Advancements in Online Edge Coloring Algorithms

Ola Svensson, based on joint works with Joakim Blikstad, Radu Vintan, and David Wajc

Outline

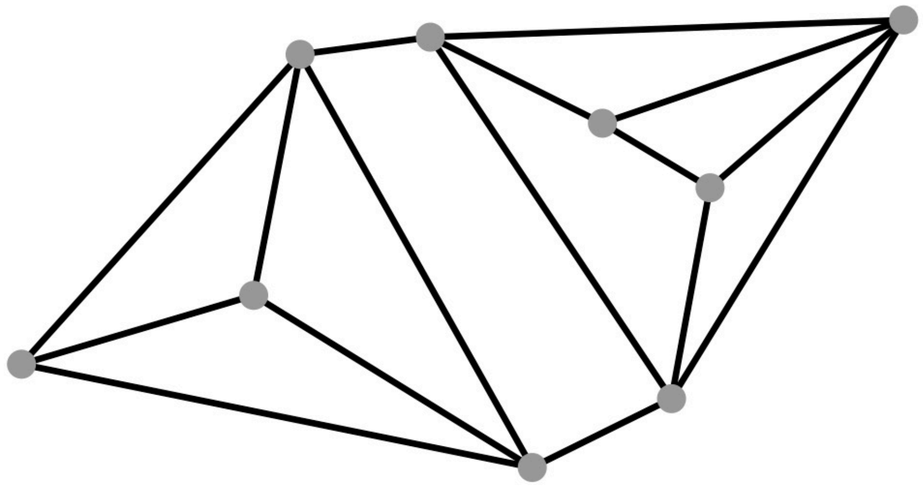
- Edge Coloring and Online Algorithms
- Greedy algorithm is optimal
- Improving upon Greedy
    - Fair Matchings
    - Embracing Correlations via Martingales

Edge Coloring

Given: Graph $G = (V, E)$

Goal: Color *edges* with few colors

Constraint: No two incident edges get the same color

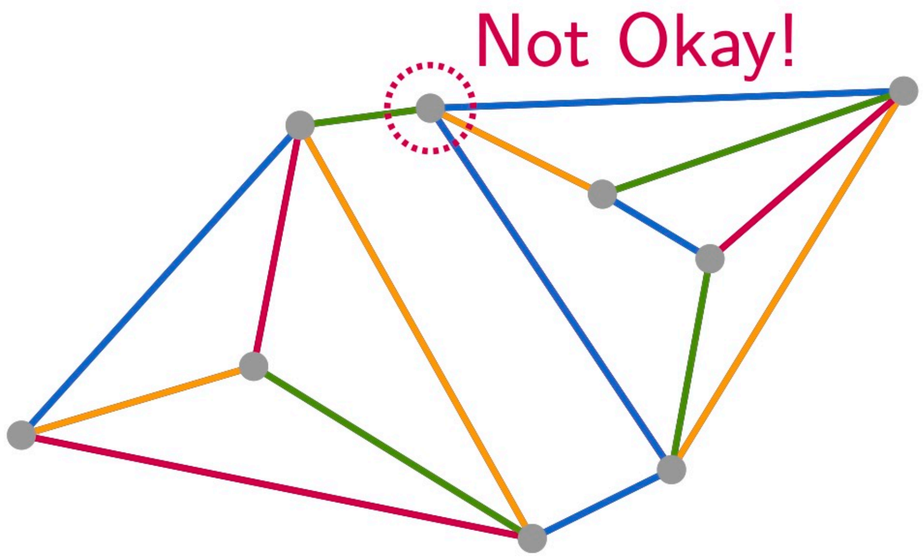


Edge Coloring

Given: Graph $G = (V, E)$

Goal: Color *edges* with few colors

Constraint: No two incident edges get the same color



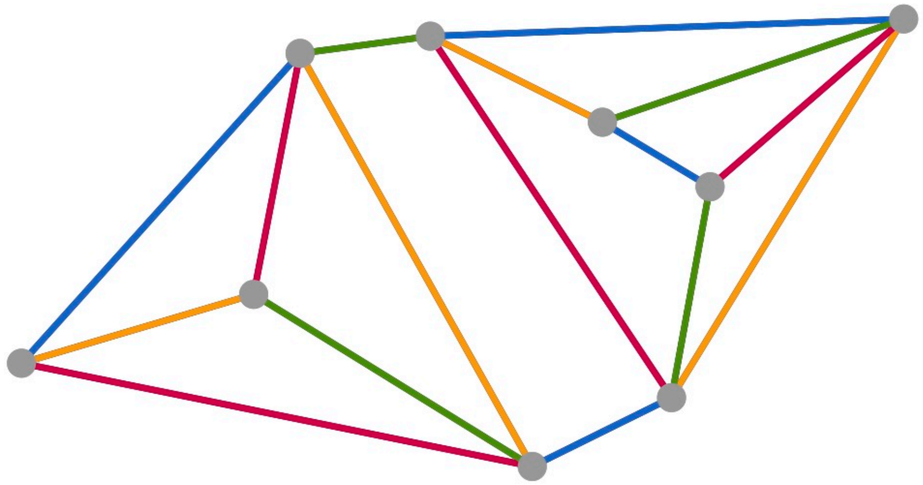
4 colors?

Edge Coloring

Given: Graph $G = (V, E)$

Goal: Color *edges* with few colors

Constraint: No two incident edges get the same color



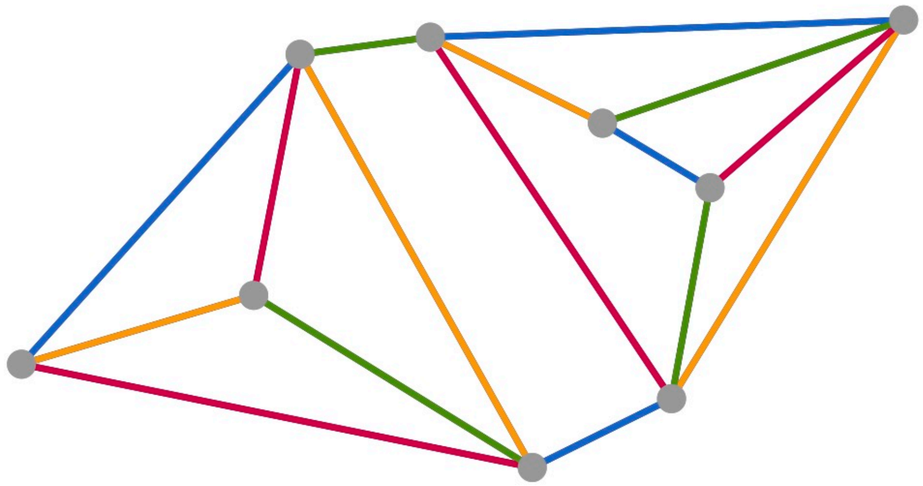
4 colors?

Edge Coloring

Given: Graph $G = (V, E)$

Goal: Color *edges* with few colors

Constraint: No two incident edges get the same color



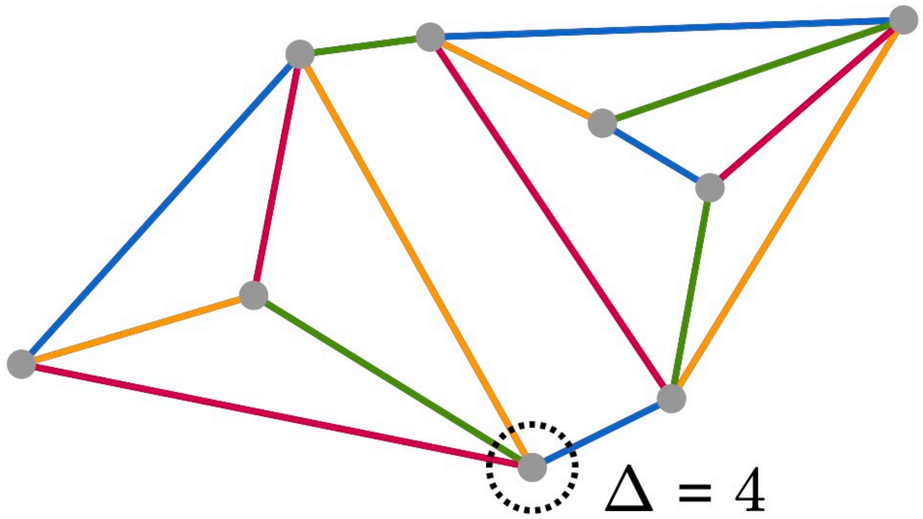
4 colors?
Optimal?

Edge Coloring

Given: Graph $G = (V, E)$

Goal: Color *edges* with few colors

Constraint: No two incident edges get the same color



4 colors?
Optimal?

$\Delta := \max_{v \in V} \deg(v)$

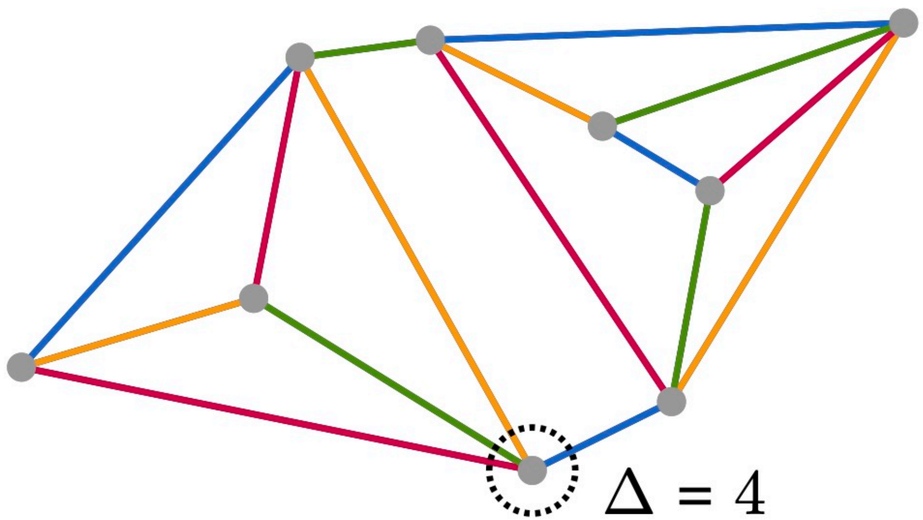
Claim: #Colors ≥ Δ

Edge Coloring

Given: Graph $G = (V, E)$

Goal: Color *edges* with few colors

Constraint: No two incident edges get the same color



4 colors?
Optimal?

$\Delta := \max_{v \in V} \deg(v)$

Claim: #Colors ≥ Δ

Theorem: #Colors ≤ Δ + 1

[Vizing 1964]

Edge Coloring Algorithms

Edge Coloring Algorithms

- Many algorithms compute $(\Delta + 1)$-edge colorings, now near-linear time
  [Vizing'64, Gabow/Nishizeki/Kariv/Leven/Osmau'85,…, AssadiBehnezhadBhattacharyaCostaSolomonZhang'25]

Edge Coloring Algorithms

- Many algorithms compute $(\Delta + 1)$-edge colorings, now near-linear time
  [Vizing'64, Gabow/Nishizeki/Kariv/Leven/Osmau'85,…, AssadiBehnezhadBhattacharyaCostaSolomonZhang'25]

- NP-hard to Δ-edge color
  [Holyer'81]

Edge Coloring Algorithms

- Many algorithms compute $(\Delta + 1)$-edge colorings, now near-linear time
  [Vizing'64, Gabow/Nishizeki/Kariv/Leven/Osmau'85,…, AssadiBehnezhadBhattacharyaCostaSolomonZhang'25]

- NP-hard to Δ-edge color
  [Holyer'81]

- Many algorithms compute Δ-edge-colors in bipartite graphs
  [Cole/Hopcroft'82,Cole/Ost/Schirra'01,Alon'03,Goel/Kapralov/Khanna'13,…]

Edge Coloring Algorithms

- Many algorithms compute $(\Delta + 1)$-edge colorings, now near-linear time
  [Vizing'64, Gabow/Nishizeki/Kariv/Leven/Osmau'85,…, AssadiBehnezhadBhattacharyaCostaSolomonZhang'25]

- NP-hard to Δ-edge color
  [Holyer'81]

- Many algorithms compute Δ-edge-colors in bipartite graphs
  [Cole/Hopcroft'82,Cole/Ost/Schirra'01,Alon'03,Goel/Kapralov/Khanna'13,…]

- Studied in various computational models

  Distributed [PanconesiSrinivasan'97,DubhashiGrablePanconessi'98,...]

  PRAM [LevPippengerValiant'81,...]

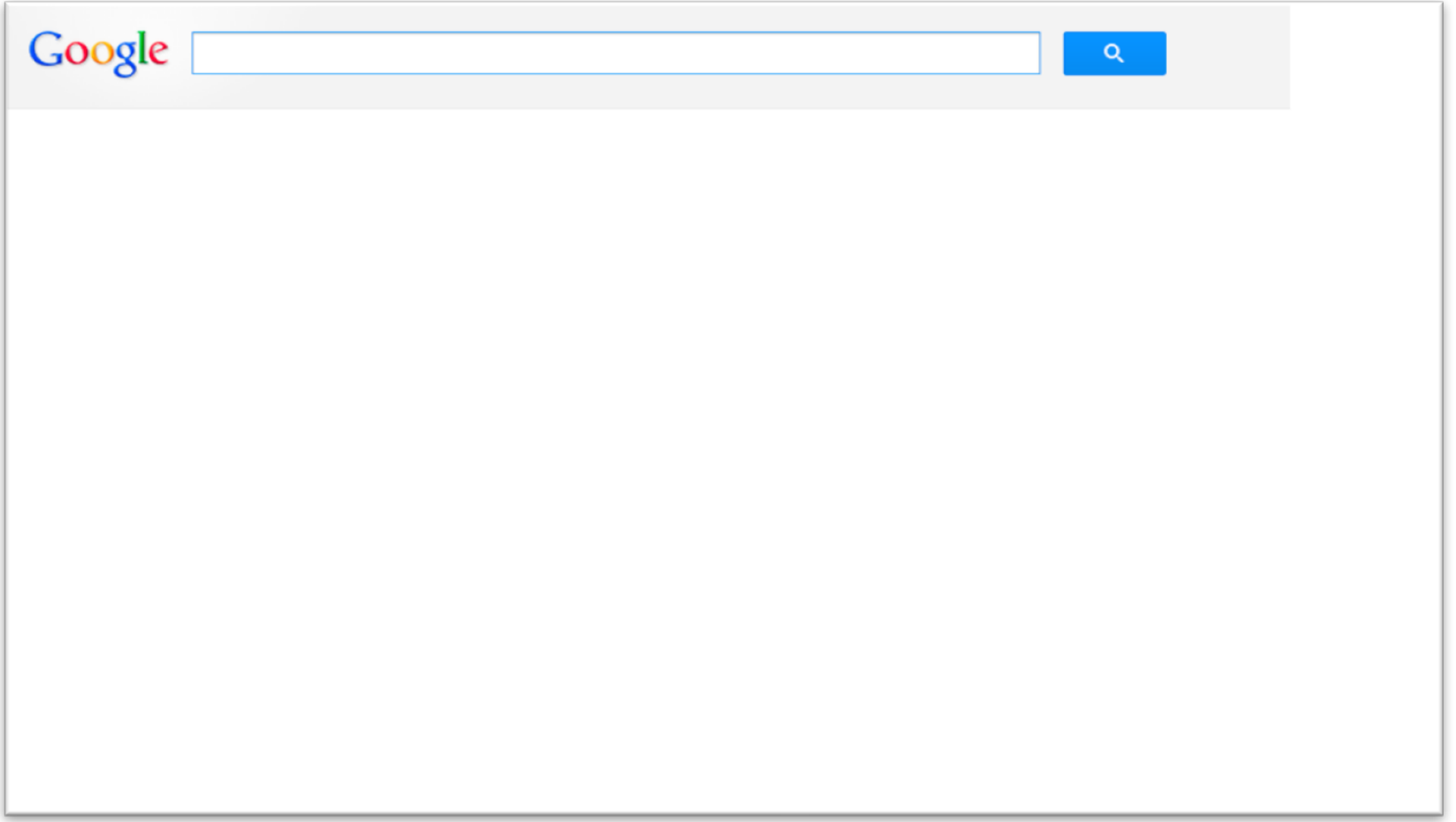
  NC & RNC [KarloffShmoys'87, MotwaniNaorNaor'94,...]

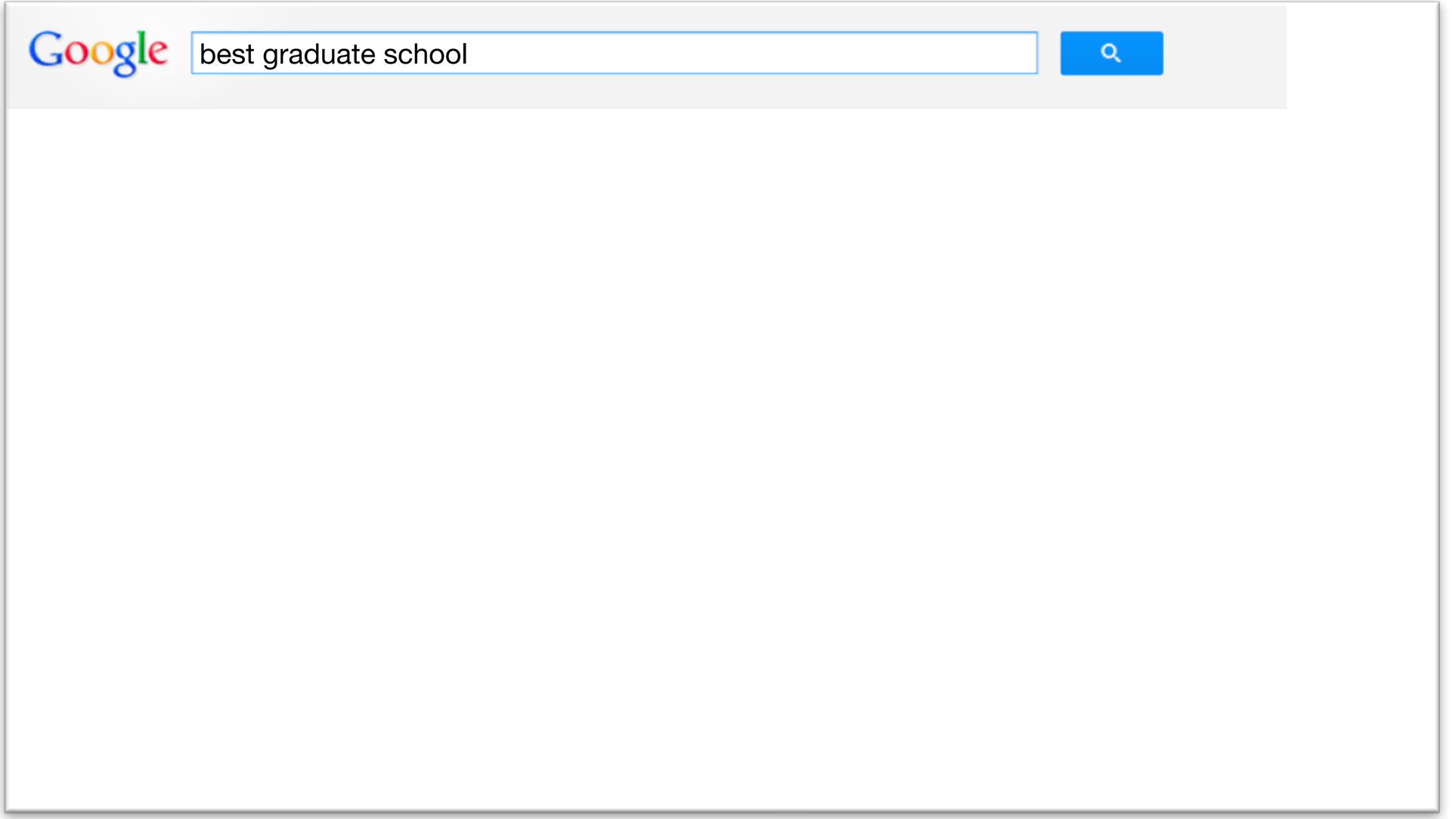
  Dynamic [Bhattacharya/Chakrabarty/Henzinger/Nanongkai'18,...]

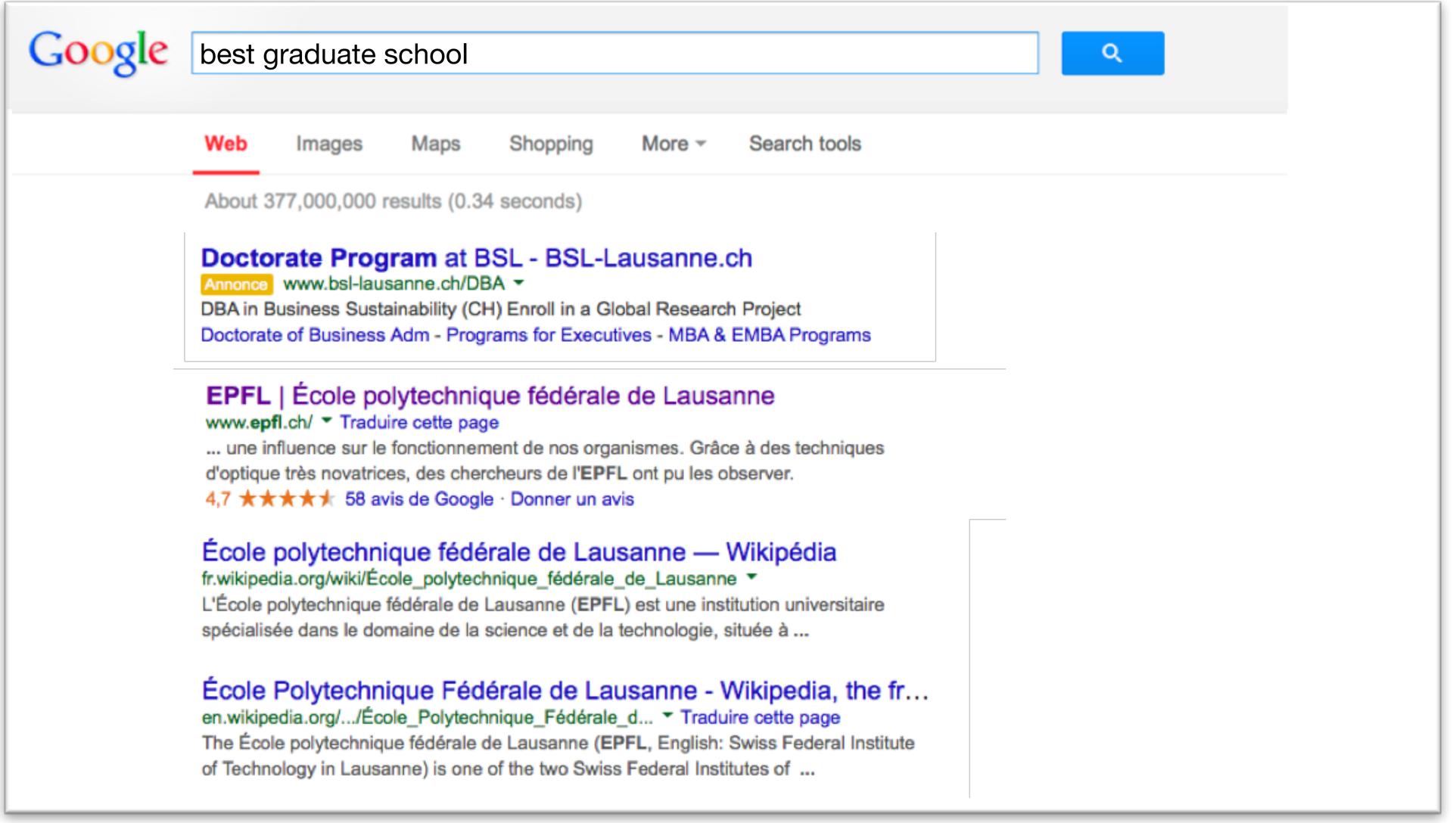
  …

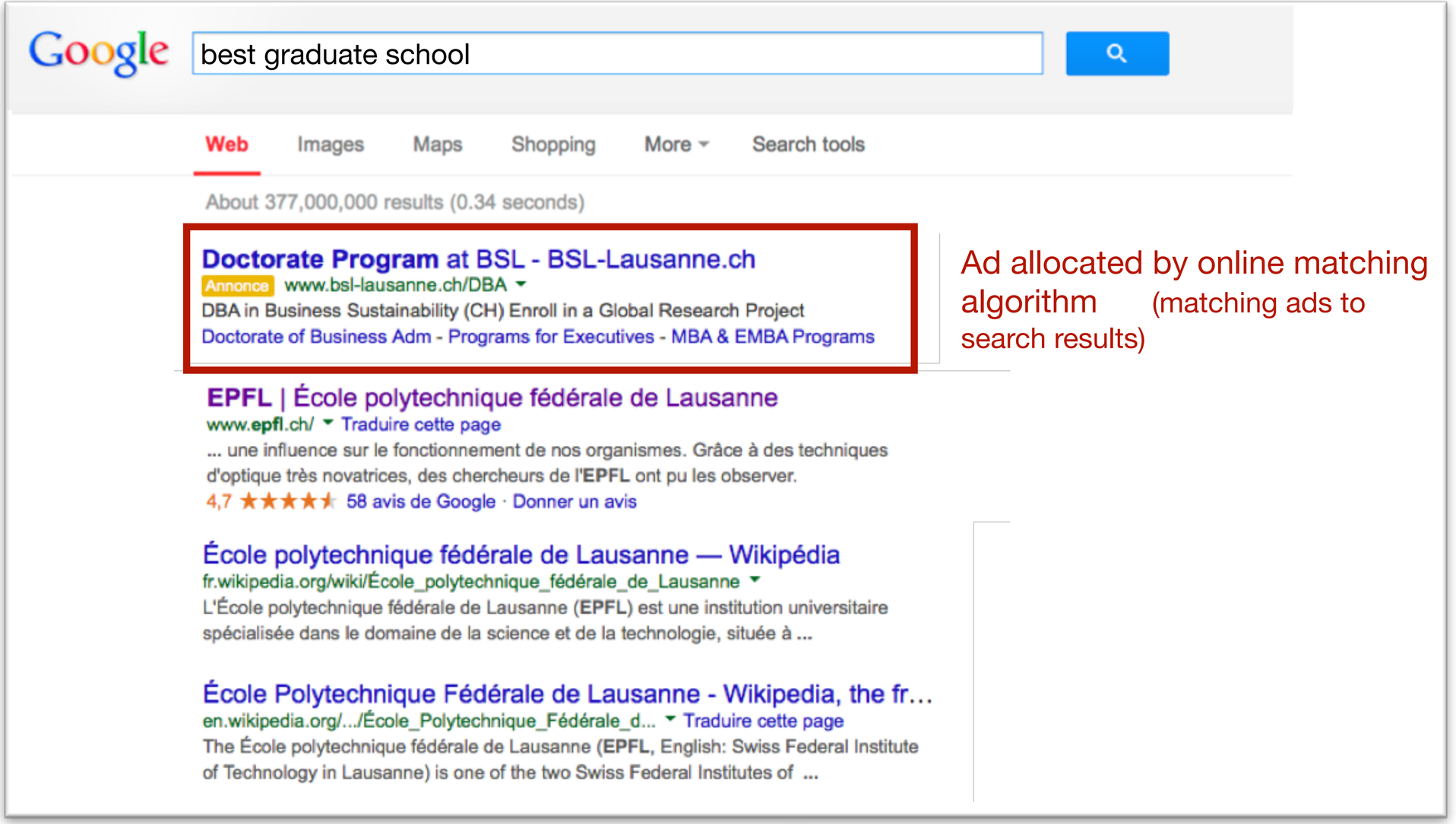
Rent or buy?

Today: Online Algorithms


Google


Google
best graduate school


Google
best graduate school
Web
Images
Maps
Shopping
More
Search tools
About 377,000,000 results (0.34 seconds)
Doctorate Program at BSL - BSL-Lausanne.ch
Annonce www.bsl-lausanne.ch/DBA
DBA in Business Sustainability (CH) Enroll in a Global Research Project
Doctorate of Business Adm - Programs for Executives - MBA & EMBA Programs
EPFL | École polytechnique fédérale de Lausanne
www.epfl.ch/ Traduire cette page
... une influence sur le fonctionnement de nos organismes. Grâce à des techniques d'optique très novatrices, des chercheurs de l'EPFL ont pu les observer.
4,7 58 avis de Google · Donner un avis
École polytechnique fédérale de Lausanne — Wikipédia
fr.wikipedia.org/wiki/École_polytechnique_fédérale_de_Lausanne
L'École polytechnique fédérale de Lausanne (EPFL) est une institution universitaire spécialisée dans le domaine de la science et de la technologie, située à ...
École Polytechnique Fédérale de Lausanne - Wikipedia, the fr...
en.wikipedia.org/.../École_Polytechnique_Fédérale_d... Traduire cette page
The École polytechnique fédérale de Lausanne (EPFL, English: Swiss Federal Institute of Technology in Lausanne) is one of the two Swiss Federal Institutes of ...



Ad allocated by online matching algorithm (matching ads to search results)

Online Edge Coloring

Online: Graph revealed over time. Max degree Δ known

Task: Color edge irrevocably when revealed

Online Edge Coloring

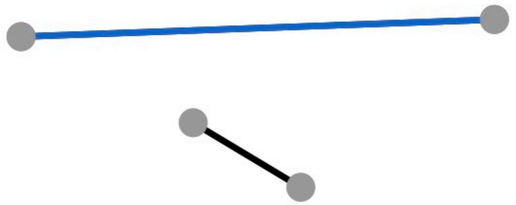
Online: Graph revealed over time. Max degree Δ known

Task: Color edge irrevocably when revealed

Online Edge Coloring

Online: Graph revealed over time. Max degree Δ known

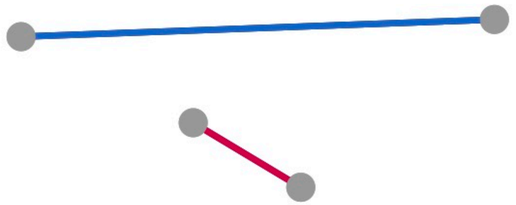
Task: Color edge irrevocably when revealed



Online Edge Coloring

Online: Graph revealed over time. Max degree Δ known

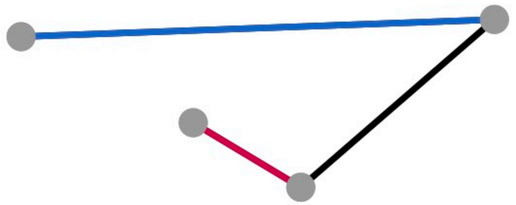
Task: Color edge irrevocably when revealed



Online Edge Coloring

Online: Graph revealed over time. Max degree Δ known

Task: Color edge irrevocably when revealed



Online Edge Coloring

Online: Graph revealed over time. Max degree Δ known

Task: Color edge irrevocably when revealed

Online Edge Coloring

Online: Graph revealed over time. Max degree Δ known

Task: Color edge irrevocably when revealed

Variant:

Edge arrivals

Adversarial order

General graphs

Online Edge Coloring

Online: Graph revealed over time. Max degree Δ known

Task: Color edge irrevocably when revealed

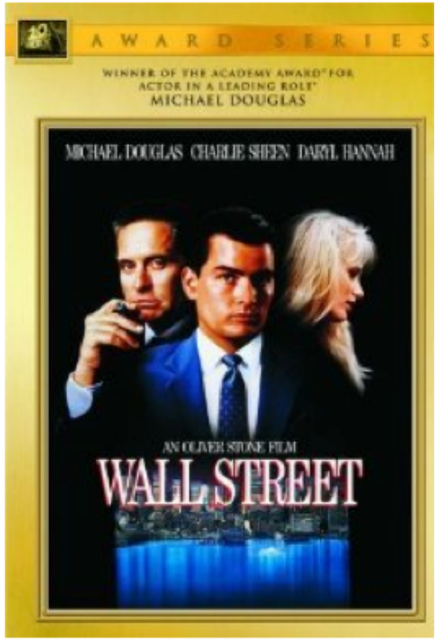
Variant:

Edge arrivals

Adversarial order

General graphs

How many colors do we need? Still $\approx \Delta$?



Greed is good. Greed is right. Greed works. Greed clarifies, cuts through and captures the essence of the evolutionary spirit.

- Gordon Gekko

Warm-up: Greedy Algorithm

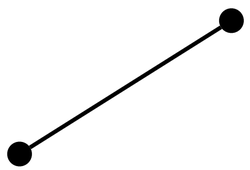
Greedy: Color edge with “lowest” available color

Colors = {1, 2, 3, …}

Warm-up: Greedy Algorithm

Greedy: Color edge with “lowest” available color

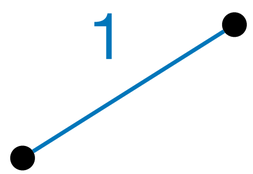
Colors = {1, 2, 3, …}



Warm-up: Greedy Algorithm

Greedy: Color edge with “lowest” available color

Colors = {1, 2, 3, …}



Warm-up: Greedy Algorithm

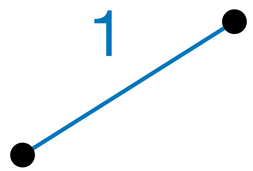
Greedy: Color edge with “lowest” available color

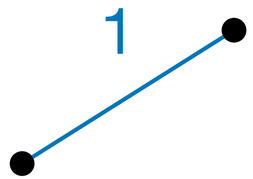
Colors = {1, 2, 3, …}

Warm-up: Greedy Algorithm

Greedy: Color edge with “lowest” available color

Colors = {1, 2, 3, …}

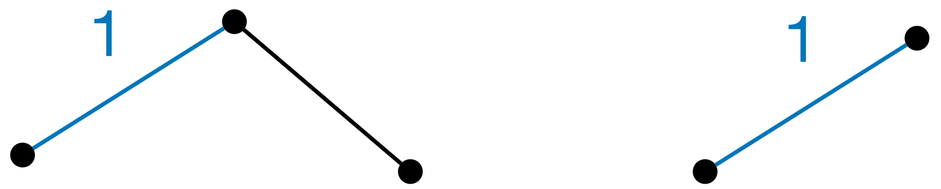




Warm-up: Greedy Algorithm

Greedy: Color edge with “lowest” available color

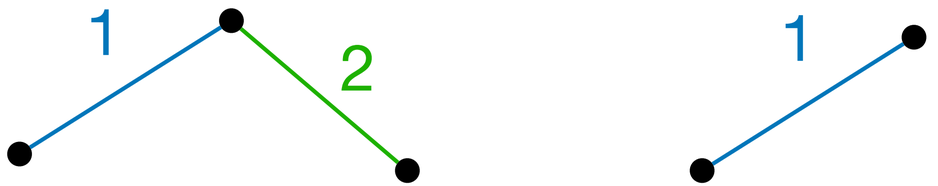
Colors = {1, 2, 3, …}



Warm-up: Greedy Algorithm

Greedy: Color edge with “lowest” available color

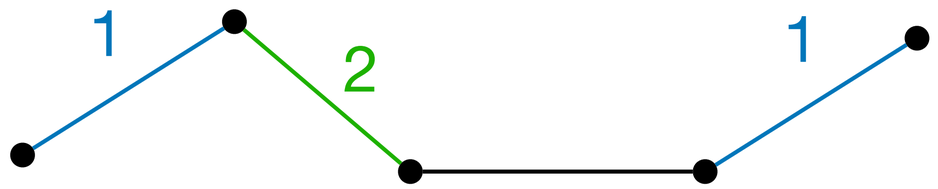
Colors = {1, 2, 3, …}



Warm-up: Greedy Algorithm

Greedy: Color edge with “lowest” available color

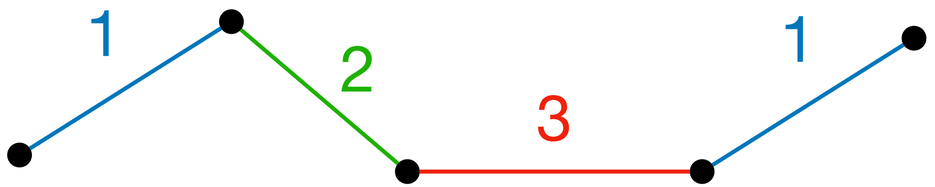
Colors = {1, 2, 3, …}



Warm-up: Greedy Algorithm

Greedy: Color edge with “lowest” available color

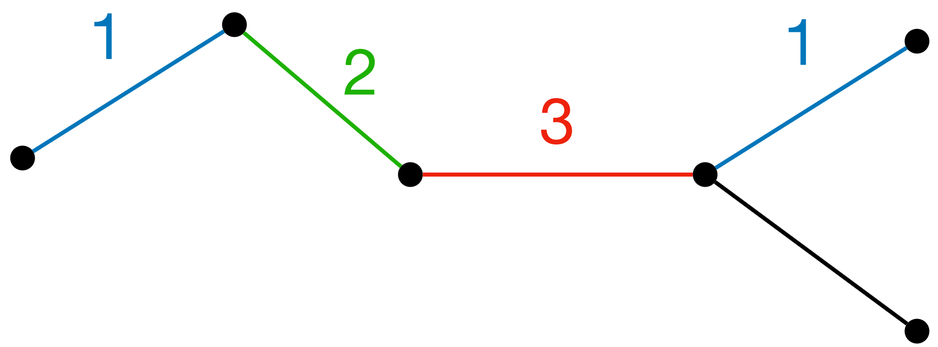
Colors = {1, 2, 3, …}



Warm-up: Greedy Algorithm

Greedy: Color edge with “lowest” available color

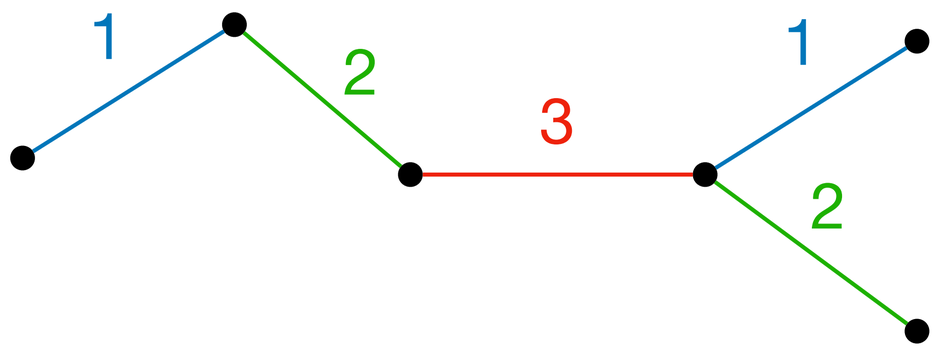
Colors = {1, 2, 3, …}



Warm-up: Greedy Algorithm

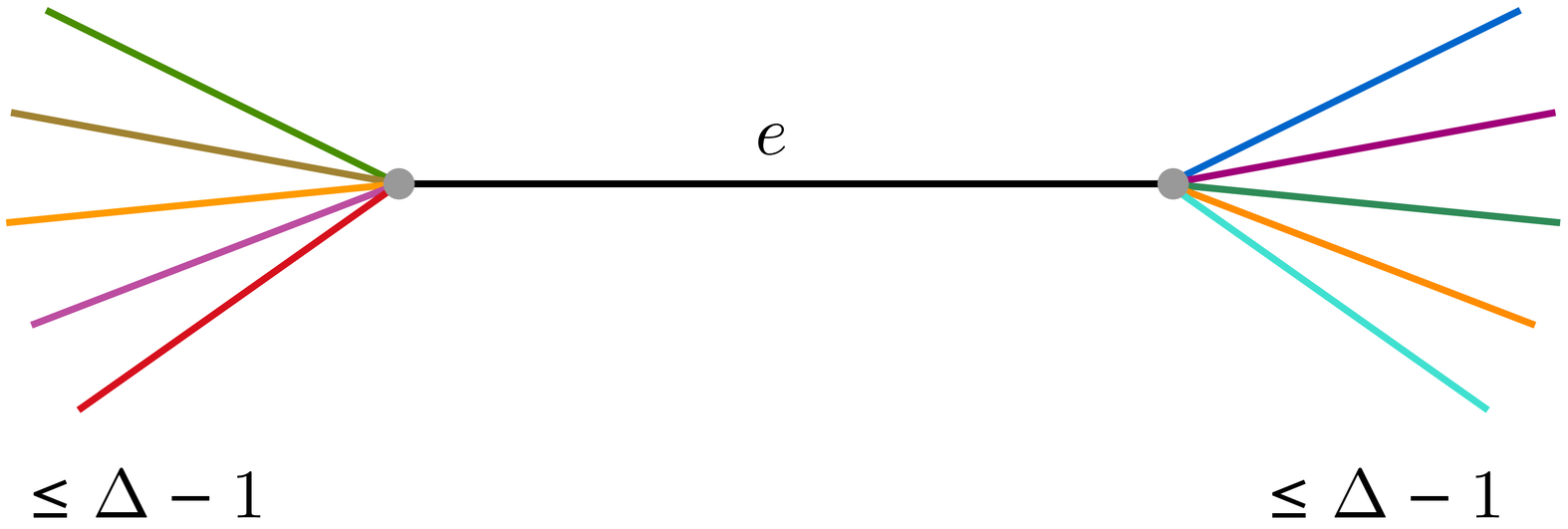
Greedy: Color edge with “lowest” available color

Colors = {1, 2, 3, …}



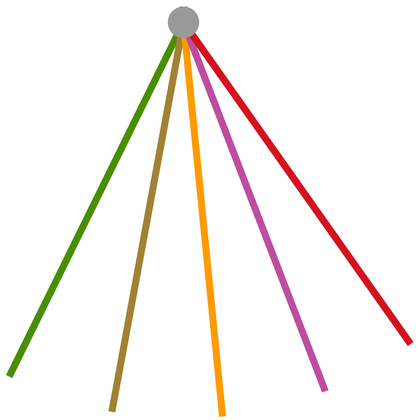
Claim: Greedy uses at most $2\Delta - 1$ colors

Claim: Greedy uses at most $2\Delta - 1$ colors



Claim: Greedy may use $2\Delta - 1$ colors

Claim: Greedy may use $2\Delta - 1$ colors

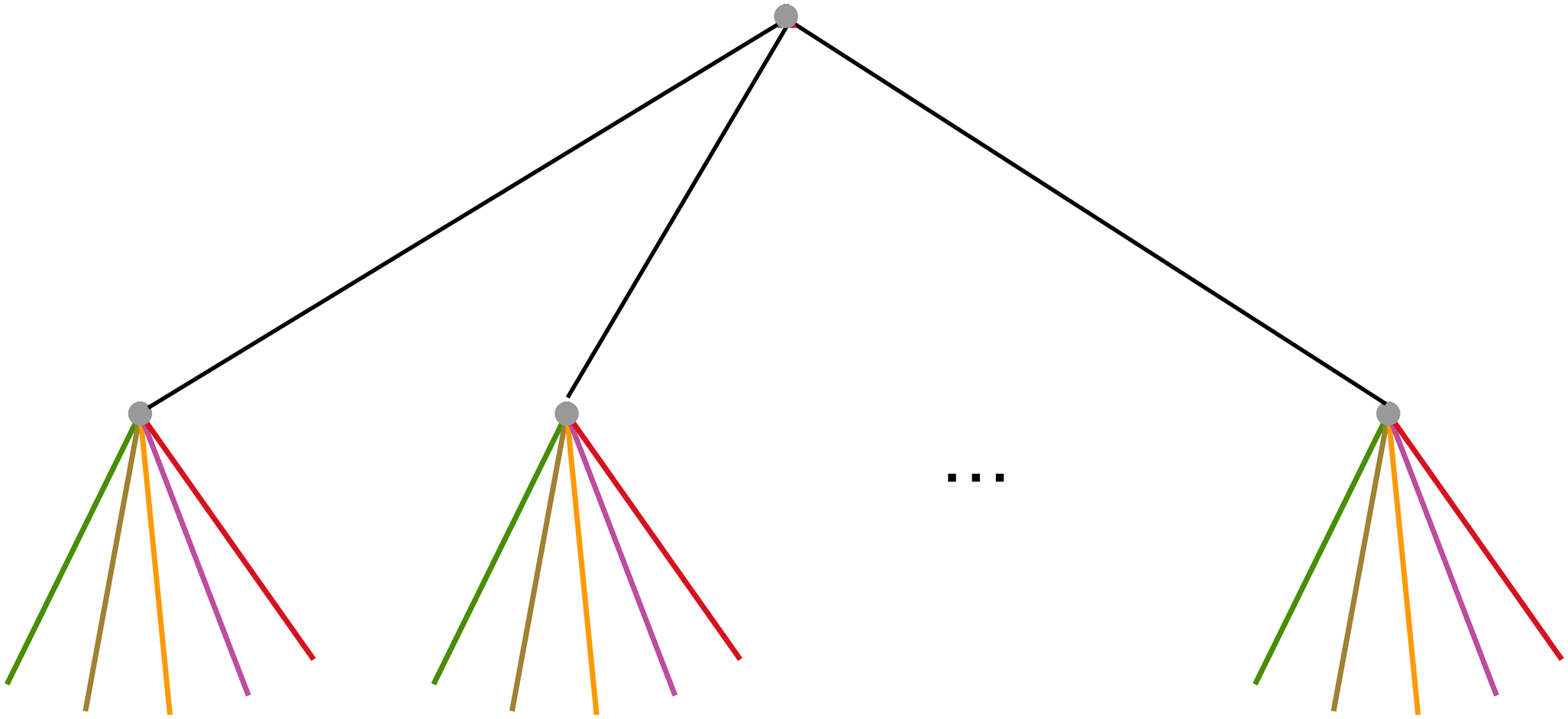


Claim: Greedy may use $2\Delta - 1$ colors

Claim: Greedy may use $2\Delta - 1$ colors

Δ identically colored stars with $\Delta - 1$ pedals

Claim: Greedy may use $2\Delta - 1$ colors



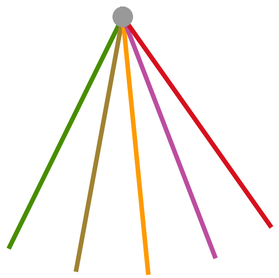
Δ identically colored stars with $\Delta - 1$ pedals

What about other algorithms?

- May not color the stars equally
- But given enough stars, then Δ of them must be colored the same

What about other algorithms?

- May not color the stars equally
- But given enough stars, then Δ of them must be colored the same

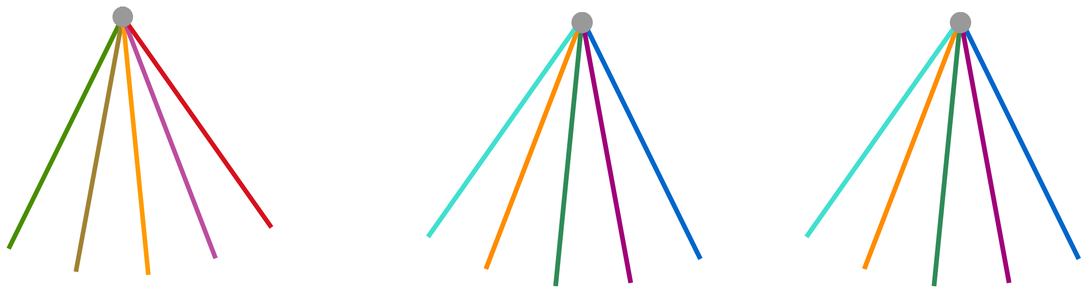


What about other algorithms?

- May not color the stars equally
- But given enough stars, then Δ of them must be colored the same

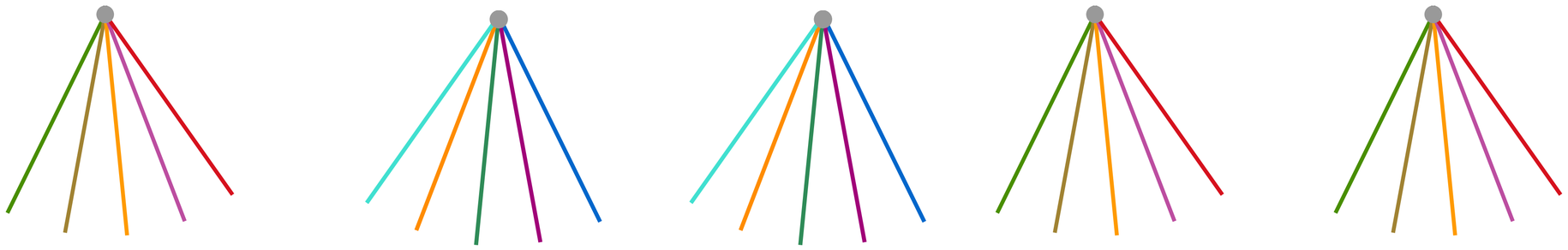
What about other algorithms?

- May not color the stars equally
- But given enough stars, then Δ of them must be colored the same



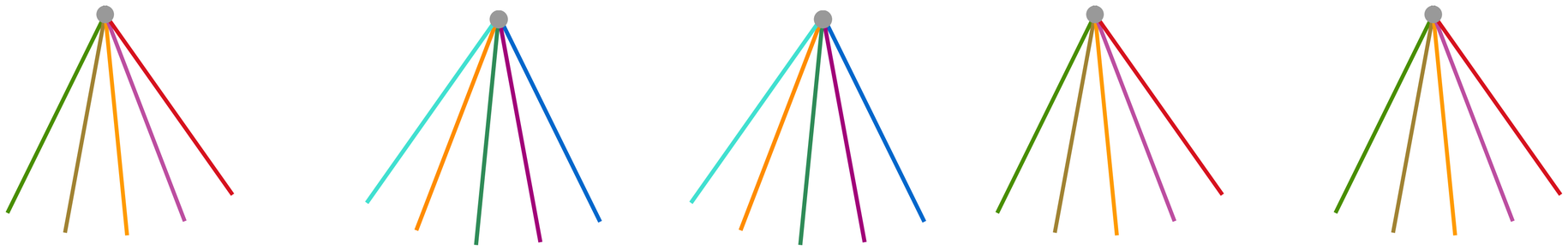
What about other algorithms?

- May not color the stars equally
- But given enough stars, then Δ of them must be colored the same



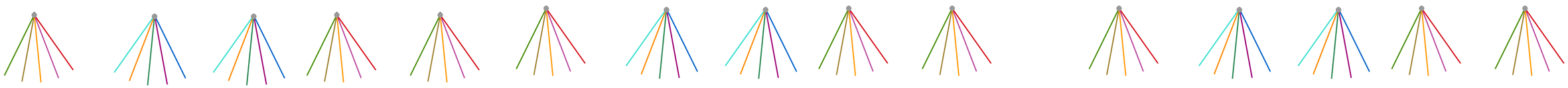
What about other algorithms?

- May not color the stars equally
- But given **enough** stars, then Δ of them must be colored the same



What about other algorithms?

- May not color the stars equally
- But given **enough** stars, then Δ of them must be colored the same

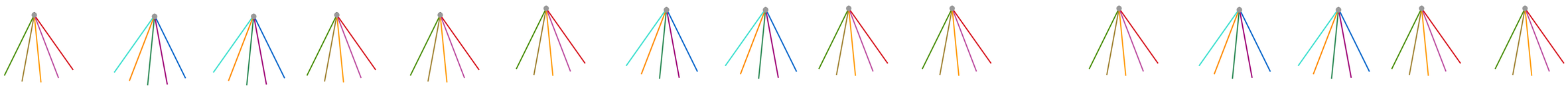


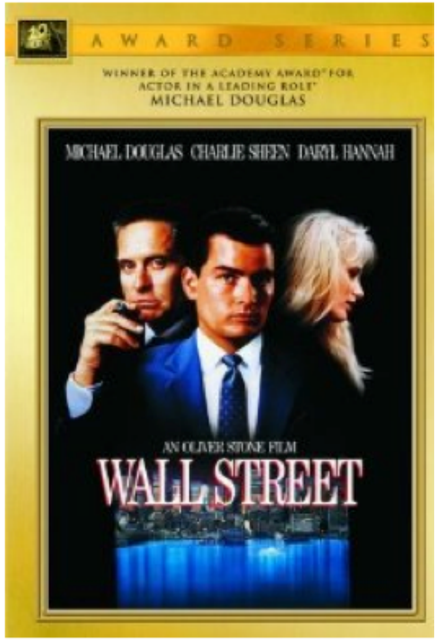
What about other algorithms?

- May not color the stars equally
- But given **enough** stars, then Δ of them must be colored the same

[Bar-Noy/Motwani/Naor 1992]

No algorithm can do better if $\Delta = \begin{cases} o(\log n) & \text{if deterministic} \\ o(\sqrt{\log n}) & \text{if randomized} \end{cases}$





Greed is good. Greed is right. Greed works. Greed clarifies, cuts through and captures the essence of the evolutionary spirit.

- Gordon Gekko

Greed is good. Greed is right. Greed works. Greed clarifies, cuts through and captures the essence of the evolutionary spirit.

- Gordon Gekko

Conjecture: [Bar-Noy/Motwani/Naor 1992]

$(1 + o(1))\Delta$-colors sufficient when $\Delta = \omega(\log n)$

Conjecture: [Bar-Noy/Motwani/Naor 1992]

$(1 + o(1))\Delta$-colors sufficient when $\Delta = \omega(\log n)$

- **Random order** edge arrivals:
  - [Aggarwal/Motwani/Shah/Zhu'03]: $\approx \Delta$ -coloring if $\boldsymbol{\Delta = \omega(n^2)}$ **(multigraphs)**
  - [Bahmani/Mehta/Motwani'10]: $\approx 1.27\Delta$ -coloring **general graphs**
  - [Bhattacharya/Grandoni/Wajc'21]: $\approx \Delta$ -coloring **general graphs**

Conjecture: [Bar-Noy/Motwani/Naor 1992]

$(1 + o(1))\Delta$-colors sufficient when $\Delta = \omega(\log n)$

- **Random order** edge arrivals:
  - [Aggarwal/Motwani/Shah/Zhu'03]: $\approx \Delta$ -coloring if $\boldsymbol{\Delta = \omega(n^2)}$ **(multigraphs)**
  - [Bahmani/Mehta/Motwani'10]: $\approx 1.27\Delta$ -coloring **general graphs**
  - [Bhattacharya/Grandoni/Wajc'21]: $\approx \Delta$ -coloring **general graphs**

- Adversarial vertex arrivals:
  - [Cohen/Peng/Wajc'19] (simplified [Blikstad/**S.**/Vintan/Wajc '24]):
    $\approx \Delta$ -coloring **bipartite graphs**

    For unknown Δ: $\approx \frac{e}{e-1}\Delta$ -coloring **bipartite graphs (optimal)**
  - [Saberi/Wajc'21]: $\approx 1.9\Delta$ -coloring **general graphs**

Conjecture: [Bar-Noy/Motwani/Naor 1992]

$(1 + o(1))\Delta$-colors sufficient when $\Delta = \omega(\log n)$

- **Random order** edge arrivals:
  - [Aggarwal/Motwani/Shah/Zhu'03]: $\approx \Delta$ -coloring if $\boldsymbol{\Delta = \omega(n^2)}$ **(multigraphs)**
  - [Bahmani/Mehta/Motwani'10]: $\approx 1.27\Delta$ -coloring **general graphs**
  - [Bhattacharya/Grandoni/Wajc'21]: $\approx \Delta$ -coloring **general graphs**
- Adversarial vertex arrivals:
  - [Cohen/Peng/Wajc'19] (simplified [Blikstad/**S.**/Vintan/Wajc '24]):
    $\approx \Delta$ -coloring **bipartite graphs**

    For unknown Δ: $\approx \frac{e}{e-1}\Delta$ -coloring **bipartite graphs (optimal)**
  - [Saberi/Wajc'21]: $\approx 1.9\Delta$ -coloring **general graphs**
- Adversarial edge arrivals
  - [Kulkarni/Liu/Sah/Sawhney/Tarnawski'22]:
    $\approx \frac{e}{e-1}\Delta$ -coloring **bipartite graphs**

Our result

Conjecture:

$(1 + o(1))\Delta$-colors sufficient when $\Delta = \omega(\log n)$

Our result

~~**Conjecture:**~~ **Theorem:**

$(1 + o(1))\Delta$-colors sufficient when $\Delta = \omega(\log n)$

Fresh extension of techniques

[Bar-Noy/Motwani/Naor 1992]

No algorithm can do better if $\Delta = \begin{cases} o(\log n) & \text{if deterministic} \\ o(\sqrt{\log n}) & \text{if randomized} \end{cases}$

Fresh extension of techniques

[Bar-Noy/Motwani/Naor 1992]

No algorithm can do better if $\Delta = \begin{cases} o(\log n) & \text{if deterministic} \\ o(\sqrt{\log n}) & \text{if randomized} \end{cases}$

[BSVW'25]

Theorem:

$(1 + o(1))\Delta$-colors sufficient for when $\Delta = \begin{cases} \omega(\log n) & \text{if deterministic} \\ \omega(\sqrt{\log n}) & \text{if randomized} \end{cases}$

Outline of ideas

- Edge Coloring $\Longleftrightarrow$ Fair Matchings
- Online Fair Matching Algorithm
  - Previous attempts to control correlations
  - Embracing correlations via Martingales

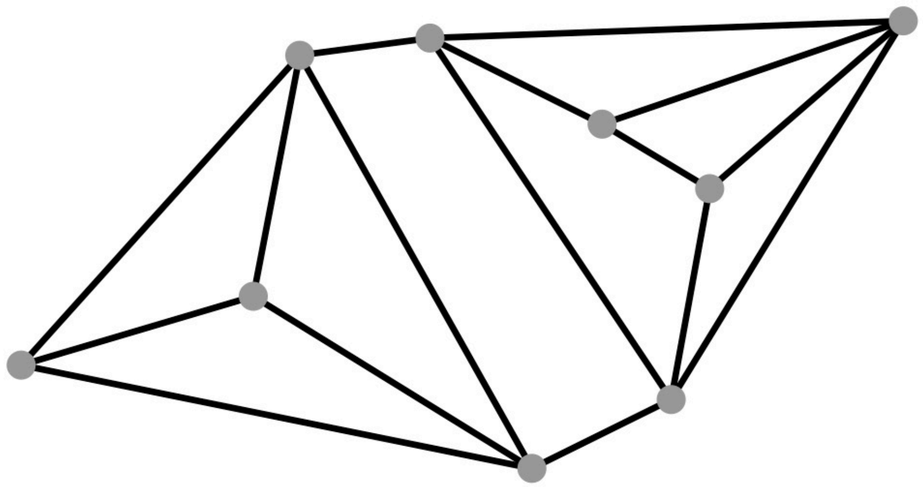
Edge Coloring $\Longleftrightarrow$ Fair Matchings

Fair Matching Problem

Given: Graph $G = (V, E)$

Goal: Find a matching M

α-Fairness: $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for every edge $e \in E$



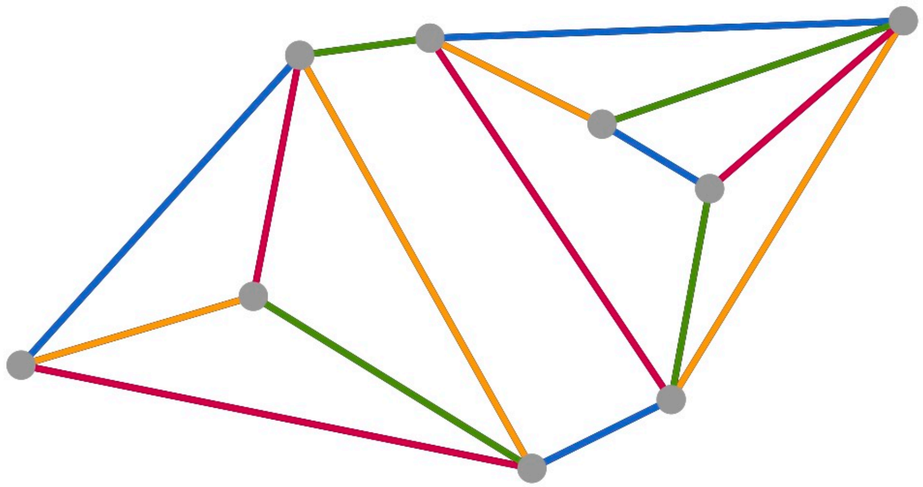
Fair Matching Problem

Given: Graph $G = (V, E)$

Goal: Find a matching M

α-Fairness: $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for every edge $e \in E$

Claim: $\alpha\Delta$-edge-coloring algorithm $\Longrightarrow$ α-fair matching algorithm



Fair Matching Problem

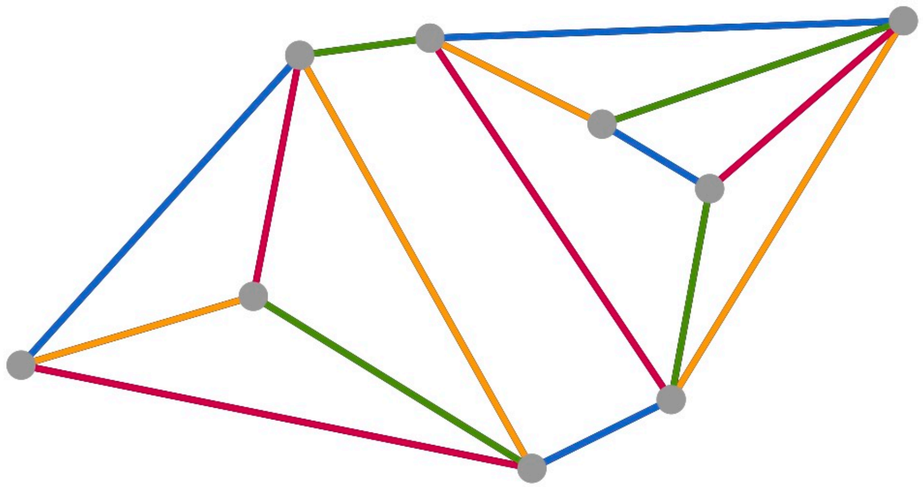
Given: Graph $G = (V, E)$

Goal: Find a matching M

α-Fairness: $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for every edge $e \in E$

Claim: $\alpha\Delta$-edge-coloring algorithm $\implies$ α-fair matching algorithm

Proof: Pick random color as matching



Fair Matching Problem

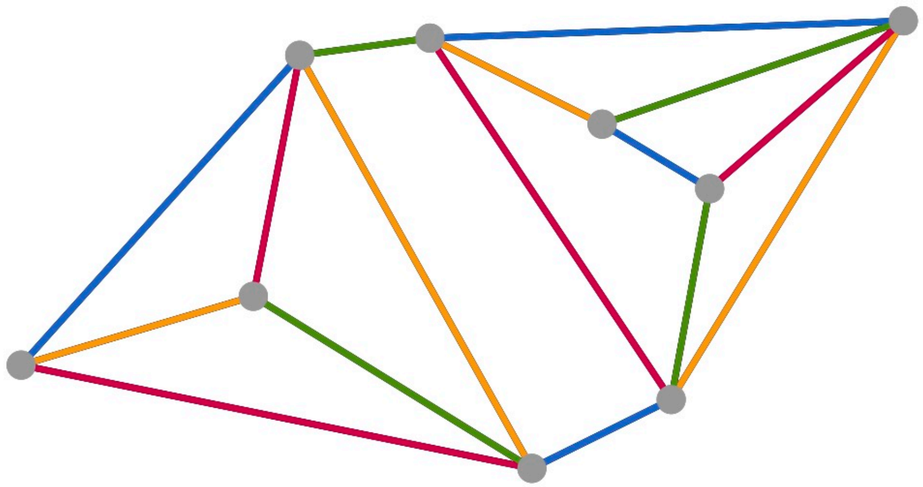
Given: Graph $G = (V, E)$

Goal: Find a matching M

α-Fairness: $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for every edge $e \in E$

Claim: $\alpha\Delta$-edge-coloring algorithm $\implies$ α-fair matching algorithm

Proof: Pick random color as matching



Lemma: α-fair matching $\implies$
$(1 + o(1))\alpha\Delta$-edge-coloring

[Cohen/Peng/Wajc'19]

Fair Matching Problem

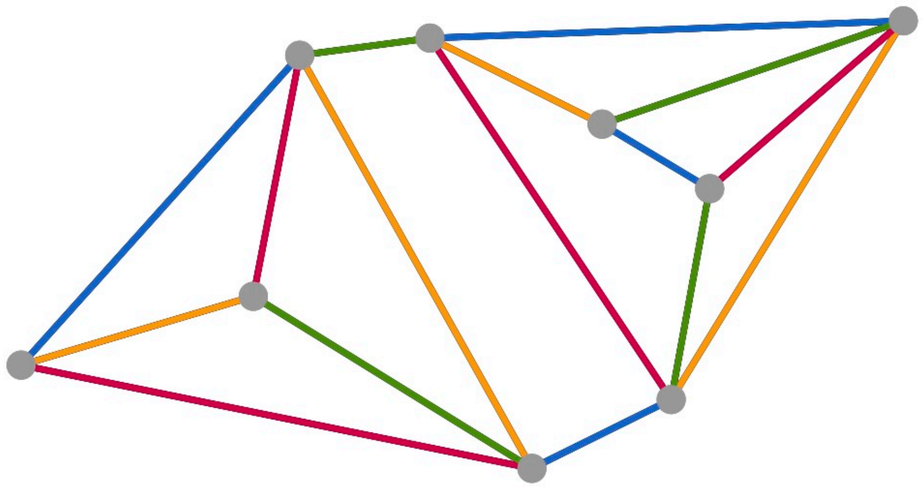
Given: Graph $G = (V, E)$

Goal: Find a matching M

α-Fairness: $\Pr[e \in M] \geq \frac{1}{\alpha\Delta}$ for every edge $e \in E$

Claim: $\alpha\Delta$-edge-coloring algorithm $\implies$ α-fair matching algorithm

Proof: Pick random color as matching



Lemma: α-fair matching $\implies$
$(1 + o(1))\alpha\Delta$-edge-coloring

[Cohen/Peng/Wajc'19]

Note: the above results are for online (similar reductions used by e.g. Kahn for list coloring)

New Objective

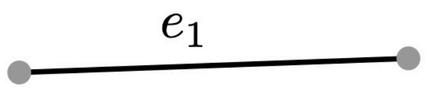
Find $(1 + o(1))$-fair matching, i.e., match each edge with probability

$$\Pr[e \in M] \approx \frac{1}{\Delta}$$

Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

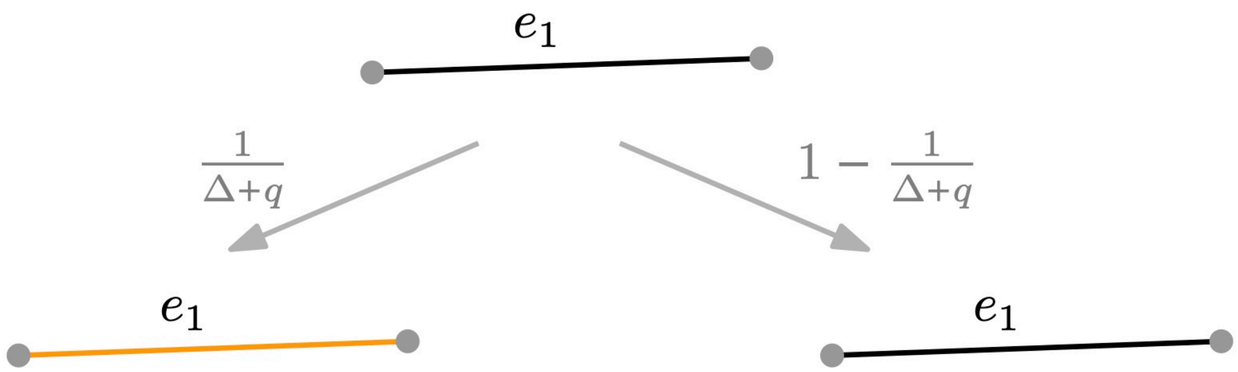
$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

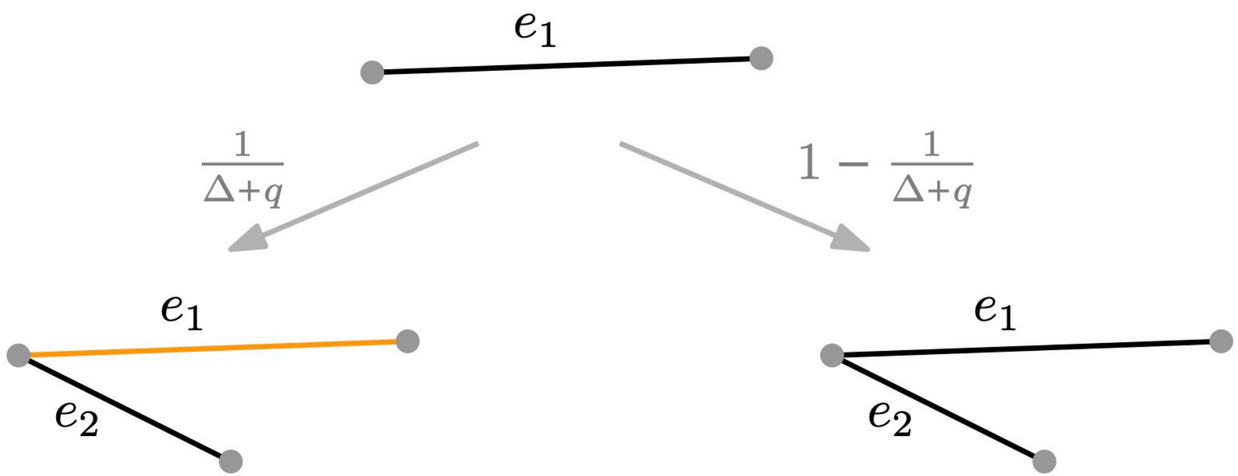
$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

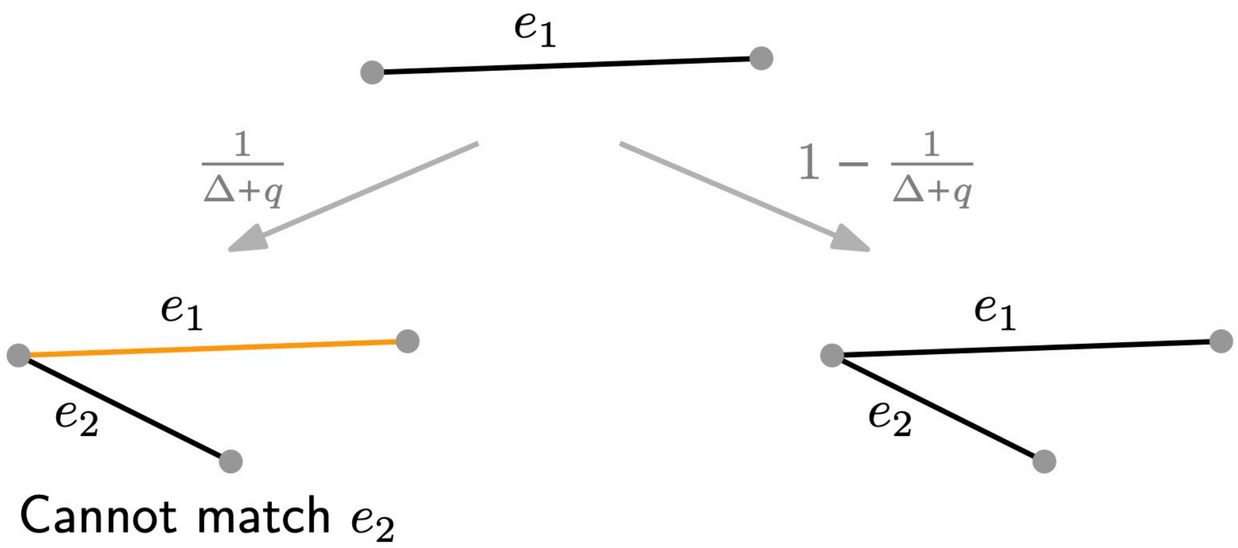
$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

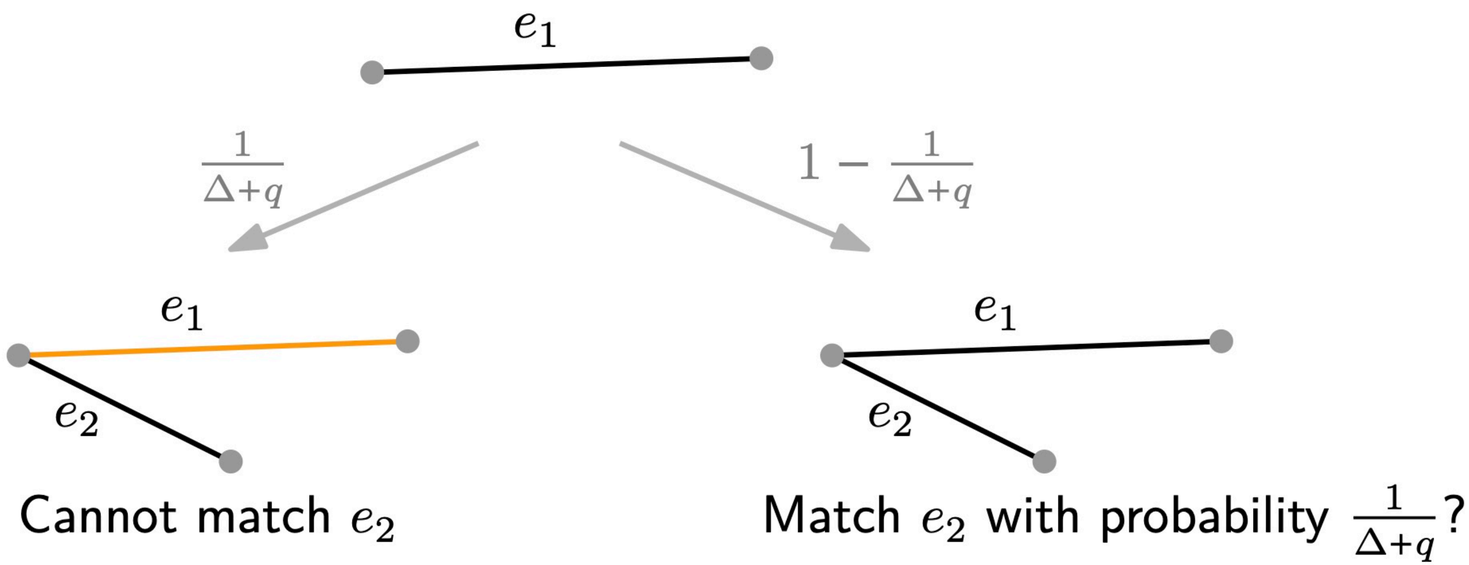
$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

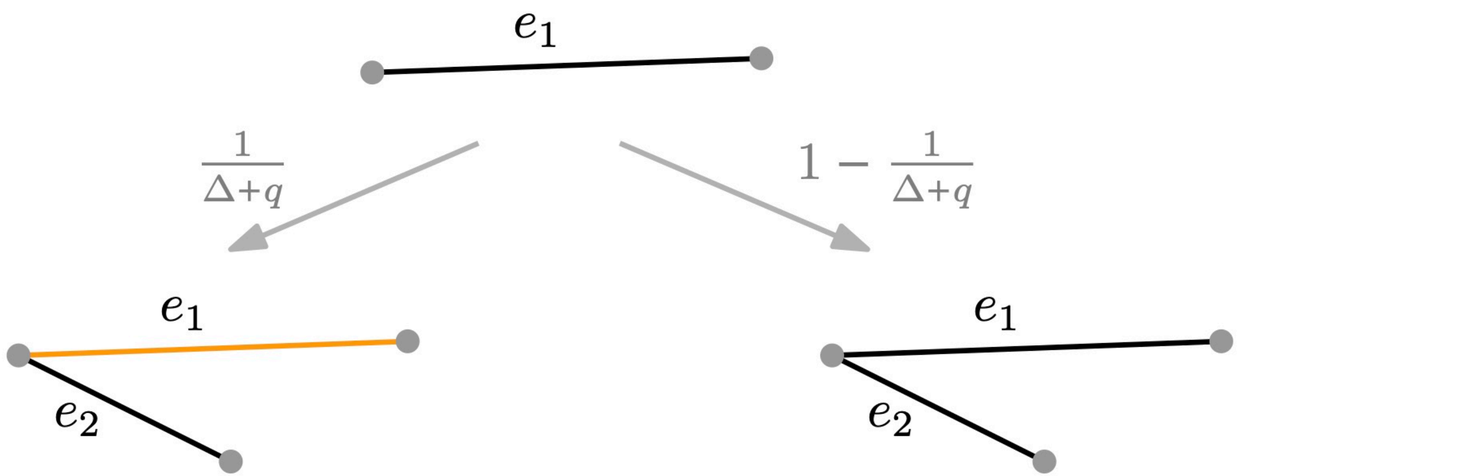
$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$



Cannot match e_2

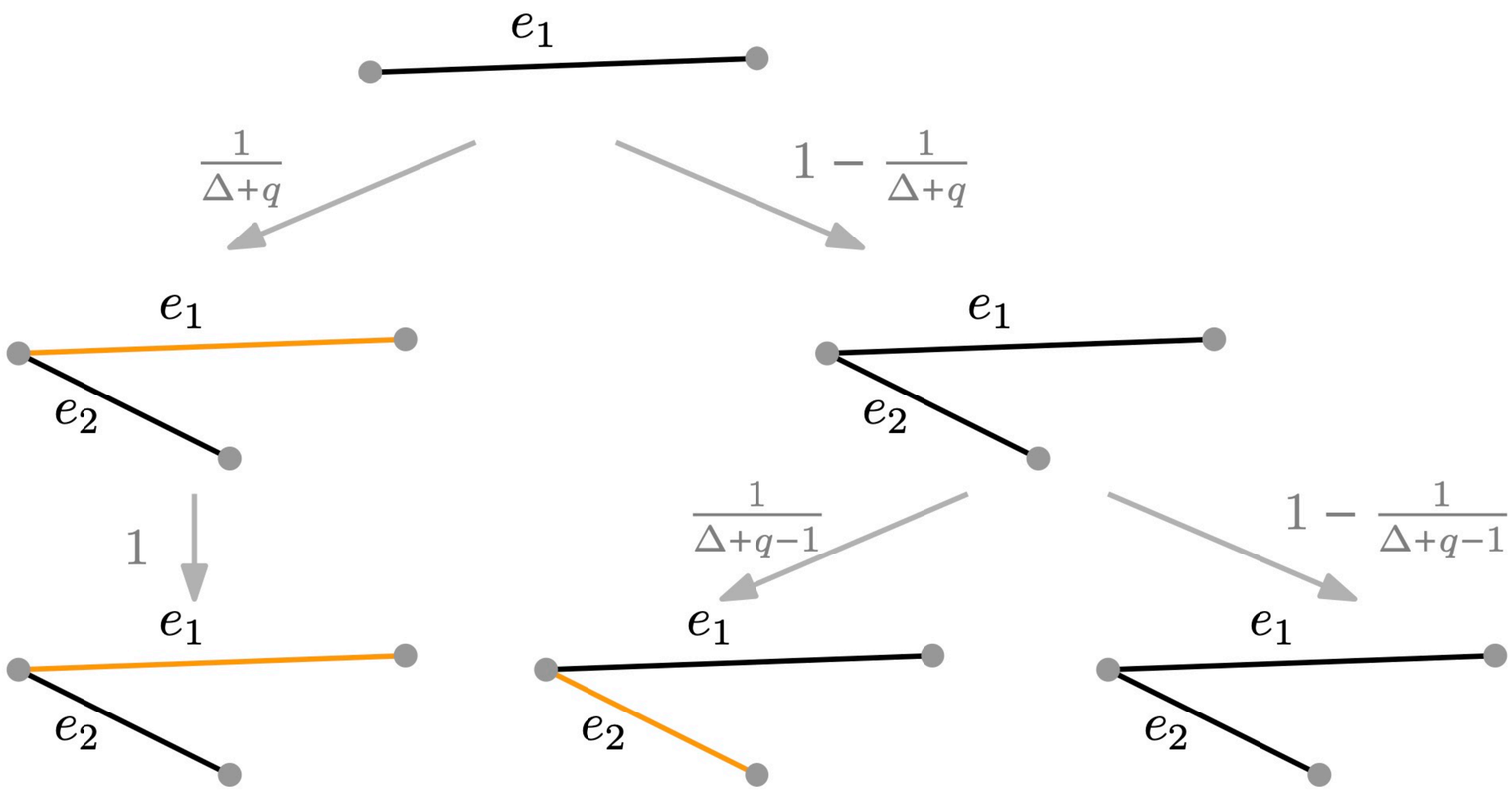
Match e_2 with probability ~~$\frac{1}{\Delta+q}$~~ ?

Must scale up: $\frac{1}{\Delta+q} / (1 - \frac{1}{\Delta+q})$

Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$



Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$

Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$

When $e_t = (u, v)$ arrives and u, v free:

Match e_t with probability $p_t = \dfrac{1/(\Delta + q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives }]}$

Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$

When $e_t = (u, v)$ arrives and u, v free:

Match e_t with probability $p_t = \dfrac{1/(\Delta + q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives }]}$

By definition, $\Pr[e_t \in M] = \frac{1}{\Delta + q}$

Fair Matching Algorithm

Goal: Match each edge with probability $\Pr[e \in M] \approx \frac{1}{\Delta}$

$$\Pr[e \in M] = \frac{1}{\Delta + q} \quad \text{with} \quad q = o(\Delta)$$

When $e_t = (u, v)$ arrives and u, v free:

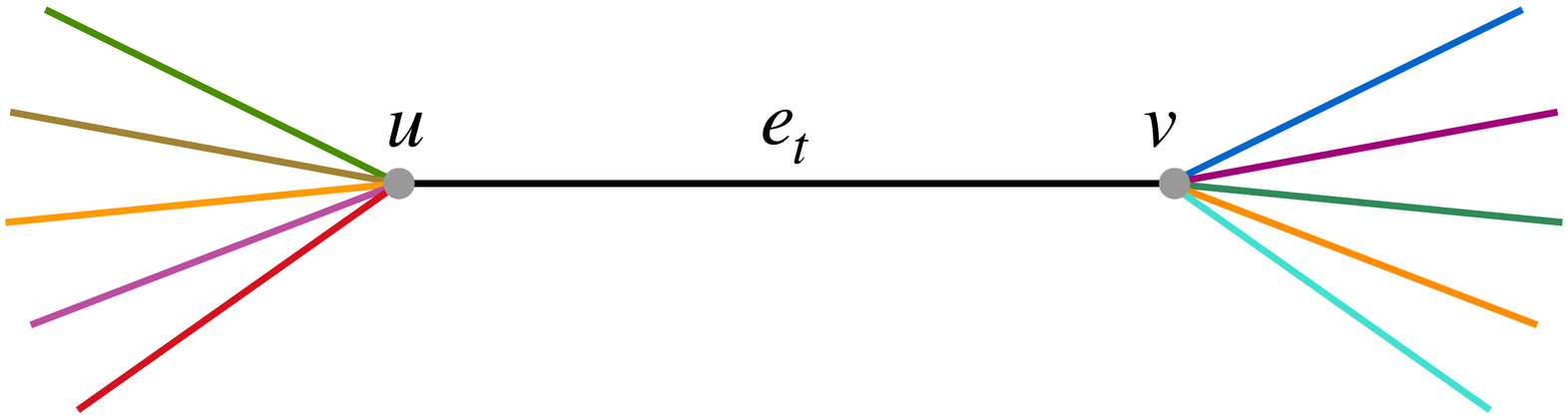
Match e_t with probability $p_t = \dfrac{1/(\Delta + q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives }]}$

By definition, $\Pr[e_t \in M] = \frac{1}{\Delta + q}$

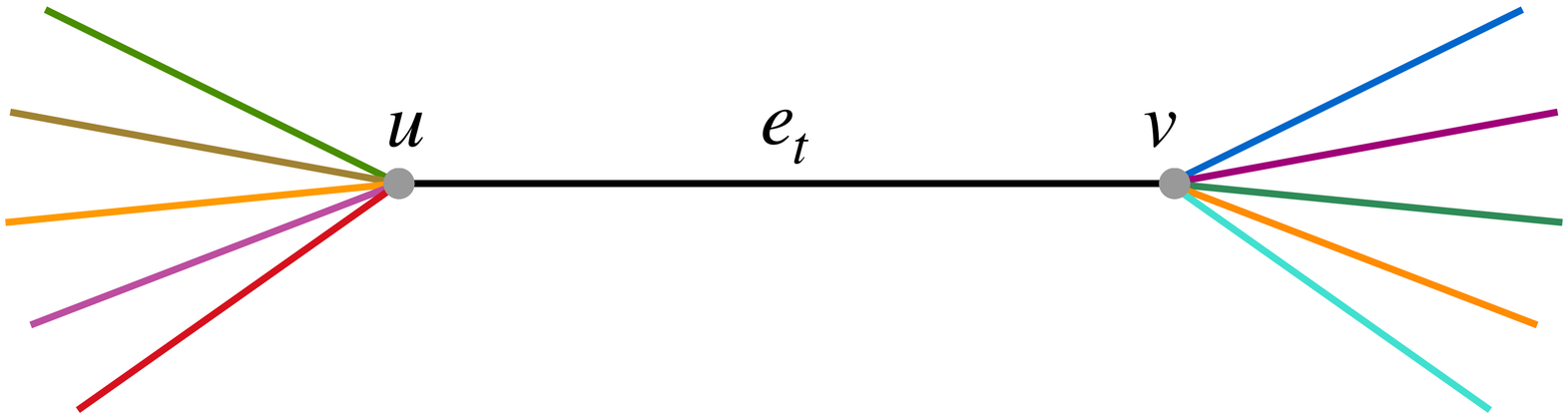
Potential problem: $p_t > 1$

Easy to prove: works if G is tree

Easy to prove: works if G is tree

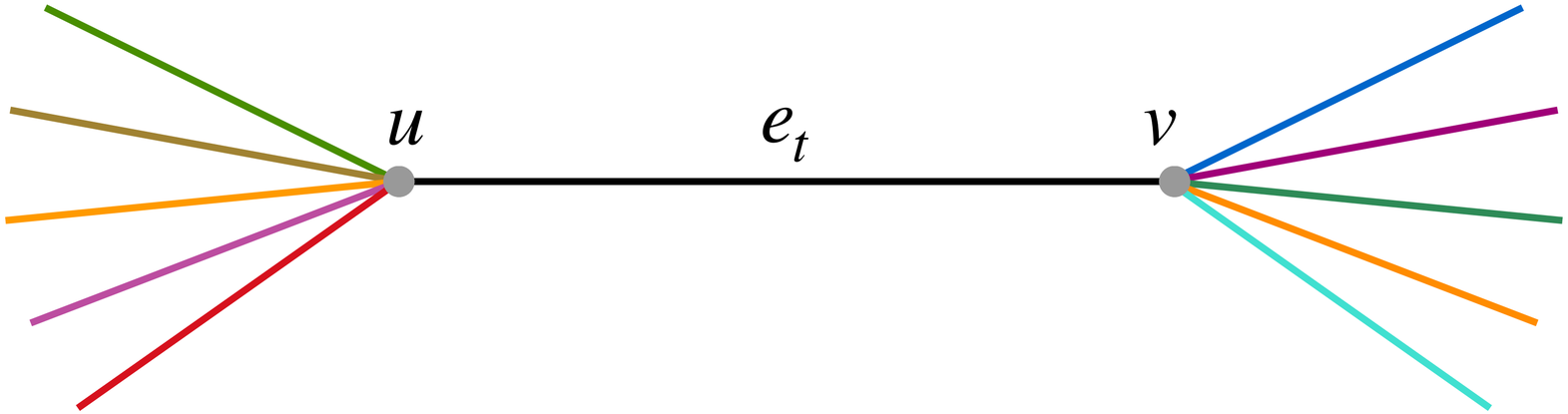


Easy to prove: works if G is tree



In this case "u is free" and "v is free" are **independent** so

Easy to prove: works if G is tree



In this case "u is free" and "v is free" are **independent** so

Pr[u,v free] = Pr[u free] · Pr[v free] $\geq \left(\frac{q}{\Delta + q}\right)^2 \implies$ things work out if $q = \omega(\sqrt{\Delta})$

Problem with general graphs

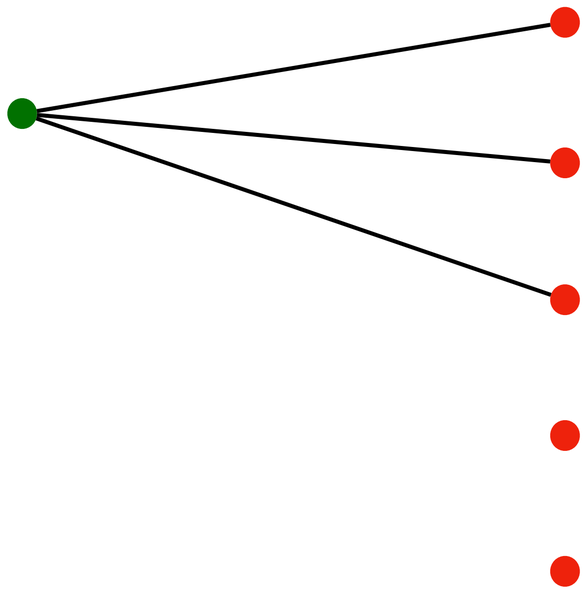
- Arbitrary (positive and negative) correlations

Prior work **controlled** these correlations

Vertex arrivals

Cohen, Peng, Wajc'19

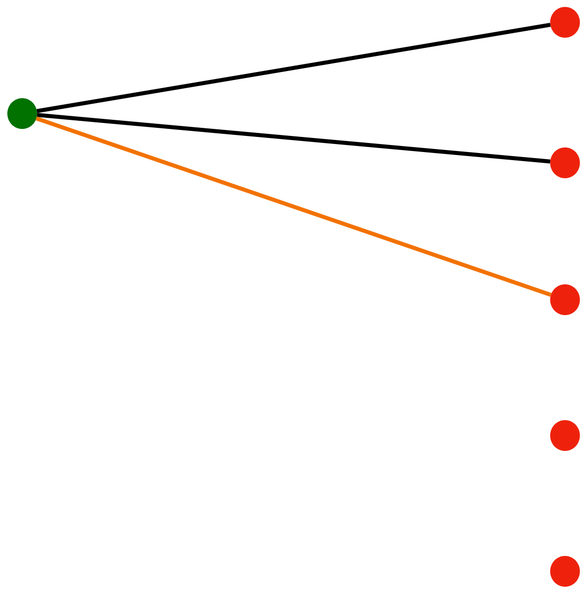
Online vertices **Offline vertices**



Vertex arrivals

Cohen, Peng, Wajc'19

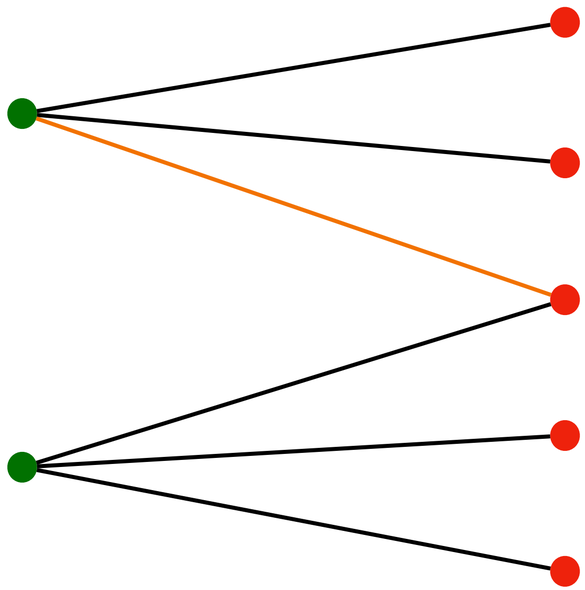
Online vertices **Offline vertices**



Vertex arrivals

Cohen, Peng, Wajc'19

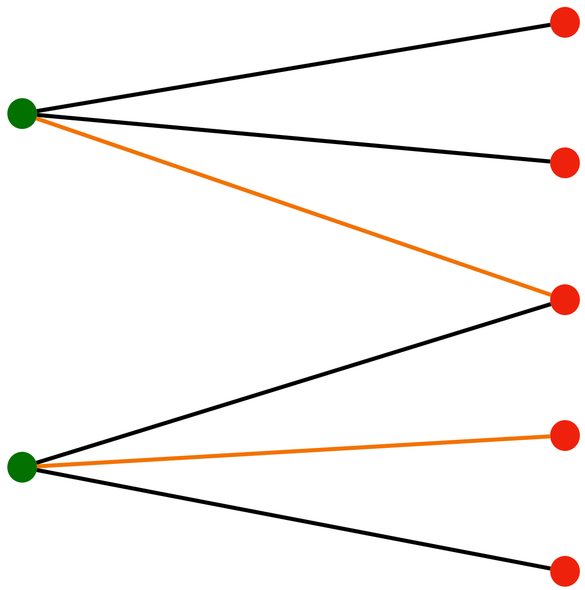
Online vertices **Offline vertices**



Vertex arrivals

Cohen, Peng, Wajc'19

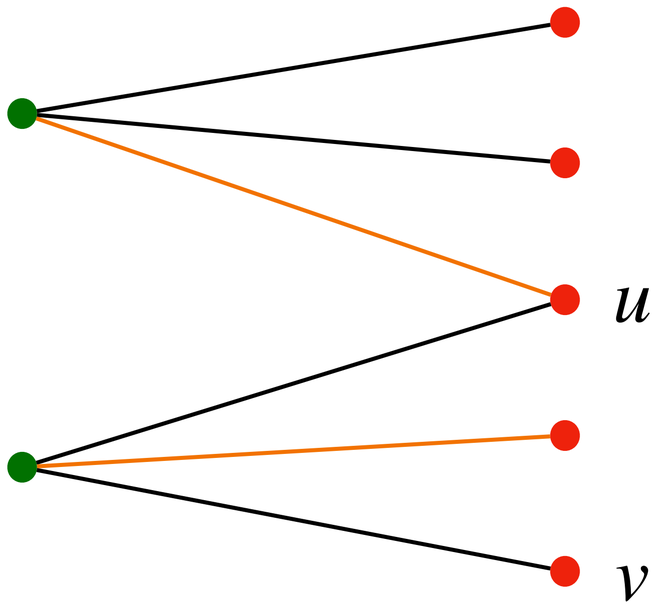
Online vertices **Offline vertices**



Vertex arrivals

Cohen, Peng, Wajc'19

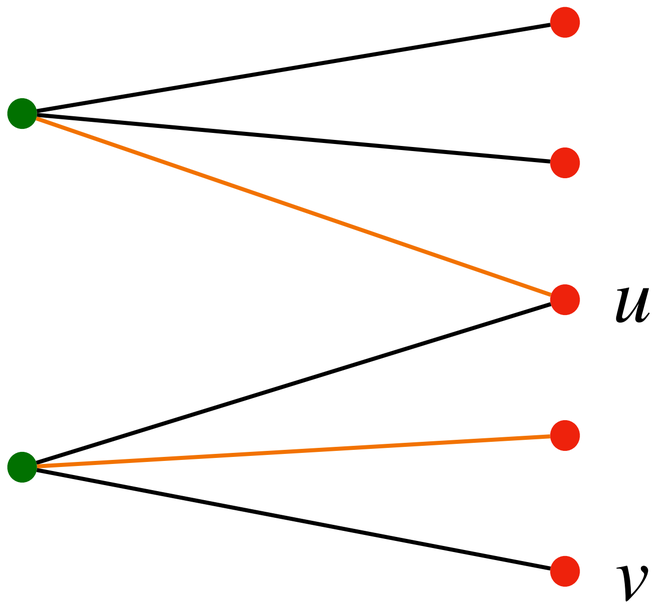
Online vertices **Offline vertices**



Vertex arrivals

Cohen, Peng, Wajc'19

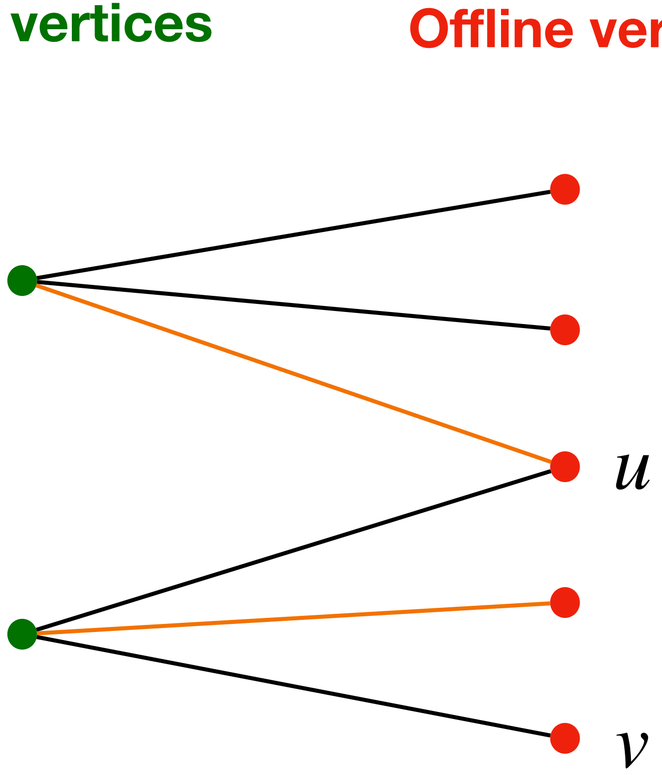
Online vertices **Offline vertices**



- "u free" and "v free" **negatively** correlated

Vertex arrivals

Cohen, Peng, Wajc'19



- "u free" and "v free" **negatively** correlated
- Chernoff bounds ensure concentration

General arrivals

Kulkarni, Liu, Sah, Sawhney, Tarnawski’22

General arrivals

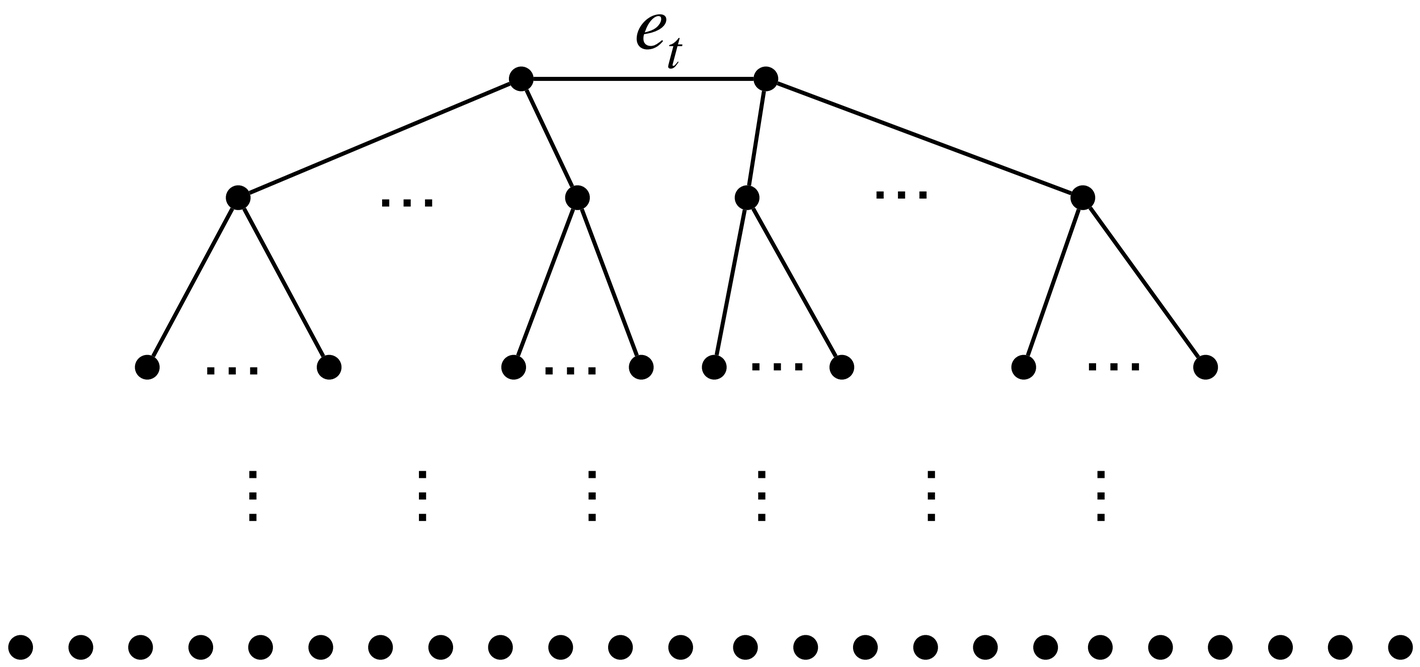
Kulkarni, Liu, Sah, Sawhney, Tarnawski'22

- Subsample graph G into sparse copies G_1, G_2 each copy sparse and locally tree like

General arrivals

Kulkarni, Liu, Sah, Sawhney, Tarnawski'22

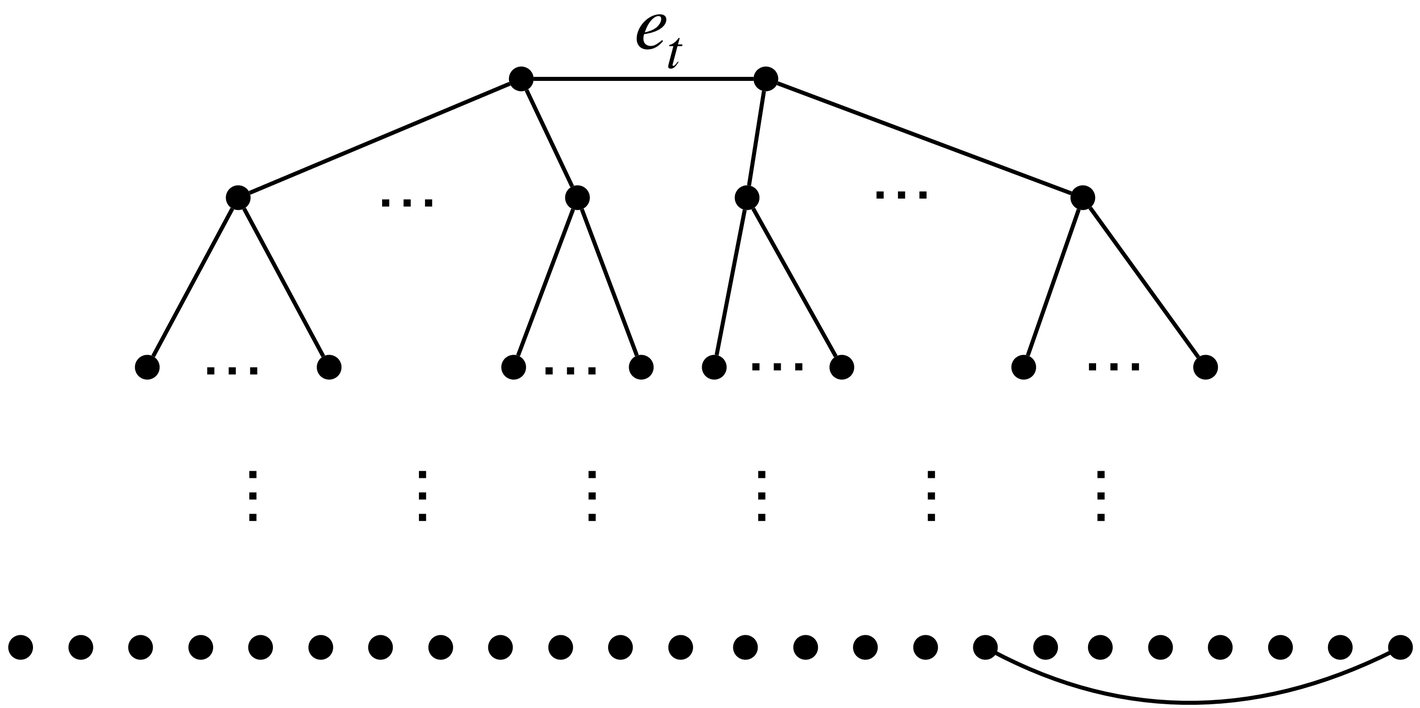
- Subsample graph G into sparse copies G_1, G_2 each copy sparse and locally tree like



General arrivals

Kulkarni, Liu, Sah, Sawhney, Tarnawski'22

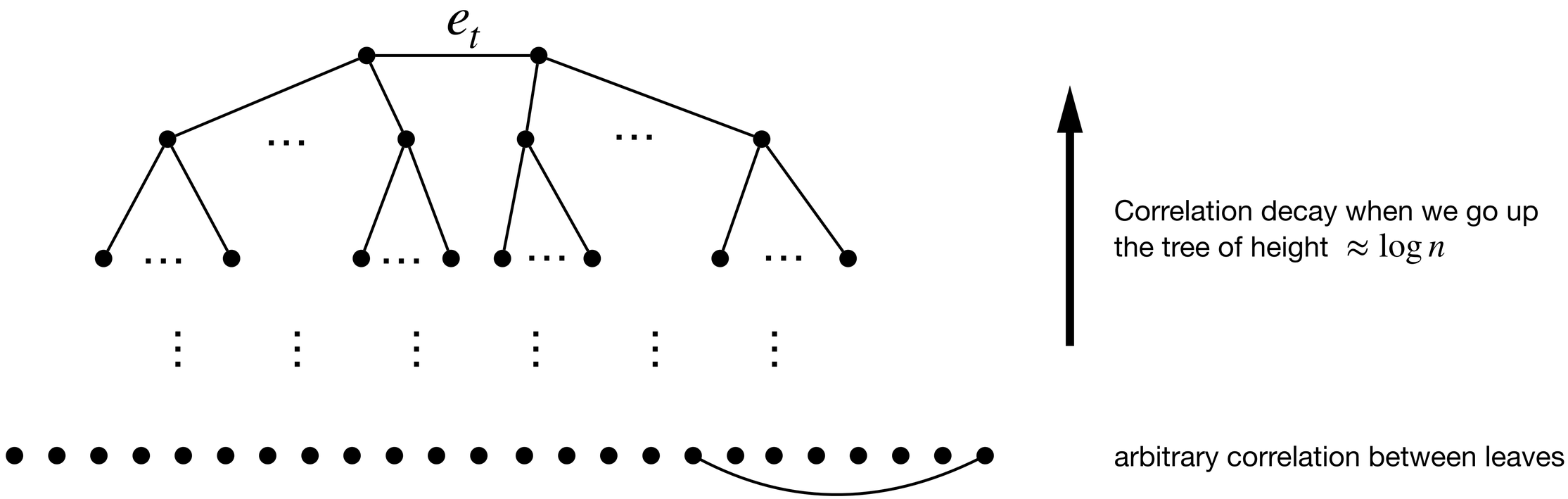
- Subsample graph G into sparse copies G_1, G_2 each copy sparse and locally tree like



General arrivals

Kulkarni, Liu, Sah, Sawhney, Tarnawski'22

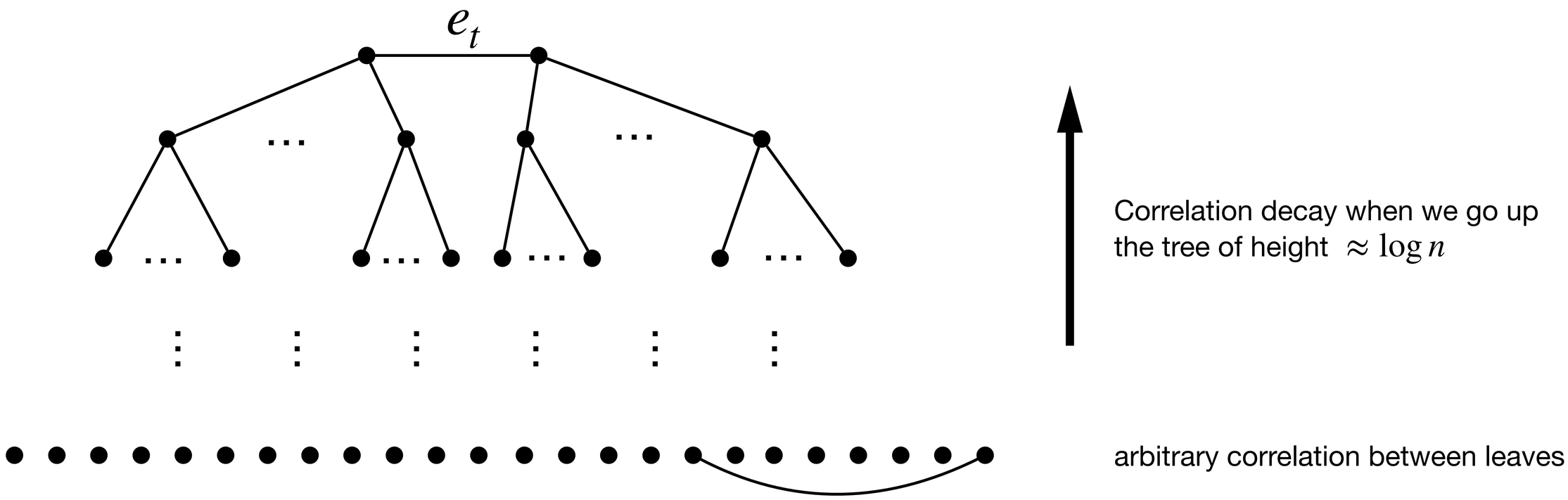
- Subsample graph G into sparse copies G_1, G_2 each copy sparse and locally tree like



General arrivals

Kulkarni, Liu, Sah, Sawhney, Tarnawski'22

- Subsample graph G into sparse copies G_1, G_2 each copy sparse and locally tree like
- Correlation decay allow to treat as independent case but requires #colors $\approx e/(e-1)\Delta$



Our Work

Embrace correlations instead of controlling them

Embrace correlations instead of controlling them

- Prior work got Chernoff-type concentration by considering special cases or controlling the correlation

Embrace correlations instead of controlling them

- Prior work got Chernoff-type concentration by considering special cases or controlling the correlation
- We keep correlations but change viewpoint

Embrace correlations instead of controlling them

- Prior work got Chernoff-type concentration by considering special cases or controlling the correlation
- We keep correlations but change viewpoint

Key takeaway:

Formulate **quantity as a martingale**, bound stepsize and observed variance

Chernoff-type concentration

Recall algorithm

When $e_t = (u, v)$ arrives and u, v free:

$$\text{Match } e_t \text{ with probability } p_t = \frac{1/(\Delta + q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives }]}$$

Recall algorithm

When $e_t = (u, v)$ arrives and u, v free:

Match e_t with probability $p_t = \dfrac{1/(\Delta + q)}{\Pr[u, v \text{ both free when } e_t \text{ arrives }]}$

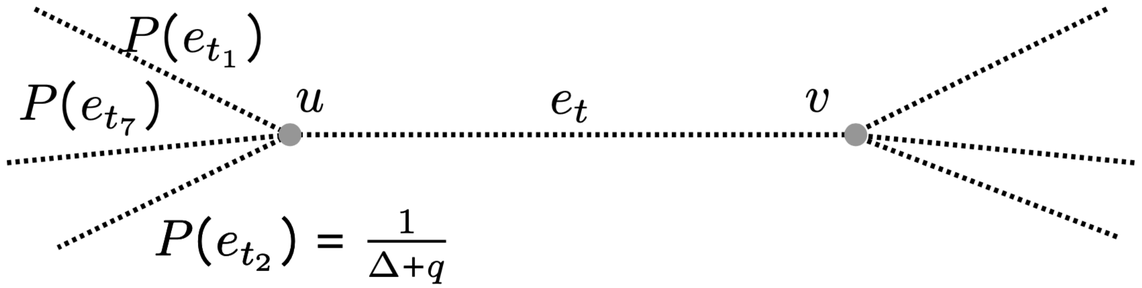
When $e_t = (u, v)$ arrives and u, v free:

Match e_t with probability $p_t = \dfrac{1/(\Delta + q)}{\text{execution-dependent scaling factor}}$

Our alternative algorithm

When $e_t = (u, v)$ arrives:

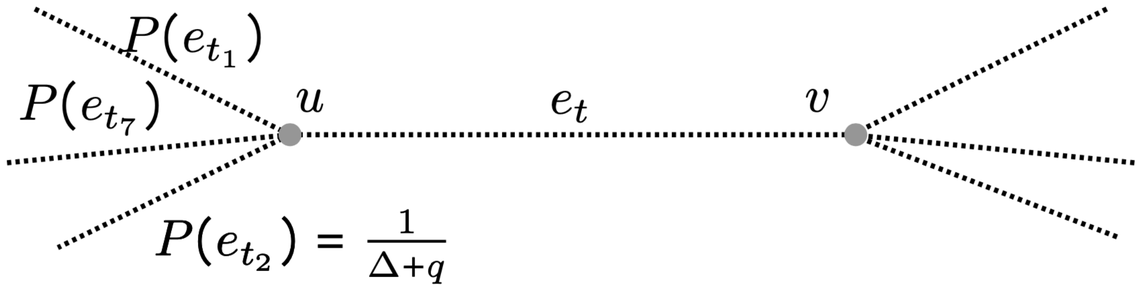
Match e_t with probability $P(e_t) = \begin{cases} 0 & \text{if } u \text{ or } v \text{ are matched} \\ \dfrac{1/(\Delta + q)}{\prod_{e_{t_j} \in \delta(u) \cup \delta(v)} (1 - P(e_{t_j}))} & \text{otherwise} \end{cases}$



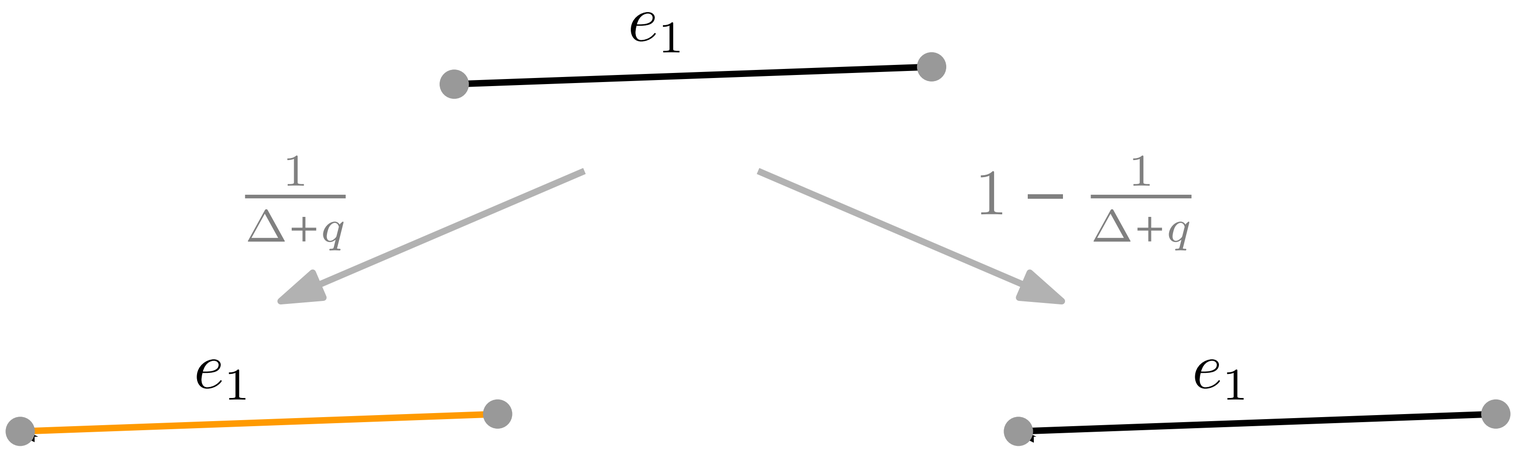
Our alternative algorithm

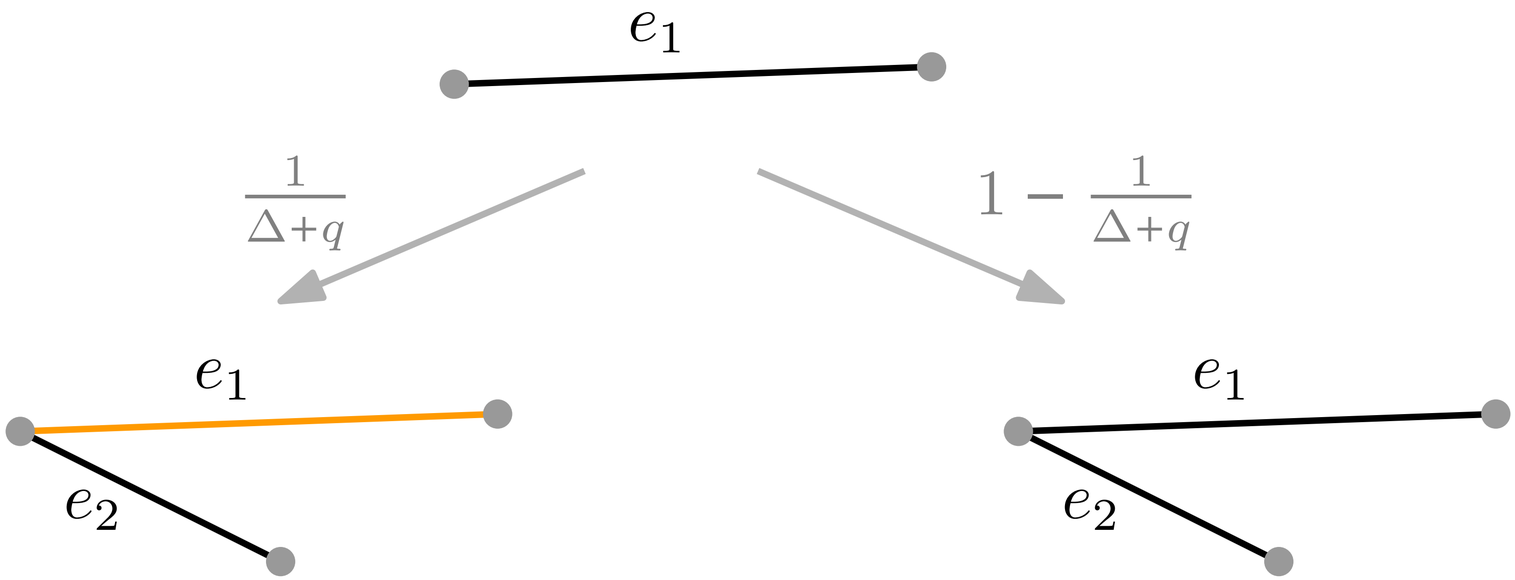
When $e_t = (u, v)$ arrives:

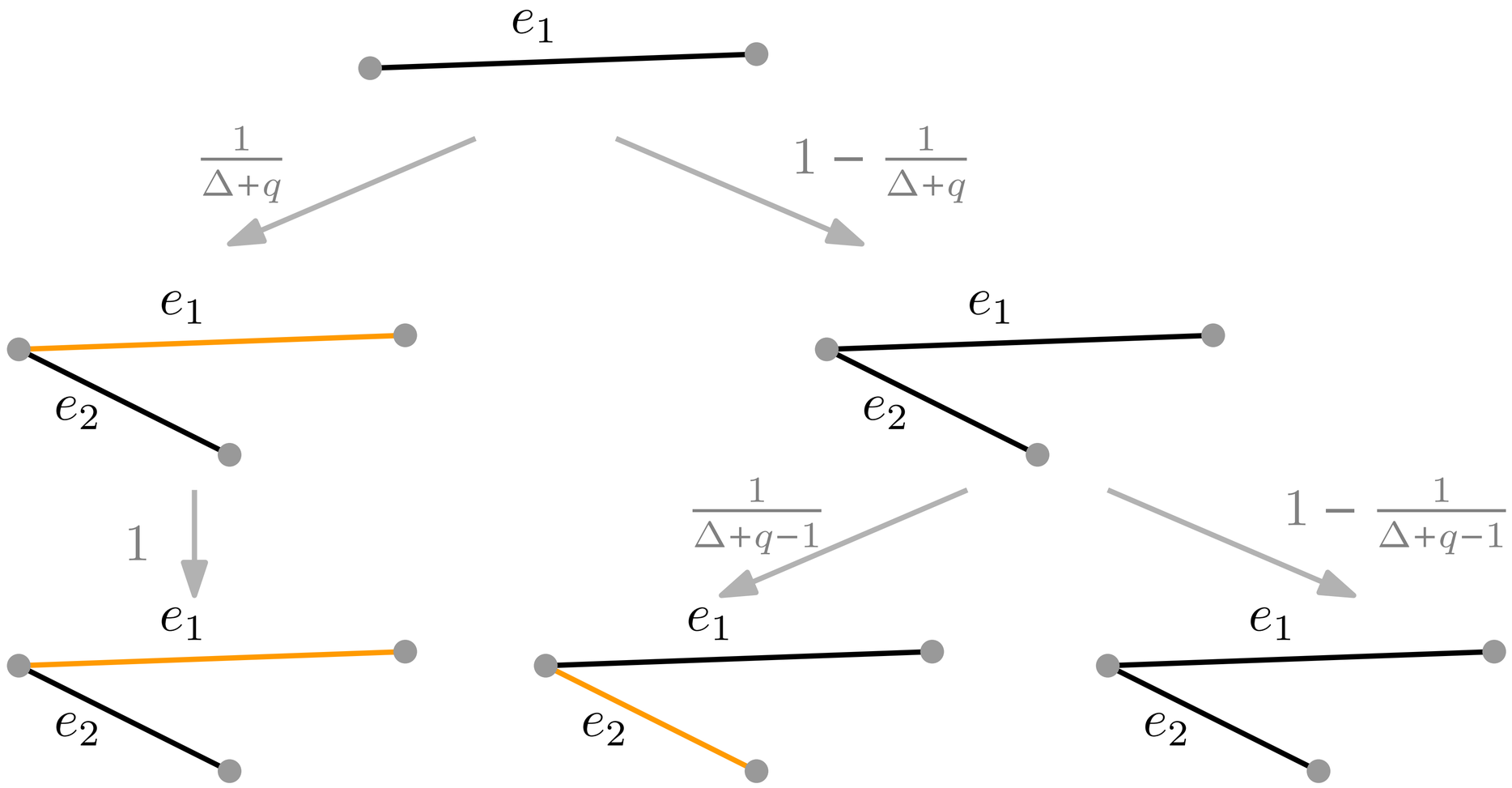
Match e_t with probability $P(e_t) = \begin{cases} 0 & \text{if } u \text{ or } v \text{ are matched} \\ \dfrac{1/(\Delta + q)}{\prod_{e_{t_j} \in \delta(u) \cup \delta(v)} (1 - P(e_{t_j}))} & \text{otherwise} \end{cases}$

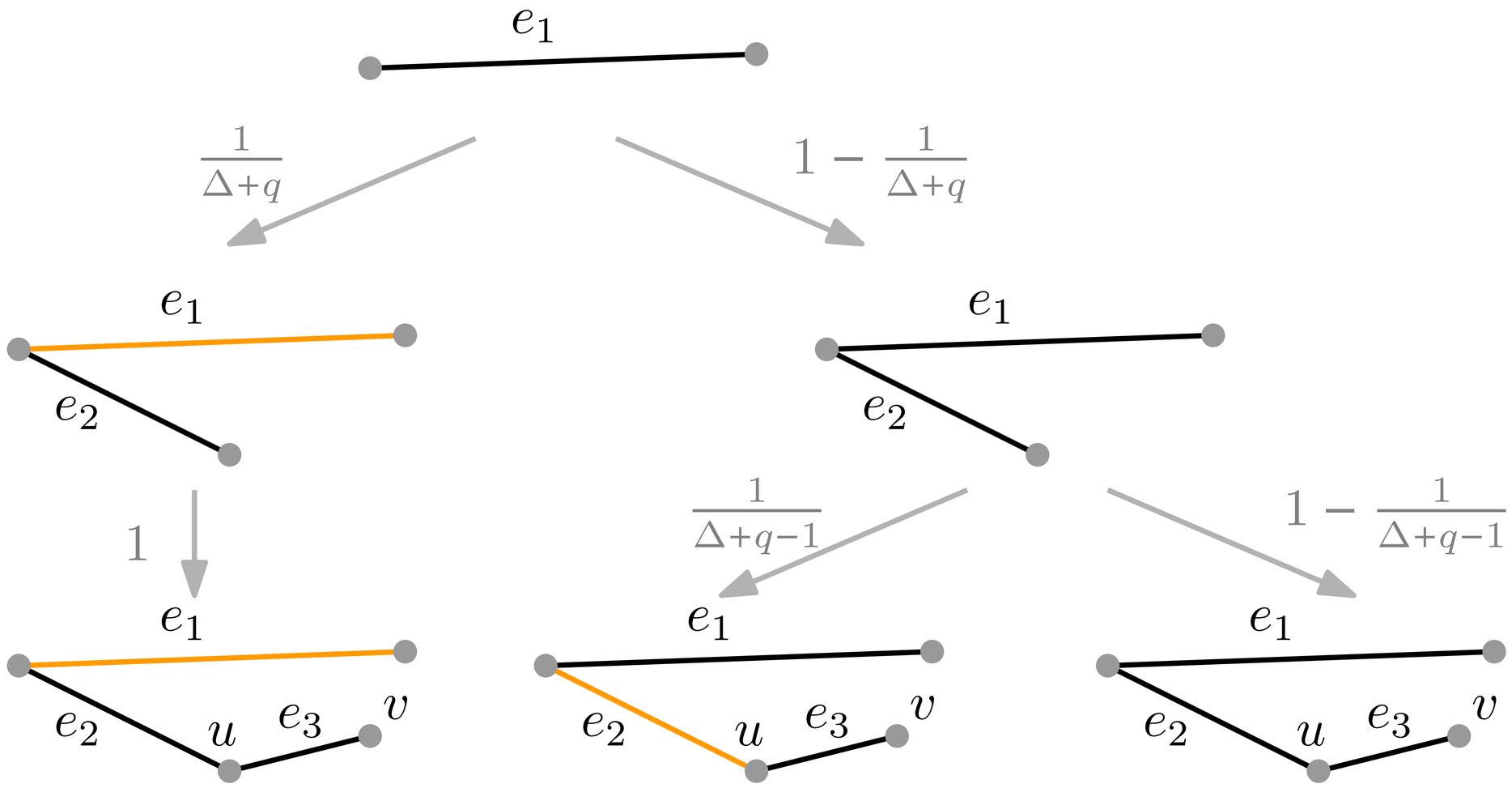


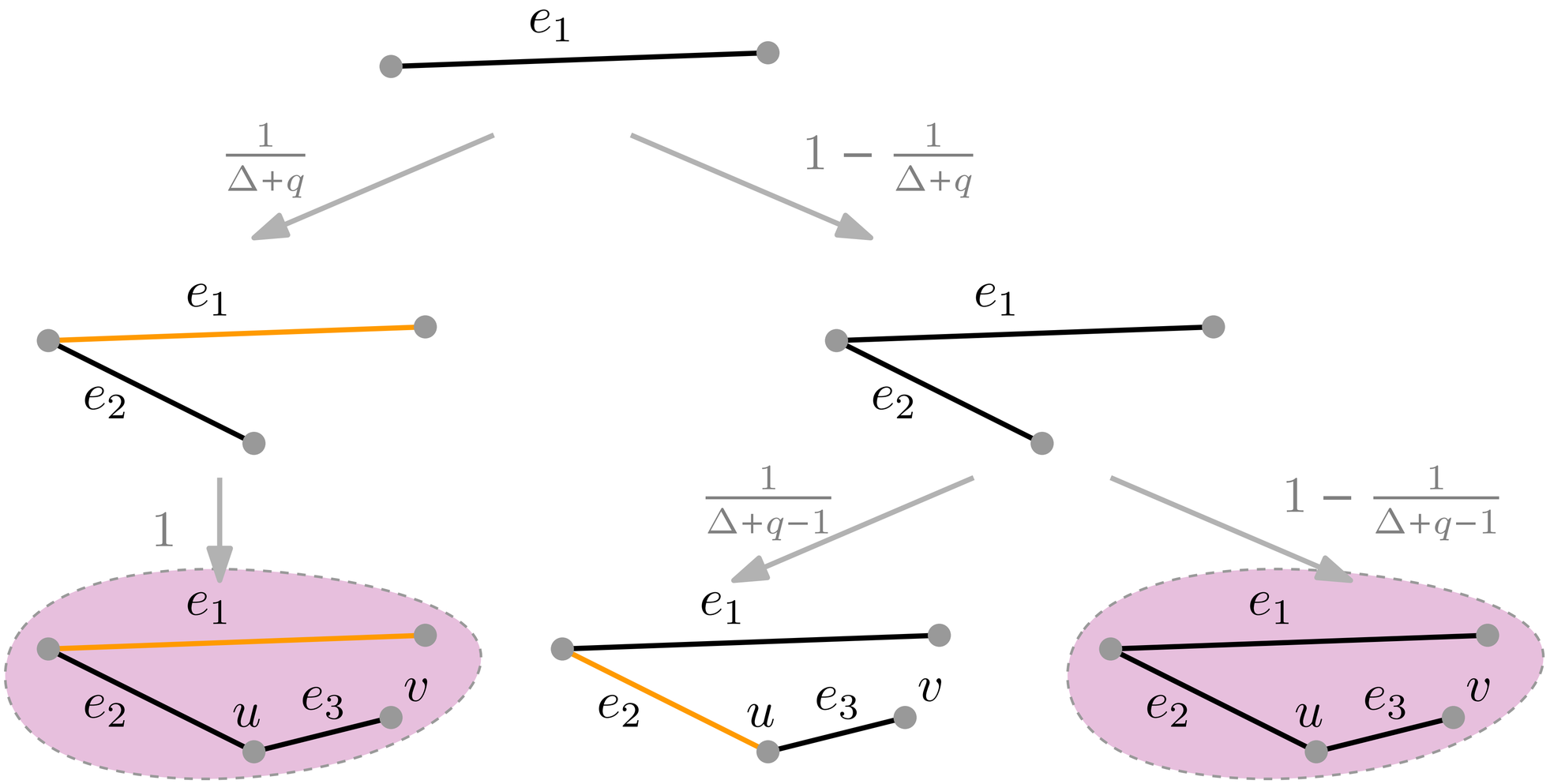
Potential problem: $P(e_t) > 1$

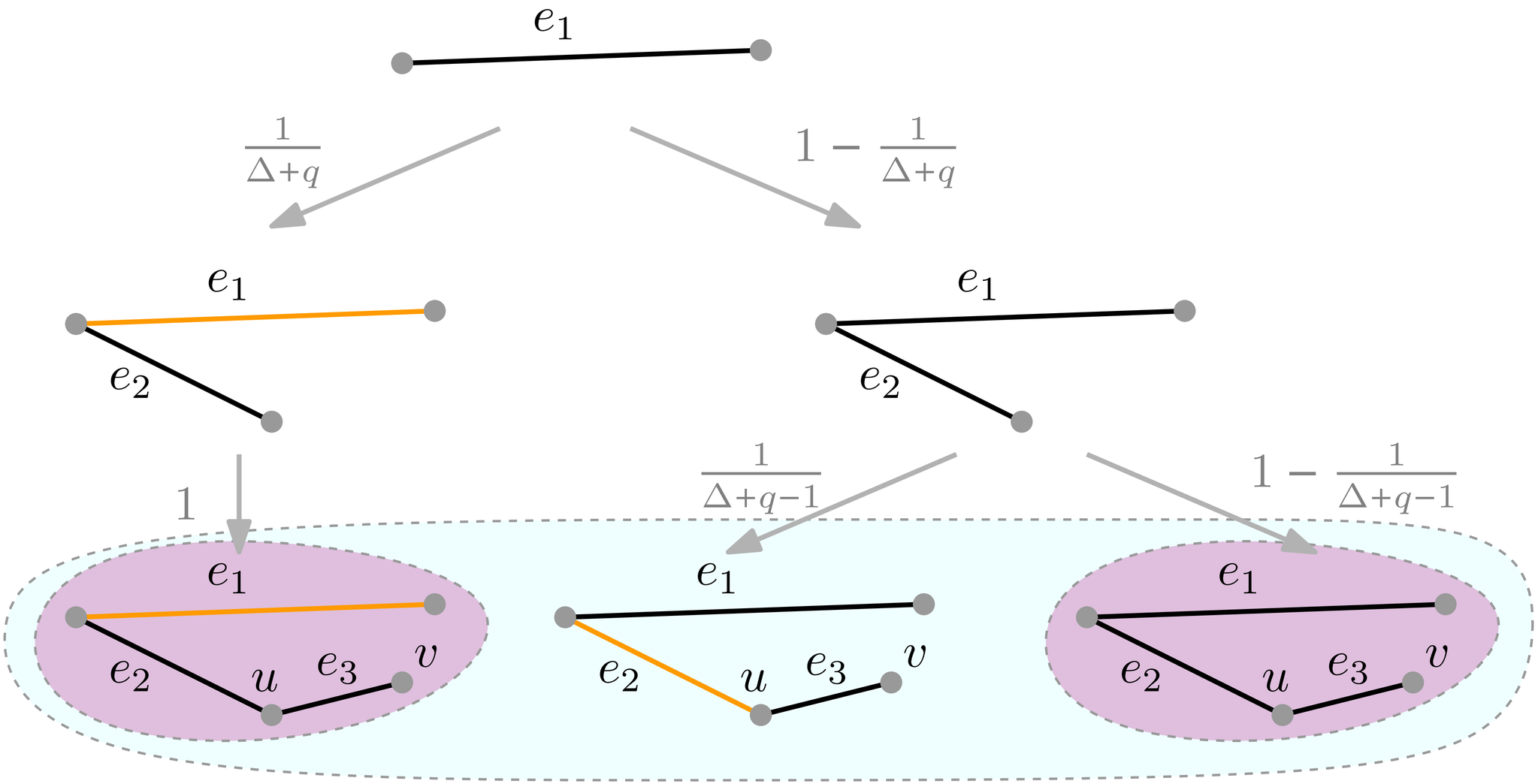

e_1
$\frac{1}{\Delta+q}$
$1-\frac{1}{\Delta+q}$
e_1
e_1


e_1
$\frac{1}{\Delta+q}$
$1 - \frac{1}{\Delta+q}$
e_1
e_2
e_1
e_2


e_1
$\frac{1}{\Delta+q}$
$1-\frac{1}{\Delta+q}$
e_1
e_2
e_1
e_2
1
e_1
e_2
$\frac{1}{\Delta+q-1}$
$1-\frac{1}{\Delta+q-1}$
e_1
e_2
e_1
e_2

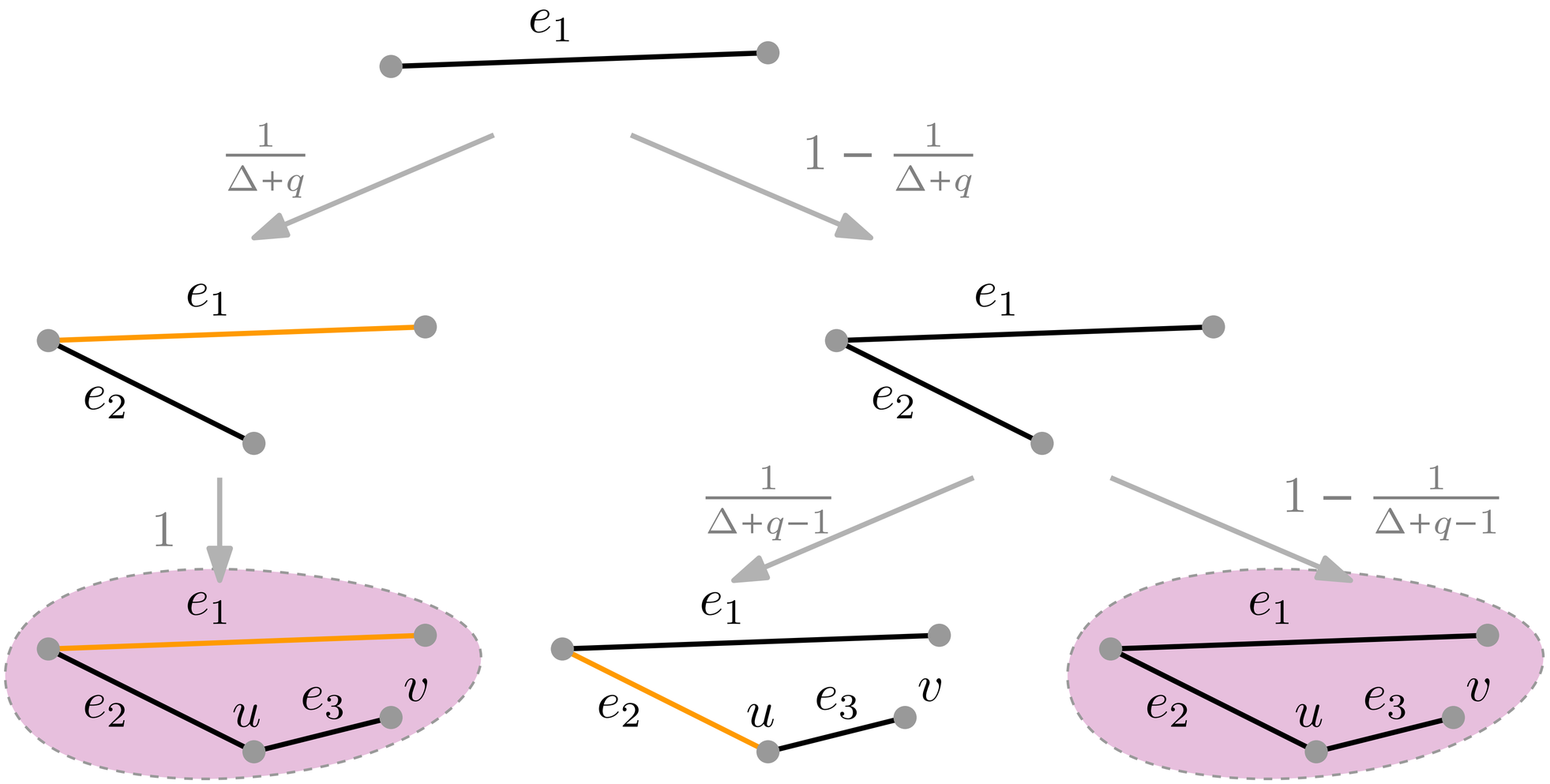

e_1
$\frac{1}{\Delta+q}$
$1-\frac{1}{\Delta+q}$
e_1
e_2
e_1
e_2
1
$\frac{1}{\Delta+q-1}$
$1-\frac{1}{\Delta+q-1}$
e_1
e_2 u e_3 v
e_1
e_2 u e_3 v
e_1
e_2 u e_3 v


e_1
$\frac{1}{\Delta+q}$
$1-\frac{1}{\Delta+q}$
e_1
e_2
e_1
e_2
1
$\frac{1}{\Delta+q-1}$
$1-\frac{1}{\Delta+q-1}$
e_1
e_2
u
e_3
v
e_1
e_2
u
e_3
v
e_1
e_2
u
e_3
v

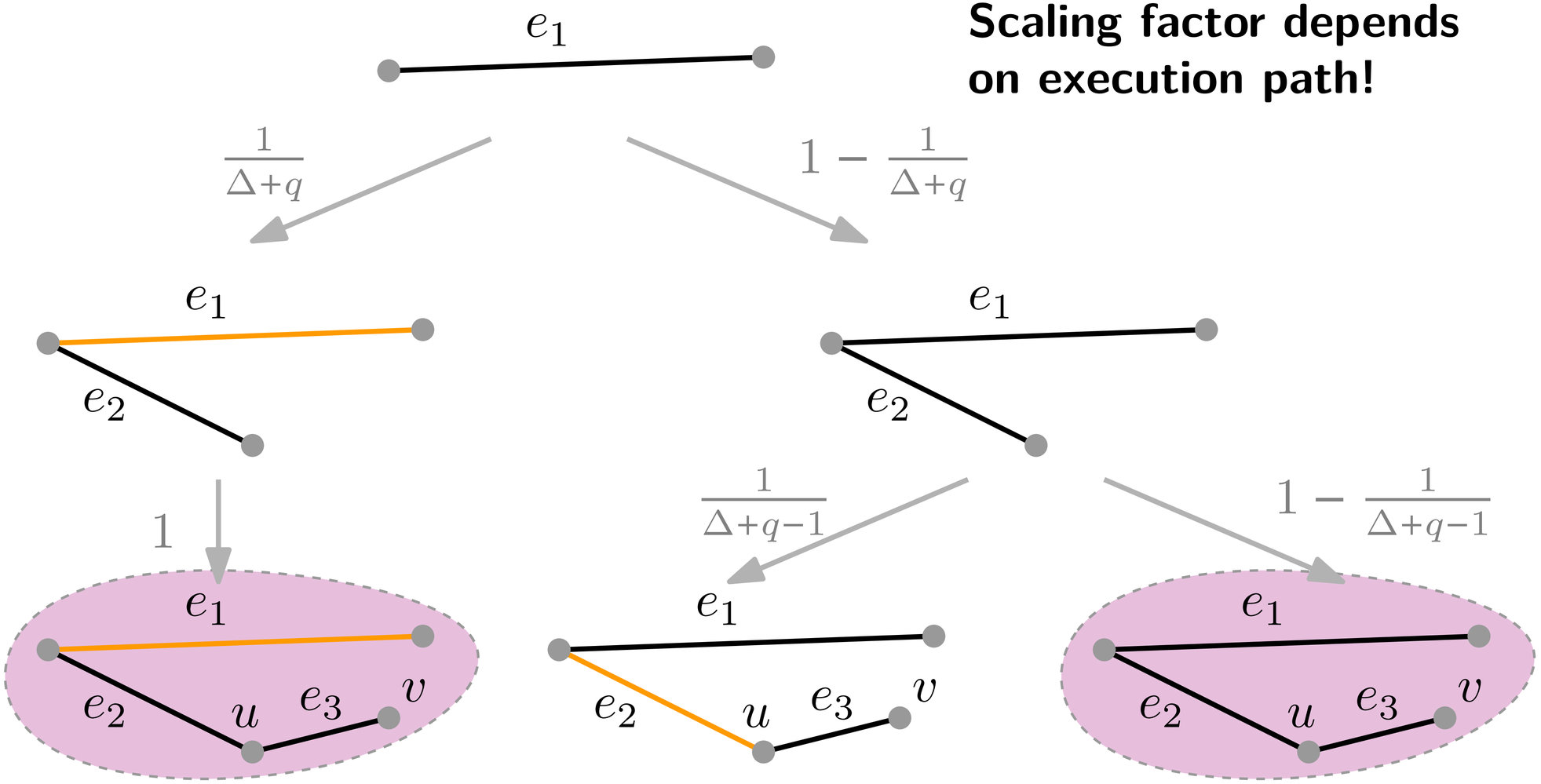


Old Algo: $\Pr[u, v \text{ free}] = (1 - \frac{1}{\Delta+q})$

Uses same scaling factor


e_1
$\frac{1}{\Delta+q}$
$1-\frac{1}{\Delta+q}$
e_1
e_2
e_1
e_2
1
$\frac{1}{\Delta+q-1}$
$1-\frac{1}{\Delta+q-1}$
e_1
e_2
u
e_3
v
e_1
e_2
u
e_3
v
e_1
e_2
u
e_3
v

Scaling factor depends on execution path!



scaling $= 1/(1 - P(e_2)) = 1$

scaling $= 1/(1 - P(e_2)) = 1/(1 - \frac{1}{\Delta+q-1})$

Analysis Idea — Random walk

Core of Analysis: Prove $P(e_t) \leq 1$

Our Algorithm: $P(e_t) := \frac{1/(\Delta+q)}{\prod_{e \in \delta(u) \cup \delta(v)} (1-P(e))}$

$$q := \approx \Delta^{3/4}$$

Analysis Idea — Random walk

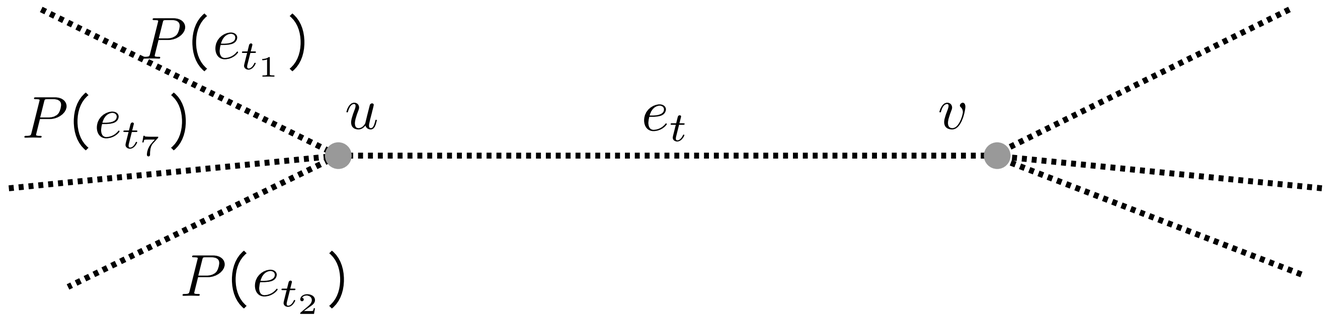
Core of Analysis: Prove $P(e_t) \leq \frac{10}{\sqrt{\Delta}}$

Our Algorithm: $P(e_t) := \frac{1/(\Delta+q)}{\prod_{e \in \delta(u) \cup \delta(v)} (1-P(e))}$

$$q := \approx \Delta^{3/4}$$

Analysis Idea — Random walk

Scaling factor $S_u := \prod_{e_{t_j} \in \delta(u)} (1 - P(e_{t_j}))$ **Goal:** Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$

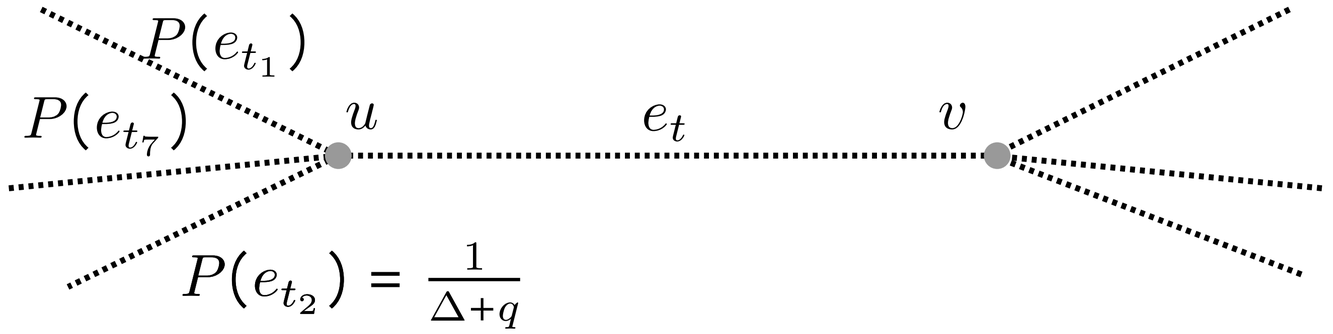


Our Algorithm: $P(e_t) := \frac{1/(\Delta+q)}{\prod_{e \in \delta(u) \cup \delta(v)} (1-P(e))}$

$q := \approx \Delta^{3/4}$

Analysis Idea — Random walk

Scaling factor $S_u := \prod_{e_{t_j} \in \delta(u)} (1 - P(e_{t_j}))$ **Goal:** Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$

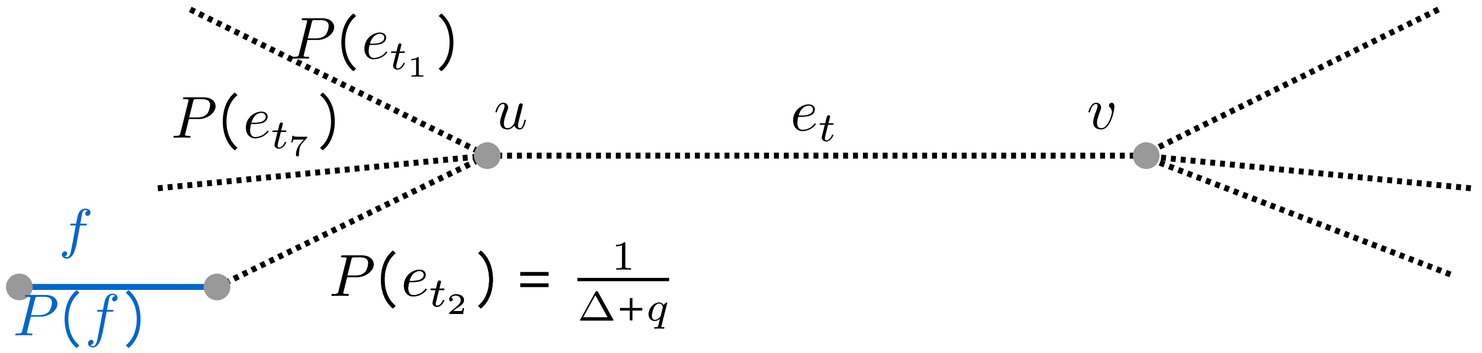


Our Algorithm: $P(e_t) := \frac{1/(\Delta+q)}{\prod_{e \in \delta(u) \cup \delta(v)} (1-P(e))}$

$q := \approx \Delta^{3/4}$

Analysis Idea — Random walk

Scaling factor $S_u := \prod_{e_{t_j} \in \delta(u)} (1 - P(e_{t_j}))$ **Goal:** Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$

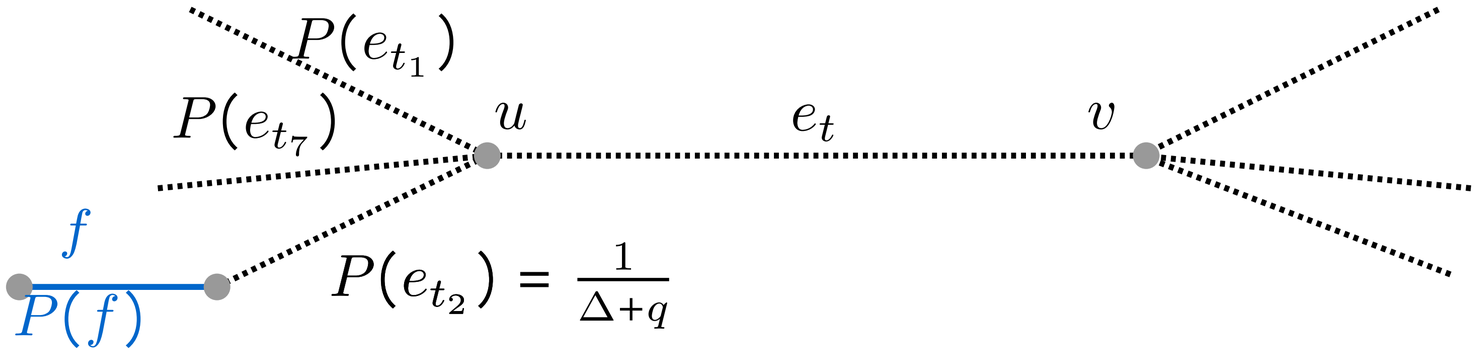


Our Algorithm: $P(e_t) := \frac{1/(\Delta+q)}{\prod_{e \in \delta(u) \cup \delta(v)} (1-P(e))}$

$q := \approx \Delta^{3/4}$

Analysis Idea — Random walk

Scaling factor $S_u := \prod_{e_{t_j} \in \delta(u)} (1 - P(e_{t_j}))$ **Goal:** Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$



If f matched $\implies P^{new}(e_{t_2}) \leftarrow 0$

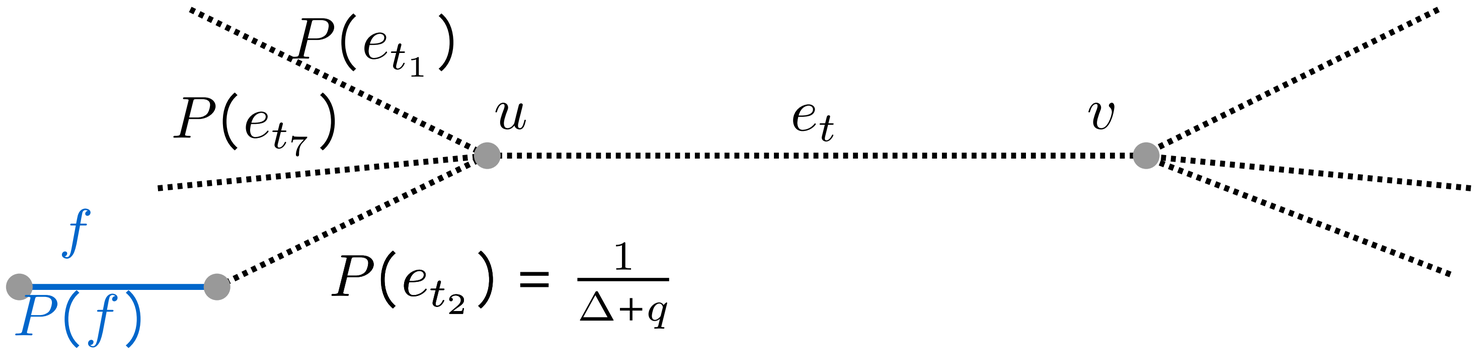
If f not matched $\implies P^{new}(e_{t_2}) \leftarrow P(e_{t_2})/(1 - P(f))$

Our Algorithm: $P(e_t) := \frac{1/(\Delta+q)}{\prod_{e \in \delta(u) \cup \delta(v)} (1-P(e))}$

$q := \approx \Delta^{3/4}$

Analysis Idea — Random walk

Scaling factor $S_u := \prod_{e_{t_j} \in \delta(u)} (1 - P(e_{t_j}))$ **Goal:** Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$



If f matched $\implies P^{new}(e_{t_2}) \leftarrow 0$ $\mathbb{E}[S_u^{new}] = S_u$

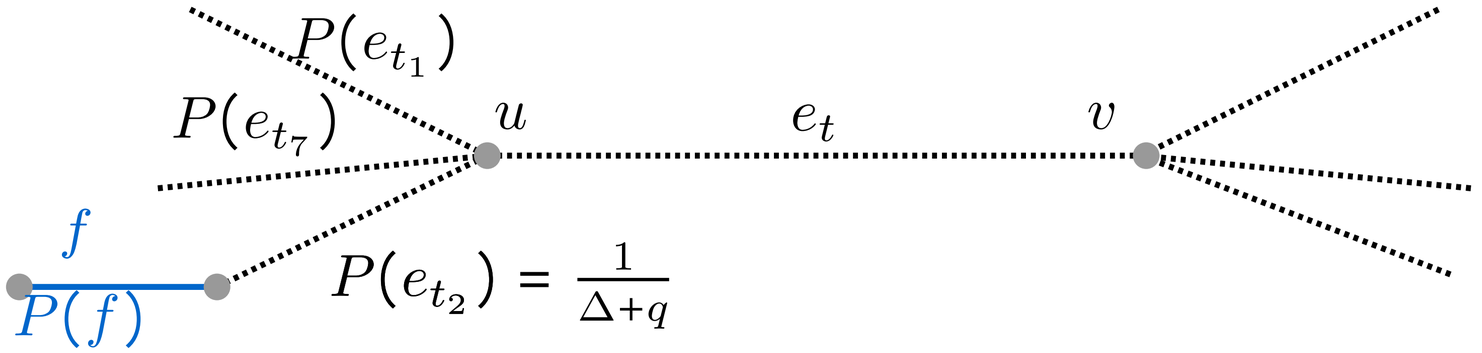
If f not matched $\implies P^{new}(e_{t_2}) \leftarrow P(e_{t_2})/(1 - P(f))$

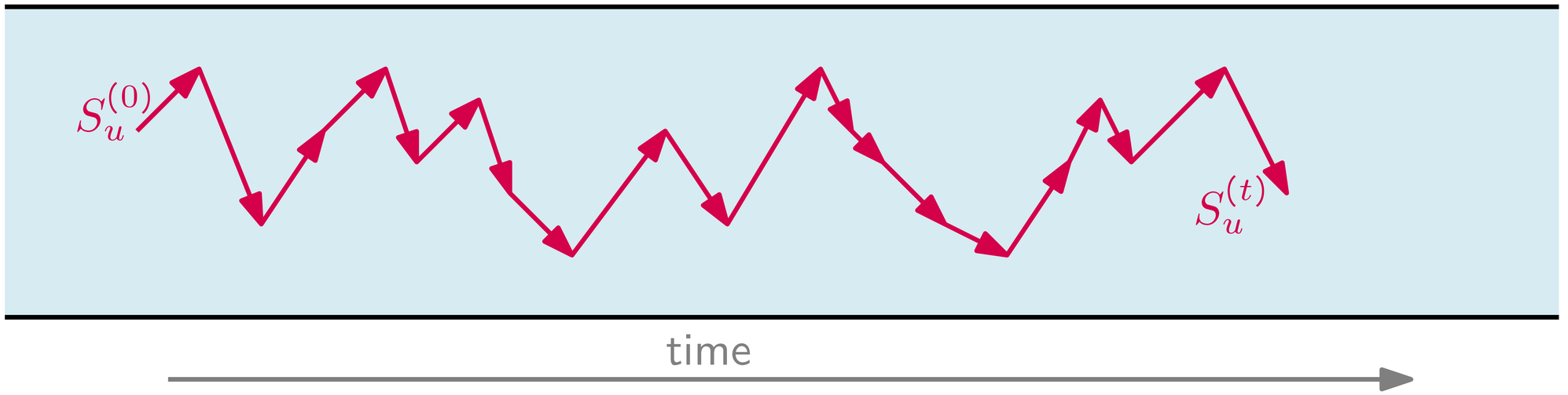
Our Algorithm: $P(e_t) := \frac{1/(\Delta+q)}{\prod_{e \in \delta(u) \cup \delta(v)} (1-P(e))}$

$q := \approx \Delta^{3/4}$

Analysis Idea — Random walk

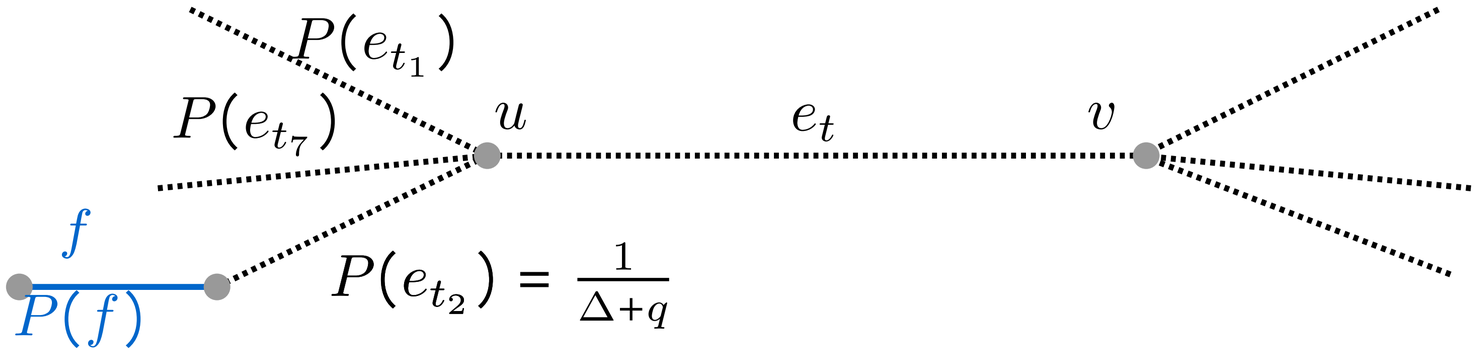
Scaling factor $S_u := \prod_{e_{t_j} \in \delta(u)} (1 - P(e_{t_j}))$ **Goal:** Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$



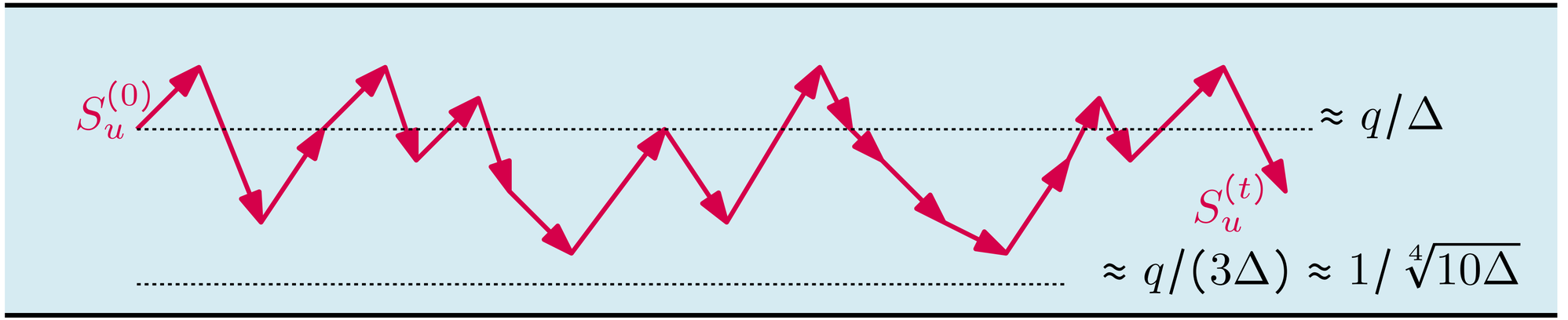


Analysis Idea — Random walk

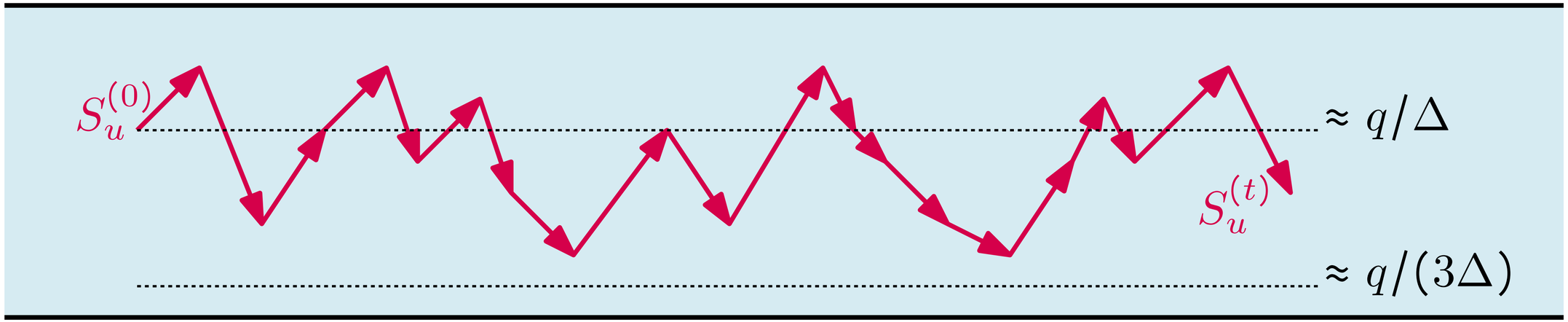
Scaling factor $S_u := \prod_{e_{t_j} \in \delta(u)} (1 - P(e_{t_j}))$ **Goal:** Show $S_u \gtrapprox \sqrt[4]{\frac{1}{10\Delta}}$



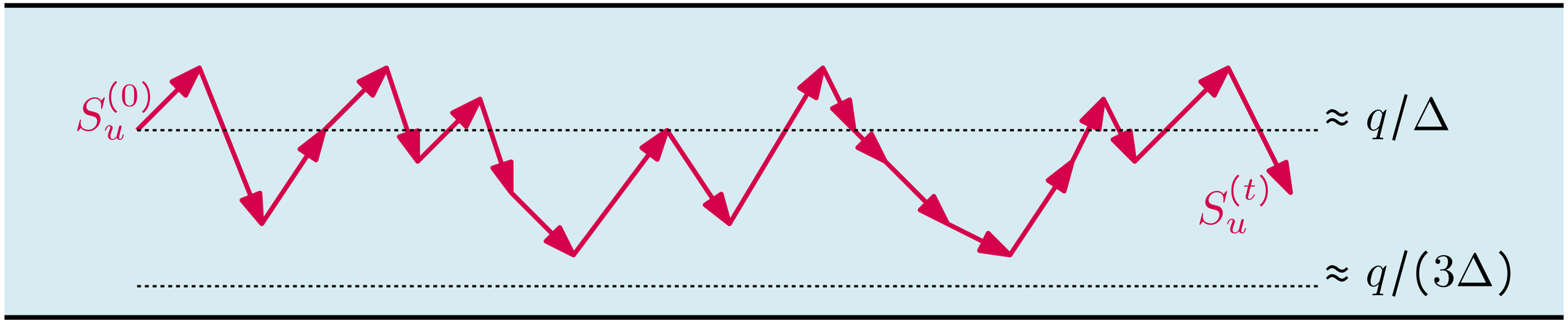
$q := \approx \Delta^{3/4}$



Martingale Process



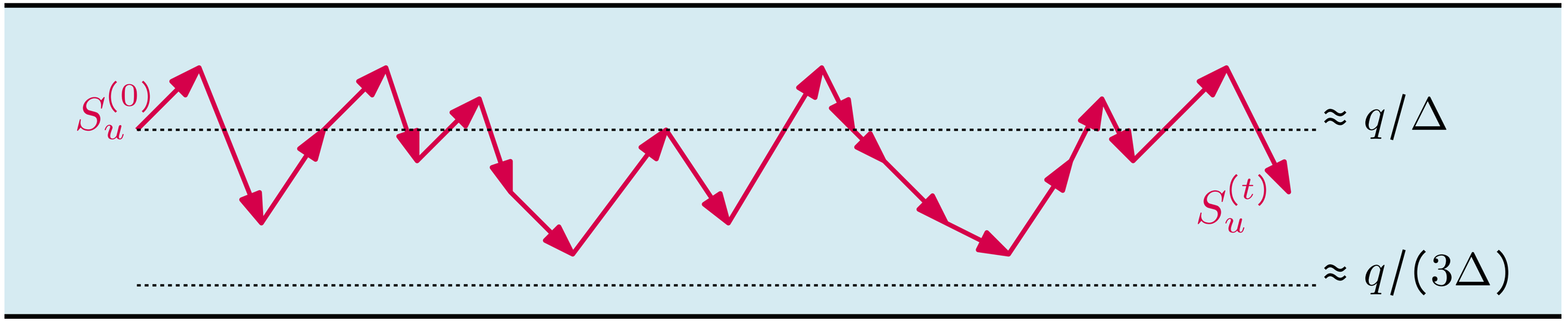
Martingale Process



(If no correlations: Chernoff bound)

$q :\approx \Delta^{3/4}$

Martingale Process

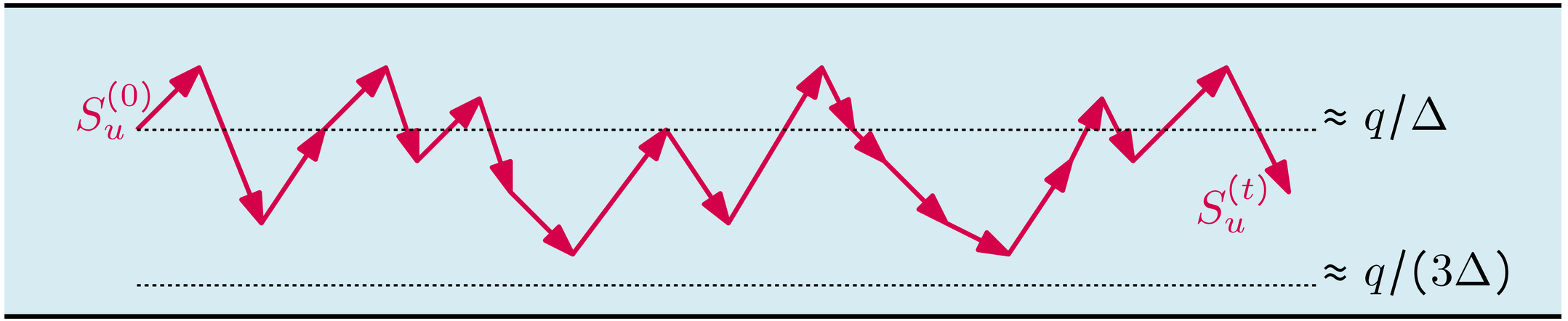


(If no correlations: Chernoff bound) $q :\approx \Delta^{3/4}$

Freedman's Inequality:

- Martingale $\mathbb{E}[S_u^{(t+1)} - S_u^{(t)} \mid S_u^{(1)}, S_u^{(2)}, \ldots, S_u^{(t)}] = 0$
- Step size $|S_u^{(t+1)} - S_u^{(t)}| \leq A$
- Observed variance $\sum_t \mathbb{E}\left[(S_u^{(t+1)} - S_u^{(t)})^2 \mid S_u^{(1)}, \ldots, S_u^{(t)}\right] \leq \sigma^2$

Martingale Process



(If no correlations: Chernoff bound) $\quad q :\approx \Delta^{3/4}$

Freedman's Inequality:

- Martingale $\mathbb{E}[S_u^{(t+1)} - S_u^{(t)} \mid S_u^{(1)}, S_u^{(2)}, \ldots, S_u^{(t)}] = 0$
- Step size $|S_u^{(t+1)} - S_u^{(t)}| \le A$
- Observed variance $\sum_t \mathbb{E}\left[(S_u^{(t+1)} - S_u^{(t)})^2 \mid S_u^{(1)}, \ldots, S_u^{(t)}\right] \le \sigma^2$

$$\Longrightarrow \Pr\left[|S_u^{(t)} - S_u^{(0)}| \ge \varepsilon\right] \le 2\exp\left(-\frac{\varepsilon^2}{2(\sigma^2 + A\varepsilon/3)}\right)$$

Fair Matching Result

Main Technical Theorem:

There is an online algorithm which outputs a random matching M so that

$$\Pr[e \in M] \geq \frac{1}{\Delta + q} \ \forall e \in E, \qquad \text{where } q = O(\Delta^{3/4}\sqrt{\log \Delta})$$

Fair Matching Result

Main Technical Theorem:

There is an online algorithm which outputs a random matching M so that

$$\Pr[e \in M] \geq \frac{1}{\Delta + q} \ \forall e \in E, \qquad \text{where } q = O(\Delta^{3/4}\sqrt{\log \Delta})$$

- This result has no requirement on Δ

Summary

Extensions

Open problems

Summary

Summary

- For low-degree graphs $(2\Delta - 1)$-edge-coloring is optimal

Summary

- For low-degree graphs $(2\Delta - 1)$-edge-coloring is optimal
- Otherwise, online edge coloring is (nearly) “as easy as offline”:

Summary

- For low-degree graphs $(2\Delta - 1)$-edge-coloring is optimal
- Otherwise, online edge coloring is (nearly) “as easy as offline”:

Main Theorem: online $(1 + o(1))$-fair matching algorithm

Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

Summary

- For low-degree graphs $(2\Delta - 1)$-edge-coloring is optimal
- Otherwise, online edge coloring is (nearly) “as easy as offline”:

Main Theorem: online $(1 + o(1))$-fair matching algorithm

Corollary: online $(1 + o(1))\Delta$-edge-coloring algorithm when $\Delta = \omega(\log n)$

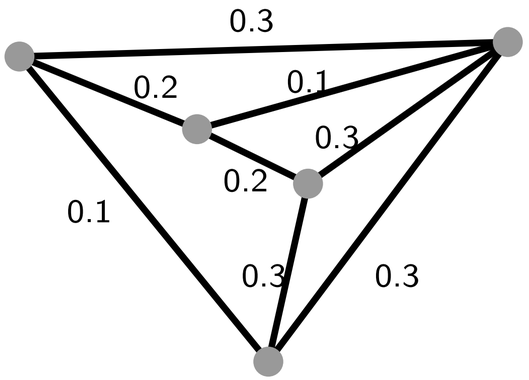
Key takeaway:

Formulate **quantity as a martingale**, bound stepsize and observed variance

Chernoff-type concentration

Extensions (1/3)
Fractional Matchings

Online Rounding of "Spread Out" Fractional Matchings:
Given (online) fractional matching $x \in \mathbb{R}^E$ satisfying $x_e \leq \varepsilon^5$,
output matching M so that $\Pr[e \in M] \geq (1-\varepsilon)x_e$.



$x_e := \frac{1}{\Delta}$ recovers fair matching theorem

Extensions (2/3)

List edge-coloring

Theorem:

There exists an online algorithm that computes an edge coloring which, with high probability, assigns each edge e a color from its list $L(e)$ (revealed online, with edge e), provided each list has sufficiently large size $(1 + o(1))\Delta$ and that $\Delta = \omega(\log n)$.

- Generalizes Kahn's seminal work from 96 to the online setting (albeit with worse bounds)

Extensions (3/3)

Local edge-coloring

Theorem:

There exists an online algorithm that computes an edge coloring assigning each edge $e = (u, v)$ a color from the set $\{1, 2, \ldots, d_{\max}(e) \cdot (1 + o(1))\}$ with high probability, where $d_{\max}(e) := \max\{\deg(u), \deg(v)\}$, provided that $d_{\max}(e) = \omega(\log n)$.

- Generalizes Christansen'23 work on local colorings (albeit with worse bounds)

Sharp Thresholds

Sharp Thresholds

[BSVW'25]

Theorem:

$$(1+o(1))\Delta\text{-colors sufficient for when } \Delta = \begin{cases} \omega(\log n) & \text{if deterministic} \\ \omega(\sqrt{\log n}) & \text{if randomized} \end{cases}$$

Sharp Thresholds

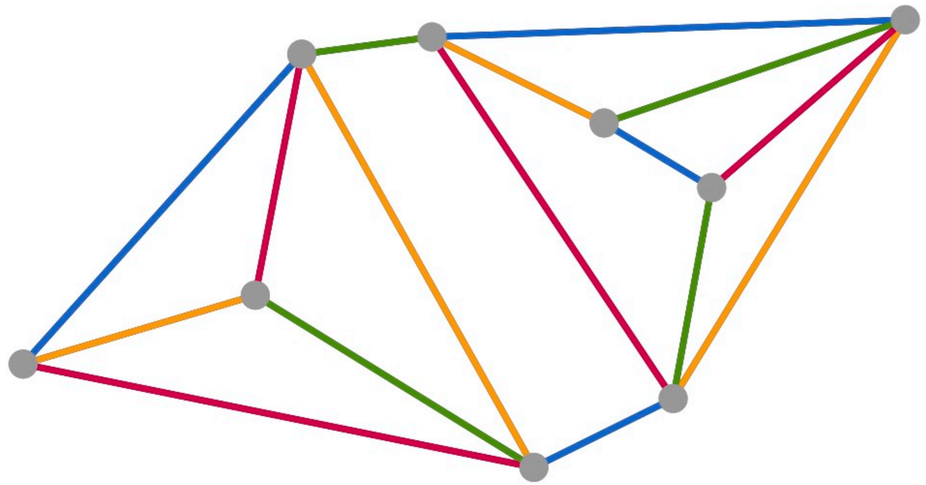
[BSVW'25]

Theorem:

$(1+o(1))\Delta$-colors sufficient for when $\Delta = \begin{cases} \omega(\log n) & \text{if deterministic} \\ \omega(\sqrt{\log n}) & \text{if randomized} \end{cases}$

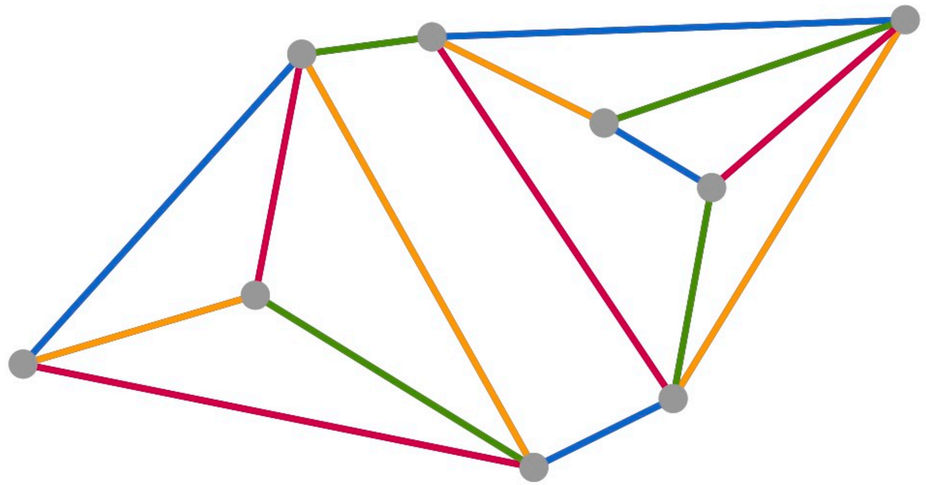
- Cannot go via fair matchings
- Interestingly, this stronger result cannot generalize to list edge coloring

One slide intuition



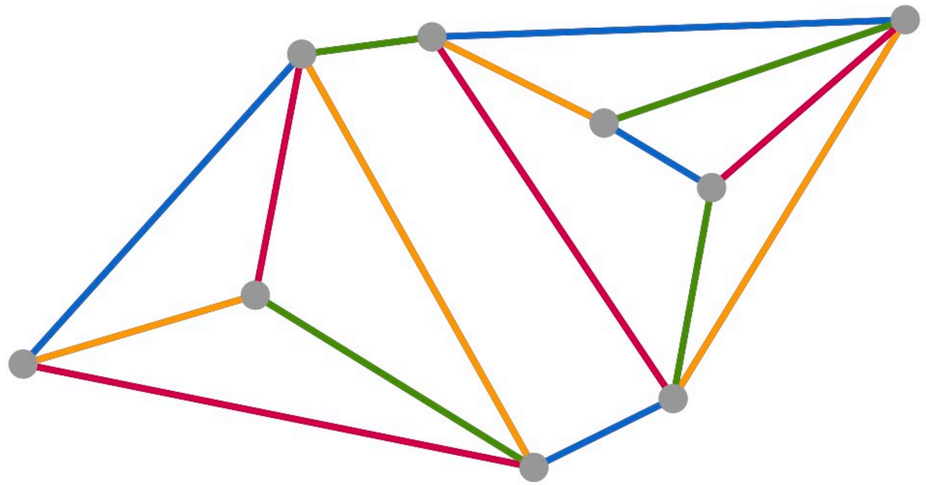
One slide intuition

- First result shows that every matching <=> every color is good with probability $e^{-\Delta}$ so we need to select $\Delta = \Omega(\log n)$



One slide intuition

- First result shows that every matching <=> every color is good with probability $e^{-\Delta}$ so we need to select $\Delta = \Omega(\log n)$
- Instead show that the probability that more than a vanishing number of colors $\epsilon\Delta$ fail is at most $\left(e^{-\Delta}\right)^{\epsilon\Delta}$ so we can select $\Delta = \Omega(\sqrt{\log n})$



Open Problems (1/3)

Randomized Greedy

When $e_t = (u, v)$ arrives:

Color e_t with a **uniformly at random** color in $\mathscr{C}$ available to both u and v

Open Problems (1/3)

Randomized Greedy

- Let $\mathscr{C}$ be a pallette of $(1 + o(1))\Delta$ colors

When $e_t = (u, v)$ arrives:

Color e_t with a **uniformly at random** color in $\mathscr{C}$ available to both u and v

Open Problems (1/3)

Randomized Greedy

- Let $\mathscr{C}$ be a pallette of $(1 + o(1))\Delta$ colors

> When $e_t = (u, v)$ arrives:
>
> Color e_t with a **uniformly at random** color in $\mathscr{C}$ available to both u and v

- Does randomized greedy work w.h.p when $\Delta = \omega(\log n)$?

Open Problems (1/3)
Randomized Greedy

- Let $\mathscr{C}$ be a pallette of $(1 + o(1))\Delta$ colors

> When $e_t = (u, v)$ arrives:
>
> Color e_t with a **uniformly at random** color in $\mathscr{C}$ available to both u and v

- Does randomized greedy work w.h.p when $\Delta = \omega(\log n)$?
- Simulations indicate less good concentration than our approach but may work against weaker oblivious adversary

Open Problems (1/3)
Randomized Greedy

- Let $\mathscr{C}$ be a pallette of $(1 + o(1))\Delta$ colors

> When $e_t = (u, v)$ arrives:
>
> Color e_t with a **uniformly at random** color in $\mathscr{C}$ available to both u and v

- Does randomized greedy work w.h.p when $\Delta = \omega(\log n)$?
- Simulations indicate less good concentration than our approach but may work against weaker oblivious adversary
- Very recent positive results for special cases by Dudeja, Goswami, Saks'24

Open Problems (2/3)

Efficient Deterministic algorithms

> *... An interesting open problem is whether better bounds [than greedy's] can be achieved for graphs whose maximal degree is larger [than* $\log n$*]. This seems less plausible in the deterministic case, but perhaps one can devise a randomized algorithm that would edge color a graph better than the greedy algorithm for high degree graphs.*

— Bar-Noy, Motwani, Naor'92

Open Problems (2/3)

Efficient Deterministic algorithms

> *... An interesting open problem is whether better bounds [than greedy's] can be achieved for graphs whose maximal degree is larger [than* $\log n$*]. This seems less plausible in the deterministic case, but perhaps one can devise a randomized algorithm that would edge color a graph better than the greedy algorithm for high degree graphs.*

— Bar-Noy, Motwani, Naor'92

Open Problems (2/3)

Efficient Deterministic algorithms

> *... An interesting open problem is whether better bounds [than greedy's] can be achieved for graphs whose maximal degree is larger [than* $\log n$*]. This seems less plausible in the deterministic case, but perhaps one can devise a randomized algorithm that would edge color a graph better than the greedy algorithm for high degree graphs.*
>
> — Bar-Noy, Motwani, Naor'92

- We now know that deterministic algortihms are possible

Open Problems (2/3)

Efficient Deterministic algorithms

> *... An interesting open problem is whether better bounds [than greedy's] can be achieved for graphs whose maximal degree is larger [than* $\log n$*]. This seems less plausible in the deterministic case, but perhaps one can devise a randomized algorithm that would edge color a graph better than the greedy algorithm for high degree graphs.*
>
> — Bar-Noy, Motwani, Naor'92

- We now know that deterministic algortihms are possible
- However, proof very indirect and no explicit/efficient deterministic algorithm

Open Problems (2/3)

Efficient Deterministic algorithms

> *... An interesting open problem is whether better bounds [than greedy's] can be achieved for graphs whose maximal degree is larger [than* $\log n$*]. This seems less plausible in the deterministic case, but perhaps one can devise a randomized algorithm that would edge color a graph better than the greedy algorithm for high degree graphs.*
>
> — Bar-Noy, Motwani, Naor'92

- We now know that deterministic algortihms are possible
- However, proof very indirect and no explicit/efficient deterministic algorithm
  - Randomized against adaptive => deterministic

Open Problems (3/3)

Tight guarantees on Δ

Open Problems (3/3)

Tight guarantees on Δ

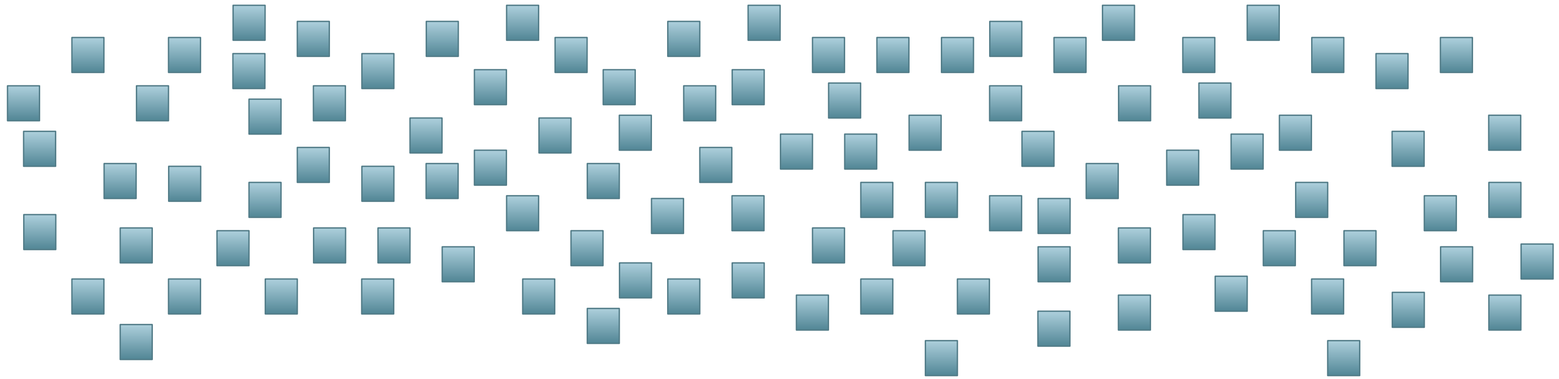
- What are the tight dependency on Δ in $(1 + o(1))\Delta$?

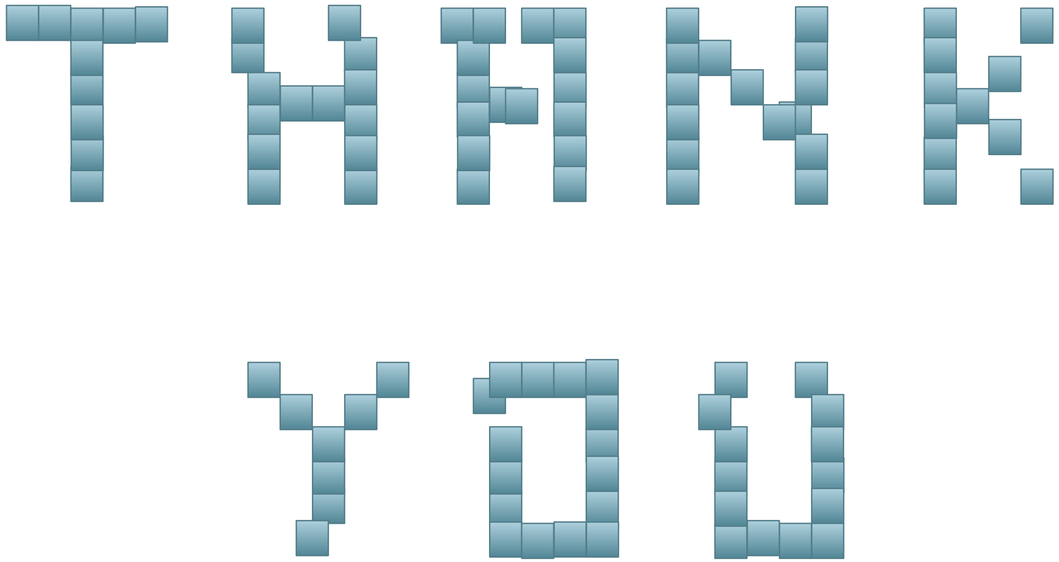
Open Problems (3/3)

Tight guarantees on Δ

- What are the tight dependency on Δ in $(1 + o(1))\Delta$?

- Lower bound is $\Omega(\sqrt{\Delta})$
- Our proof yields $O(\Delta^{3/4} \log \Delta)$




THANK
YOU