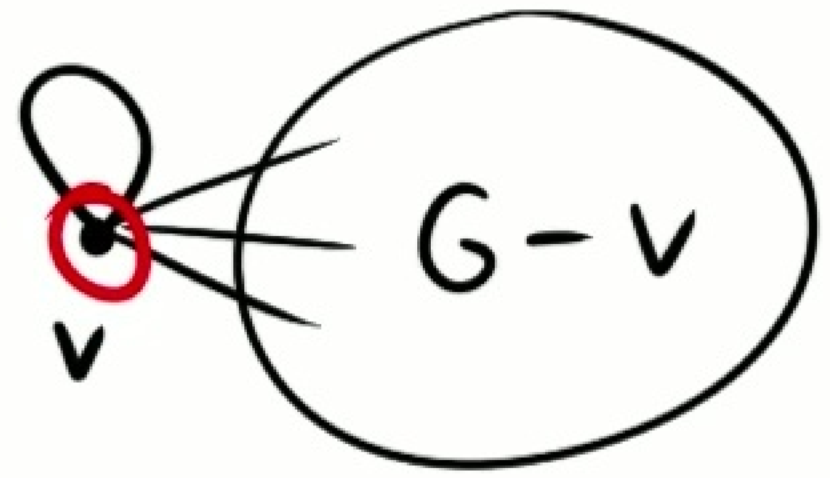

v doesn't belong in any cycle

Remove v: (G, k) —> (G-v, k)

# Common Reductions

- Reduction Rule: $(G, k) \longrightarrow (G', k')$ where G has size-k FVS iff G′ has size-k′ FVS

(1) Self-loop at v:

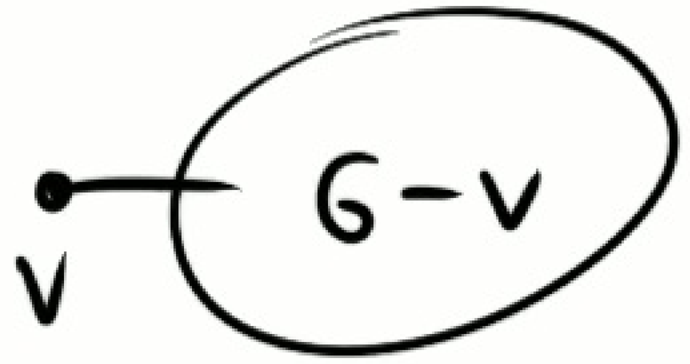



Must select v in FVS

$(G, k) \longrightarrow (G-v, k-1)$

Remove v and decrease k by 1

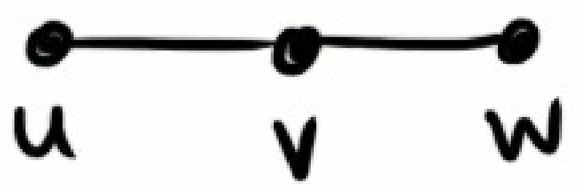
(2) Leaf vertex v (degree 1):




v doesn't belong in any cycle

Remove v: $(G, k) \longrightarrow (G-v, k)$

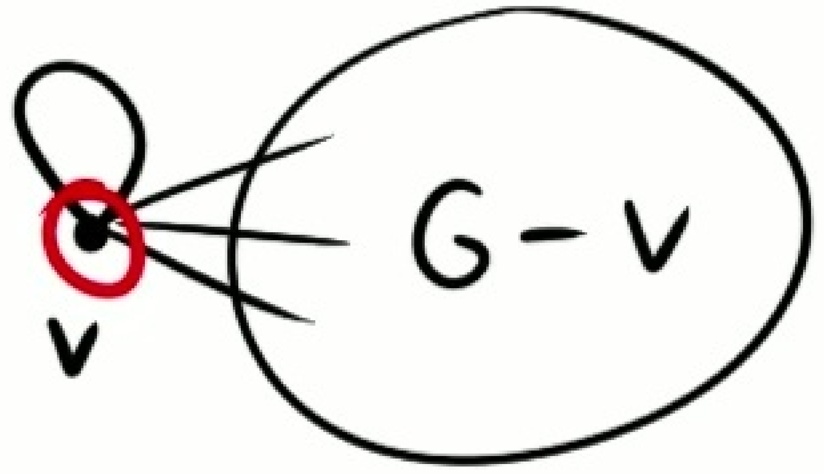
(3) Degree-2 vertex v:



# Common Reductions

- Reduction Rule: (G, k) —> (G', k') where G has size-k FVS iff G' has size-k' FVS

(1) Self-loop at v:

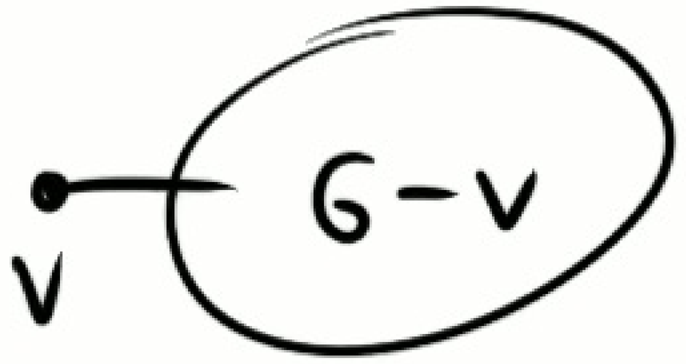



Must select v in FVS

(G, k) —> (G-v, k-1)

Remove v and decrease k by 1

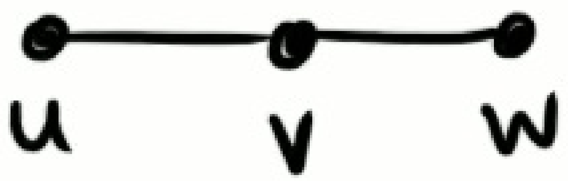
(2) Leaf vertex v (degree 1):




v doesn't belong in any cycle

Remove v: (G, k) —> (G-v, k)

(3) Degree-2 vertex v:



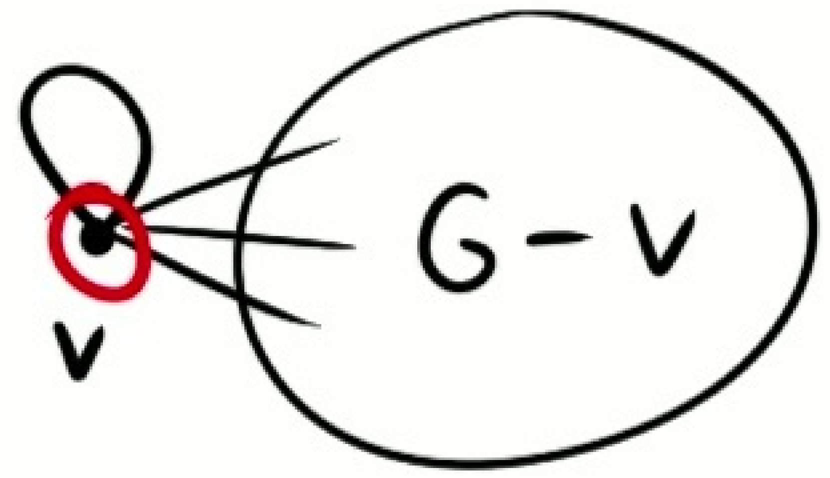

Any cycle containing v also contains u

If select v in FVS, then might as well select u instead

# Common Reductions

- Reduction Rule: (G, k) —> (G′, k′) where G has size-k FVS iff G′ has size-k′ FVS

(1) Self-loop at v:

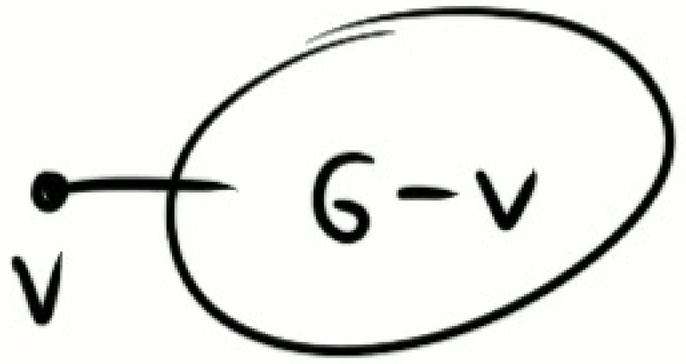



Must select v in FVS

(G, k) —> (G-v, k-1)

Remove v and decrease k by 1

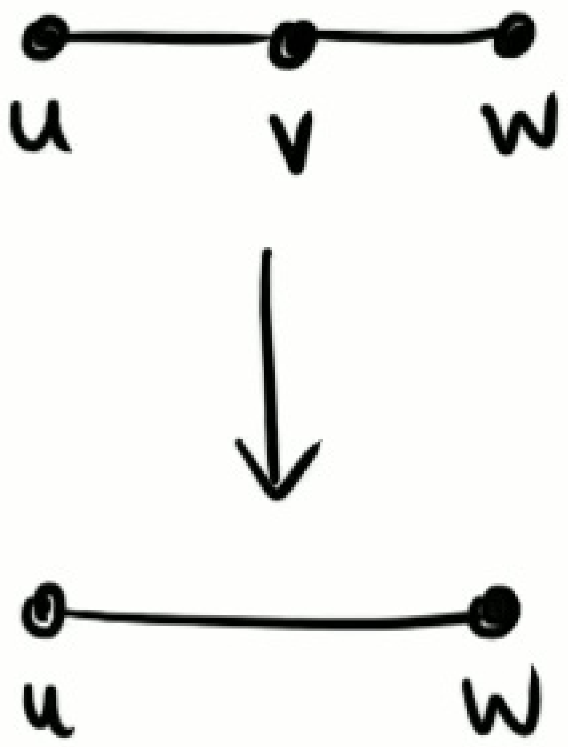
(2) Leaf vertex v (degree 1):




v doesn't belong in any cycle

Remove v: (G, k) —> (G-v, k)

(3) Degree-2 vertex v:




Any cycle containing v also contains u

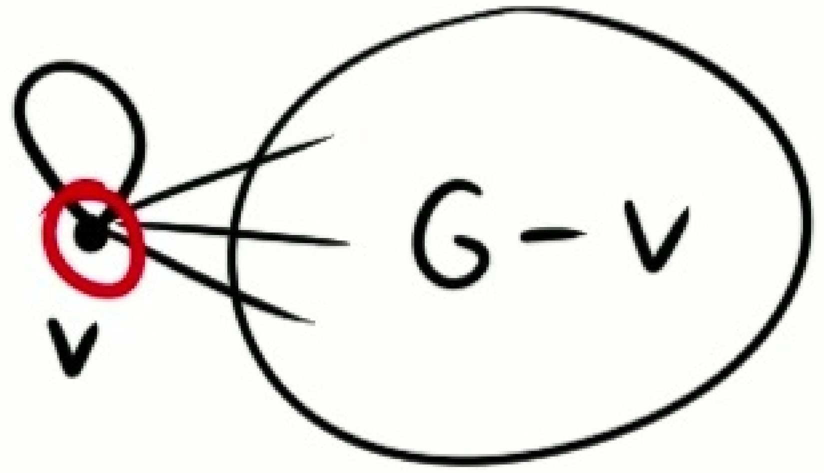
If select v in FVS, then might as well select u instead

Delete v and add edge (u,w)

# Common Reductions

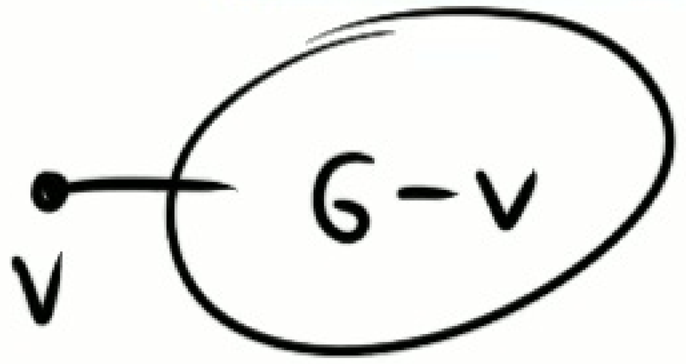
- Reduction Rule: (G, k) —> (G′, k′) where G has size-k FVS iff G′ has size-k′ FVS

(1) Self-loop at v:




**When (1),(2),(3) no longer apply:**
- no self-loops
- minimum degree 3

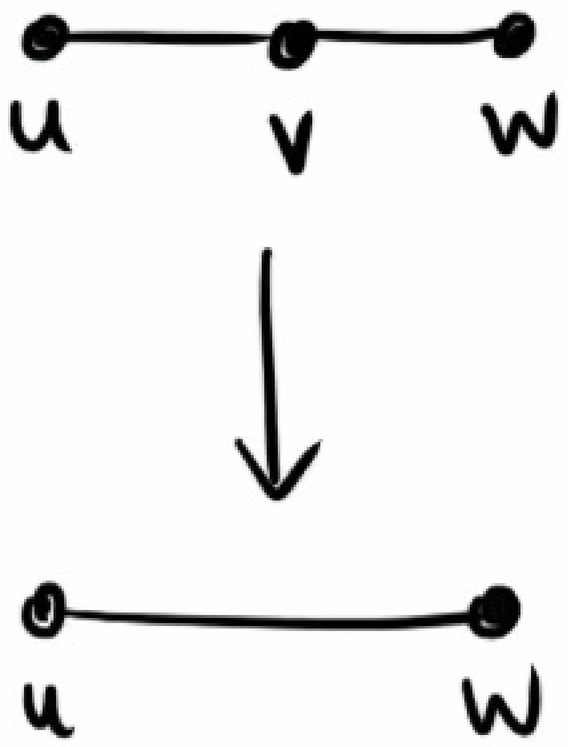
(2) Leaf vertex v (degree 1):




v doesn't belong in any cycle

Remove v: (G, k) —> (G-v, k)

(3) Degree-2 vertex v:




Any cycle containing v also contains u

If select v in FVS, then might as well select u instead

Delete v and add edge (u,w)

# Becker et al. Reduction

- Reduction Rule: (G, k) —> (G', k') where G has size-k FVS iff G' has size-k' FVS

# Becker et al. Reduction

- Reduction Rule: (G, k) —> (G′, k′) where G has size-k FVS iff G′ has size-k′ FVS
- Probabilistic Reduction: (G, k) —> (G′, k′) where with probability p, G has size-k FVS iff G′ has size-k′ FVS

# Becker et al. Reduction

- Reduction Rule: (G, k) —> (G′, k′) where G has size-k FVS iff G′ has size-k′ FVS
- Probabilistic Reduction: (G, k) —> (G′, k′) where with probability p, G has size-k FVS iff G′ has size-k′ FVS

Becker et al. [BBG’00]: sample v proportional to deg(v)

# Becker et al. Reduction

- Reduction Rule: (G, k) —> (G′, k′) where G has size-k FVS iff G′ has size-k′ FVS
- Probabilistic Reduction: (G, k) —> (G′, k′) where with probability p, G has size-k FVS iff G′ has size-k′ FVS

Becker et al. [BBG’00]: sample v proportional to deg(v)

Add v to FVS: (G, k) -> (G-v, k-1)

# Becker et al. Reduction

- Reduction Rule: (G, k) —> (G', k') where G has size-k FVS iff G' has size-k' FVS
- Probabilistic Reduction: (G, k) —> (G', k') where with probability p, G has size-k FVS iff G' has size-k' FVS

Becker et al. [BBG'00]: sample v proportional to deg(v)

Add v to FVS: (G, k) -> (G-v, k-1)

Theorem: if G has minimum degree ≥3 and a FVS of size k, then with probability ≥1/4, v is in the FVS

# Becker et al. Reduction

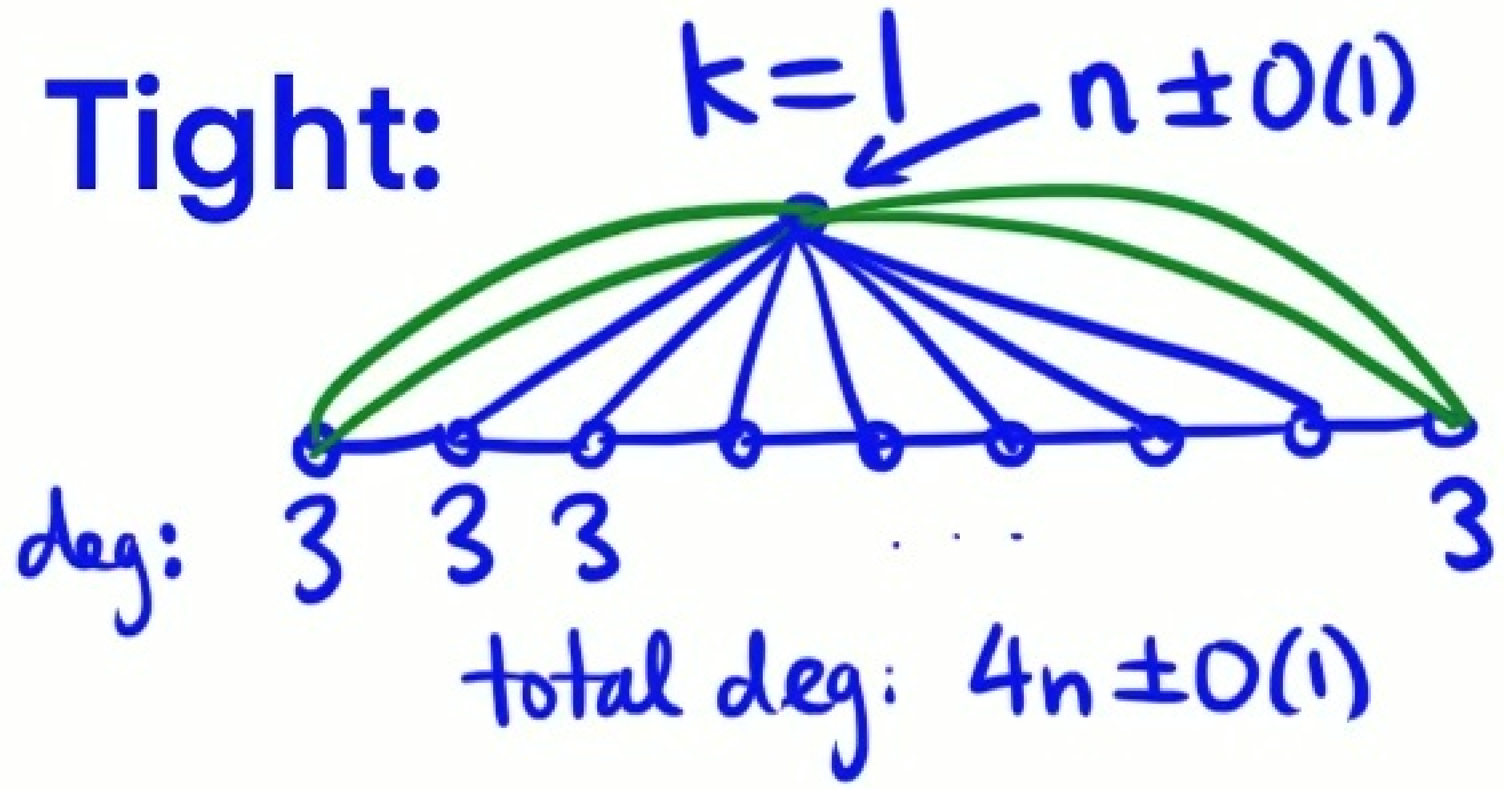
- Reduction Rule: (G, k) —> (G', k') where G has size-k FVS iff G' has size-k' FVS
- Probabilistic Reduction: (G, k) —> (G', k') where with probability p, G has size-k FVS iff G' has size-k' FVS

Becker et al. [BBG'00]: sample v proportional to deg(v)

Add v to FVS: (G, k) -> (G-v, k-1)

**Theorem: if G has minimum degree ≥3 and a FVS of size k, then with probability ≥1/4, v is in the FVS**

Tight:



# Becker et al. Reduction

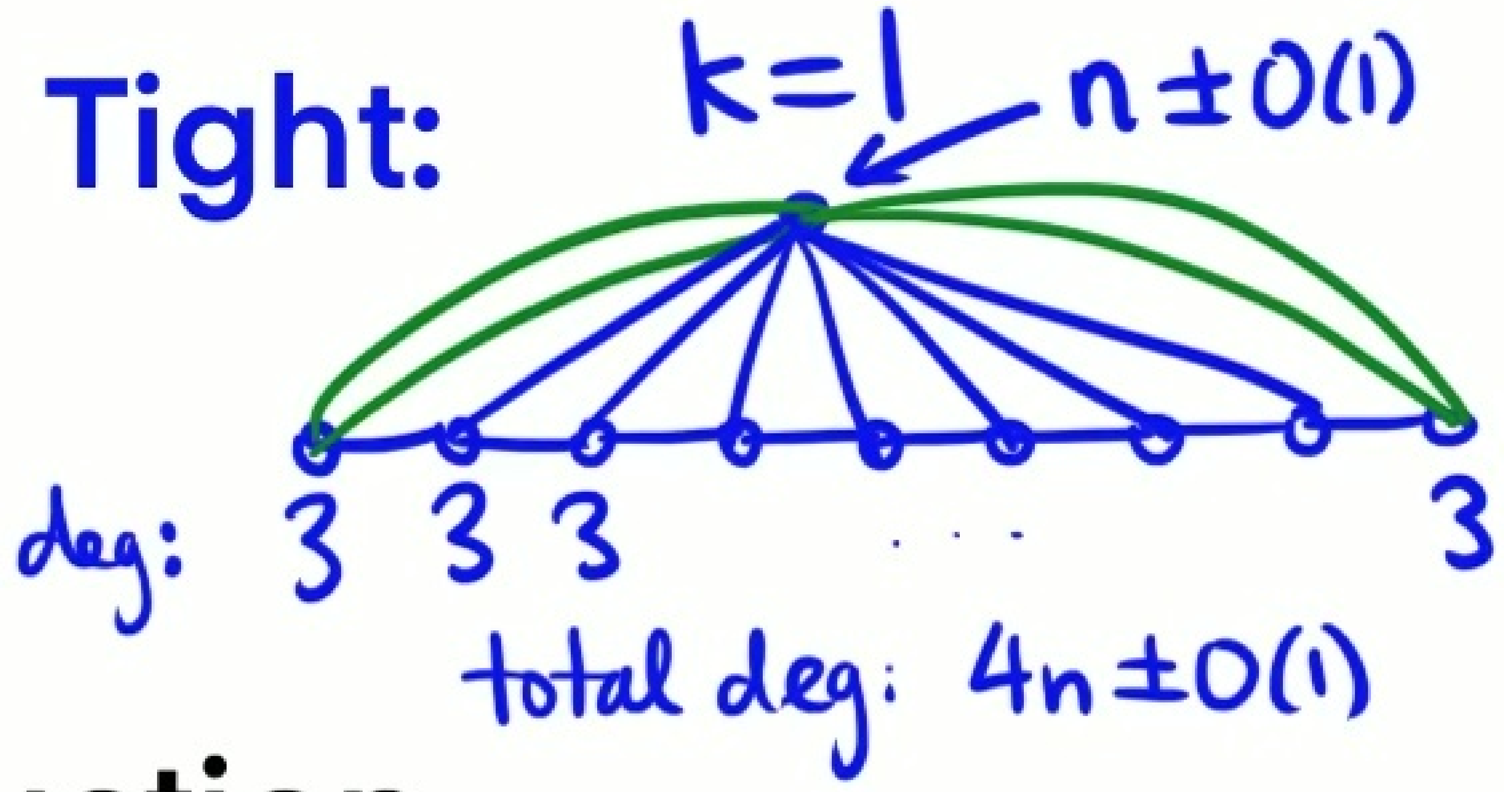
- Reduction Rule: (G, k) —> (G′, k′) where G has size-k FVS iff G′ has size-k′ FVS
- Probabilistic Reduction: (G, k) —> (G′, k′) where with probability p, G has size-k FVS iff G′ has size-k′ FVS

Becker et al. [BBG'00]: sample v proportional to deg(v)

Add v to FVS: (G, k) -> (G-v, k-1)

Theorem: if G has minimum degree ≥3 and a FVS of size k, then with probability ≥1/4, v is in the FVS

Tight:




Prob. 1/4 to decrease k by 1 and preserve reduction

=> prob. 1/4^k to go all the way. Repeat 4^k times: O*(4^k) algo.

# Our Approach

Dense case: $m >> O(k)$:

Sparse case: $m \leq O(k)$:

# Our Approach

Dense case: m >> O(k):

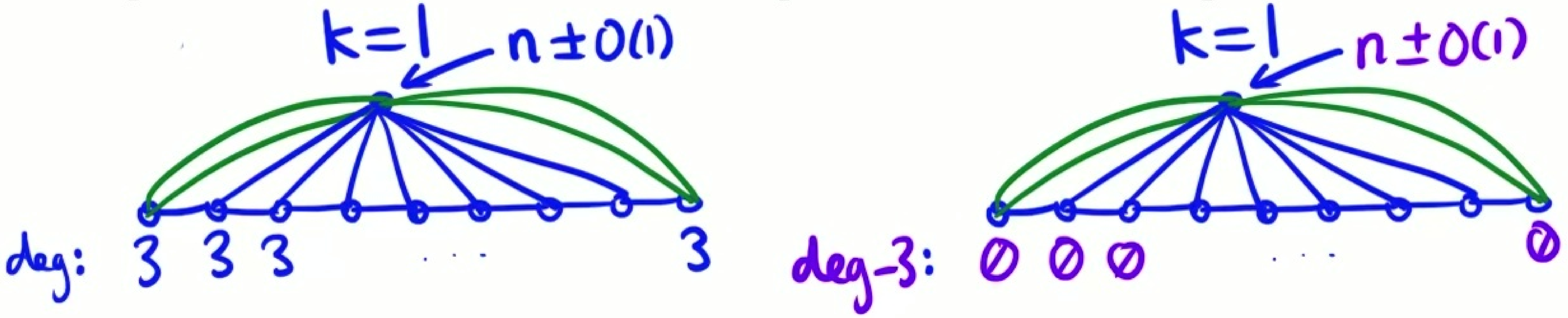
- Modify [BBG'00]: sample v with prob. ~ deg(v)-3

Sparse case: m ≤ O(k):

# Our Approach

Dense case: m >> O(k):

- Modify [BBG'00]: sample v with prob. ~ deg(v)-3



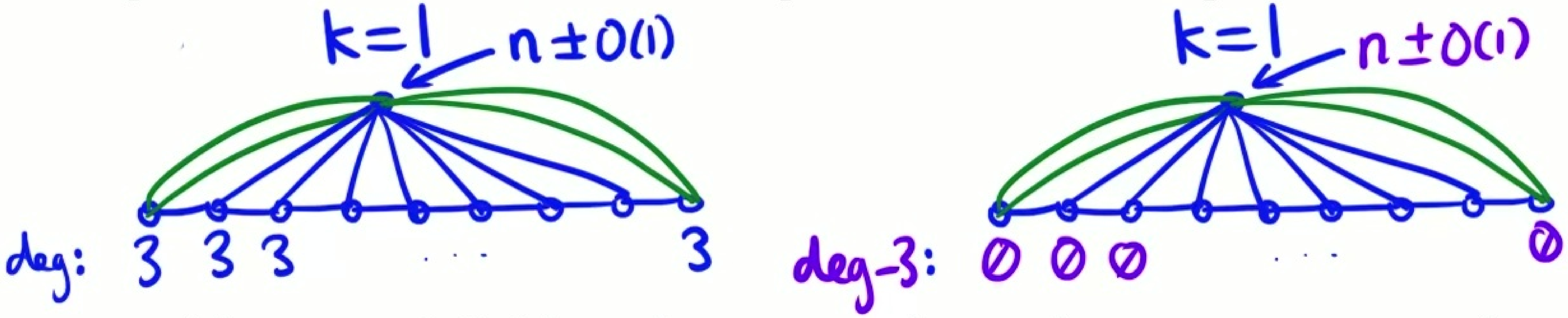

Success w.p. 1

Sparse case: m ≤ O(k):

# Our Approach

Dense case: m >> O(k):

- Modify [BBG'00]: sample v with prob. ~ deg(v)-3




Success w.p. 1

Lemma: if $m \geq 100k$, then reduction succeeds w.p. 1/2.99

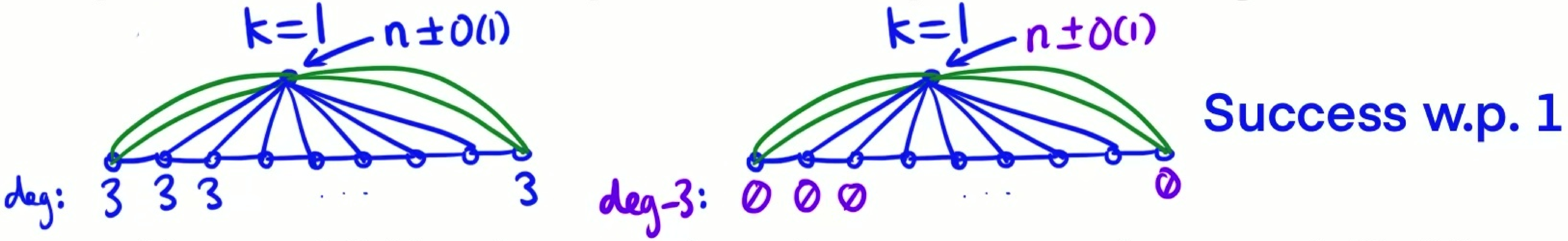
Repeat k times => 2.99^k algo

Sparse case: $m \leq O(k)$:

# Our Approach

Dense case: m >> O(k):

- Modify [BBG'00]: sample v with prob. ~ deg(v)-3




Lemma: if m ≥ 100k, then reduction succeeds w.p. 1/2.99

Repeat k times => 2.99^k algo

Sparse case: m ≤ O(k):

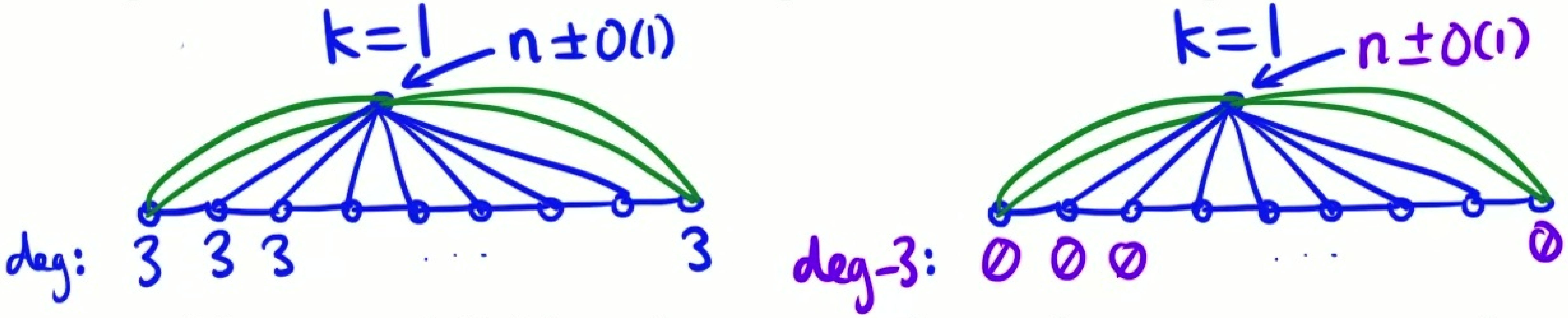
Lemma: if m ≤ 100k and exists FVS size k, then

G has treewidth (1-Ω(1))k

# Our Approach

Dense case: m >> O(k):

- Modify [BBG'00]: sample v with prob. ~ deg(v)-3




Success w.p. 1

Lemma: if m ≥ 100k, then reduction succeeds w.p. 1/2.99

Repeat k times => 2.99^k algo

Sparse case: m ≤ O(k):

Lemma: if m ≤ 100k and exists FVS size k, then

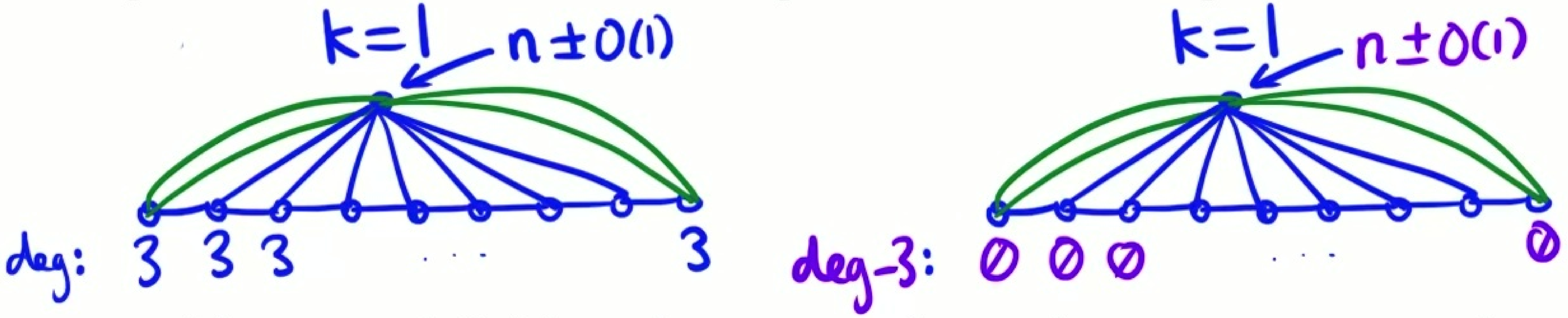
G has treewidth (1-Ω(1))k

Can find such a tree decomposition. Run 3^tw algo [CNP+11]

# Our Approach

## Dense case: m >> O(k):

- Modify [BBG'00]: sample v with prob. ~ deg(v)-3




Success w.p. 1

Lemma: if m ≥ 100k, then reduction succeeds w.p. 1/2.99

Repeat k times => 2.99^k algo

## Sparse case: m ≤ O(k):

Lemma: if m ≤ 100k and exists FVS size k, then

G has treewidth (1-Ω(1))k ⇒ time $3^{(1-\Omega(1))k} = (3-\varepsilon)^k$.

Can find such a tree decomposition. Run 3^tw algo [CNP+11]

# Dense case: m >> O(k)

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4

Proof:

# Dense case: m >> O(k)

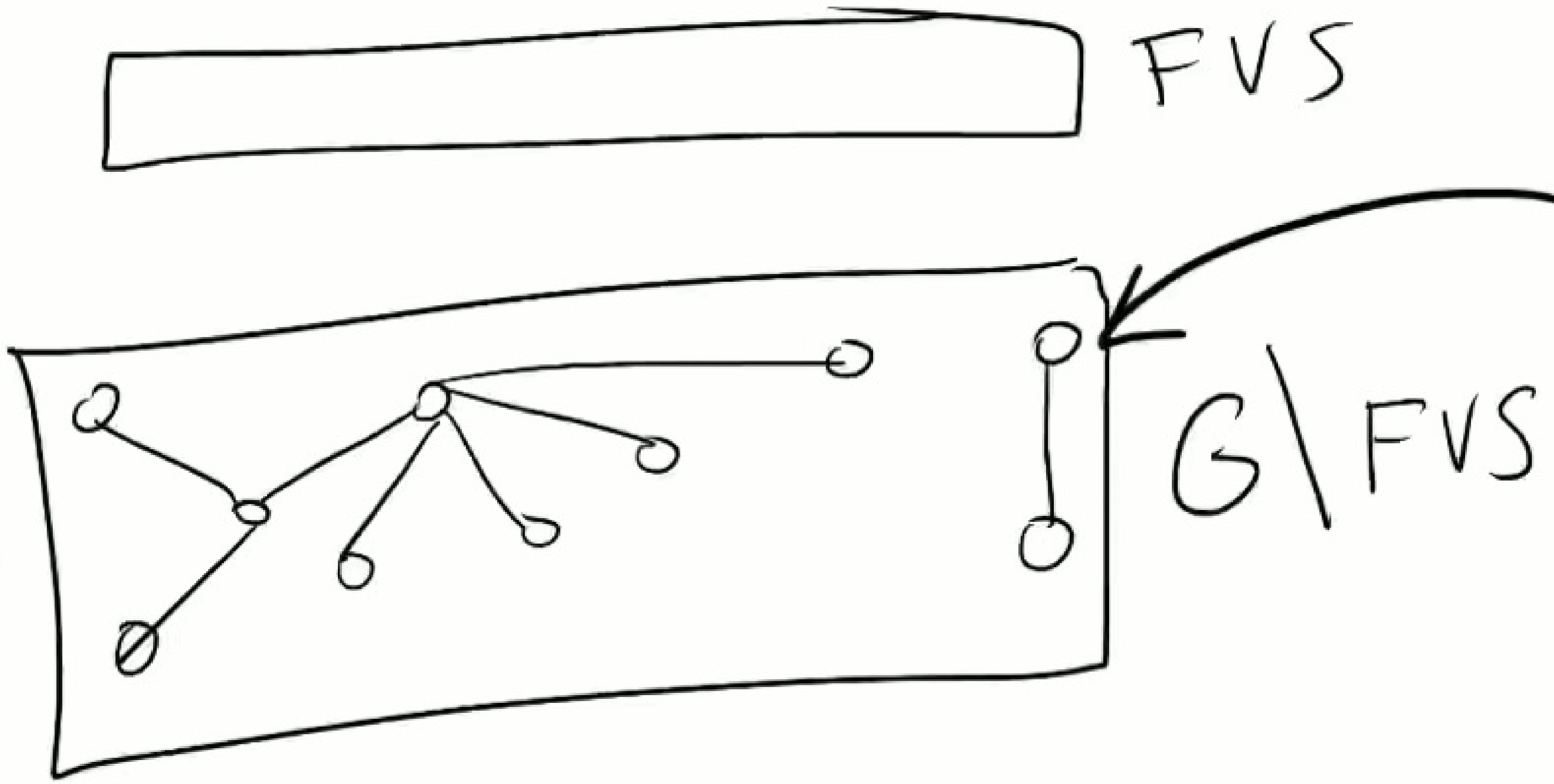
Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:



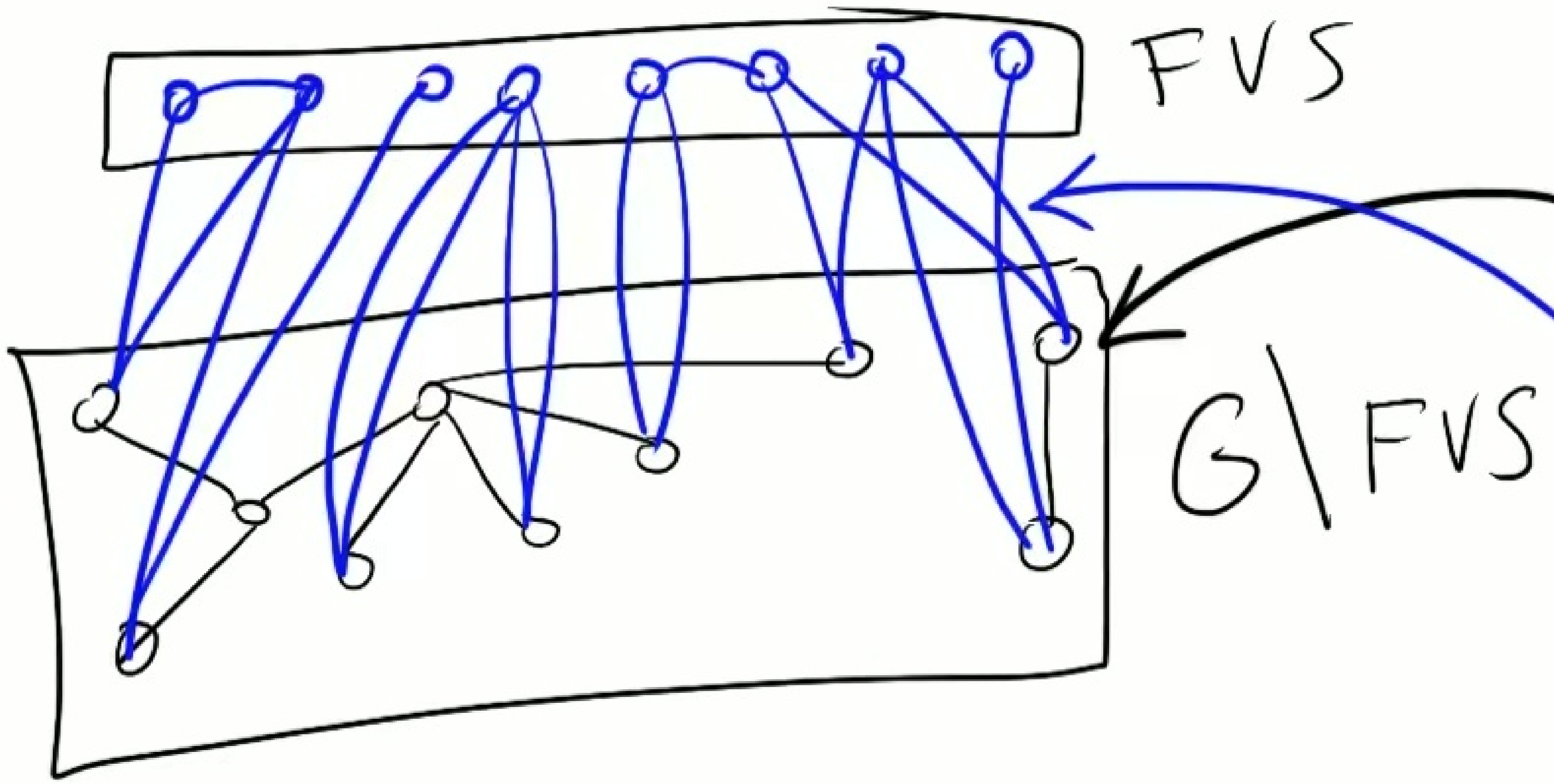

(min deg 3)

Acyclic : average degree <= 2

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:




(min deg 3)

Acyclic : average degree <= 2

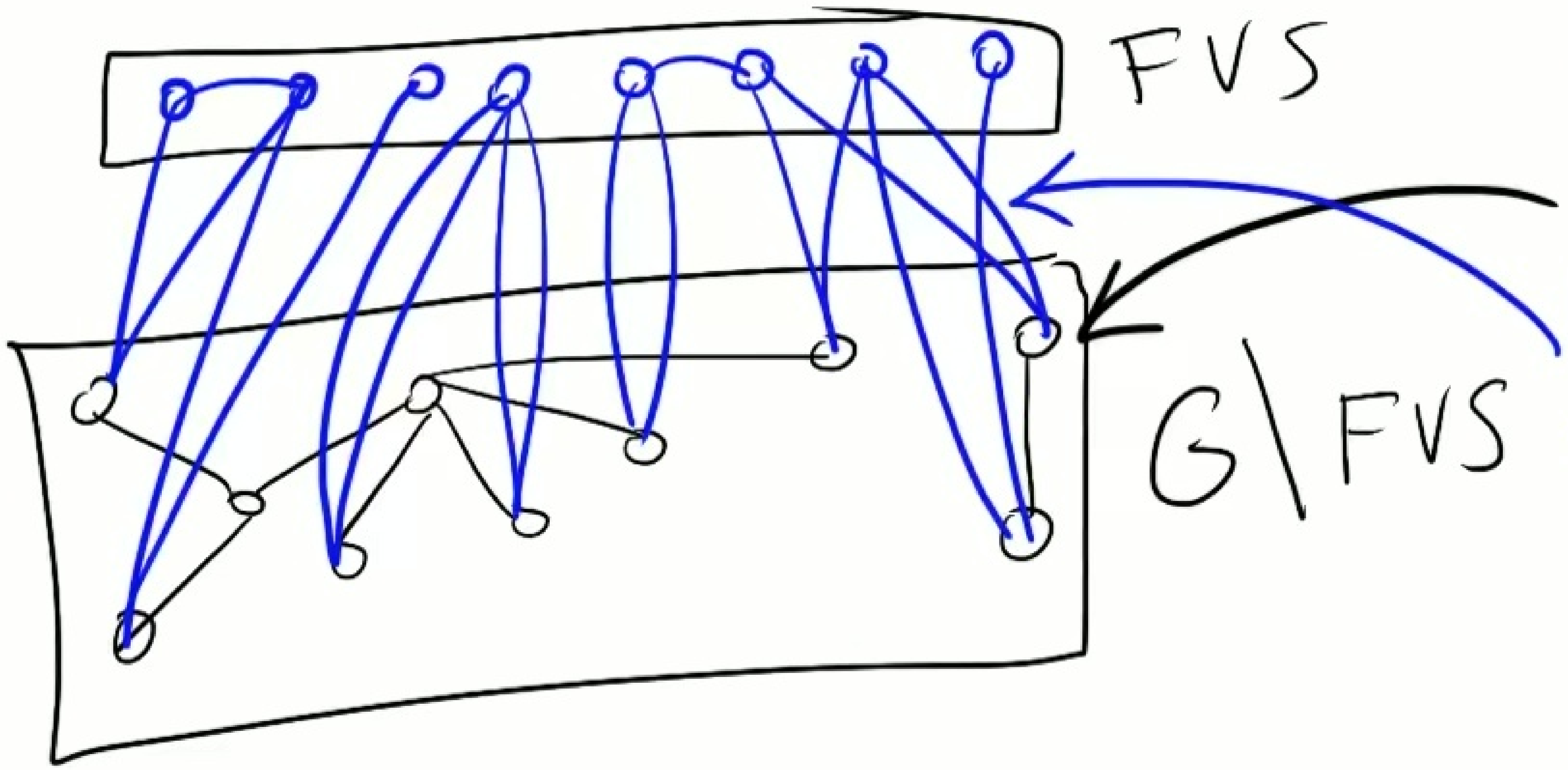
Must add >= 1 edge per vertex in G\FVS on average

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:

(min deg 3)




Acyclic : average degree <= 2

Must add >= 1 edge per vertex in G\FVS on average

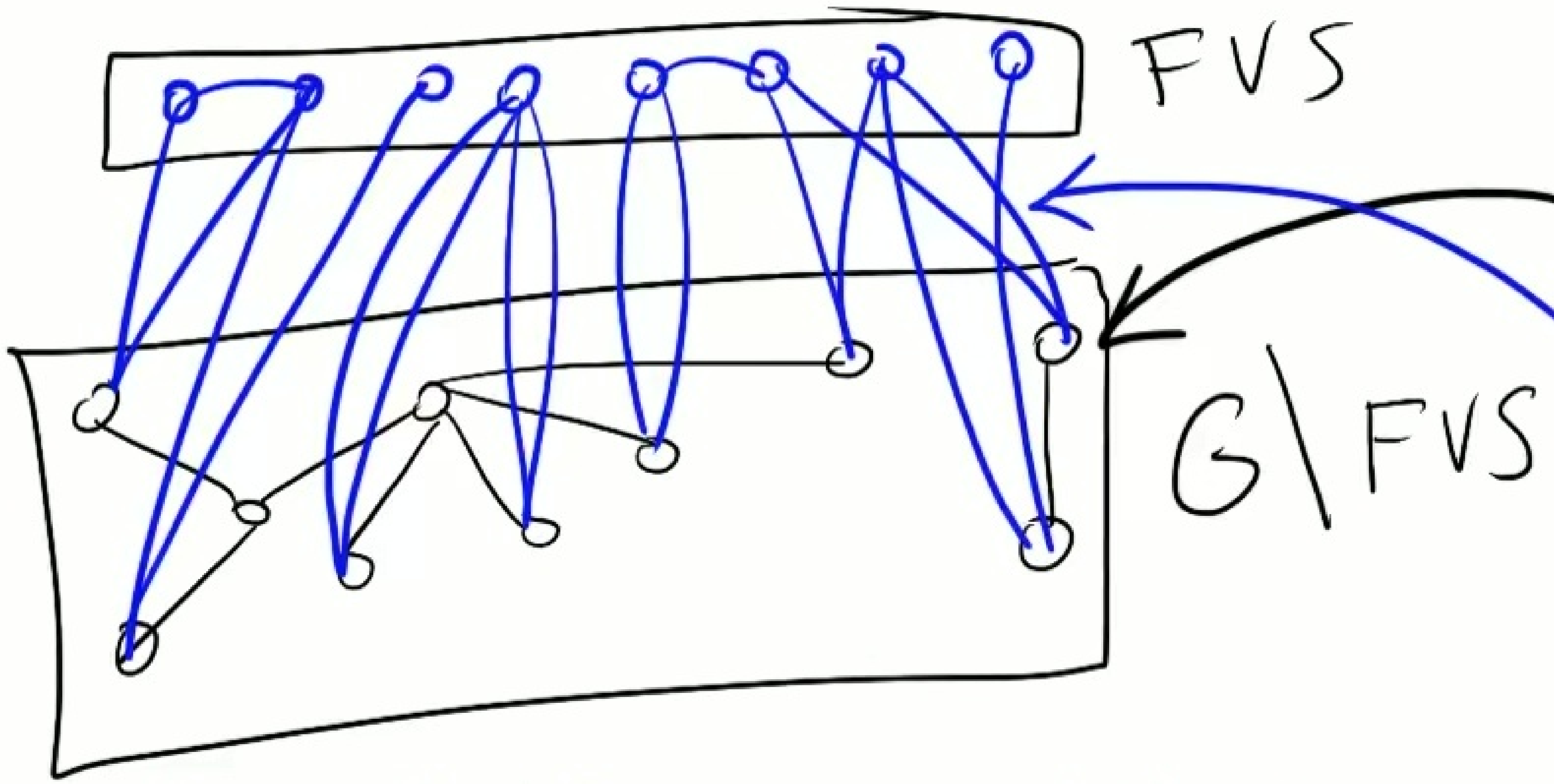
Acyclic: <= 1 edge in G\FVS per vertex in G\FVS on average

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:

(min deg 3)




Acyclic : average degree <= 2

Must add >= 1 edge per vertex in G\FVS on average

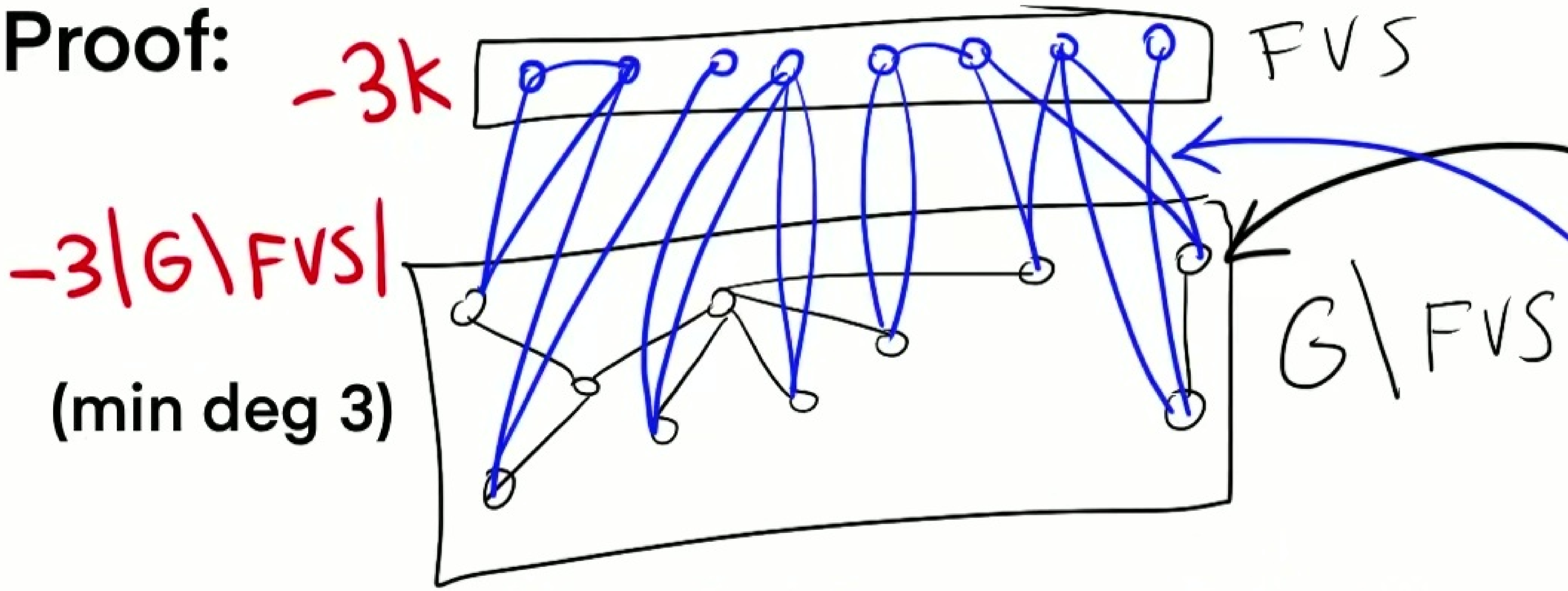
Acyclic: <= 1 edge in G\FVS per vertex in G\FVS on average

Sample blue w.p. >=1/2, then sample v in FVS w.p. 1/2 => 1/4 overall

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:




(min deg 3)

Acyclic : average degree <= 2

Must add >= 1 edge per vertex in G\FVS on average

Acyclic: <= 1 edge in G\FVS per vertex in G\FVS on average

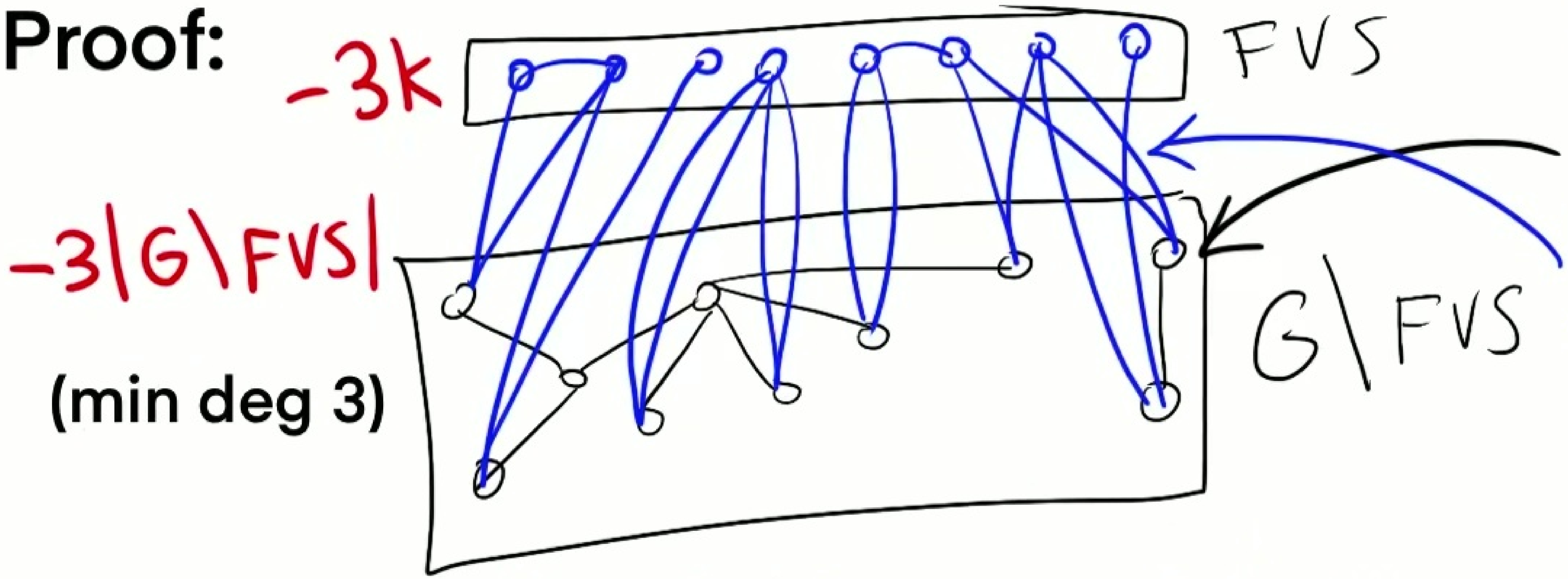
Sample blue w.p. >=1/2, then sample v in FVS w.p. 1/2 => 1/4 overall

Sample v with prob. ~ deg(v) - 3 ?

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:




(min deg 3)

Acyclic : average degree <= 2

Must add >= 1 edge per vertex in G\FVS on average

Acyclic: <= 1 edge in G\FVS per vertex in G\FVS on average

Sample blue w.p. >=1/2, then sample v in FVS w.p. 1/2 => 1/4 overall

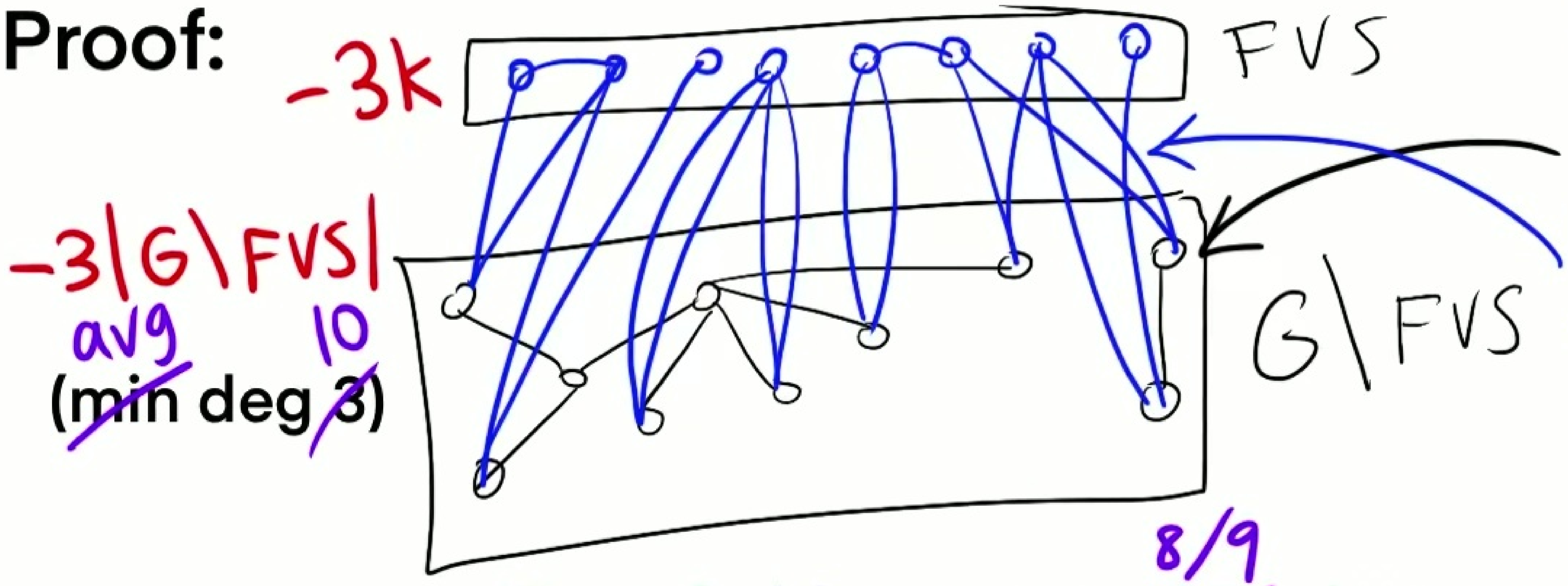
Sample v with prob. ~ deg(v) - 3 ?

- Suppose m >= 10n (high average degree)...

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:




Acyclic : average degree <= 2

Must add >= ~~1~~ 8 edge per vertex in G\FVS on average

Acyclic: <= 1 edge in G\FVS per vertex in G\FVS on average

Sample blue w.p. >= ~~1/2~~ 8/9, then sample v in FVS w.p. 1/2 => 1/4 overall

Sample v with prob. ~ deg(v) ~~- 3~~ ?

- Suppose m >= 10n (high average degree)... Success prob >= 1/2.99

# Dense case: m >> O(k)

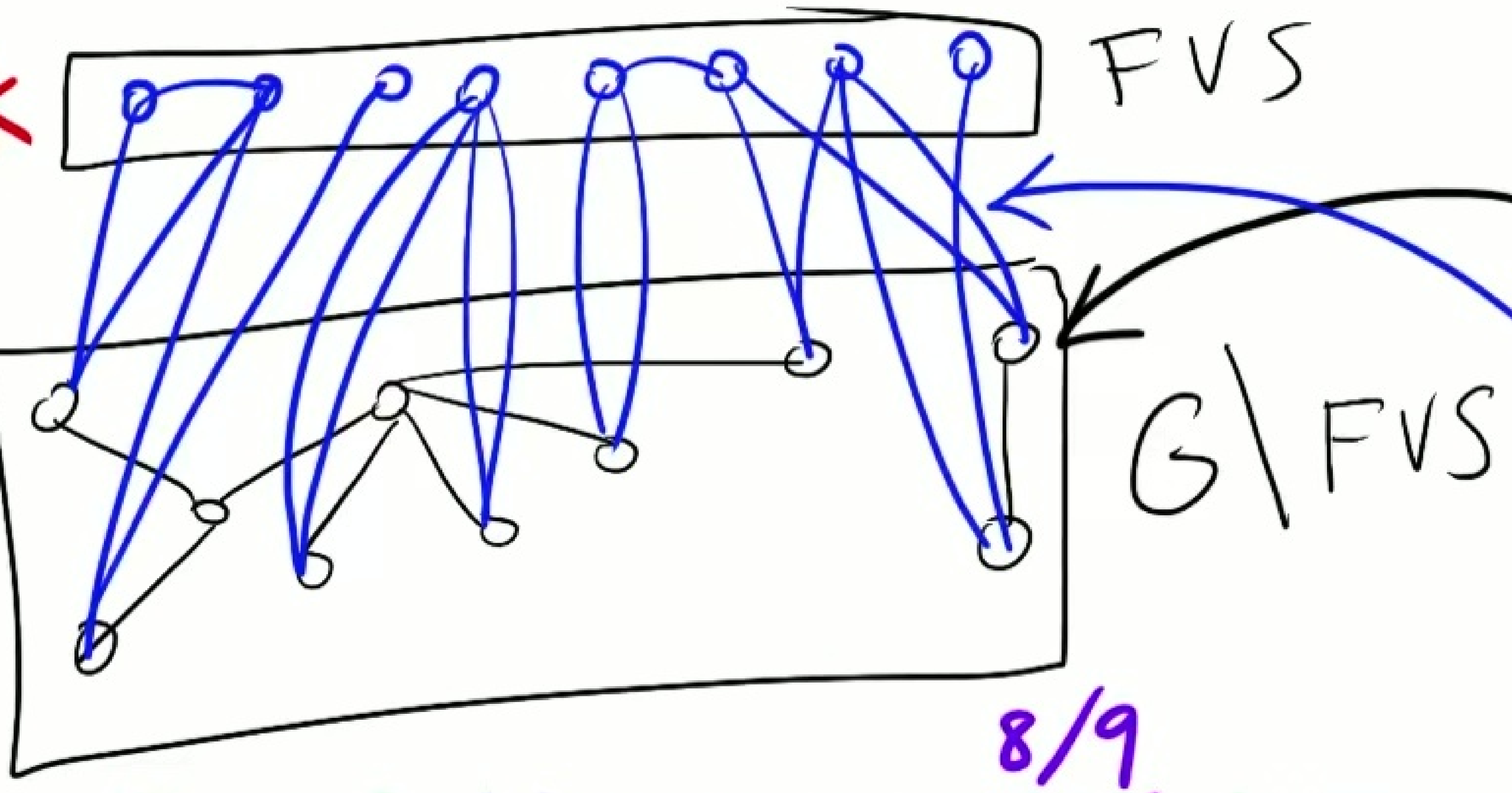
Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:

−3k

−3|G\FVS|

(~~min~~ avg deg ~~3~~ 10)




Acyclic: average degree <= 2

Must add >= ~~1~~ 8 edge per vertex in G\FVS on average

Acyclic: <= 1 edge in G\FVS per vertex in G\FVS on average

Sample blue w.p. >= ~~1/2~~ 8/9, then sample v in FVS w.p. 1/2 => 1/4 overall

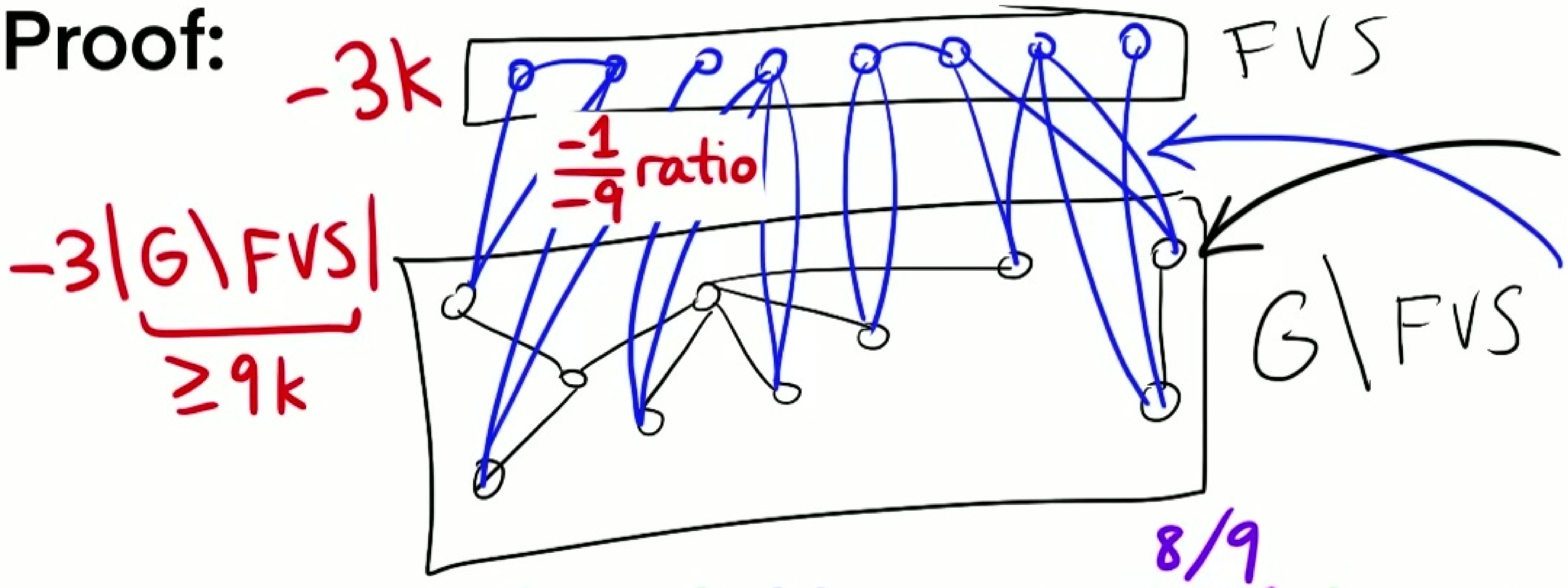
Sample v with prob. ~ deg(v) ~~- 3~~ ?

- Suppose m >= 10n (high average degree)... Success prob >= 1/2.99
- Suppose n >= 10k (many vertices)...

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4 <=> sample random edge, then random endpoint

Proof:




Acyclic : average degree <= 2

Must add >= ~~1~~ 8 edge per vertex in G\FVS on average

Acyclic: <= 1 edge in G\FVS per vertex in G\FVS on average

Sample blue w.p. >= ~~1/2~~ 8/9, then sample v in FVS w.p. 1/2 => 1/4 overall

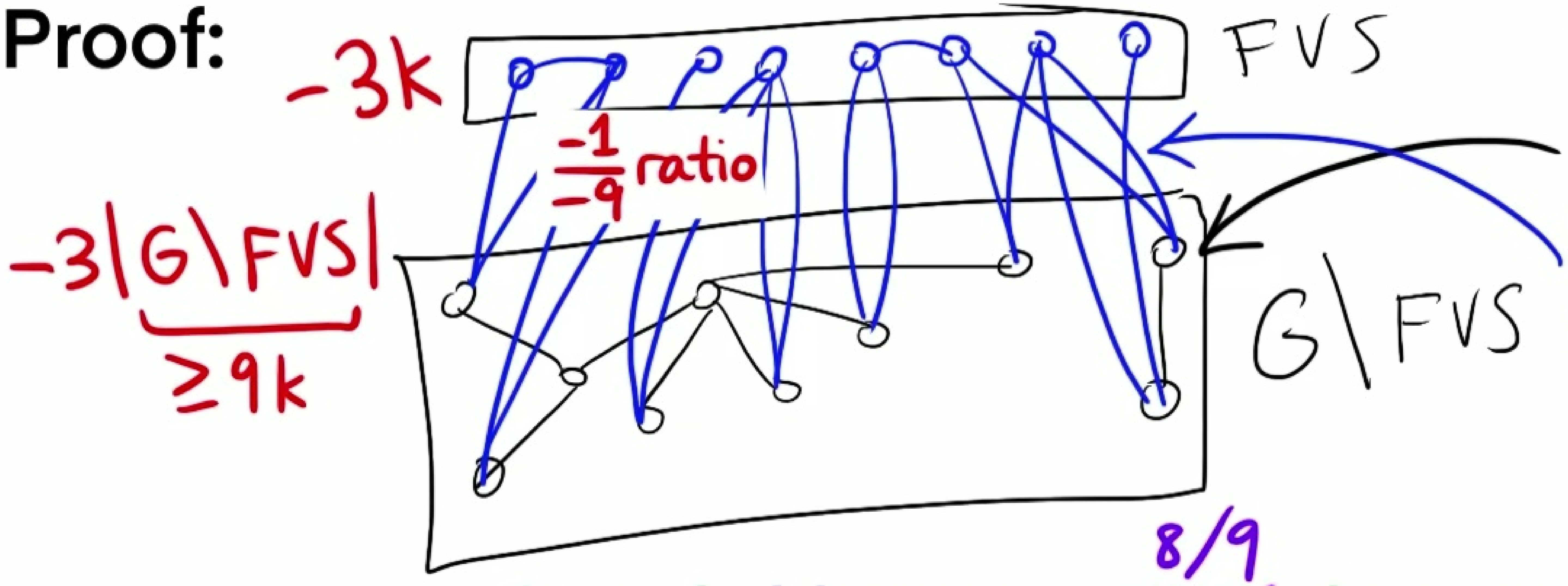
Sample v with prob. ~ deg(v) ~~- 3~~ ?

- Suppose m >= 10n (high average degree)... Success prob >= 1/2.99
- Suppose n >= 10k (many vertices)... Success prob >= 1/2.99

# Dense case: m >> O(k)

Lemma [BBG'00]: if sample v with prob. ~ deg(v), then sample v in FVS with prob. >= 1/4  <=> sample random edge, then random endpoint

Proof:




Acyclic : average degree <= 2

Must add >= ~~1~~ 8 edge per vertex in G\FVS on average

Acyclic: <= 1 edge in G\FVS per vertex in G\FVS on average

Sample blue w.p. >= ~~1/2~~ 8/9, then sample v in FVS w.p. 1/2 => 1/4 overall

Sample v with prob. ~ deg(v) ~~- 3~~ ?

- Suppose m >= 10n (high average degree)... Success prob >= 1/2.99
- Suppose n >= 10k (many vertices)... Success prob >= 1/2.99

If m >= 100k, then either m >= 10n or n >= 10k, so success prob >= 1/2.99

# Iterative Compression

Original problem: given graph G, find FVS size k, or determine none exist.

# Iterative Compression

Original problem: given graph G, find FVS size k, or determine none exist.

Iterative Compression reduces to: given graph G and FVS size (k+1), find FVS size k or determine none exist.

# Iterative Compression

Original problem: given graph G, find FVS size k, or determine none exist.

Iterative Compression reduces to: given graph G and FVS size (k+1), find FVS size k or determine none exist.

- Get size (k+1) FVS as input for free

# Iterative Compression

Original problem: given graph G, find FVS size k, or determine none exist.

Iterative Compression reduces to: given graph G and FVS size (k+1), find FVS size k or determine none exist.

- Get size (k+1) FVS as input for free

Idea: if $S_i$ is FVS of size k to $G[\{v_1,\ldots,v_i\}]$, then

$S_i \cup \{v_{i+1}\}$ is FVS of size k+1 for $G[\{v_1,\ldots,v_{i+1}\}]$

# Iterative Compression

Original problem: given graph G, find FVS size k, or determine none exist.

Iterative Compression reduces to: given graph G and FVS size (k+1), find FVS size k or determine none exist.

- Get size (k+1) FVS as input for free

Idea: if $S_i$ is FVS of size k to $G[\{v_1,\ldots,v_i\}]$, then

$S_i \cup \{v_{i+1}\}$ is FVS of size k+1 for $G[\{v_1,\ldots,v_{i+1}\}]$

Solve on $(G[\{v_1,\ldots,v_{i+1}\}], S_i \cup \{v_{i+1}\})$ to get

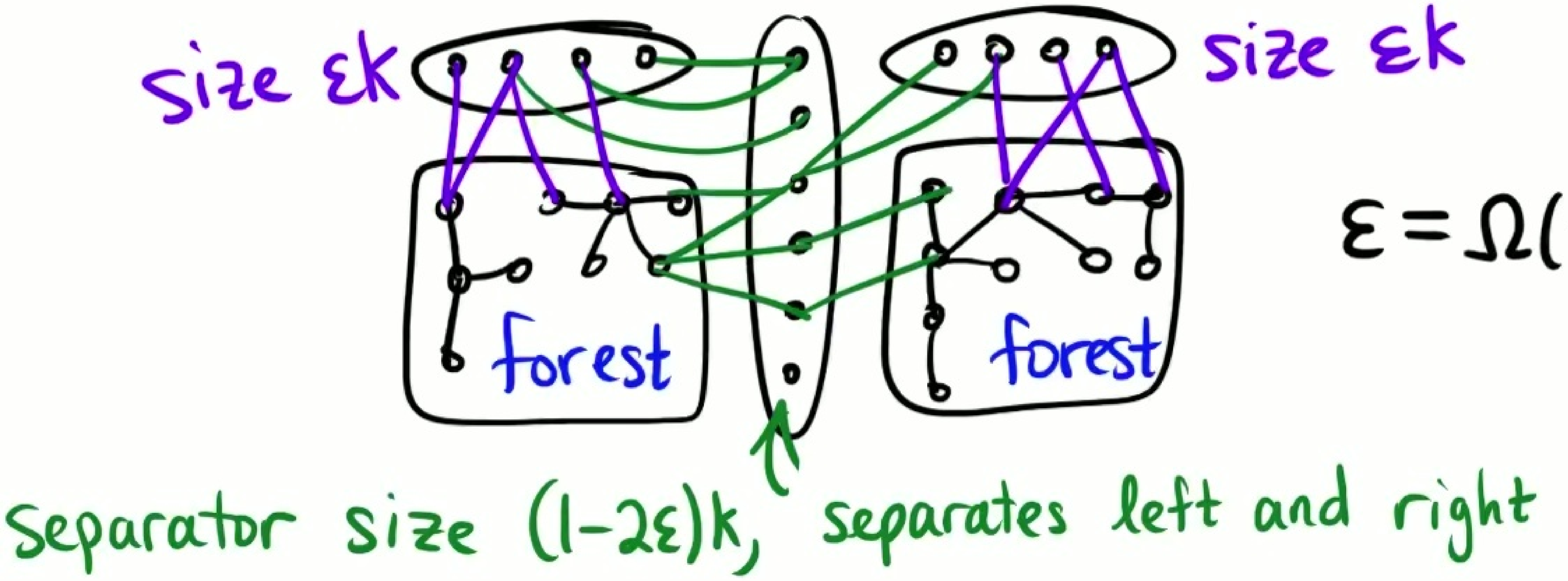
FVS $S_{i+1}$ of size k on $G[\{v_1,\ldots,v_{i+1}\}]$. Repeat

# Sparse case

Lemma: Given a graph with $m \leq 100k$, and given a FVS of size k+1,

# Sparse case

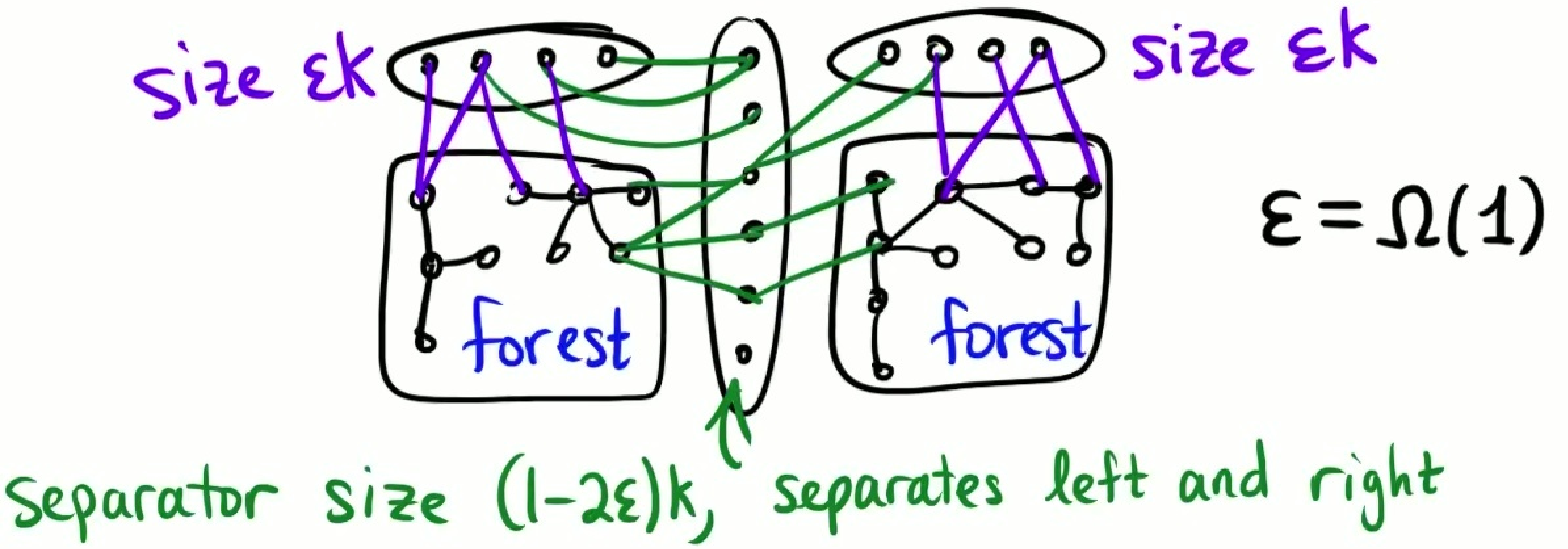
Lemma: Given a graph with $m \leq 100k$, and given a FVS of size $k+1$, we can decompose the graph into:




$\varepsilon = \Omega(1)$

# Sparse case

Lemma: Given a graph with $m \leq 100k$, and given a FVS of size k+1, we can decompose the graph into:

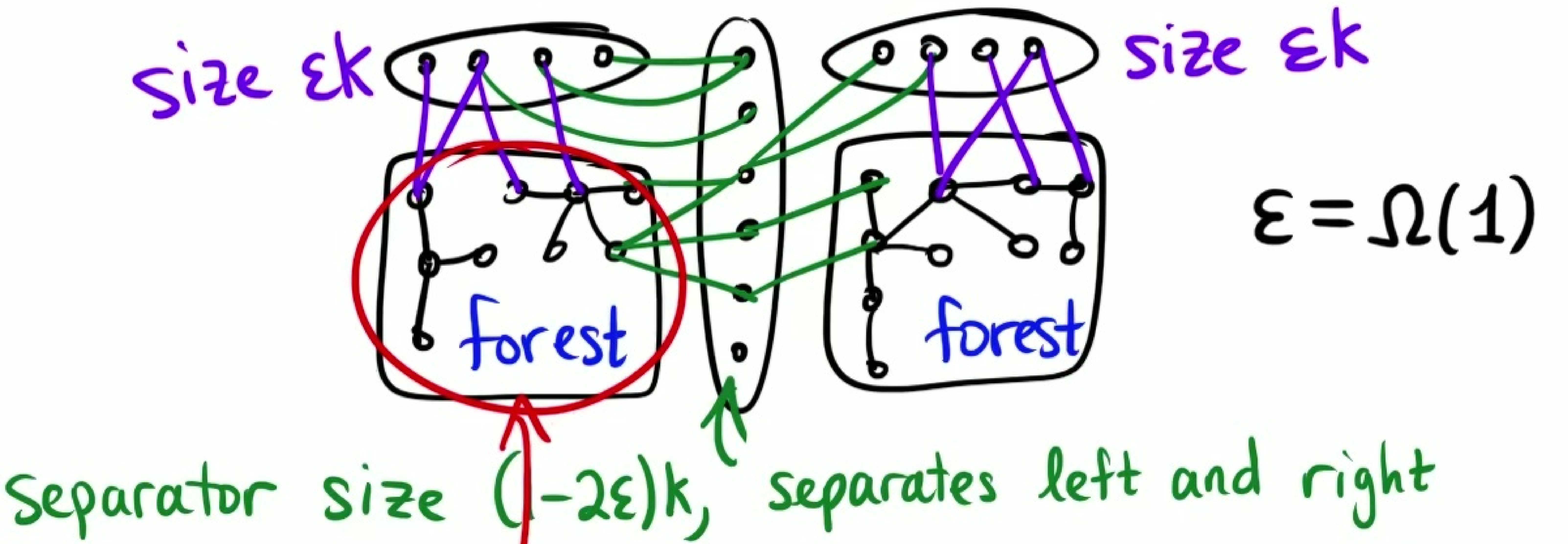



Claim: a graph with this decomposition has treewidth (1-Ω(1))k

Proof:

# Sparse case

Lemma: Given a graph with $m \le 100k$, and given a FVS of size $k+1$, we can decompose the graph into:



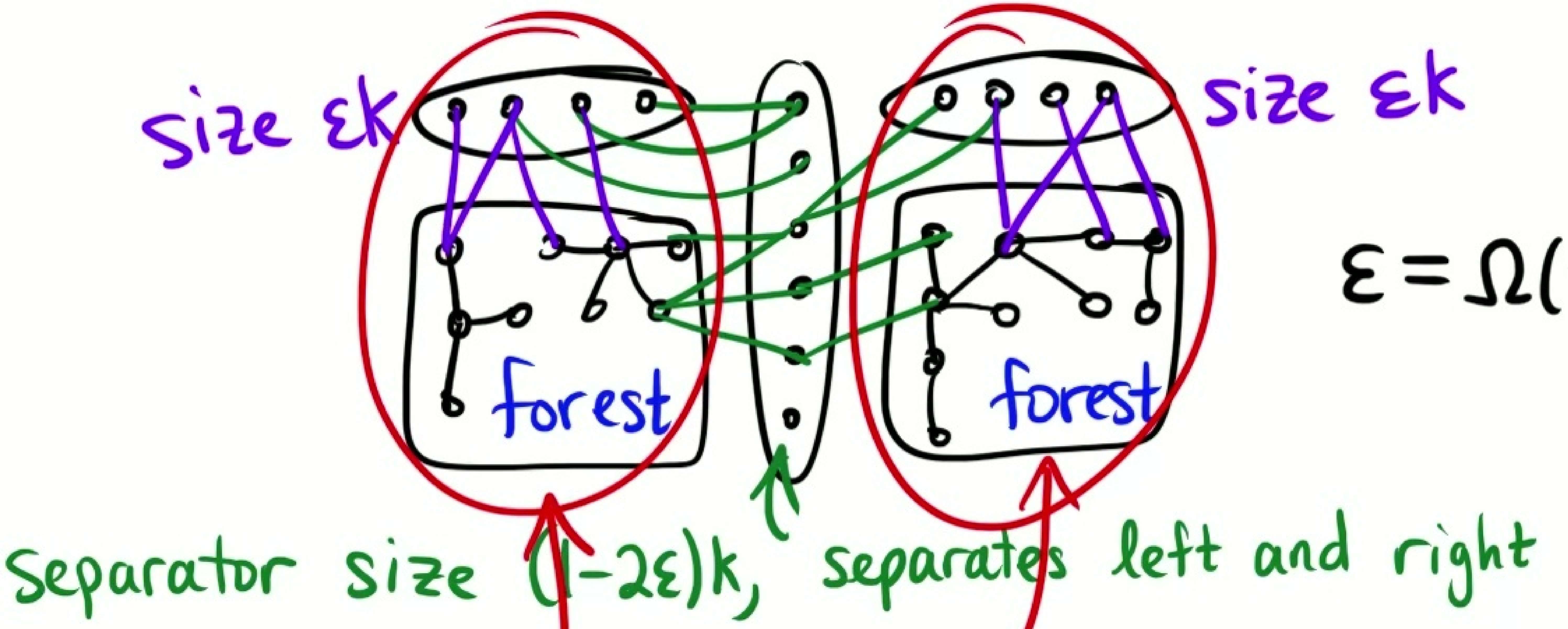

Claim: a graph with this decomposition has treewidth $(1-\Omega(1))k$

Proof:

tw( ) = 1 (forest)

# Sparse case

Lemma: Given a graph with m ≤ 100k, and given a FVS of size k+1, we can decompose the graph into:




$\varepsilon = \Omega(1)$

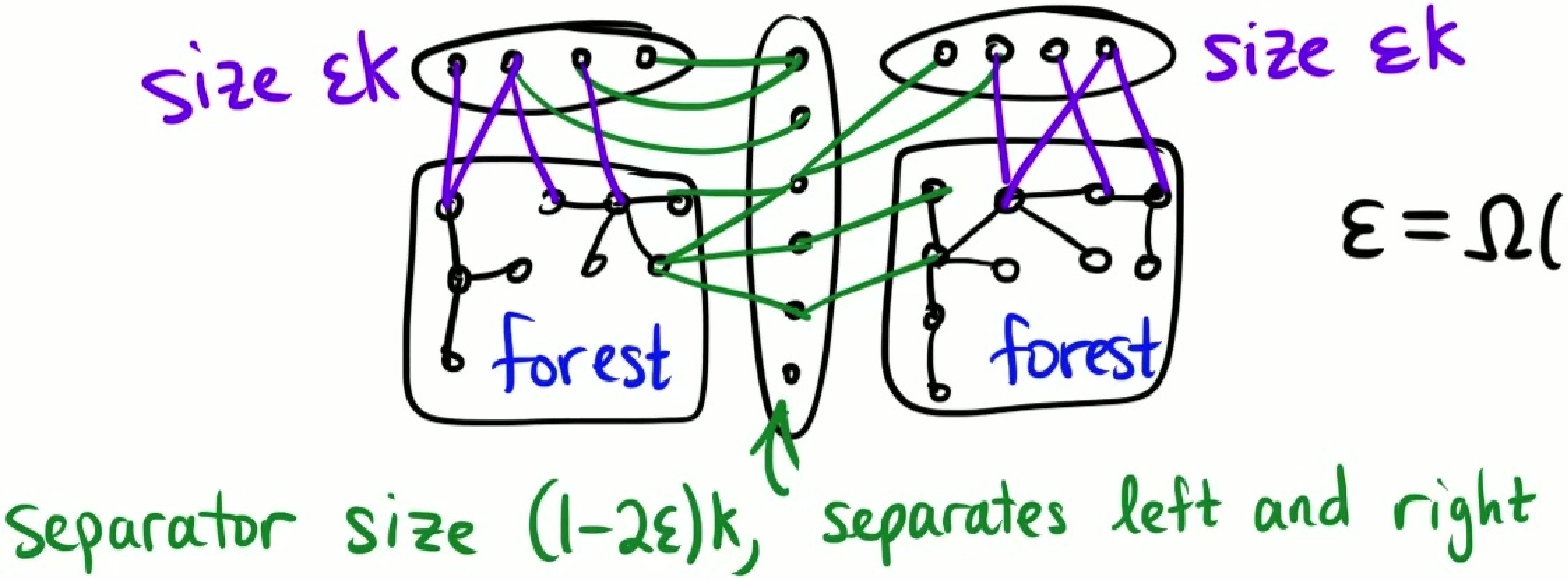
$tw(\quad) \le 1 + \varepsilon k$ (forest + εk vertices)

$tw(G) \le 1 + \varepsilon k + (1-2\varepsilon)k$ (add last $(1-2\varepsilon)k$ vertices)

$= (1-\Omega(1))k$

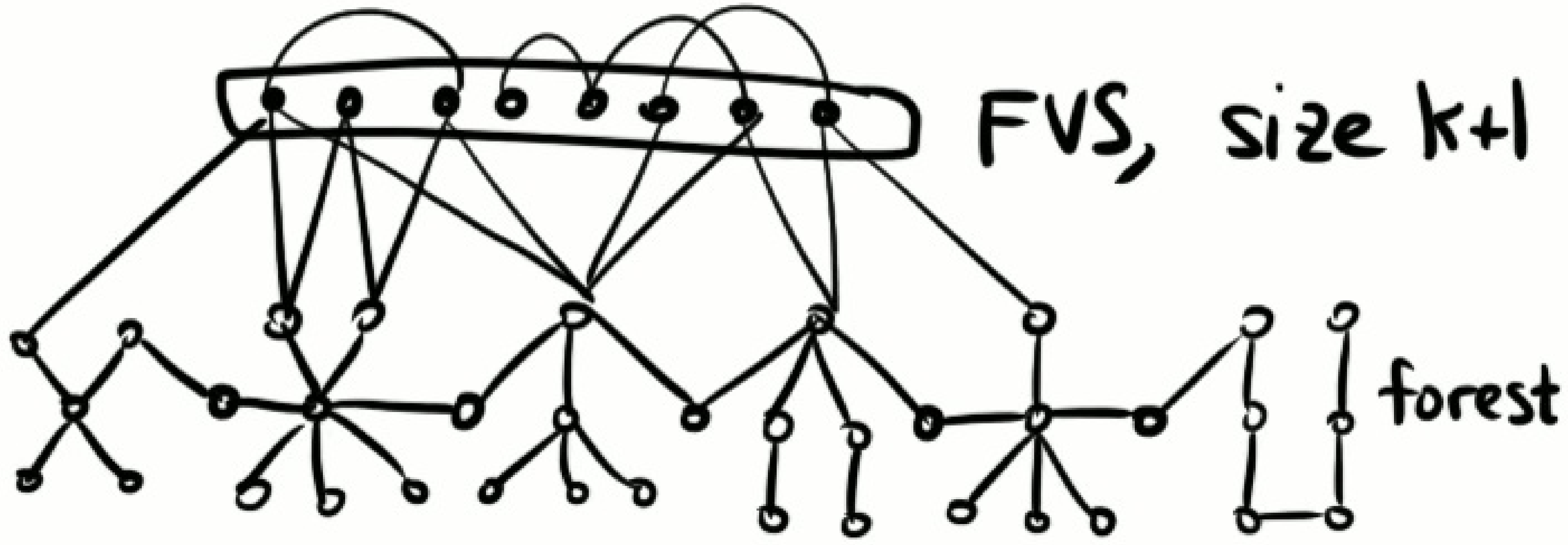
# Sparse case

Lemma: Given a graph with $m \leq 100k$, and given a FVS of size $k+1$, we can decompose the graph into:



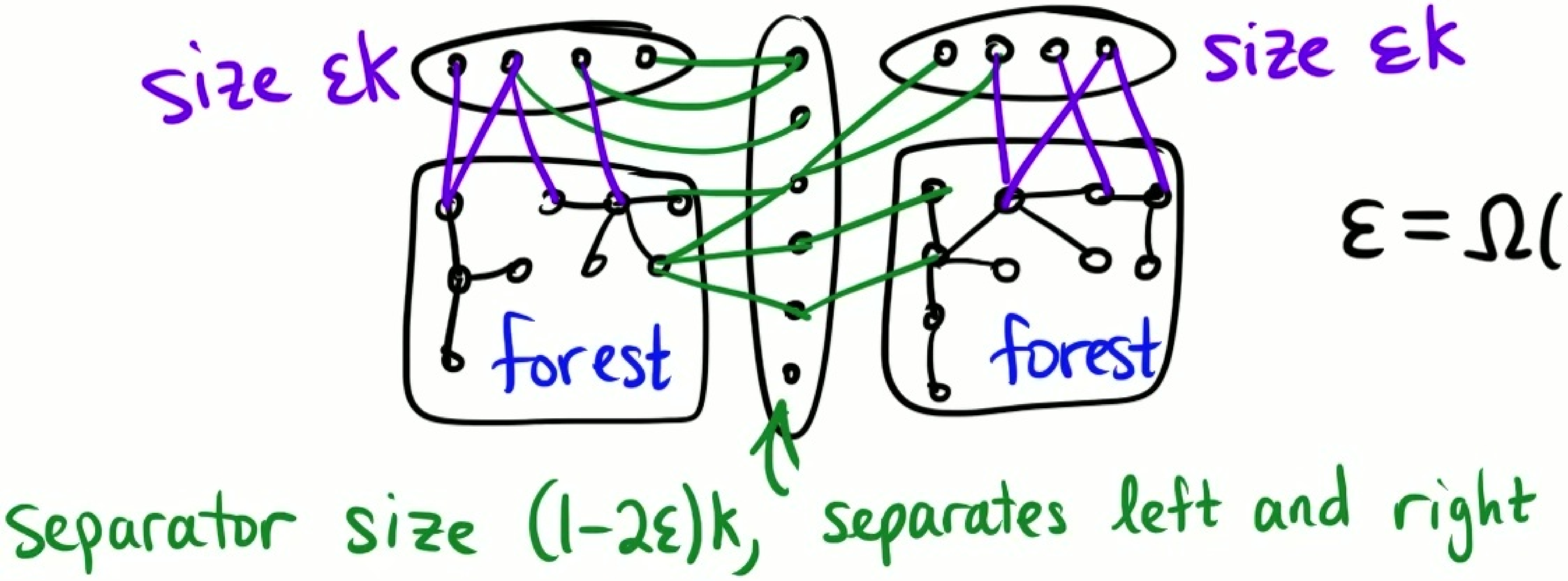

$\varepsilon = \Omega(1)$

Proof of Lemma:



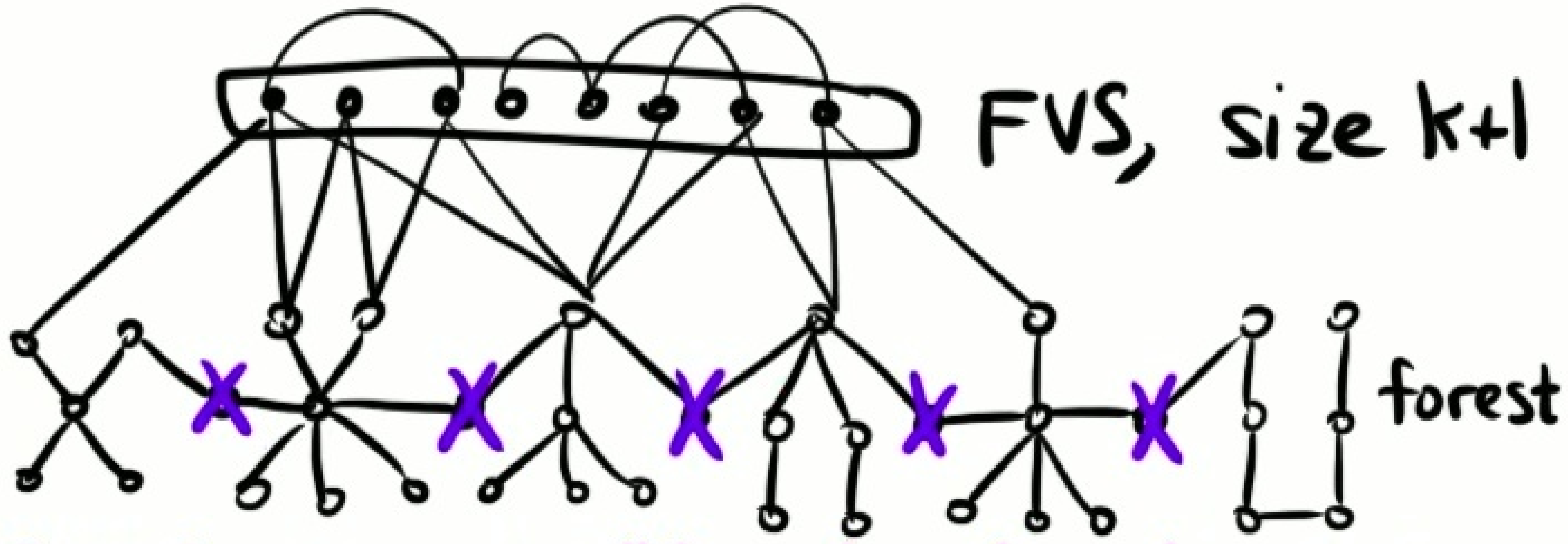
# Sparse case

Lemma: Given a graph with $m \leq 100k$, and given a FVS of size $k+1$, we can decompose the graph into:




$\varepsilon = \Omega(1)$

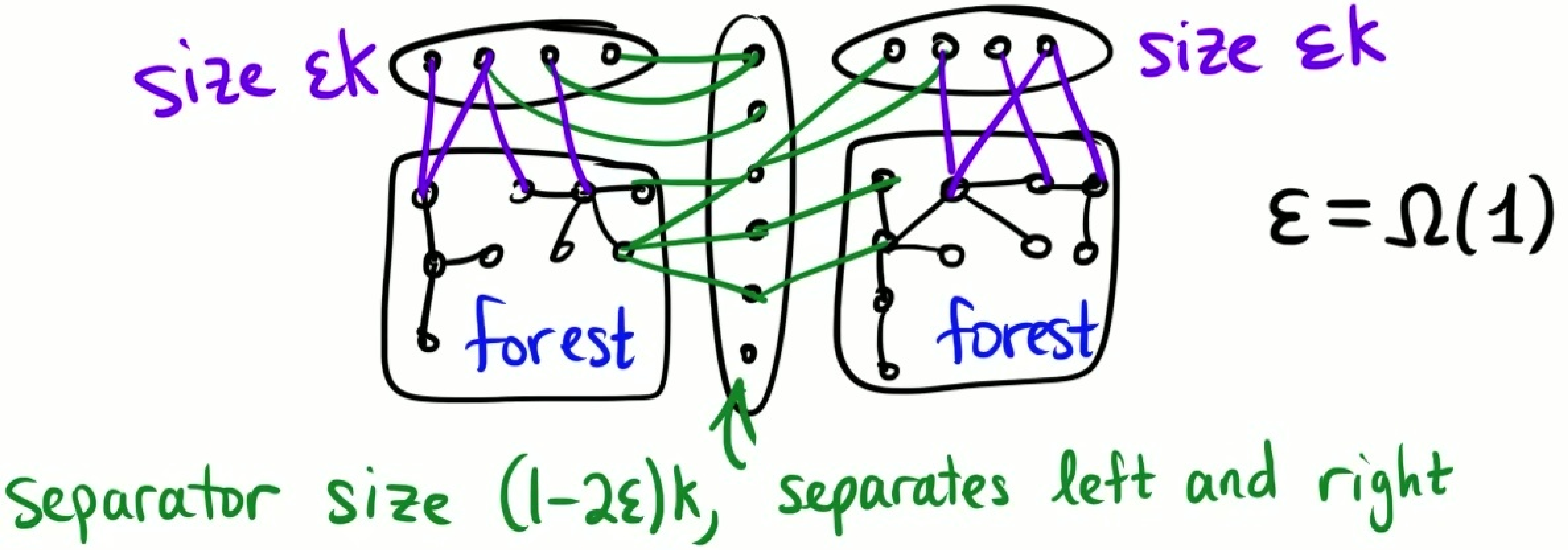
Proof of Lemma:



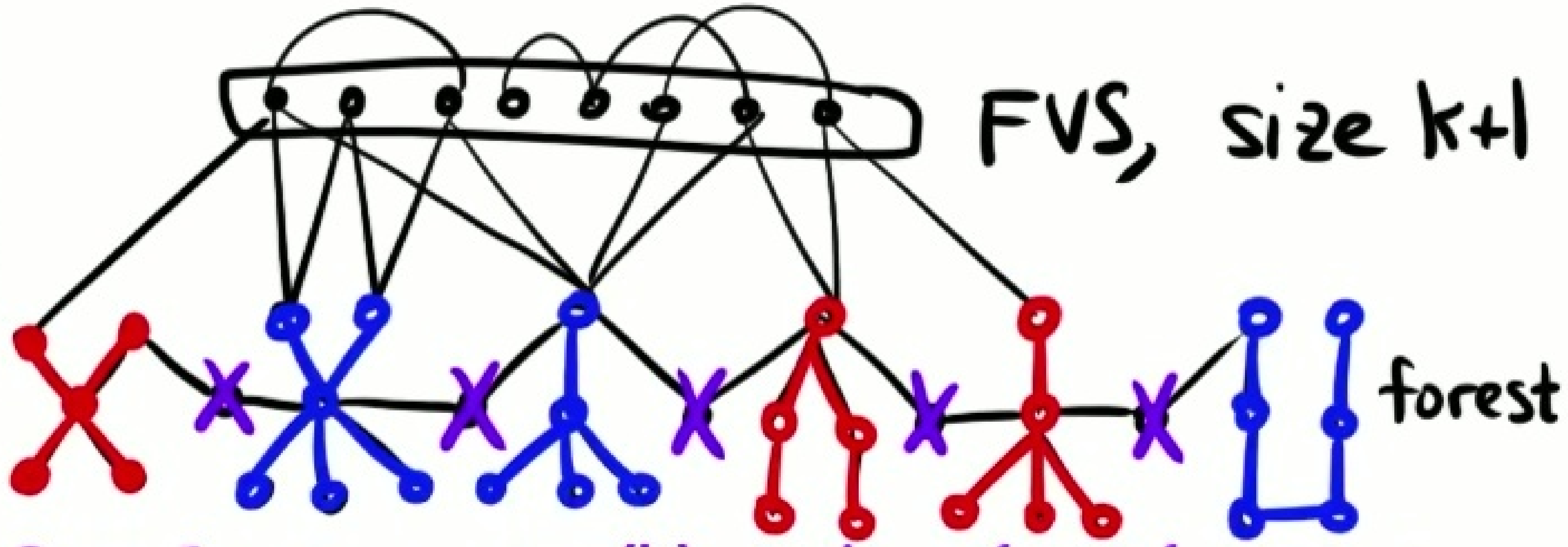

Step 1: remove t=o(k) vertices from forest s.t. each remaining component is "small". These are added to separator

# Sparse case

Lemma: Given a graph with m ≤ 100k, and given a FVS of size k+1, we can decompose the graph into:




Proof of Lemma:

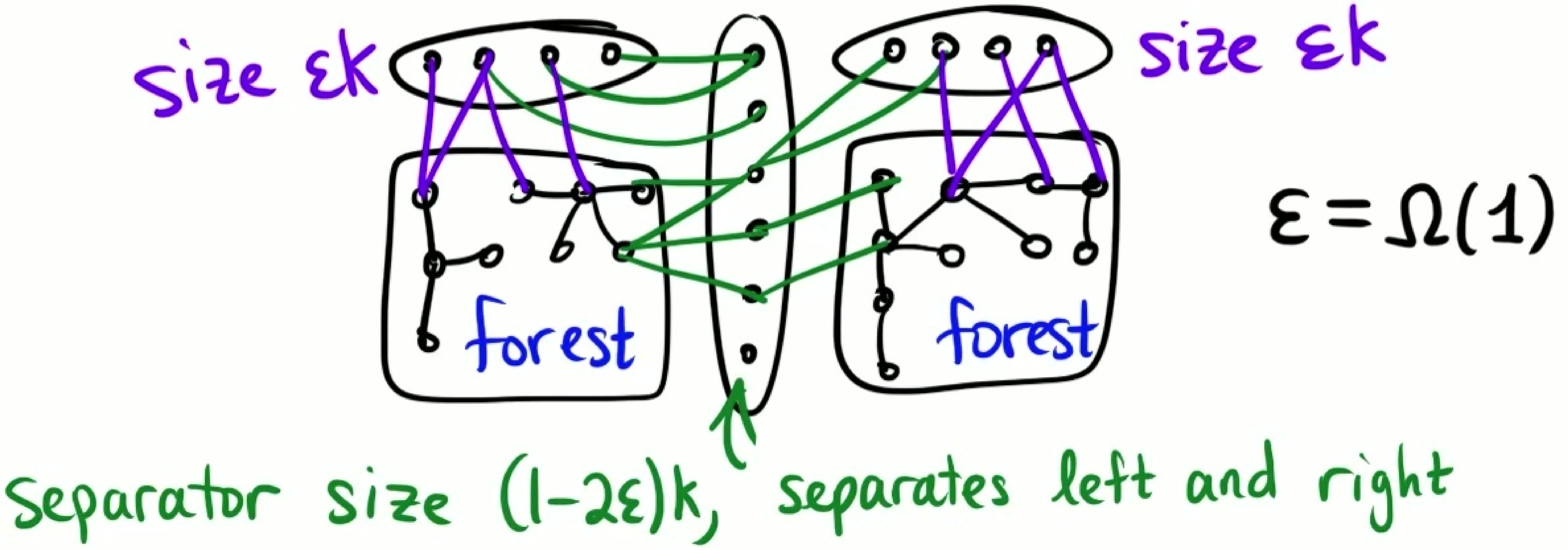



Step 1: remove t=o(k) vertices from forest s.t. each remaining component is "small". These are added to separator

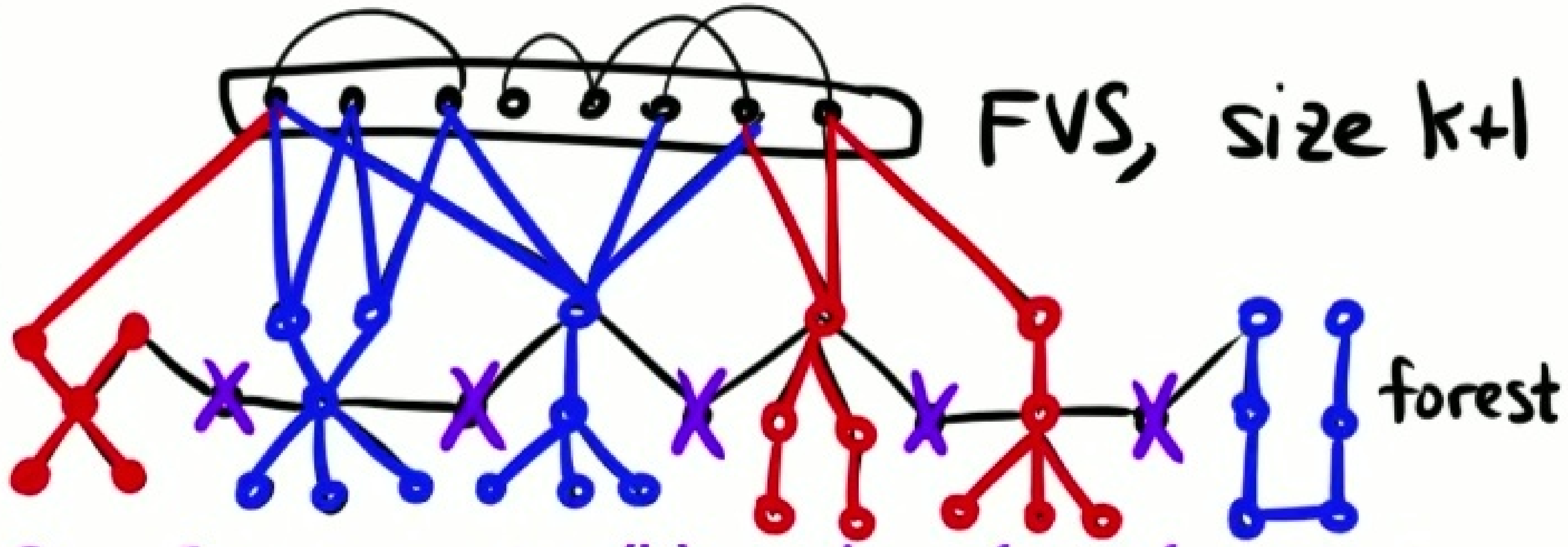
Step 2: for each component, color red/blue w.p. 1/2 each

# Sparse case

Lemma: Given a graph with $m \le 100k$, and given a FVS of size $k+1$, we can decompose the graph into:




## Proof of Lemma:

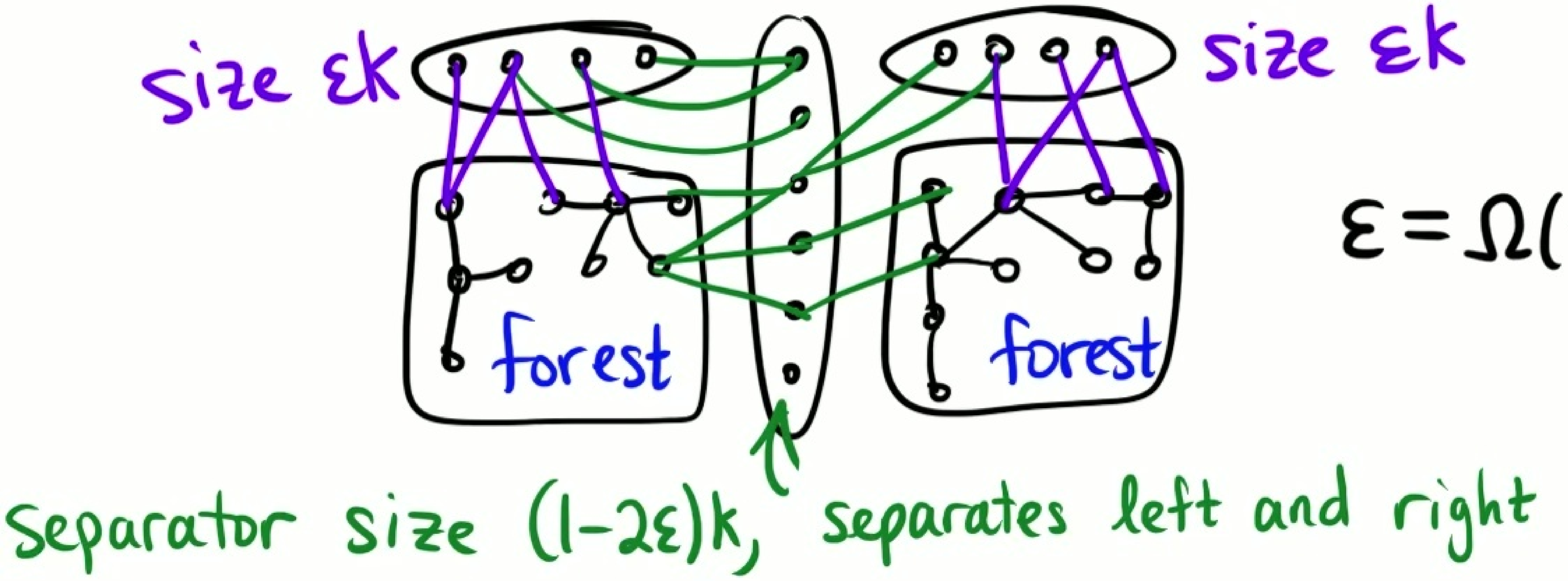



Step 1: remove t=o(k) vertices from forest s.t. each remaining component is "small". These are added to separator

Step 2: for each component, color red/blue w.p. 1/2 each

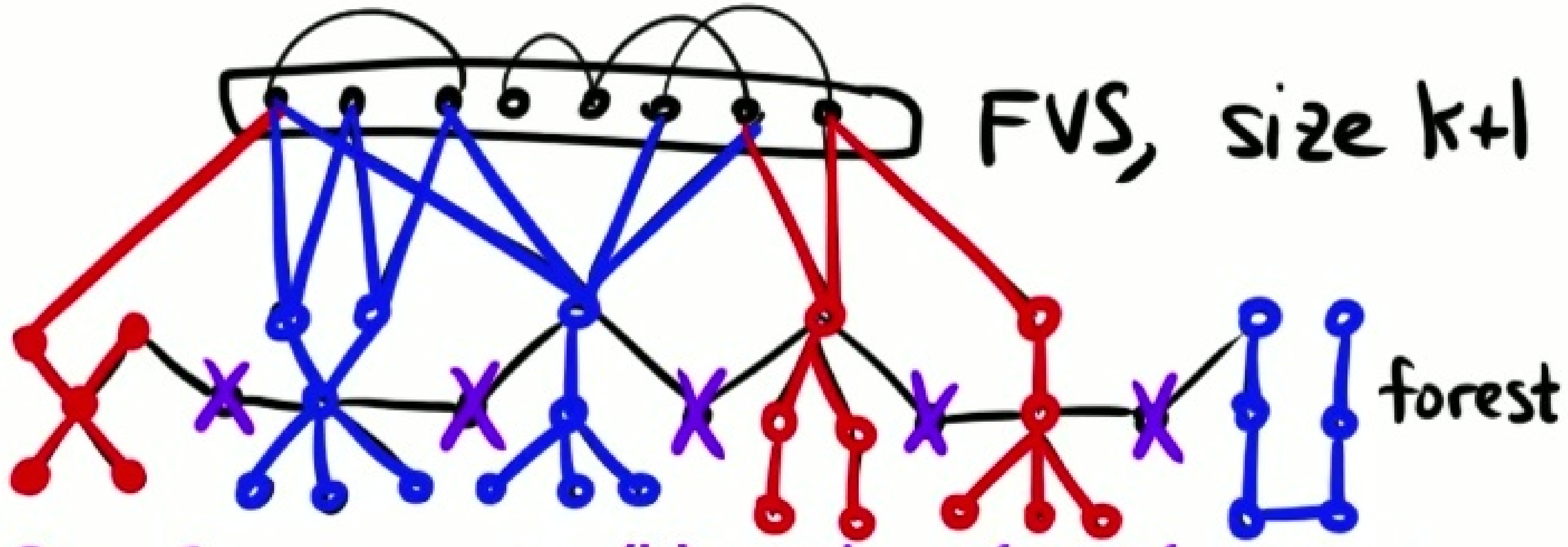
# Sparse case

Lemma: Given a graph with $m \le 100k$, and given a FVS of size $k+1$, we can decompose the graph into:




$\varepsilon = \Omega(1)$

## Proof of Lemma:



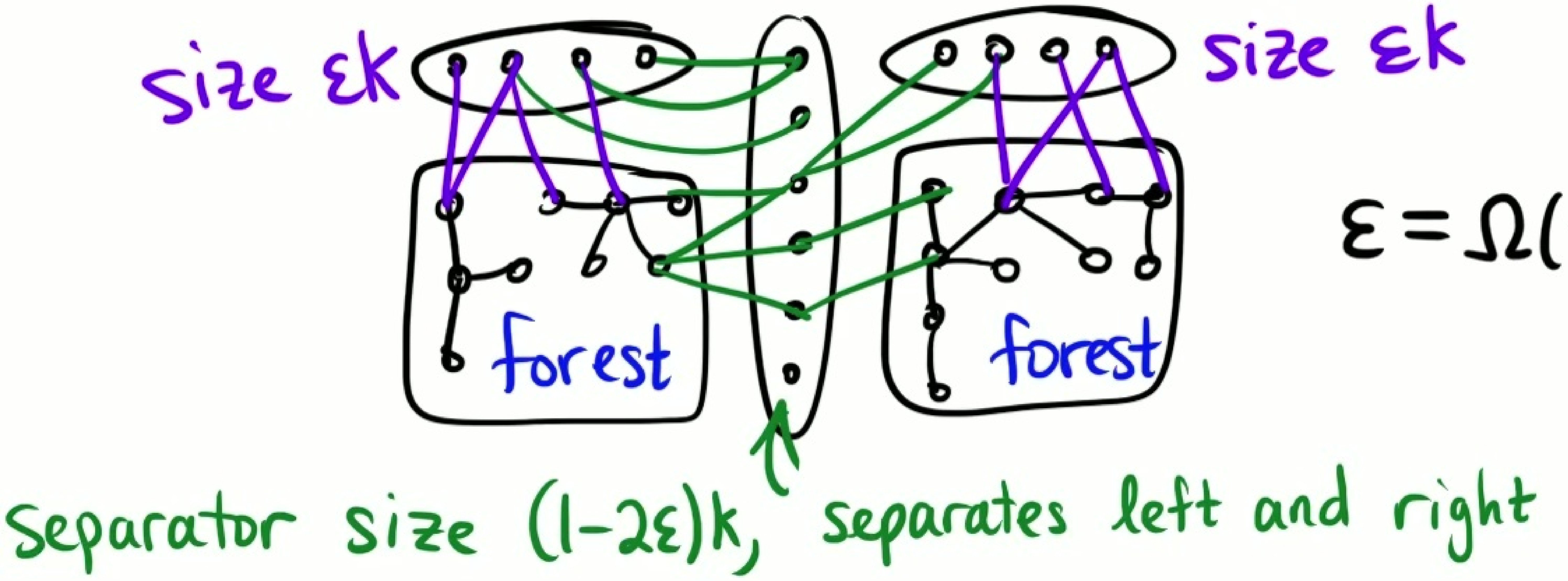

Step 1: remove $t=o(k)$ vertices from forest s.t. each remaining component is "small". These are added to separator

Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each

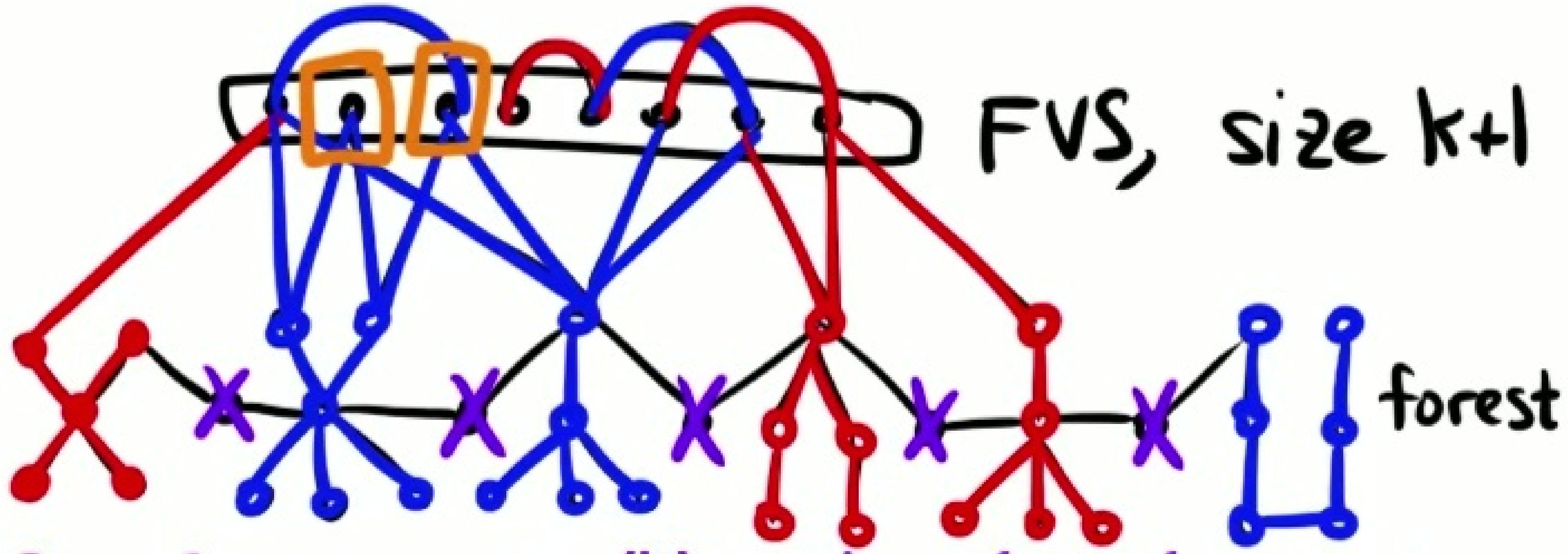
# Sparse case

Lemma: Given a graph with $m \leq 100k$, and given a FVS of size $k+1$, we can decompose the graph into:




$\varepsilon = \Omega(1)$

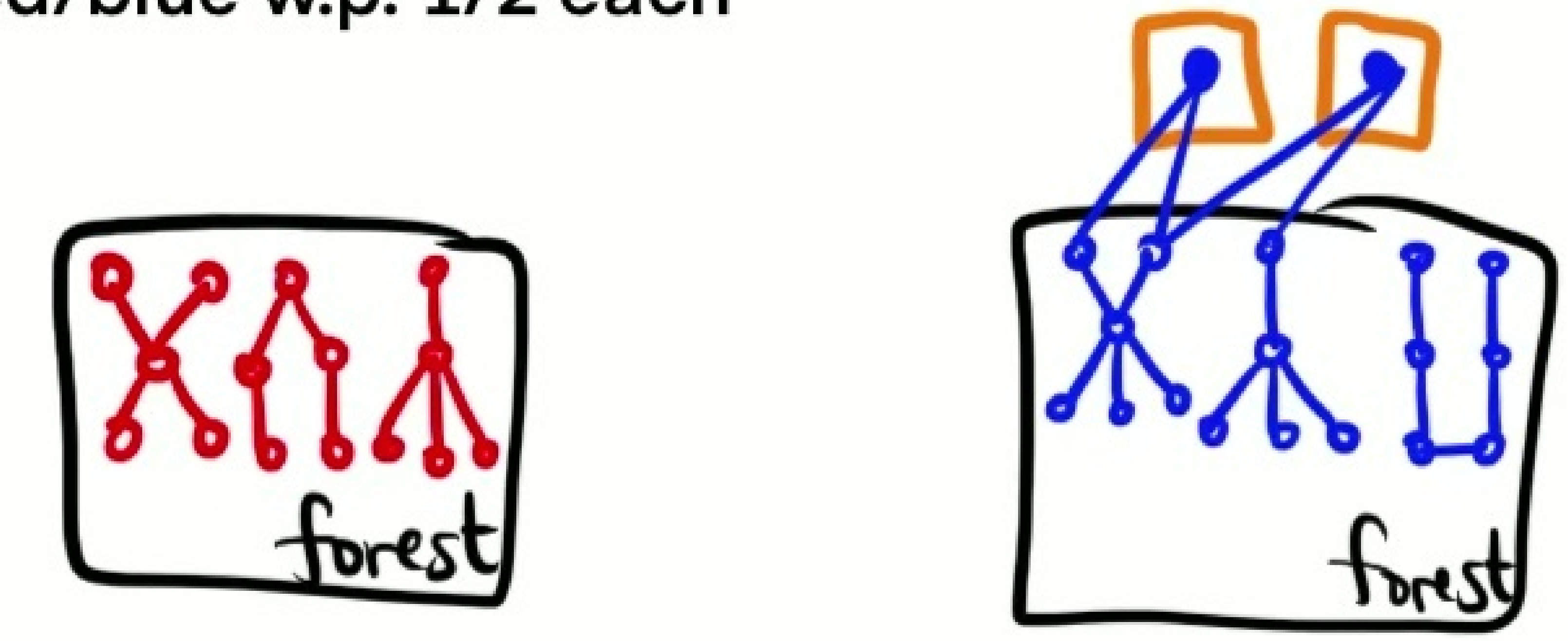
## Proof of Lemma:




Step 1: remove $t=o(k)$ vertices from forest s.t. each remaining component is "small". These are added to separator

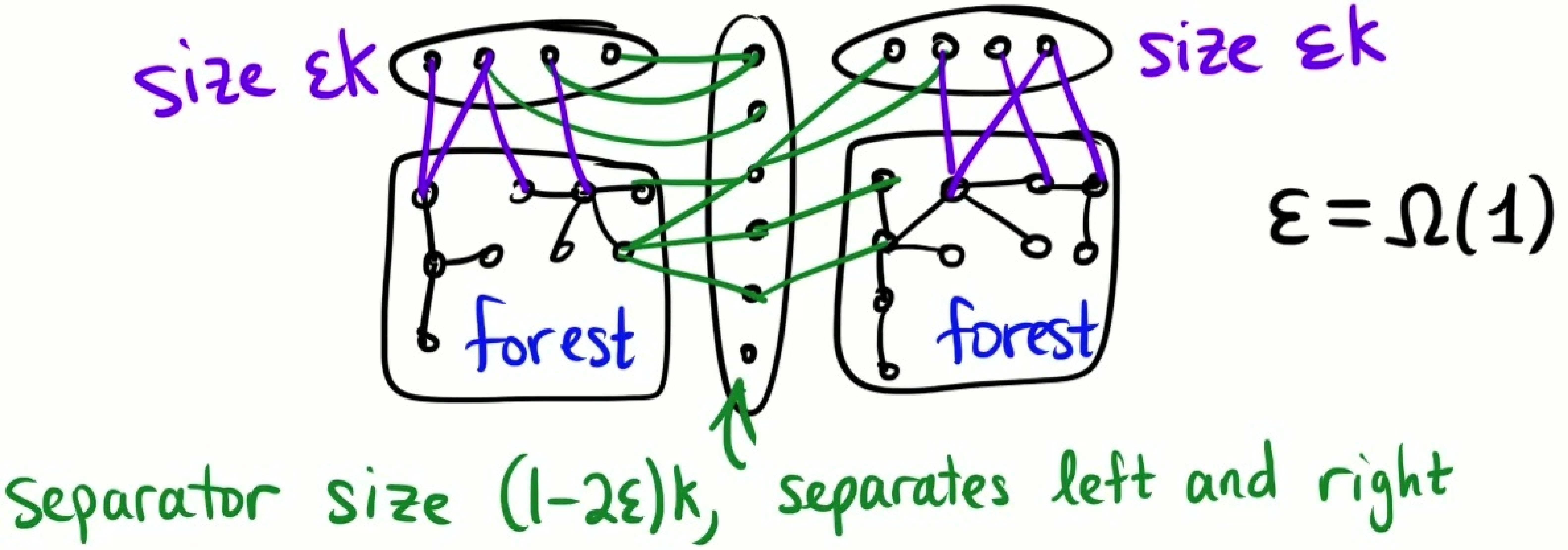
Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each

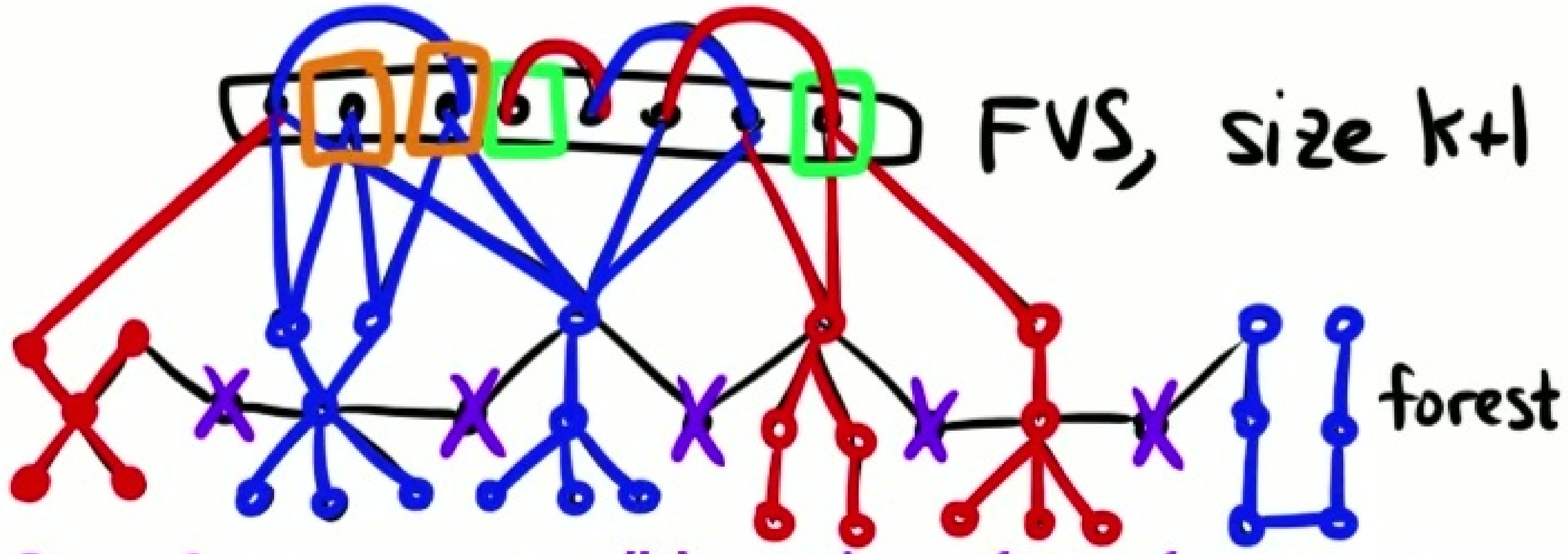


# Sparse case

Lemma: Given a graph with $m \le 100k$, and given a FVS of size $k$+1, we can decompose the graph into:



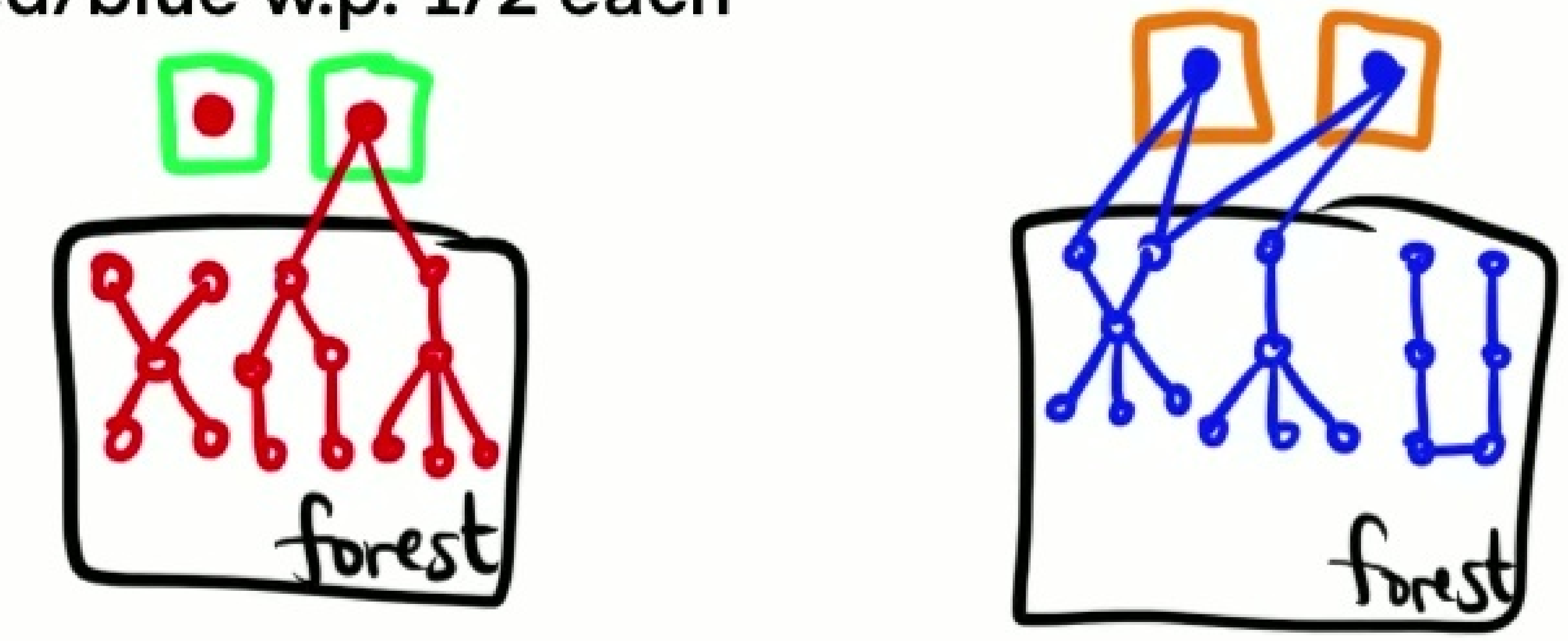

## Proof of Lemma:




Step 1: remove $t = o(k)$ vertices from forest s.t. each remaining component is "small". These are added to separator

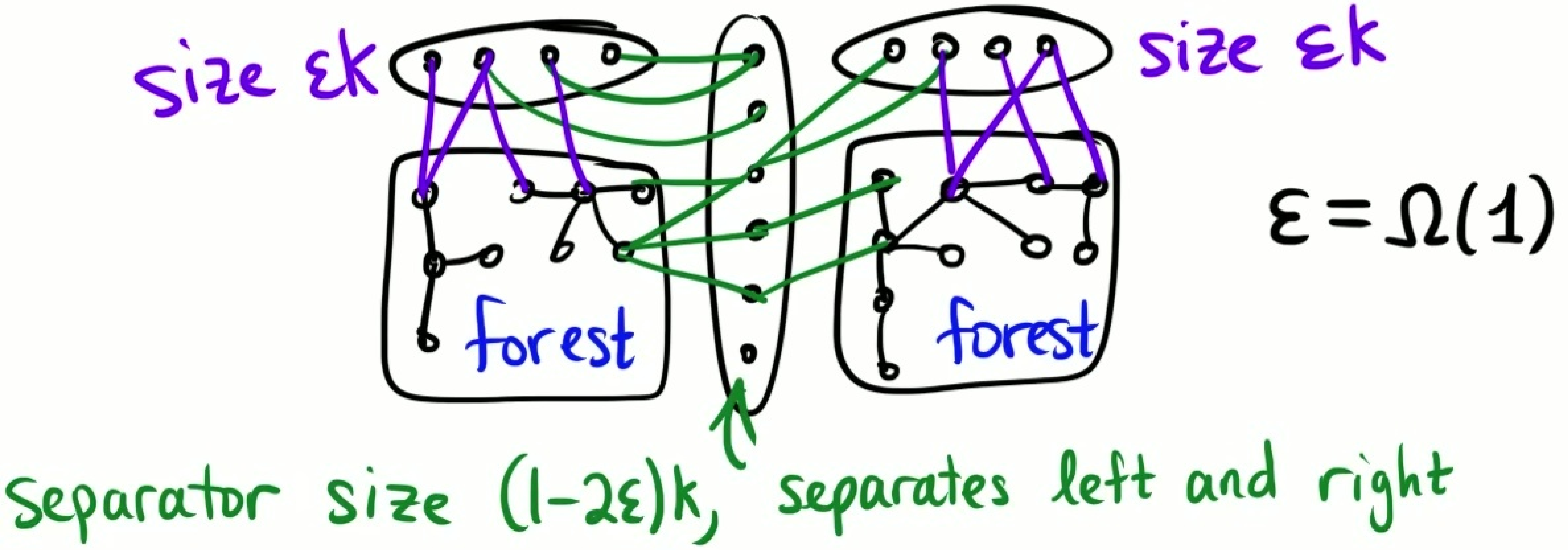
Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each

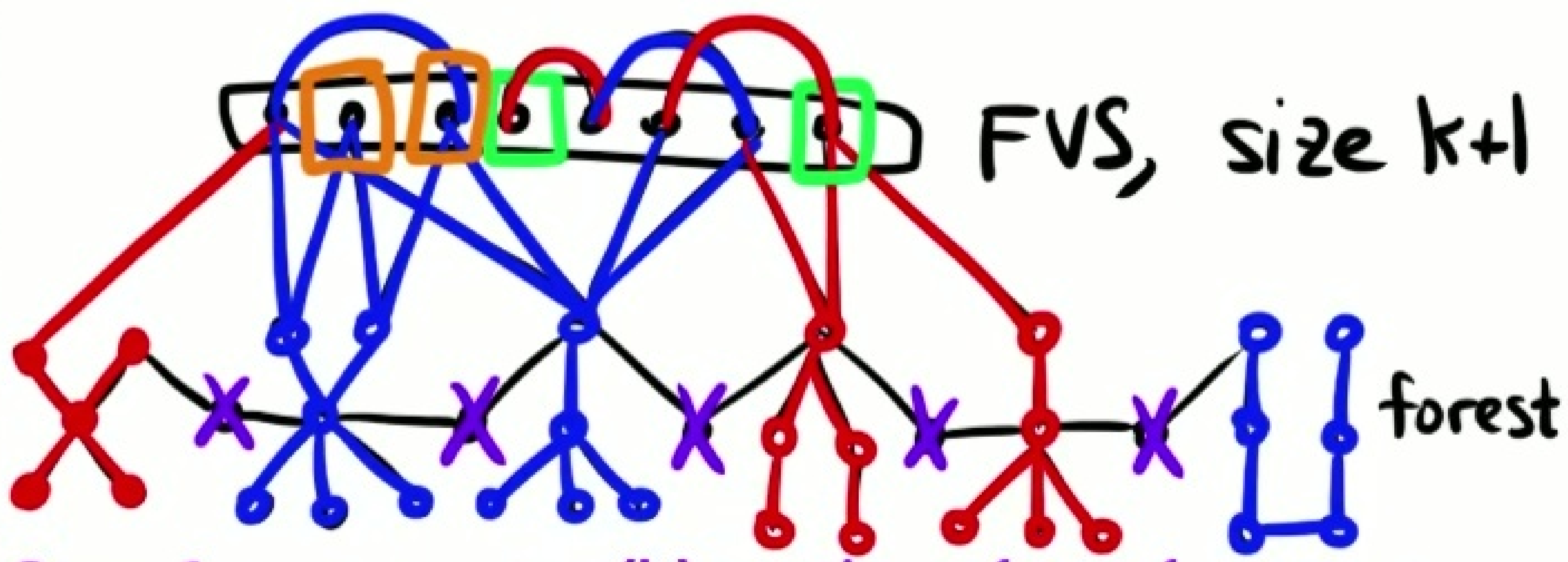


# Sparse case

Lemma: Given a graph with m ≤ 100k, and given a FVS of size k+1, we can decompose the graph into:



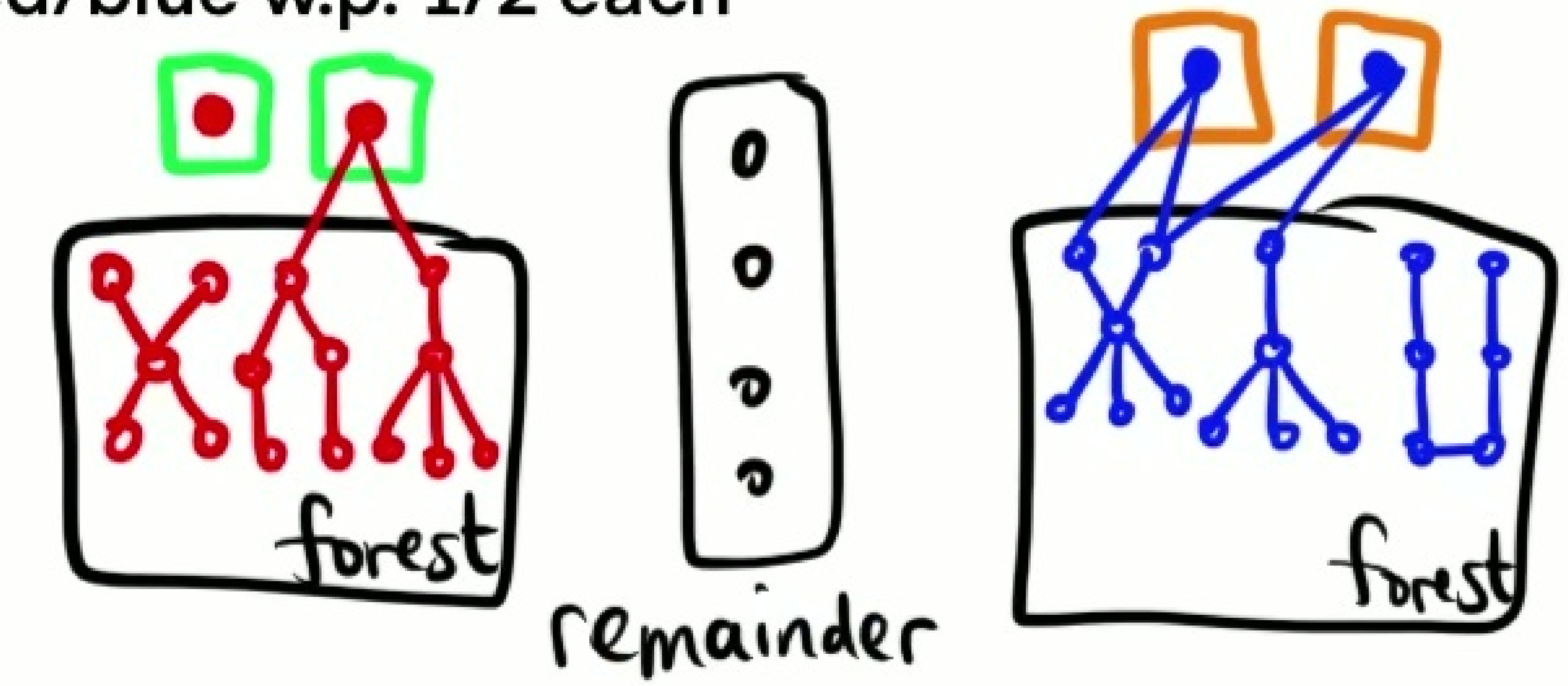

## Proof of Lemma:




Step 1: remove t=o(k) vertices from forest s.t. each remaining component is "small". These are added to separator

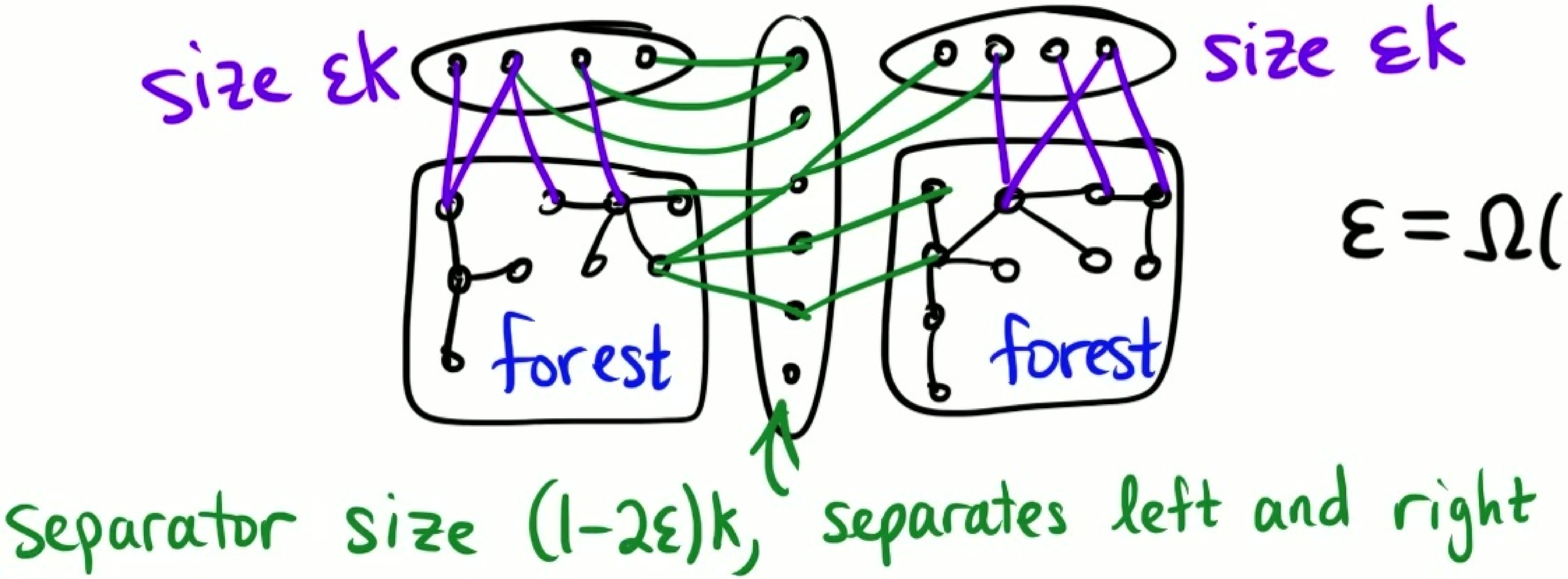
Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each



# Sparse case

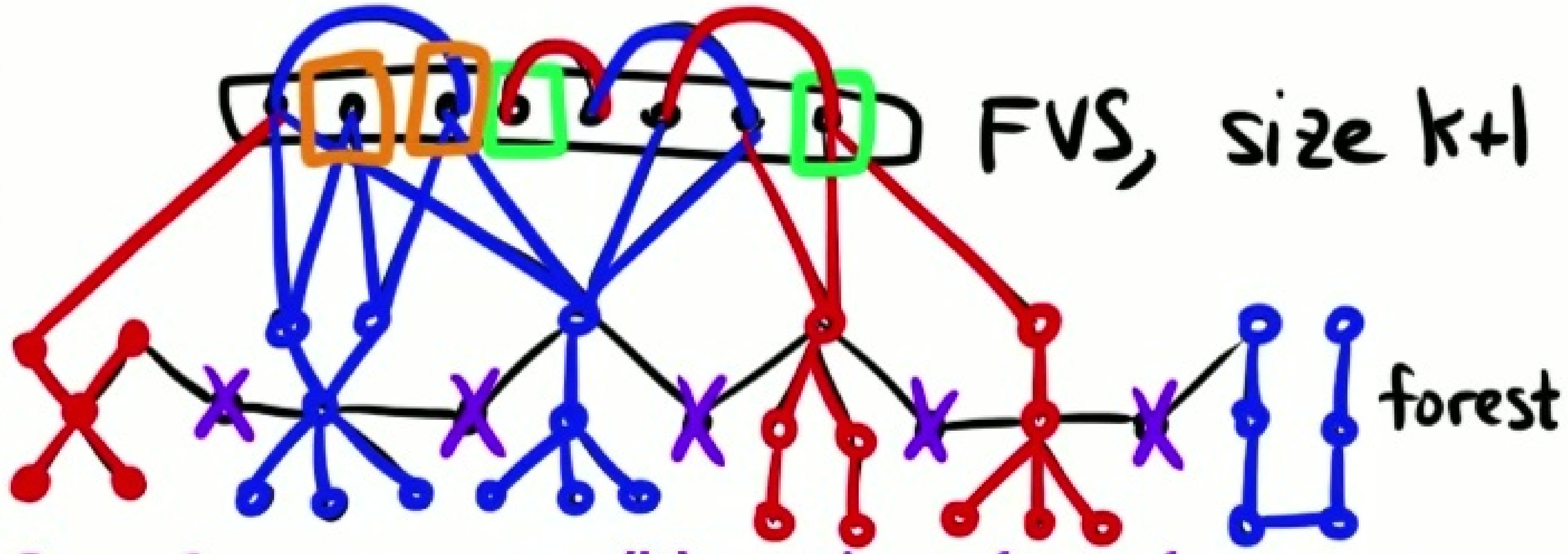
Lemma: Given a graph with $m \leq 100k$, and given a FVS of size k+1, we can decompose the graph into:




$\varepsilon = \Omega(1)$

$\Pr[\text{all incident edges red}] \geq 2^{-\deg(v)}$

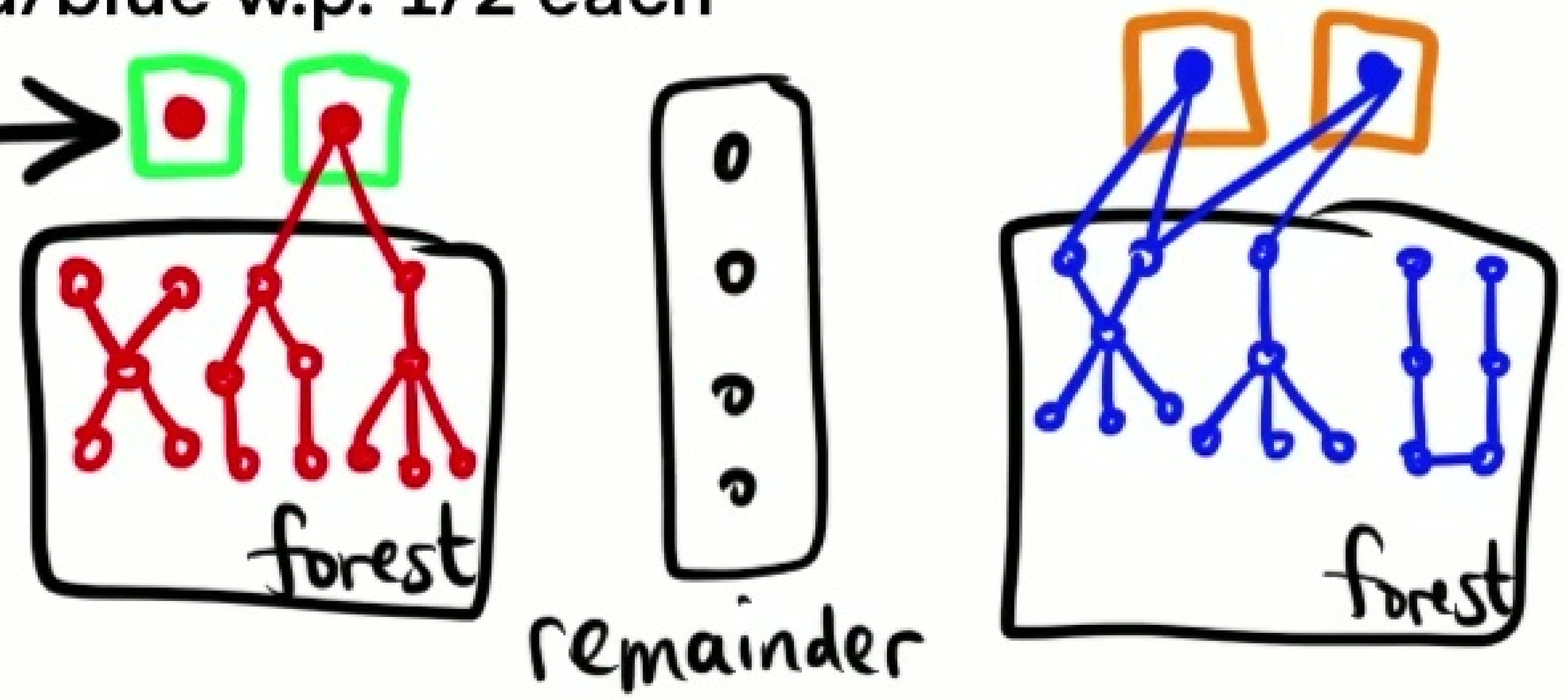
Proof of Lemma:




Step 1: remove t=o(k) vertices from forest s.t. each remaining component is "small". These are added to separator

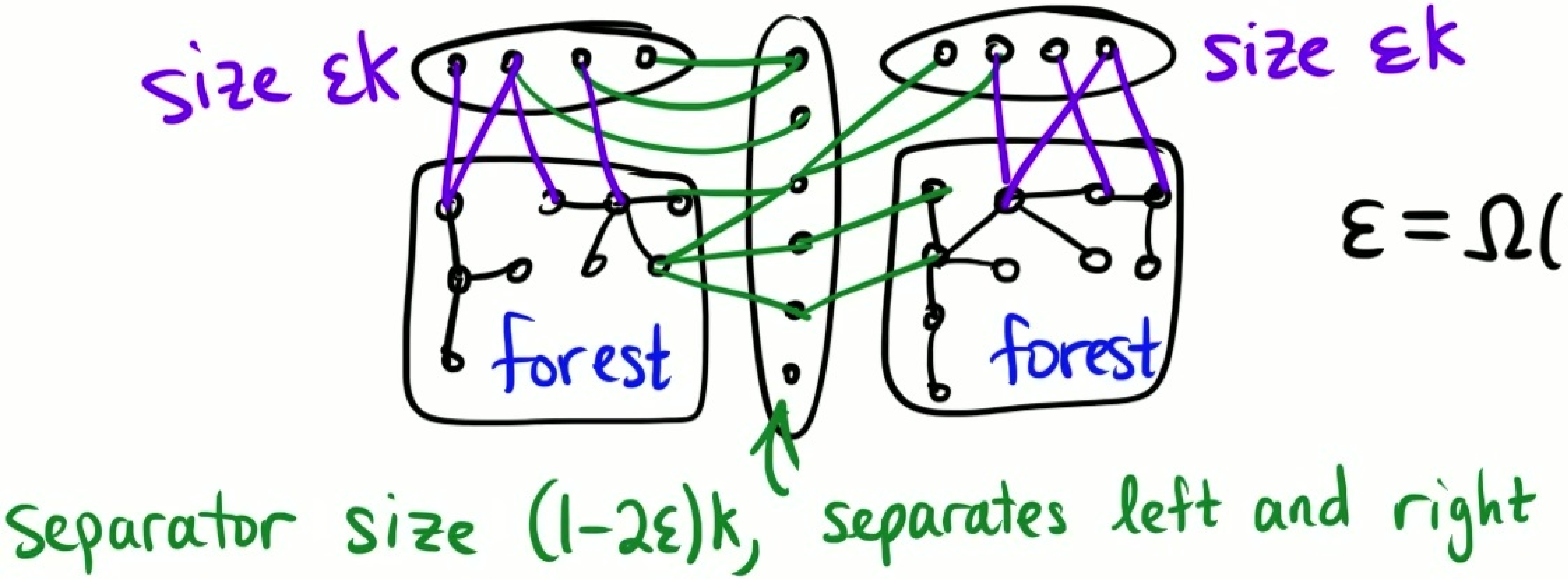
Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each



# Sparse case

Lemma: Given a graph with $m \le 100k$, and given a FVS of size $k+1$, we can decompose the graph into:

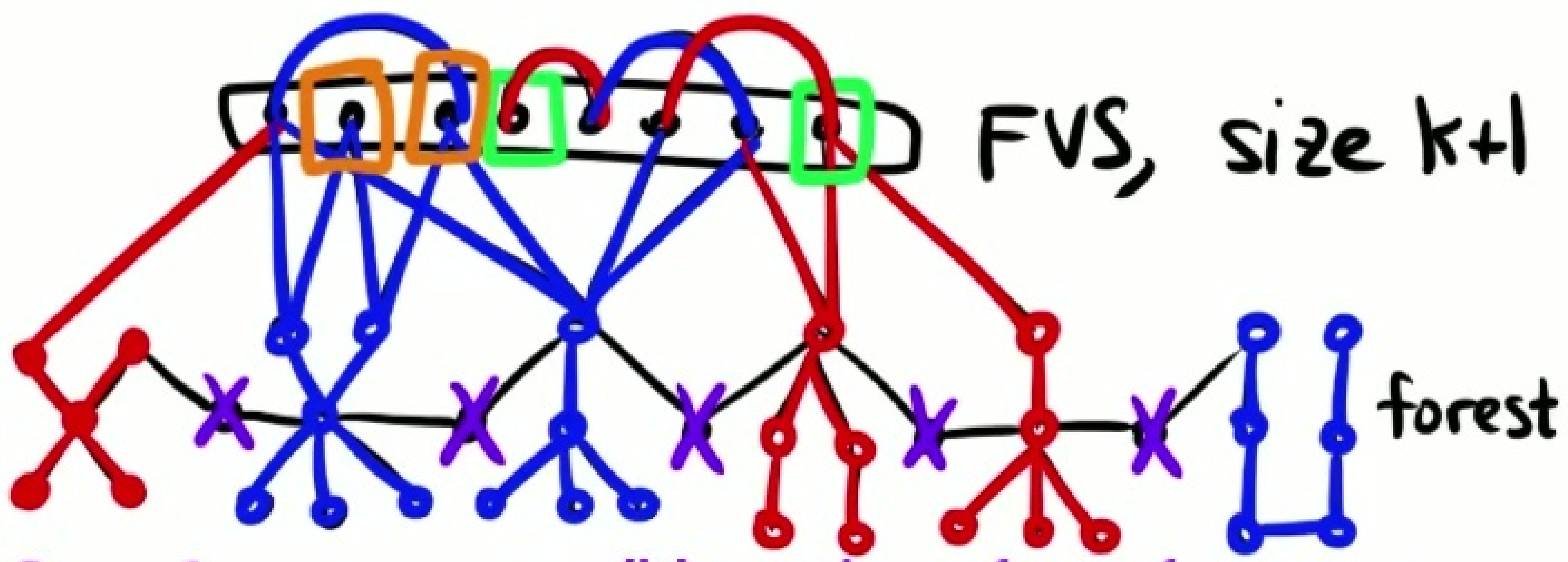



$\varepsilon = \Omega(1)$

$$\Pr[\text{all incident edges red}] \ge 2^{-\deg(v)}$$

$$\mathbb{E}[\#\text{ such } v] \ge \sum_{v \in FVS} 2^{-\deg(v)} \underset{\text{Jensen}}{\ge} |FVS| \cdot 2^{-(\text{avg deg})}$$

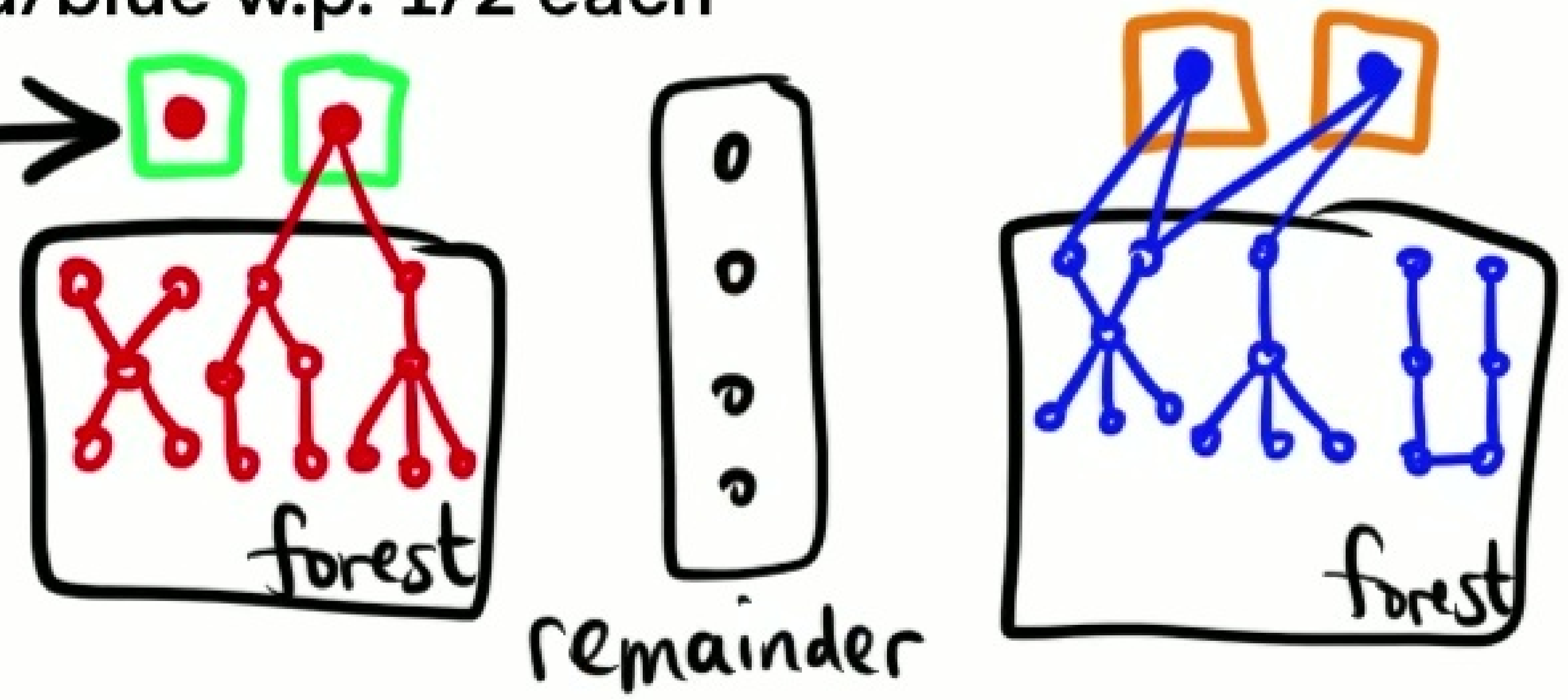
## Proof of Lemma:




Step 1: remove $t=o(k)$ vertices from forest s.t. each remaining component is "small". These are added to separator

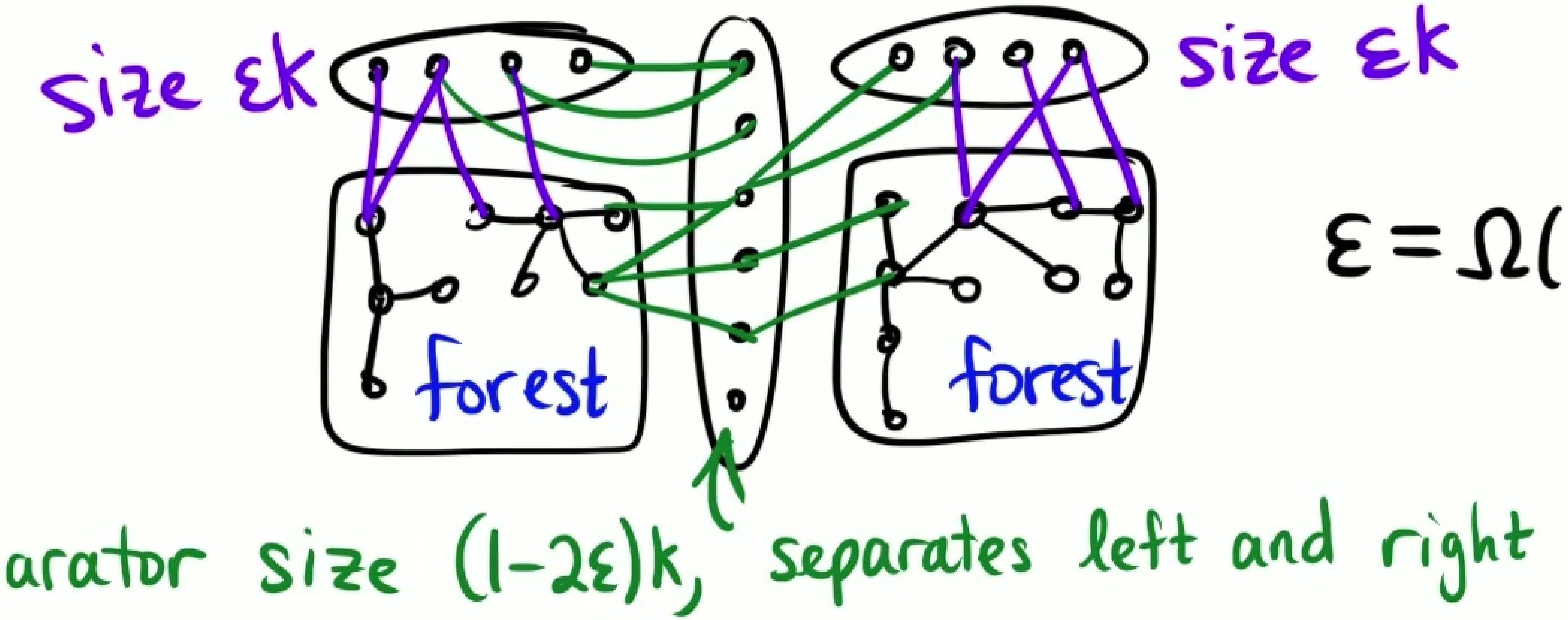
Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each



# Sparse case

Lemma: Given a graph with $m \leq 100k$, and given a FVS of size $k+1$, we can decompose the graph into:

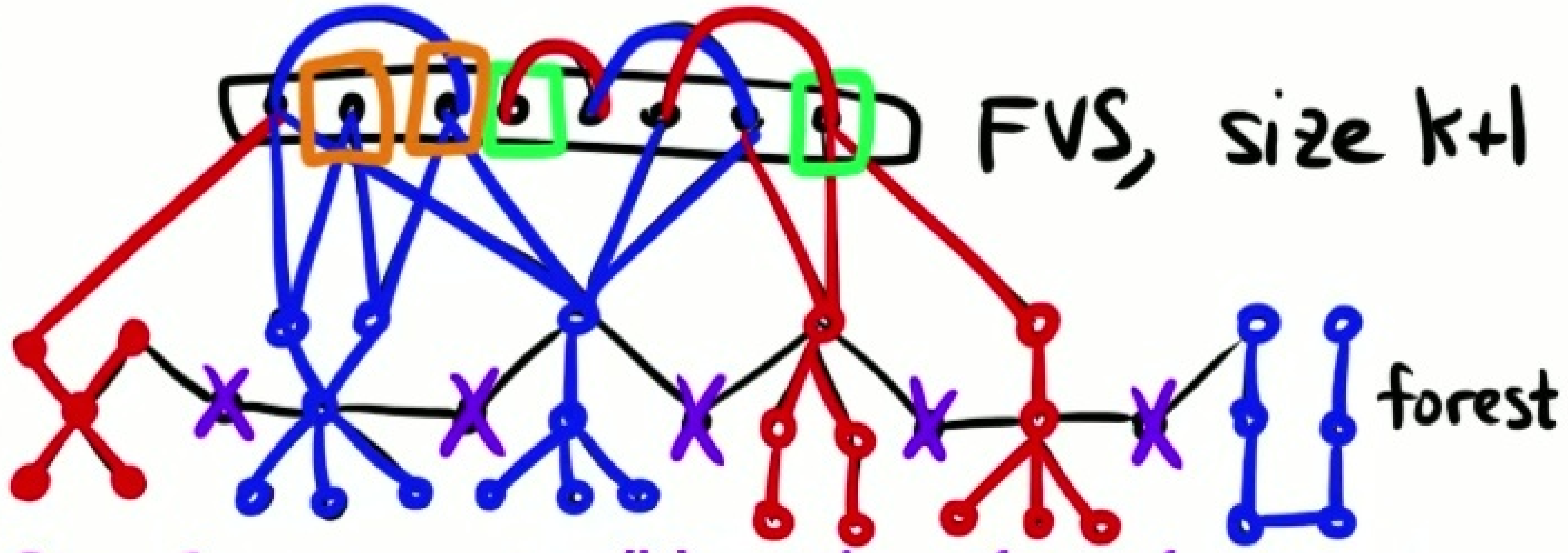



$\varepsilon = \Omega(1)$

$\Pr[\text{all incident edges red}] \geq 2^{-\deg(v)}$

$\mathbb{E}[\#\text{ such } v] \geq \sum_{v \in FVS} 2^{-\deg(v)} \overset{\text{Jensen}}{\geq} \overbrace{|FVS|}^{k+1} \cdot 2^{-\overbrace{(\text{avg deg})}^{\leq \frac{2m}{k+1} \leq 200}} \geq 2^{-200} k.$

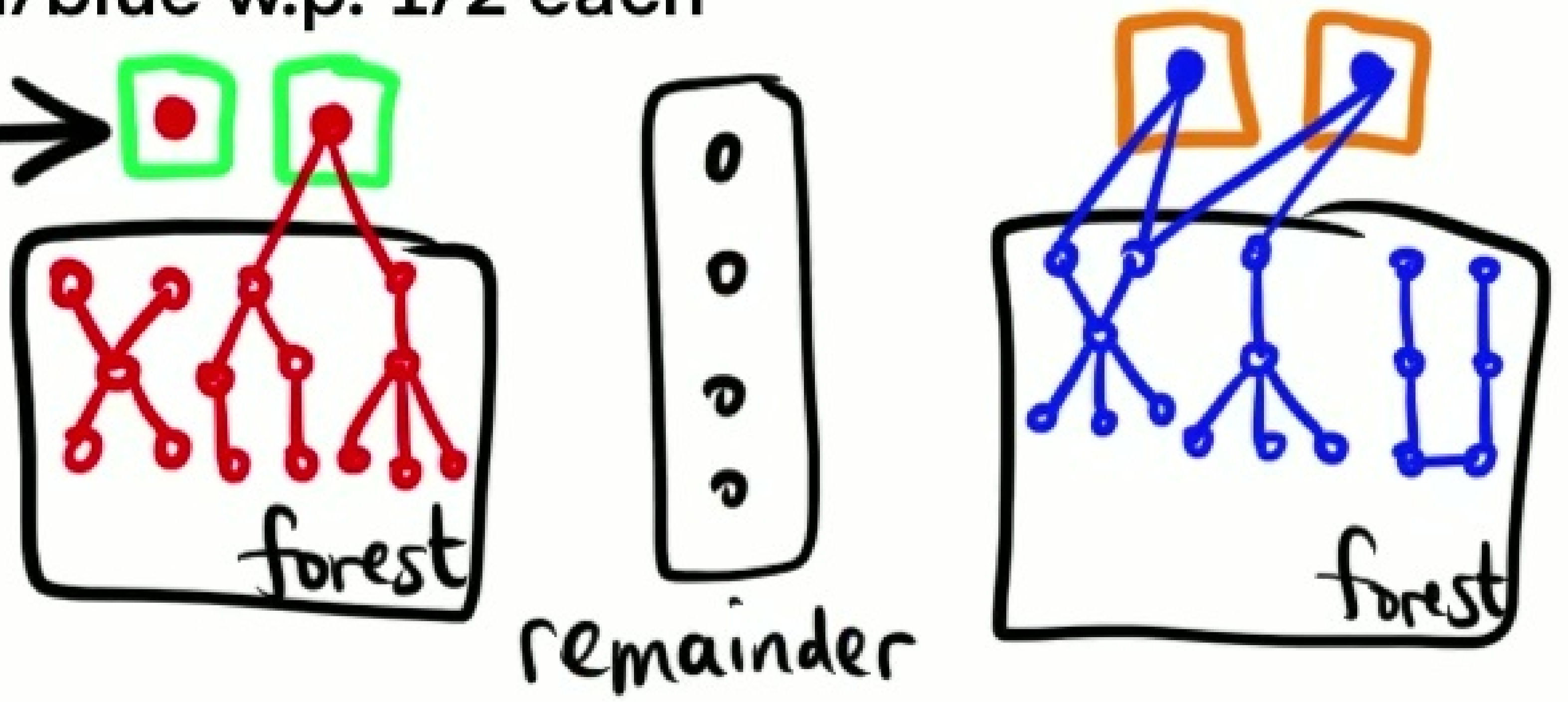
Proof of Lemma:




Step 1: remove $t = o(k)$ vertices from forest s.t. each remaining component is "small". These are added to separator

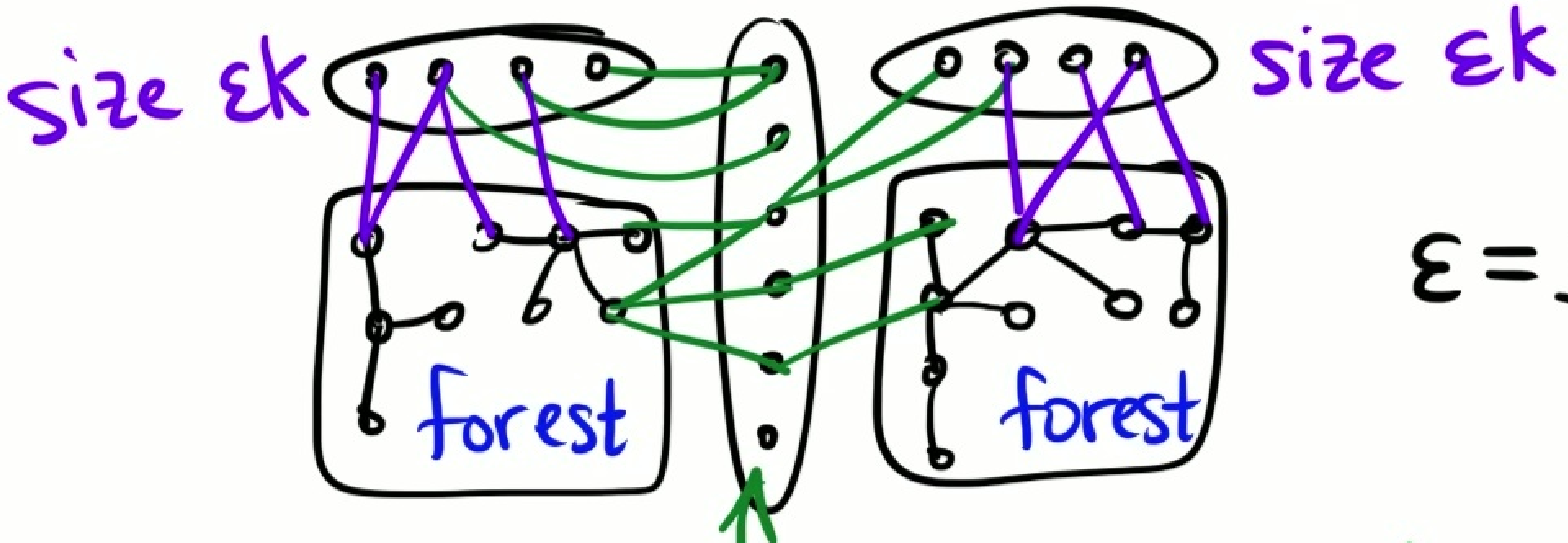
Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each



# Sparse case

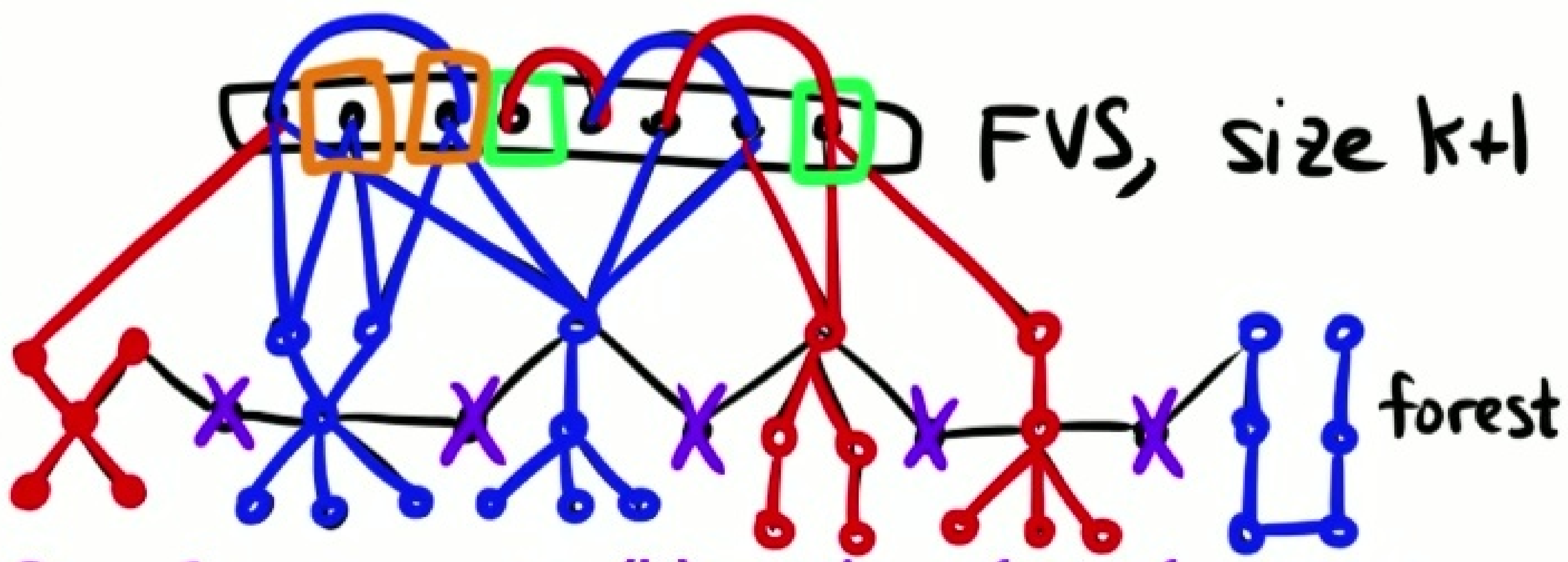
Lemma: Given a graph with $m \le 100k$, and given a FVS of size $k+1$, we can decompose the graph into:




Separator size $(1-2\varepsilon)k$, separates left and right

$\varepsilon = \Omega(1)$

## Proof of Lemma:

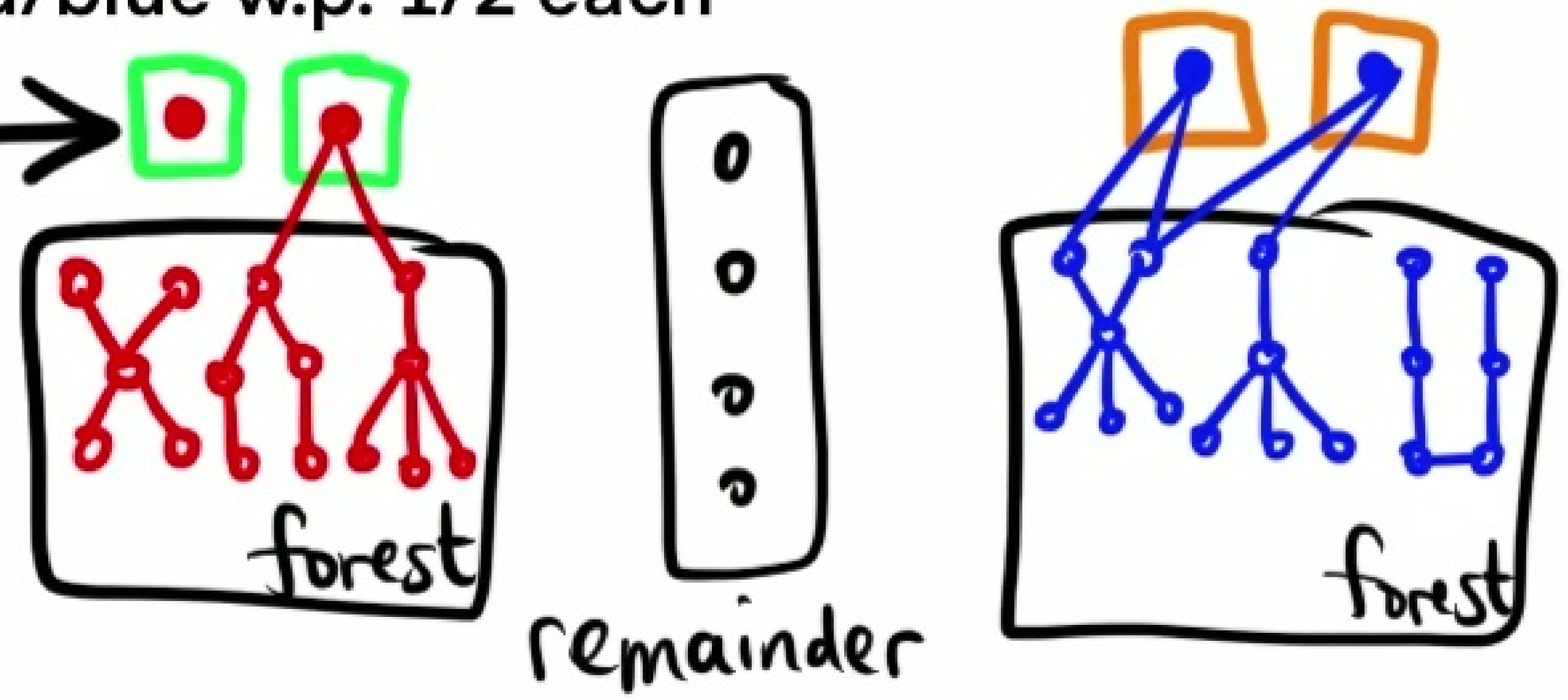



Step 1: remove $t=o(k)$ vertices from forest s.t. each remaining component is "small". These are added to separator

Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each

$$\Pr[\text{all incident edges red}] \ge 2^{-\deg(v)} \rightarrow$$

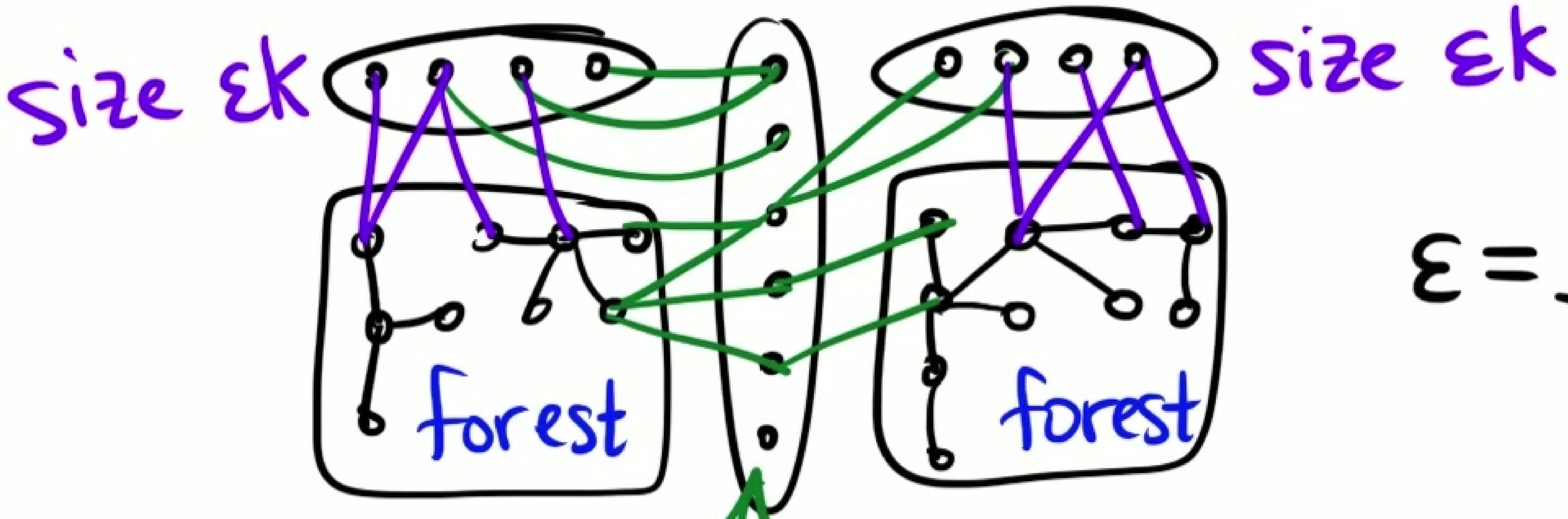



$$\mathbb{E}[\#\text{ such } v] \ge \sum_{v \in FVS} 2^{-\deg(v)} \underset{\text{Jensen}}{\ge} \underbrace{|FVS|}_{k+1} \cdot 2^{-(\overbrace{\text{avg deg}}^{\le \frac{2m}{k+1} \le 200})} \ge 2^{-200} k.$$

Chernoff bound (since each [tree] is "small"): #(green box, red dot) $\approx$ #(orange box, blue dot) $\approx \varepsilon k$ for some $\varepsilon \ge 2^{-200}$

# Sparse case

Lemma: Given a graph with $m \le 100k$, and given a FVS of size $k+1$, we can decompose the graph into:




$\varepsilon = \Omega(1)$

separator size $(1-2\varepsilon)k$, separates left and right

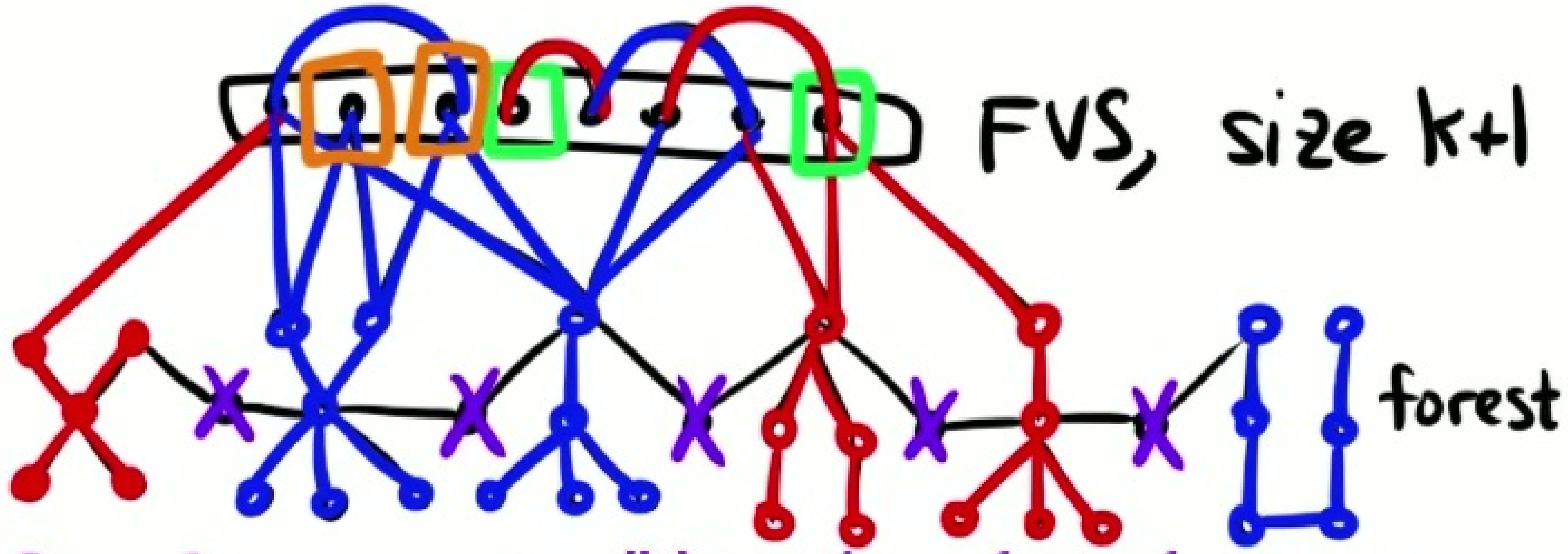
$\Pr[\text{all incident edges red}] \ge 2^{-\deg(v)}$

$\mathbb{E}[\#\text{ such } v] \ge \sum_{v \in FVS} 2^{-\deg(v)}$

Jensen: $\ge |FVS| \cdot 2^{-(\text{avg deg})} \ge 2^{-200} k.$ ($|FVS| = k+1$; avg deg $\le \frac{2m}{k+1} \le 200$)

Chernoff bound (since each is "small"): # ≈ # ≈ $\varepsilon k$ for some $\varepsilon \ge 2^{-200}$

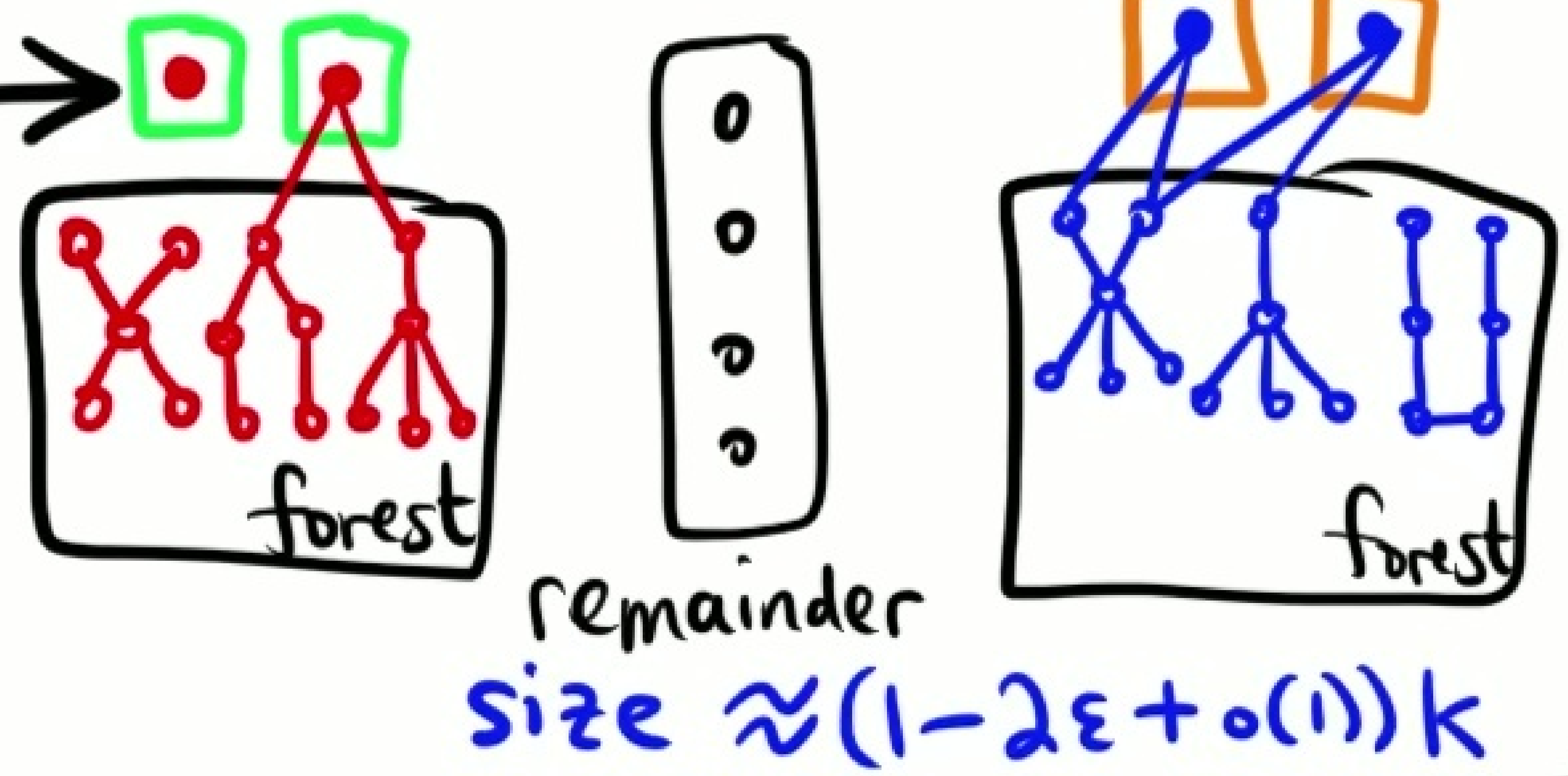
## Proof of Lemma:




Step 1: remove $t=o(k)$ vertices from forest s.t. each remaining component is "small". These are added to separator

Step 2: for each component, color red/blue w.p. 1/2 each

Step 3: Color each edge between FVS vertices red/blue w.p. 1/2 each



# Speedup: O*(2.7^k) time

- Tighten (deg(v)-3) analysis and open 3^tw algorithm [CNP+11]
- [CNP+11] actually solves a counting problem
  - special arithmetic structure: speed up via fast matrix multiplication

## Speedup: O*(2.7^k) time

- Tighten (deg(v)-3) analysis and open 3^tw algorithm [CNP+11]
- [CNP+11] actually solves a counting problem
  - special arithmetic structure: speed up via fast matrix multiplication

## Open problems

- Our main conceptual message: 3^k can be broken (randomized)
  - Faster deterministic algorithm? [BBG'00] is inherently randomized
- 2^k possible?
- SETH lower bound? No 1.00001^k lower bound known!