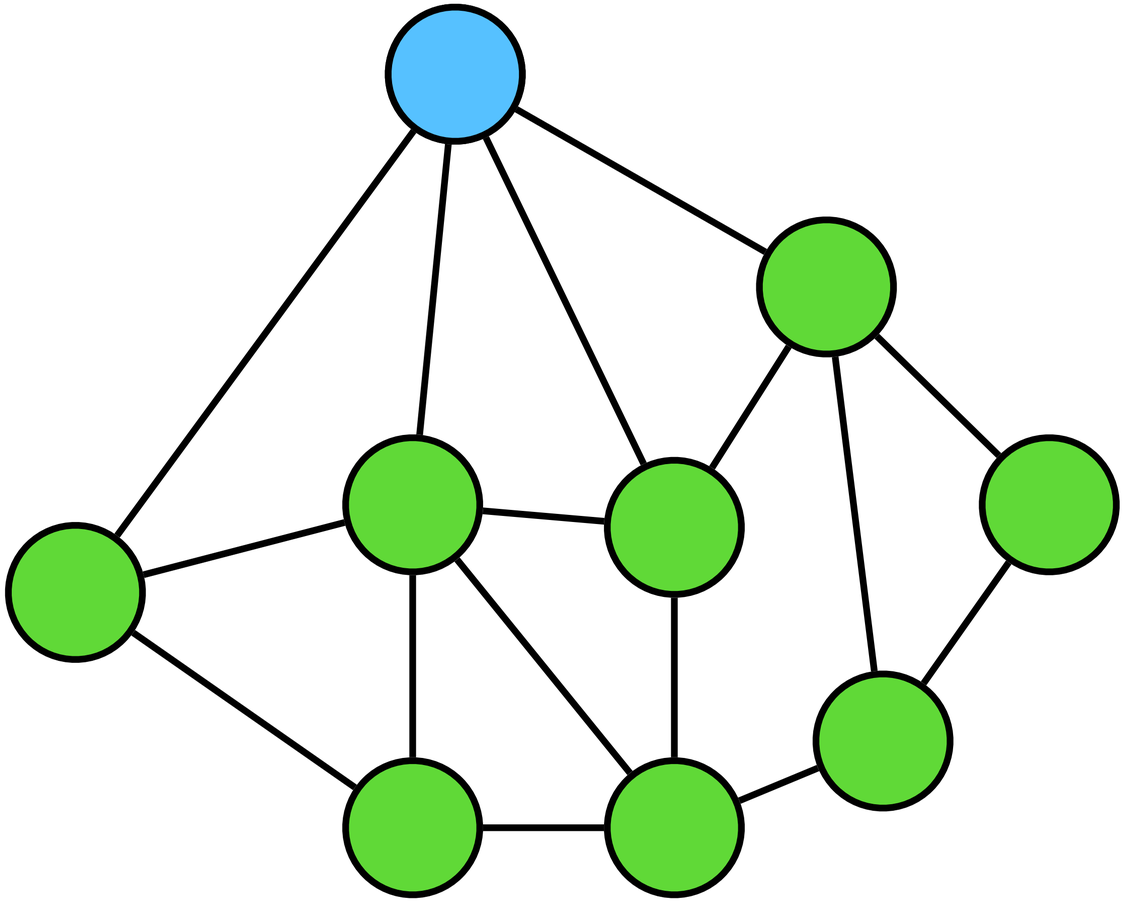
Planar Length-Constrained Minimum Spanning Trees

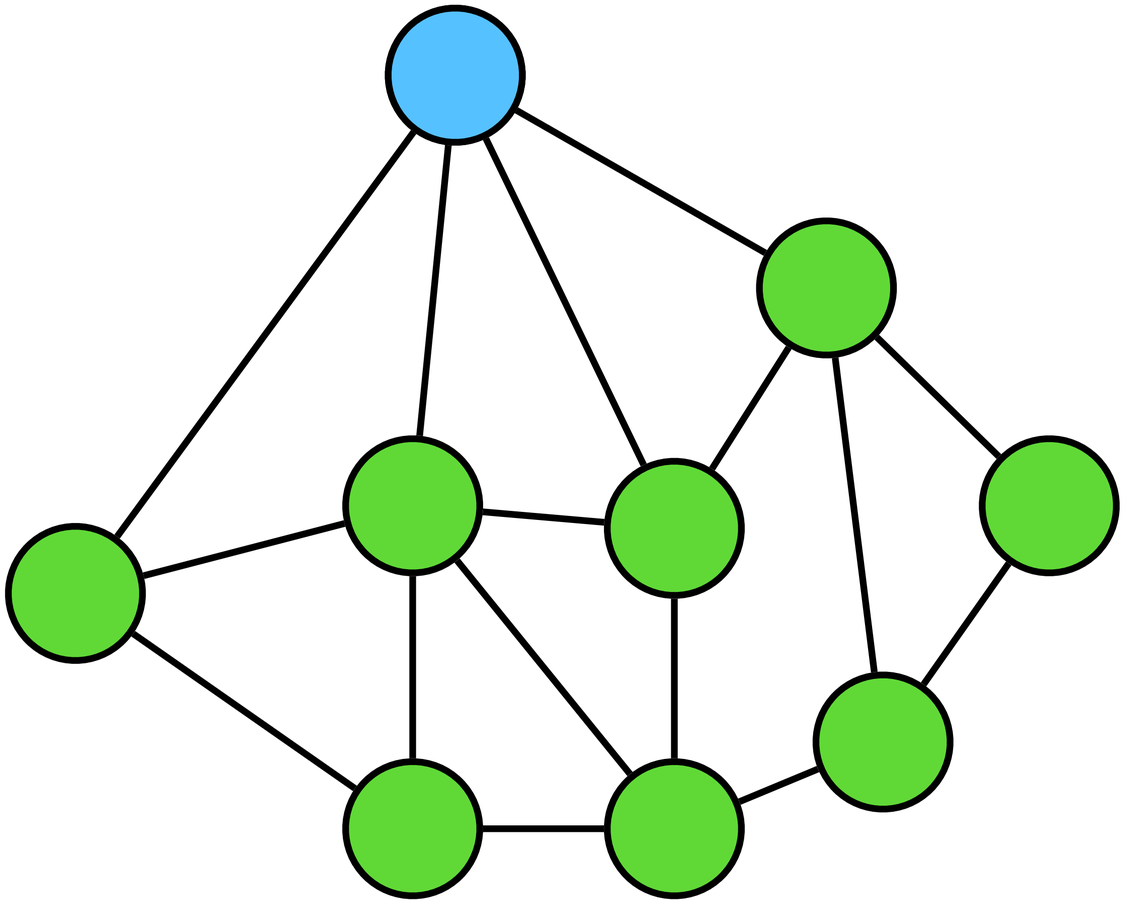
Ellis Hershkowitz & Richard Huang
Brown University

Length-Constrained
Minimum Spanning Tree (MST)



Length-Constrained Minimum Spanning Tree (MST)

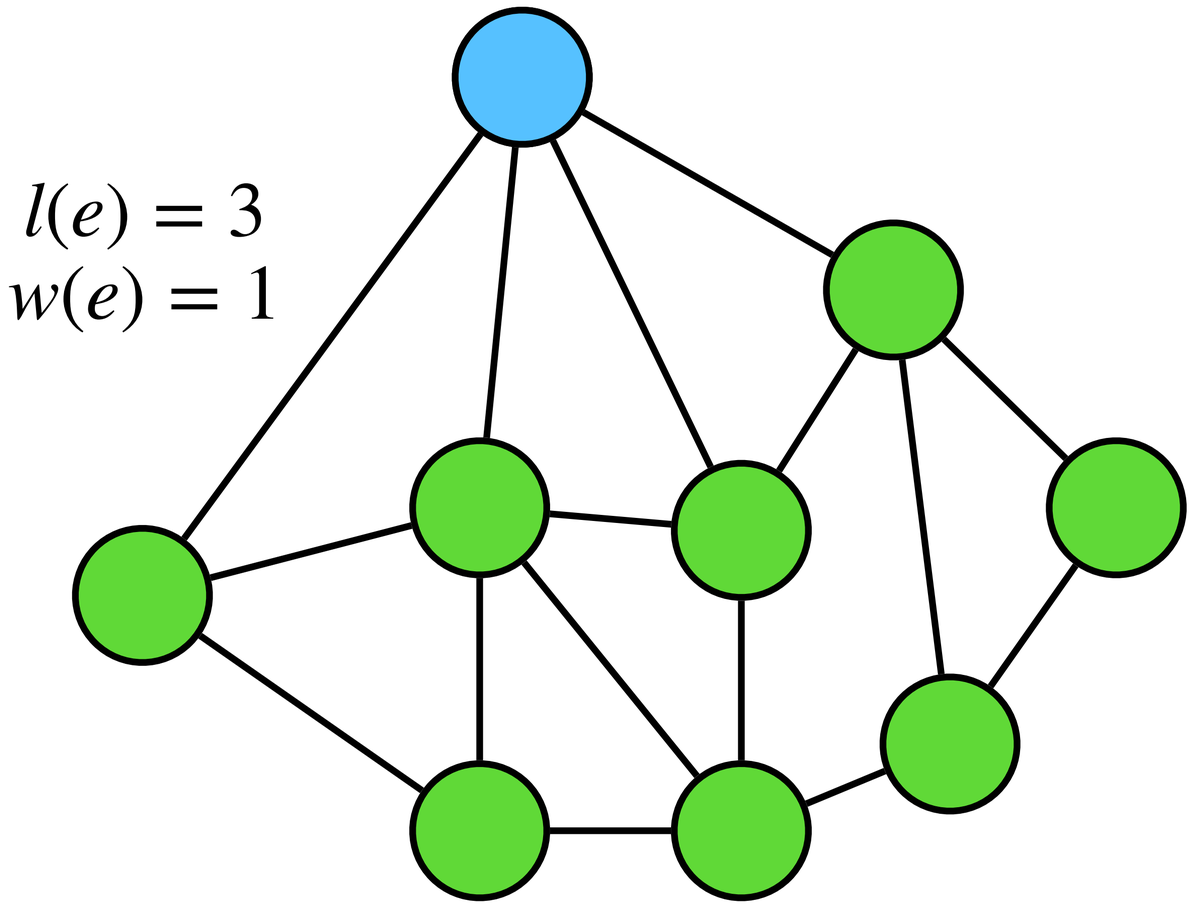
Given connected graph $G = (V, E)$



Length-Constrained Minimum Spanning Tree (MST)

Given connected graph $G = (V, E)$

with edge lengths and weights l, w,

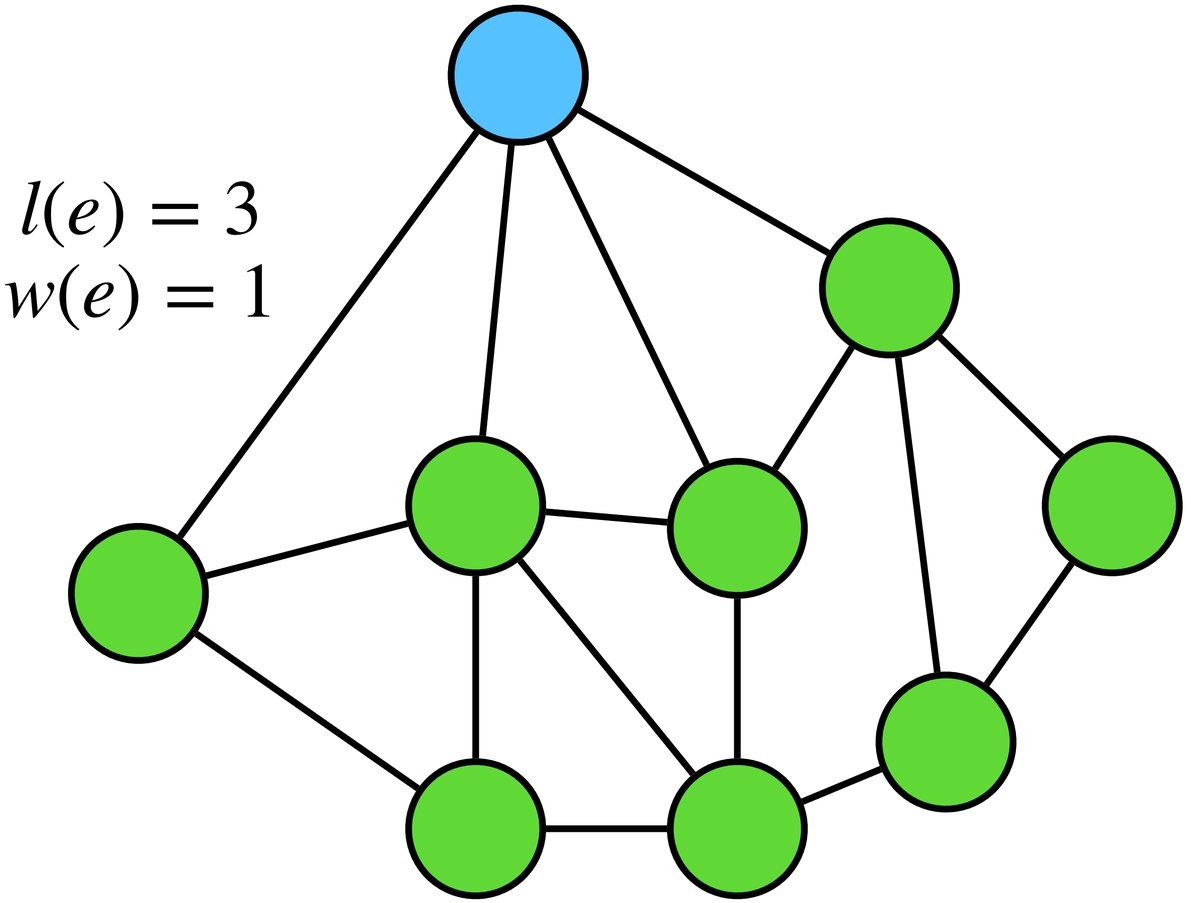


Length-Constrained Minimum Spanning Tree (MST)

Given connected graph $G = (V, E)$

with edge lengths and weights l, w,

root $r \in V$,



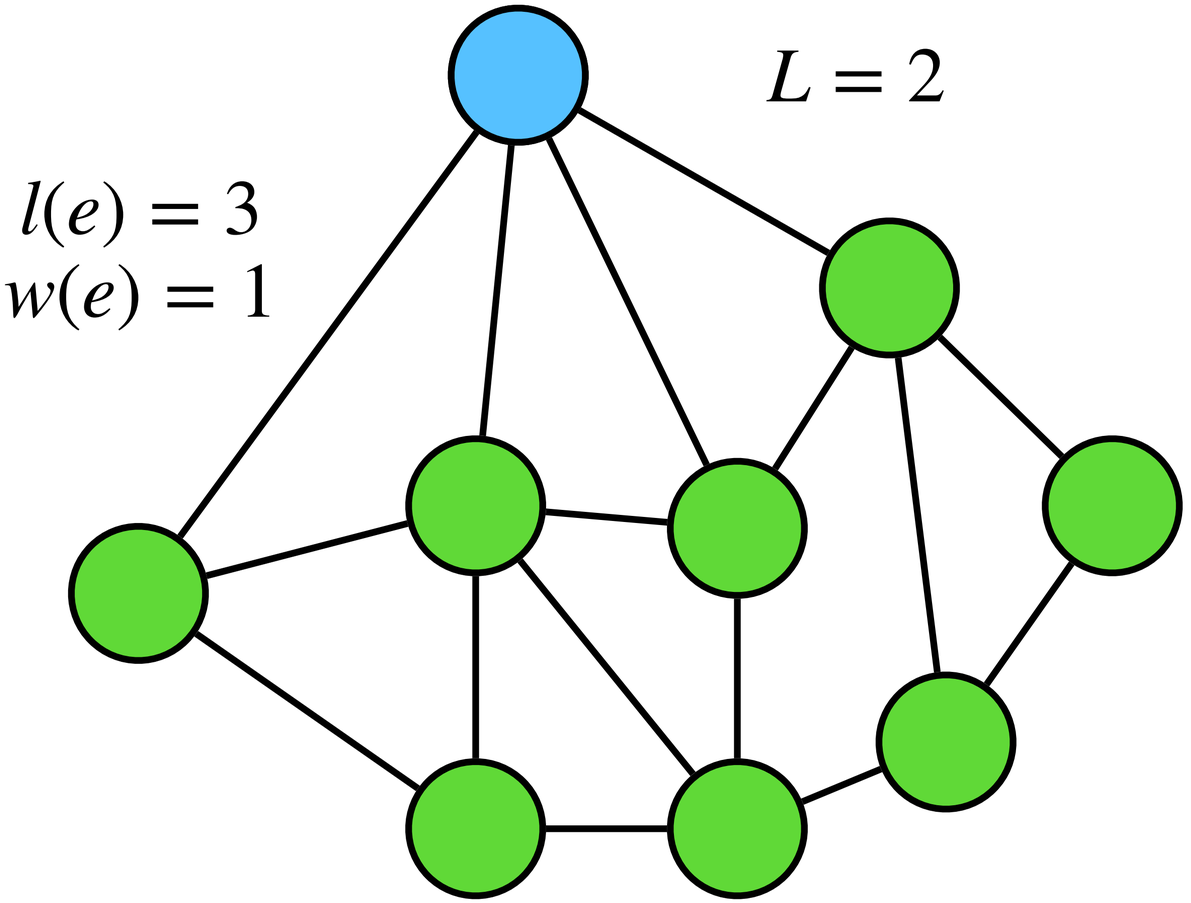
Length-Constrained Minimum Spanning Tree (MST)

Given connected graph $G = (V, E)$

with edge lengths and weights l, w,

root $r \in V$,

length constraint $L > 0$…



Length-Constrained Minimum Spanning Tree (MST)

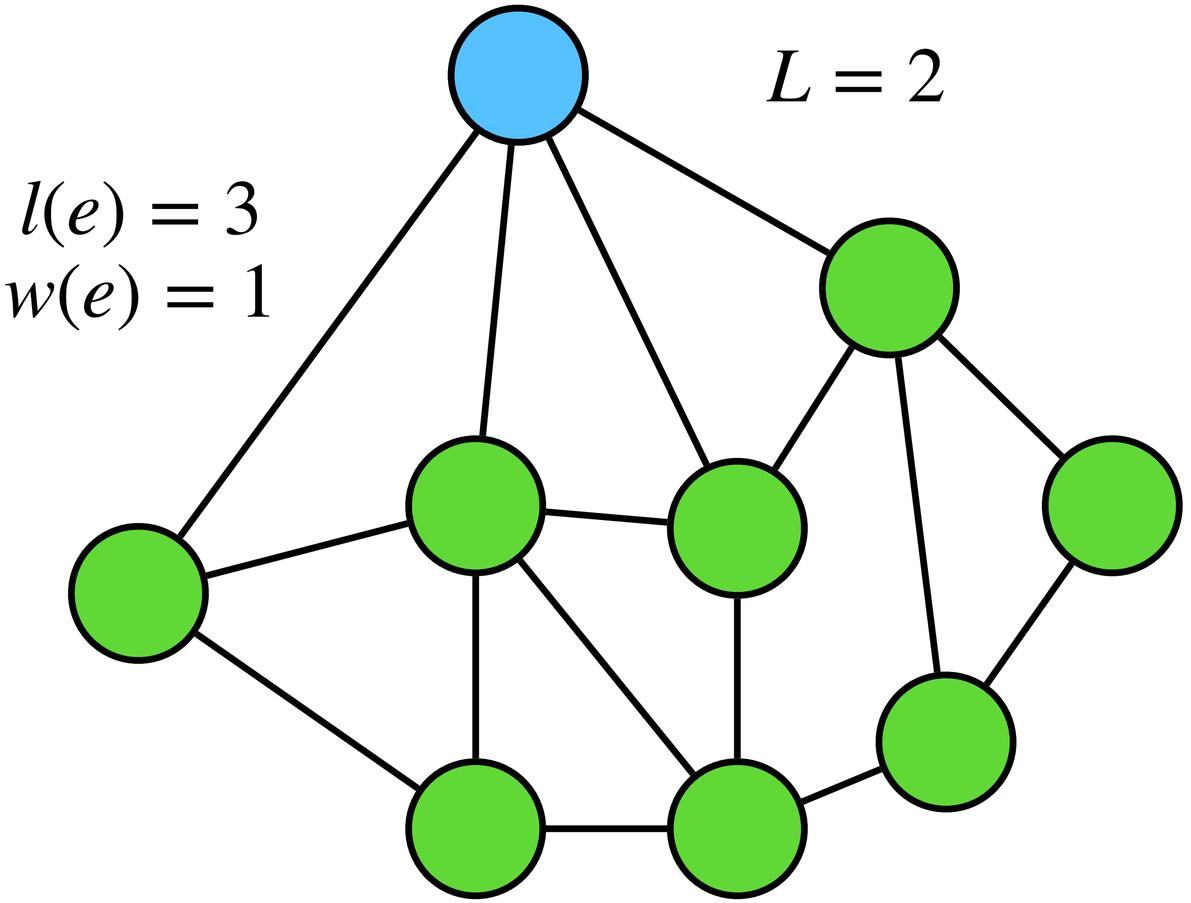
Given connected graph $G = (V, E)$

with edge lengths and weights l, w,

root $r \in V$,

length constraint $L > 0$…

find a spanning tree of G such that



Length-Constrained Minimum Spanning Tree (MST)

Given connected graph $G = (V, E)$

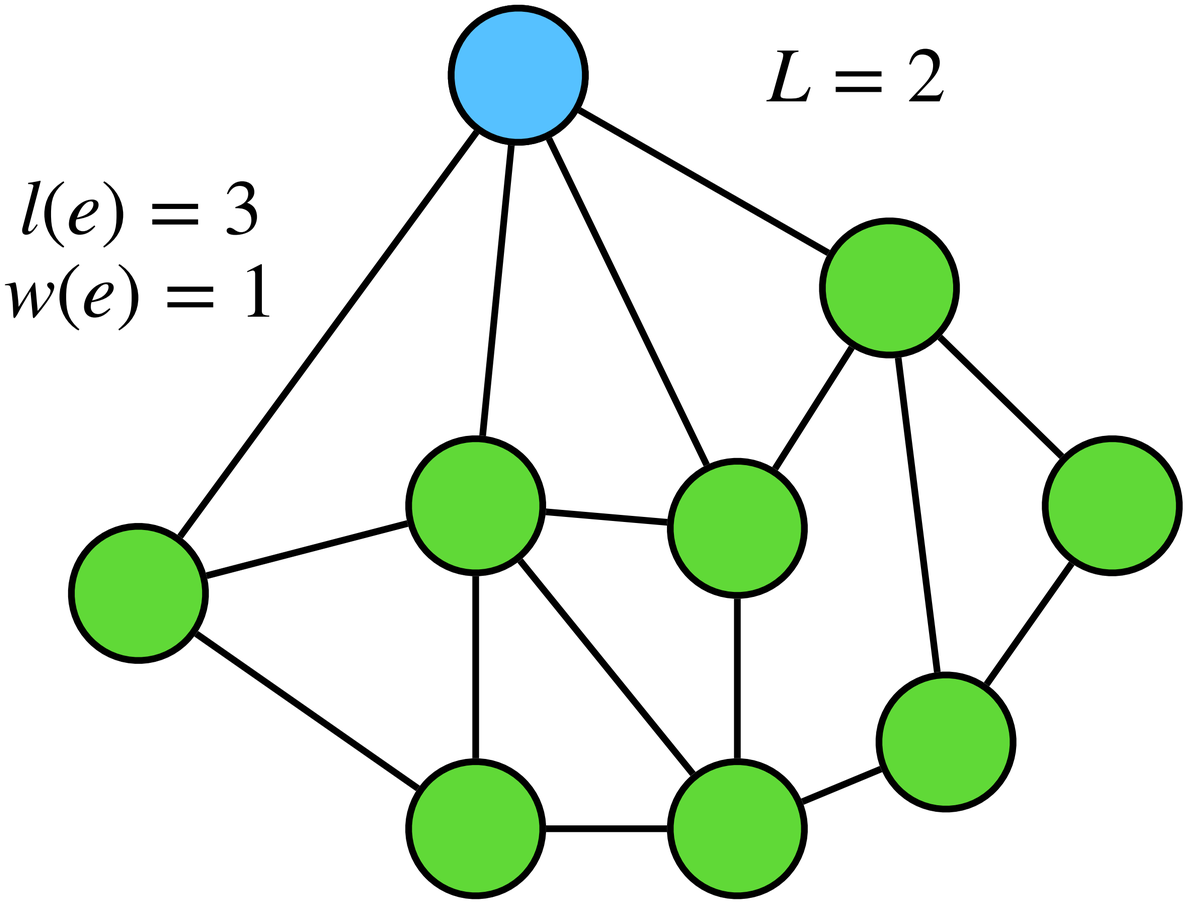
with edge lengths and weights l, w,

root $r \in V$,

length constraint $L > 0$…

find a spanning tree of G such that

there is an $\leq L$-length r, v path $\forall v \in V$



Length-Constrained Minimum Spanning Tree (MST)

Given connected graph $G = (V, E)$

with edge lengths and weights l, w,

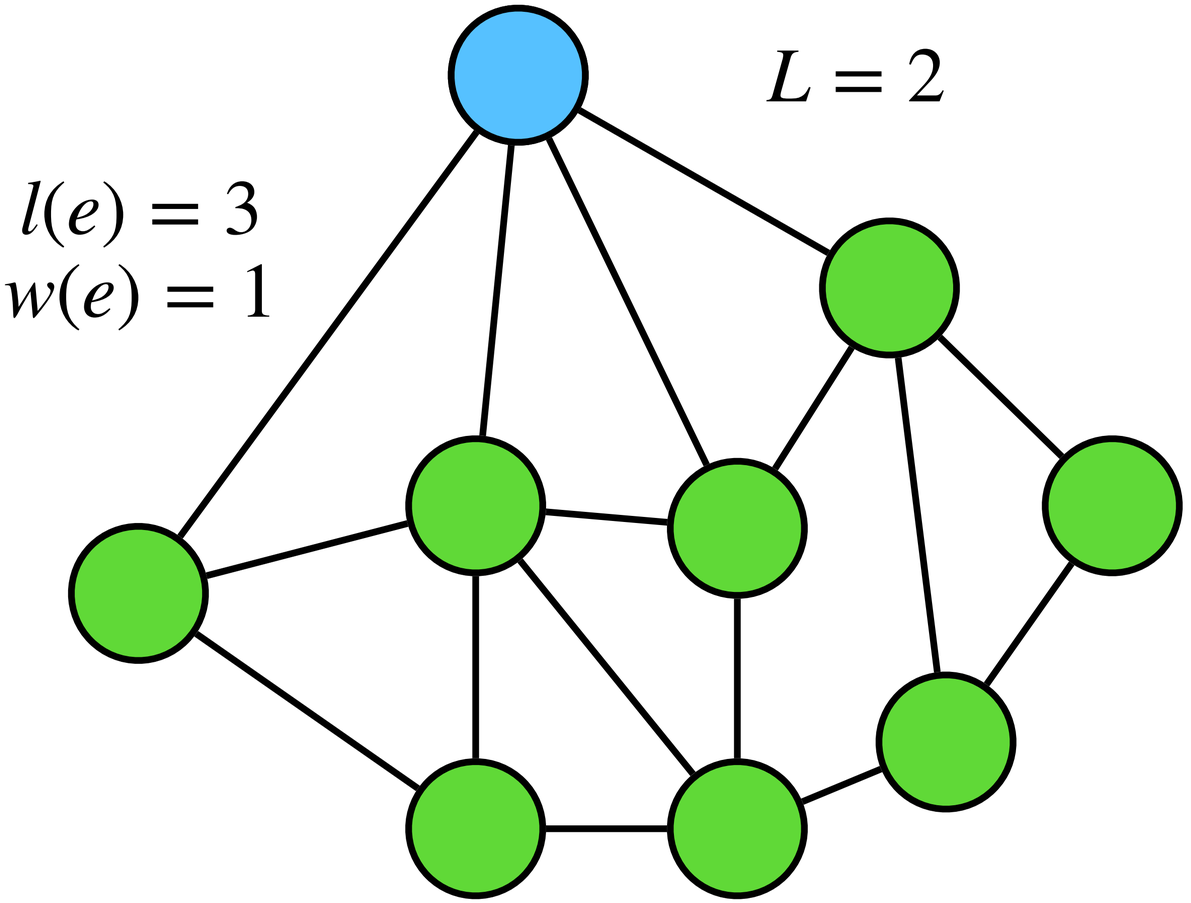
root $r \in V$,

length constraint $L > 0$...

find a spanning tree of G such that

there is an $\leq L$-length r, v path $\forall v \in V$

minimizing the sum of edge weights



Length-Constrained Minimum Spanning Tree (MST)

Given connected graph $G = (V, E)$

with edge lengths and weights l, w,

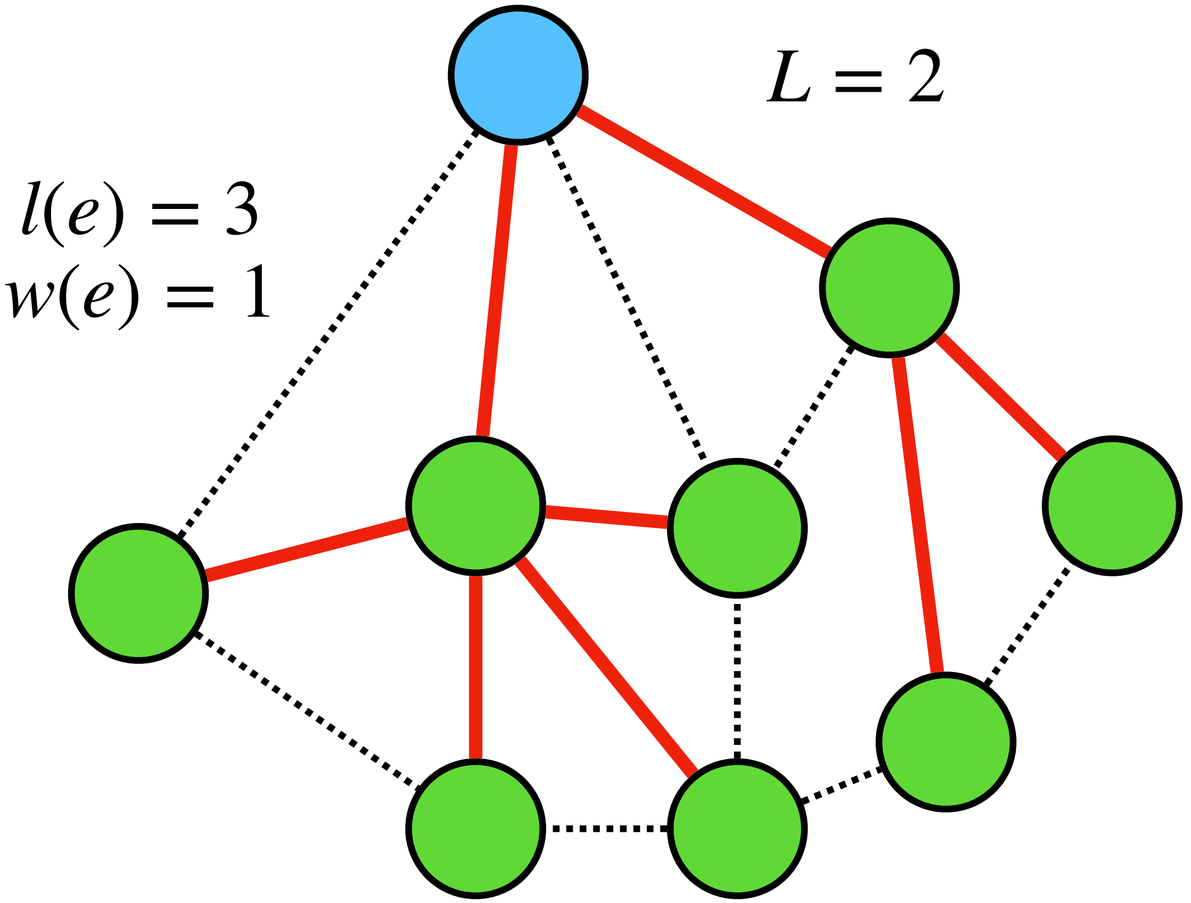
root $r \in V$,

length constraint $L > 0$...

find a spanning tree of G such that

there is an $\leq L$-length r, v path $\forall v \in V$

minimizing the sum of edge weights



Approximating
Length-Constrained MST

Approximating Length-Constrained MST

NP-hard problem

Approximating Length-Constrained MST

NP-hard problem

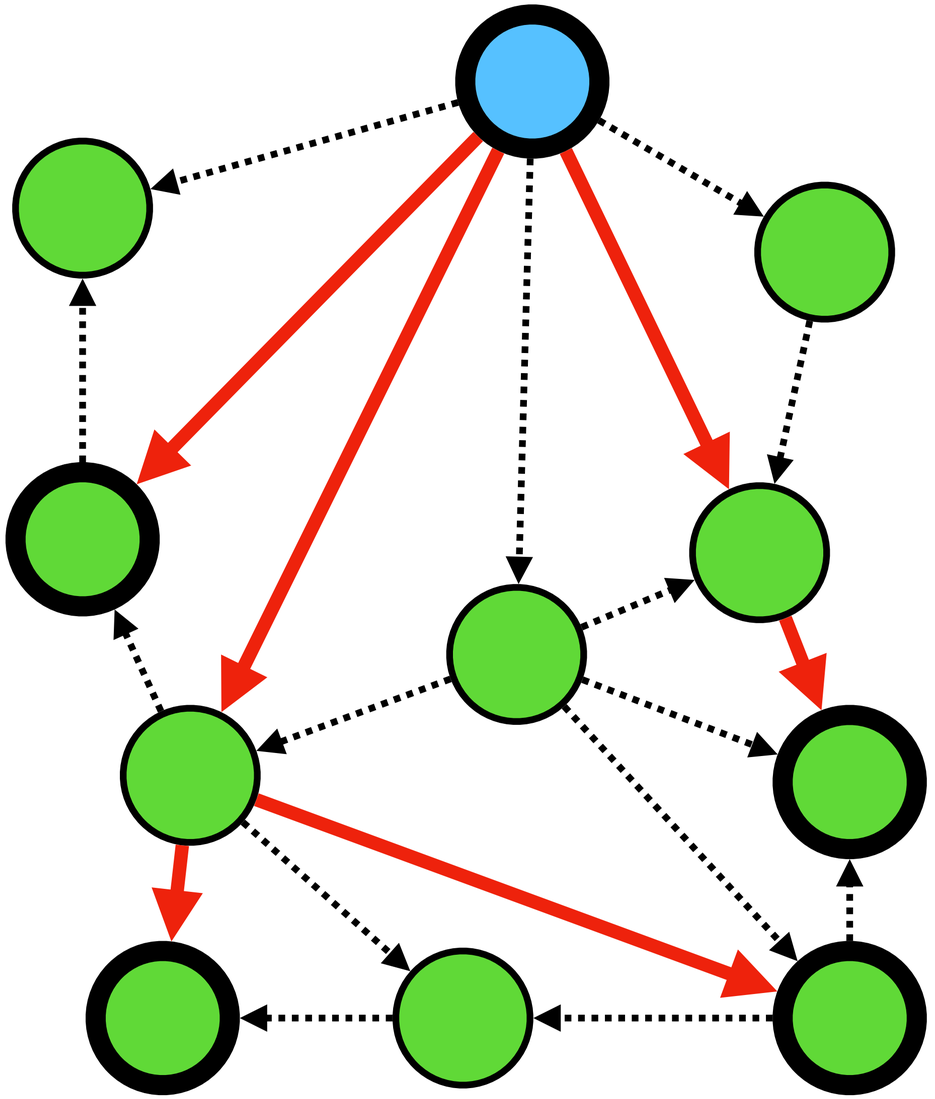
$\Longrightarrow$ approximation algorithms

Approximating
Length-Constrained MST

NP-hard problem

$\Longrightarrow$ approximation algorithms

Equivalent to Directed Steiner Tree



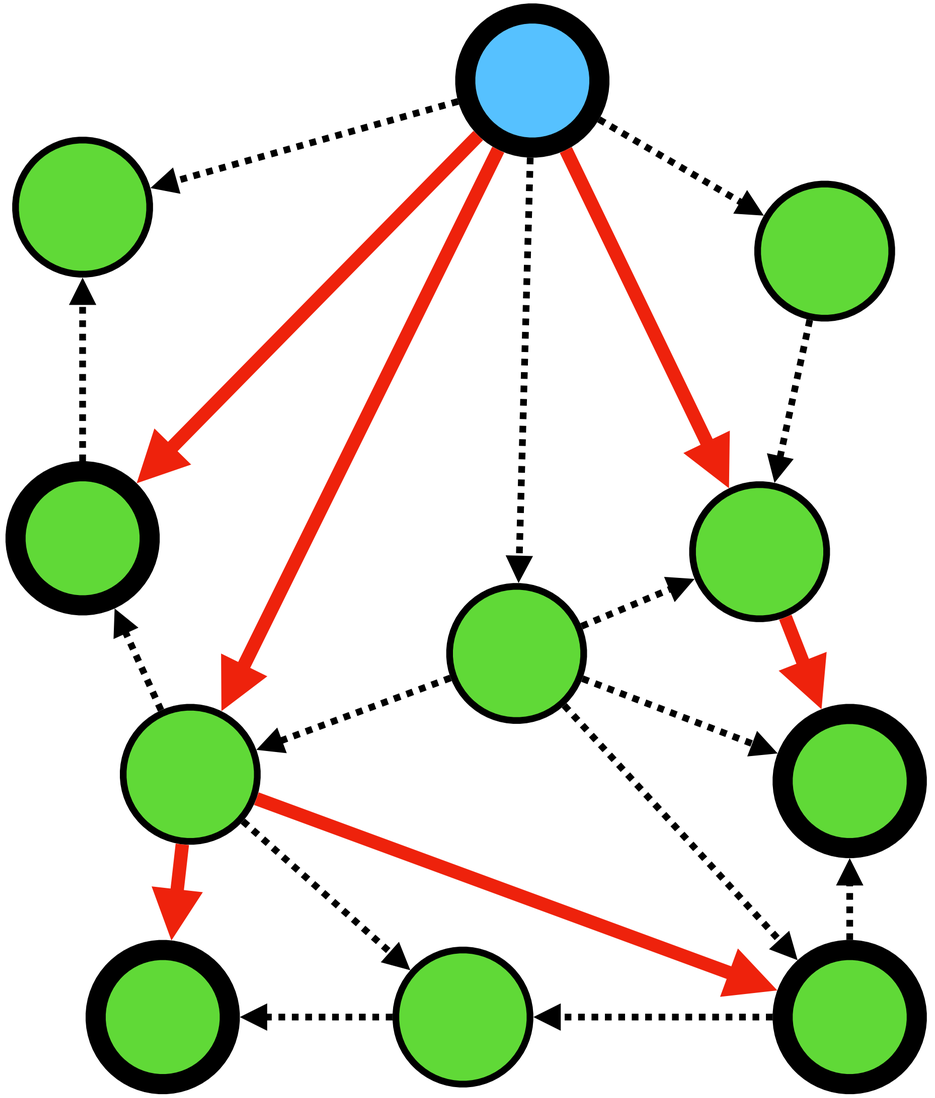
Approximating
Length-Constrained MST

NP-hard problem

$\Longrightarrow$ approximation algorithms

Equivalent to Directed Steiner Tree

$\Longrightarrow$ ***bicriteria*** approximation algorithms



Bicriteria Approximating Length-Constrained MST

An α **approximation**

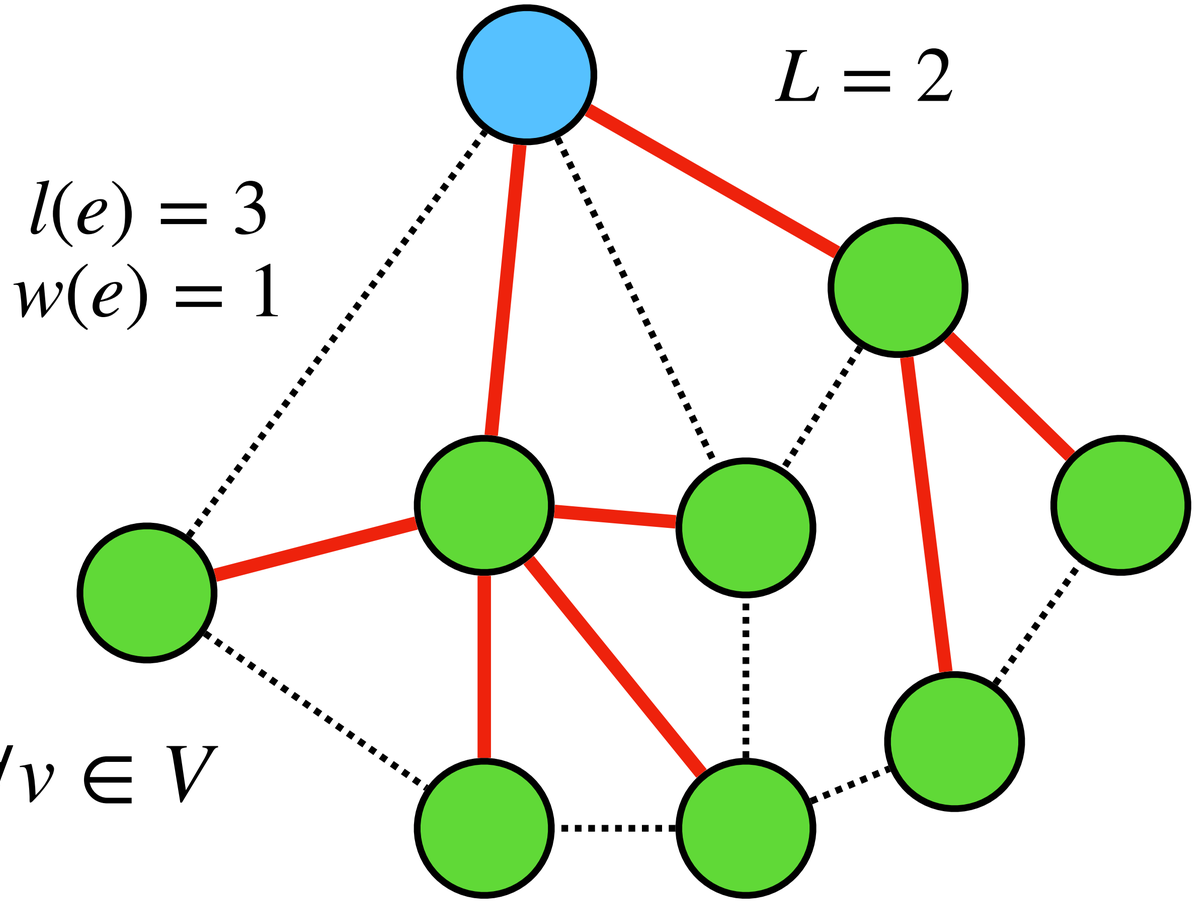
for length-constrained MST

with **length slack** β

gives a spanning tree s.t.

there is a $\leq (\beta \cdot L)$-length r, v path $\forall v \in V$

with weight $\leq \alpha \cdot \text{OPT}$



Bicriteria Approximating Length-Constrained MST

An α **approximation**

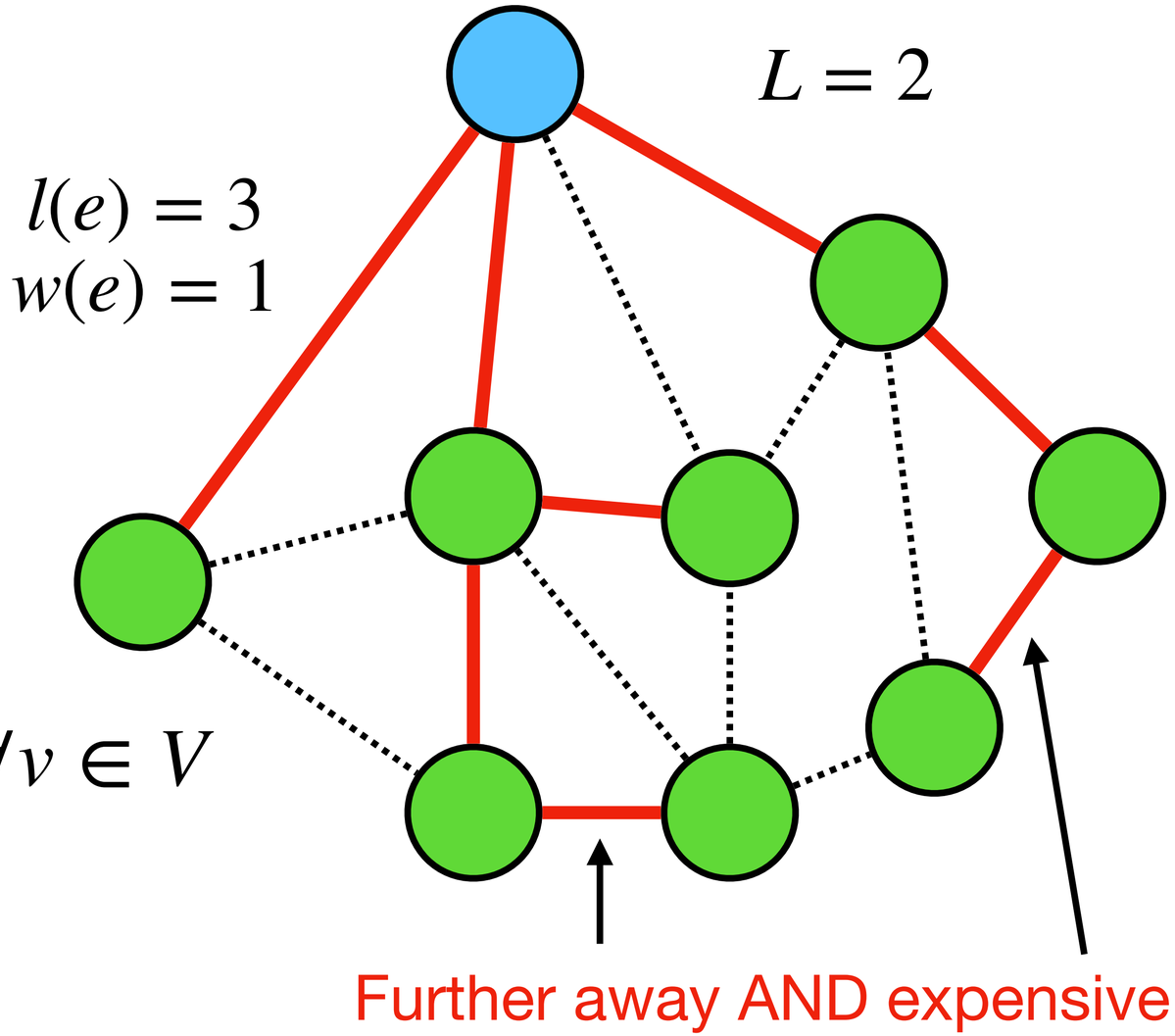
for length-constrained MST

with **length slack** β

gives a spanning tree s.t.

there is a $\leq (\beta \cdot L)$-length r, v path $\forall v \in V$

with weight $\leq \alpha \cdot \text{OPT}$



Bicriteria Approximating Length-Constrained MST

How good of a weight approximation can we get while keeping length-slack low?

Bicriteria Approximating Length-Constrained MST

How good of a weight approximation can we get while keeping length-slack low?

n^{ϵ} weight approximation, 1 length slack; CCCDGGL (1998)

Bicriteria Approximating Length-Constrained MST

How good of a weight approximation can we get while keeping length-slack low?

n^{ϵ} weight approximation, 1 length slack; CCCDGGL (1998)

$\log(n)$ weight approximation, $\log(n)$ length slack; MRSRRH (1995)

Bicriteria Approximating Length-Constrained MST

How good of a weight approximation can we get while keeping length-slack low?

n^{ϵ} weight approximation, 1 length slack; CCCDGGL (1998)

$\log(n)$ weight approximation, $\log(n)$ length slack; MRSRRH (1995)

polylog(n) weight approximation, $O(1)$ length slack

Bicriteria Approximating Length-Constrained Steiner Tree

How good of a weight approximation can we get while keeping length-slack low?

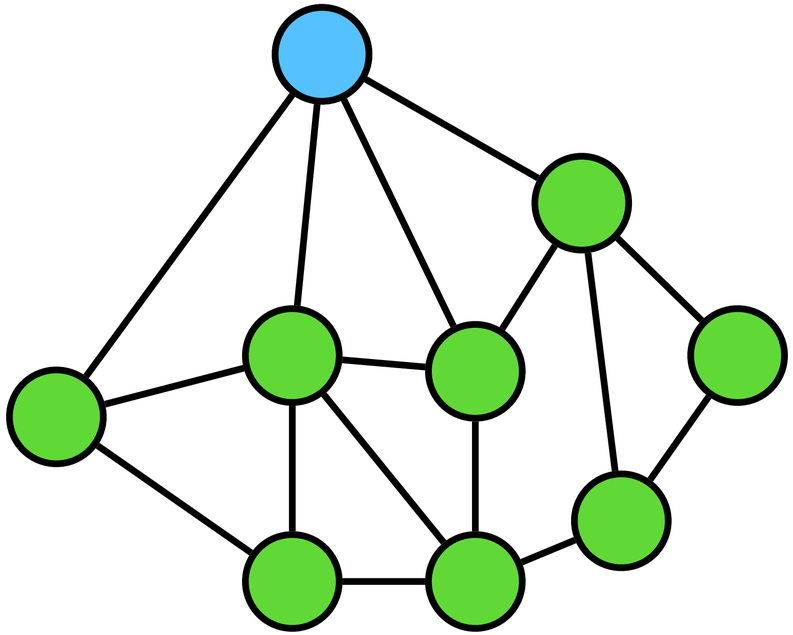
t = # of terminals

t^{ϵ} weight approximation, 1 length slack; CCCDGGL (SODA 1998)

$\log(t)$ weight approximation, $\log(t)$ length slack; MRSRRH (1995)

Bicriteria Approximating Length-Constrained MST

How good of a weight approximation can we get while keeping length-slack low *when restricted to planar graphs*?



Planar directed Steiner tree

Planar directed Steiner tree

Good news: $\log n$ approximation for planar directed Steiner tree by Friggstad and Mousavi (2023)

Planar directed Steiner tree

Good news: $\log n$ approximation for planar directed Steiner tree by Friggstad and Mousavi (2023)

$\log n$ weight approximation for planar length-constrained MST with 1 length slack?

Planar directed Steiner tree

Good news: $\log n$ approximation for planar directed Steiner tree by Friggstad and Mousavi (2023)

$\log n$ weight approximation for planar length-constrained MST with 1 length slack?

But reduction between directed Steiner tree and length-constrained MST doesn't preserve planarity

Planar directed Steiner tree

Good news: $\log n$ approximation for planar directed Steiner tree by Friggstad and Mousavi (2023)

$\log n$ weight approximation for planar length-constrained MST with 1 length slack?

But reduction between directed Steiner tree and length-constrained MST doesn't preserve planarity

Planar directed Steiner tree

Good news: $\log n$ approximation for planar directed Steiner tree by Friggstad and Mousavi (2023)

~~$\log n$ weight approximation for planar length-constrained MST with 1 length slack?~~

But reduction between directed Steiner tree and length-constrained MST doesn't preserve planarity

Our Result

How good of a weight approximation can we get while keeping length-slack low *when restricted to planar graphs*?

Our Result

How good of a weight approximation can we get while keeping length-slack low *when restricted to planar graphs*?

Our Result

How good of a weight approximation can we get while keeping length-slack low *when restricted to planar graphs*?

1. A new *bicriteria* hardness for length-constrained MST

Our Result

How good of a weight approximation can we get while keeping length-slack low *when restricted to planar graphs*?

1. A new *bicriteria* hardness for length-constrained MST
2. An algorithm for planar graphs that beats the hardness

Our Result

How good of a weight approximation can we get while keeping length-slack low *when restricted to planar graphs*?

1. A $\log^{2-\epsilon} n$ - approximation must have ≥ 2 length slack for general graphs
2. A $\log^{1+\epsilon} n$ - approximation with $1+\epsilon$ length slack for planar graphs via *length-constrained separator hierarchies*

Our Result

How good of a weight approximation can we get while keeping length-slack low *when restricted to planar graphs*?

1. A $\log^{2-\epsilon} n$ - approximation must have ≥ 2 length slack for general graphs
2. A $\log^{1+\epsilon} n$ - approximation with $1+\epsilon$ length slack for planar graphs via *length-constrained separator hierarchies*

*runtime dependent on ϵ

This Talk

This Talk

The main tool: length-constrained separator hierarchies

This Talk

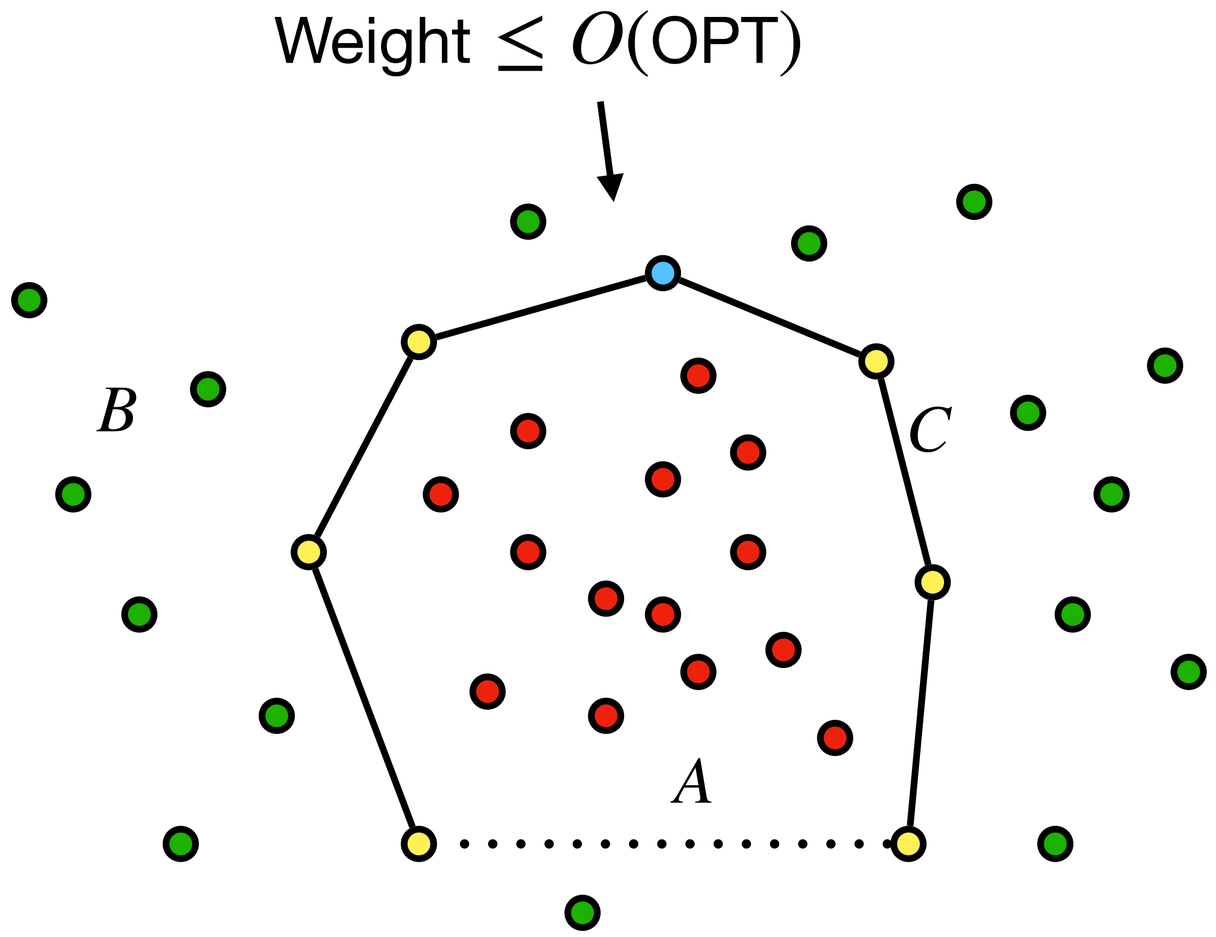
The main tool: length-constrained separator hierarchies

Sketch of the main algorithm

Classic planar separators

A fundamental cycle C of a tree T of G such that $G - C$ partitions into two sides A, B such that

- **Balanced**: $|A|, |B| \leq 2n/3$
- **Separated:** No edges between A, B and in the planar embedding A (resp. B) is completely inside (resp. outside) of C

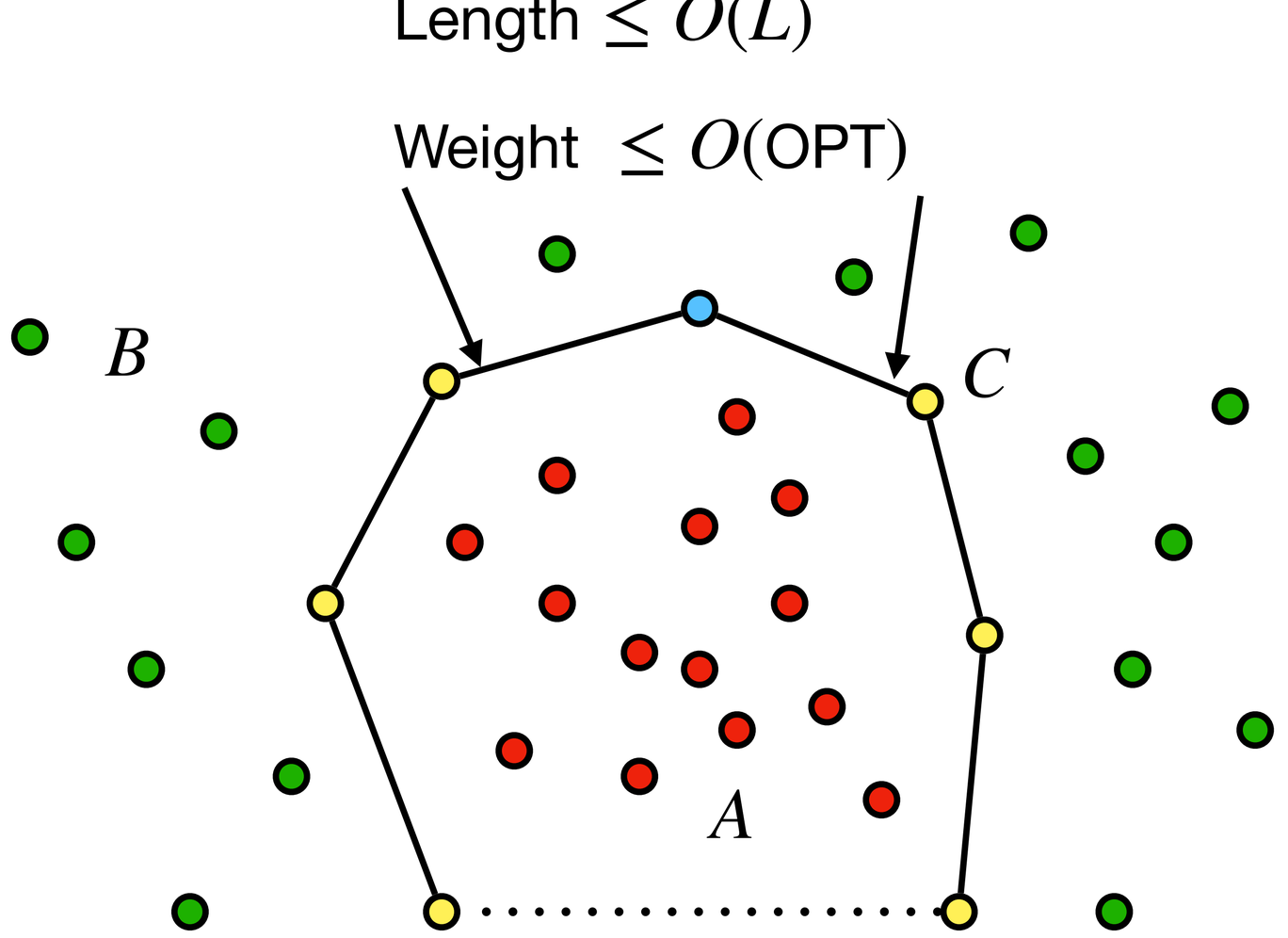


Can further guarantee C is made up of 2 shortest paths

Length-constrained separators

A fundamental cycle C of a tree T of G such that $G - C$ partitions into two sides A, B such that

- **Balanced**: $|A|, |B| \leq 2n/3$
- **Separated:** No edges between $A, B$and in the planar embedding A (resp. B) is completely inside (resp. outside) of C
- **Length-constrained**: C has length $O(L)$ and weight $O(\text{OPT})$

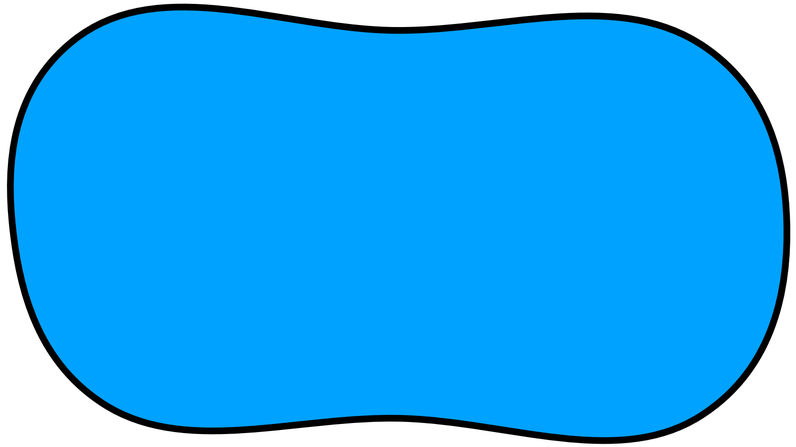


Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a decomposition hierarchy

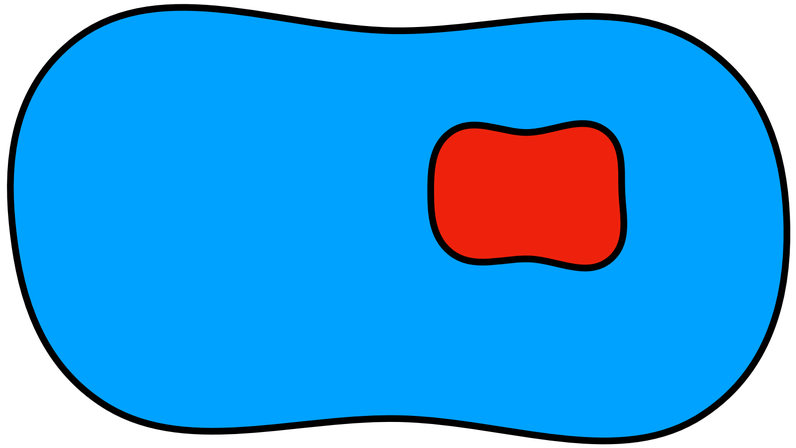
Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a decomposition hierarchy



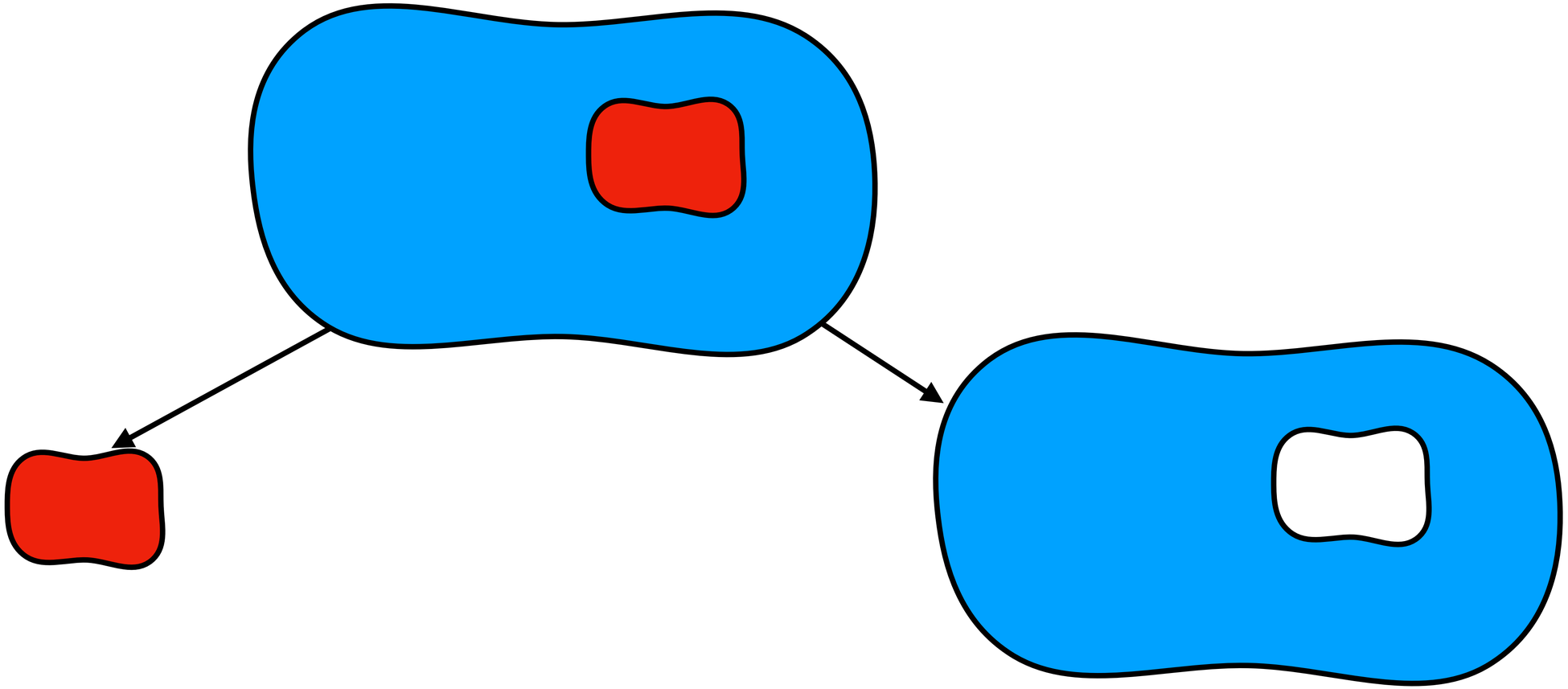
Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a decomposition hierarchy



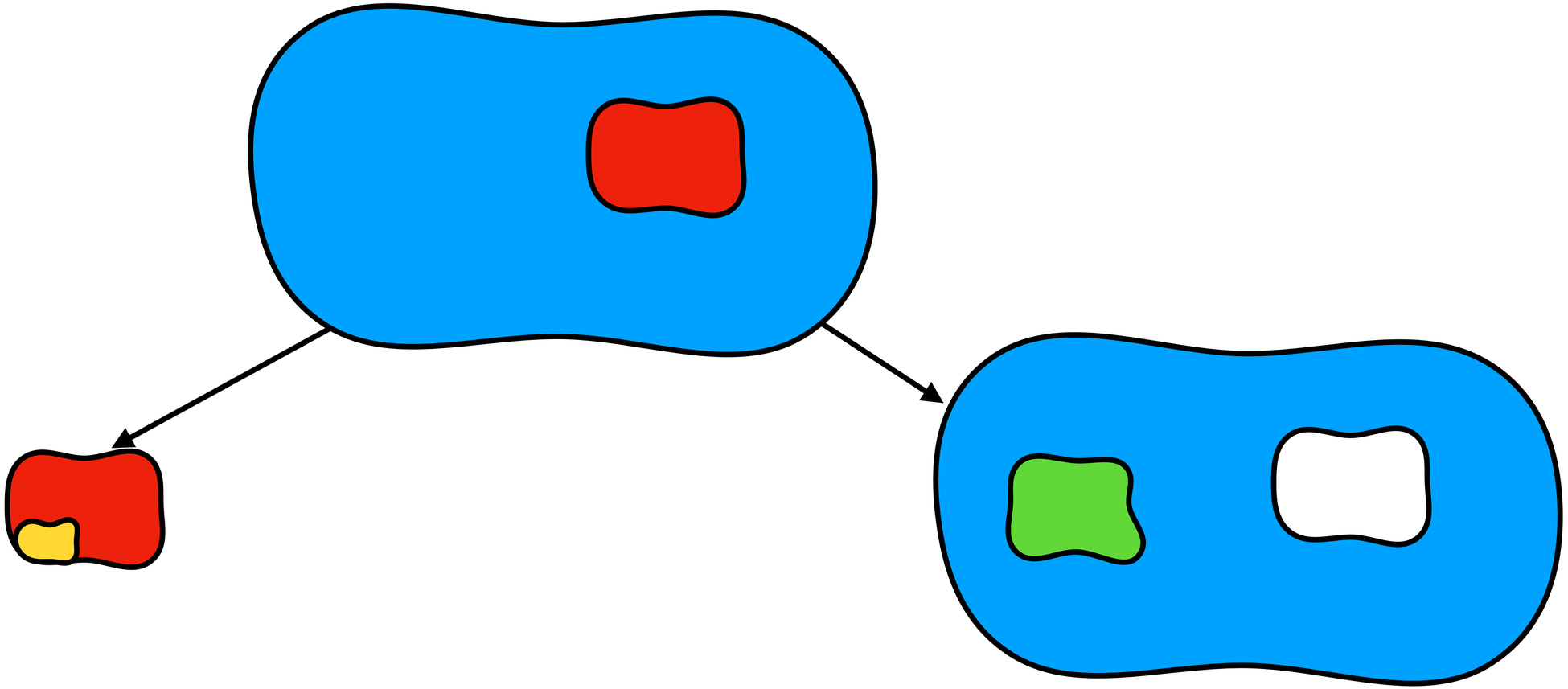
Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a decomposition hierarchy



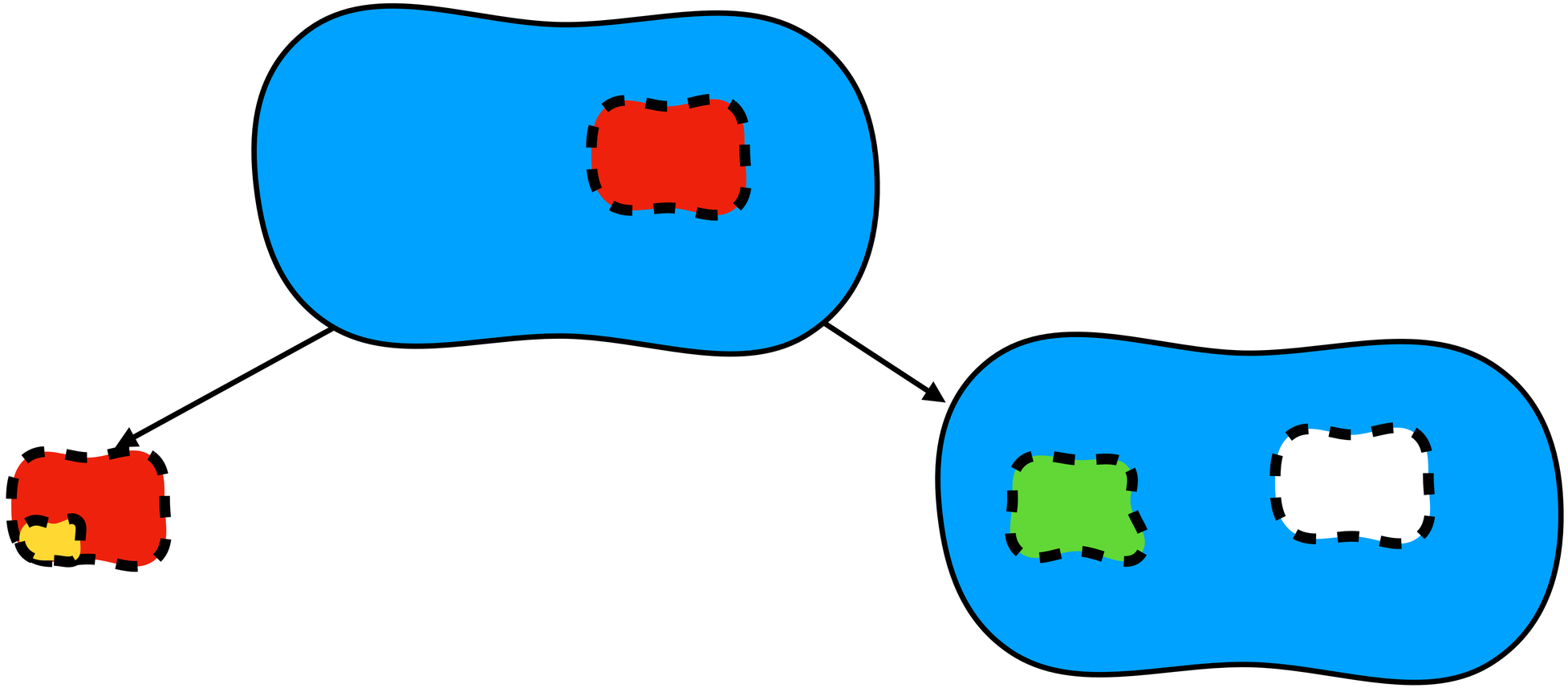
Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a decomposition hierarchy



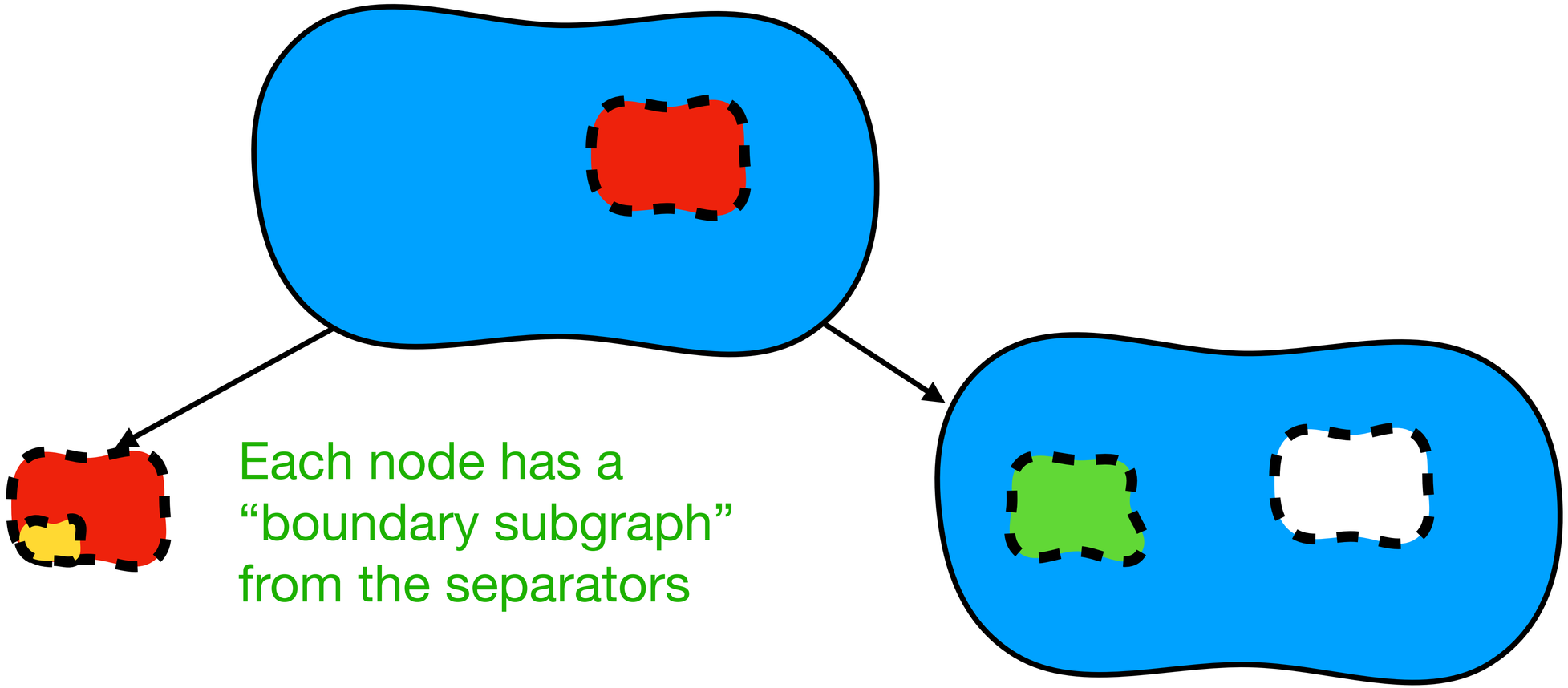
Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a decomposition hierarchy



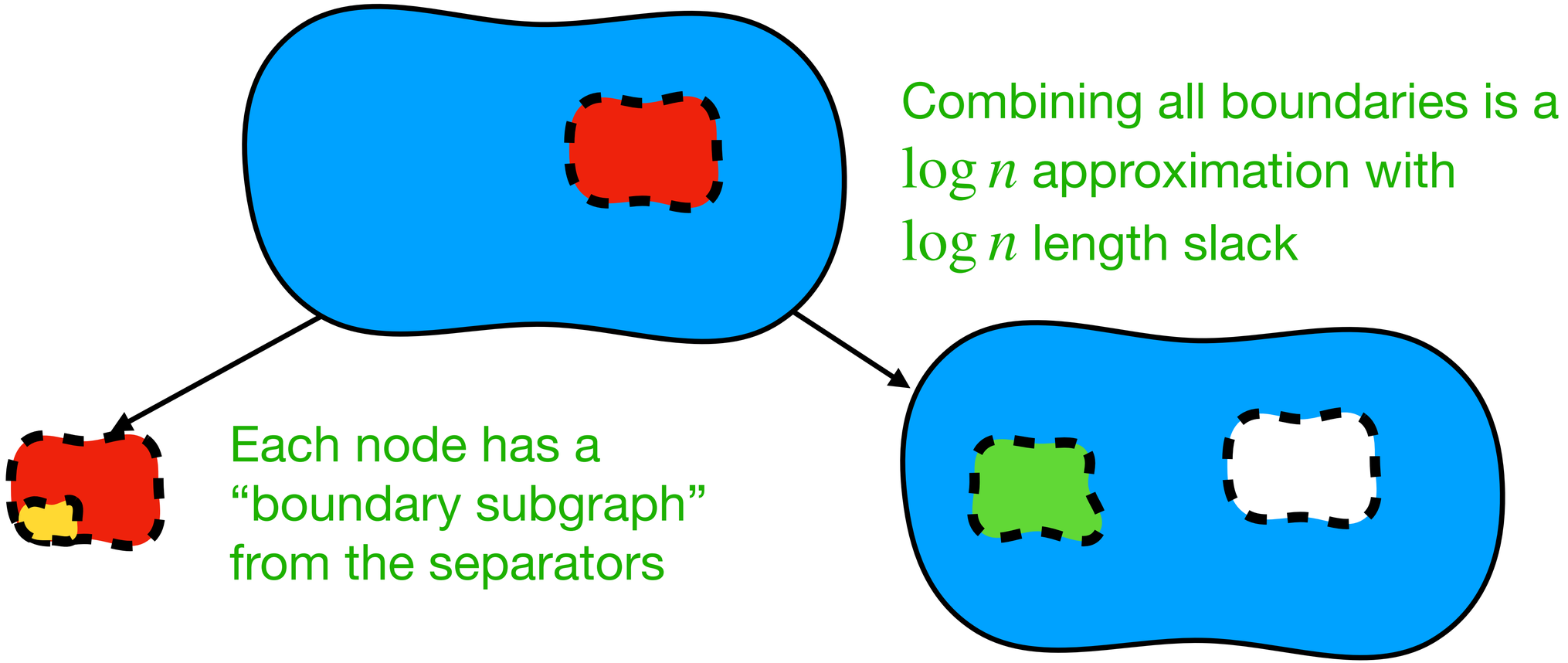
Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a decomposition hierarchy



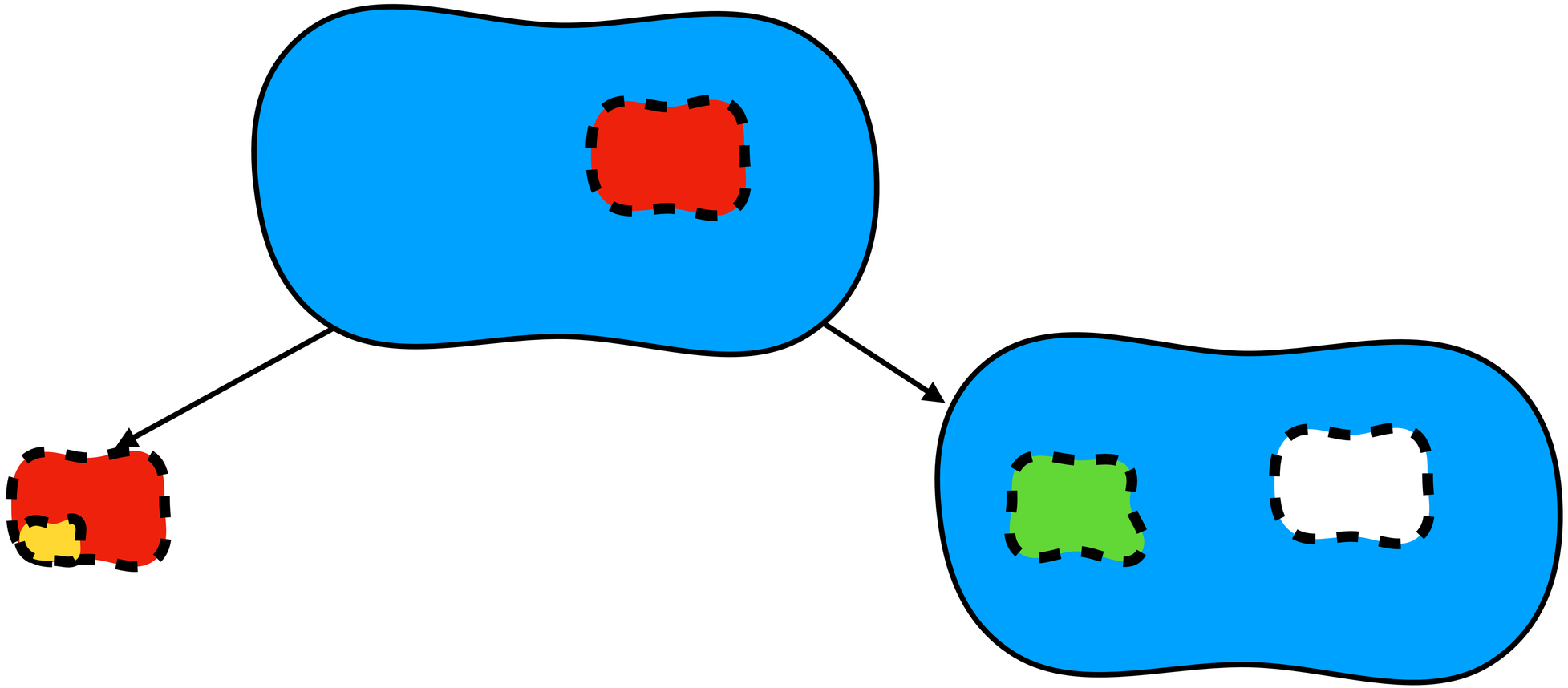
Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a decomposition hierarchy



Length-constrained hierarchies

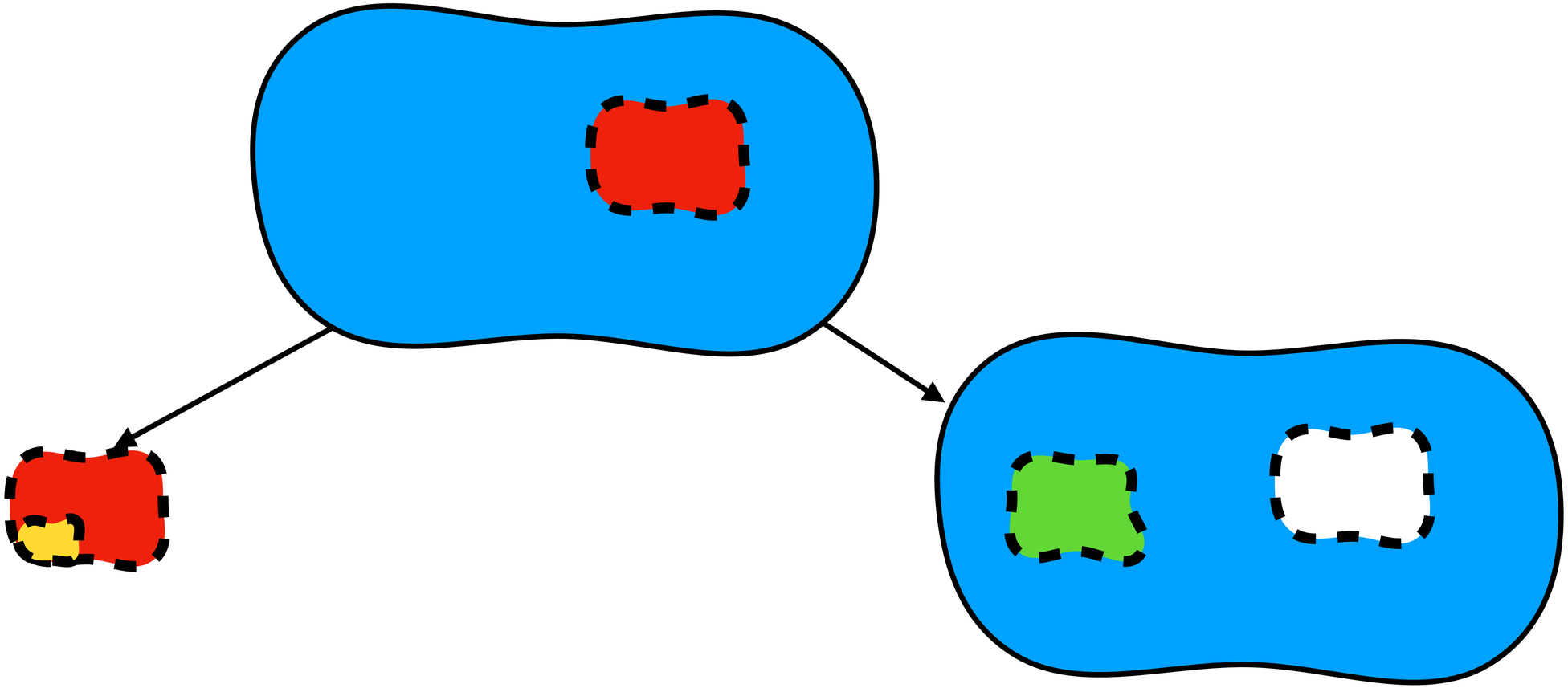
Balancing number of vertices is not enough



Length-constrained hierarchies

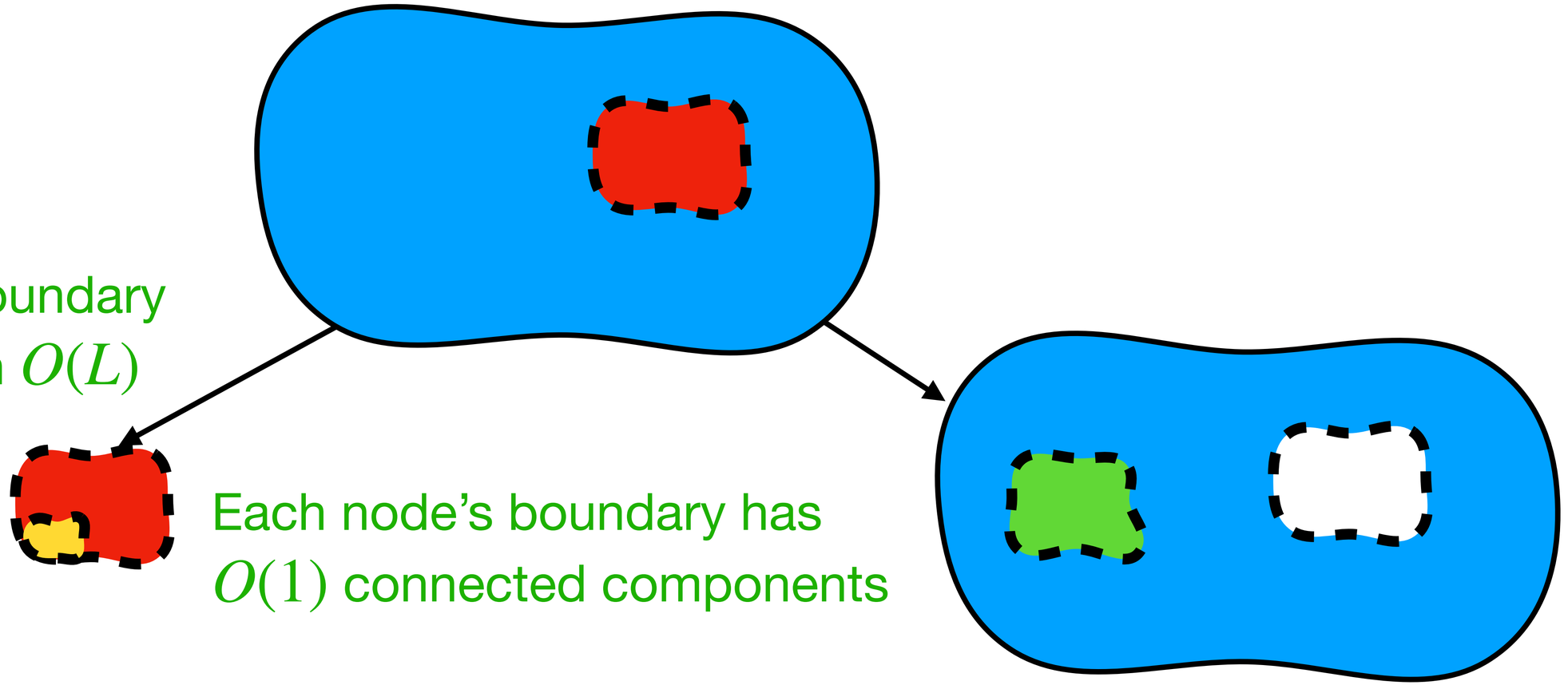
Balancing number of vertices is not enough

Need to balance lengths (and more)



Length-constrained hierarchies

Cleverly reweighting vertices gives



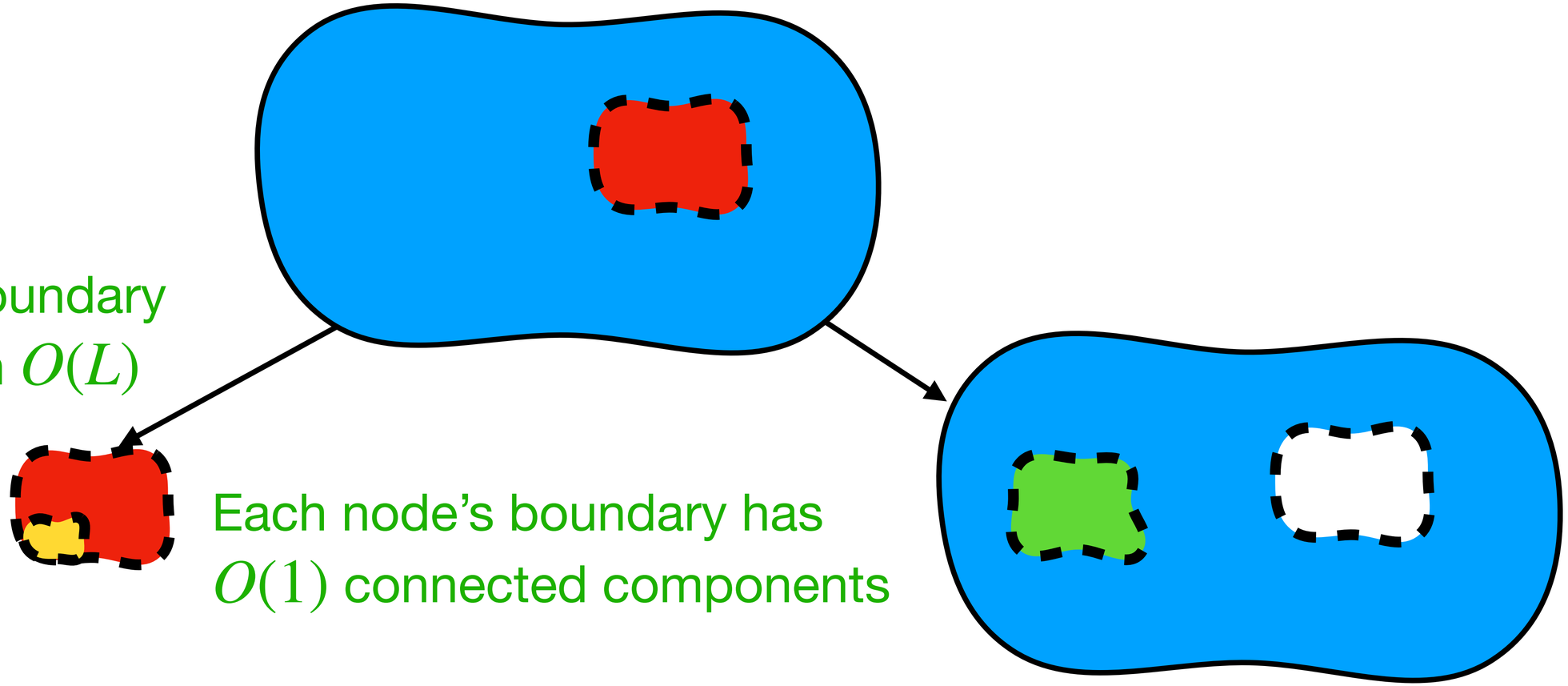
Each node's boundary
has total length $O(L)$

Each node's boundary has
$O(1)$ connected components

Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a length-constrained decomposition hierarchy

Cleverly reweighting vertices gives



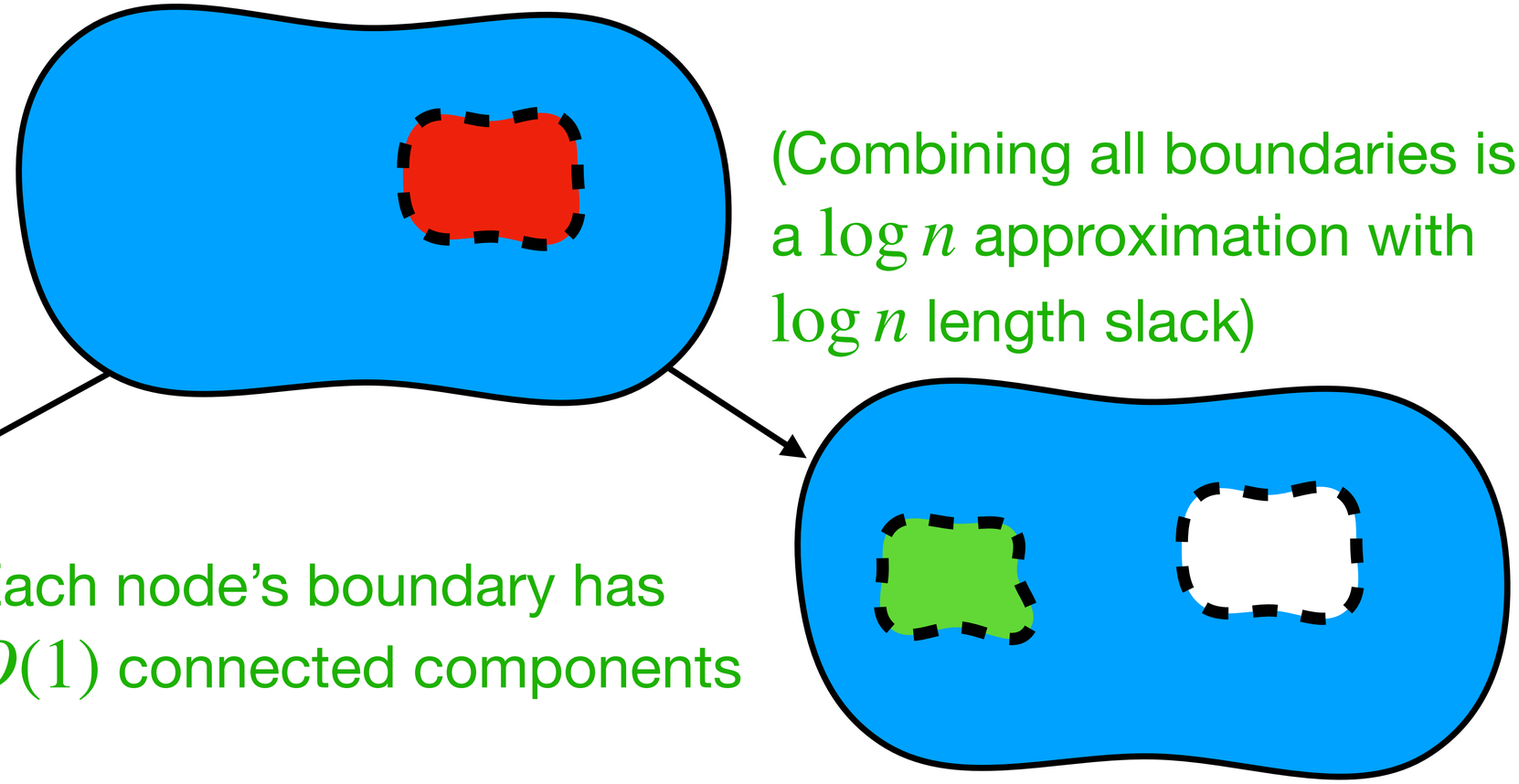
Each node's boundary has total length $O(L)$

Each node's boundary has $O(1)$ connected components

Length-constrained hierarchies

Taking separators for $\log n$ levels $\Longrightarrow$ a length-constrained decomposition hierarchy

Cleverly reweighting vertices gives



(Combining all boundaries is still a $\log n$ approximation with $\log n$ length slack)

Each node's boundary has total length $O(L)$

Each node's boundary has $O(1)$ connected components

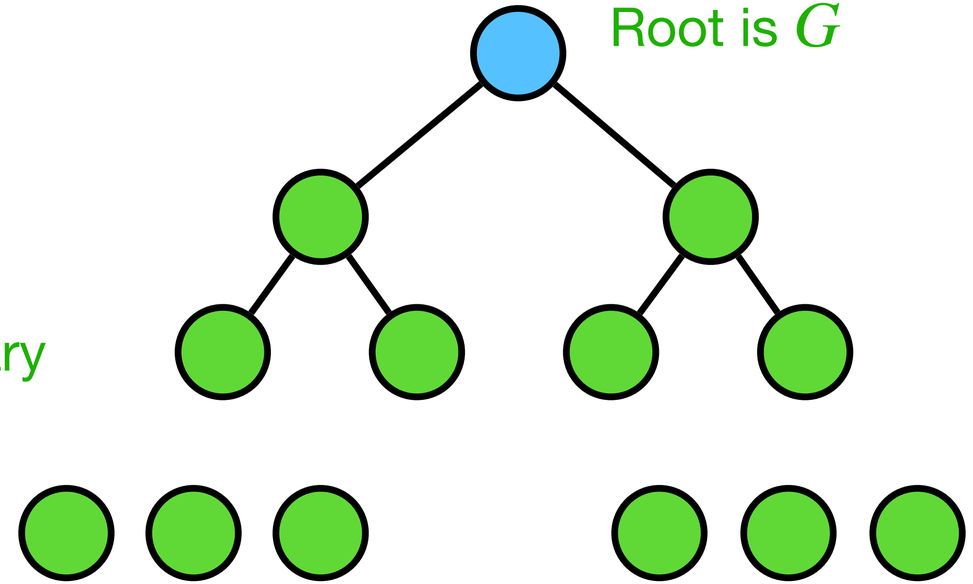
Sketch of the main algorithm

Sketch of the main algorithm

1. Buy a length-constrained hierarchy: boundaries have $\log n$ approximate weight with $\log n$ length slack

Root is G

Each node is a length-constrained region/boundary



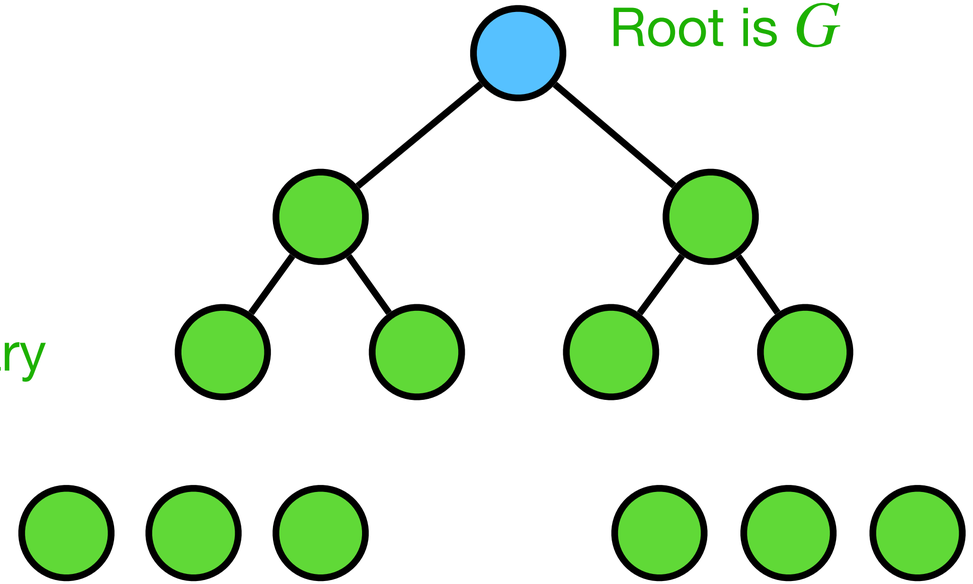
Leaves are regions with only boundary vertices

Sketch of the main algorithm

1. Buy a length-constrained hierarchy: boundaries have $\log n$ approximate weight with $\log n$ length slack

Root is G

Each node is a length-constrained region/boundary



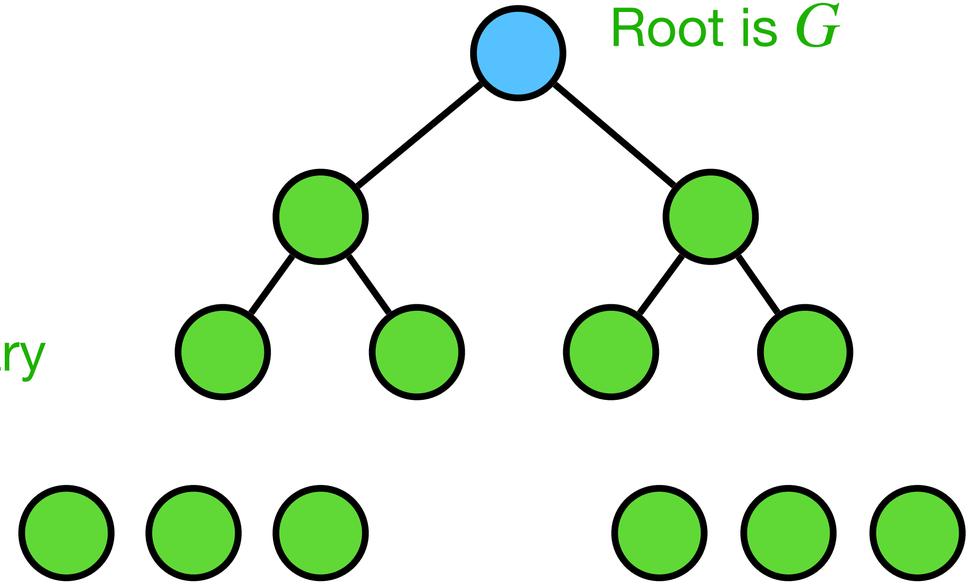
Leaves are regions with only boundary vertices

Sketch of the main algorithm

1. Buy a length-constrained hierarchy: boundaries have $\log n$ approximate weight with $\log n$ length slack

Root is G

Each node is a length-constrained region/boundary



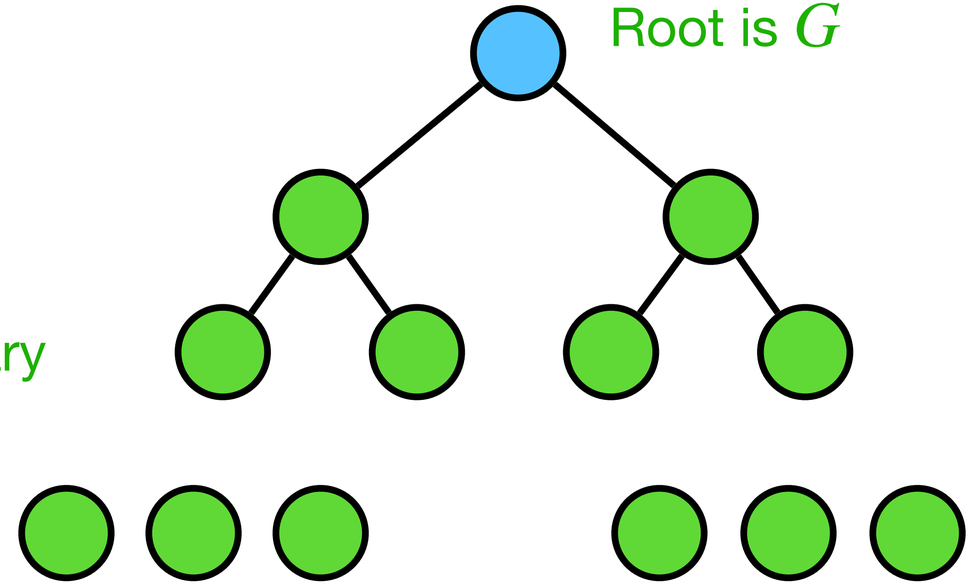
Leaves are regions with only boundary vertices

Sketch of the main algorithm

1. Buy a length-constrained hierarchy: boundaries have $\log n$ approximate weight with $\log n$ length slack

Root is G

Each node is a length-constrained region/boundary



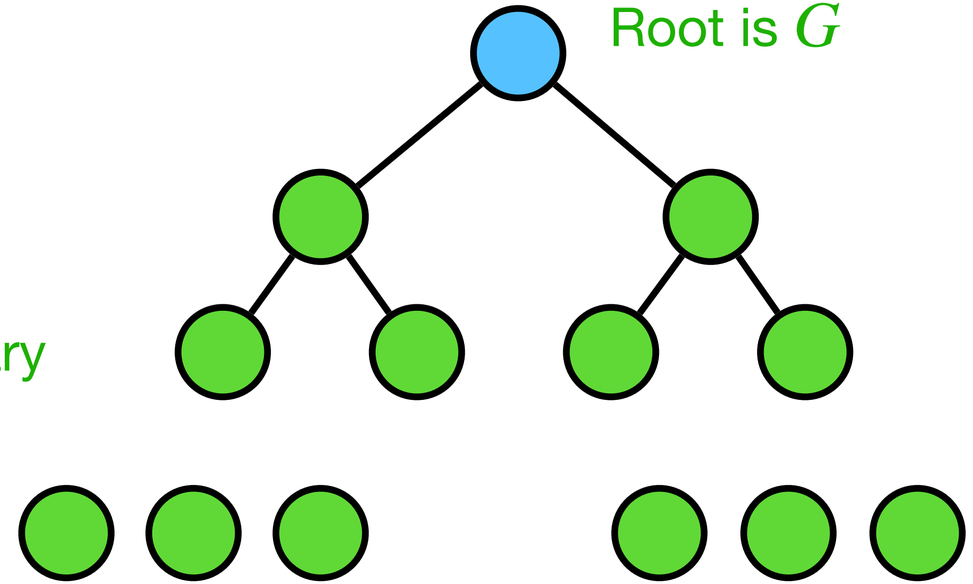
Leaves are regions with only boundary vertices

Sketch of the main algorithm

1. Buy a length-constrained hierarchy: boundaries have $\log n$ approximate weight with $\log n$ length slack

Root is G

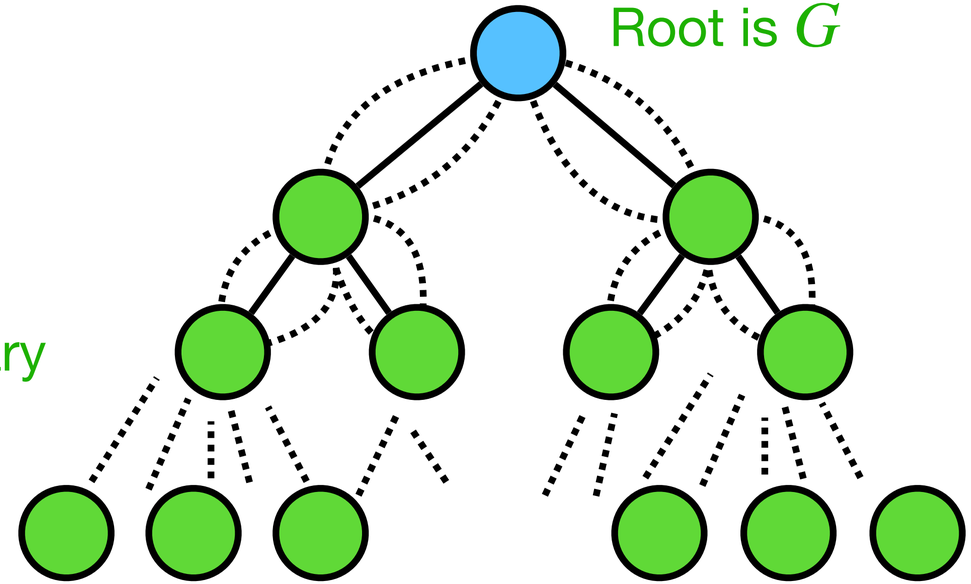
Each node is a length-constrained region/boundary



Leaves are regions with only boundary vertices

Sketch of the main algorithm

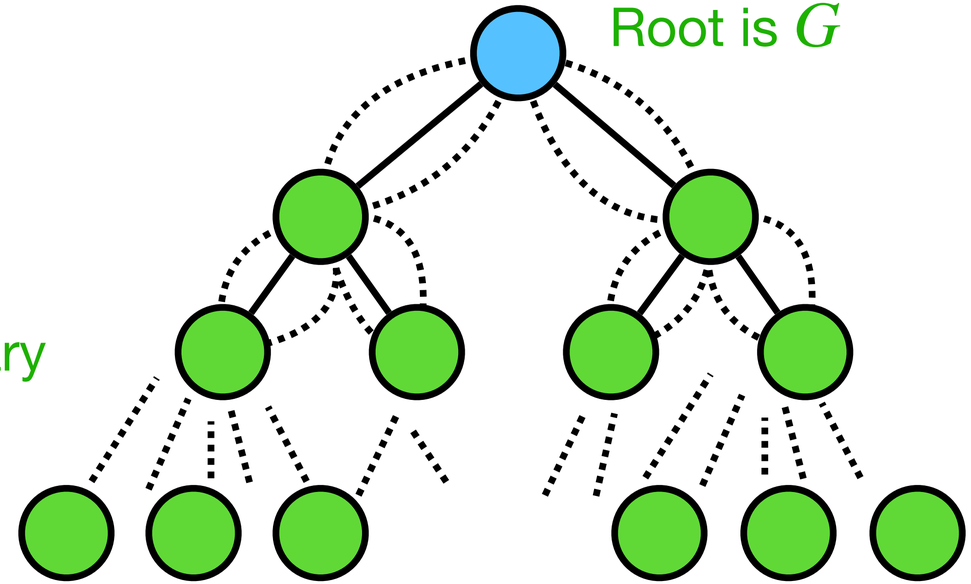
1. Buy a length-constrained hierarchy: boundaries have $\log n$ approximate weight with $\log n$ length slack



2. Smartly add edges between boundaries to reduce the length without increasing the weight by much

Sketch of the main algorithm

1. Buy a length-constrained hierarchy: boundaries have $\log n$ approximate weight with $\log n$ length slack



2. Smartly add edges between boundaries to reduce the length to $(1+\epsilon)L$ without increasing the weight beyond $(\log^{1+\epsilon} n)$OPT

Sketch of the main algorithm

Goal: smartly add edges to reduce the length to $(1+\epsilon)L$ without increasing the weight beyond $\left(\log^{1+\epsilon} n\right)$ OPT

Sketch of the main algorithm

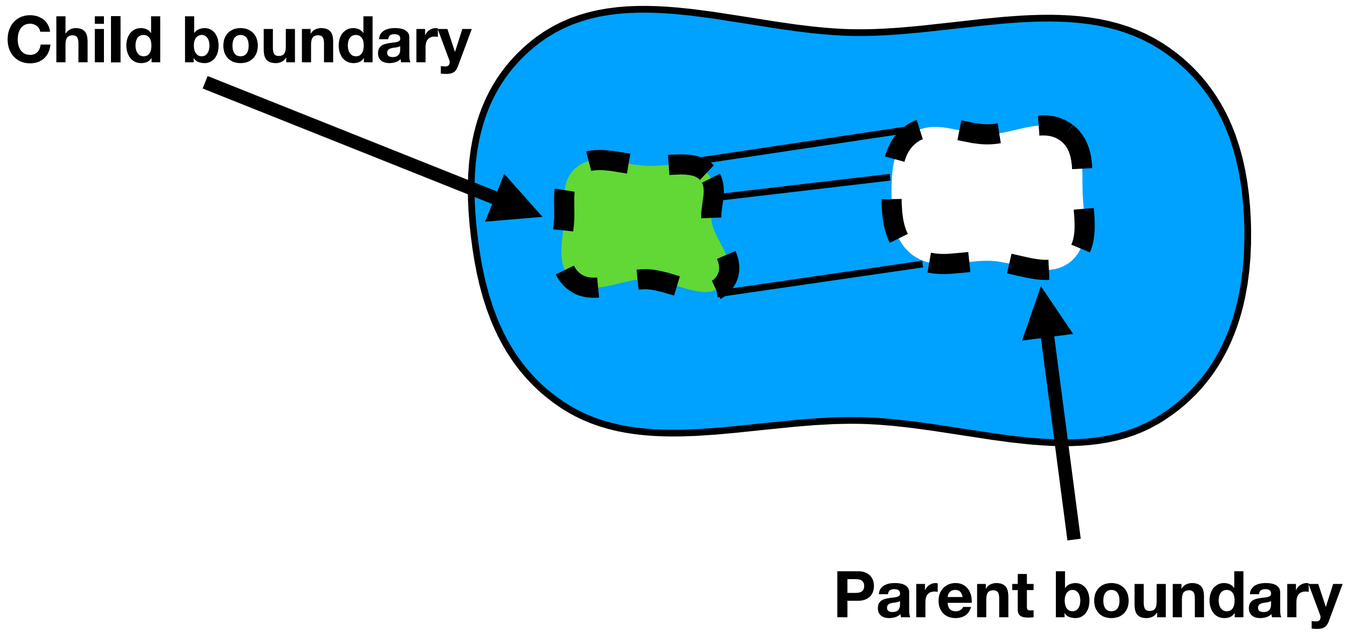
Goal: smartly add edges to reduce the length to $(1+\epsilon)L$ without increasing the weight beyond $\left(\log^{1+\epsilon} n\right)$ OPT

Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children

Sketch of the main algorithm

Goal: smartly add edges to reduce the length to $(1+\epsilon)L$ without increasing the weight beyond $\left(\log^{1+\epsilon} n\right)$ OPT

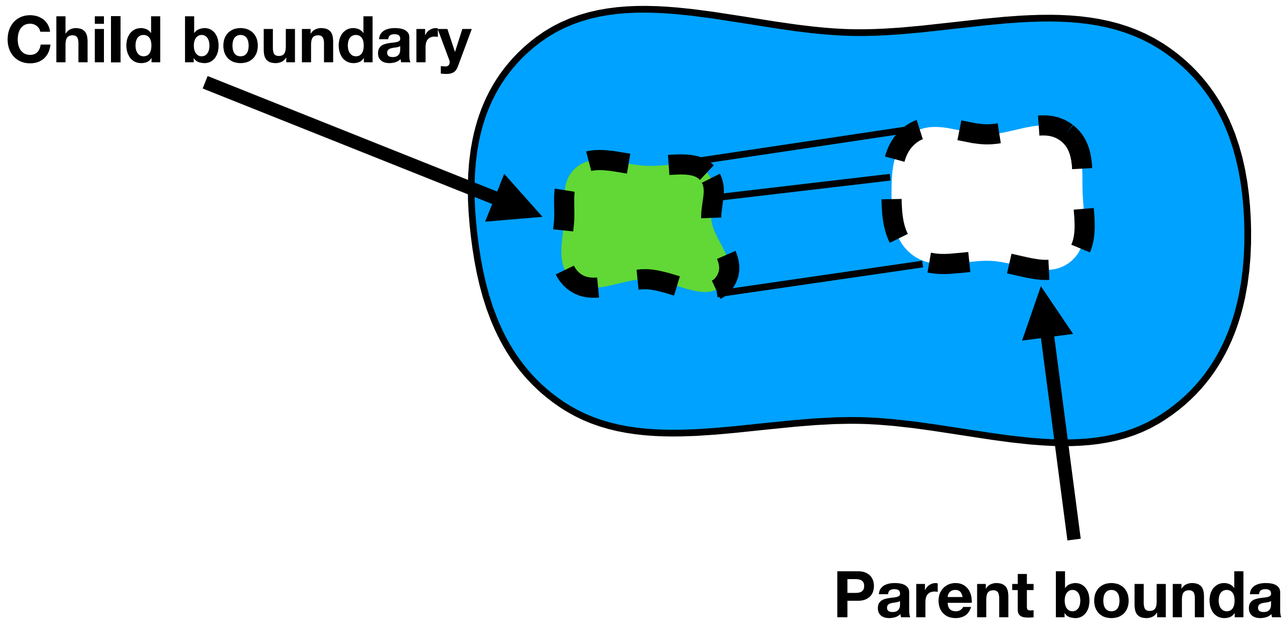
Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children



Sketch of the main algorithm

Goal: smartly add edges to reduce the length to $(1+\epsilon)L$ without increasing the weight beyond $\left(\log^{1+\epsilon} n\right)$ OPT

Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children

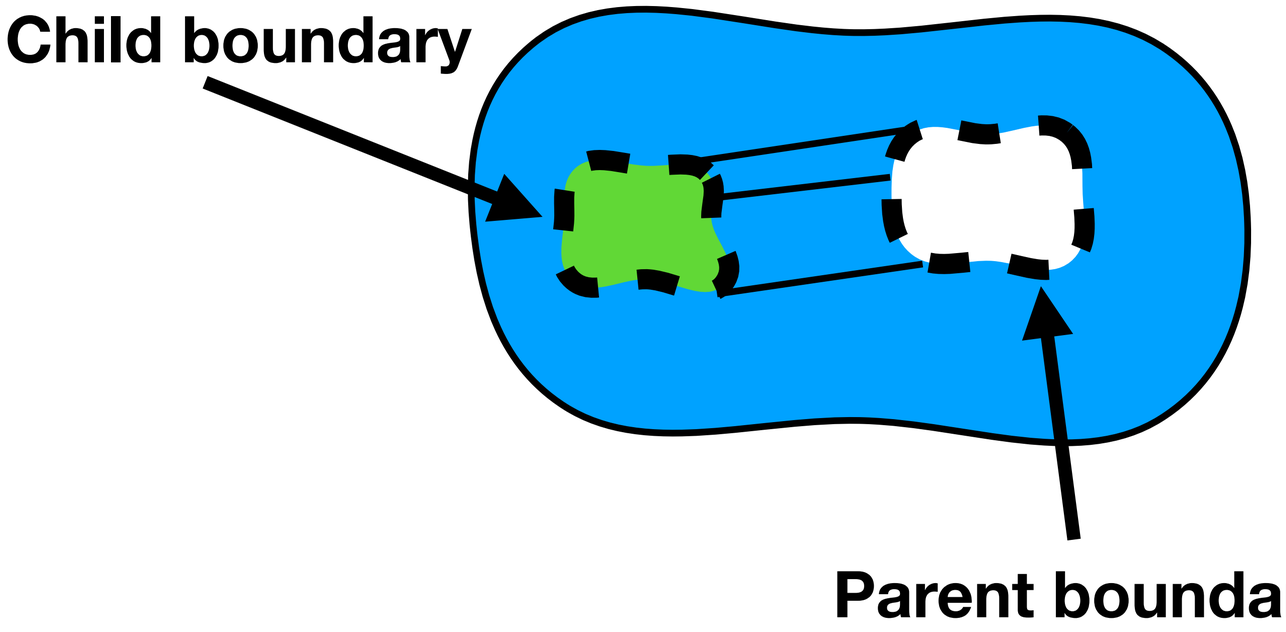


Terminals?

Sketch of the main algorithm

Goal: smartly add edges to reduce the length to $(1+\epsilon)L$ without increasing the weight beyond $\left(\log^{1+\epsilon} n\right)$ OPT

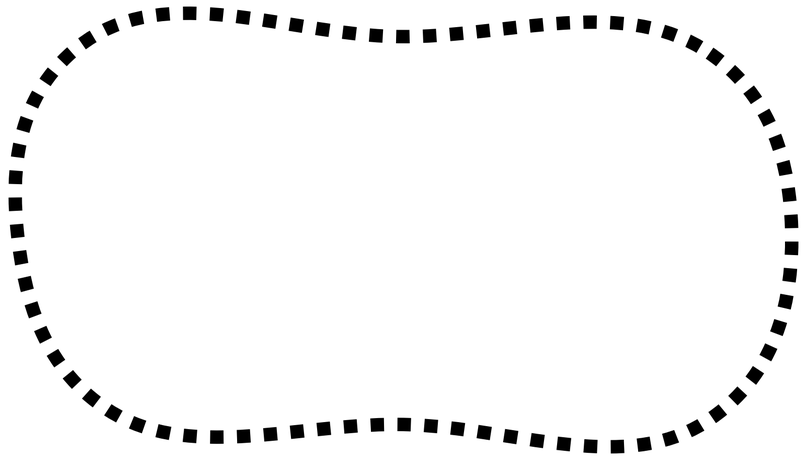
Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children



Break boundaries into ***pieces***: low diameter components that contain a “portal” shortcutting to the root.

Sketch of the main algorithm

Some node's boundary



Break boundaries into ***pieces***: low diameter components that contain a "portal" shortcutting to the root.

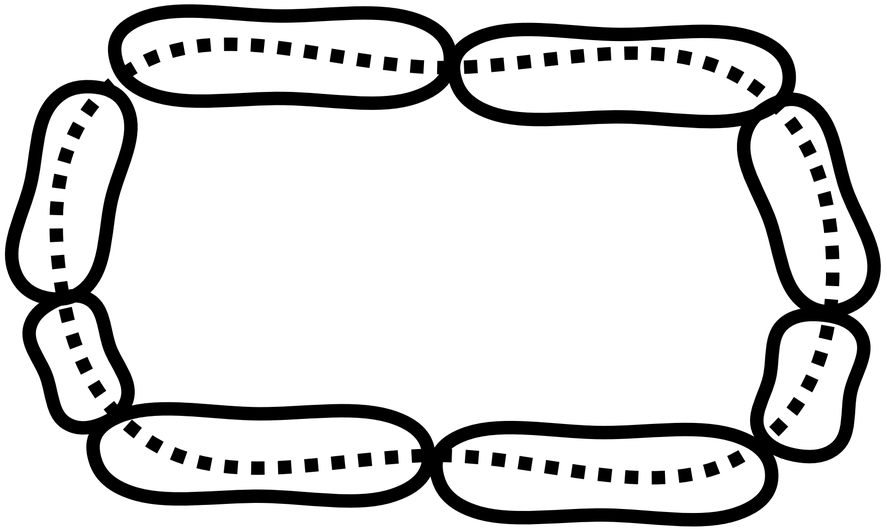
Sketch of the main algorithm

Crucial length-constrained division hierarchy property: boundaries have total length $O(L)$ and $O(1)$ components

$\Longrightarrow$

Boundary is partitionable into few pieces where each piece has low diameter

Partitioning boundary into pieces

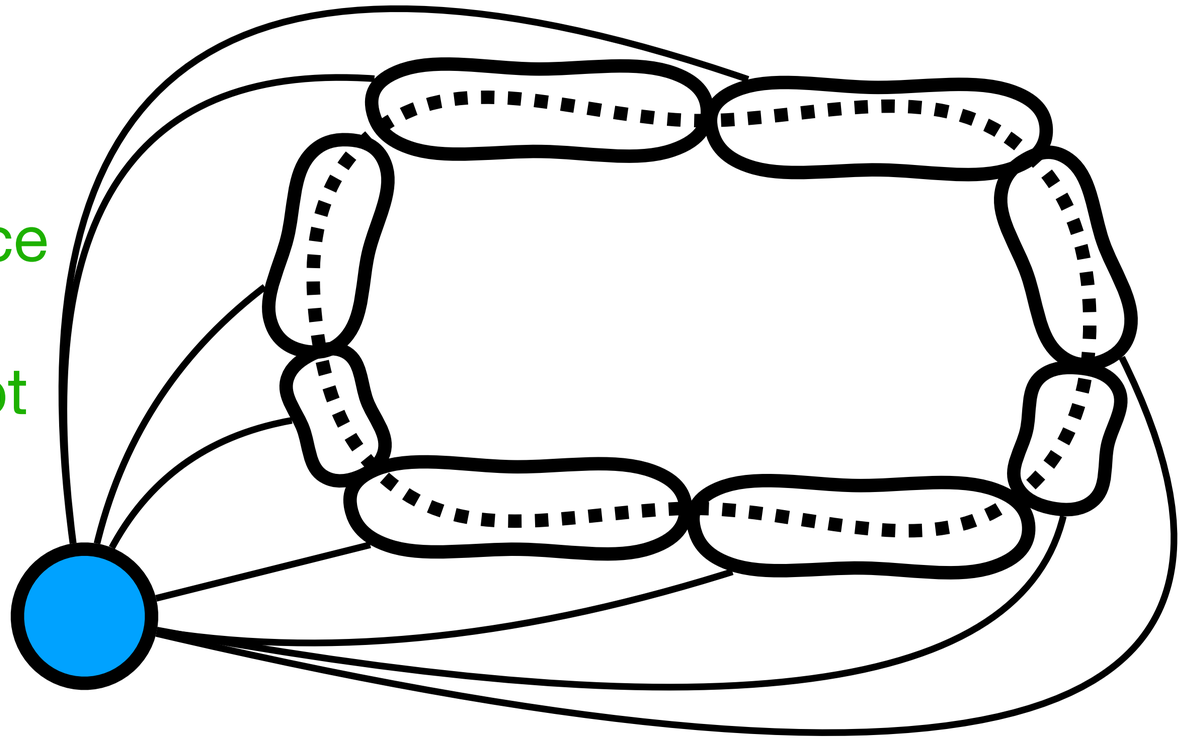


Break boundaries into ***pieces***: low diameter components that contain a “portal” shortcutting to the root.

Sketch of the main algorithm

Every vertex has a short walk within the piece before reaching the piece's shortcut

Each piece shortcuts to the root

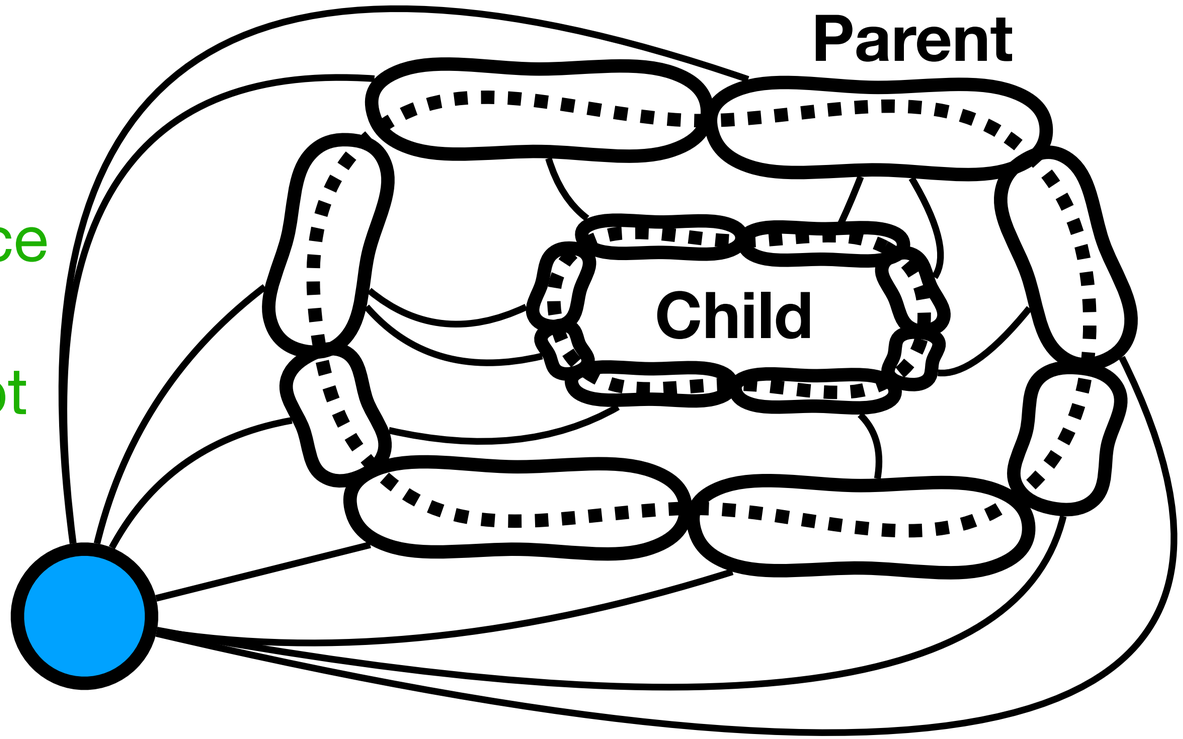


Break boundaries into ***pieces***: low diameter components that contain a "portal" shortcutting to the root.

Sketch of the main algorithm

Every vertex has a short walk within the piece before reaching the piece's shortcut

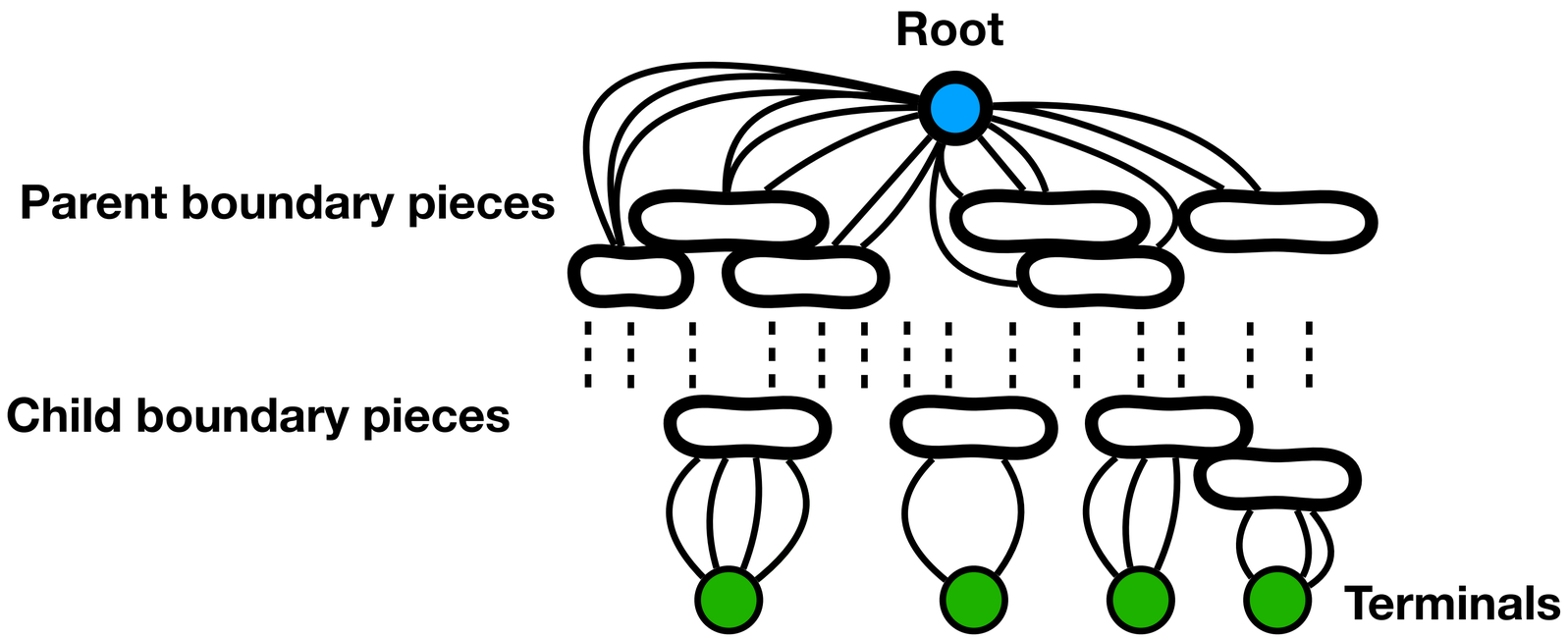
Each piece shortcuts to the root



Break boundaries into ***pieces***: low diameter components that contain a "portal" shortcutting to the root (via the parent).

Sketch of the main algorithm

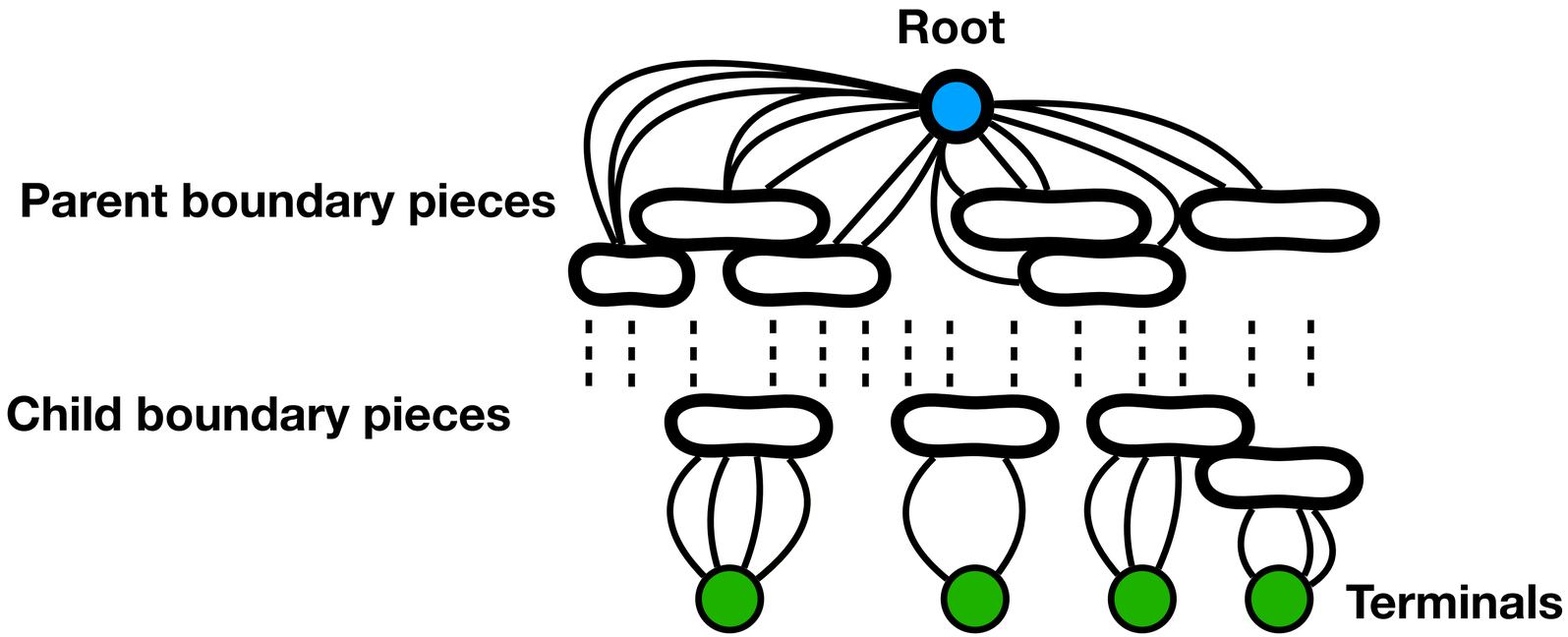
Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children



Sketch of the main algorithm

Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children

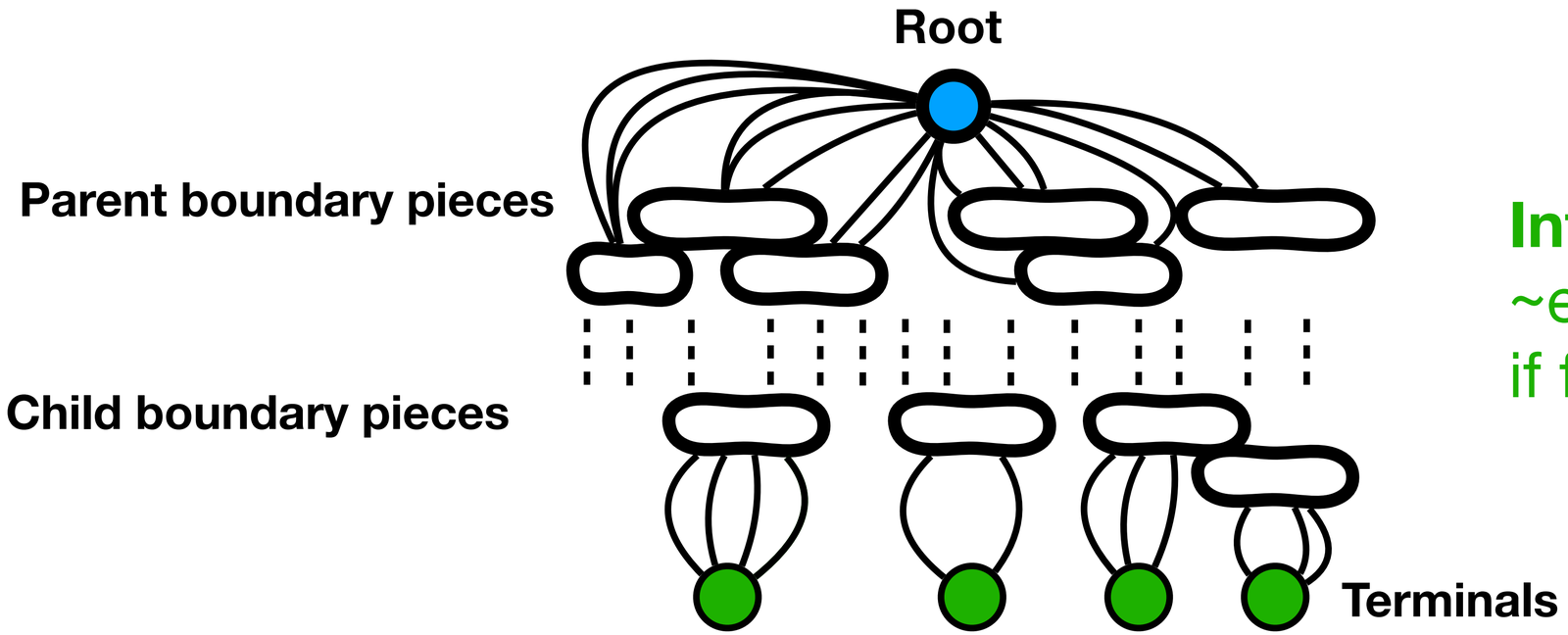
Use the single criteria t^{ϵ} - approximation algorithm from CCCDGGL 1998, with child pieces as the terminals



Sketch of the main algorithm

Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children

Use the single criteria t^{ϵ} - approximation algorithm from CCCDGGL 1998, with child pieces as the terminals

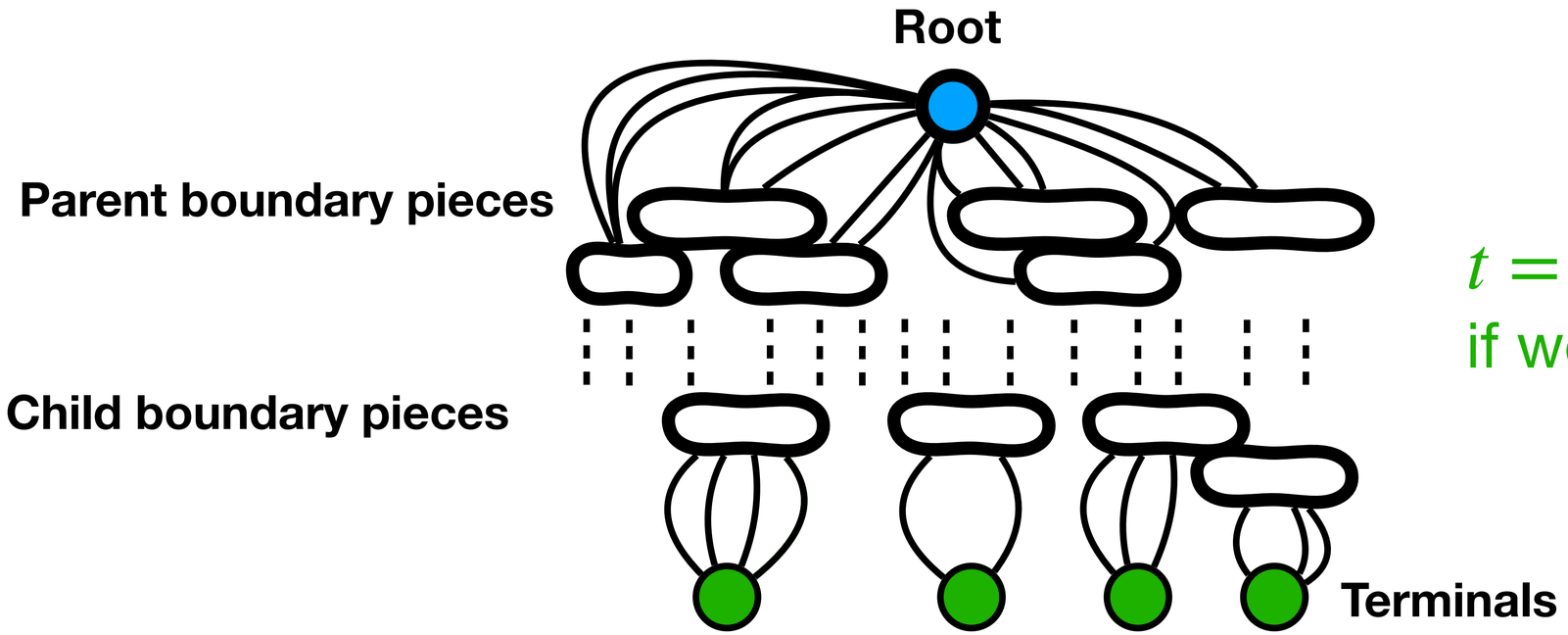


Intuition: length is ~exact, weight is low if few terminals

Sketch of the main algorithm

Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children

Use the single criteria t^{ϵ} - approximation algorithm from CCCDGGL 1998, with child pieces as the terminals

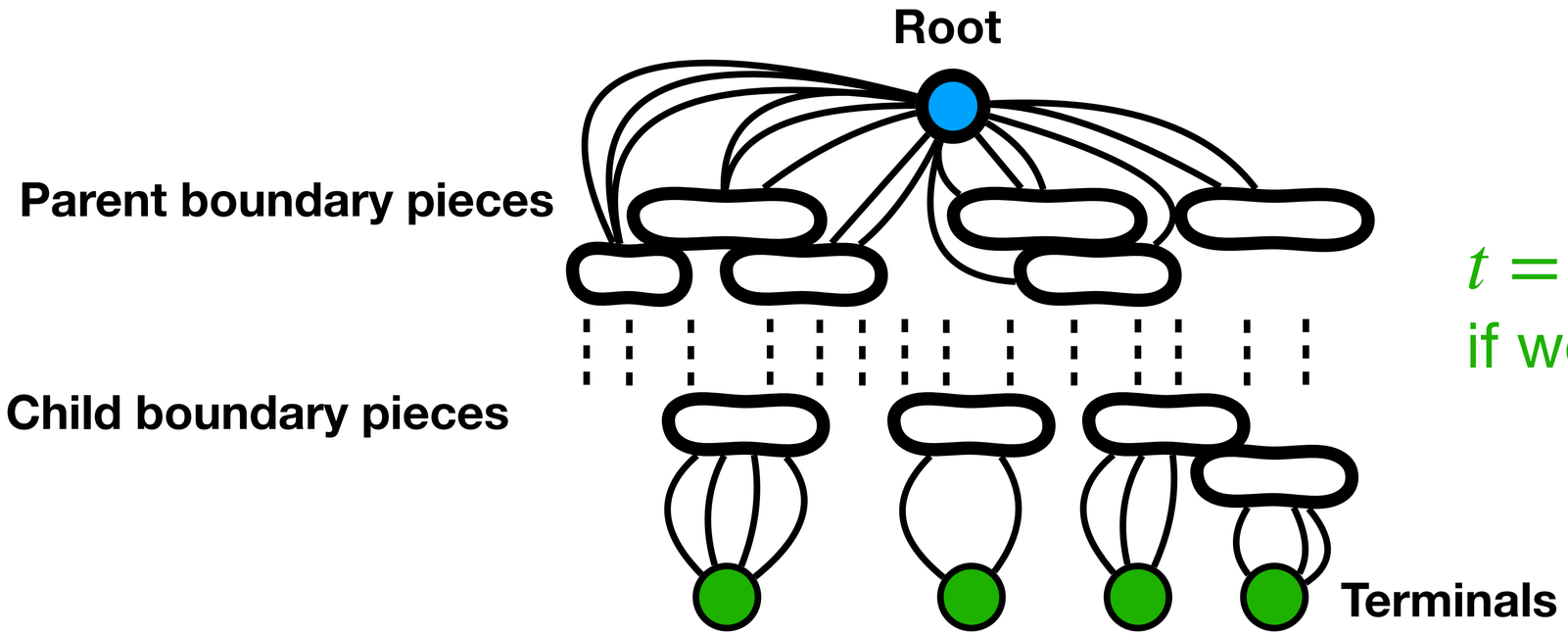


$t = O(\log n)$ terminals if we are more careful

Sketch of the main algorithm

Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children

$O(t^\epsilon) = O(\log^\epsilon n)$ **weight approximation at each node of the hierarchy**

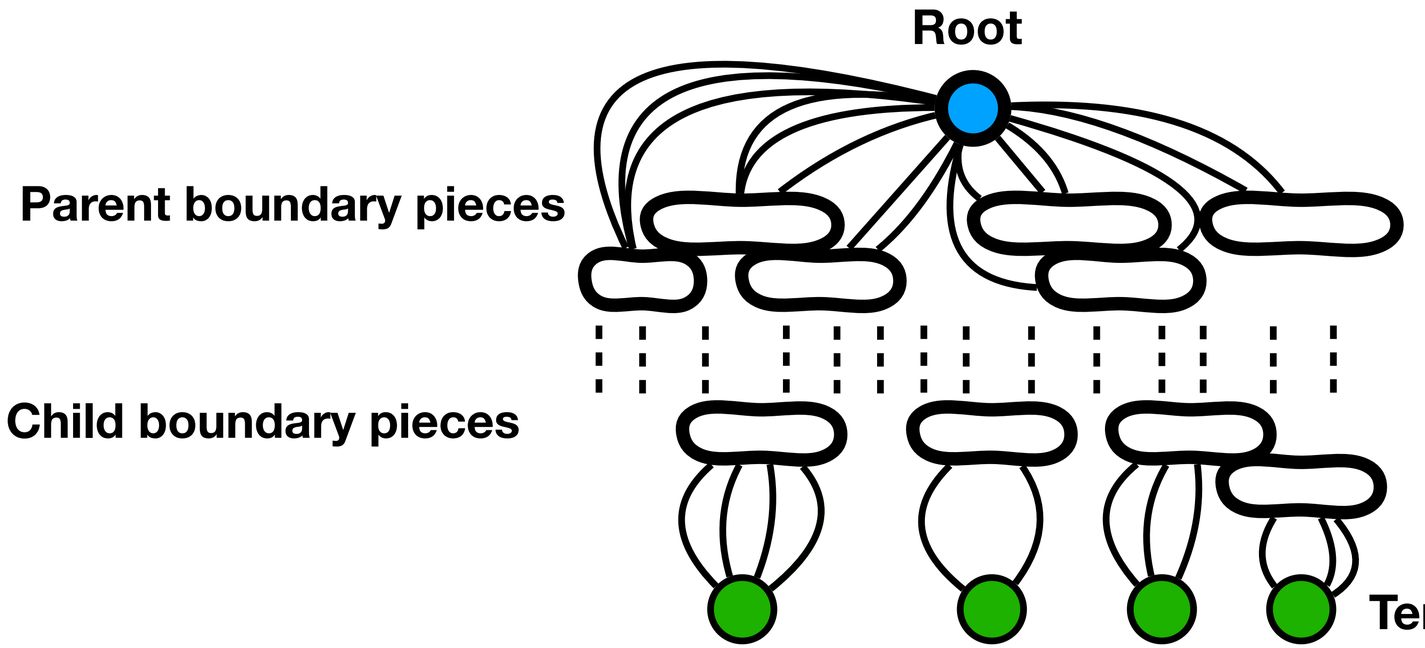


$t = O(\log n)$ terminals
if we are more careful

Sketch of the main algorithm

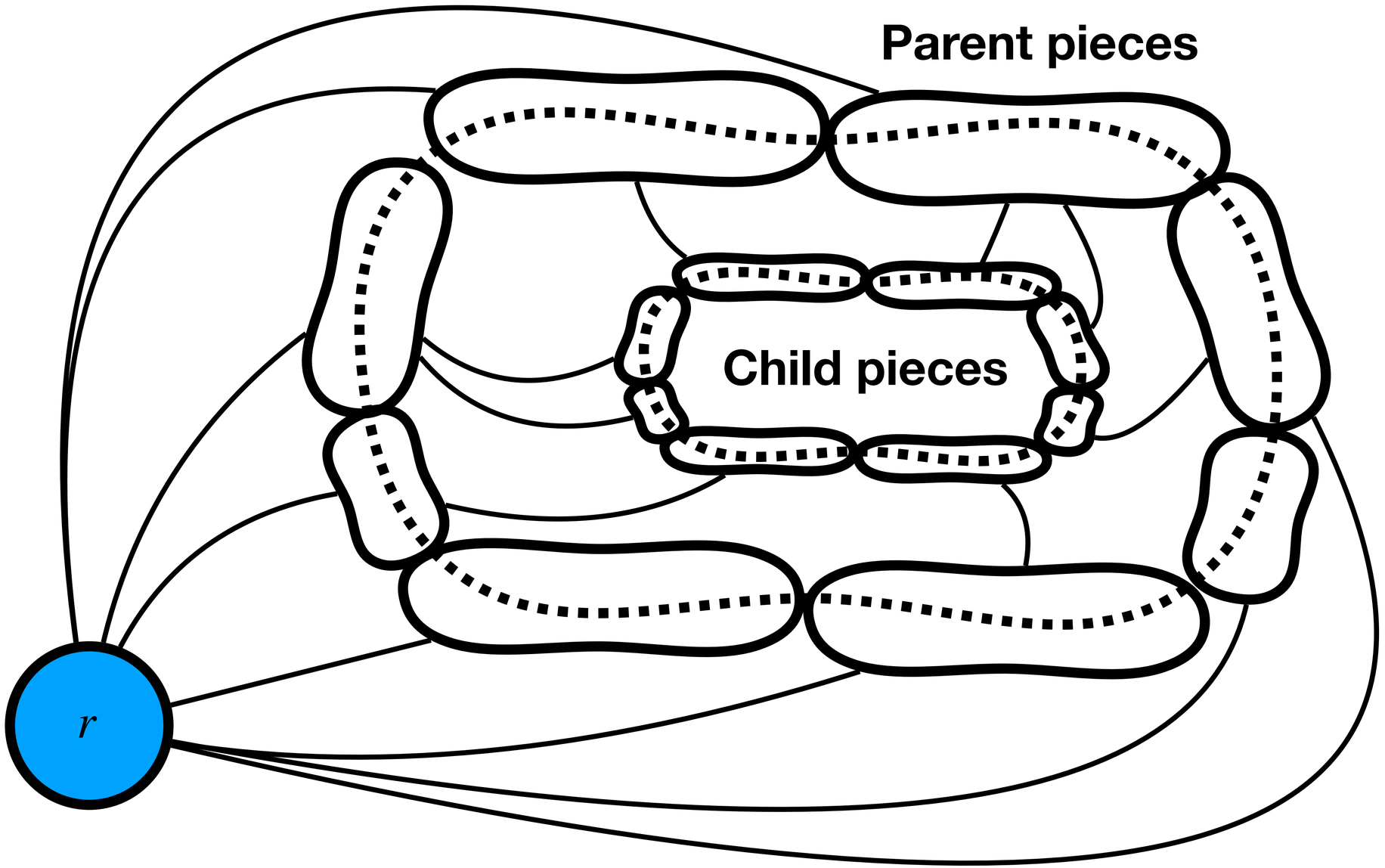
Idea: at each node of the division hierarchy, solve a length-constrained Steiner instance connecting its boundary to the boundaries of its children

$O(\log^{1+\epsilon} n)$ **weight approximation overall**

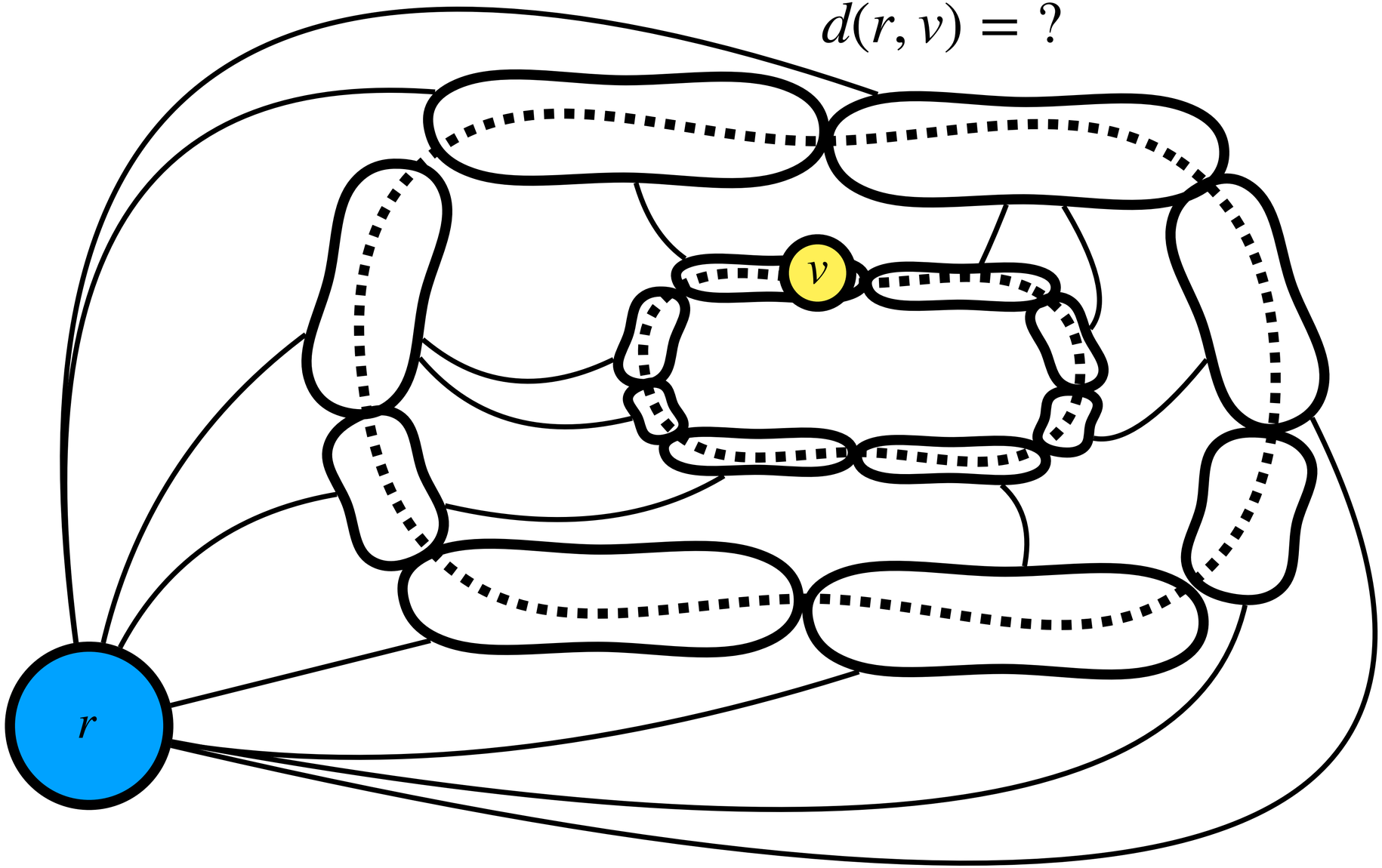


Height of the hierarchy is $\log n$

Sketch of the main algorithm

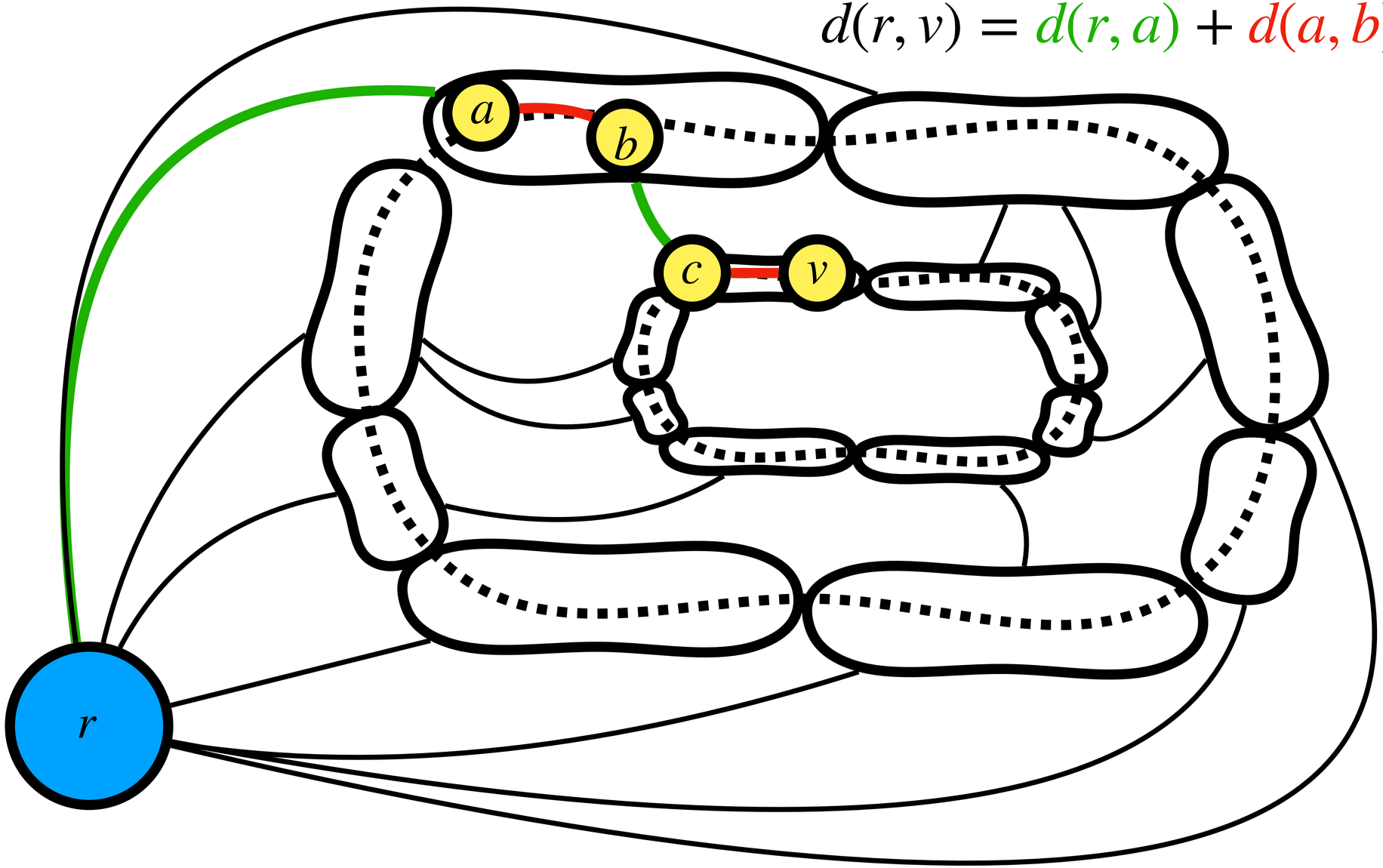


Sketch of the main algorithm



Sketch of the main algorithm

$$d(r,v) = d(r,a) + d(a,b) + d(b,c) + d(c,v)$$

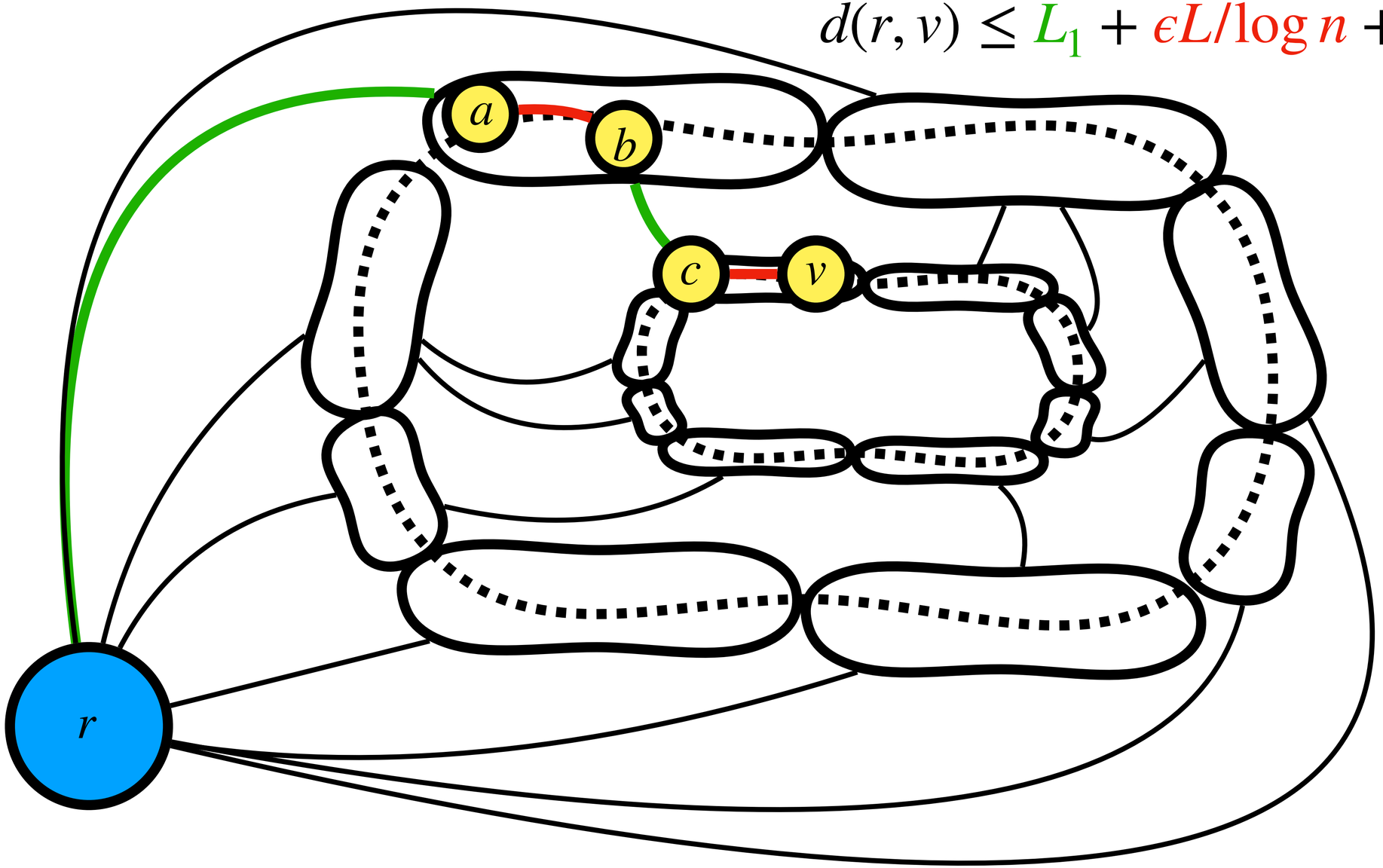


LC Steiner constraint

Piece diameter

Sketch of the main algorithm

$$d(r,v) \le L_1 + \epsilon L/\log n + L_2 + \epsilon L/\log n$$

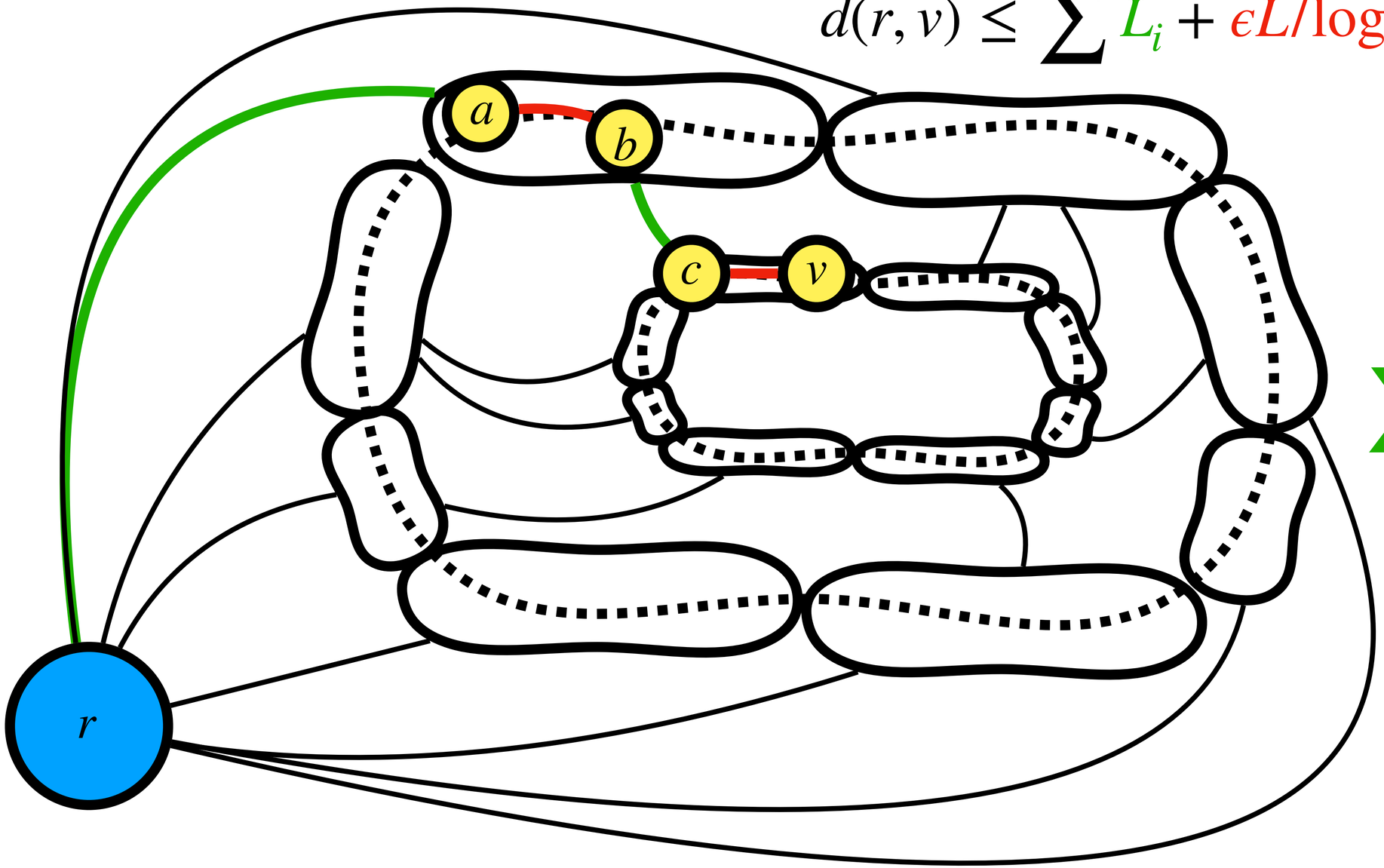


LC Steiner constraint

Piece diameter

Sketch of the main algorithm

$$d(r,v) \le \sum L_i + \epsilon L/\log n \le L + \epsilon L$$



$\sum L_i \le L$ by guessing

$\le \log n$ pieces to walk

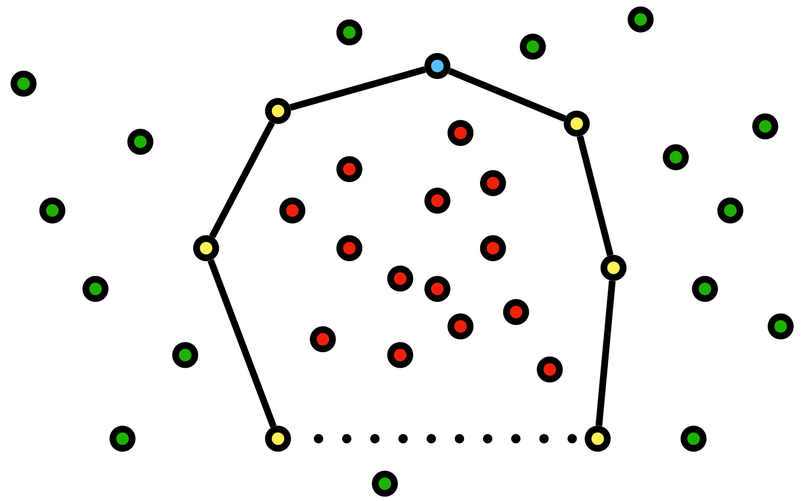
Length-constrained MST is easier to approximate in planar graphs than in general graphs

Length-constrained MST is easier to approximate in planar graphs than in general graphs

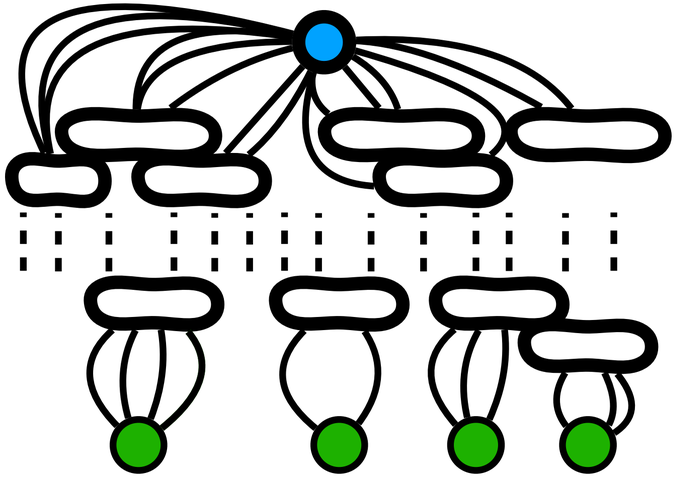
Planar graphs: can get $\log^{1+\epsilon} n$ approximation with $1 + \epsilon$ length slack

Length-constrained MST is easier to approximate in planar graphs than in general graphs

Planar graphs: can get $\log^{1+\epsilon} n$ approximation with $1+\epsilon$ length slack

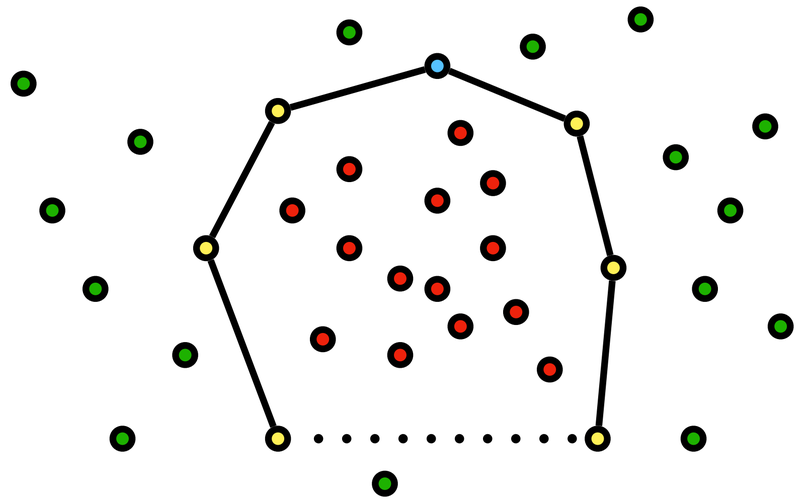


+ *length constraints* +

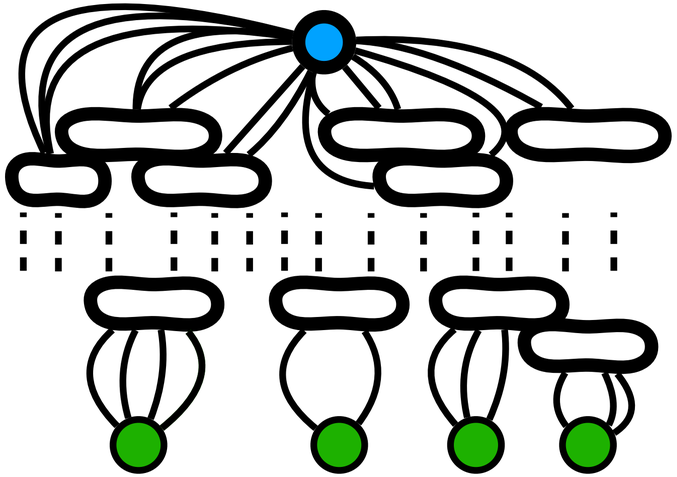


Length-constrained MST is easier to approximate in planar graphs than in general graphs

Planar graphs: can get $\log^{1+\epsilon} n$ approximation with $1+\epsilon$ length slack

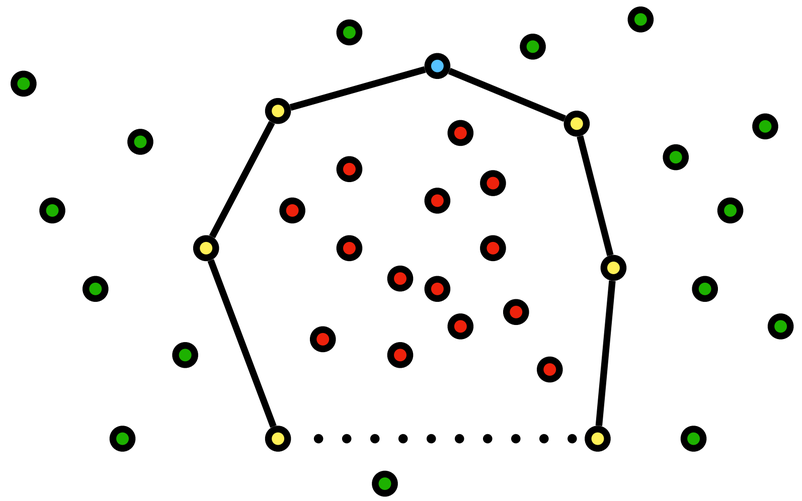


+ *length constraints* +

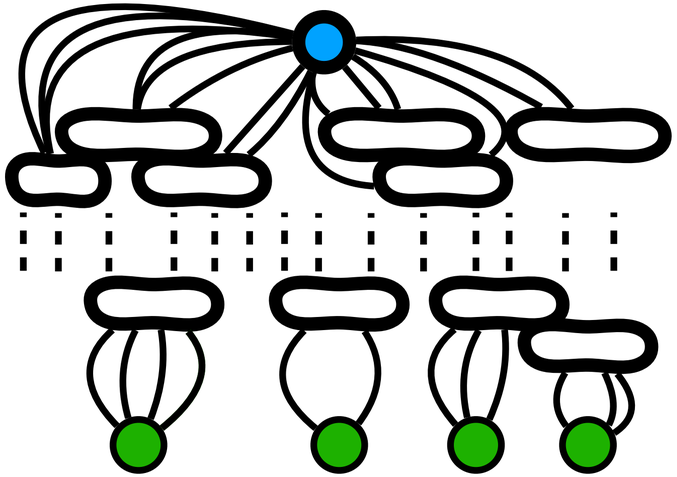


Length-constrained MST is easier to approximate in planar graphs than in general graphs

Planar graphs: can get $\log^{1+\epsilon} n$ approximation with $1+\epsilon$ length slack

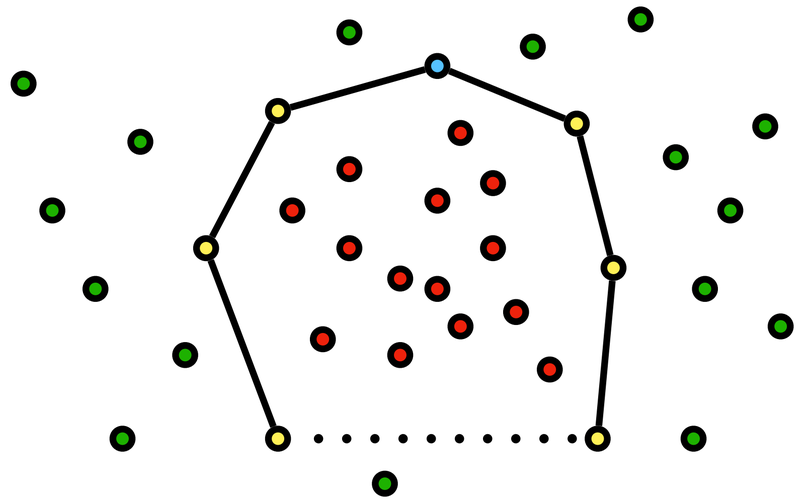


+ *length constraints* +

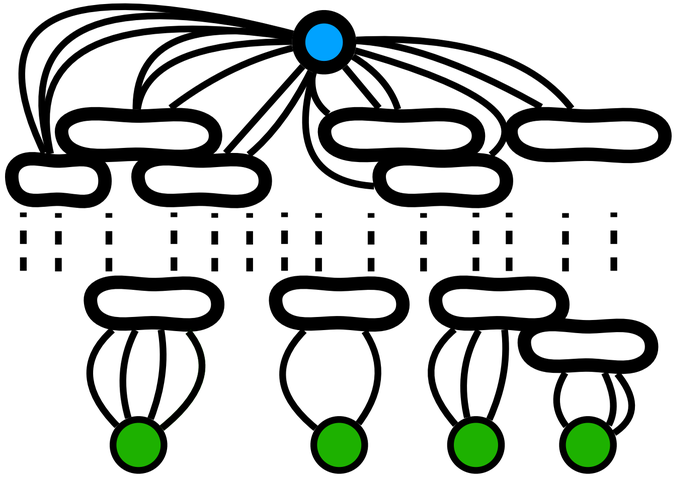


Length-constrained MST is easier to approximate in planar graphs than in general graphs

Planar graphs: can get $\log^{1+\epsilon} n$ approximation with $1+\epsilon$ length slack

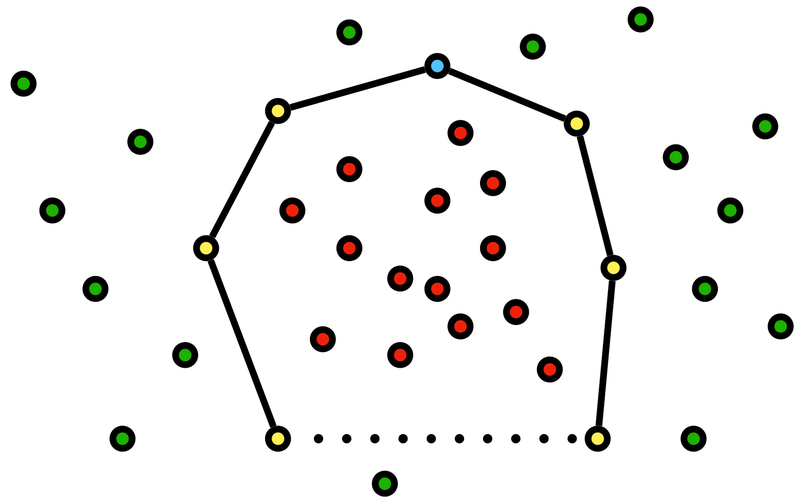


+ *length constraints* +

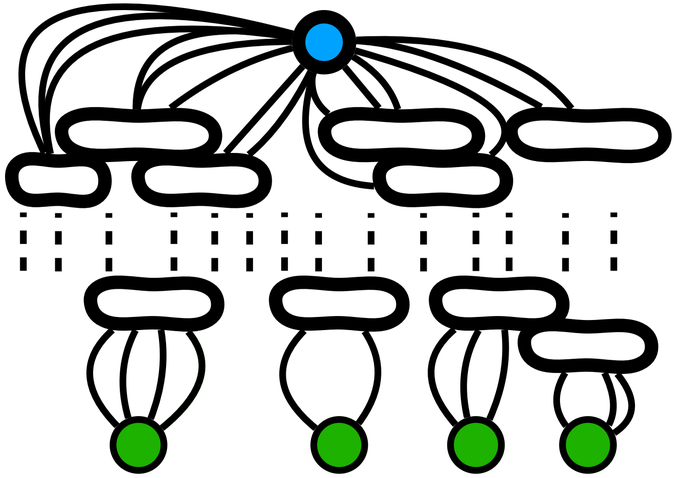


Length-constrained MST is easier to approximate in planar graphs than in general graphs

Planar graphs: can get $\log^{1+\epsilon} n$ approximation with $1+\epsilon$ length slack



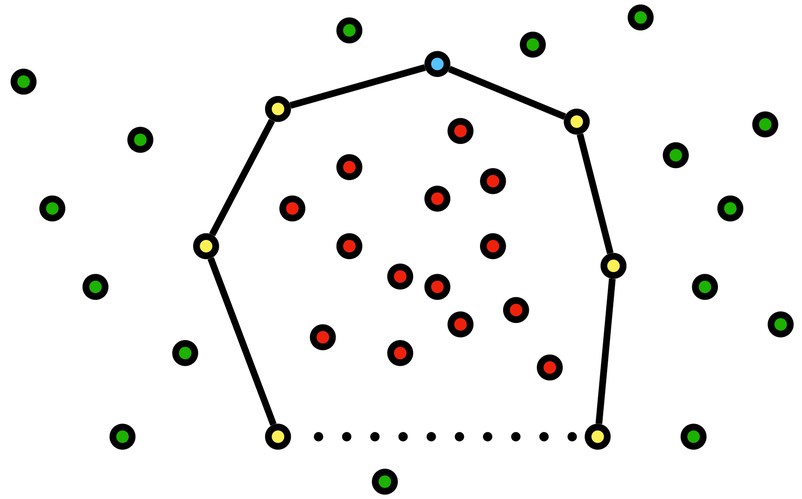
+ *length constraints* +



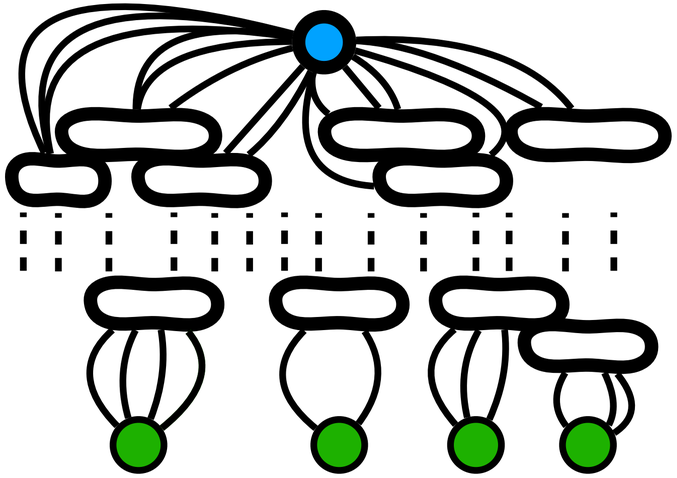
General graphs: cannot get $\log^{2-\epsilon} n$ approximation with < 2 length slack

Length-constrained MST is easier to approximate in planar graphs than in general graphs

Planar graphs: can get $\log^{1+\epsilon} n$ approximation with $1+\epsilon$ length slack



$+$ *length constraints* $+$



General graphs: cannot get $\log^{2-\epsilon} n$ approximation with < 2 length slack

Thank you