

Minimizing Congestion in General Networks

(Harald Räcke FOCs 2002)

Maverick Woo

Online Routing Problem

Given a connected graph $G = (V, E)$, with $|V| = n$.

Each link e has a positive bandwidth $b(e)$.

A sequence of routing request $\sigma_1, \sigma_2, \dots$ comes in, each of the form (s_i, t_i) for σ_i .

Requirement:

as each σ_i comes in, specify a path from s_i to t_i

Goal:

minimize the congestion of the network

Congestion

For a link,

load

number of requests routed over a link

relative load

load / bandwidth

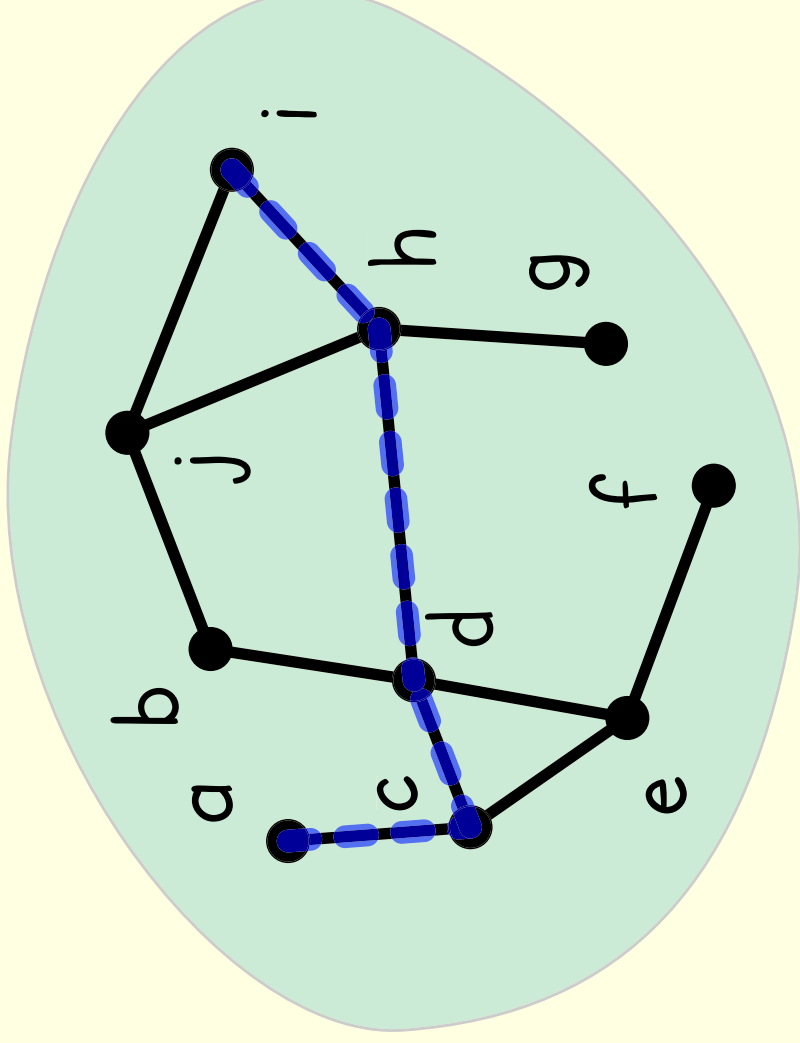
For the network,

congestion

maximum relative load over all links

Example Network

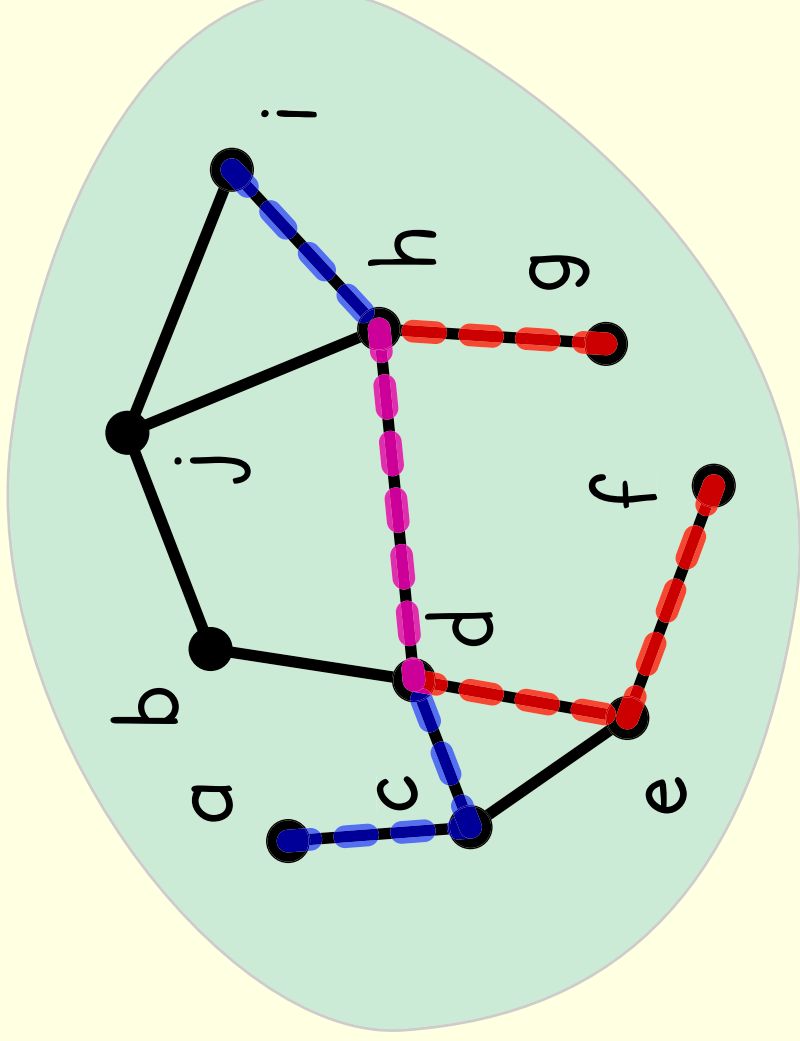
$\sigma_1: (a,i)$



Example Network

$\sigma_1: (a,i)$

$\sigma_2: (f,g)$

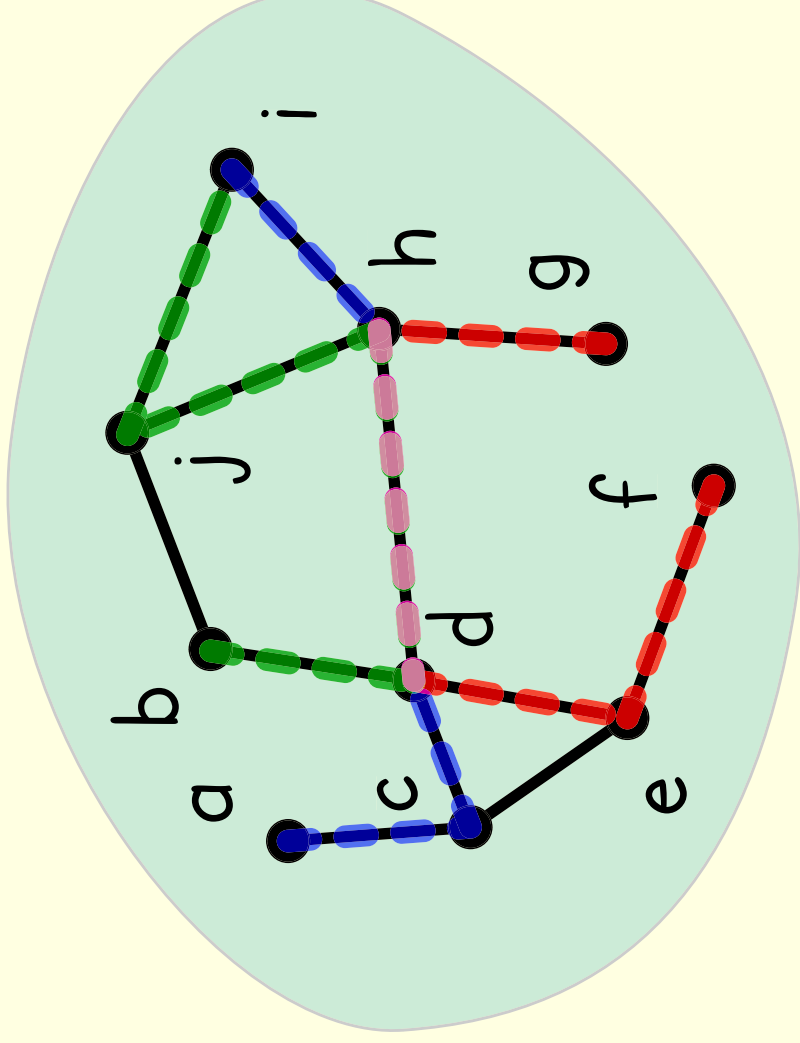


Example Network

$\sigma_1: (a,i)$

$\sigma_2: (f,g)$

$\sigma_3: (i,b)$



Online A vs. Offline OPT

Suppose OPT is the optimal offline algorithm.

Given an input sequence σ , denote the congestions by $C_A(\sigma)$ and $C_{\text{OPT}}(\sigma)$.

A is c -competitive if

$$\exists k : \forall \sigma : C_A(\sigma) \leq c \cdot C_{\text{OPT}}(\sigma) + k.$$

We assume that A may use randomization but the adversary is oblivious.

Previous Results

Aspnes, Azar, Fiat, Plotkin and Waarts STOC 1993

- centralized $O(\log n)$ - competitive

Awerbuch and Azar

FOCS 1994

- decisions can be made locally (but needs global state)

Previous Results

Aspnes, Azar, Fiat, Plotkin and Waarts STOC 1993

- centralized $O(\log n)$ - competitive
- hence easy to know current load of a link

Awerbuch and Azar

FOCS 1994

- decisions can be made locally (but needs global state)
- now knowing the load of a link is difficult

Previous Results

Aspnes, Azar, Fiat, Plotkin and Waarts STOC 1993

- centralized $O(\log n)$ - competitive
- hence easy to know current load of a link
- $\Omega(\log n)$ lower bound (holds for randomized)

Awerbuch and Azar

FOCS 1994

- decisions can be made locally (but needs global state)
- now knowing the load of a link is difficult

Towards Distributed Routing

Do we really need to know the current load?

Towards Distributed Routing

Do we really need to know the current load? **No.**

Towards Distributed Routing

Do we really need to know the current load? **No.**

A routing algorithm is **oblivious** if the path selected for a request is independent of the path for any other requests.

Towards Distributed Routing

Do we really need to know the current load? **No.**

A routing algorithm is **oblivious** if the path selected for a request is independent of the path for any other requests. Hence the path for σ_i only depends on s_i , t_i and perhaps some random bits too.

Since all decisions are made at end-points, any oblivious routing algorithm can easily be implemented in an distributed fashion.

Previous Results - Oblivious

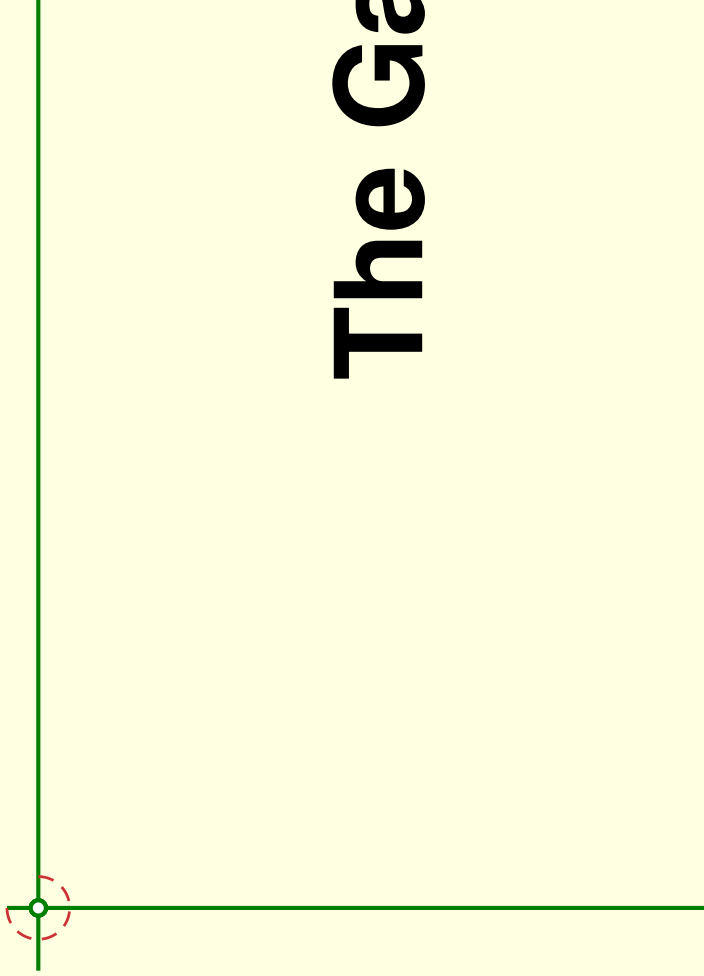
Lower bounds: “deterministic oblivious is hopeless”

- Borodin and Hopcroft $\Omega(\sqrt{n}/d^{3/2})$ JCSS 1985
- Kalamanis, Krizanc and Tsantilas SPAA 1990

Upper bounds: “for special topologies only”

- Valiant, Brebner STOC 1981
- Aiello, Leighton, Maggs and Newman MST 1991
- Vöcking STOC 2001

The Game Plan



“Realize there is no tree.”

Given a graph $G = (V, E)$, first produce a decomposition tree $T_G = (V_t, E_t)$.

“Realize there is no tree.”

Given a graph $G = (V, E)$, first produce a decomposition tree $T_G = (V_t, E_t)$.

When a request on G comes in, fantasize how we will execute it on T_G to obtain a tree path.

In the Matrix, T_G is simulating G .

“Realize there is no tree.”

Given a graph $G = (V, E)$, first produce a decomposition tree $T_G = (V_t, E_t)$.

When a request on G comes in, fantasize how we will execute it on T_G to obtain a tree path.

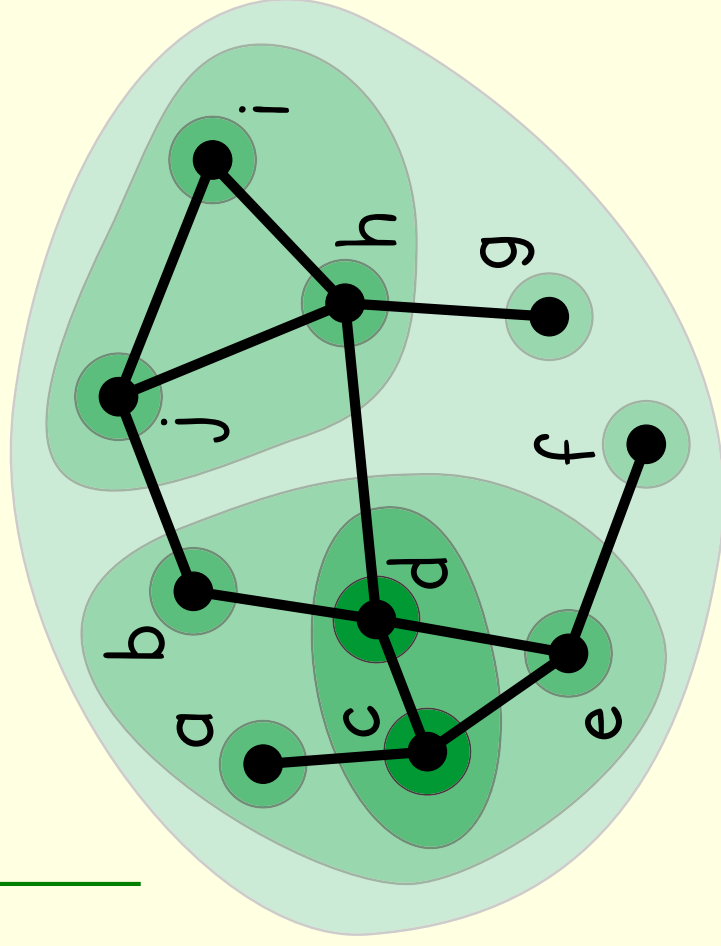
In the Matrix, T_G is simulating G .

Take the tree path on T_G , figure out how it looks like in G and do the actual routing.

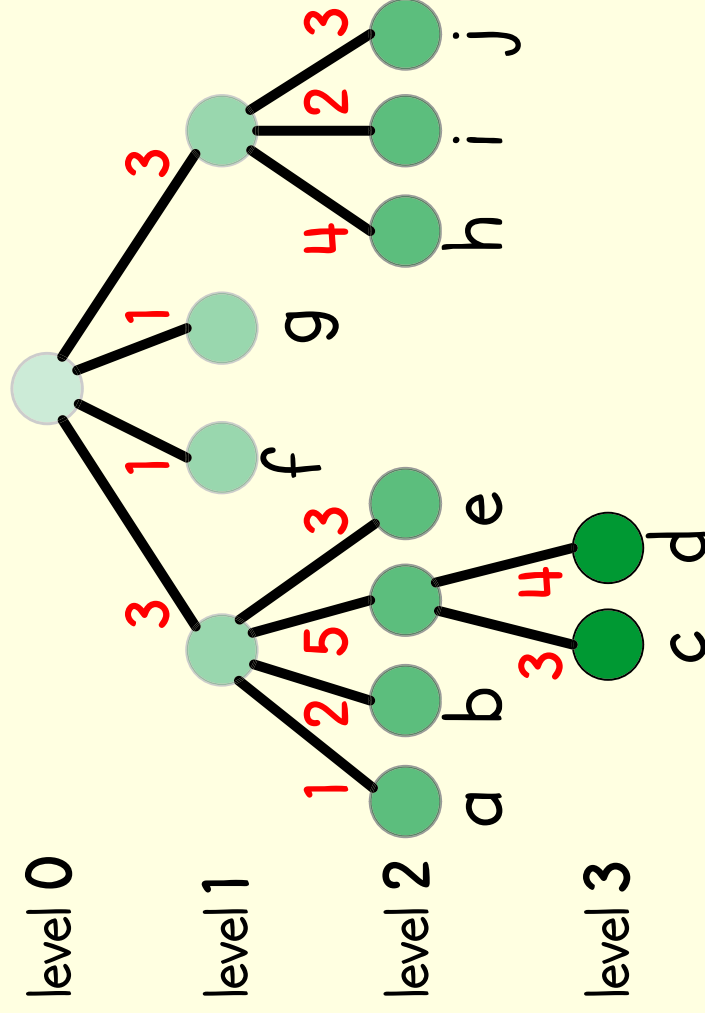
In the Real World, G is simulating T_G .

The Decomposition Tree

$$G = (V, E)$$



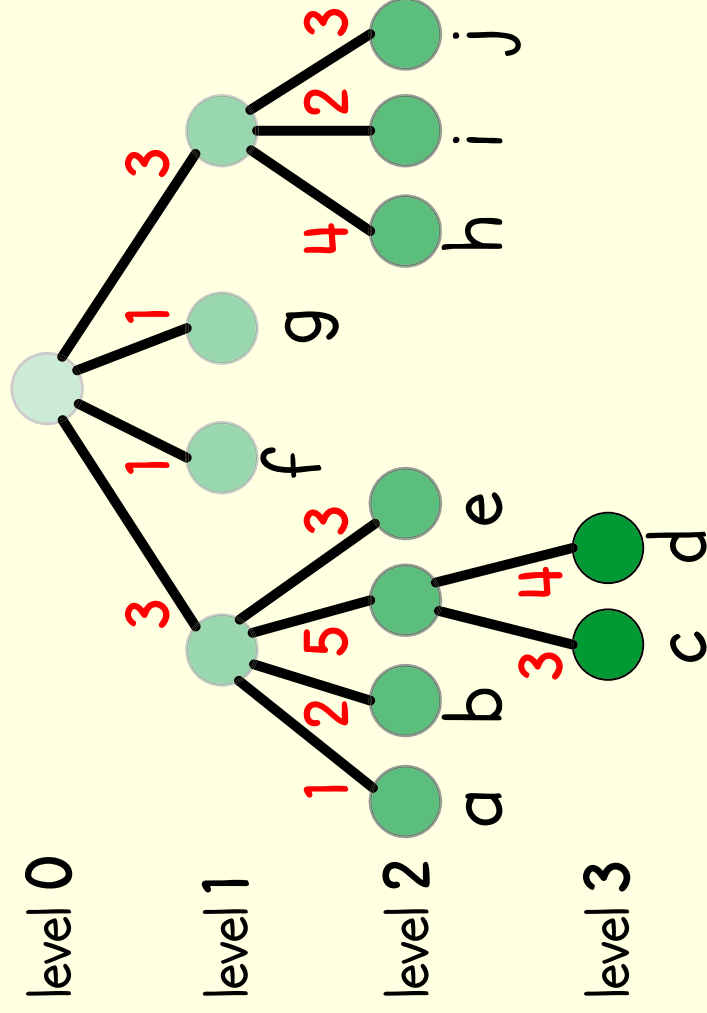
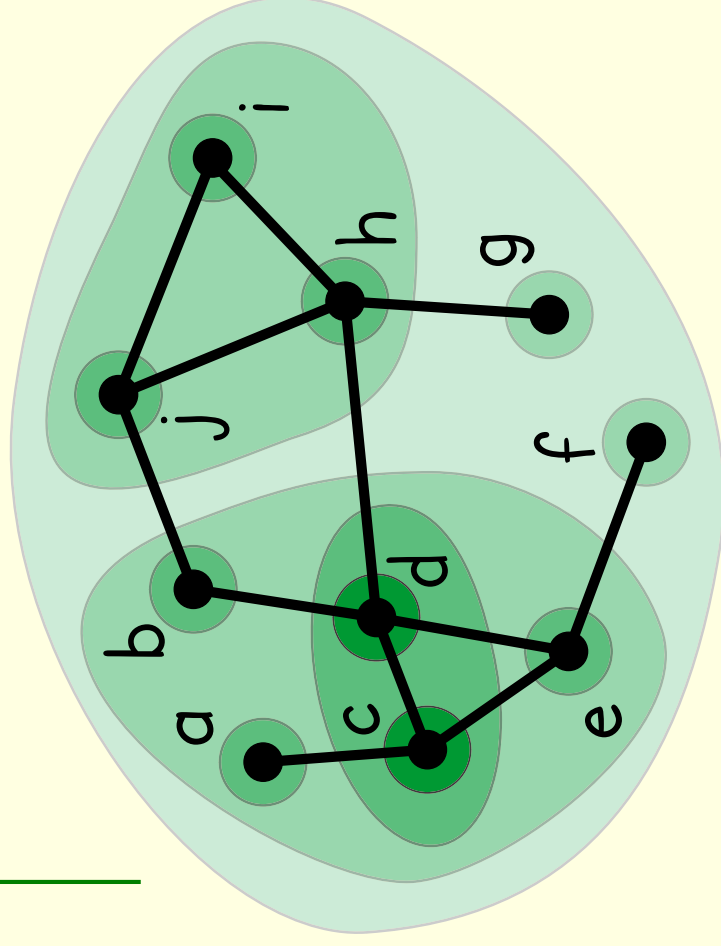
$$T_G = (V_t, E_t)$$



cap(X, Y) and out(X)

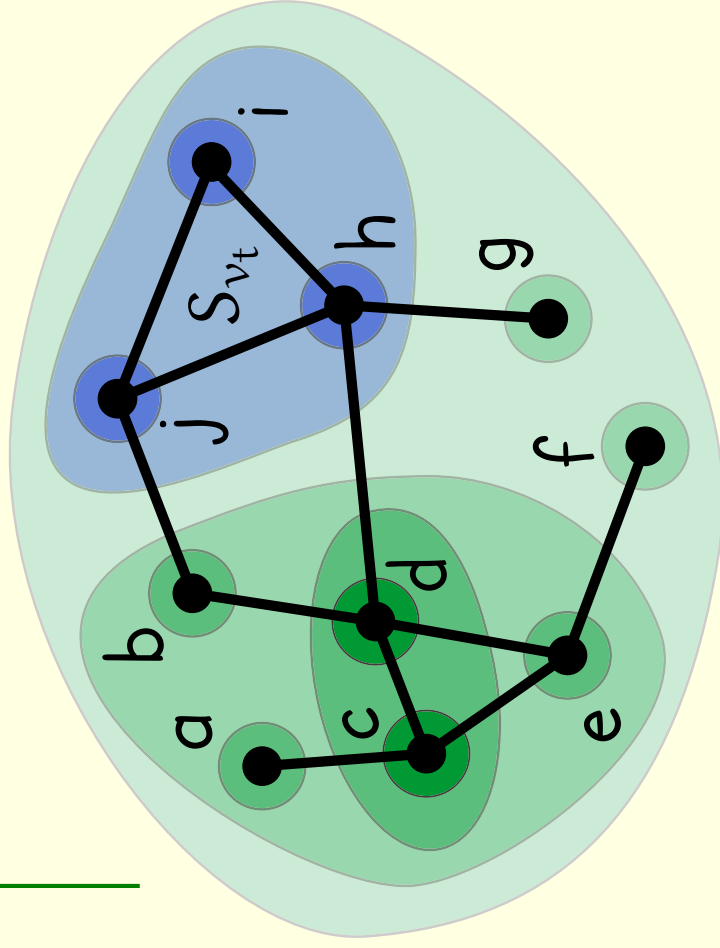
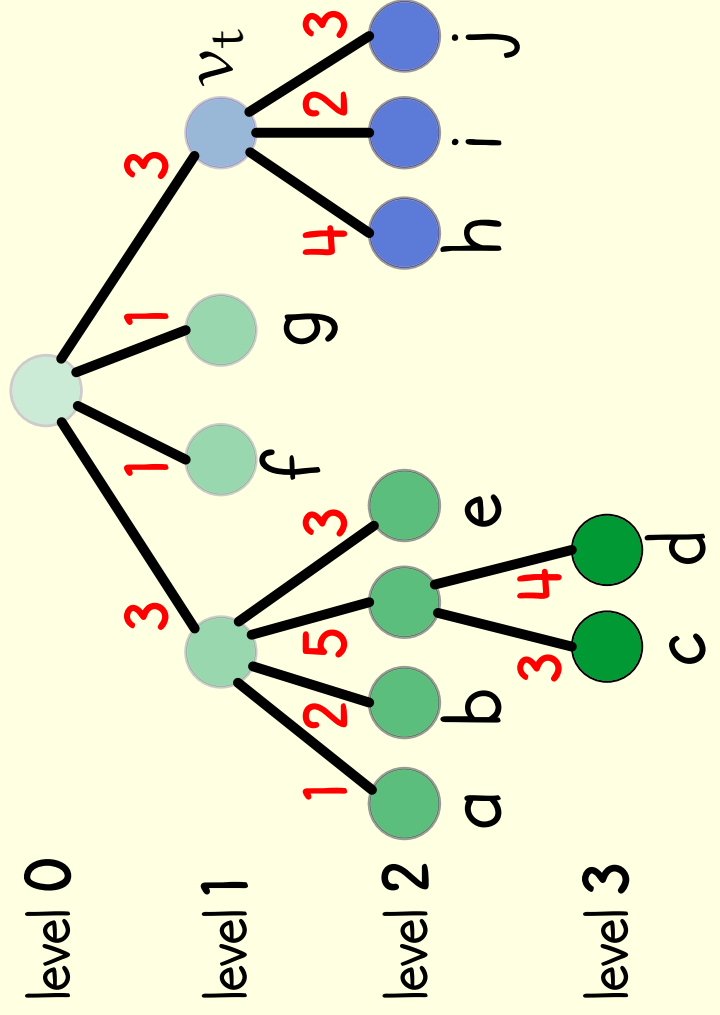
$$\text{cap}(X, Y) := \sum_{x \in X, y \in Y} b((x, y))$$

$$\text{out}(X) := \text{cap}(X, V \setminus X)$$



S_{v_t} : The Cluster of v_t

Everything with a subscript of t is on the tree.

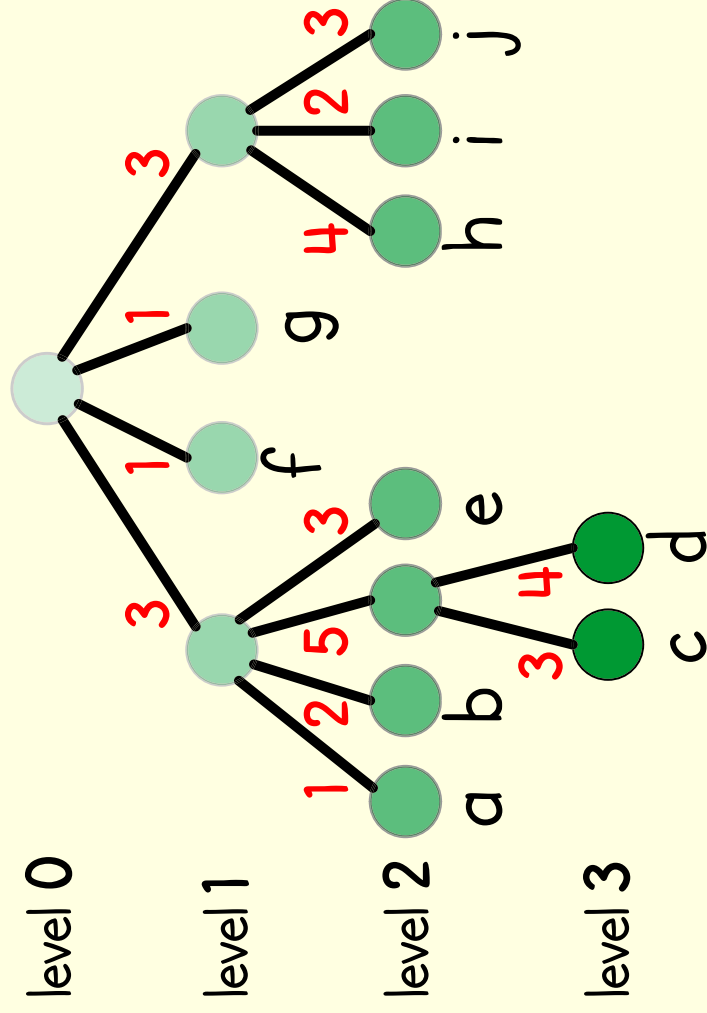
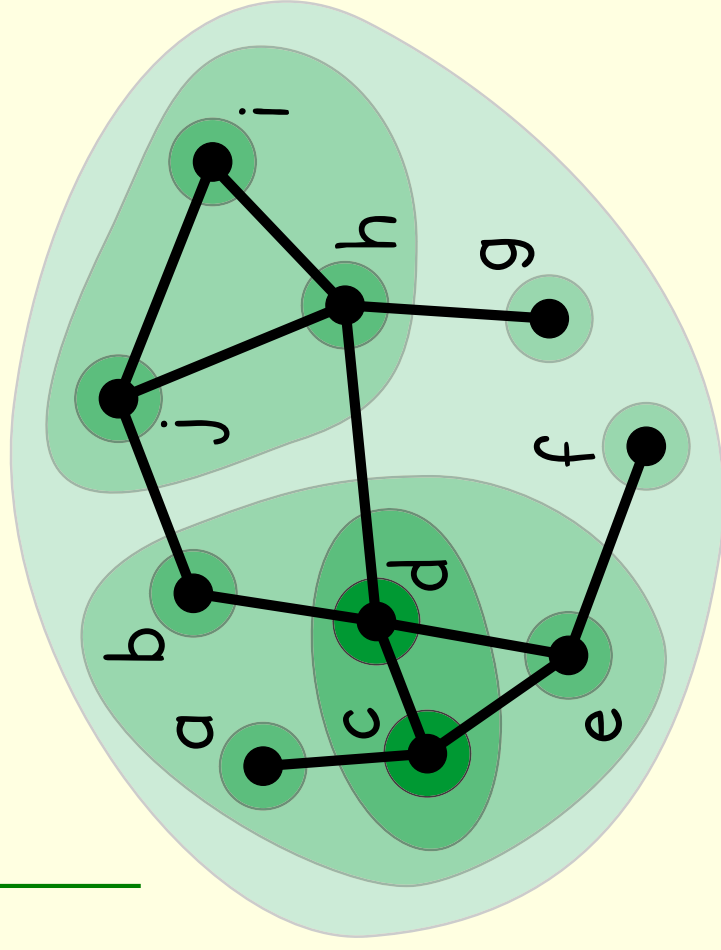




Simone: Simulating G on T_G

Simulating G on T_G

Straightforward: there is a unique tree path.



T_G can easily simulate G

Theorem Given any σ for an online problem on G with congestion C , the straightforward simulation on T_G yields congestion $C_t \leq C$.

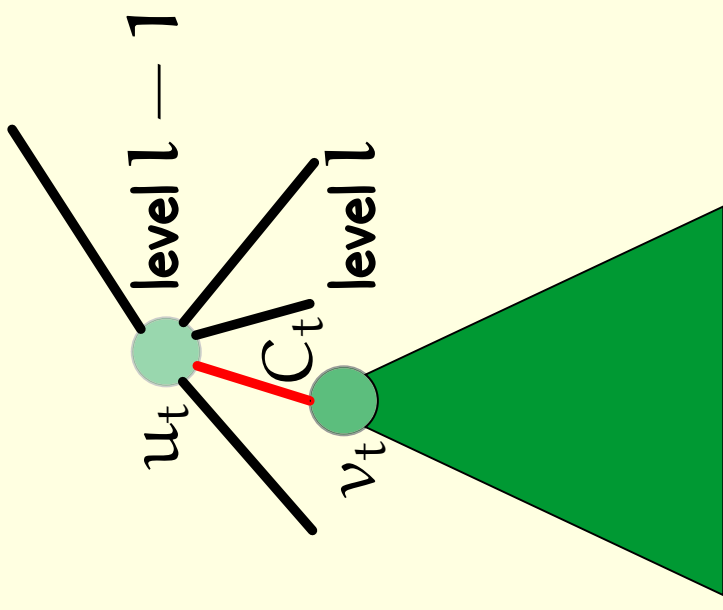
T_G can easily simulate G

Theorem Given any σ for an online problem on G with congestion C , the straightforward simulation on T_G yields congestion $C_t \leq C$.

Proof Suppose $e_t = (u_t, v_t)$ has relative load C_t when processing σ on T_G . WLOG assume u_t is on level $l - 1$ and v_t is on level l .

Fact: absolute load of e_t is

$$C_t \cdot b_t(e_t)$$

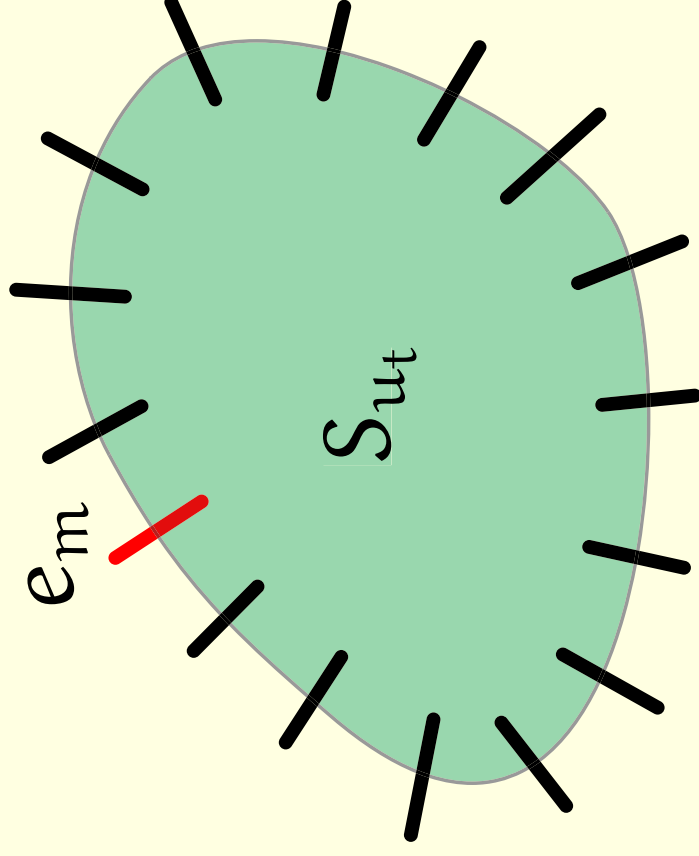


What are the messages?

Fact: any message that crosses e_t in T_G must cross the boundary of S_{v_t}

Let the boundary be e_1, \dots, e_m , each with relative load c_1, \dots, c_m respectively.

WLOG
assume e_m be the link with the maximum relative load.



How many messages?

Consider the total number of messages across boundary.

$$\begin{aligned}\sum_{i=1}^m c_i \cdot b(e_i) &\leq \sum_{i=1}^m c_m \cdot b(e_i) \\ &= c_m \cdot \sum_{i=1}^m b(e_i) \\ &= c_m \cdot b_t(e_t)\end{aligned}$$

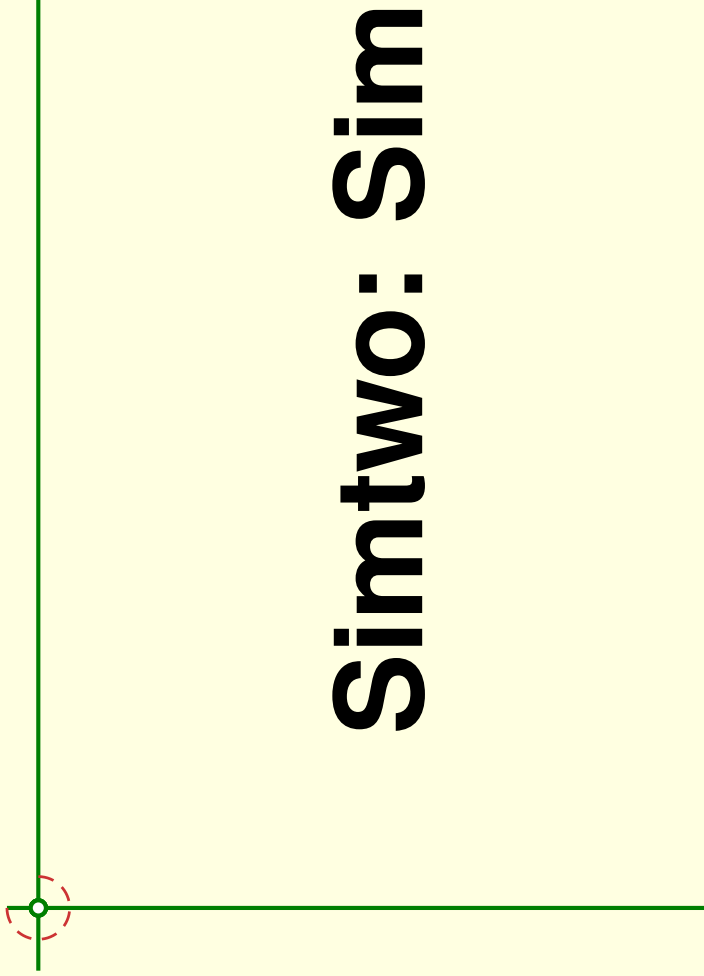
Recall: absolute load of e_t is $C_t \cdot b_t(e_t)$

Conclusion

$$\therefore C_t \leq C_m \leq C \quad \blacksquare$$

Theorem Given any σ for an online problem on G with congestion C , the straightforward simulation on T_G yields congestion $C_t \leq C$.

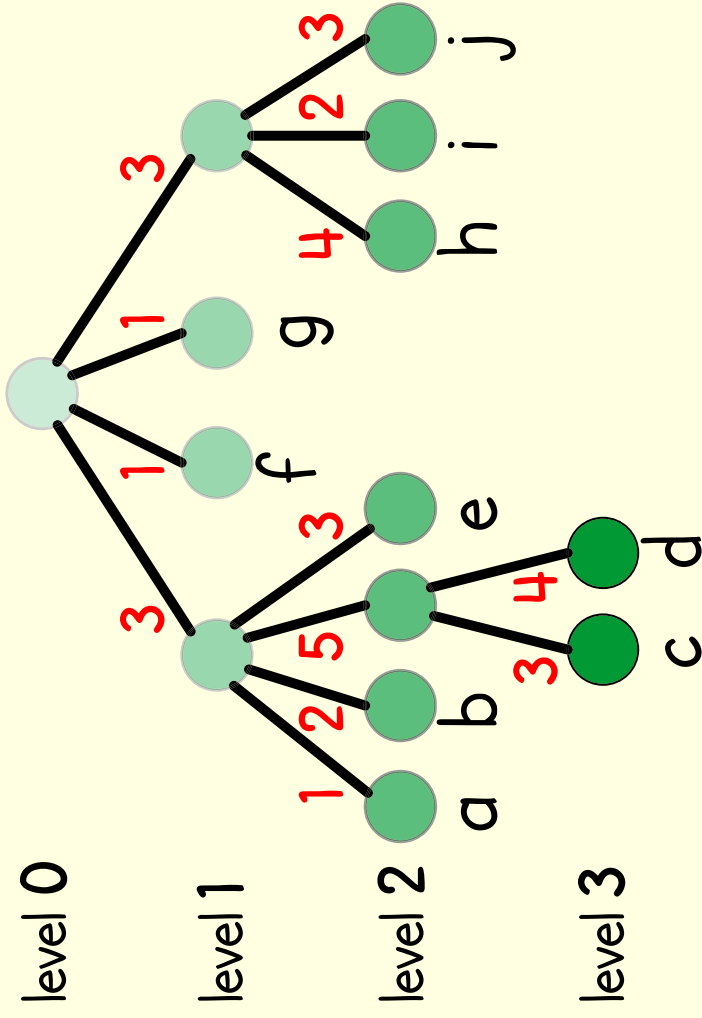
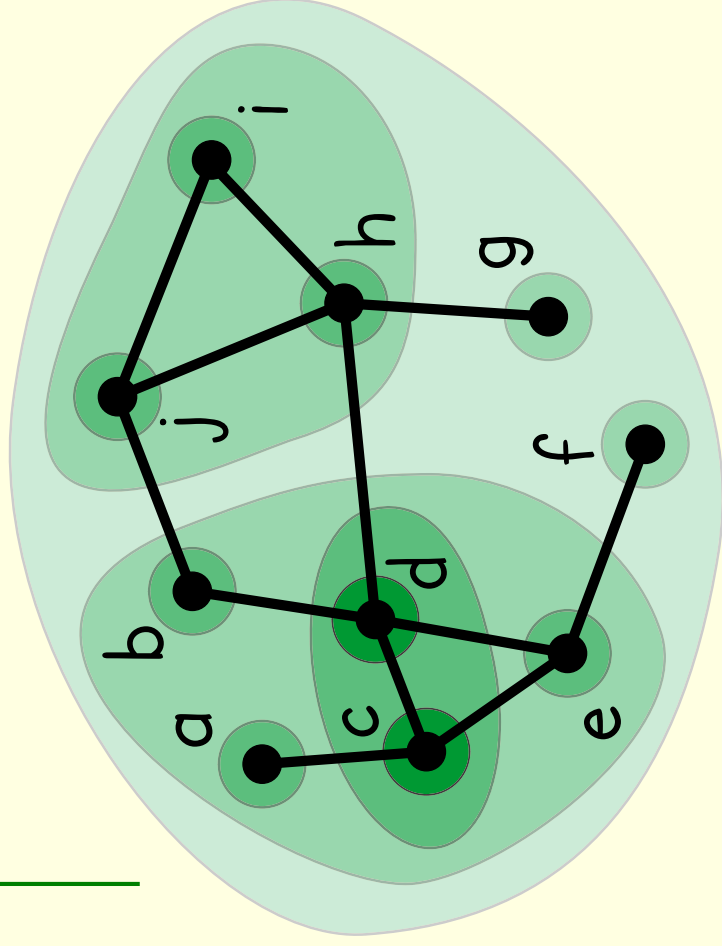
(This is true for any decomposition tree!)



Simtwo: Simulating T_G on G

Simulating T_G on G

Tough!!! The internal tree nodes are absent in the graph!

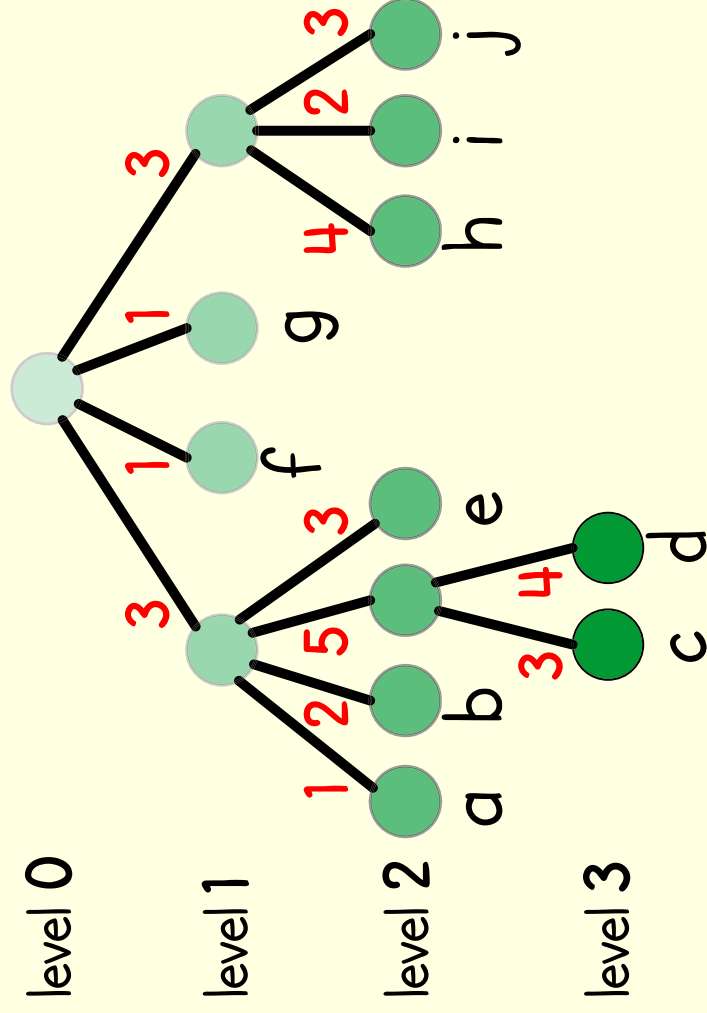
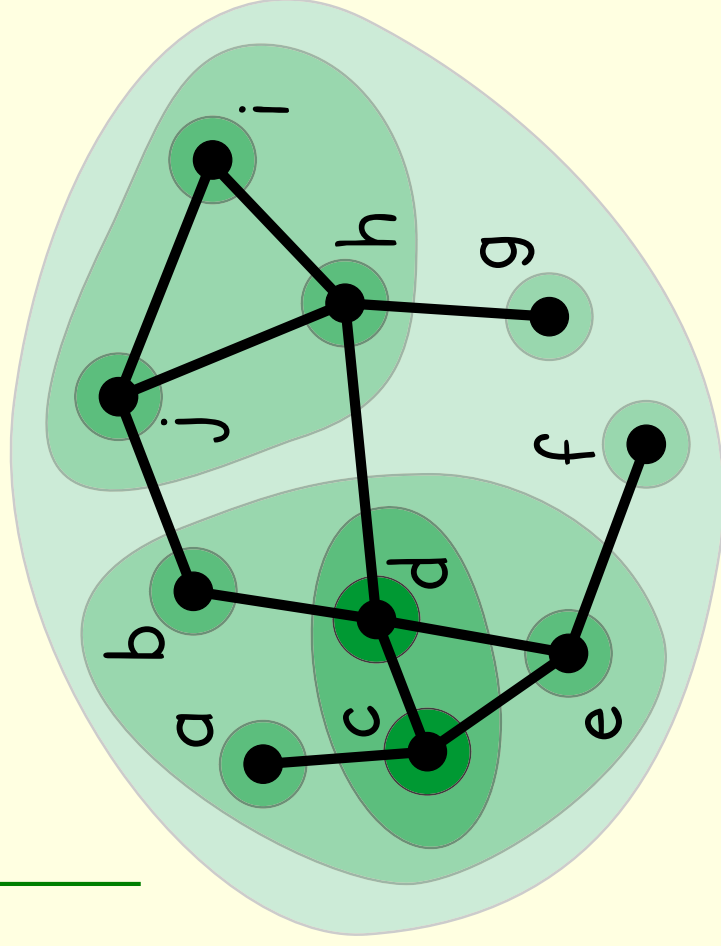


For The Moment

Assume G can simulate T_G such that

$$C \leq c \cdot C_t + K.$$

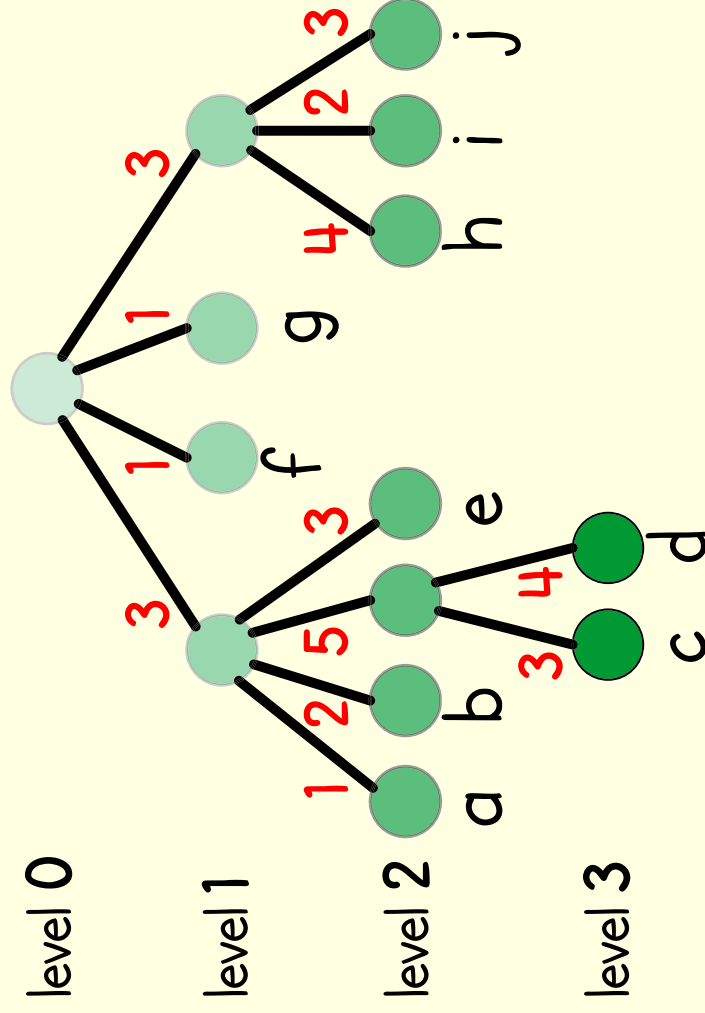
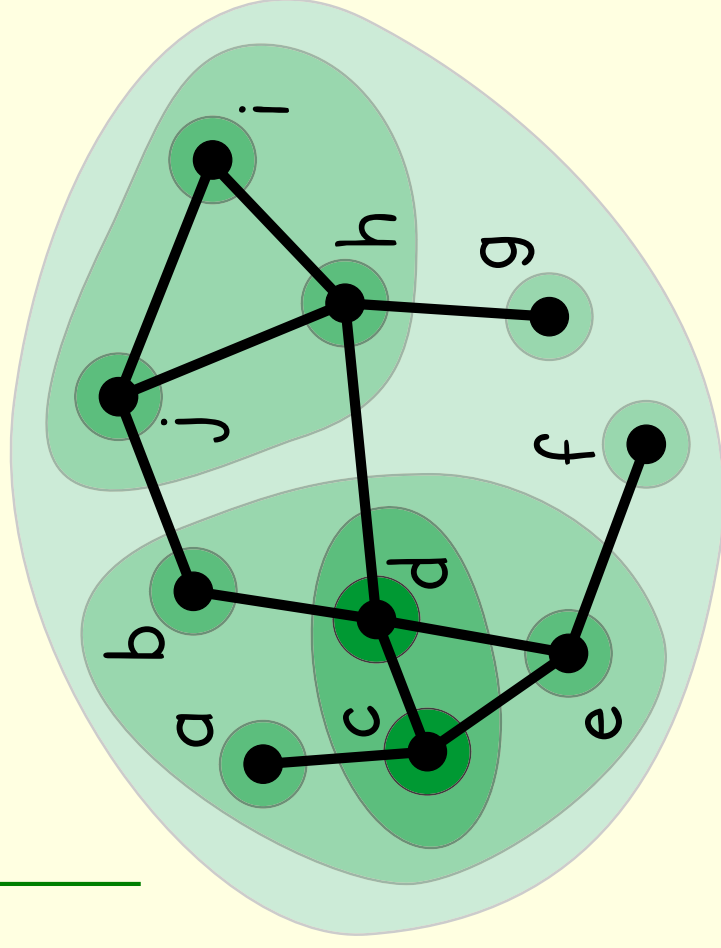
We surely hope c is small in our problem.



Competitiveness Assumption

We also need to assume there is a C_t -competitive algorithm for the online problem on T_G such that

$$C_{\text{onl}}(T_G) \leq C_t \cdot C_{\text{opt}}(T_G) + K_t.$$



A Man, A Plan, A Canal, Panama!

A Man, A Plan, A Canal, Panama!

Simone:

$$C_{\text{opt}}(T_G) \leq C_{\text{opt}}(G)$$

C_t -competitive: $C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(T_G) + K_t$

A Man, A Plan, A Canal, Panama!

Simone:

$$C_{\text{opt}}(T_G) \leq C_{\text{opt}}(G)$$

C_t -competitive: $C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(T_G) + K_t$

$$\therefore C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(T_G) + K_t$$

A Man, A Plan, A Canal, Panama!

Simone:

$$C_{\text{opt}}(T_G) \leq C_{\text{opt}}(G)$$

C_t -competitive: $C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(T_G) + K_t$

$$\therefore C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(G) + K_t$$

A Man, A Plan, A Canal, Panama!

Simone:

$$C_{\text{opt}}(T_G) \leq C_{\text{opt}}(G)$$

c_t -competitive: $C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(T_G) + K_t$

$$\therefore C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(G) + K_t$$

Simtwo: $C_{\text{onl}}(G) \leq c \cdot C_{\text{onl}}(T_G) + K$

A Man, A Plan, A Canal, Panama!

Simone:

$$C_{\text{opt}}(T_G) \leq C_{\text{opt}}(G)$$

C_t -competitive: $C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(T_G) + K_t$

$$\therefore C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(G) + K_t$$

Simtwo: $C_{\text{onl}}(G) \leq c \cdot C_{\text{onl}}(T_G) + K$

$$\therefore C_{\text{onl}}(G) \leq c \cdot$$

$$+ K$$

A Man, A Plan, A Canal, Panama!

Simone:

$$C_{\text{opt}}(T_G) \leq C_{\text{opt}}(G)$$

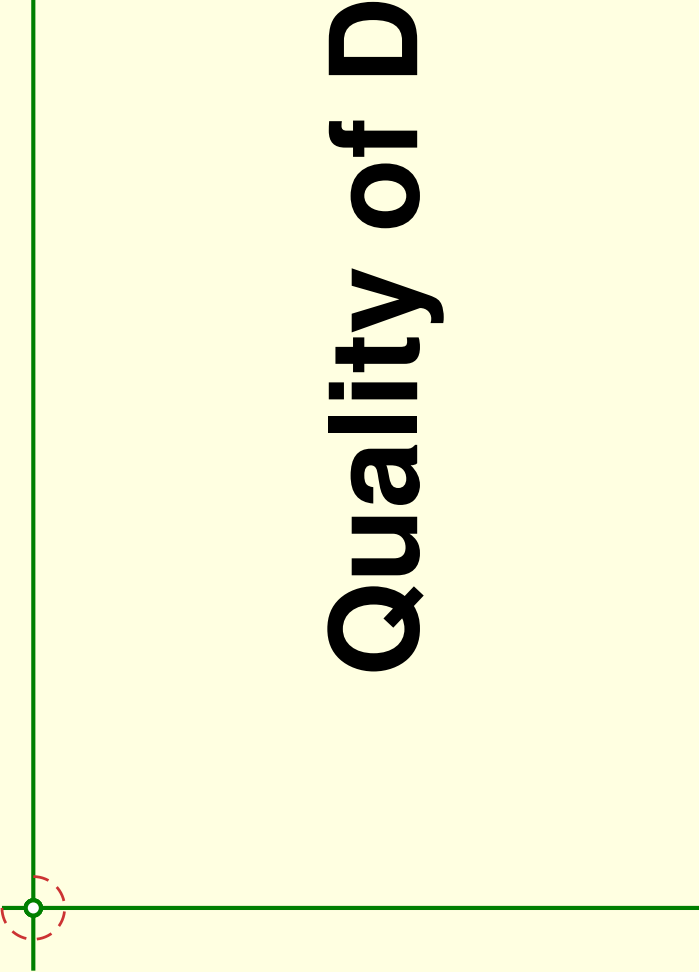

c_t -competitive: $C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(T_G) + K_t$

$$\therefore C_{\text{onl}}(T_G) \leq c_t \cdot C_{\text{opt}}(G) + K_t$$

Simtwo: $C_{\text{onl}}(G) \leq c \cdot C_{\text{onl}}(T_G) + K$

$$\begin{aligned} \therefore C_{\text{onl}}(G) &\leq c \cdot (c_t \cdot C_{\text{opt}}(G) + K_t) + K \\ &= c \cdot c_t \cdot C_{\text{opt}}(G) + c \cdot K_t + K \end{aligned}$$

Now we have an $*\text{almost}*(c \cdot c_t)$ -competitive online algorithm for the graph!



Quality of Decomposition

Weight Function $w_1(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^l} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

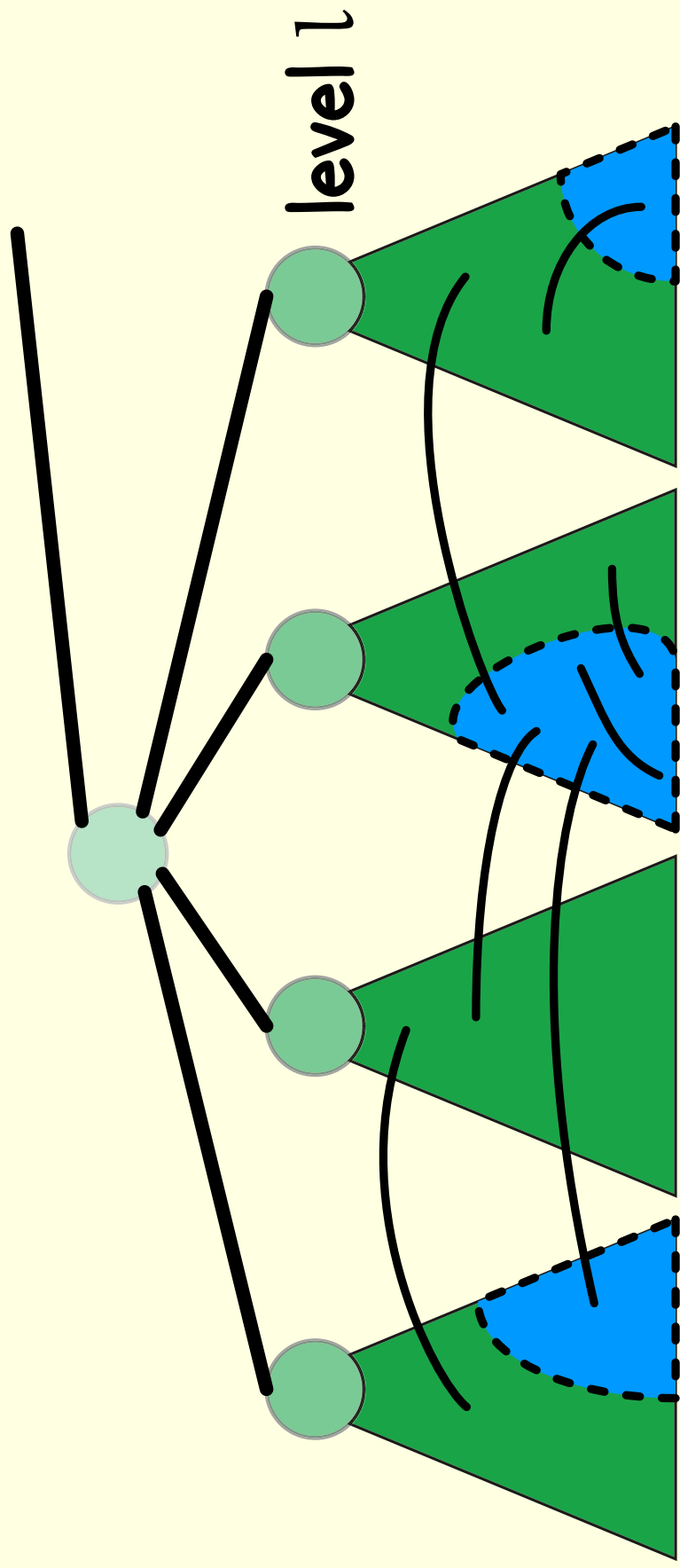
sum of the degrees on
each vertices

minus
the degrees that stay
within the same cluster

with respect to a particular decomposition.

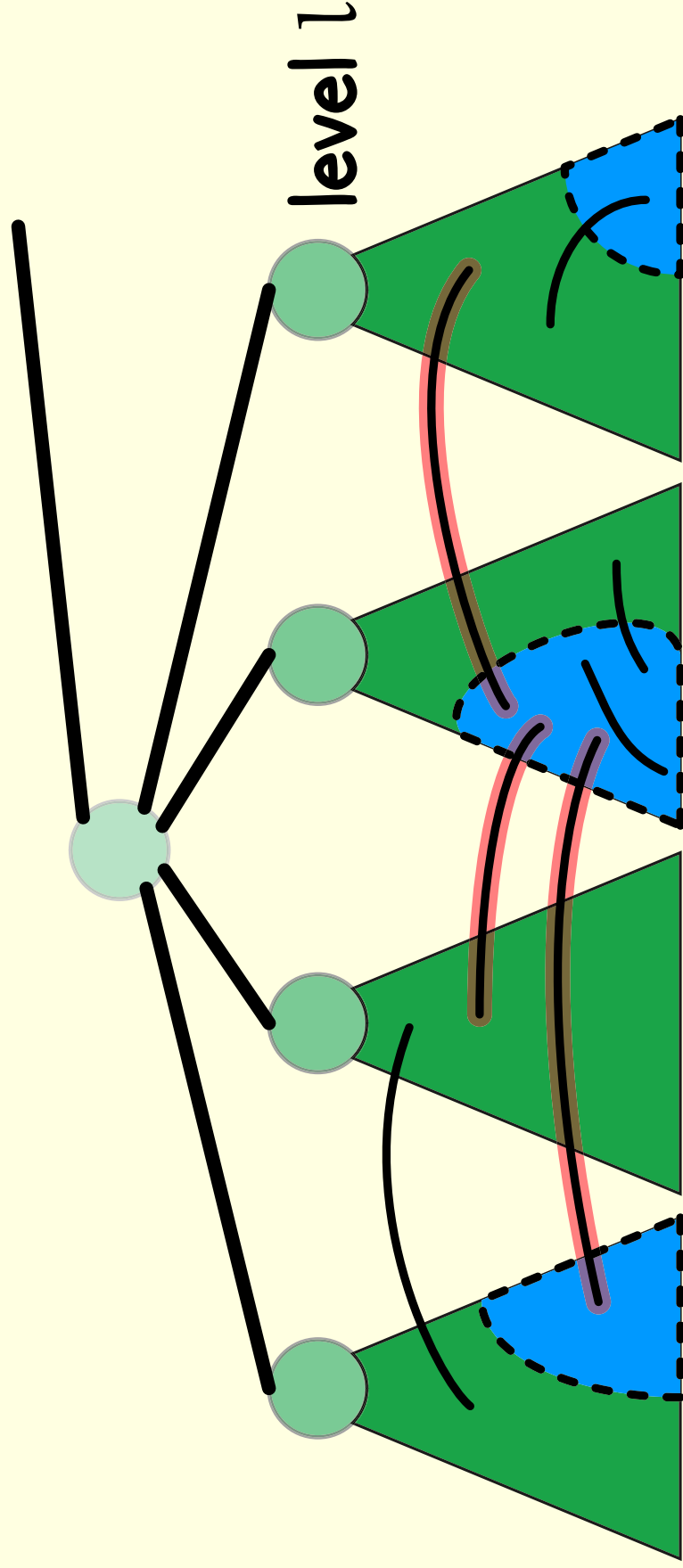
Who's in? Who's out?

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



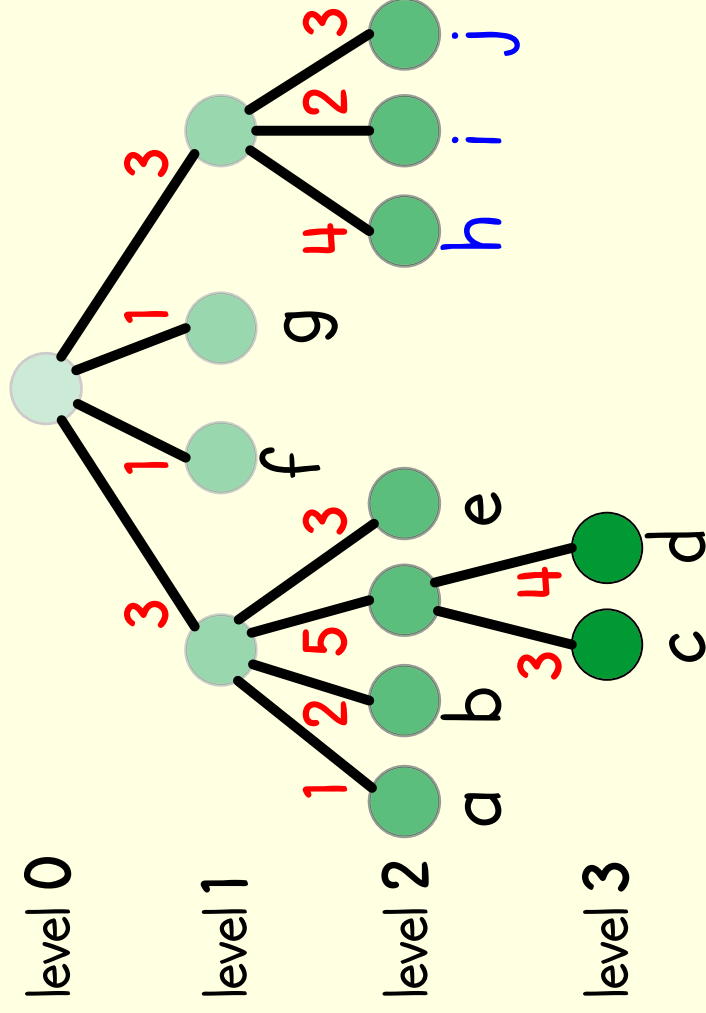
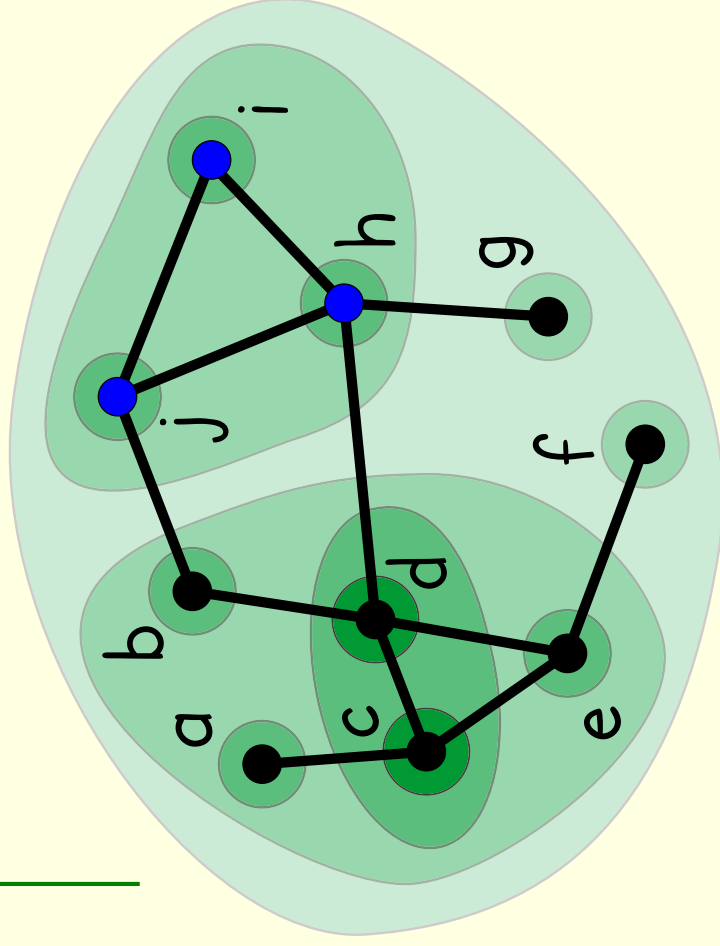
Who's in? Who's out?

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



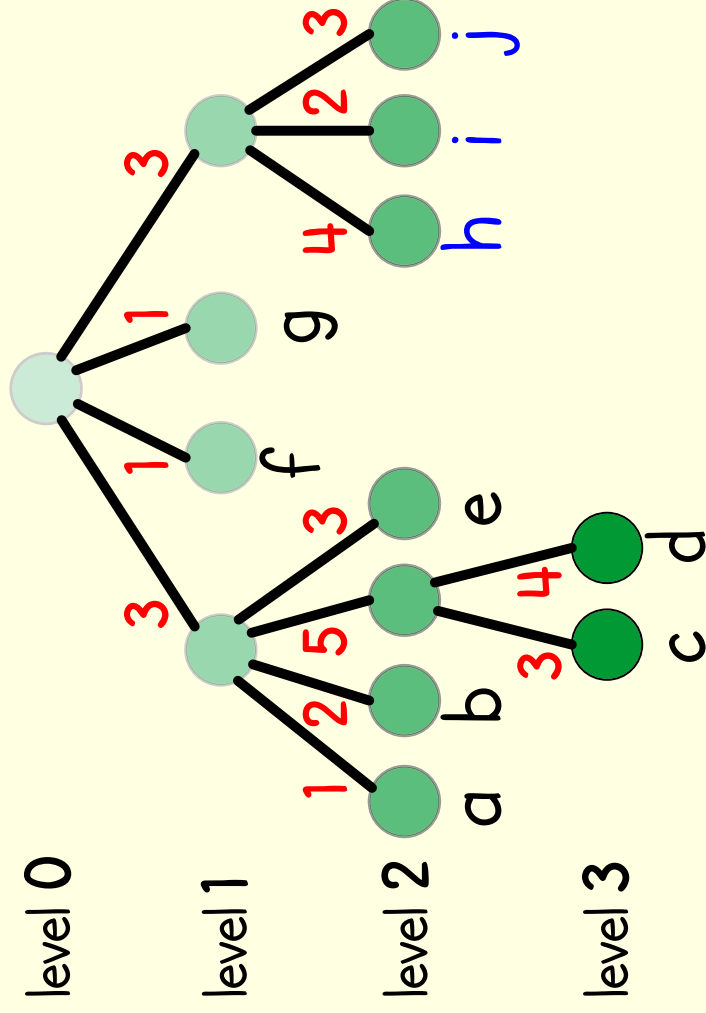
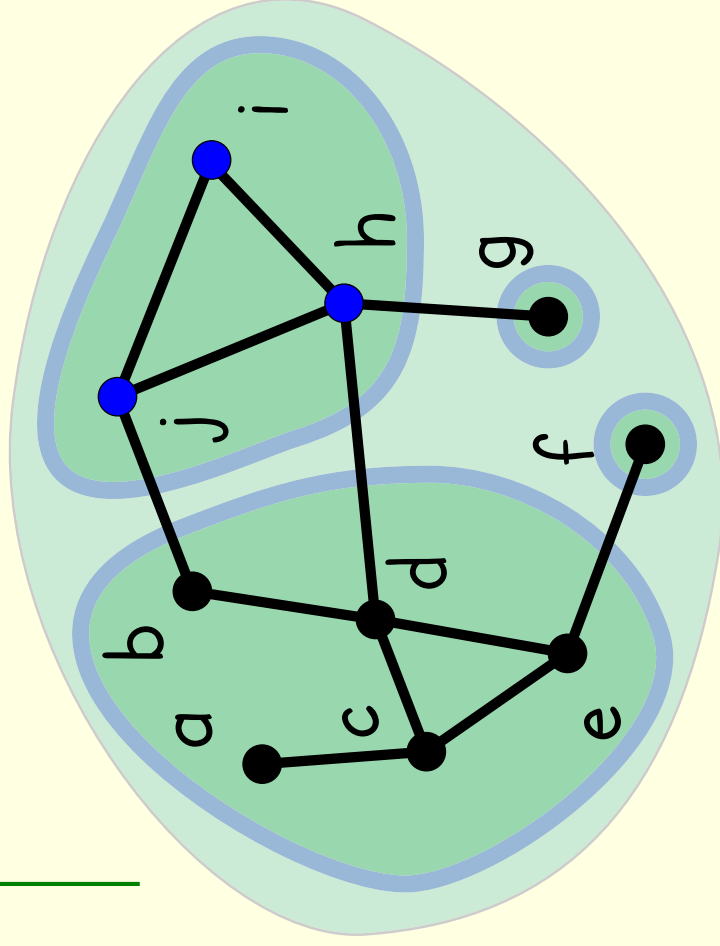
Pop Quiz 1: $w_1(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



Pop Quiz 1: $w_1(X)$

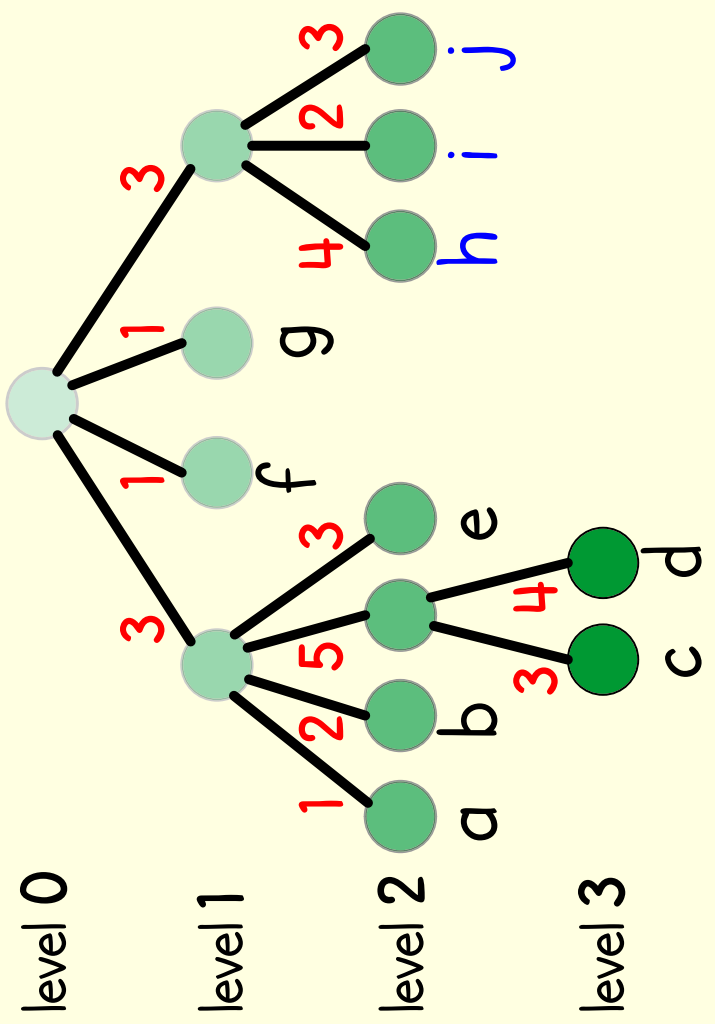
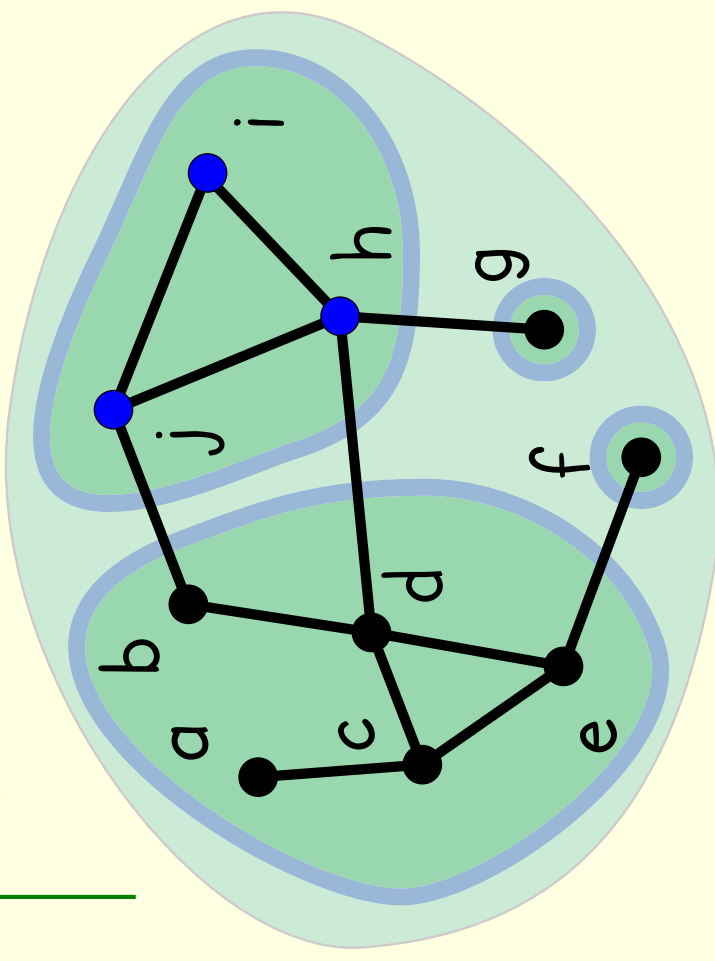
$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



Pop Quiz 1: $w_1(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

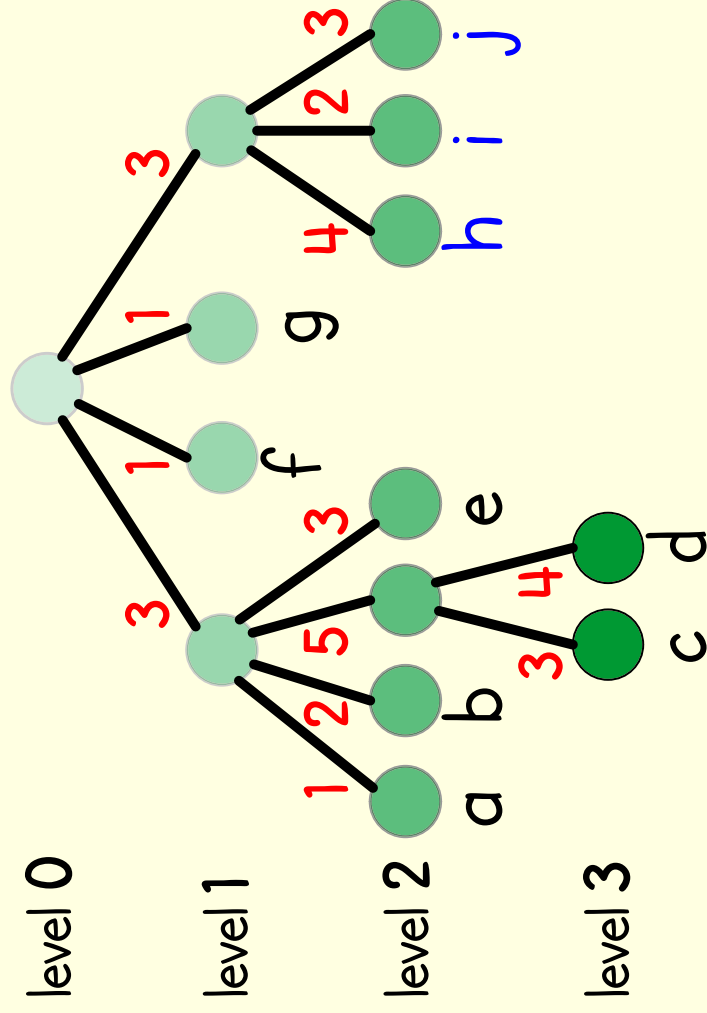
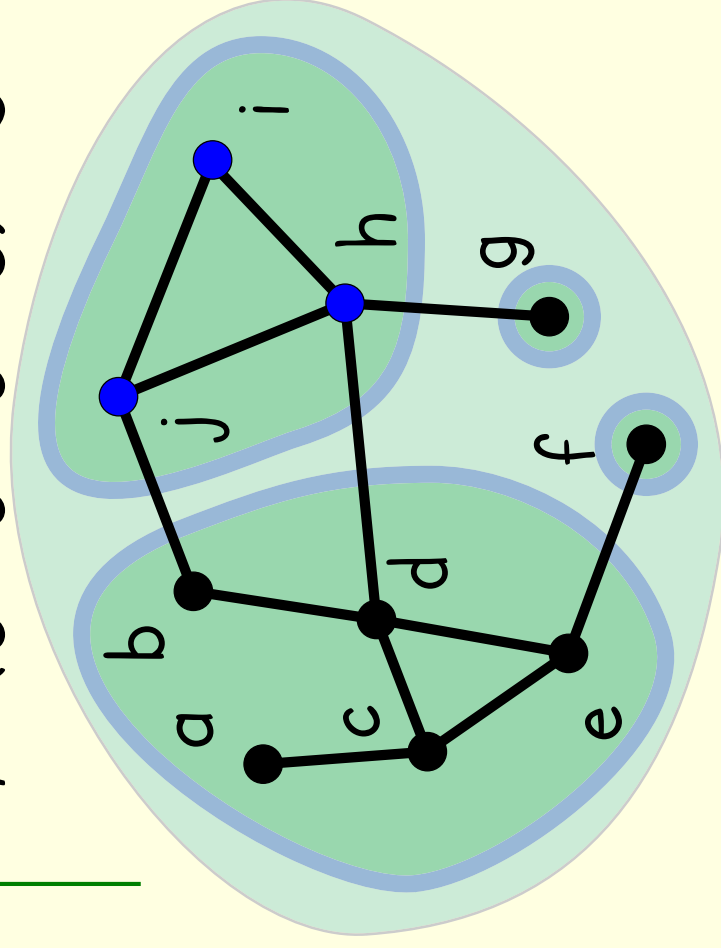
q



Pop Quiz 1: $w_1(X)$

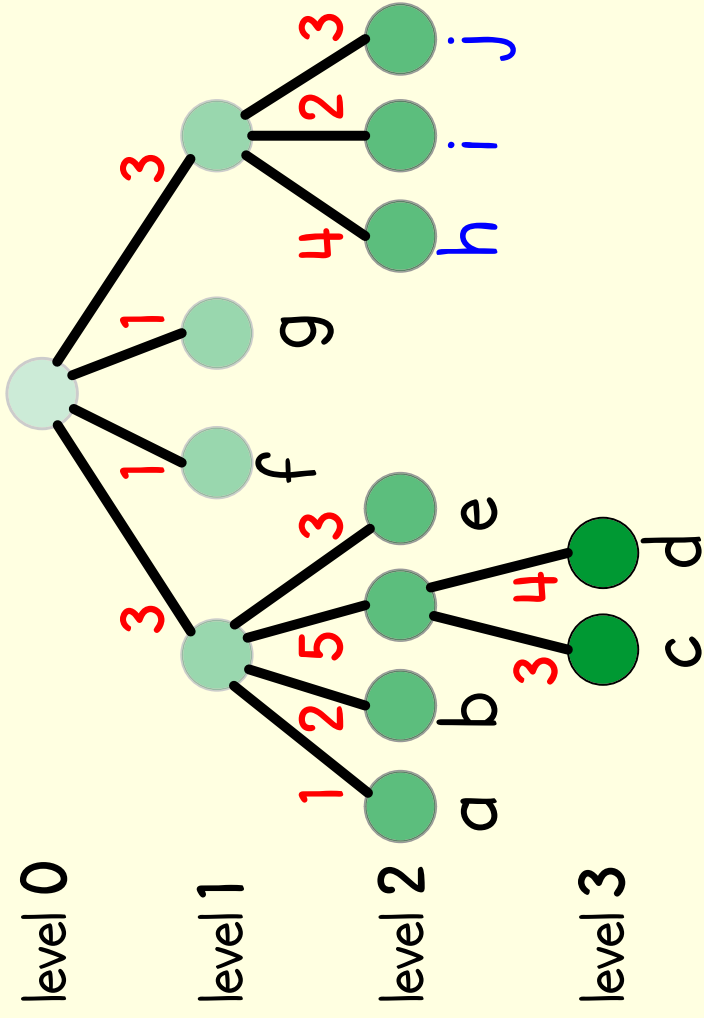
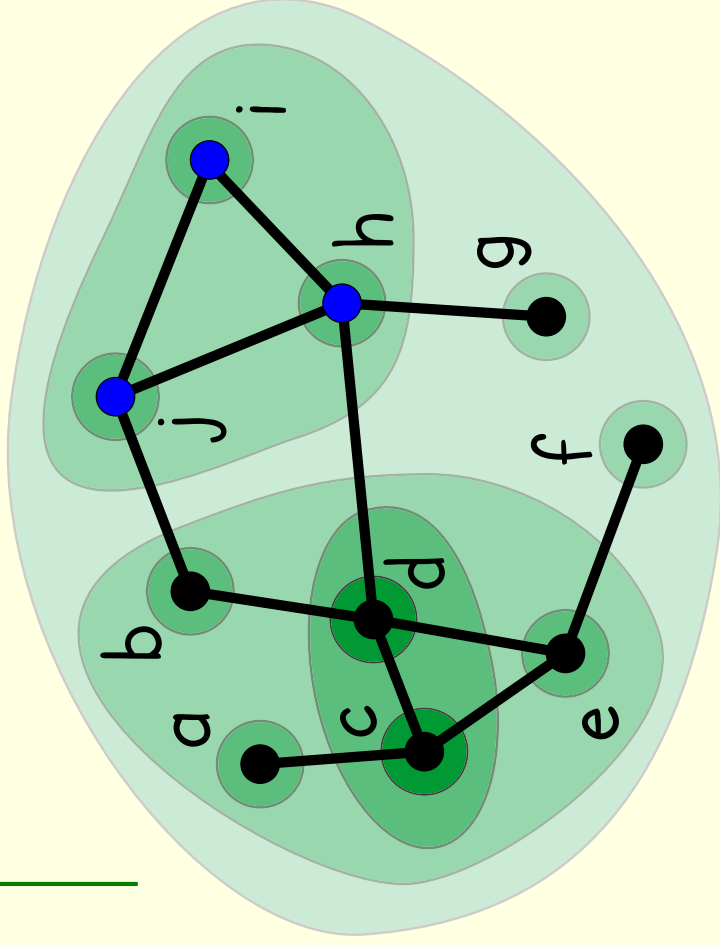
$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

$$9 - (0 + 0 + 0 + 6) = 3$$



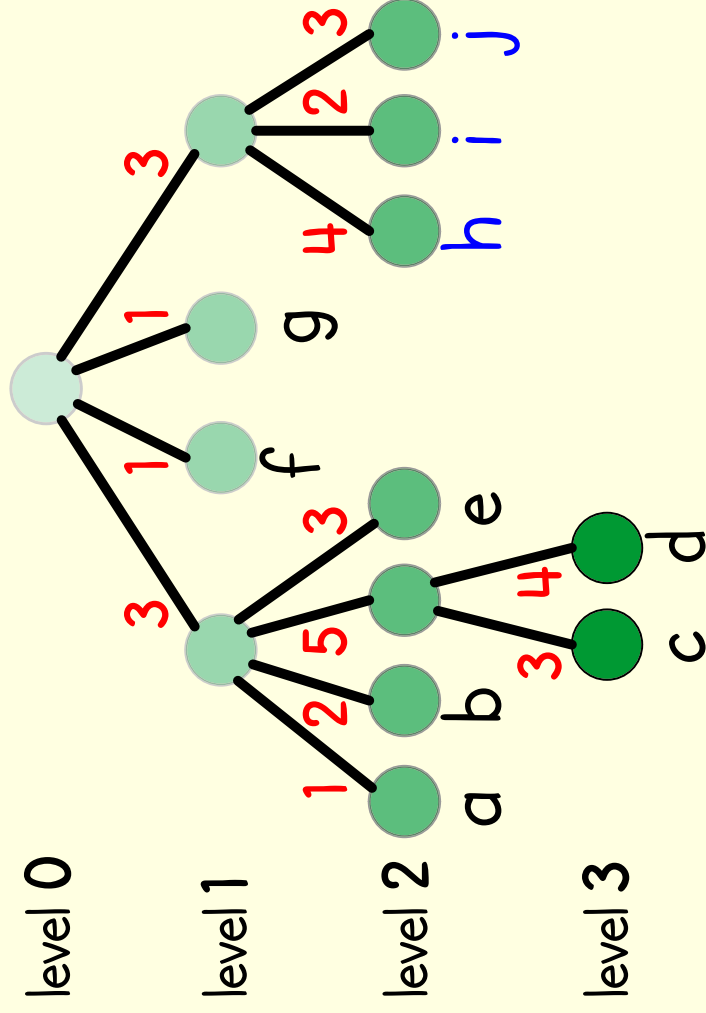
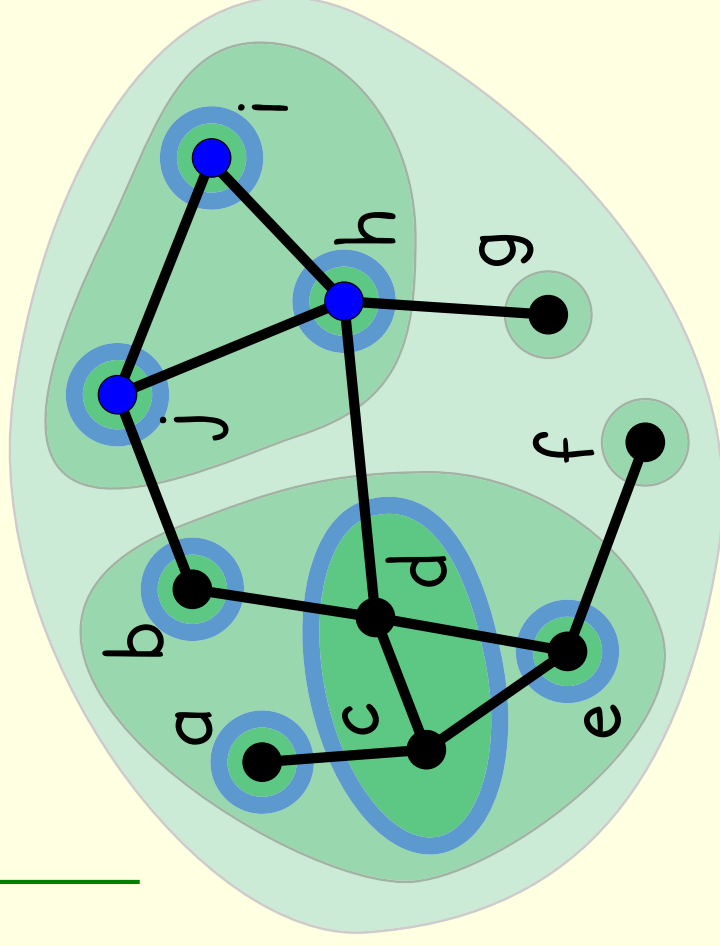
Pop Quiz 2: $w_2(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



Pop Quiz 2: $w_2(X)$

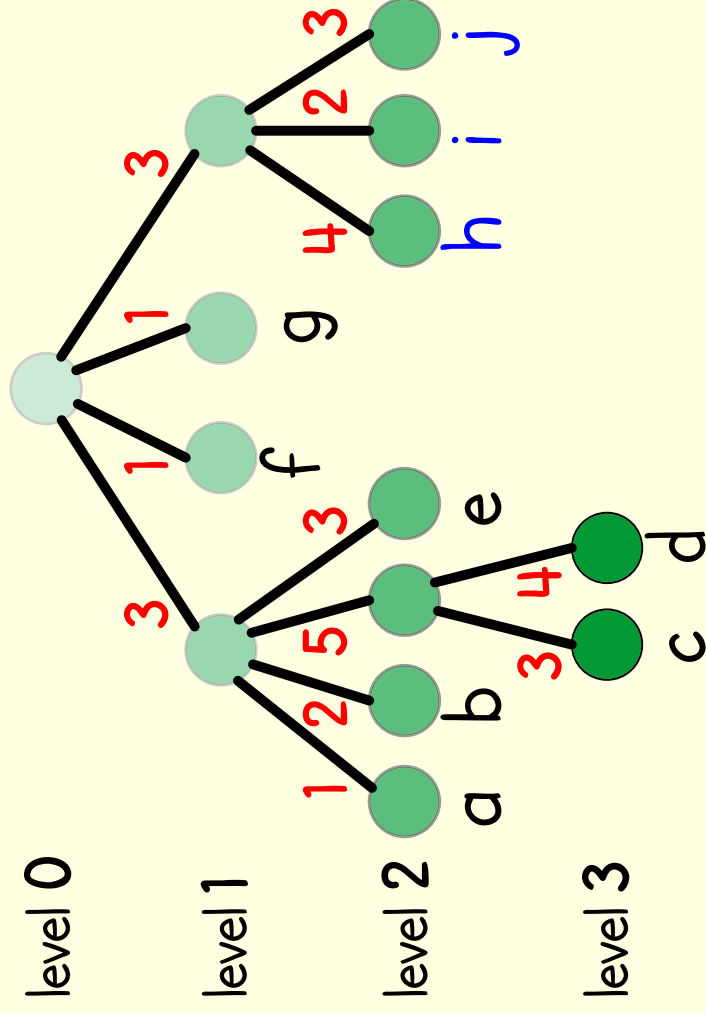
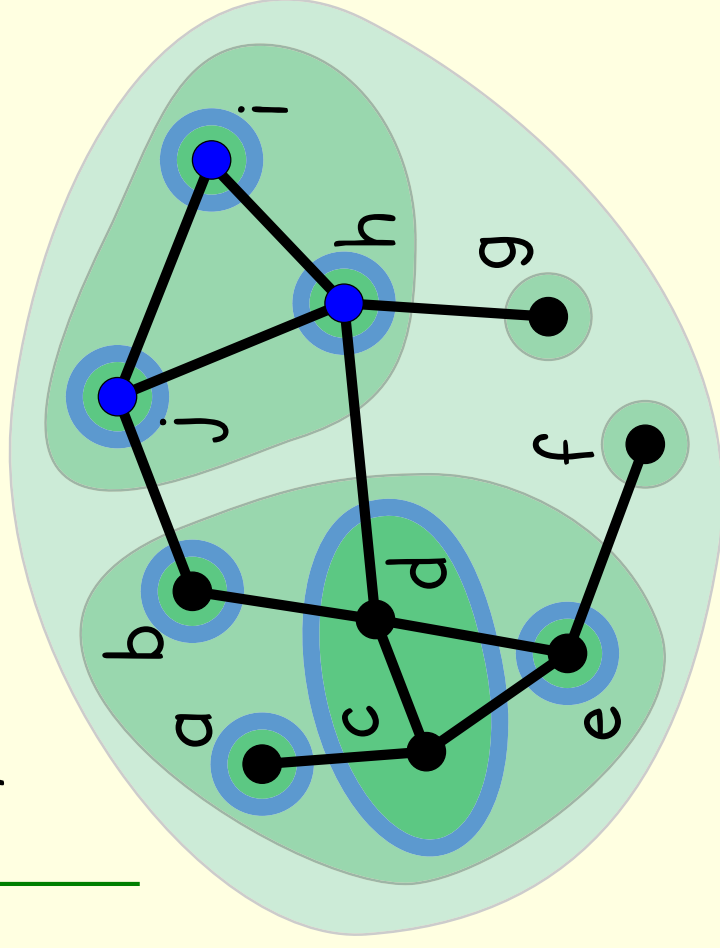
$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



Pop Quiz 2: $w_2(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

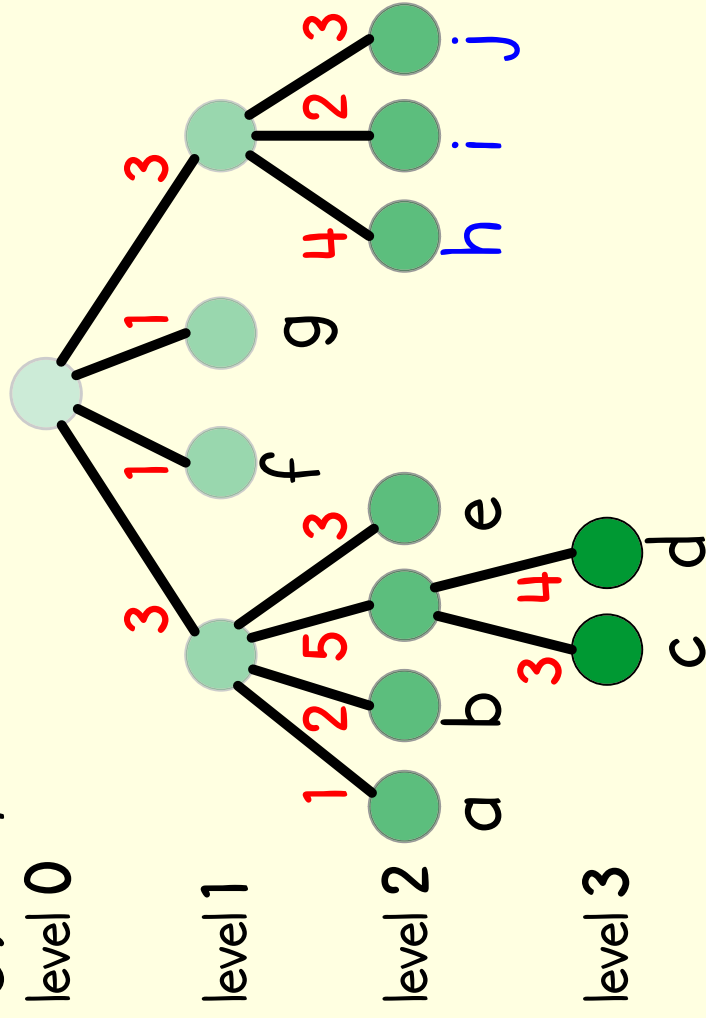
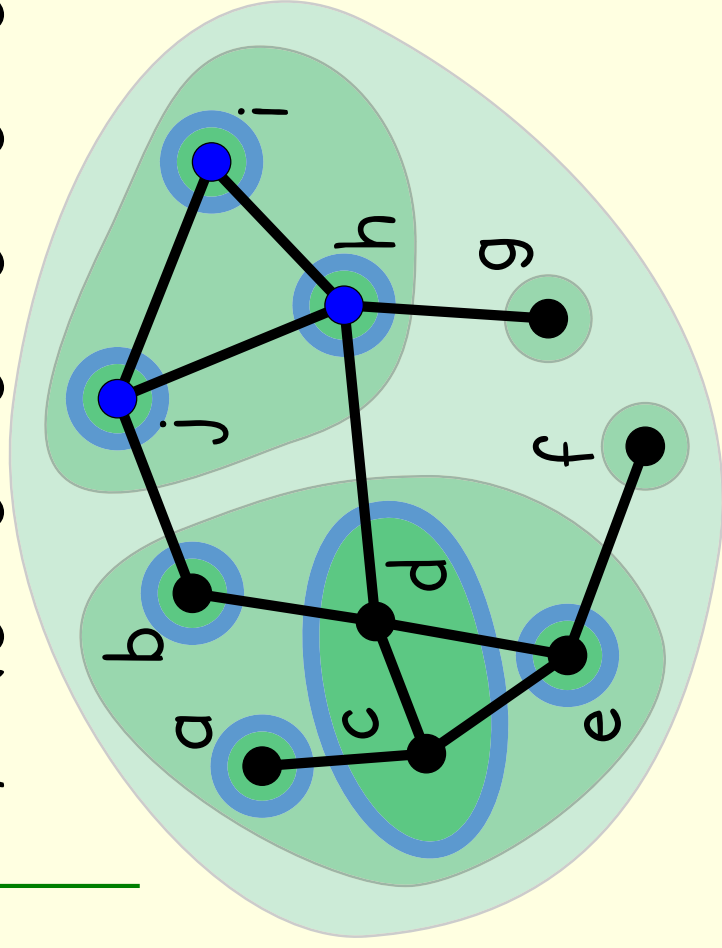
q



Pop Quiz 2: $w_2(X)$

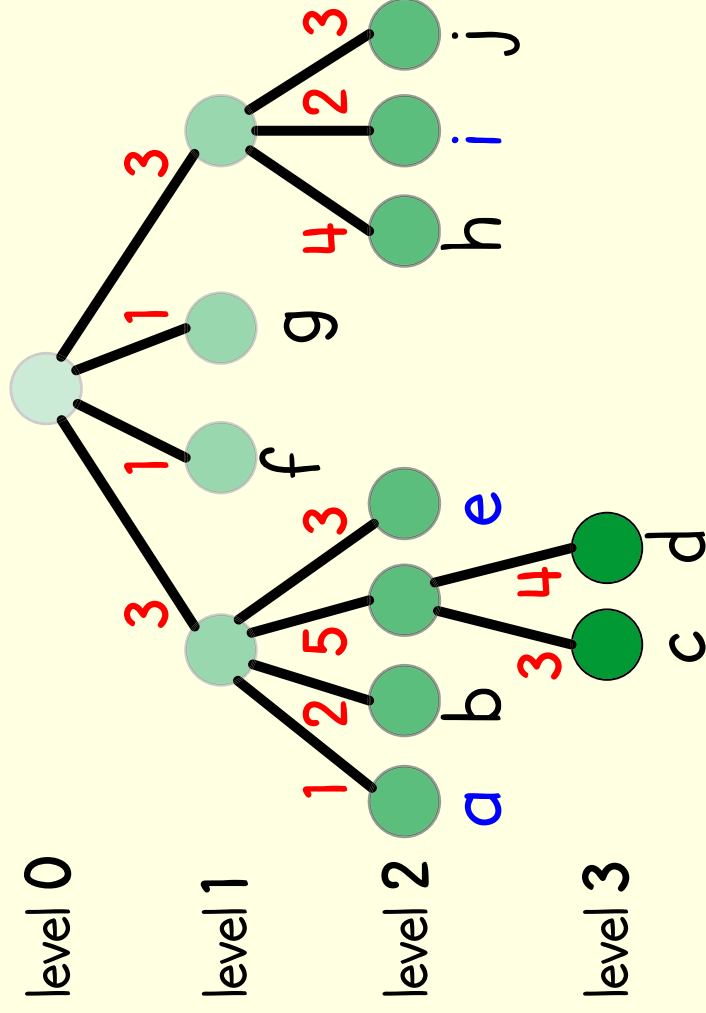
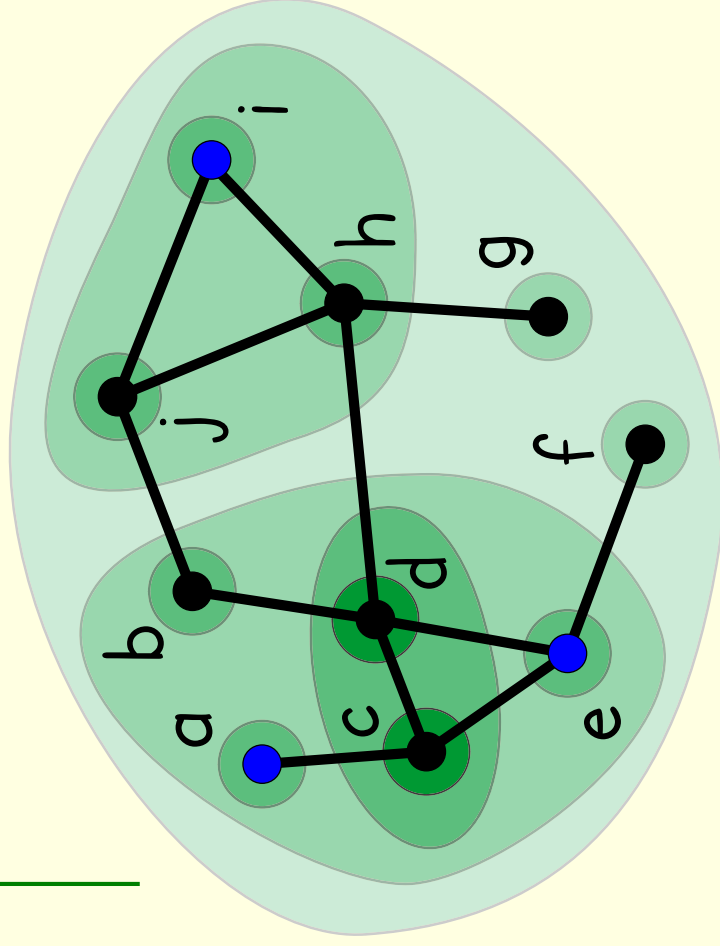
$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

$$9 - (0 + 0 + 0 + 0 + 0 + 0) = 9$$



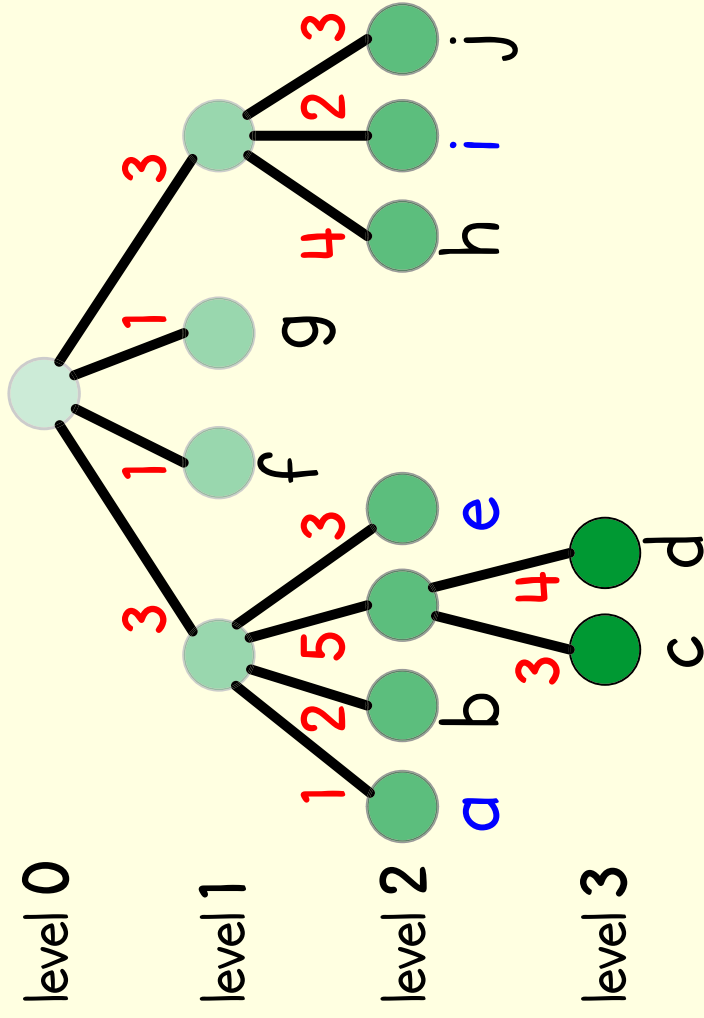
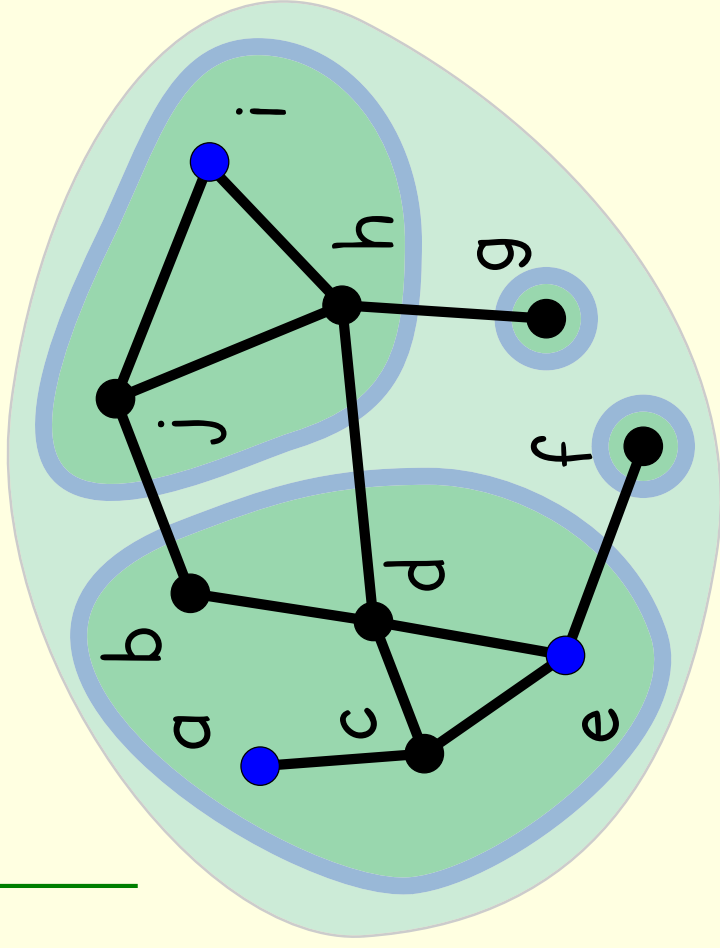
Pop Quiz 3: $w_1(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



Pop Quiz 3: $w_1(X)$

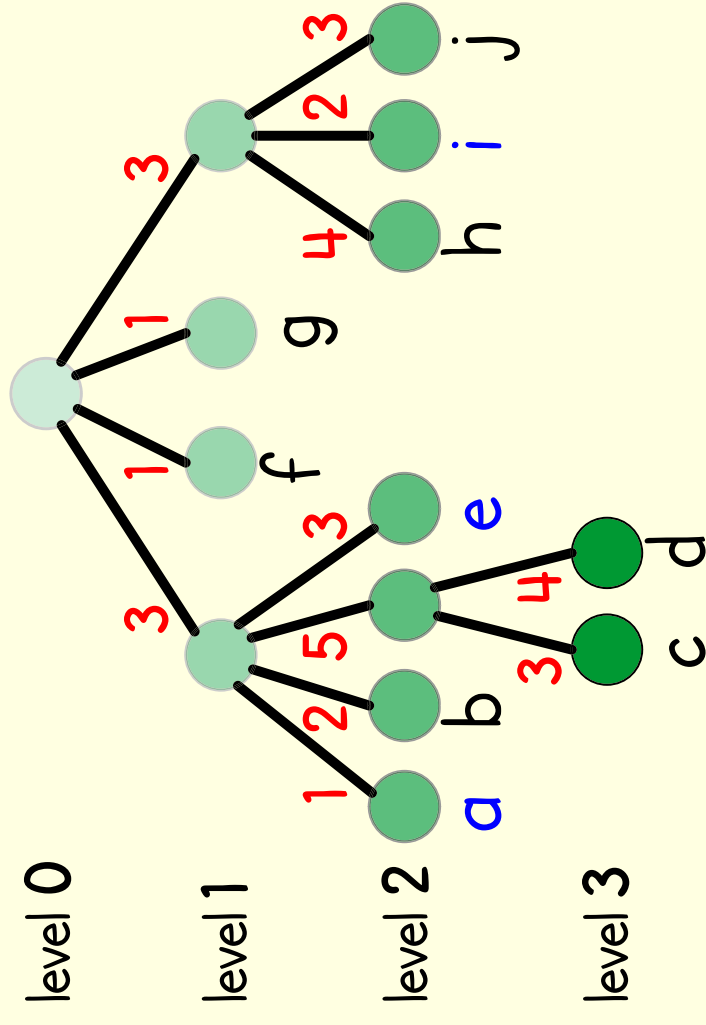
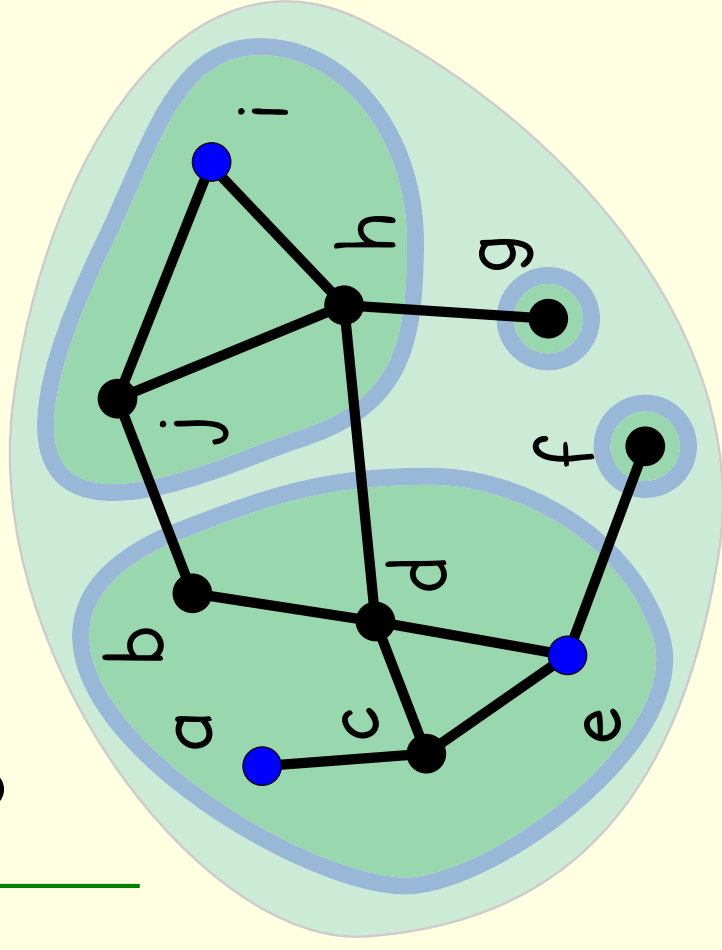
$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



Pop Quiz 3: $w_1(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

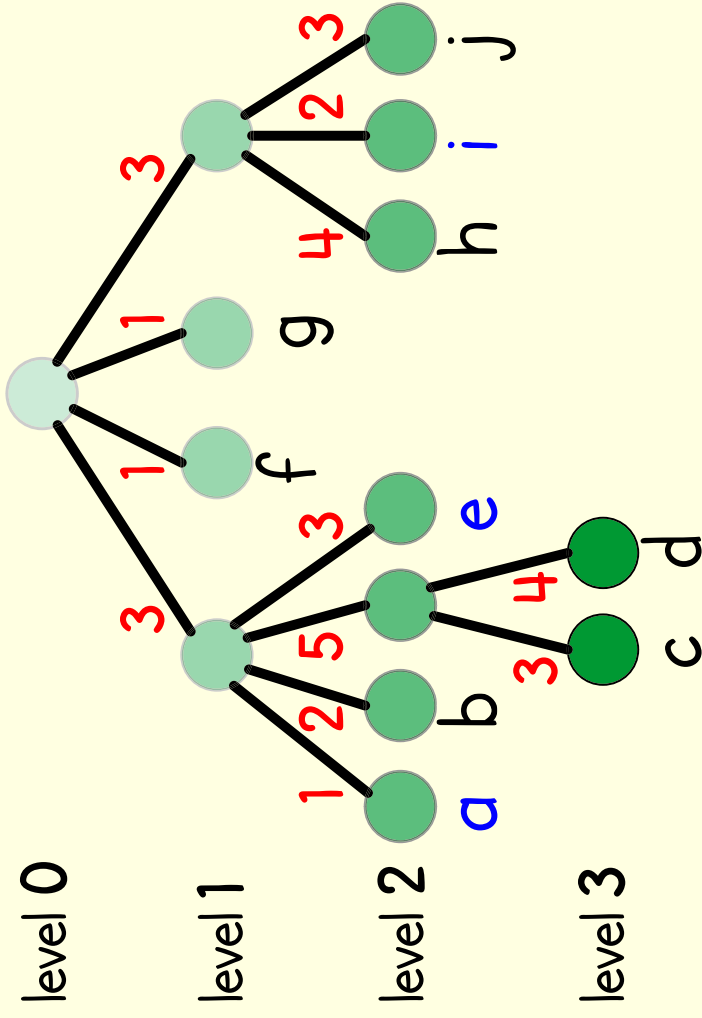
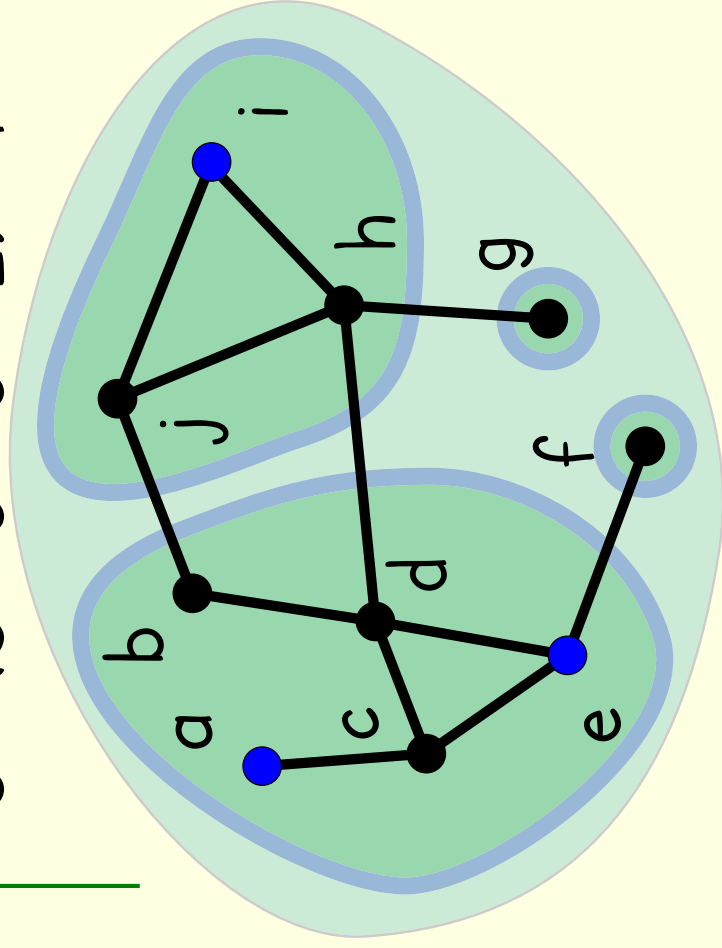
6



Pop Quiz 3: $w_1(X)$

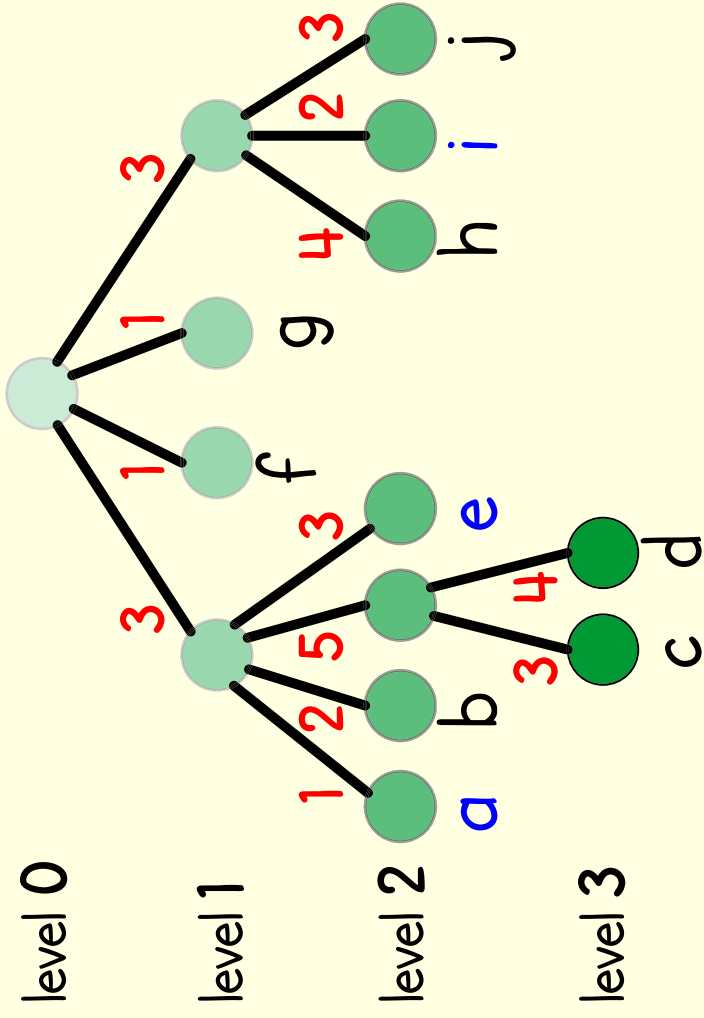
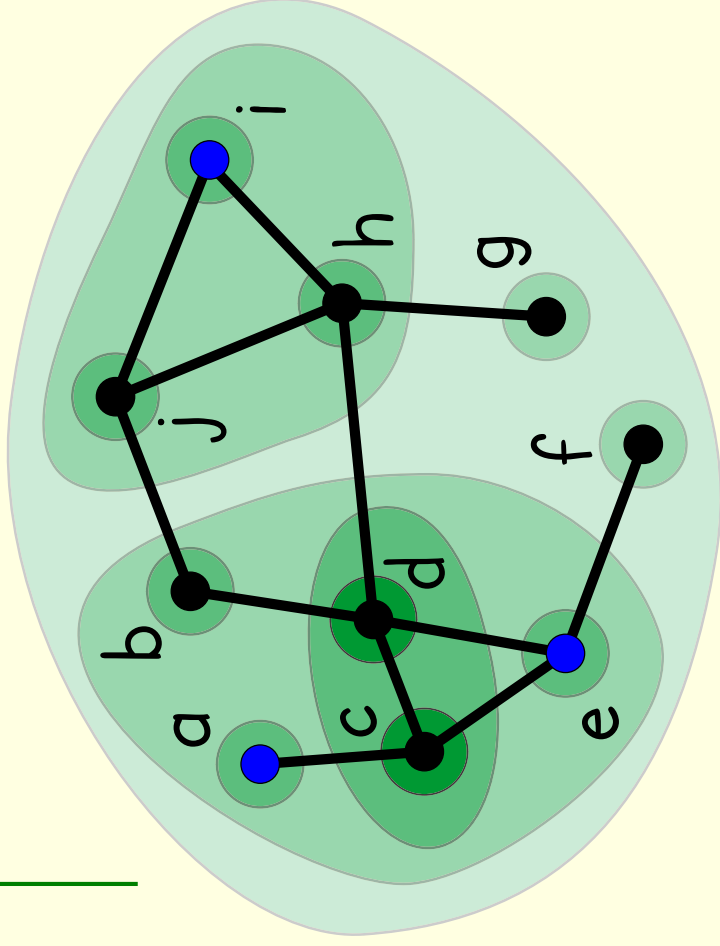
$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

$$6 - (3 + 0 + 0 + 2) = 1$$



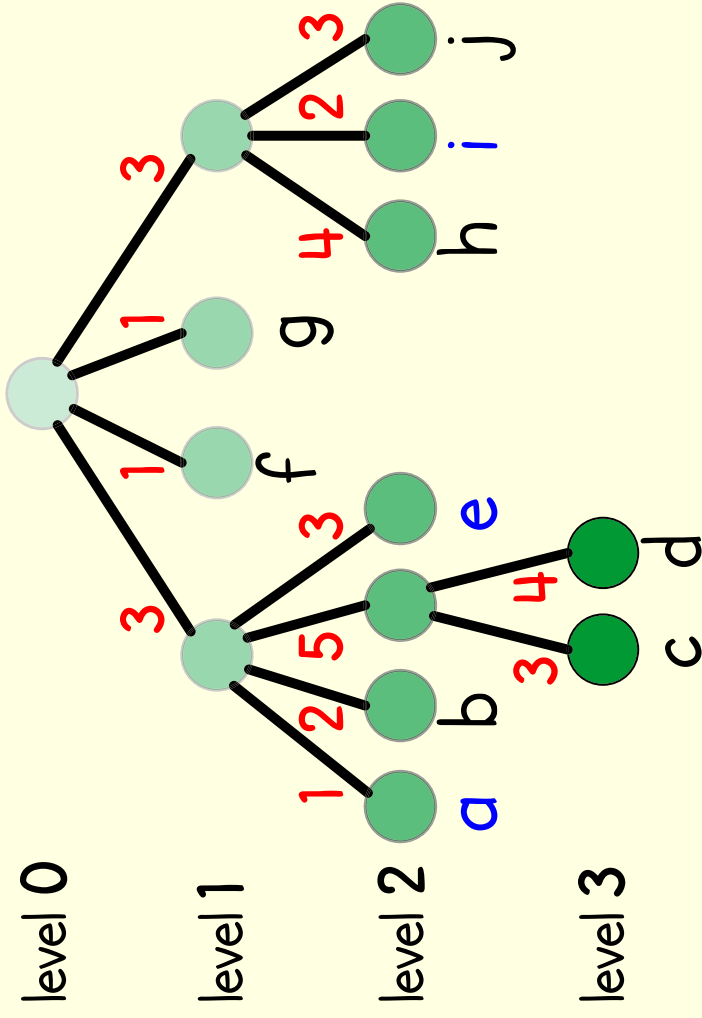
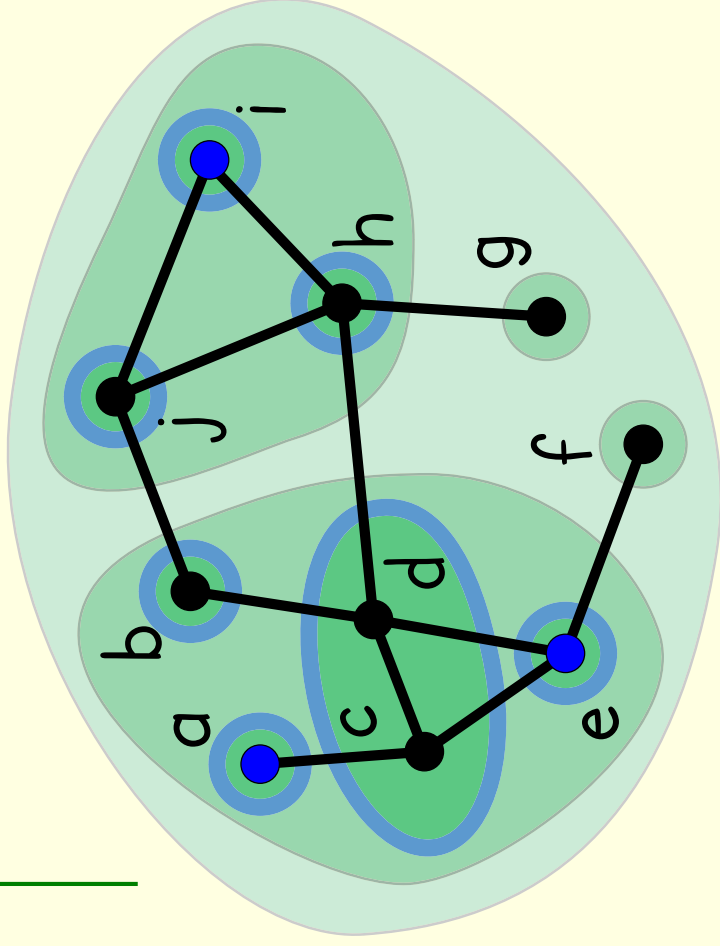
Pop Quiz 4: $w_2(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



Pop Quiz 4: $w_2(X)$

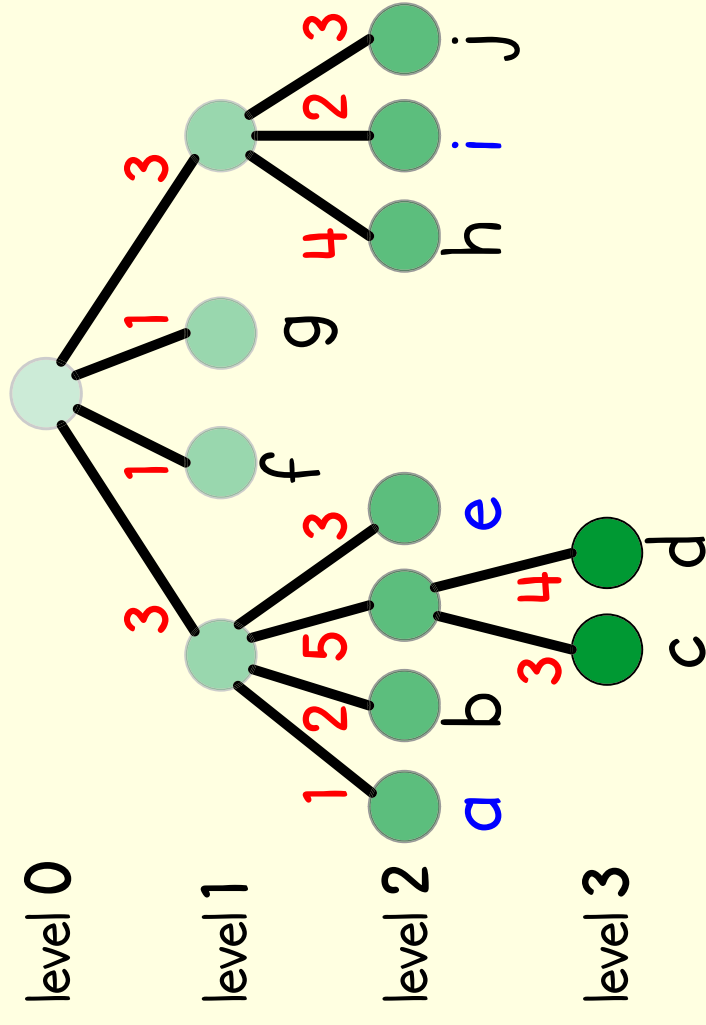
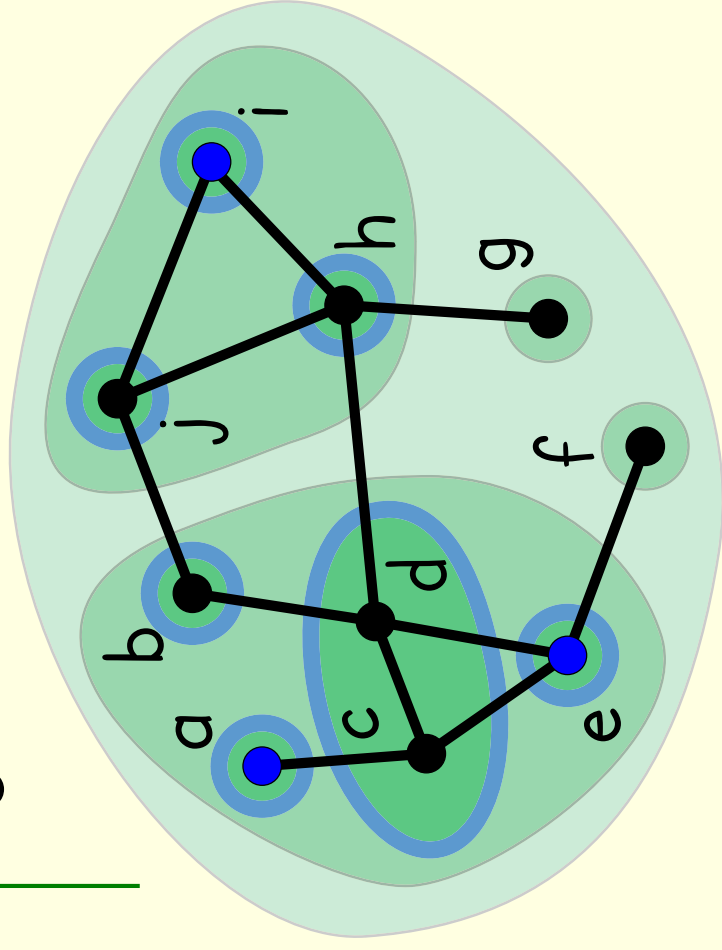
$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



Pop Quiz 4: $w_2(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

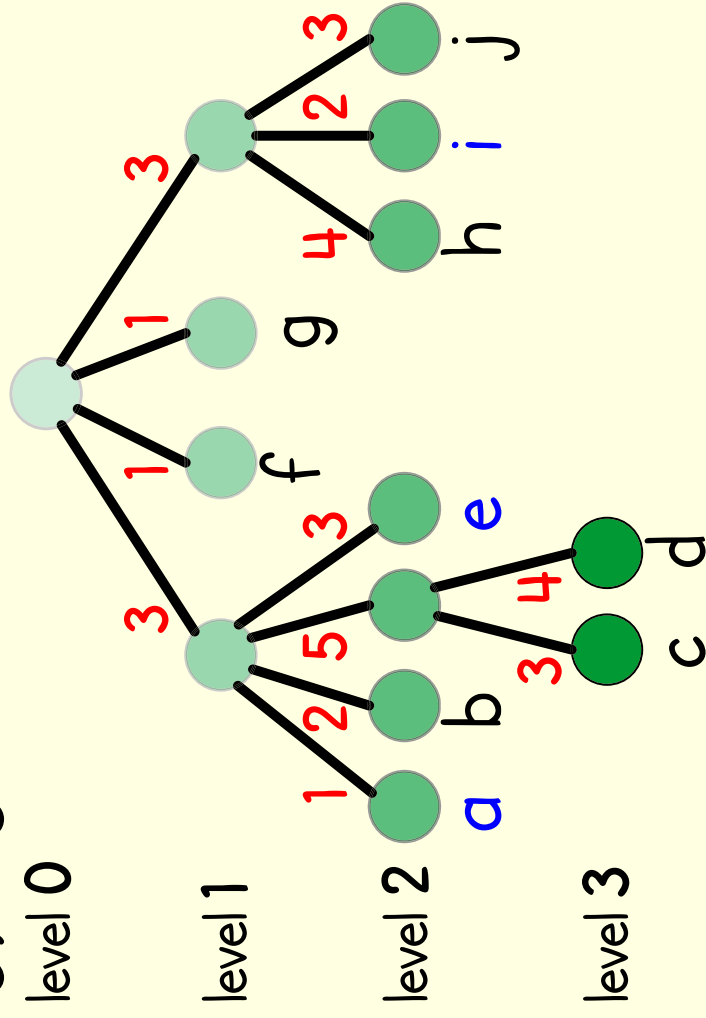
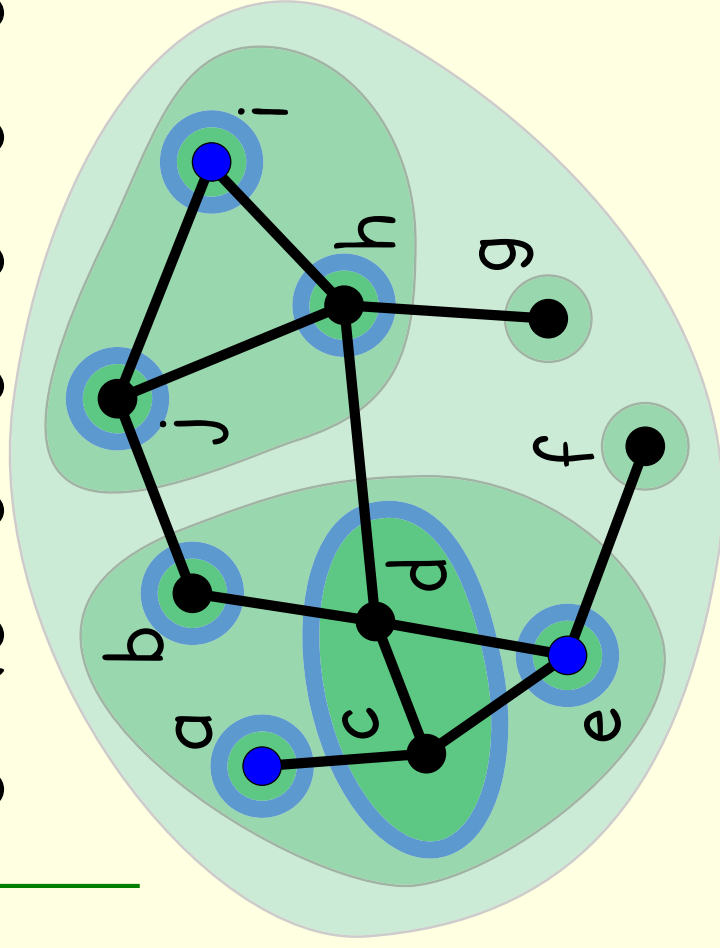
6



Pop Quiz 4: $w_2(X)$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

$$6 - (0 + 0 + 0 + 0 + 0 + 0) = 6$$



Properties Of $w_l(X)$

$$w_l(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^l} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

- Additive

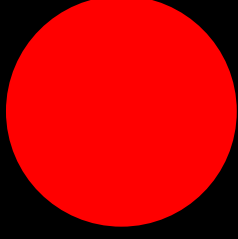
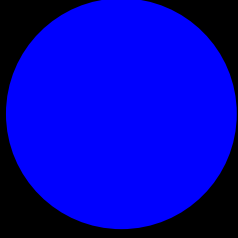
$$w_l(X_1 \uplus X_2) = w_l(X_1) + w_l(X_2)$$

- Non-decreasing (as l grows, i.e., deeper)

$$w_{l-1}(X) \leq w_l(X)$$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

$$w_1(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^1} \text{cap}(X \cap S_{v_t}, S_{v_t})$$



The Two Towers Ratios

Consider a level l node v_t .

$$\text{Bandwidth} \quad \lambda_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{\text{out}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

$$\text{Weight} \quad \delta_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{w_{l+1}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

There are special cases like $\frac{0}{0}$.

λ_{v_t} Textually

$$\lambda_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}}$$

$$\left\{ \frac{\text{out}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

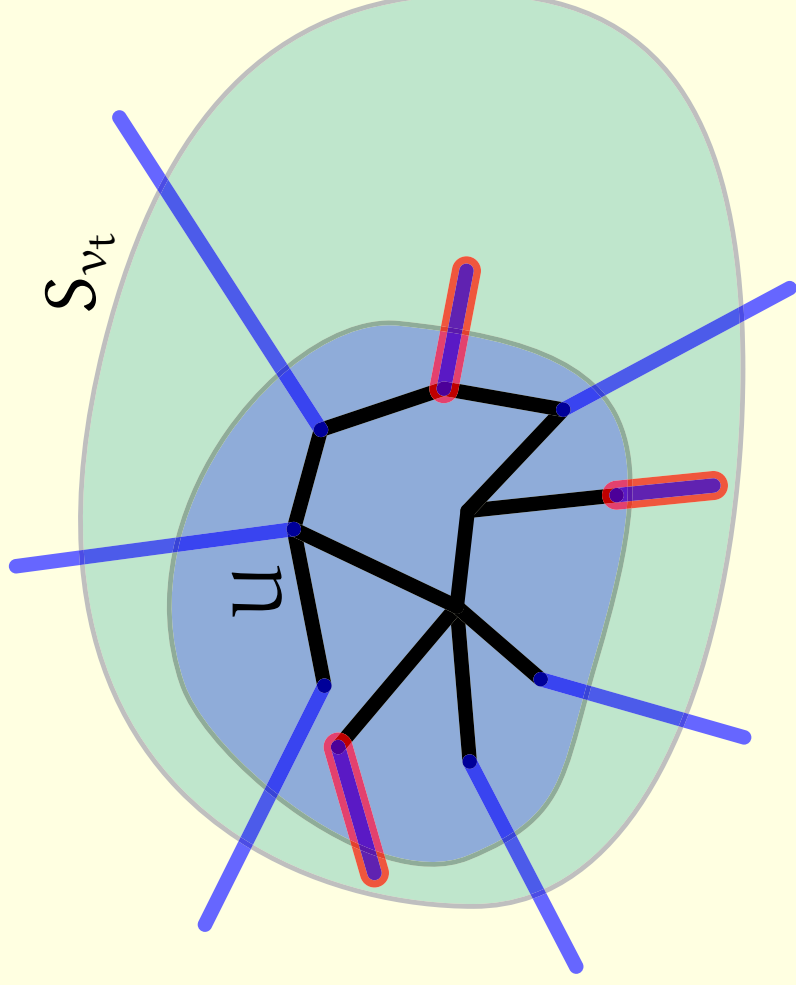
capacity of the edges
leaving a subset U of
the cluster S_{v_t}

vs. capacity of the edges
connecting U to the
rest of the cluster S_{v_t}

Claim: it is possible to have $\lambda_{v_t} = O(\log n)$.

λ_{v_t} Pictorially

$$\lambda_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{\text{out}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$



λ_{v_t} Intuitively

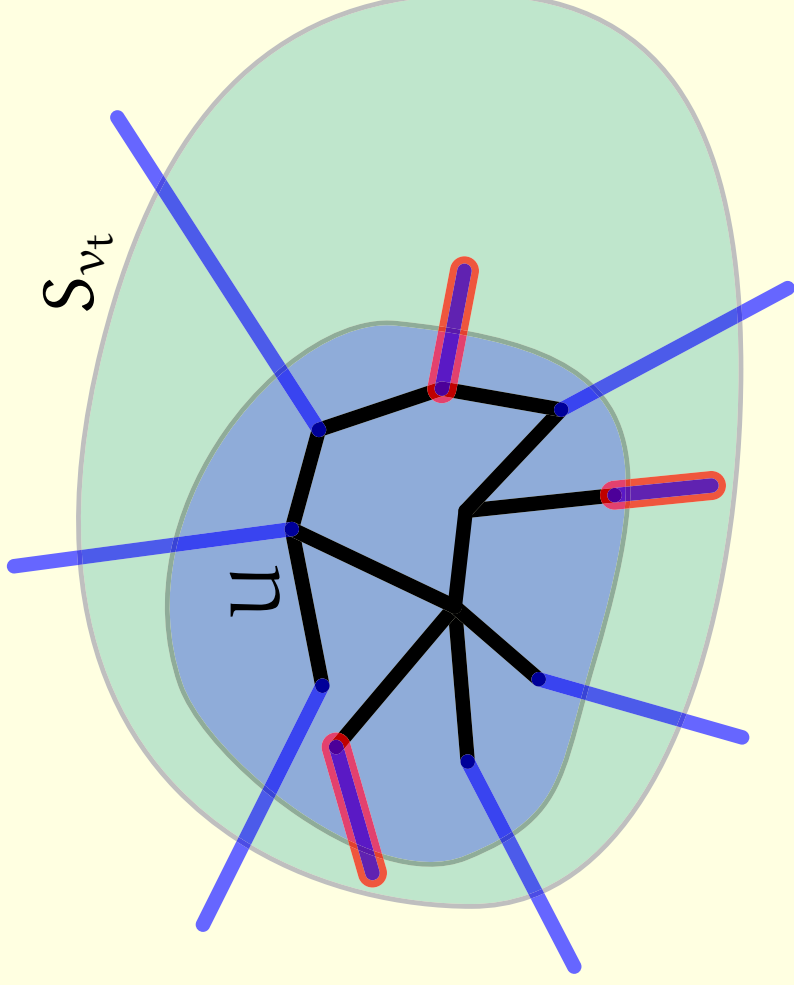
$$\lambda_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{\text{out}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

The routing algorithm will mainly use edges in the **denominator** to leave the set U .

Any subset of S_{v_t} tries to route all messages within S_{v_t} .

λ_{v_t} Pictorially Once More

$$\lambda_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{\text{out}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$



δ_{v_t} Textually

$$\delta_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}}$$

capacity of the edges
leaving the sub-clusters
of U in the
decomposition of S_{v_t}

$$\left\{ \frac{w_{l+1}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

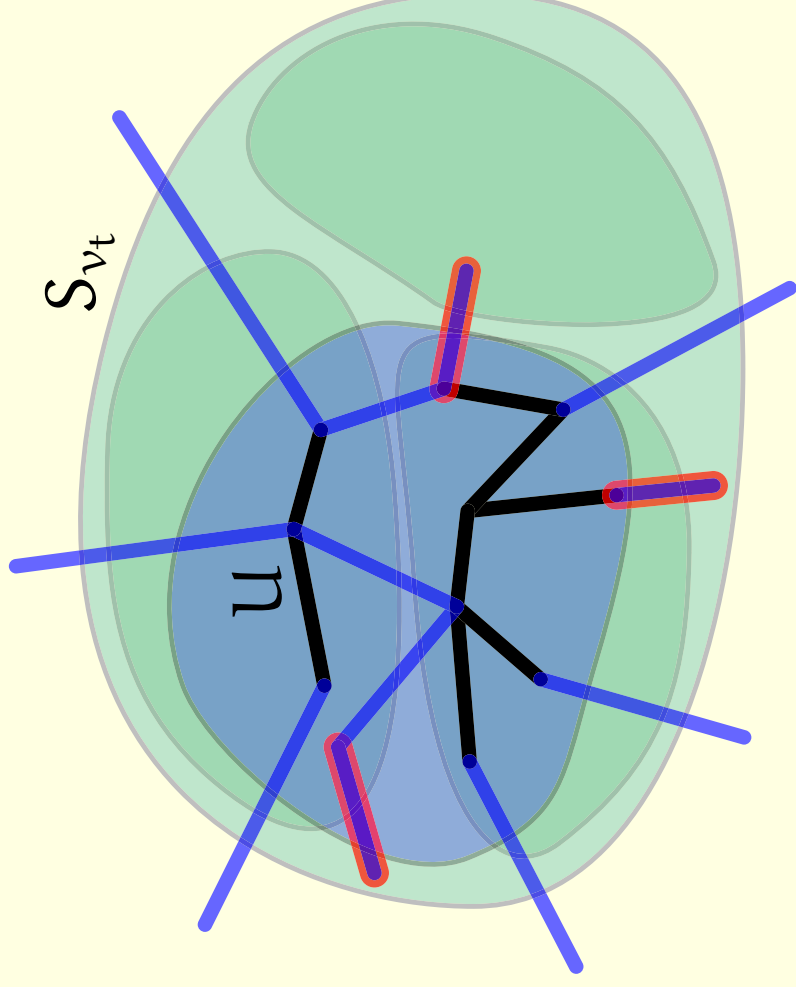
vs. capacity of the edges
connecting U to the
rest of the cluster S_{v_t}

Claim: it is also possible to have $\delta_{v_t} = O(\log n)$.

δ_{v_t} Pictorially



$$\delta_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{w_{l+1}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$



δ_{v_t} Intuitively

$$\delta_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{w_{l+1}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

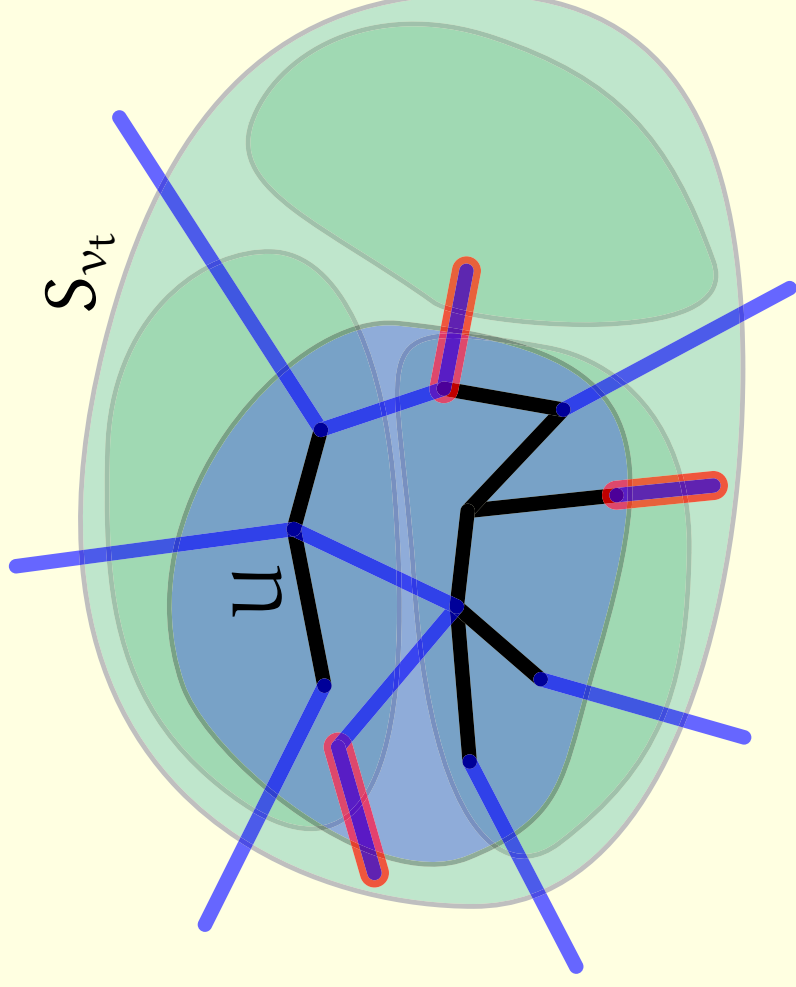
In the simulation $w_{l+1}(U)$ will be proportional to the probability that a message has to be routed to the set U .

Note that

$$w_{l+1}(U) = \sum_{v \in U} w_{l+1}(v).$$

δ_{v_t} Pictorially Once More

$$\delta_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{w_{l+1}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$





Simulating T_G on G

Tree Node Mapping

Suppose a message is to be sent from s to t . This corresponds to sending a message from s_t to t_t .

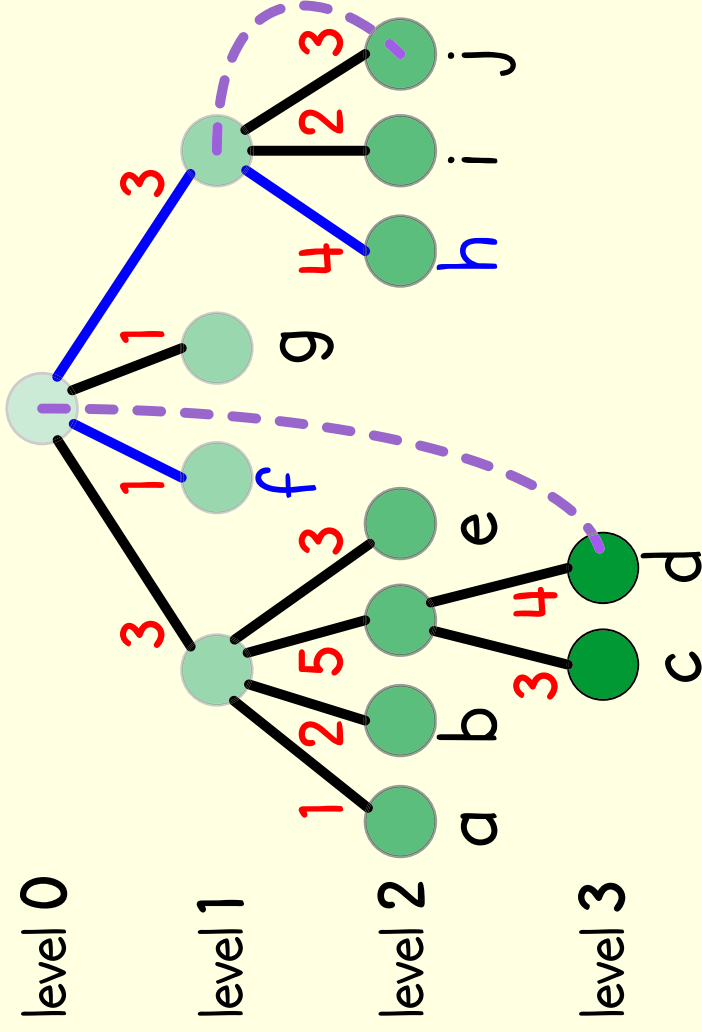
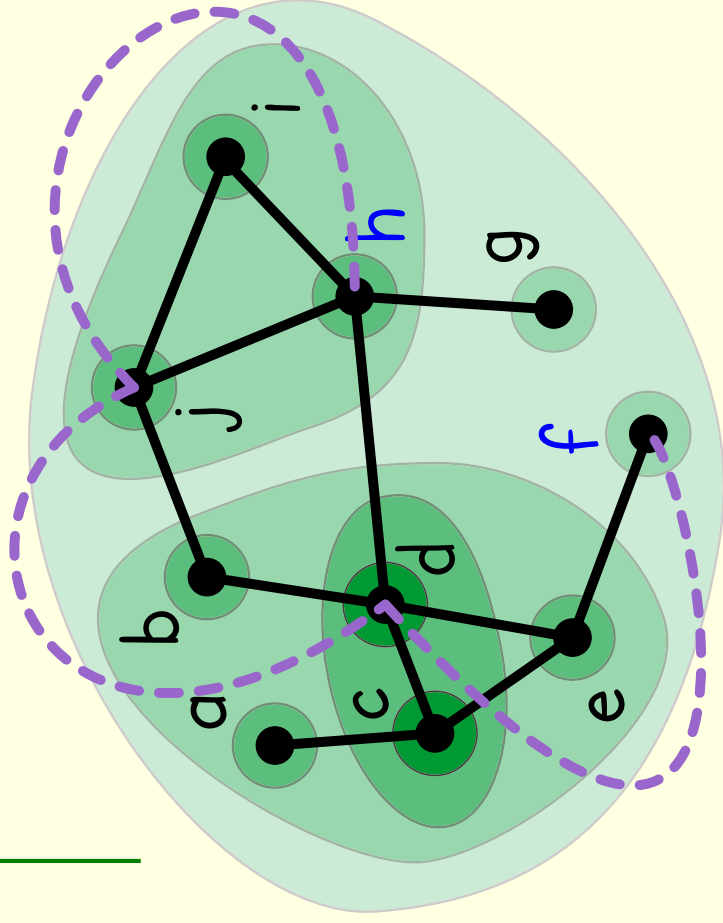
On T_G , the path from s_t to t_t is unique.

For a level l node v_t on this path, we map v_t to a node $v \in S_{v_t}$ with probability

$$\frac{w_{l+1}(v)}{w_{l+1}(S_{v_t})}.$$

Are we done yet?

No... We only know the intermediate destinations.



Tree Edge Mapping

Consider a level $l - 1$ node u_t with d children $\{v_1, v_2, \dots, v_d\}$.

Setup a concurrent multi-commodity flow within S_{u_t} :

- $|S_{u_t}|^2$ commodities $f_{u,v}$ for each $u, v \in S_{u_t}$
- source and sink of $f_{u,v}$ is u and v
- demand

$$d_{u,v} := \frac{w_l(u)}{w_l(S_{u_t})} \cdot \text{out}(S_{v_i}) \cdot \frac{w_{l+1}(v)}{w_{l+1}(S_{v_i})}$$

Demand

$$d_{u,v} := \underbrace{\frac{w_l(u)}{w_l(S_{u_t})}}_{\text{required bandwidth}} \cdot \underbrace{\text{out}(S_{v_i})}_{\text{Pr}(v_i \mapsto v)} \cdot \underbrace{\frac{w_{l+1}(v)}{w_{l+1}(S_{v_i})}}_{\text{Pr}(u_t \mapsto u)}$$

- $\text{Pr}(u_t \mapsto u)$
- required bandwidth
- $\text{Pr}(v_i \mapsto v)$

Understanding The CMCF

Let q be the throughput fraction

$$\frac{\text{demand satisfied}}{\text{original demand}}$$

of the commodity whose demand is least met.

For each $f_{u,v}$, we know we can send $q \cdot d_{u,v}$ from u to v with congestion at most **1** on any link.

Understanding The CMCF

Let q be the throughput fraction

$$\frac{\text{demand satisfied}}{\text{original demand}}$$

of the commodity whose demand is least met.

For each $f_{u,v}$, we know we can send $q \cdot d_{u,v}$ from u to v with congestion at most **1** on any link.

Question: What if we send $c \cdot d_{u,v}$ instead?

Understanding The CMCF

Let q be the throughput fraction

$$\frac{\text{demand satisfied}}{\text{original demand}}$$

of the commodity whose demand is least met.

For each $f_{u,v}$, we know we can send $q \cdot d_{u,v}$ from u to v with congestion at most **1** on any link.

Question: What if we send $c \cdot d_{u,v}$ instead? $\frac{c}{q}$

Theorem

The expectation of the relative load $L(e)$ of an edge $e \in E$ due to the simulation of a tree strategy on G is bounded by

$$E(L(e)) \leq O(\underbrace{h}_{\text{height}} \cdot \underbrace{\max\{\delta, \lambda\}}_{\text{characteristics of decomposition}} \cdot \underbrace{\log n \cdot C_t}_{\text{inefficiency of max multi-flow/min cut}}).$$

- height
- characteristics of decomposition
- inefficiency of max multi-flow/min cut

Road Map

Lemma For any edge e , $E(L(e)) = O(h \cdot C_t/q)$.

Lemma The value of the sparsest cut ϕ of the CMCF is at least $\frac{1}{5\delta + 2\lambda}$.

Theorem Any k -commodity CMCF can be satisfied up to $q = \Omega(\phi / \log k)$.

(Aumann and Rabani, SIAM JOC 1998)

$$\therefore q = \Omega\left(\frac{1}{(5\delta + 2\lambda) \cdot \log(n^2)}\right)$$

$$\Rightarrow E(L(e)) = O(h \cdot \max\{\delta, \lambda\} \cdot \log n \cdot C_t)$$

Back to Online Routing

Online routing on the tree is way too easy!

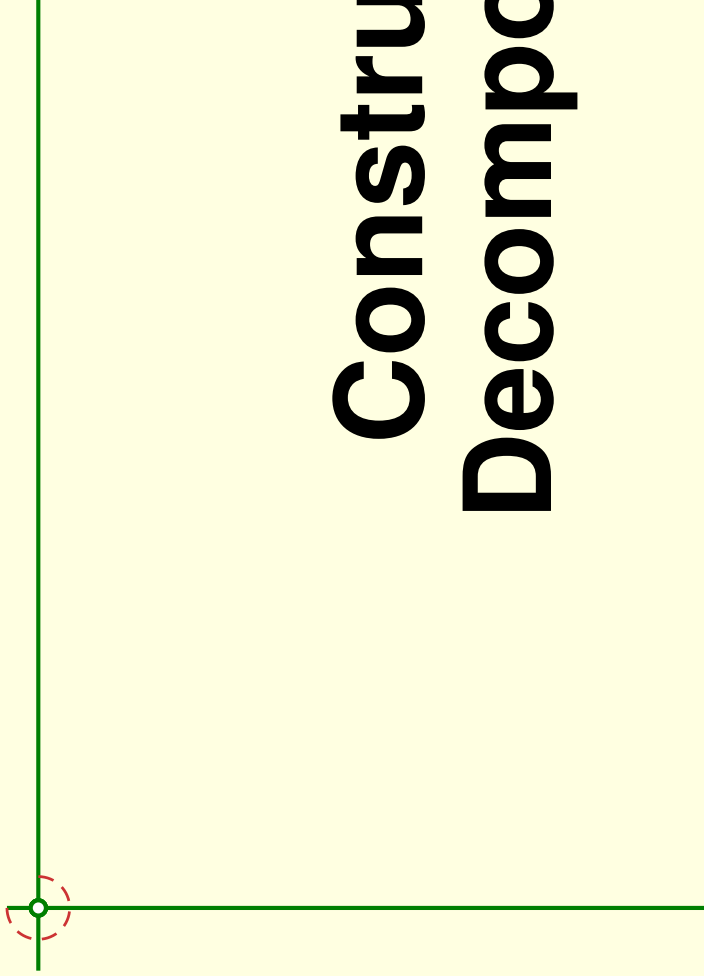
If it is possible to construct the decomposition tree s.t.

$$h = O(\log n) \quad \lambda = O(\log n) \quad \delta = O(\log n),$$

then

$$\begin{aligned} E(L(e)) &\leq O(h \cdot \max\{\delta, \lambda\}) \cdot \log n \cdot C_t \\ &= O(\log^3 n). \end{aligned}$$

The high probability bound follows from Chernoff.



Constructing The Decomposition Tree

The Two Ratios

Consider a level l node v_t .

Bandwidth

$$\lambda_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}}$$

$$\left\{ \frac{\text{out}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

Weight

$$\delta_{v_t} := \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}}$$

$$\left\{ \frac{w_{l+1}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

Fulfillment

W.r.t. a particular choice of λ^* and δ^* , we say that a subset $S \subseteq V$ **fulfills** the bandwidth property iff

$$\lambda^* \geq \max_{\substack{U \subset S_{v_t} \\ |U| \leq 3|S_{v_t}|/4}} \left\{ \frac{\text{out}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}$$

and the weight property iff

$$\delta^* \geq \max_{\substack{U \subset S_{v_t} \\ |U| \leq |S_{v_t}|/2}} \left\{ \frac{w_{l+1}(U)}{\text{cap}(U, S_{v_t} \setminus U)} \right\}.$$

Weight Function $w_l(X)$

$$w_l(X) := \text{cap}(X, V) - \sum_{v_t \in V_t^l} \text{cap}(X \cap S_{v_t}, S_{v_t})$$

If a level l node v_t has children $\{v_i \mid i \in I\}$, then for a subset $U \subseteq S_{v_t}$

$$w_{l+1}(U) = \sum_{i \in I} \text{cap}(U \cap S_{v_i}, \overline{S_{v_i}}).$$

Theorem

For any graph $G = (V, E)$ with $|V| = n$, there exists a decomposition tree T_G with $h(T_G) = O(\log n)$, $\lambda(T_G) = O(\log n)$ and $\delta(T_G) = O(\log n)$.

Theorem

For any graph $G = (V, E)$ with $|V| = n$, there exists a decomposition tree T_G with $h(T_G) = O(\log n)$, $\lambda(T_G) = O(\log n)$ and $\delta(T_G) = O(\log n)$.

Proof

Let $\lambda^* = 20 \cdot \log n + 1$

Let $\delta^* = 24 \cdot \lambda^*$. (Cheers!)

We show an algorithm that produce a decomposition tree whose λ and δ are upper-bounded by these parameters.

Key Lemma

Lemma Given a level l cluster $S \subseteq V$ that fulfills the bandwidth property, it is possible to partition S into sub-clusters S_i 's s.t.

1. $S = \uplus_i S_i$
2. Each S_i fulfills the bandwidth property
3. $\forall i : |S_i| \leq \frac{5}{6} \cdot |S|$
4. S fulfills the weight property

Key Lemma

Lemma Given a level l cluster $S \subseteq V$ that fulfills the bandwidth property, it is possible to partition S into sub-clusters S_i 's s.t.

1. $S = \cup_i S_i$
2. Each S_i fulfills the bandwidth property
3. $\forall i : |S_i| \leq \frac{5}{6} \cdot |S|$
4. S fulfills the weight property

This lemma directly implies the theorem since V fulfills the bandwidth property.

Approach

Notice that all these properties are very local:

1. $S = \cup_i S_i$ ▶ Local to S ◀
2. Each S_i fulfills the bandwidth property ▶ Local to each individual S_i ◀
3. $\forall i: |S_i| \leq \frac{5}{6} \cdot |S|$ ▶ Local to each individual S_i ◀
4. S fulfills the weight property ▶ Local to S and S_i 's ◀

Algorithm: High Level Idea

Maintain a partitioning P_S of S that simultaneously fulfills requirements 1, 2 and 3.

Then change this partitioning successively until requirement 4 is fulfilled as well.

1. $S = \uplus_i S_i$
2. Each S_i fulfills the bandwidth property
3. $\forall i : |S_i| \leq \frac{5}{6} \cdot |S|$
4. S fulfills the weight property

Requirement 1

- $S = \bigoplus_i S_i$

Cannot be any easier...

Requirement 2

- Each S_i fulfills the bandwidth property

If we have a candidate S_i that we would like to add to P_S , decompose S_i further when the bandwidth property is not fulfilled.

Lemma For any subset $R \subseteq V$, it is possible to partition R into disjoint subsets $R_i \subseteq R$ s.t. each R_i fulfills the bandwidth property and

$$\sum_i \text{out}(R_i) \leq 2 \cdot \text{out}(R).$$

Maintaining Requirement 2

BANDWIDTHPARTITION(R)

```
1  $P_R \leftarrow \{R\}$ 
2 while  $\exists R_i \in P_R$  not fulfilling the bandwidth property
3 do find  $U \subseteq R_i$ 
4     with  $|U| \leq \frac{3}{4} \cdot |R_i|$ 
5     and  $\lambda \cdot \text{cap}(U, R_i \setminus U) < \text{out}(U)$ 
6    $P_R \leftarrow P_R \setminus \{R_i\}$ 
7    $P_R \leftarrow P_R \cup \{U, R_i \setminus U\}$ 
8 return  $P_R$ 
```

Requirement 3

- $\forall i : |S_i| \leq \frac{5}{6} \cdot |S|$

First compute a $\frac{1}{6}$ -balanced min cut of S .

Then make sure that any S_i that gets added to P_S is either a subset of A or B .

Requirement 4

- S fulfills the weight property

Use the sharpened weight condition trick. Tough!!!

A subset $U \subset S$ violates the sharpened weight condition iff

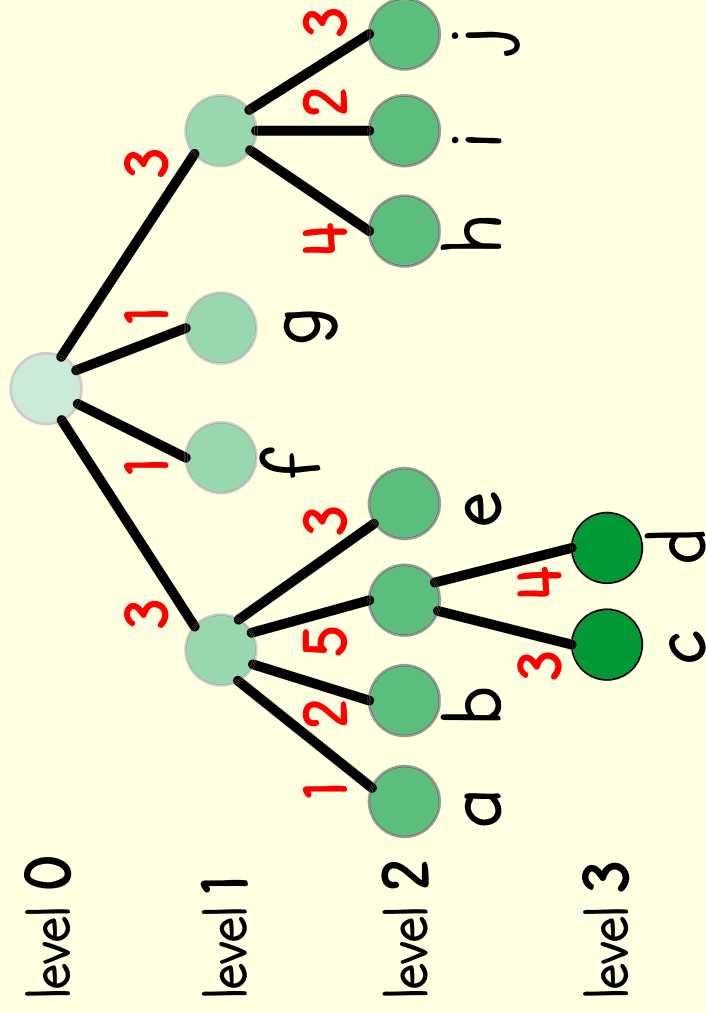
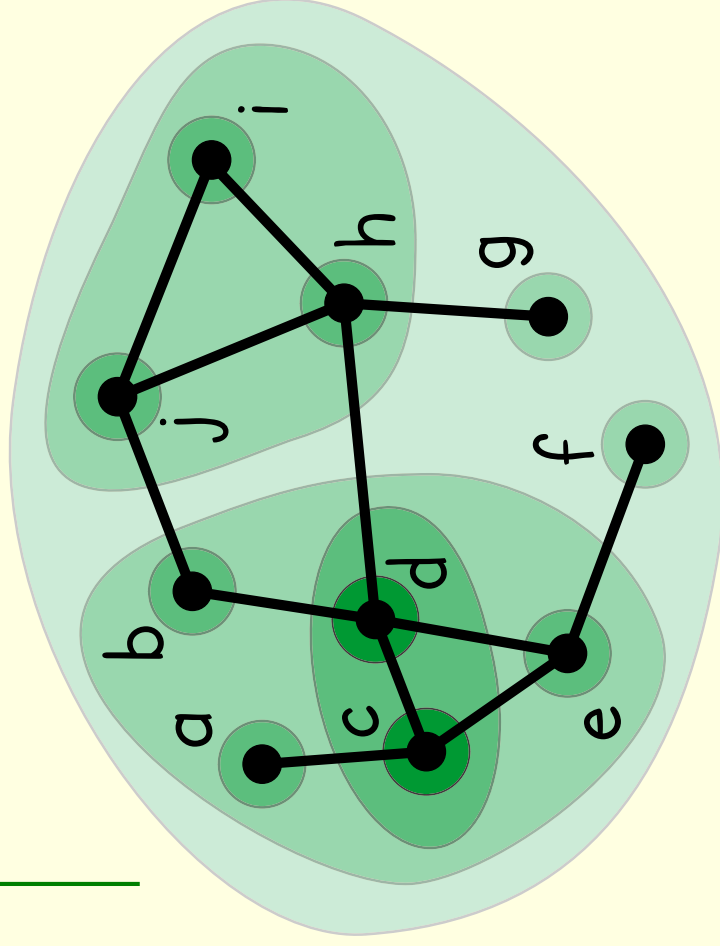
$$w_{U+1}(U) > 6 \cdot \text{out}(U)$$

for $|U| \leq \frac{2}{3} \cdot |S|$ and $\exists Q \subset P_S : U = \bigoplus_{S_i \in Q} S_i$.

Interplay: $\text{out}(U)$ and $w_{l+1}(U)$

$$w_{l+1}(U) > 6 \cdot \text{out}(U)$$

for $|U| \leq \frac{2}{3} \cdot |S|, \exists Q \subset P_S : U = \bigsqcup_{S_i \in Q} S_i$



PARTITION(S)

PARTITION(S)

1 $(A, B) \leftarrow \frac{1}{6}$ -balanced min cut S

2 $P_S \leftarrow \{\{v\} \mid v \in S\}$

3 **while** $\exists U$ that violates the sharpened weight condition

4 **do for each** $S_i \in Q$

5 **do** $P_S \leftarrow P_S \setminus \{S_i\}$

6 $P_S \leftarrow P_S \cup \text{BANDWIDTHPARTITION}(U \cap A)$

7 $P_S \leftarrow P_S \cup \text{BANDWIDTHPARTITION}(U \cap B)$

8 **return** P_S

Proof of Correctness

“By this time either everyone has left or no one is awake.”

The paper is available online. :P

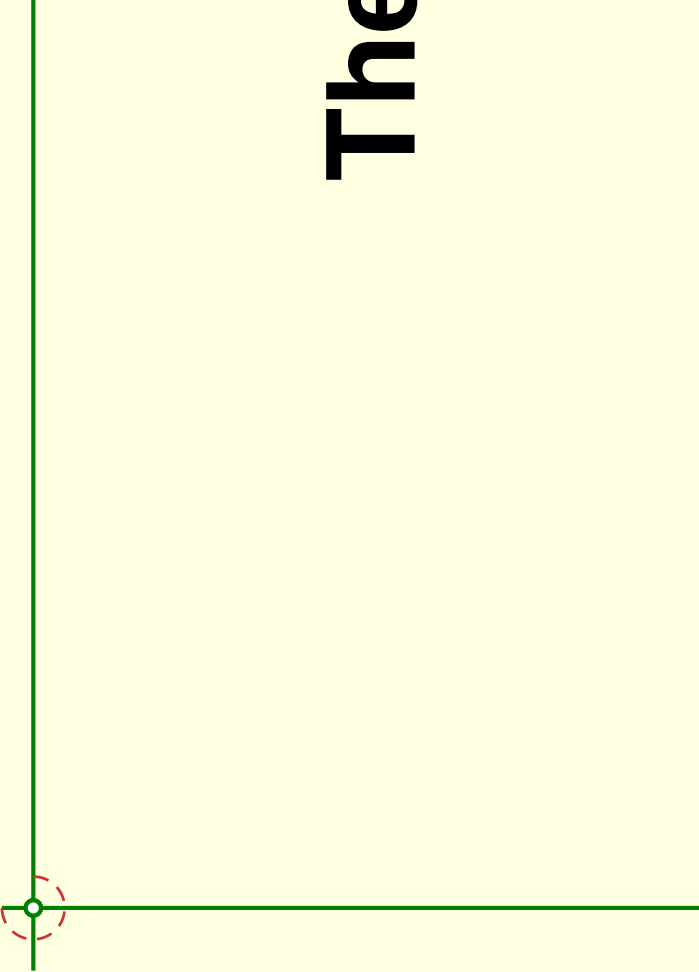

Running Time Analysis

The algorithm we presented is not quite efficient.

- Solves a NP-complete problem for each cluster.
- Finds a violating subset by perhaps enumerating all possible subsets.

But it only needs to be done once for a graph.

Then any online problem can be solved with good competitive ratio by applying the framework.



The End