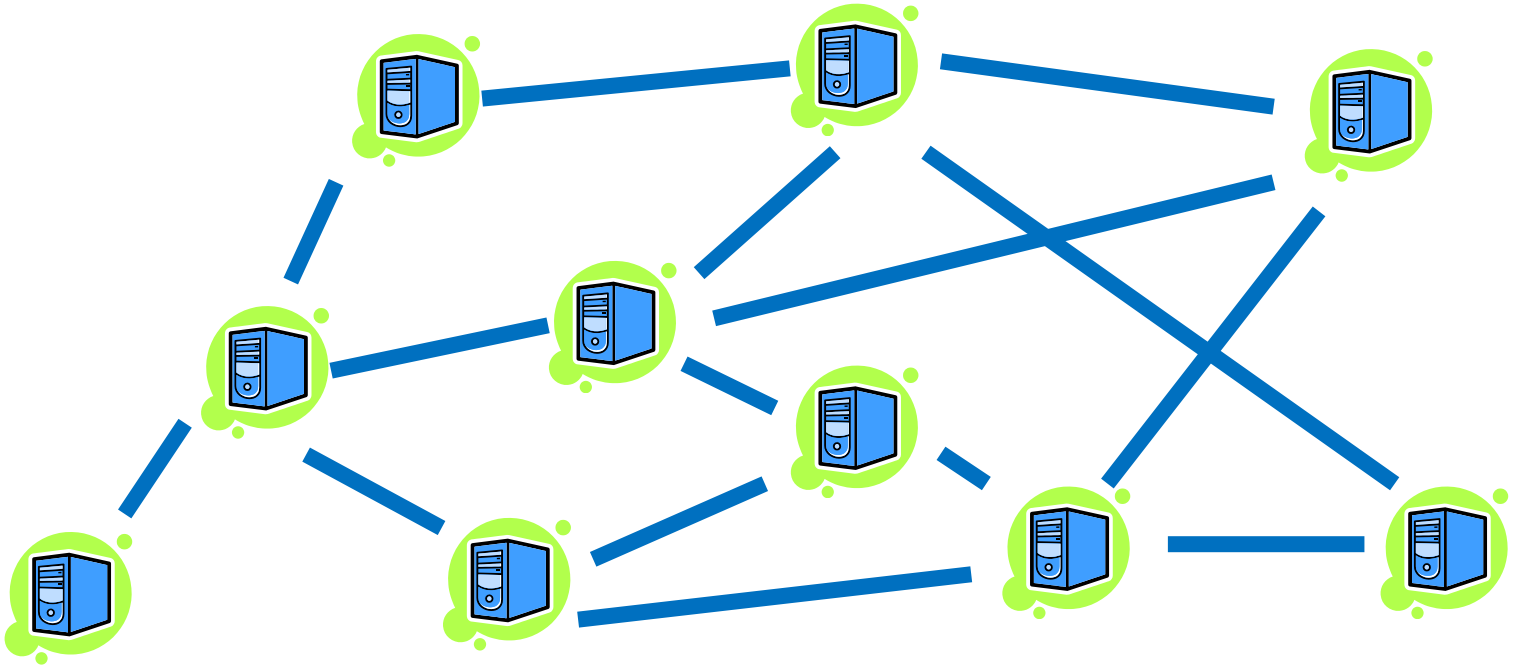
ネットワーク上の分散グラフアルゴリズムと最適化

泉　泰介（名古屋工業大学）

RIMS組み合わせ最適化セミナー(2017/07/27)

分散システム

- ネットワーク=グラフ
  - 頂点=計算機(ノード)
  - 辺=通信リンク

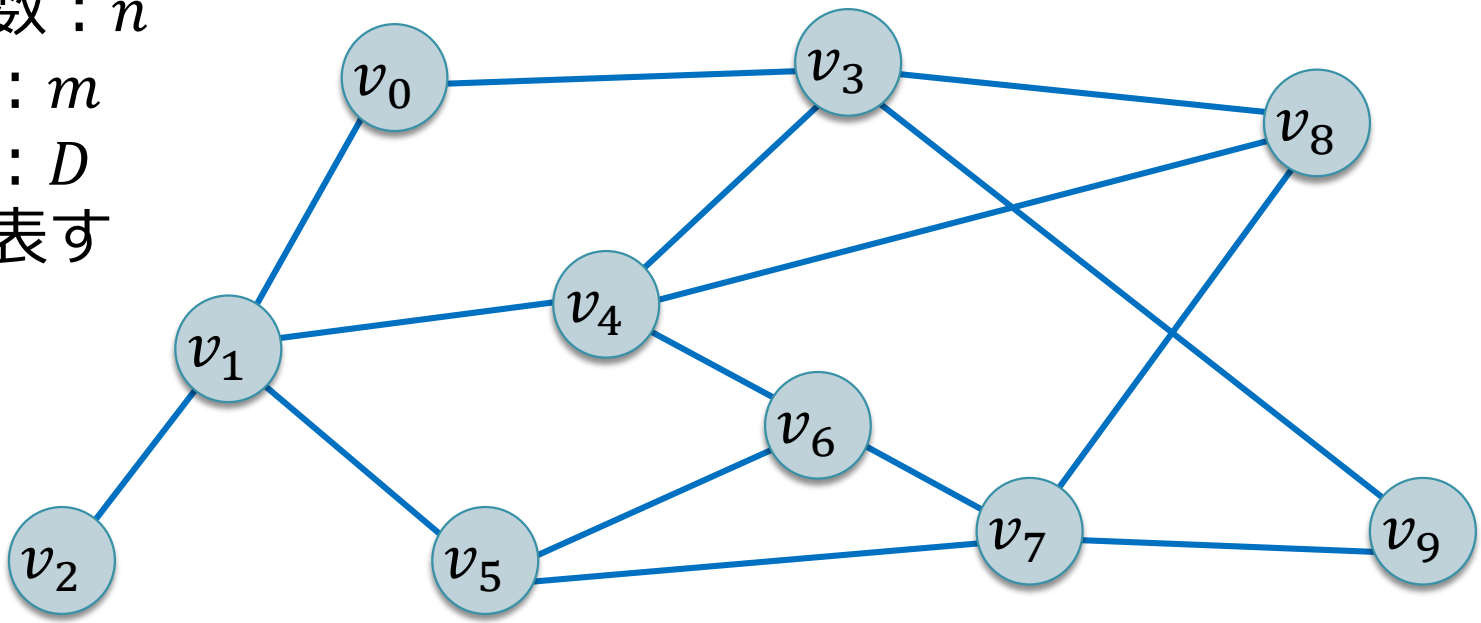


分散システム

- ネットワーク=グラフ
  - 頂点=計算機
  - 辺=通信リンク

ネットワーク自身を入力と見なして
グラフ上の問題を解く
→ 分散グラフアルゴリズム

頂点数：n
辺数：m
直径：D
で表す



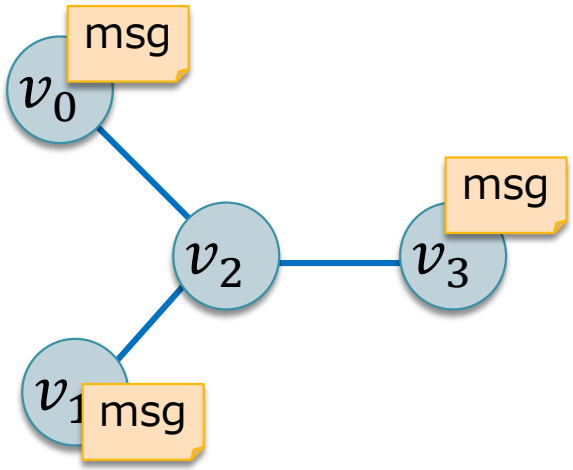
CONGESTモデル

- 計算機はラウンドに従って同期して動作
  - 隣接頂点へのメッセージ送信
  - 隣接頂点からのメッセージ受信
  - 内部計算(RAMで多項式時間)

  } で１ラウンド

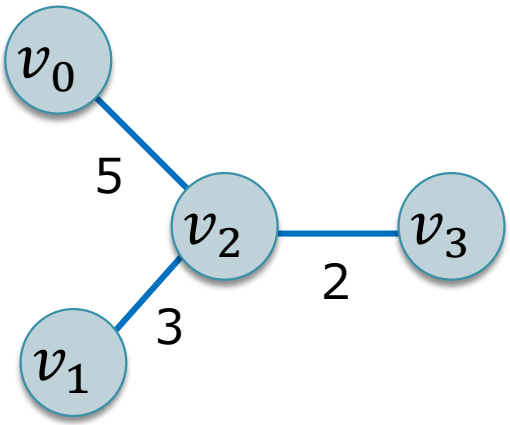
- 各リンクは単位ラウンドあたり高々$O(\log n)$ビットを（双方向に)伝送可能

逆に，帯域がunboundedなモデルをLOCALモデルと呼ぶ(今回は考えない)



CONGESTモデル

- 本発表では全体を通して重み付きグラフを考える
  - 各ノードは接続辺の重みを参照可能
  - 重みは正整数であり，$O(\log n)$ビットで表現可能



  - 重みとメッセージ伝送のコストは無関係
    - 重みの大きい辺でも単位ラウンドでメッセージを伝送可能

その他注意事項

- ノードは一意なID($O(\log n)$ビット)を持つ
  - IDを持たない(使えない)モデル(匿名モデル)も存在するが今回は考えない
  - 隣接ノードのIDは既知
    - メッセージ通信複雑性を考えない場合は本質的な仮定ではない

- 各ノードはグラフのトポロジに関する知識を持たない
  - ネットワークがどうなっているか
  - 自分がどこに居るか？

は計算して初めてわかる

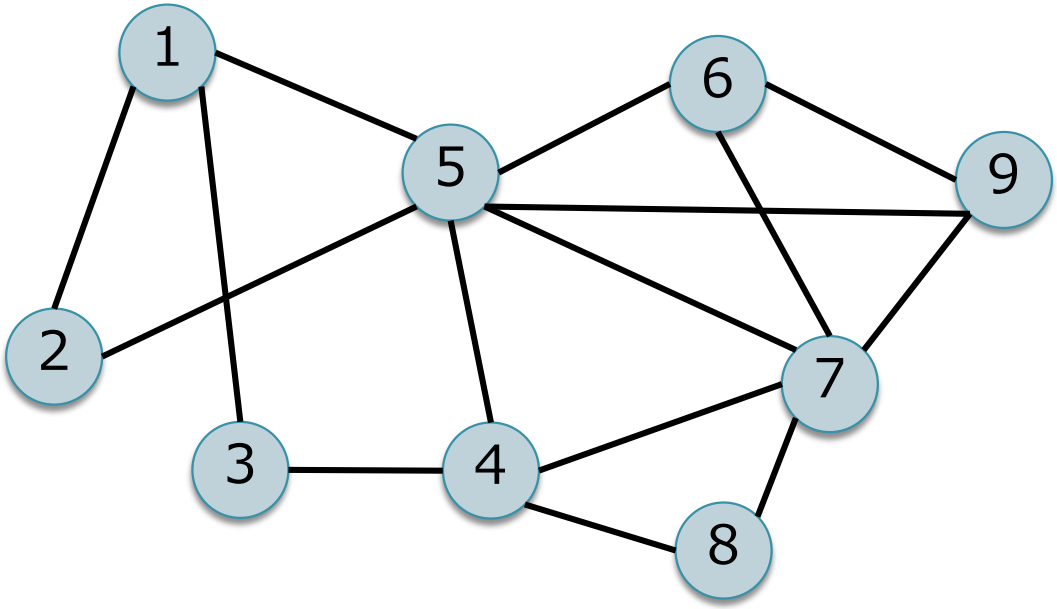
その他注意事項

- アルゴリズムはラウンド1からみんな一斉に開始される
  - 「誰が指示するのか」という点は追及してはいけない:-P
- 大域的な問題を考える場合，n, m,最大次数などは
  既知としても一般性を失わないことが多い
  - $O(D)$ラウンドの前処理で計算可能
- アルゴリズムの終了検知のためにDの(近似)値が必要と
  なる場合がある
  - 正確にDの値を求めるのは難しいが，2-近似値
    ($D \leq D' \leq 2D$なるD'の値)は容易に計算可能
    → 終了検知に利用するのであればこの値で十分

注：重み付きグラフの場合，Dは(重みなし)直径
(=重みを無視したときの直径)を表す

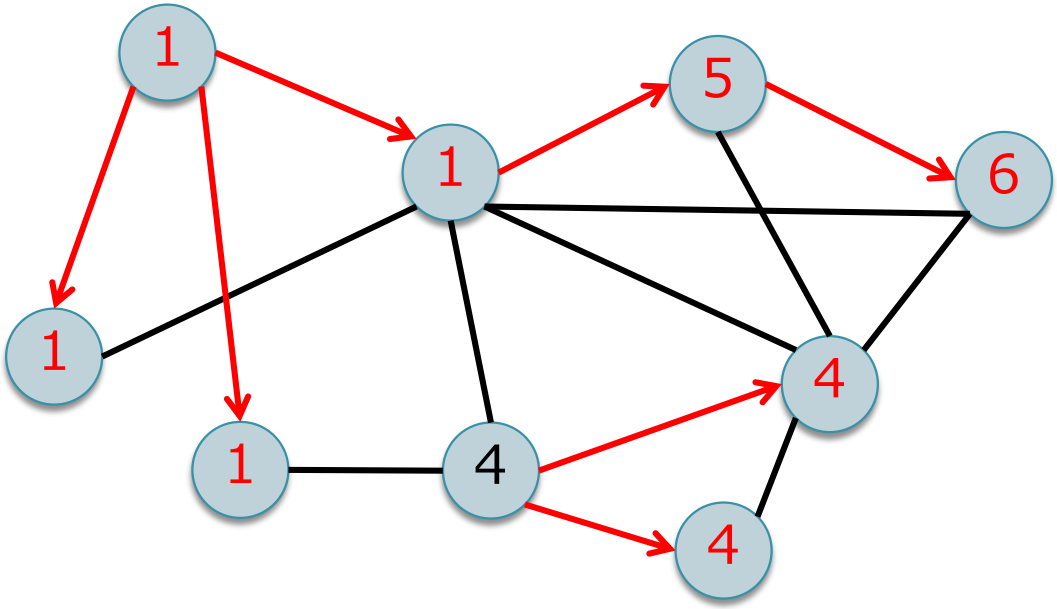
CONGESTモデルにおける万能解法

1：リーダ選挙+収集操作
2：集中型アルゴリズムを使って問題を解く
3：(必要であれば)答えを配布



CONGESTモデルにおける万能解法

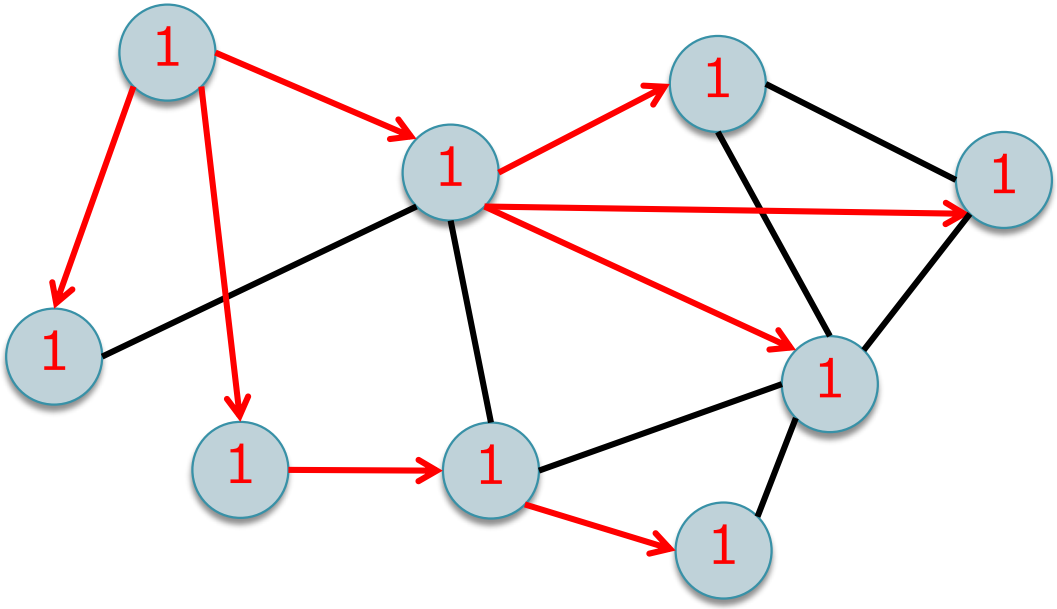
1：リーダ選挙+収集操作
2：集中型アルゴリズムを使って問題を解く
3：(必要であれば)答えを配布



CONGESTモデルにおける万能解法

1：リーダ選挙+収集操作
2：集中型アルゴリズムを使って問題を解く
3：(必要であれば)答えを配布

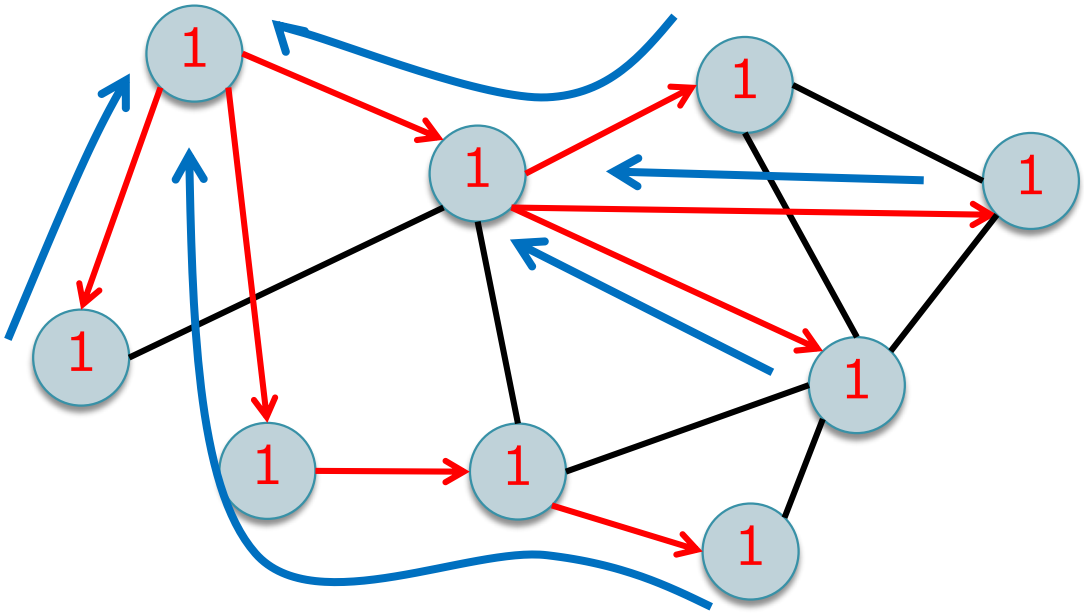
$O(D)$ラウンドで完了(終了検知はタイムアウトを利用)



CONGESTモデルにおける万能解法

1：リーダ選挙+収集操作
2：集中型アルゴリズムを使って問題を解く
3：(必要であれば)答えを配布

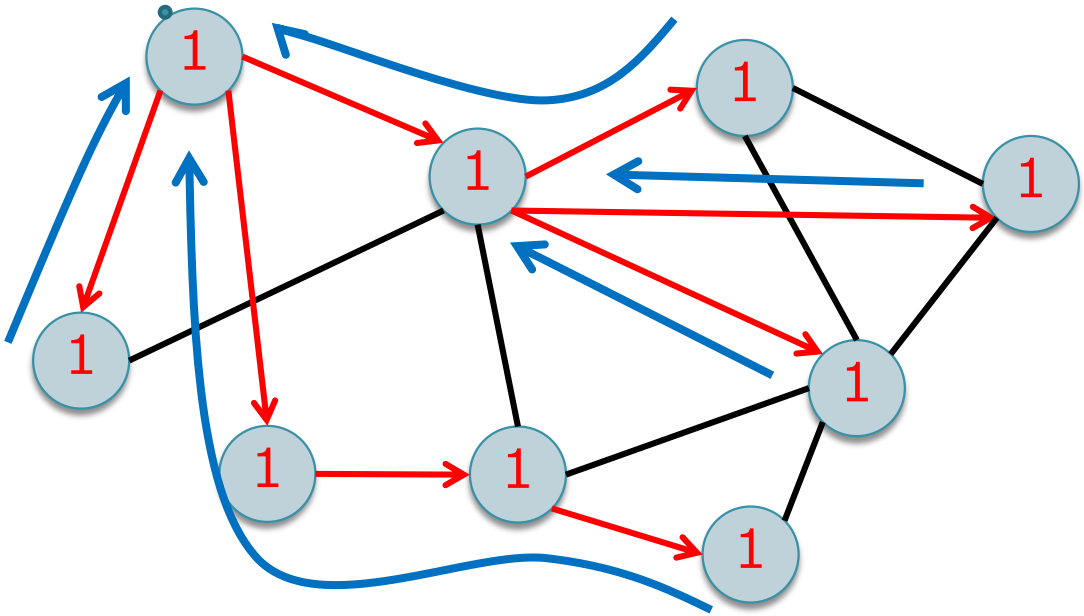
$O(m)$ラウンド(1ラウンドで伝送可能な情報=辺1本ぶん)



CONGESTモデルにおける万能解法

1：リーダ選挙+収集操作
2：集中型アルゴリズムを使って問題を解く
3：(必要であれば)答えを配布

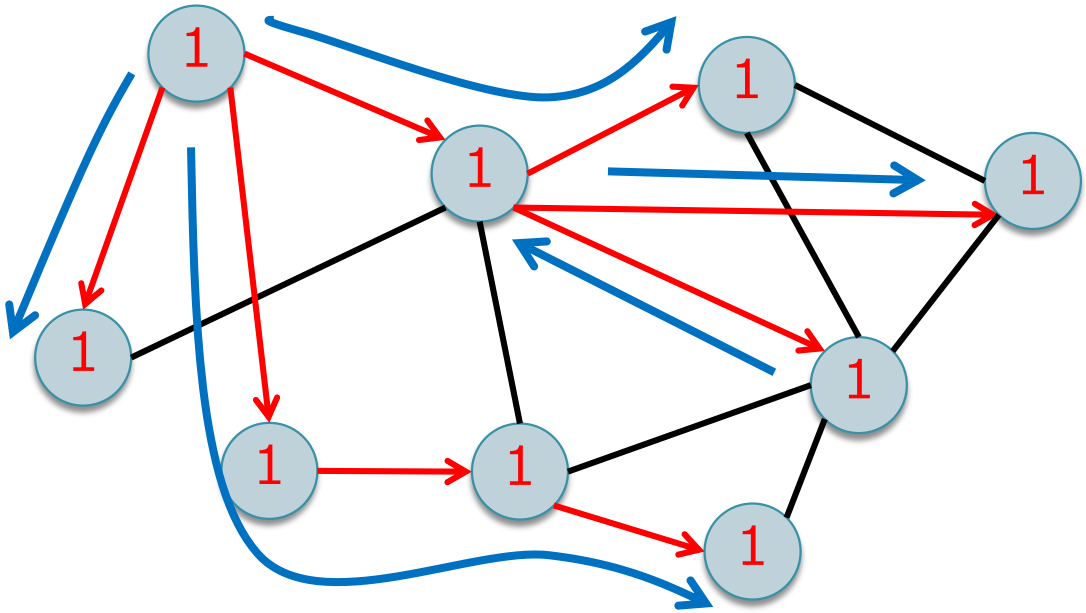
計算中…



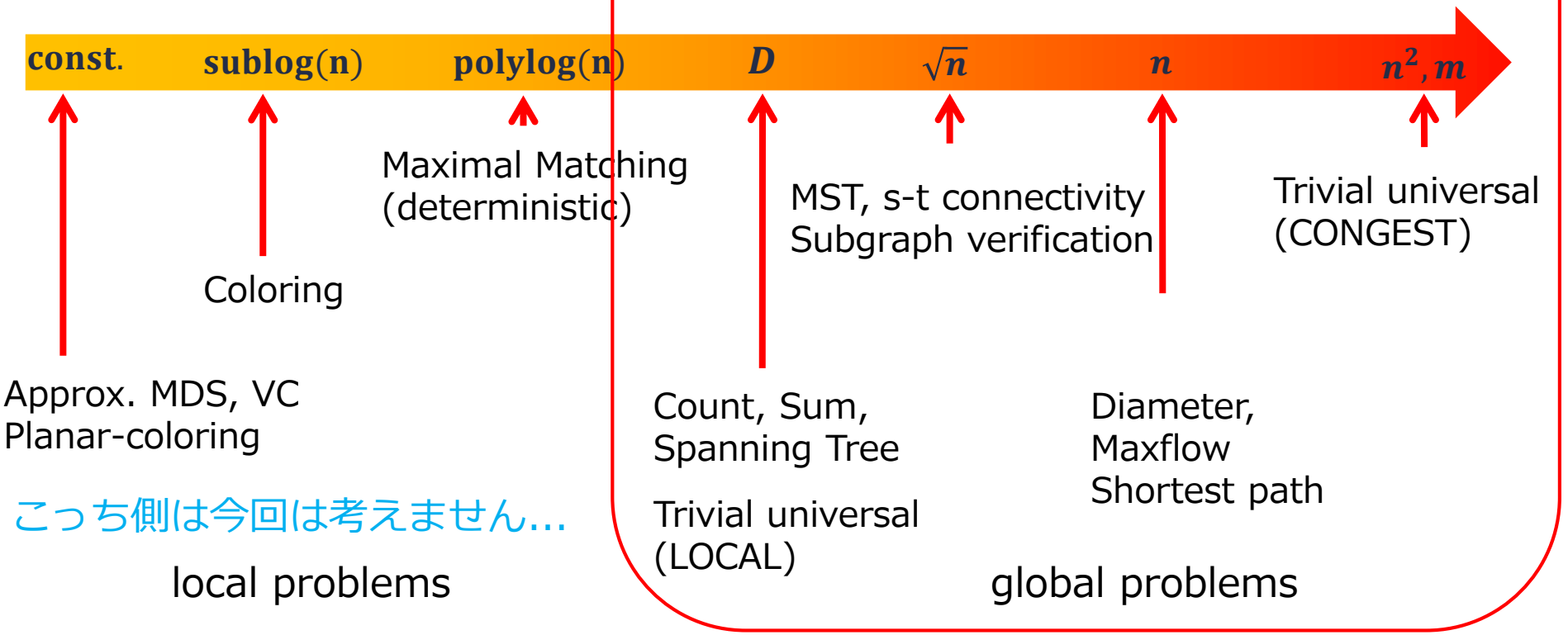
CONGESTモデルにおける万能解法

1：リーダ選挙+収集操作
2：集中型アルゴリズムを使って問題を解く
3：(必要であれば)答えを配布

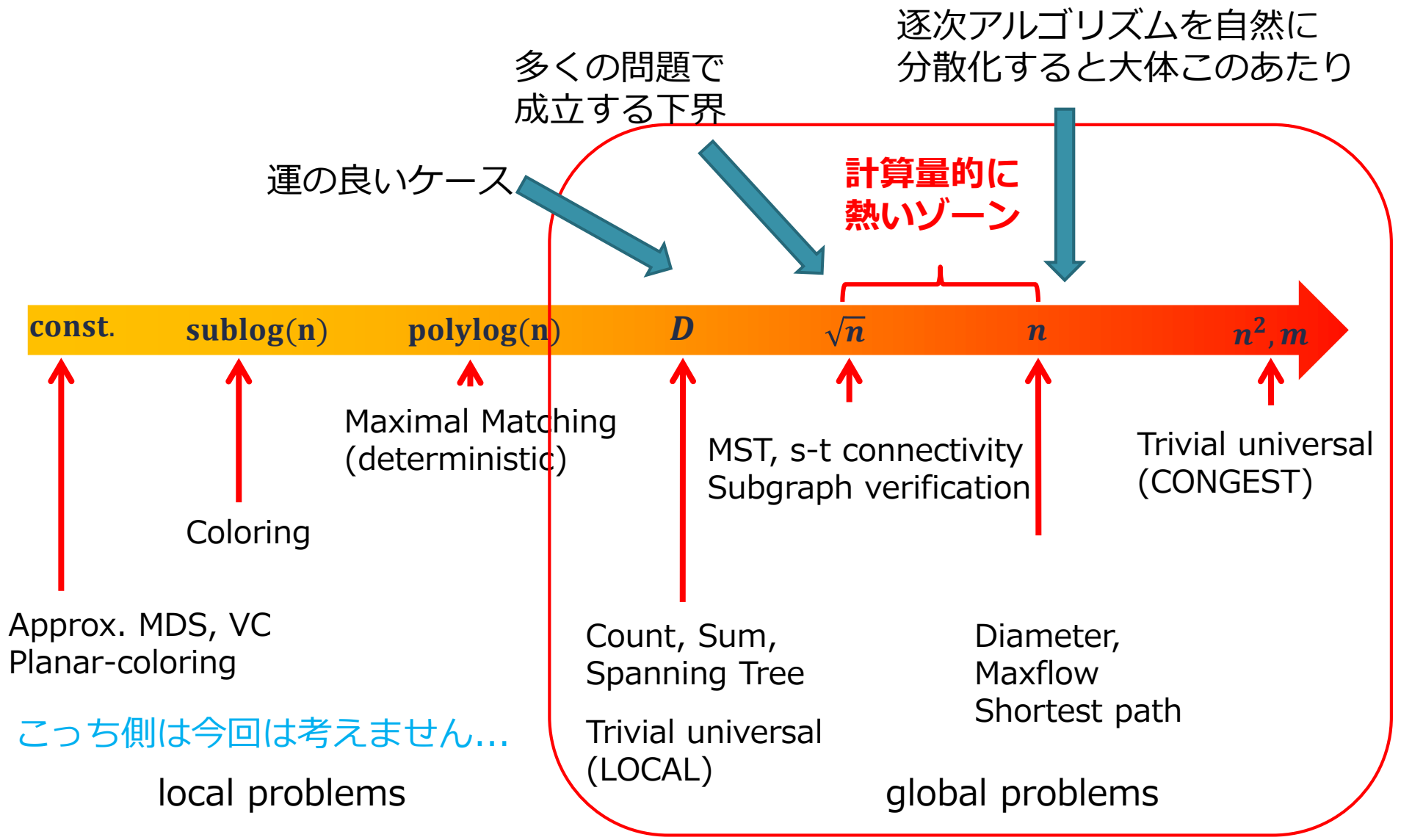
$O(?)$ラウンド(答えのサイズによる)



時間計算量の相場観


const.
sublog(n)
polylog(n)
D
√n
n
n², m
Approx. MDS, VC
Planar-coloring
Coloring
Maximal Matching
(deterministic)
Count, Sum,
Spanning Tree
Trivial universal
(LOCAL)
MST, s-t connectivity
Subgraph verification
Diameter,
Maxflow
Shortest path
Trivial universal
(CONGEST)
こっち側は今回は考えません...
local problems
global problems

時間計算量の相場観

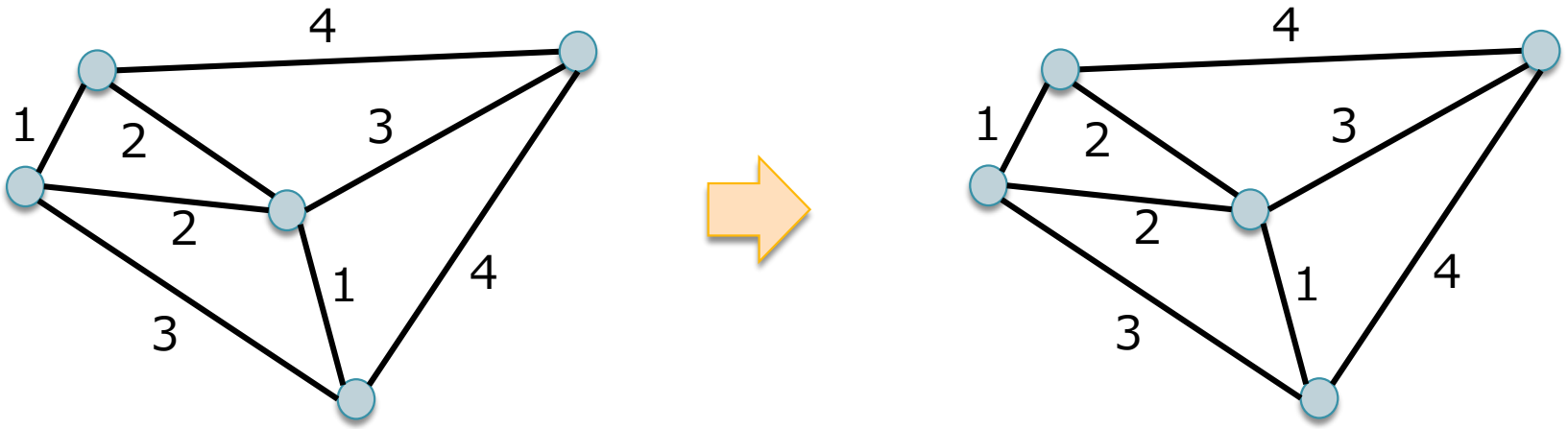

逐次アルゴリズムを自然に
分散化すると大体このあたり
多くの問題で
成立する下界
運の良いケース
計算量的に
熱いゾーン
const.
sublog(n)
polylog(n)
D
√n
n
n², m
Maximal Matching
(deterministic)
MST, s-t connectivity
Subgraph verification
Trivial universal
(CONGEST)
Coloring
Approx. MDS, VC
Planar-coloring
Count, Sum,
Spanning Tree
Trivial universal
(LOCAL)
Diameter,
Maxflow
Shortest path
こっち側は今回は考えません...
local problems
global problems

$\tilde{O}$記法

- 大域的な問題に対する分散アルゴリズムの設計では，$\mathrm{polylog}(n)$ 因子は「小さい値」として無視することが多い
  - 一般に$\mathrm{polylog}(n)$因子は長く煩雑になりがち
  - 指数時間Alg.で多項式部分を無視するのに近いかも?

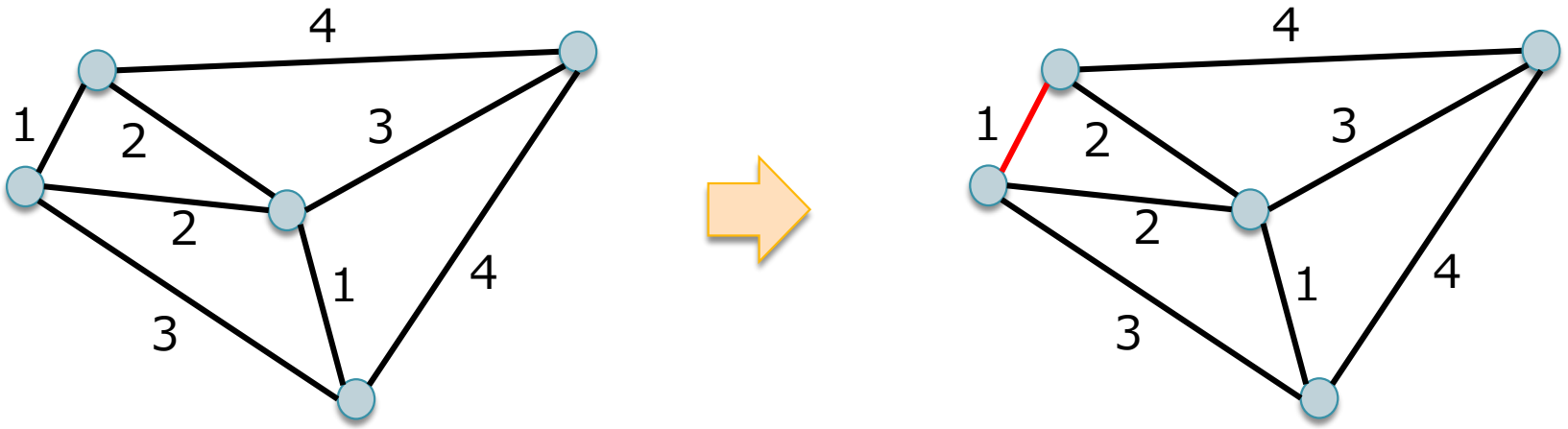
- soft-O記法 ($\tilde{O}$記法)
  - $\tilde{O}\big(f(n)\big) = O(f(n)\mathrm{polylog}(n))$
  - $\tilde{o}\big(f(n)\big) = o(f(n)\mathrm{polylog}(n))$
  - $\tilde{\Omega}\big(f(n)\big) = \Omega\left(\frac{f(n)}{\mathrm{polylog}(n)}\right)$

ケーススタディ:MST



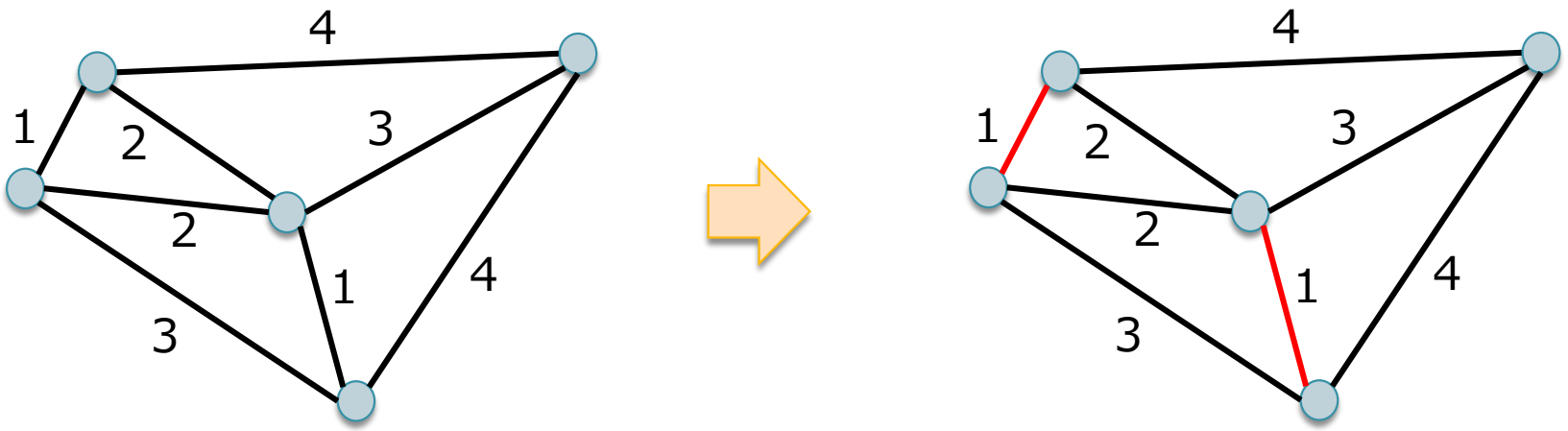
(お馴染みの) Kruskal法
1：$T \leftarrow \emptyset$
2：辺集合を重みの昇順にソート($e_1, e_2, \cdots e_m$ とする)
3：for each e_i do
4：　　if $T \cup \{e_i\}$が閉路を持たない then $T \leftarrow T \cup \{e_i\}$

ケーススタディ:MST



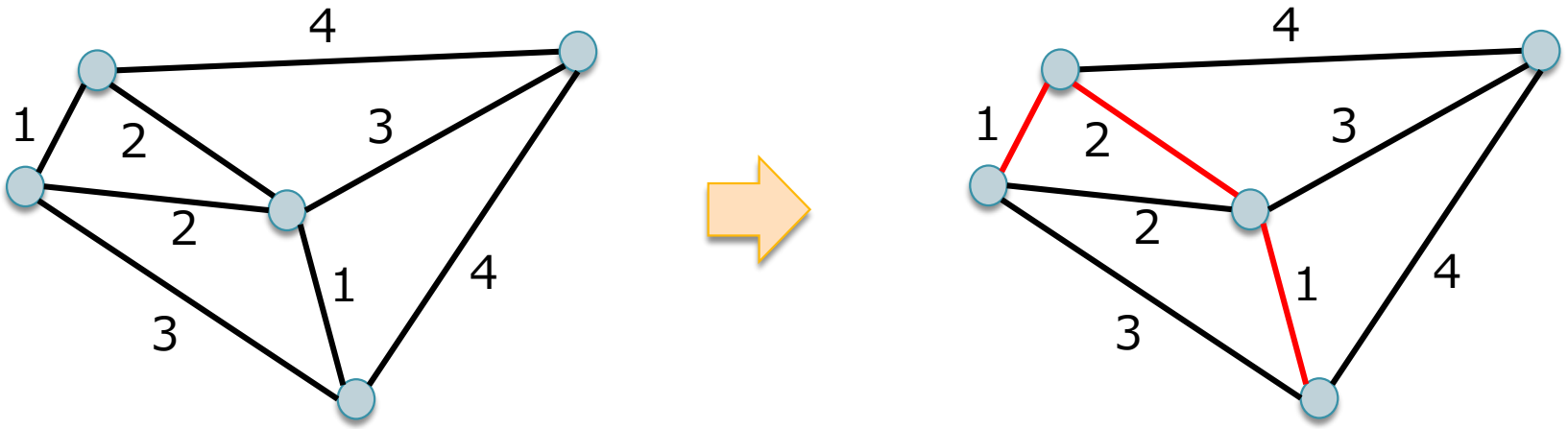
(お馴染みの) Kruskal法
1：$T \leftarrow \emptyset$
2：辺集合を重みの昇順にソート($e_1, e_2, \cdots e_m$ とする)
3：for each e_i do
4：　　if $T \cup \{e_i\}$が閉路を持たない then $T \leftarrow T \cup \{e_i\}$

ケーススタディ:MST



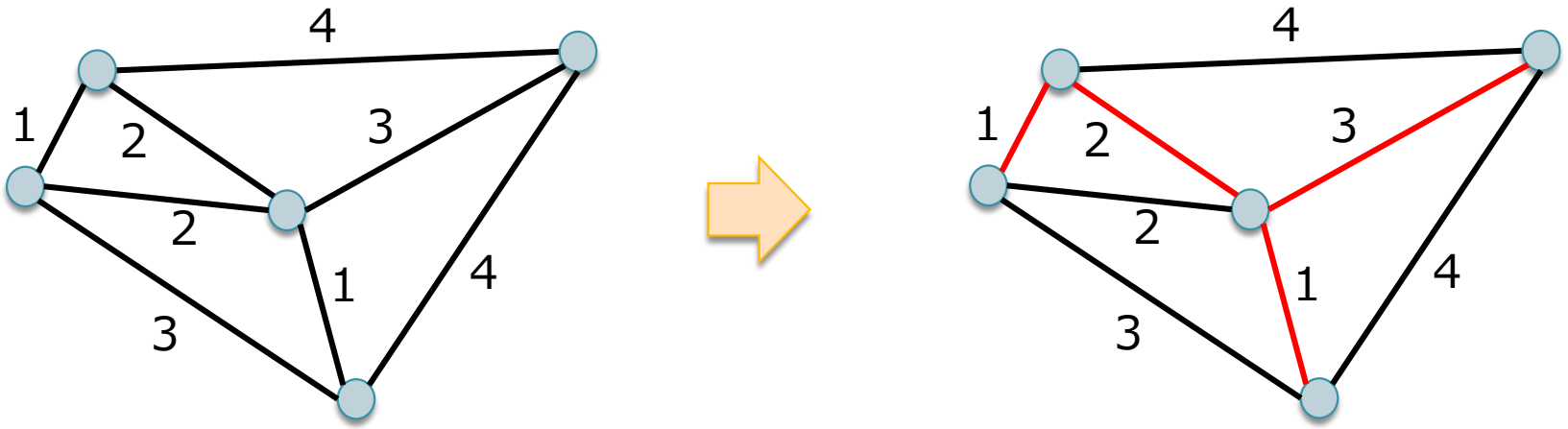
(お馴染みの) Kruskal法
1：$T \leftarrow \emptyset$
2：辺集合を重みの昇順にソート($e_1, e_2, \cdots e_m$ とする)
3：for each e_i do
4：　　if $T \cup \{e_i\}$が閉路を持たない then $T \leftarrow T \cup \{e_i\}$

ケーススタディ:MST



(お馴染みの) Kruskal法
1：$T \leftarrow \emptyset$
2：辺集合を重みの昇順にソート($e_1, e_2, \cdots e_m$ とする)
3：for each e_i do
4：　　if $T \cup \{e_i\}$が閉路を持たない then $T \leftarrow T \cup \{e_i\}$

ケーススタディ:MST



(お馴染みの) Kruskal法
1：$T \leftarrow \emptyset$
2：辺集合を重みの昇順にソート($e_1, e_2, \cdots e_m$ とする)
3：for each e_i do
4：　　if $T \cup \{e_i\}$が閉路を持たない then $T \leftarrow T \cup \{e_i\}$

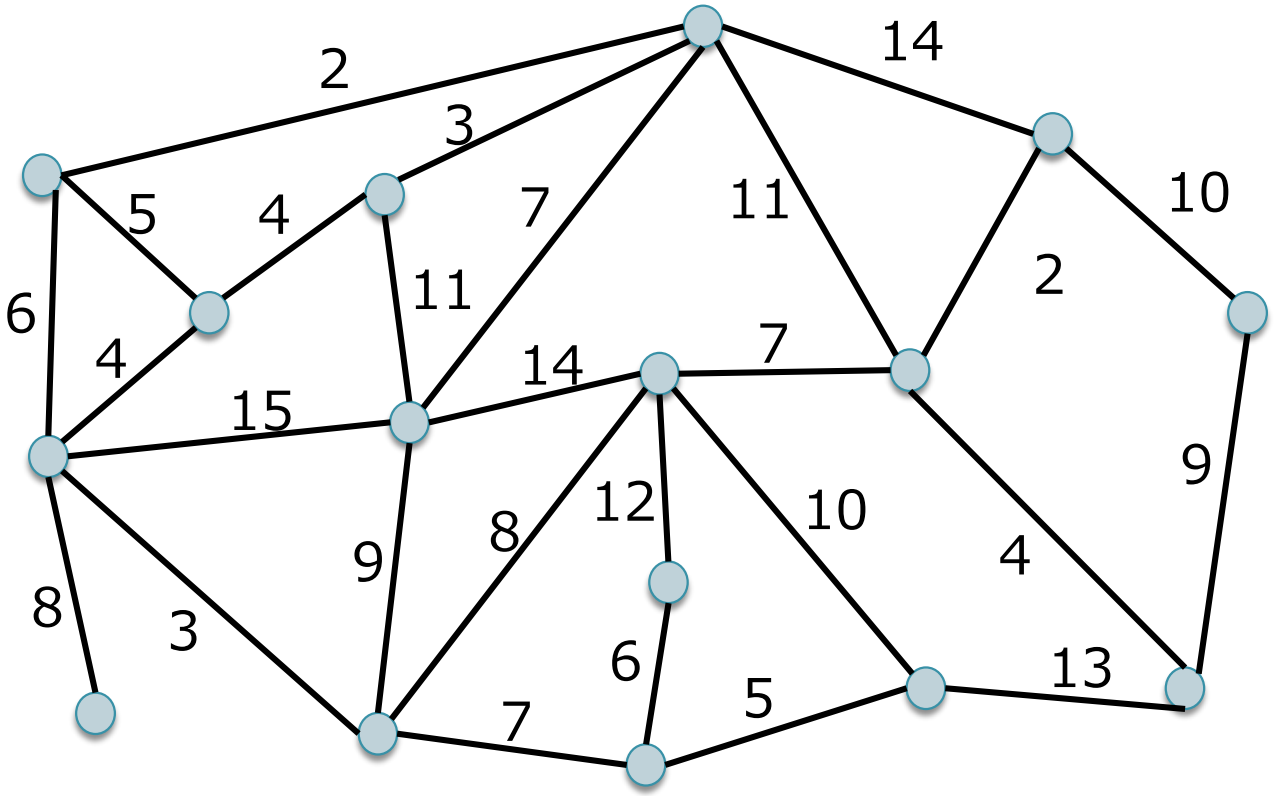
MST：分散化への道

- 「greedyはいやだ」
  - Kruskalは本質的に逐次っぽい感じがする

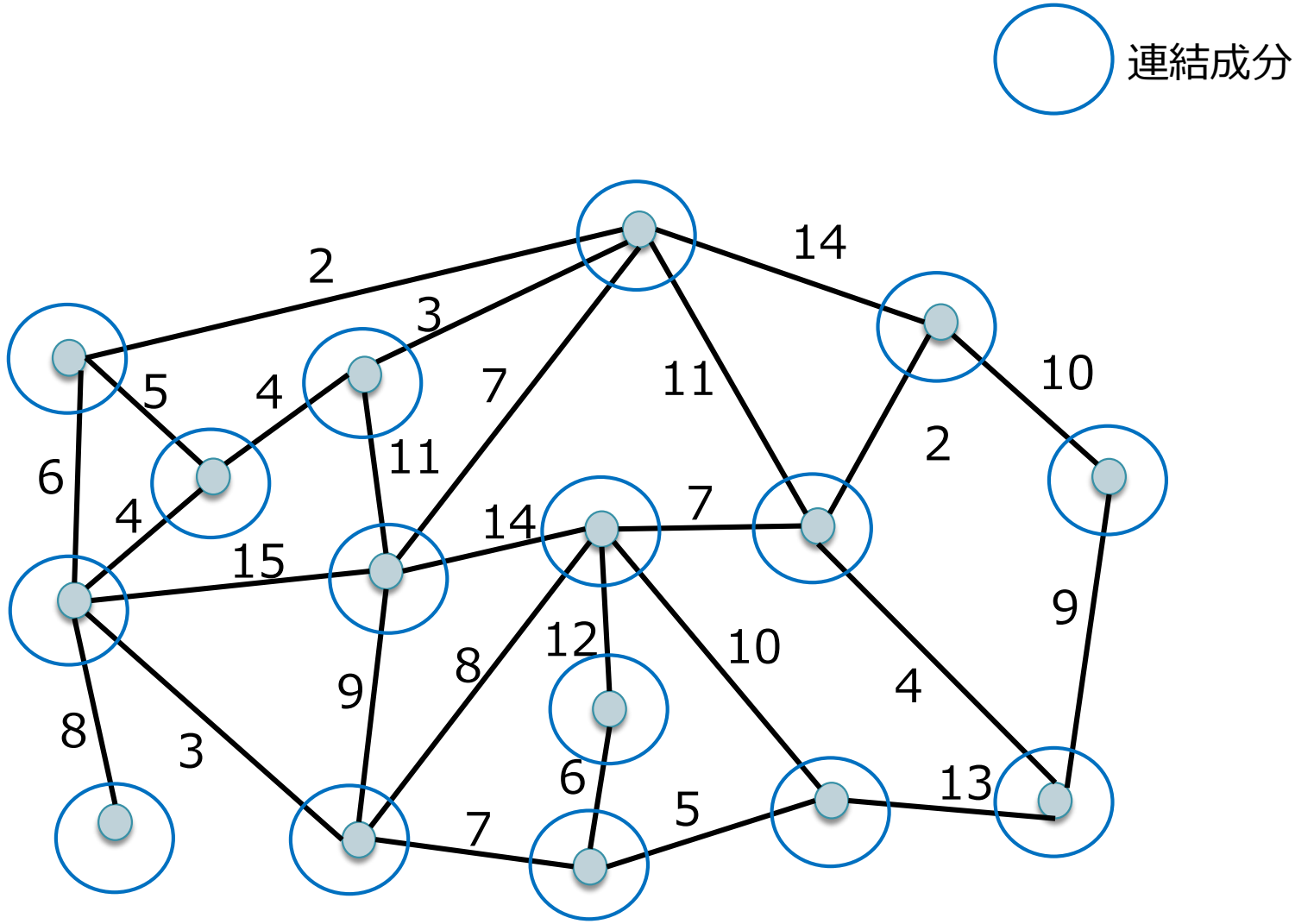
- 代替案：Boruvka法

Boruvka法
1：$T \leftarrow \emptyset$
2：$F(T) \leftarrow$ グラフ(V,T)の連結成分(森)集合
3：while Tが全域木でない do
4：　for each $f \in F$ do
5：　　$M \leftarrow$端点の一方だけがfに属する辺で重み最小のもの
　　(MOE (Minimum Outgoing Edge) of f)
6：　$T \leftarrow T \cup M$ として $F(T)$を更新

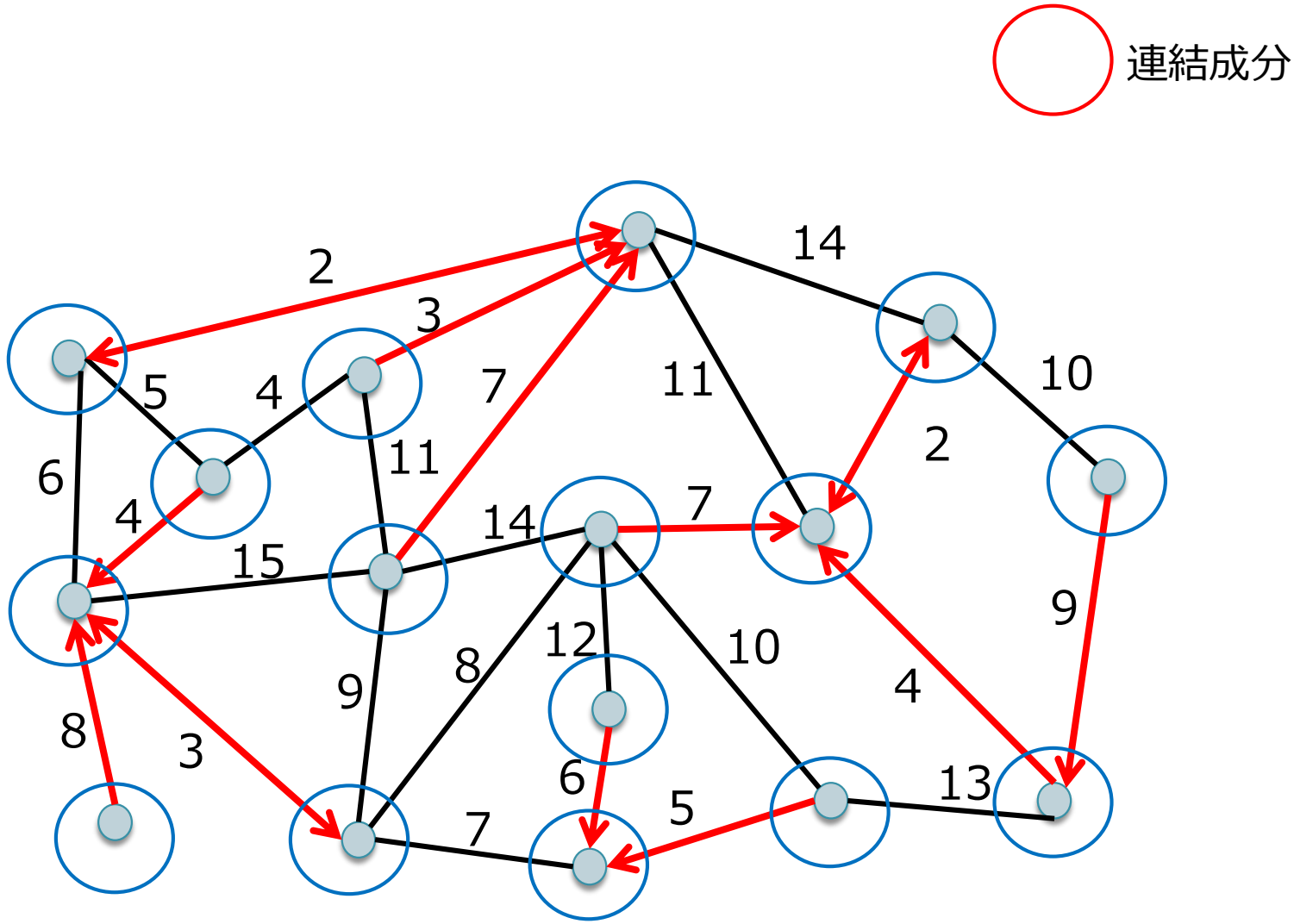
Boruvka法：動作例



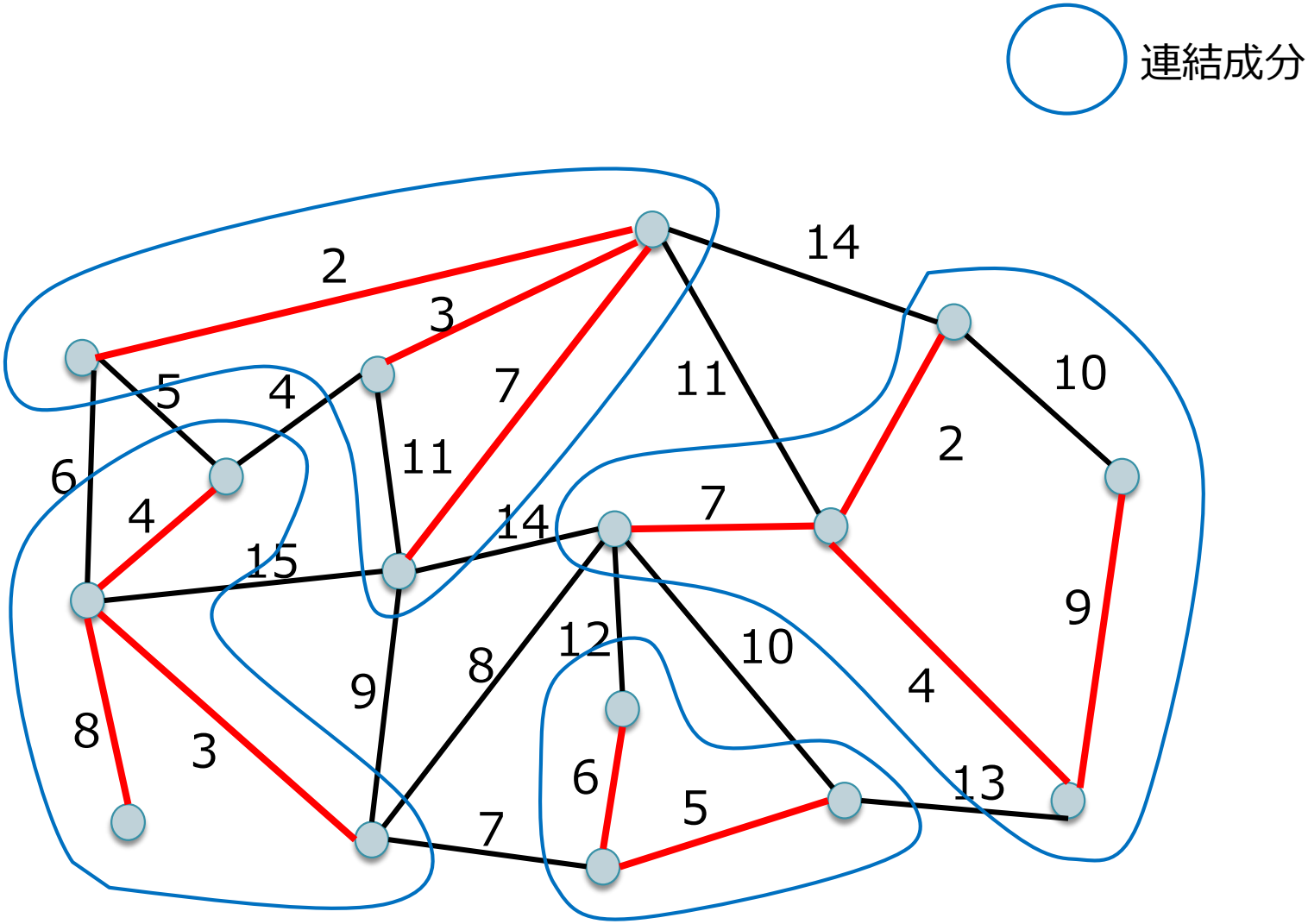
Boruvka法：動作例



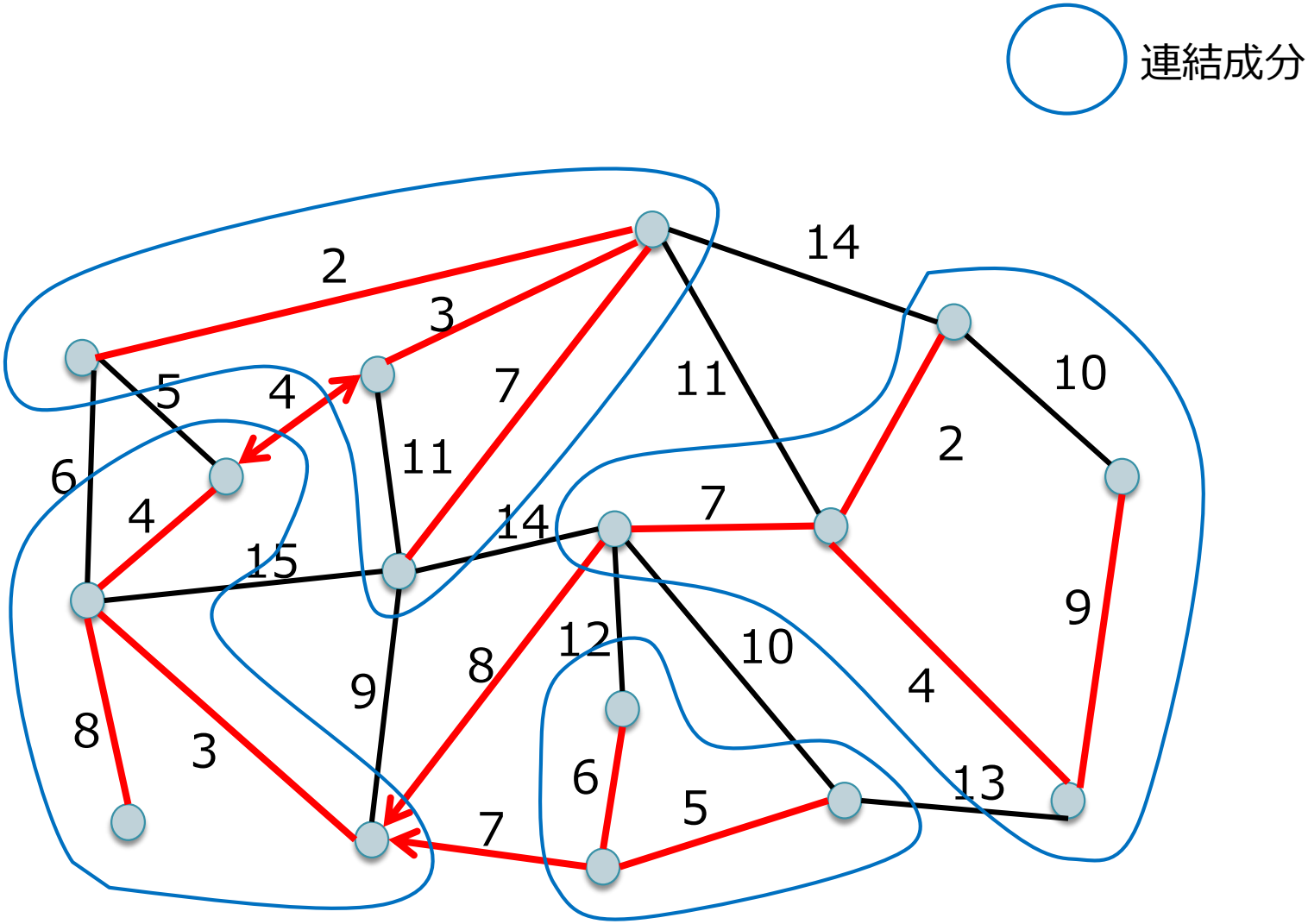
Boruvka法：動作例



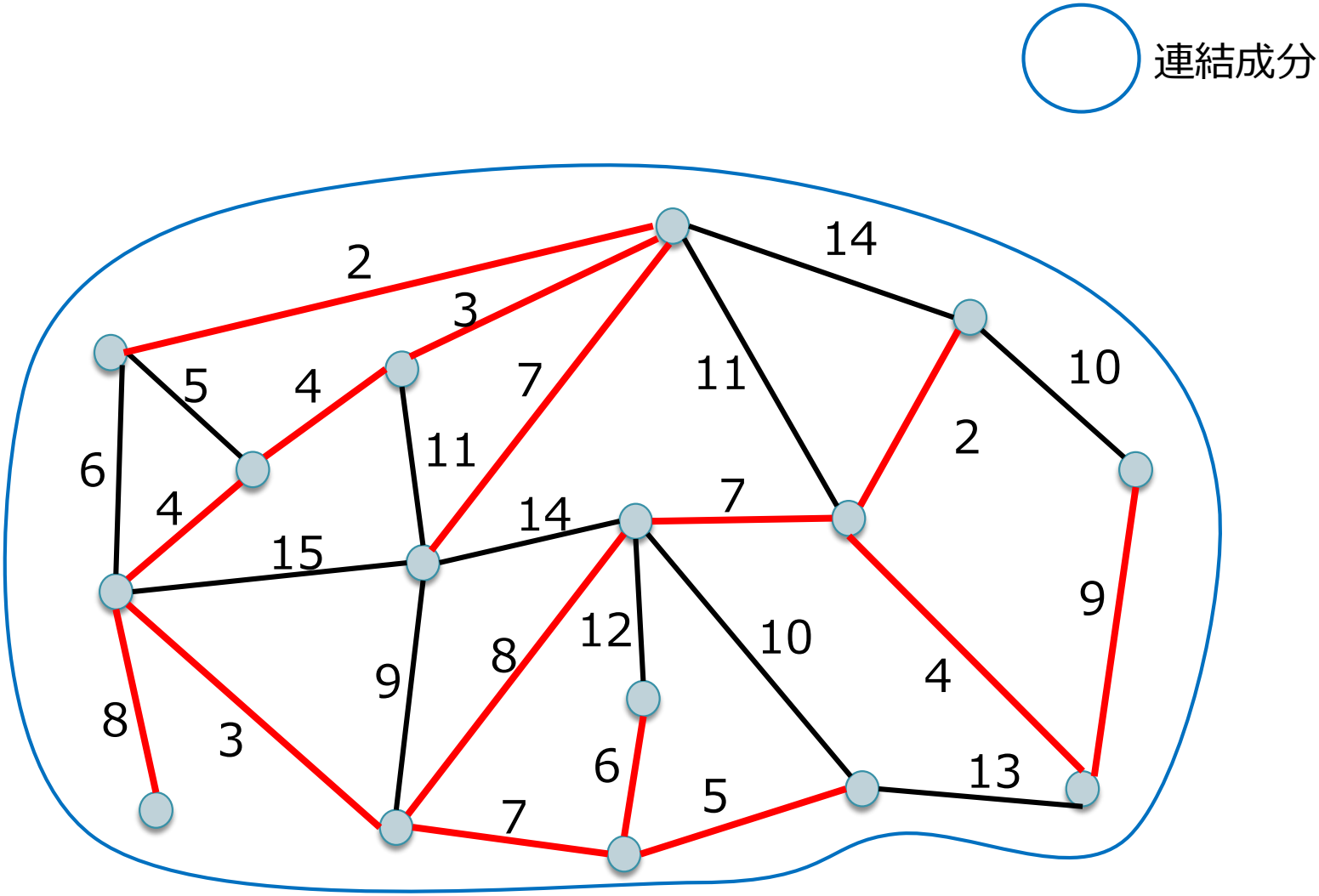
Boruvka法：動作例



Boruvka法：動作例



Boruvka法：動作例



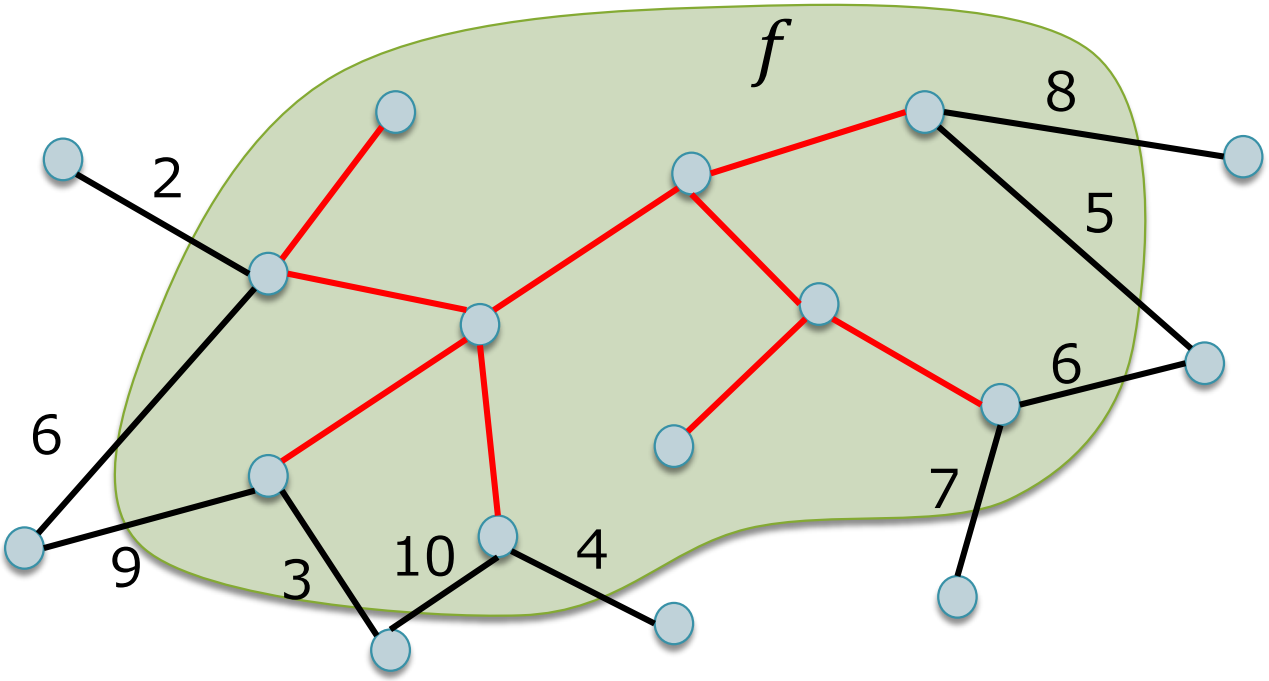
Gallager-Humble-Spiraのアルゴリズム[GHS82]

- Boruvka法の自然な分散型実装
  - 各頂点vは構成途中の森Tに対して$T \cap N(v)$を保持

Gallager-Humble-Spira (GHS)
1：$T \leftarrow \emptyset$
2：$F(T) \leftarrow$ グラフ(V,T)の連結成分集合(森)
3：while $|F(T)| > 1$ do
4：　各 $f \in F(T)$ の内部で以下のステップを並列に動作
5：　　f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6：　　f中の各頂点vにおいてMOEの候補を選出
7：　　MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8：　　MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える

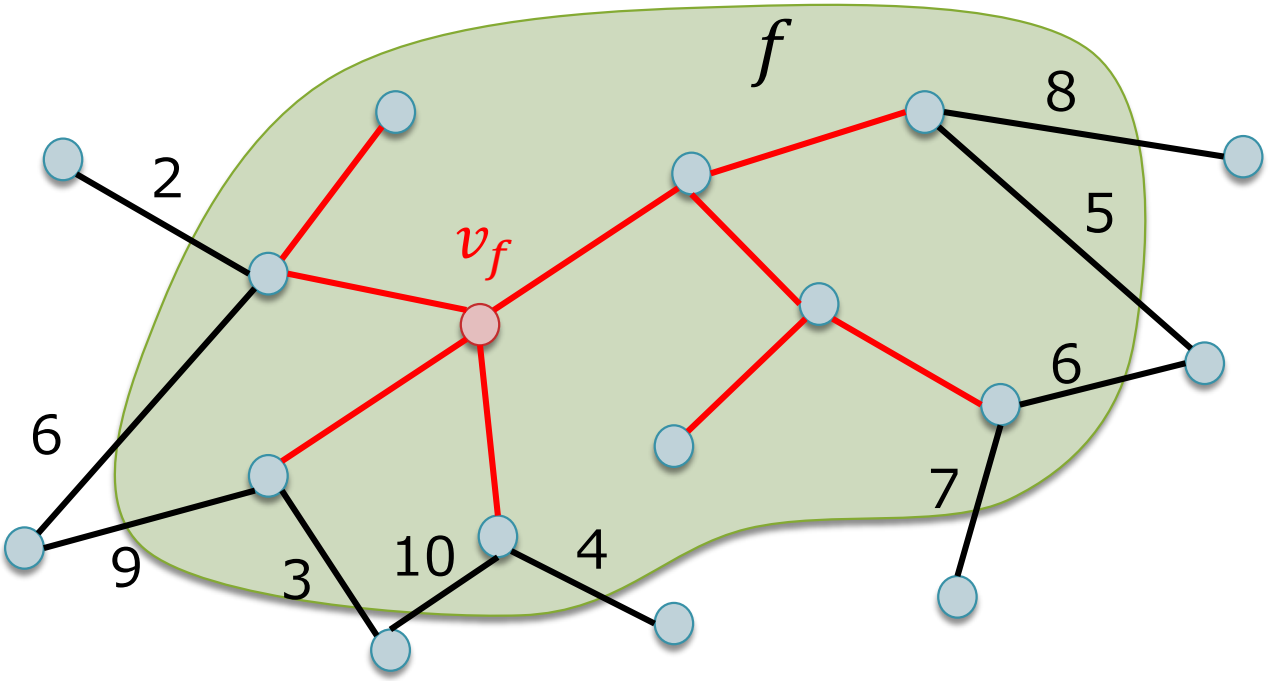
GHS：詳細

5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8： MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える



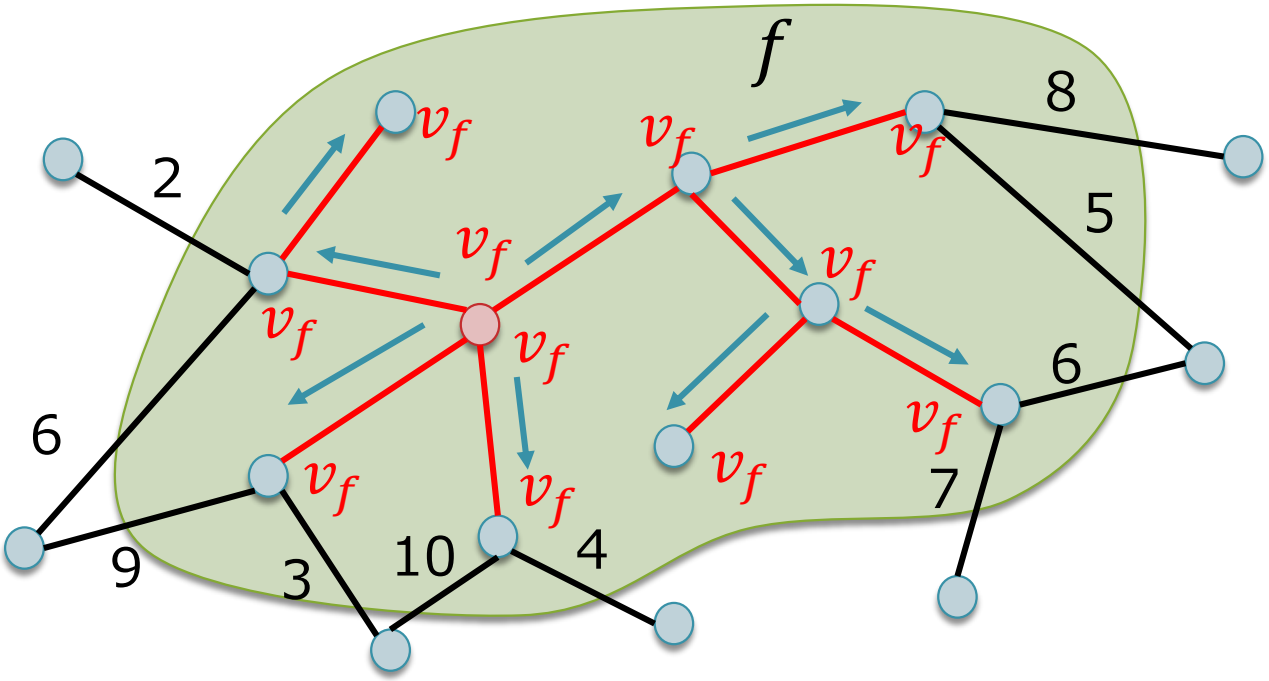
GHS：詳細

5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8： MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える



GHS：詳細

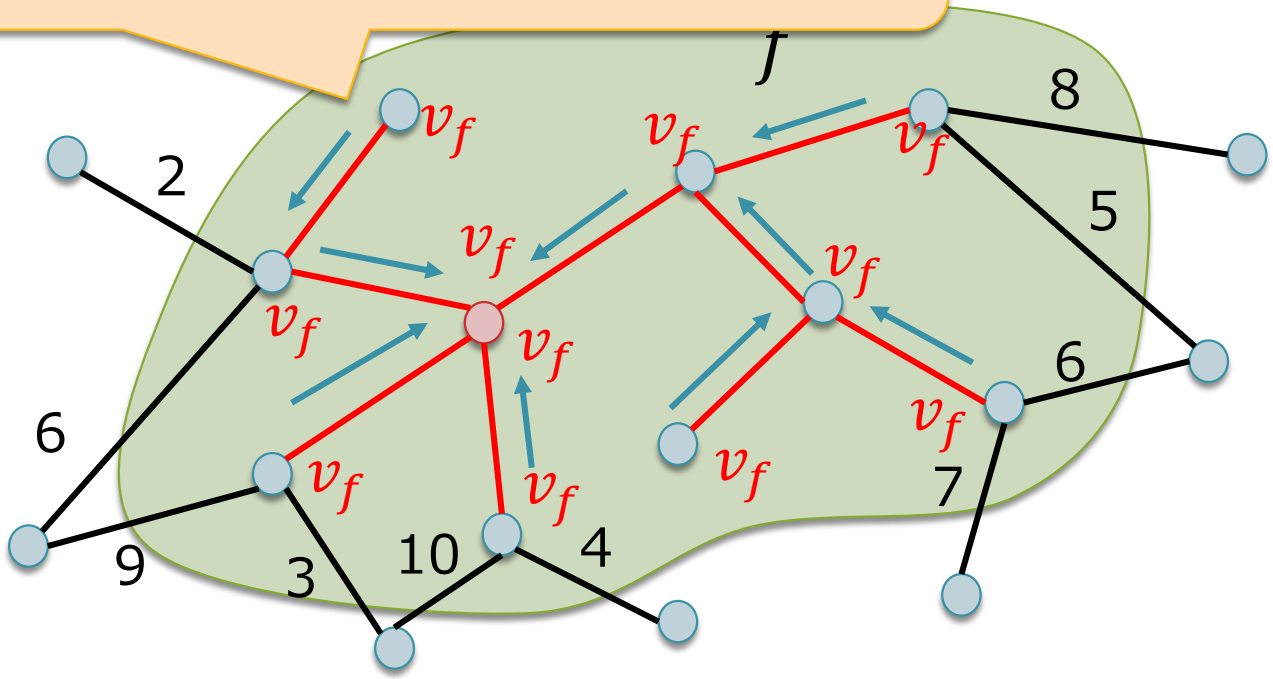
5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8： MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える



GHS：詳細

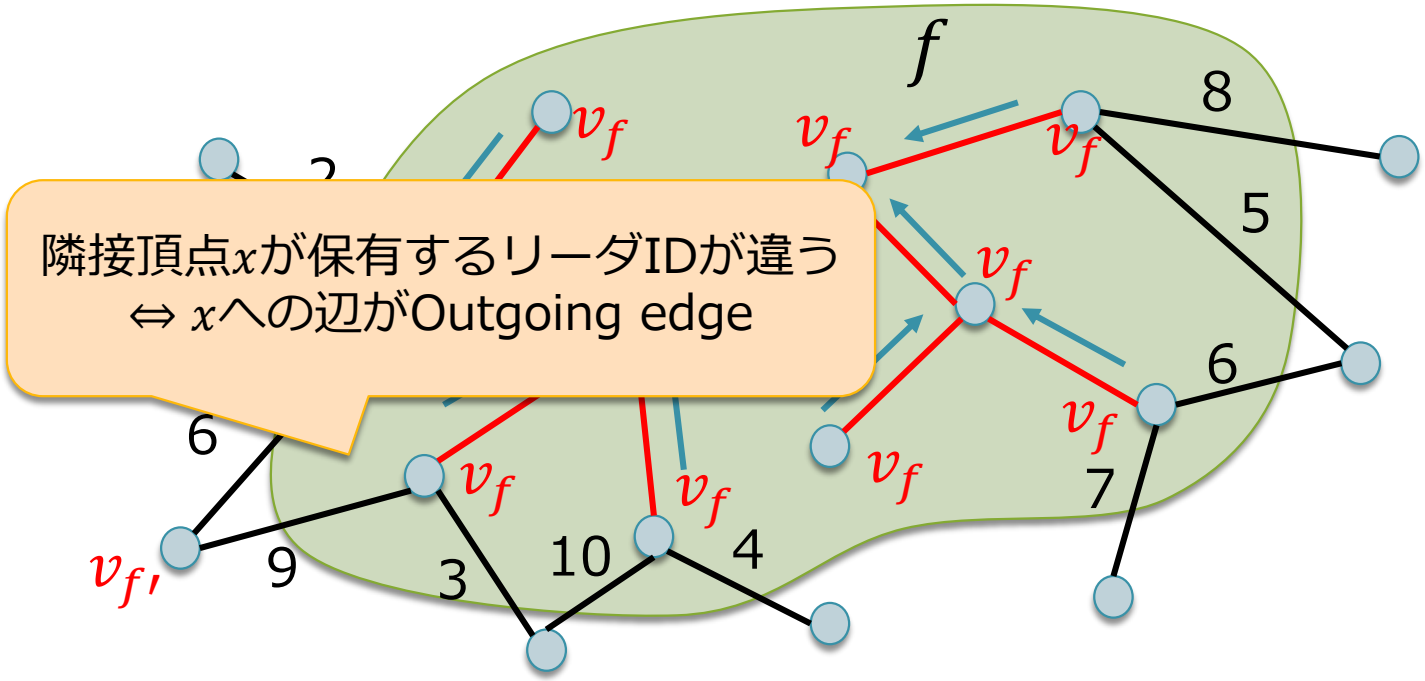
5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8： MOE(f)をf中の全頂点に放送して，MOE(f)の端点は

リーダへのパスを同時に確立
(リーダIDがやってきた方向を覚えておく)



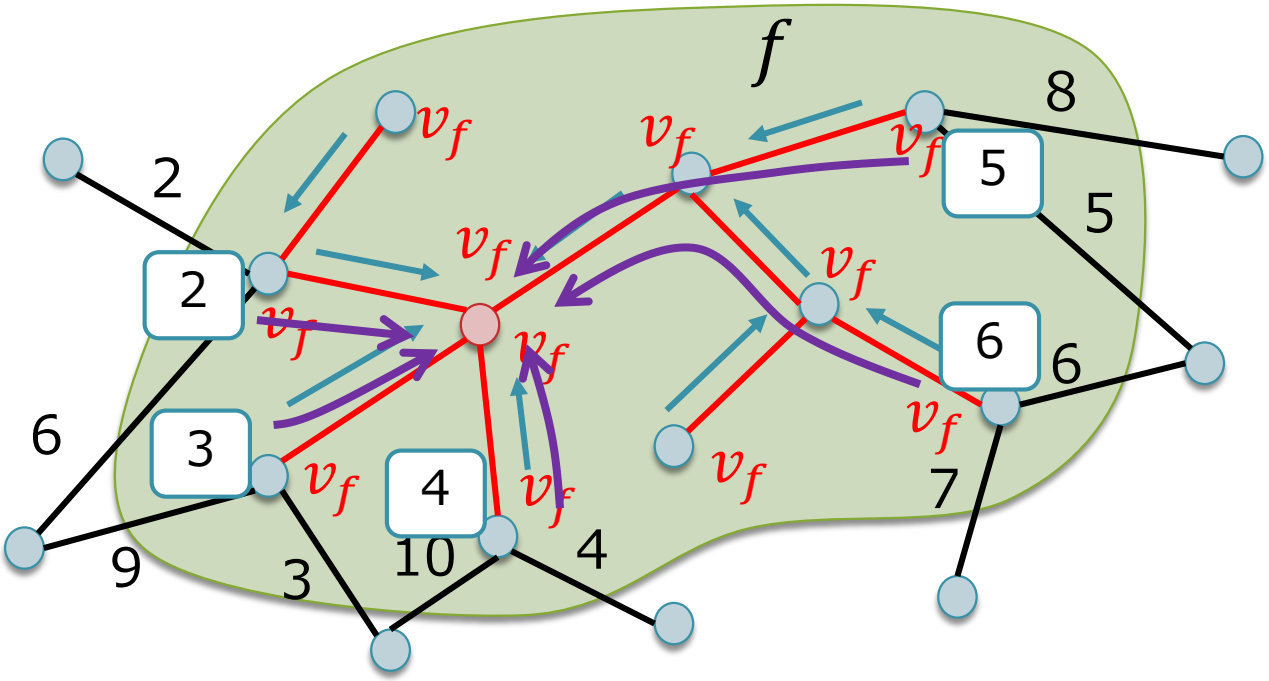
GHS：詳細

5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8： MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える



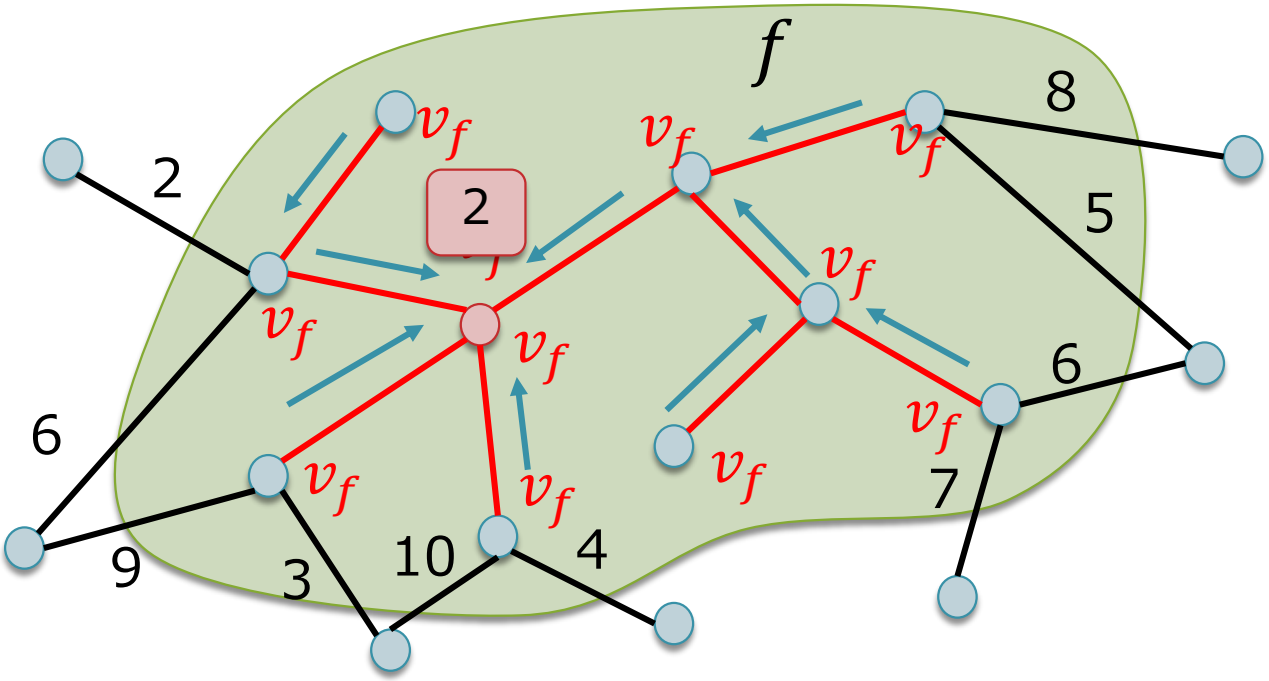
GHS：詳細

5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8： MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える



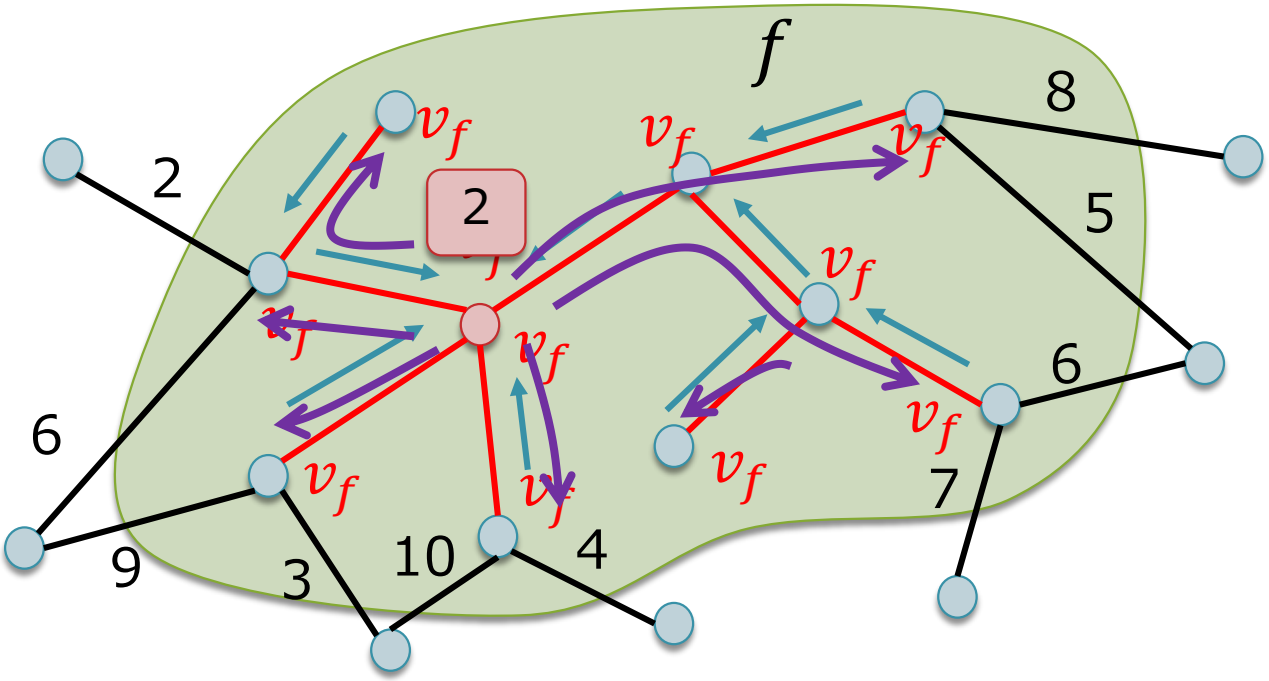
GHS：詳細

5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8： MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える



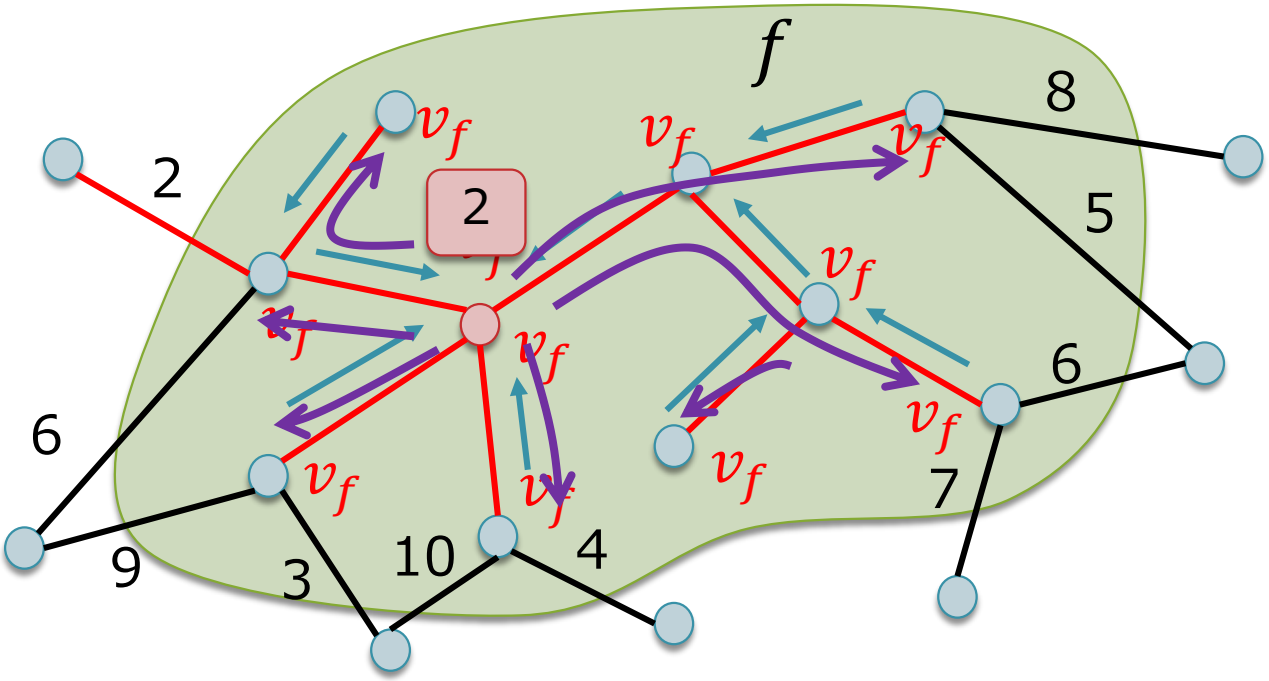
GHS：詳細

5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8： MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える



GHS：詳細

5： f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6： f中の各頂点vにおいてMOEの候補を選出
7： MOEの候補から最小重みの辺(=MOE(f))を選ぶ
6： MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える



GHS：実行時間

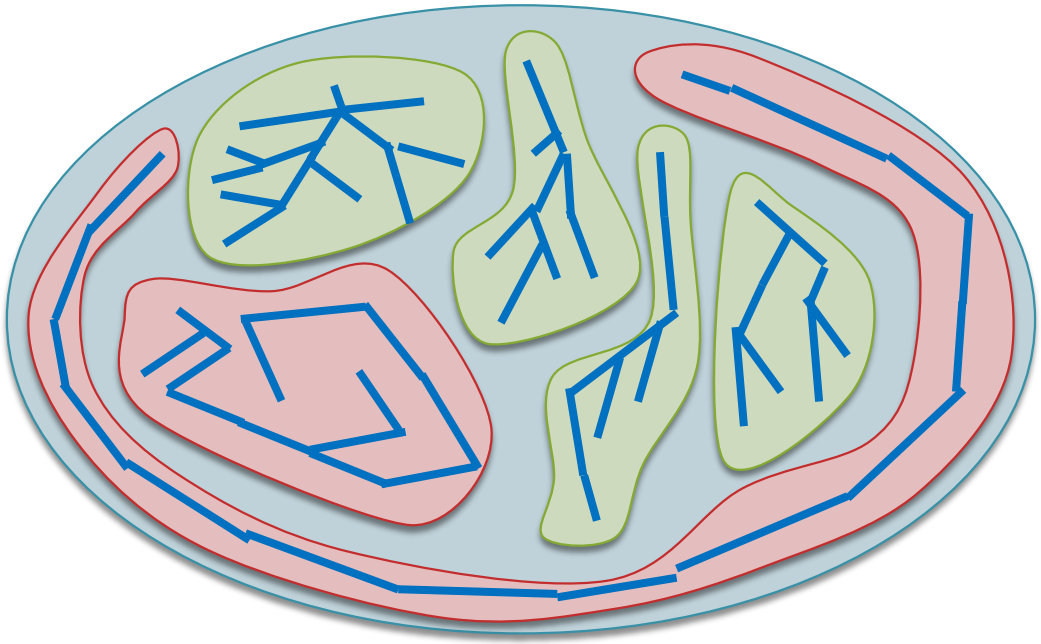
Gallager-Humble-Spira (GHS) （神の視点での記述）
1：$T \leftarrow \emptyset$
2：$F(T) \leftarrow$ グラフ(V, T)の連結成分集合(森)
3：while $|F(T)| > 1$ do
4：　各 $f \in F(T)$ の内部で以下のステップを並列に動作
5：　　f中の頂点からリーダv_fを選出し，そのIDをf中に放送
6：　　f中の各頂点vにおいてMOEの候補を選出
7：　　MOEの候補から最小重みの辺(=MOE(f))を選ぶ
8：　　MOE(f)をf中の全頂点に放送して，MOE(f)の端点はMOE(f)をTに加える

- 5-8の処理は$O(\max_{f \in F(T)} f$の直径)時間
- 5-8の反復1回で連結成分数は少なくとも半減

$\rightarrow O(\log n)$ 回の反復でアルゴリズム終了

GHS：実行時間

- fの直径は$\Omega(n)$になり得る(たとえ$D \ll n$でも)



部分グラフの直径はDでは抑えられない!

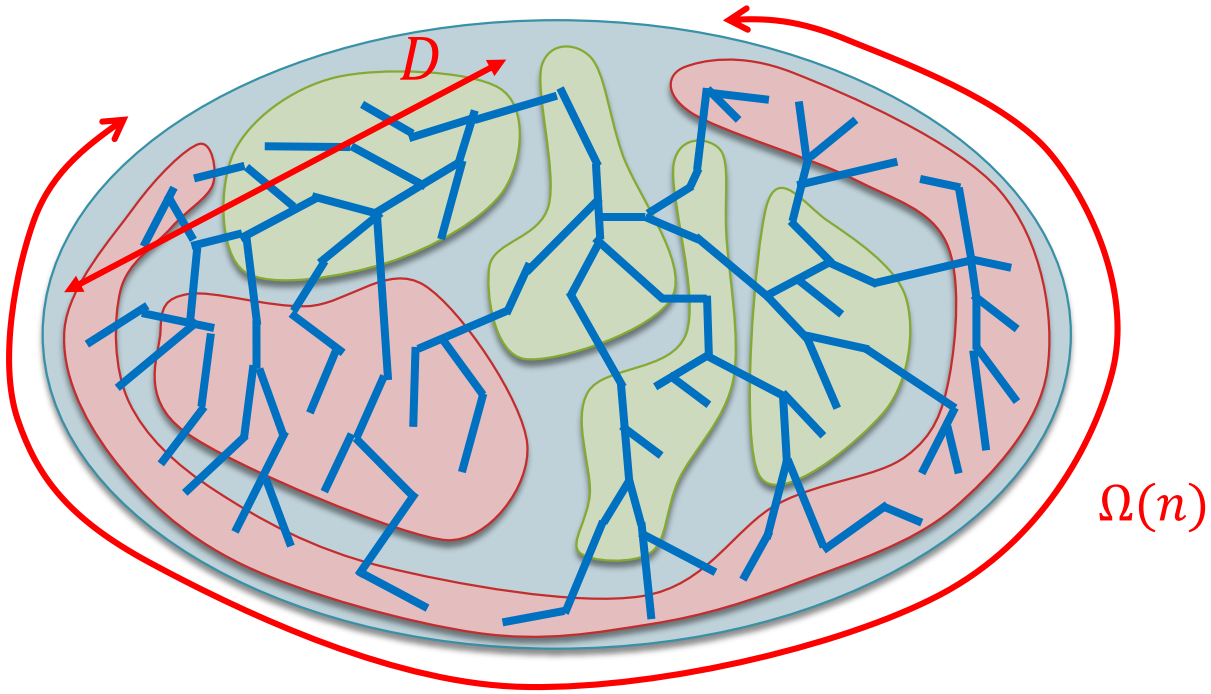
GHS：実行時間

定理1
GHSアルゴリズムは$O(n \log n)$ラウンドでMSTを構成する

- 最悪時に備えて，各回の反復(5-8の処理)は
  $O(n)$ラウンド待たないといけない
- 序盤戦において各連結成分は小さい傾向があるので
  待ち時間は無駄があるように見えるかもしれない
- が，アルゴリズム全体の実行時間評価としては
  実は漸近的にタイト
  - すなわち，$\max_f f$ の直径$=\Theta(n)$となるような反復が$\Omega(\log n)$回
    生じるようなインスタンスが存在する

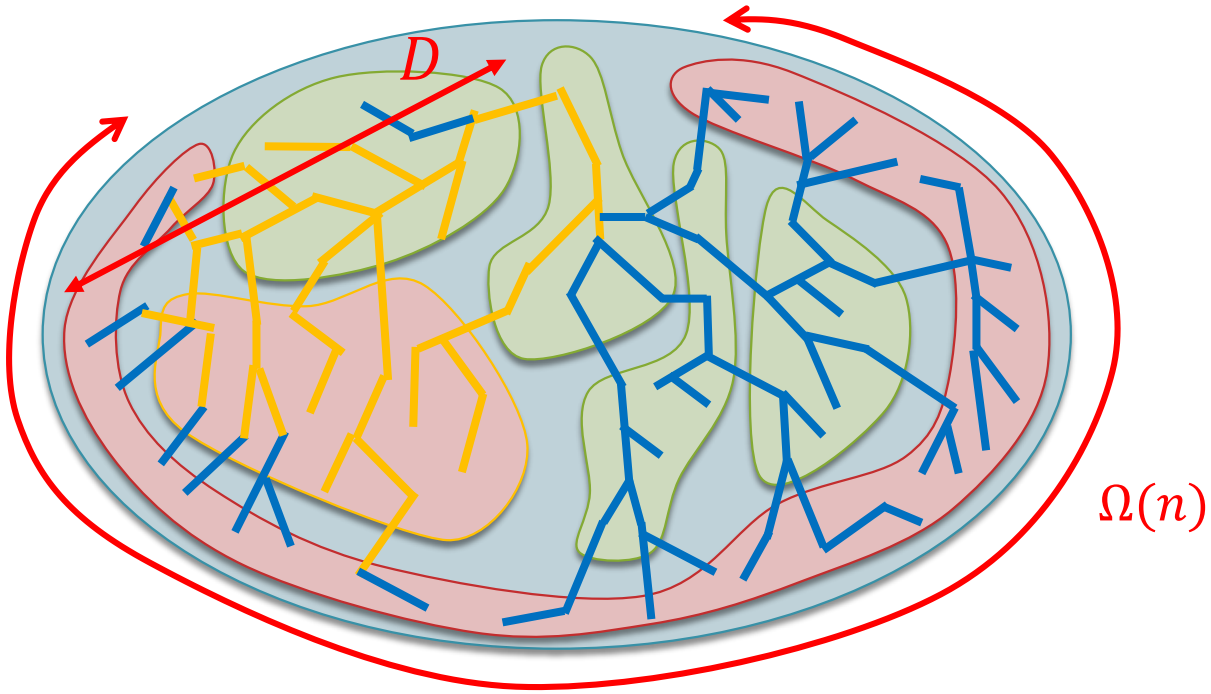
もっと速くするには？

- 問題はどこに？
  - MOE(f)の計算にf内部のみの通信網を利用
  - グラフ全体(幅優先木)で収集&放送を行えば
    もっと速いのだが，競合が問題



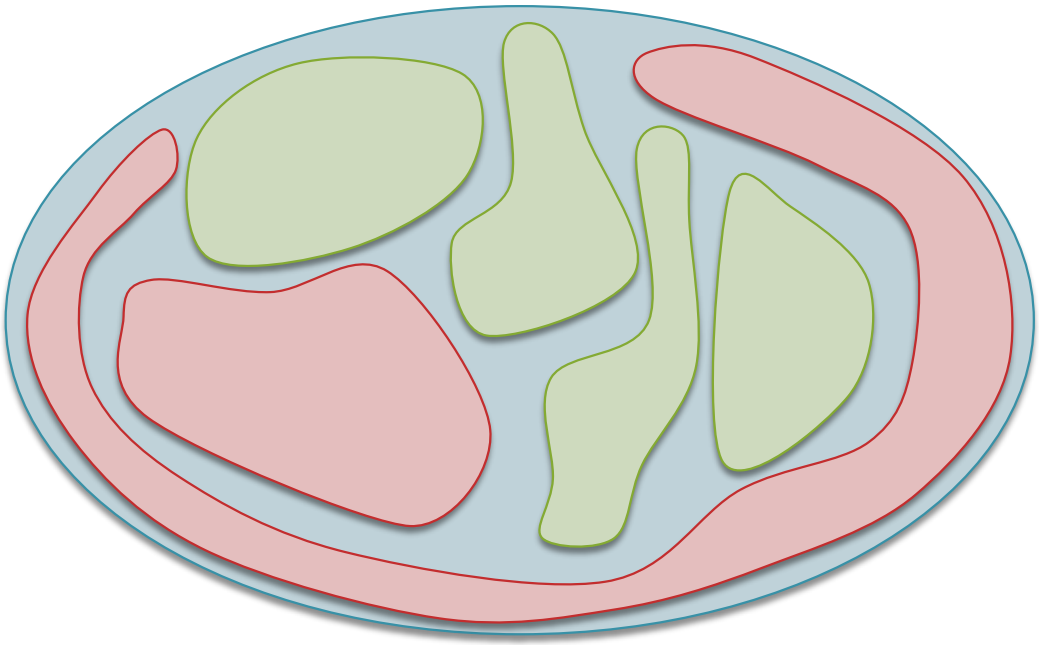
もっと速くするには？

- 問題はどこに？
  - MOE(f)の計算にf内部のみの通信網を利用
  - グラフ全体(幅優先木)で収集&放送を行えば
    もっと速いのだが，競合が問題



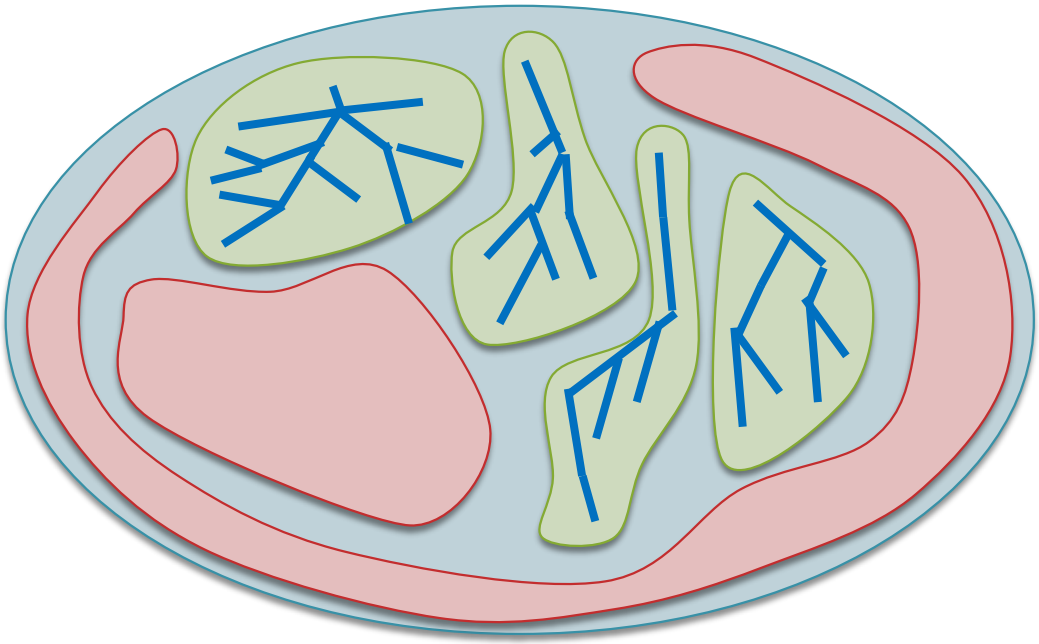
折衷的解決

- $|f| \leq \sqrt{n}$：GHSに準じる
- $|f| > \sqrt{n}$：幅優先木でMOEを発見&更新



折衷的解決

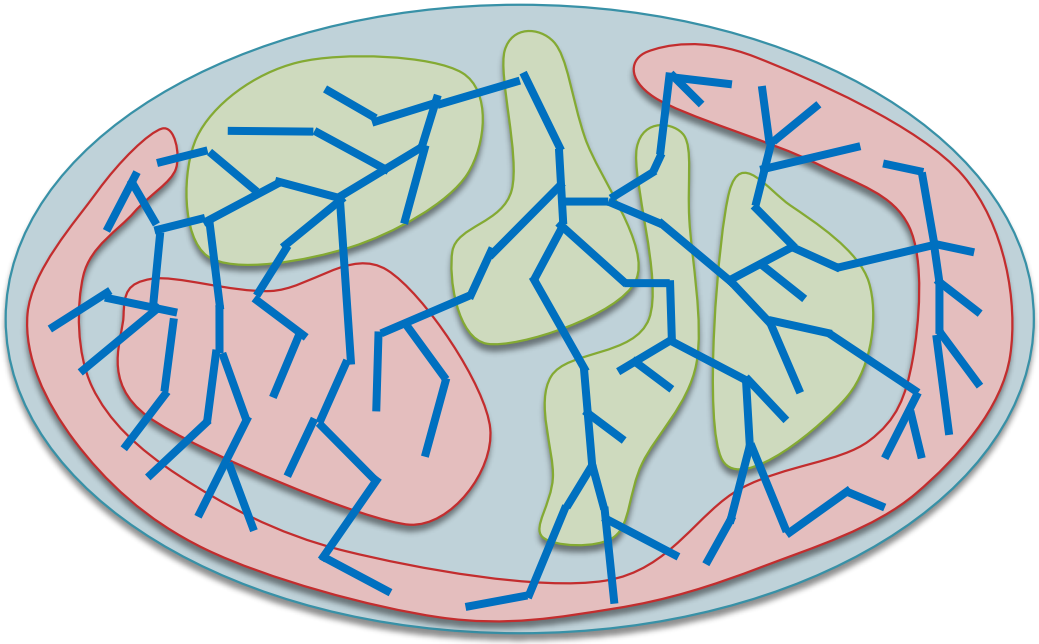
- $|f| \leq \sqrt{n}$：GHSに準じる
- $|f| > \sqrt{n}$：幅優先木でMOEを発見&更新



折衷的解決

- $|f| \leq \sqrt{n}$：GHSに準じる
- $|f| > \sqrt{n}$：幅優先木でMOEを発見&更新

→ 幅優先木の利用スケジュールが問題！



分散計算スケジューリング問題

- 入力：k個のタスク集合$\boldsymbol{T} = \{T_1, T_2, \cdots T_k\}$
  - $E_1, E_2, \ldots E_k$:通信パタン(=各タスク内でメッセージが伝送が生じる辺の(多重)集合)
  - $D_1, D_2 \ldots D_k$:各タスク(を単独で実行したときの)所要ラウンド数

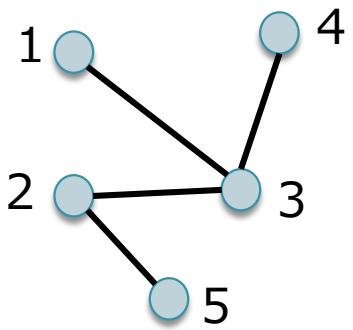
タスクT_iにおける辺eのcongestion =E_iでのeの多重度
(T_iは辺eに何回メッセージを送信するか)

$\boldsymbol{T}$のcongestion = $\max_e$($\sum_i T_i$におけるeのcongestion)

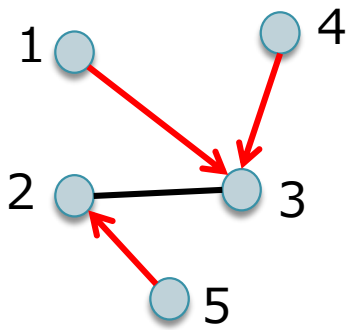
$\boldsymbol{T}$のdilation = $\max_i D_i$

分散計算スケジューリング問題

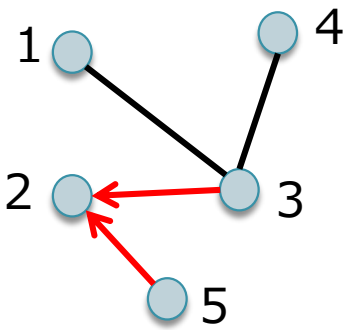
- 入力：k個のタスク集合$\boldsymbol{T} = \{T_1, T_2, \cdots T_k\}$
  - 個々のタスクはDAG(通信パタン)により定義される



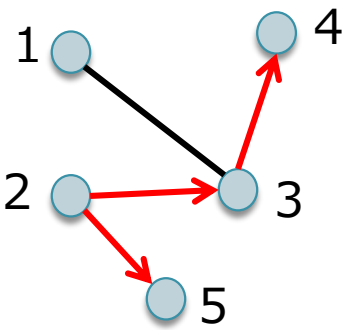
ネットワーク
トポロジ



ラウンド1



ラウンド2



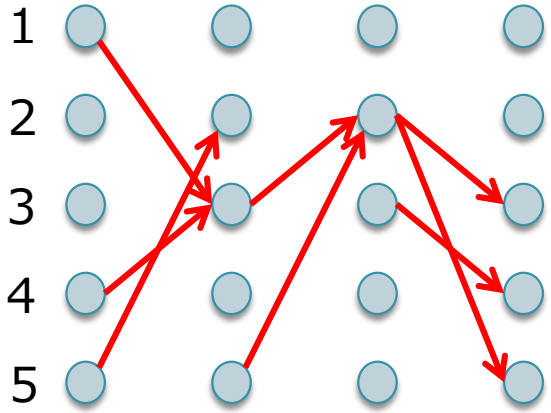
ラウンド3

タスクT_iにおける辺eのcongestion = eの利用回数

タスクT_iのdilation = T_iのラウンド数

$\boldsymbol{T}$のdilation = $\max_i T_i$のdilation

Tのcongestion = $\max_i T_i$のcongestion



分散計算スケジューリング問題

- MST(折衷案)の場合…
  - 個々のタスク：幅優先木上での収集or放送
  - タスク数は高々$\sqrt{n}$個
  - 個々のタスクはBFS木中の辺を高々1回利用

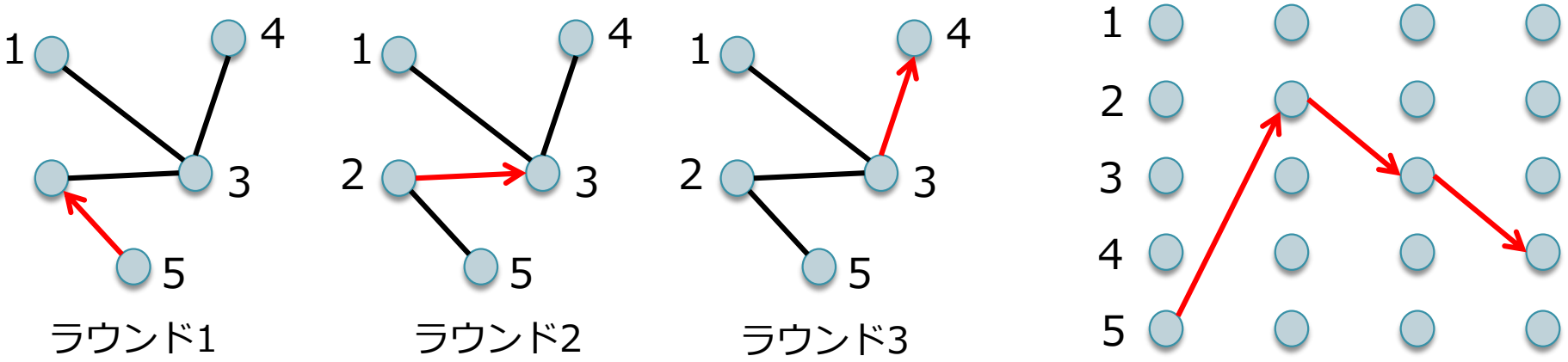
congestion = $O(\sqrt{n})$

  - 個々のタスクは$O(D)$時間で終了

dilation= $O(D)$

分散計算スケジューリング問題

- 分散計算のスケジューリングにおける最初期の結果の一つ
  - 個々のタスクをパケット転送タスクに限定



定理 [LMR94]
Congestion c, dilation dのパケット転送タスク集合を$O(c+d)$時間で終了させるスケジュールが存在(し，それを発見する乱択多項式時間アルゴリズムが存在する)

分散計算スケジューリング問題

- 近年一般の分散計算に対して拡張
  - さらに，スケジューリング自体も分散化可能に

定理 [Ghaffari15]
Congestion c, dilation dの任意のタスク集合$\boldsymbol{T}$を
$O(c + d \log^2 n)$時間で終了させる乱択分散アルゴリズムが存在

(より正確には，$O(d \log^2 n)$ラウンドの前処理で$O(c + d \log n)$ラウンドのスケジュールを生成)

  - この定理の適用において，各タスクの通信パタン，congestion, dilation等の情報は事前に知っておく必要なし

パケット転送スケジューリング問題

- ただし，今回はもう少し良い方法が使える

定理 [Topkis85, generalized by HIZ16-1]
$T_1, T_2, \ldots T_k$がすべてある共通の木上の収集/放送である
ようなCongestion c, dilation dのタスク集合$\boldsymbol{T}$を$O(c+d)$時間
で終了させる乱択分散アルゴリズムが存在

  - これはパイプライン式のメッセージ送信で容易に
    実現できる

Putting all together

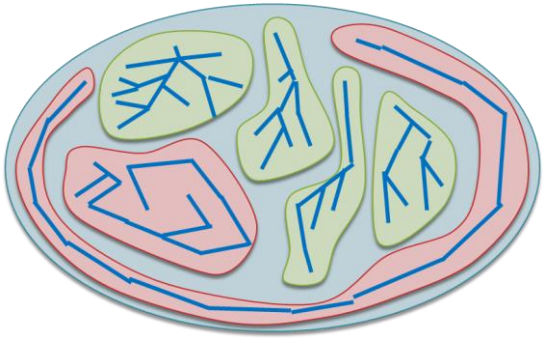
- $|f| \leq \sqrt{n}$: GHSに準じる $\rightarrow O(\sqrt{n})$ラウンド
- $|f| > \sqrt{n}$: 幅優先木でMOEを発見&更新 $\rightarrow O(\sqrt{n} + D)$ラウンド

定理 [GH16]
最小生成木問題を$\tilde{O}(\sqrt{n} + D)$ラウンドで解く
分散アルゴリズムが存在

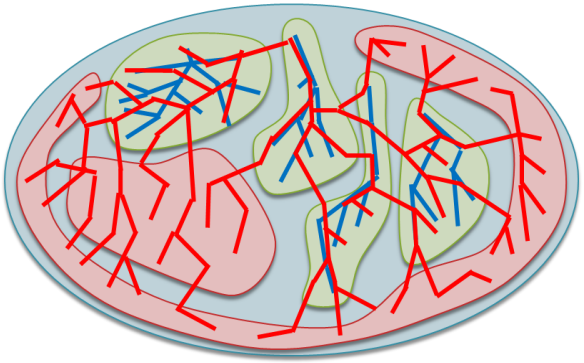
(注:$\tilde{O}(\sqrt{n} + D)$ラウンドで解く(別のアプローチでの)アルゴリズム自体はもっと昔に発見されている[KP98])

Congestion vs Dilation

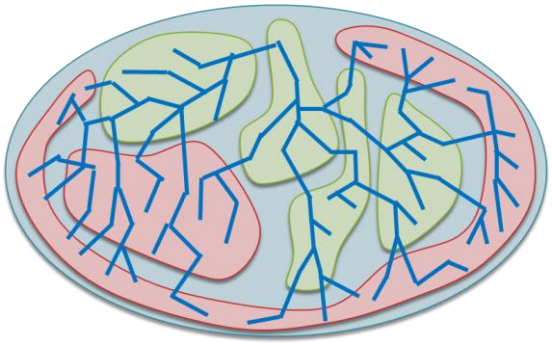
Congestion $O(1)$
Dilation $\Omega(n)$



Congestion $O(\sqrt{n})$
Dilation $0(\sqrt{n})$



Congestion $\Omega(n)$
Dilation $O(D)$

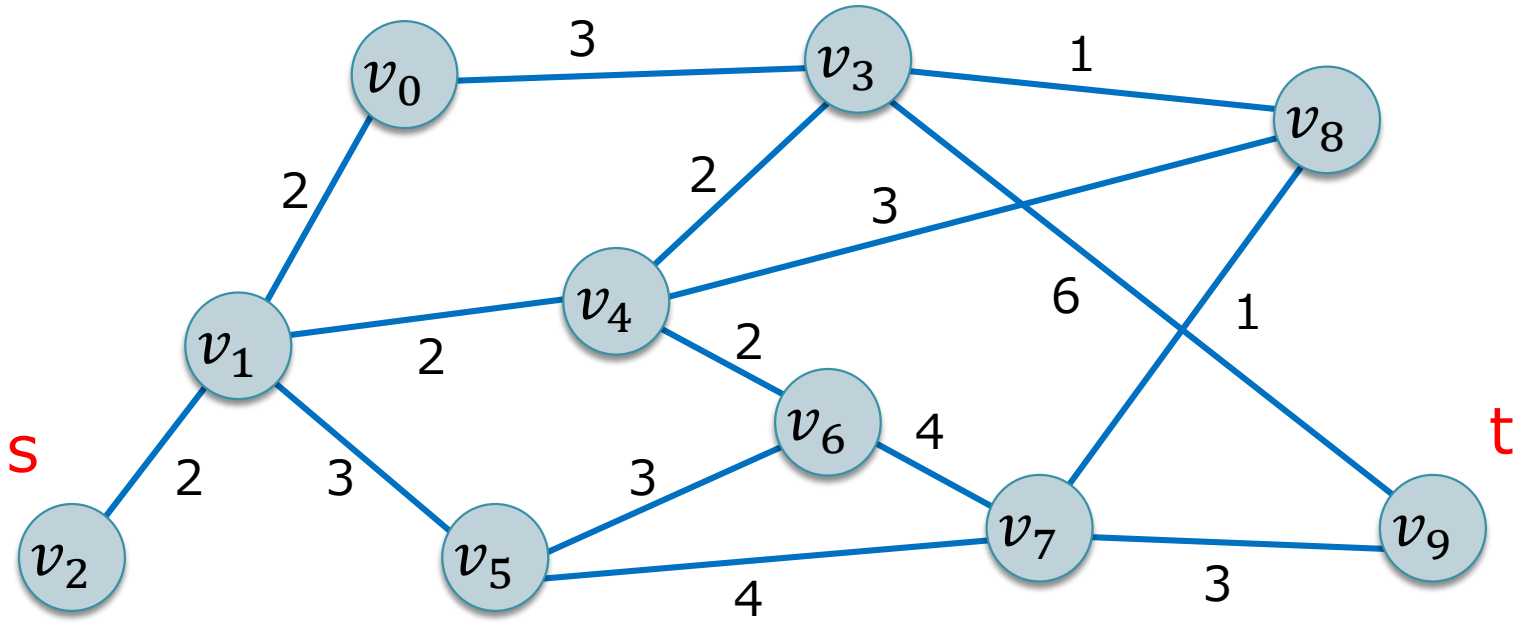


基本的な戦略：CongestionとDilationをバランスさせて$o(n)$時間アルゴリズムを得る

PRAMと大きく違うところ…かも

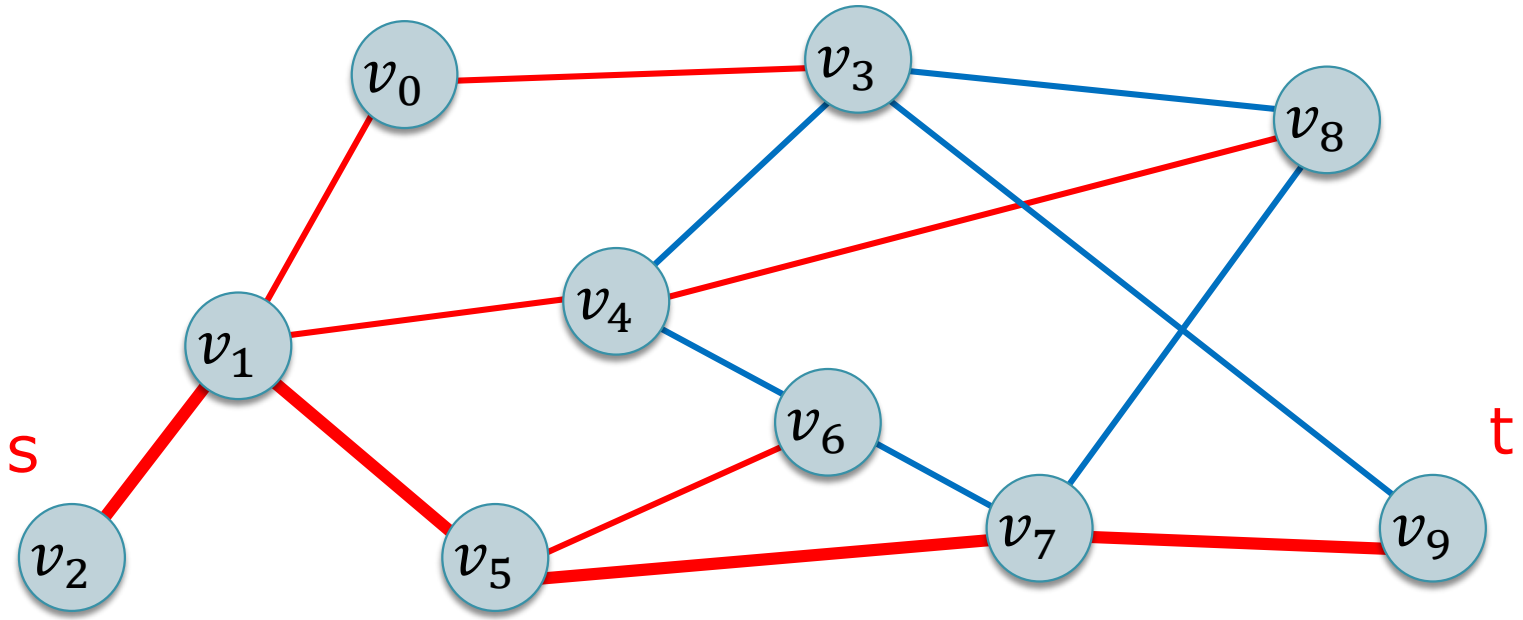
別の例：最短経路近似

- 重み付きグラフのs-t最短パスを計算(近似)したい



別の例：最短経路近似

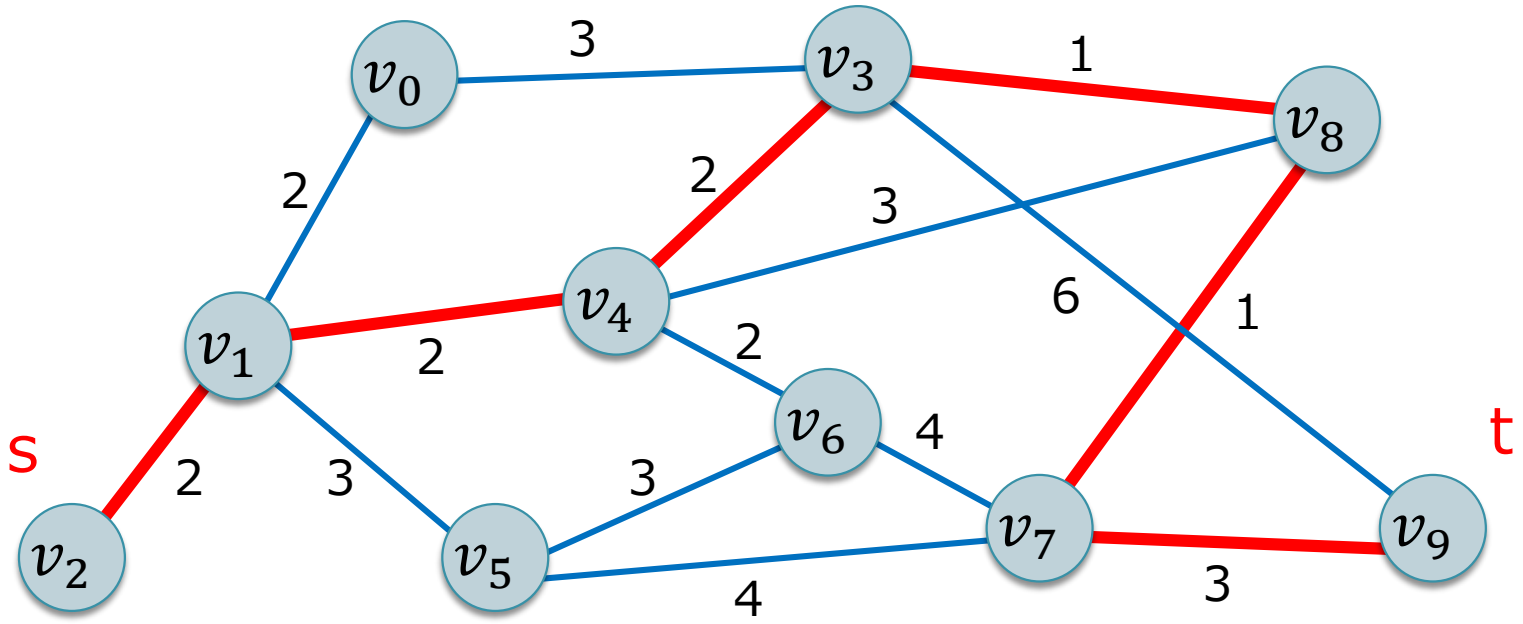
- 重み付きグラフのs-t最短パスを計算(近似)したい
  - 重みなしの場合はBFSで楽勝($O(D)$ラウンド)



別の例：最短経路近似

- 重み付きグラフのs-t最短パスを計算(近似)したい
  - 問題：最短パスのホップ長は$\Omega(n)$になり得る

⟶ Dijkstra的なアプローチを素直に適用するだけでは遅そう
(そもそも適用できるかどうかも分からないが)



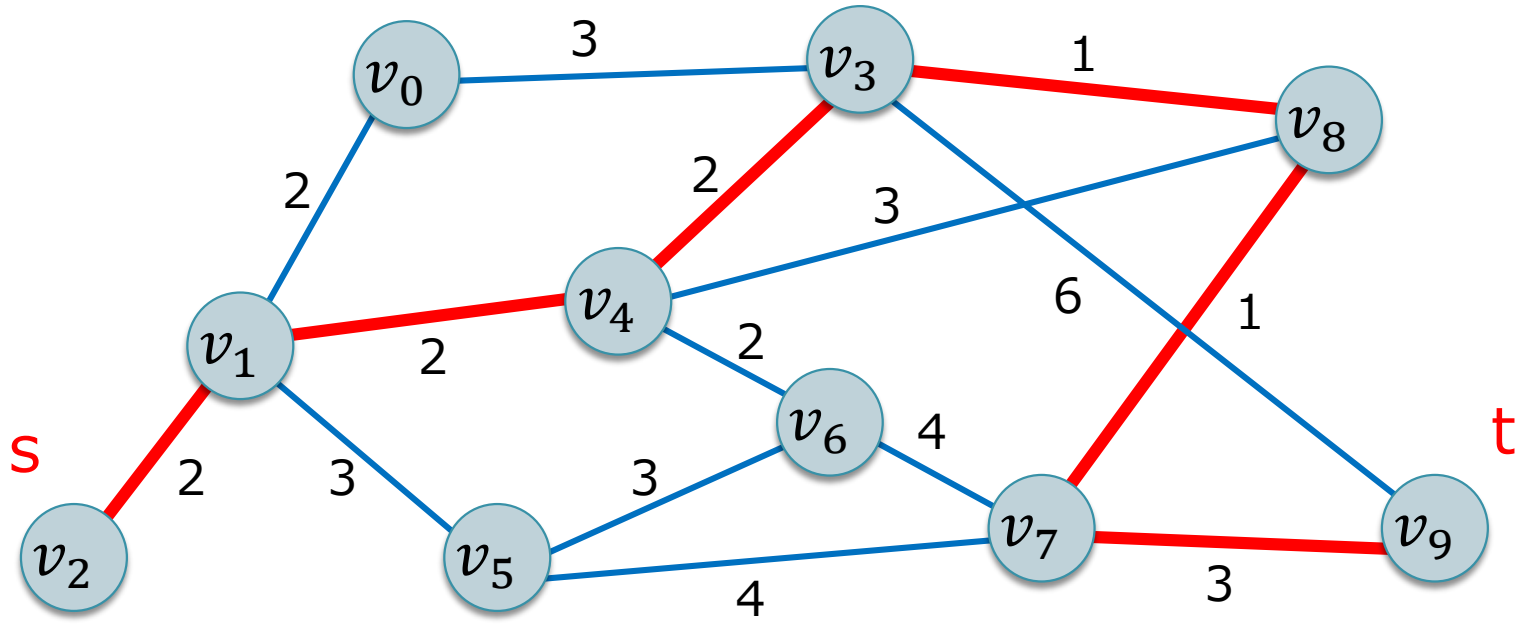
別の例：最短経路近似

- 重み付きグラフのs-t最短パスを
  - 問題：最短パスのホップ長は

Dilation = $\Omega(n)$

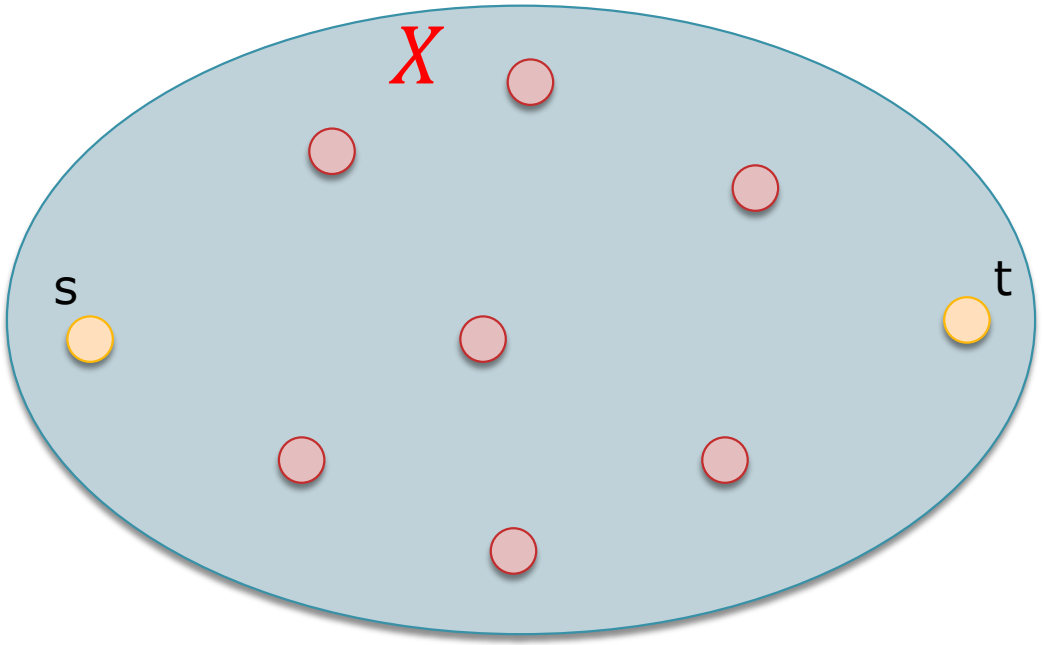
これをcongestionに
トレードしたい

⟶ Dijkstra的なアプローチを素直に適用するだけでは遅そう
(そもそも適用できるかどうかも分からないが)



アプローチ(概要)

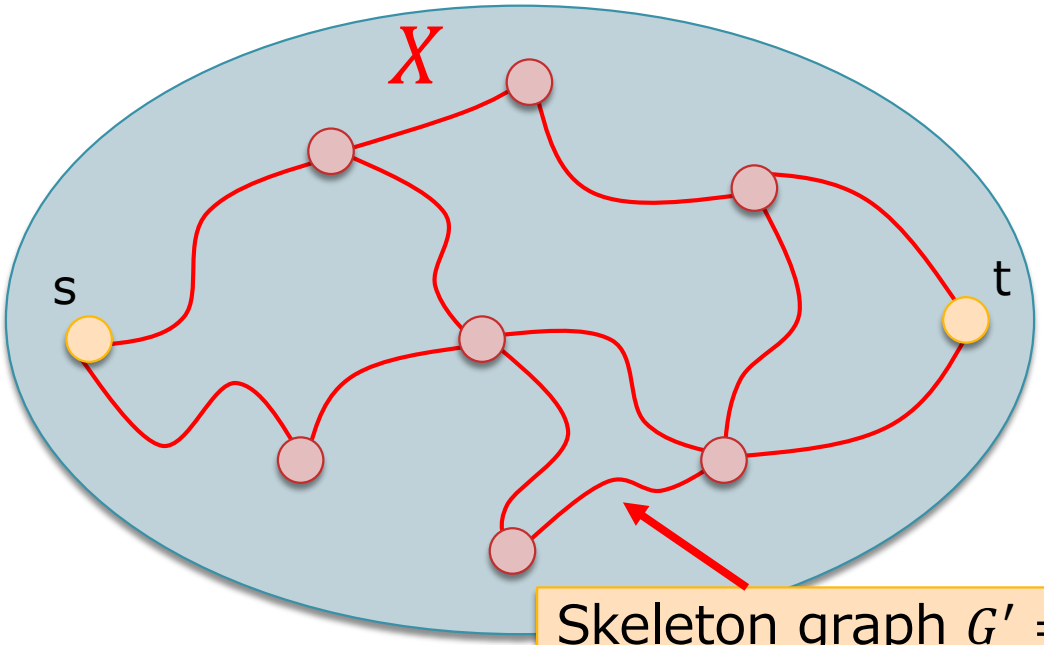
1. $\tilde{\Theta}(n^{\frac{1}{3}})$個のノードをランダムサンプリングして$X$とする



アプローチ(概要)

1. $\tilde{\Theta}(n^{\frac{1}{3}})$個のノードをランダムサンプリング($X$とする)

2. s, t, X中の頂点は最短パスのホップ数が$O(n^{\frac{2}{3}})$以下のノードに対し最短経路を同時に(近似)計算



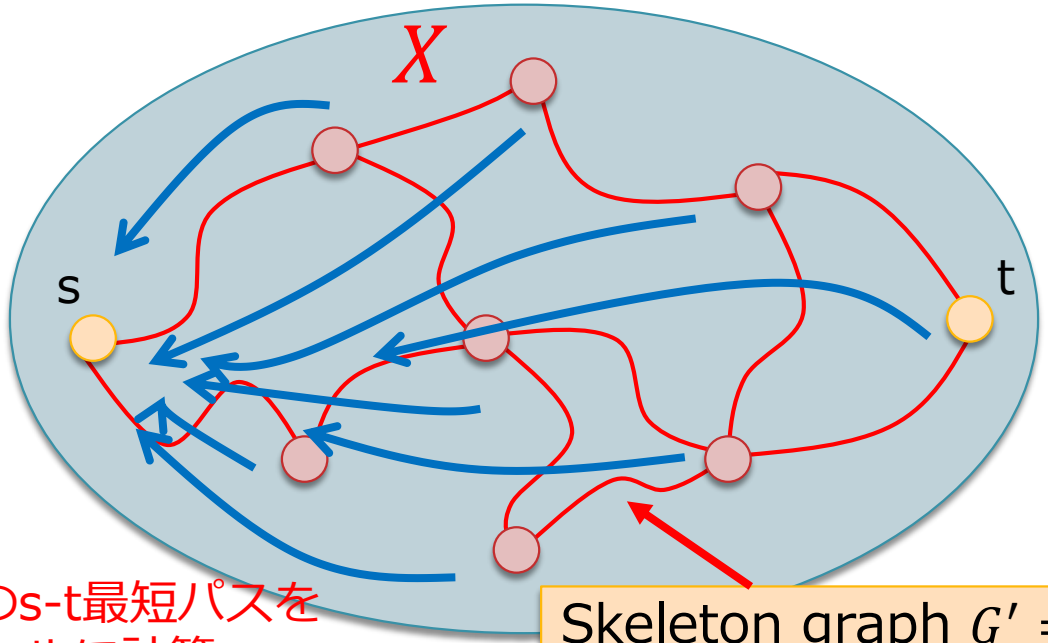
Skeleton graph $G' = (\{s, t\} \cup X, E', w')$

$(u, v) \in E' \Leftrightarrow (u, v)$間の最短パスのホップ数$\leq n^{\frac{2}{3}}$

$w'(u, v) = u, v$ 間の(近似)最短距離

アプローチ(概要)

3. Skeleton graph G'を作ってしまえば後は集中的に解ける
   - G'のサイズ=頂点数$\tilde{\Theta}(n^{\frac{1}{3}})$なので全体で$\tilde{O}(n^{\frac{2}{3}})$ビット
   - 収集に要する時間：$\tilde{O}(n^{\frac{2}{3}} + D)$ ラウンド(BFS木上で収集)



G'上のs-t最短パスを
ローカルに計算

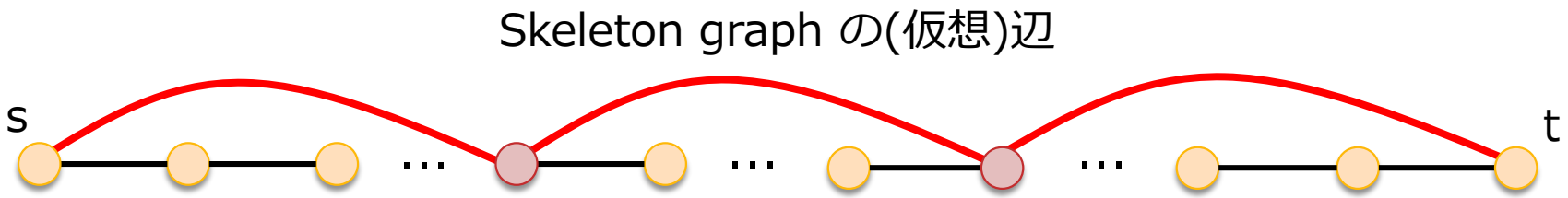
Skeleton graph $G' = (\{s, t\} \cup X, E', w')$

$(u, v) \in E' \Leftrightarrow (u, v)$間の最短パスのホップ数$\leq n^{\frac{2}{3}}$

$w'(u, v) = u, v$ 間の(近似)最短距離

性能の評価(解の質)

- 解の質は，Skeleton graphの各辺の近似率により決まる



  - s-t最短パスのホップ数が大きいとき，X中の頂点がおよそ$\tilde{\Theta}(n^{\frac{2}{3}})$ホップ毎に現れて中継する

→ G'の各辺が$(1+\epsilon)$近似されていれば，G'上のs-t最短距離は真の最短距離の$(1+\epsilon)$近似となる

性能の評価(時間)

- ランダムサンプリング: ローカルに行う(1ラウンド)
- G'の構成：$\tilde{\Theta}(n^{\frac{1}{3}})$個の単一始点最短経路アルゴリズム(ホップ数を$\tilde{\Theta}(n^{\frac{2}{3}})$に制限した版)を並列に実行
- G'の収集：既に述べた通り，$\tilde{\Theta}(n^{\frac{2}{3}})$ラウンド

ここをどう解決する？

性能の評価(時間)

補題 [Nanongkai14]
任意の定数$\epsilon > 0$に対し，ホップ数をkに制限した単一始点最短経路問題をcongestion $= \tilde{O}(1)$, dilation $= O(k)$ で解く$(1+\epsilon)$近似アルゴリズムが存在

- [Ghaffari15]のスケジューリング手法と組み合わせるとG'の構成は$\tilde{\Theta}(n^{\frac{2}{3}})$ラウンドで行える

定理 [Nanongkai14 を弱めた版]
任意の定数$\epsilon > 0$に対し，s-t最短距離の$(1+\epsilon)$-近似解を求める$\tilde{\Theta}(n^{\frac{2}{3}})$ラウンドアルゴリズムが存在

各種問題の時間計算量(上界)

- MST:$\tilde{O}(\sqrt{n}+D)$ラウンド，厳密
- 単一始点最短経路: $\tilde{O}(\sqrt{n}D^{\frac{1}{4}}+D)$ ラウンド，$(1+\epsilon)$-近似
- 最小カット： $\tilde{O}(\sqrt{n}+D)$ラウンド，$(1+\epsilon)$-近似
- 最大フロー：$\tilde{O}((\sqrt{n}+D)n^{o(1)})$ラウンド，$(1+\epsilon)$-近似
- 直径(重みなし)：$\tilde{O}(\sqrt{n}+D)$ラウンド，3/2-近似
- 直径(重みあり)：$\tilde{O}(\sqrt{n}+D)$ラウンド，2-近似

その他いろいろ

疑問

$\tilde{\Theta}(\sqrt{n})$でのトレードオフは本質的か？

- 答：Yes

定理 [DHK+12, FHW12, LP13]
以下の問題群は$\tilde{\Omega}(\sqrt{n})$ラウンドの計算時間下界を持つ
(たとえ$D = O(\log n)$であっても)

(Approx.) MST, (Approx.) weighted minimum cut, (Approx.) min s-t cut/maxflow, (Approx.) weighted shortest path, $(2-\epsilon)$-approx. unweighted diameter, (Approx.) weighted diameter, subgraph (s-t) connectivity, subgraph cycle detection, (Approx.) Generalized Steiner forest, ...

(2者間)通信複雑性

- Alice と Bobが nビットデータ列x, yを保有

- 問題：関数$f: \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$が与えられたとき，$f(x, y)$の計算に何ビットの通信が必要か？
- AliceとBobの計算能力は無限
- 複数ラウンド使っても良い
- プロトコルが乱択のときはϵ-誤りが仮定されることが多い

大雑把な定義

- プロトコルPの入力(x, y)，ランダムビット列r_1, r_2に対する通信量
  - r_1, r_2を使ったときの，プロトコルの停止までに送信されたメッセージの総ビット数　(実際は決定木を用いて形式的に定義される)

(ランダムビット列に関して平均を取る定義もあるが，ϵ-誤りを考える時はあまり差はない)

$$R_P^{MAX}(f) = \max_{(x,y)\in X\times Y, r\in\{0,1\}^*} C_P(x, y)$$

$$f\text{の}(\epsilon - \text{誤り})\text{通信複雑性} = \min_{P\in\mathcal{P}_\epsilon} R_P^{MAX}(f)$$

問題の例

- 交叉判定（set-disjointness）

Communication Link

アイテム集合を保有
{1,3,6,7}
⇕ビットベクトル表現
$x = (10100110)$

アイテム集合を保有
{2,4,5}
⇕ビットベクトル表現
$y = (01011000)$

2人が共通のアイテムを持つか？
（$=x,y$のともにビット1が立っているような場所があるか）

問題の例

- 等価性判定（equality）

Communication Link

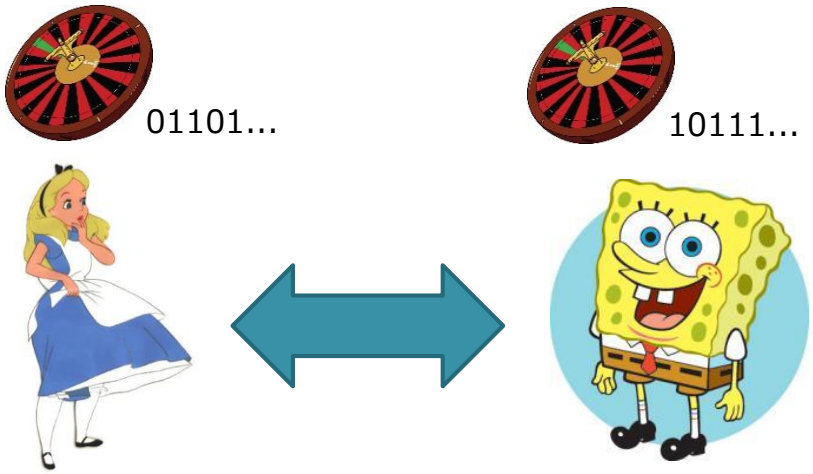
$x = (10100110)$

$y = (10100110)$

$x = y$か?

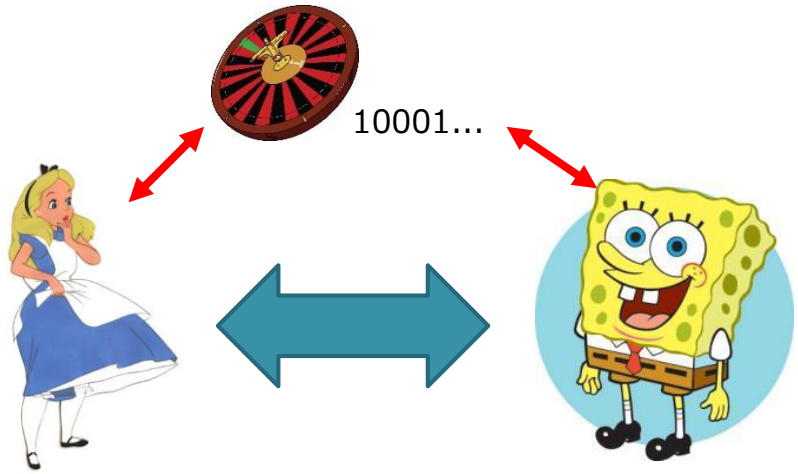
Private vs Public Randomness

- 乱択プロトコルにおける２つのモデル



Private Randomness

AliceとBobがそれぞれ独立に
乱数生成器を持つ



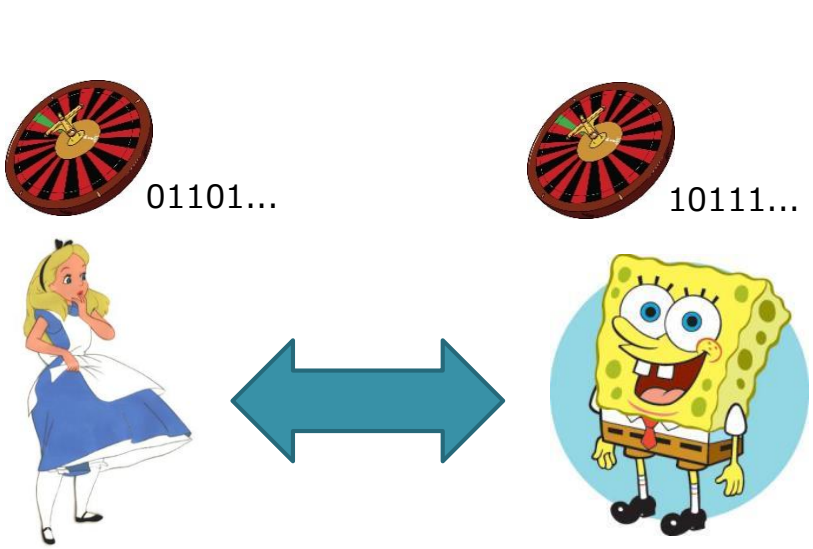
Public Randomness

AliceとBobがどちらも(コストゼロで)
アクセスできる共通の乱数生成器を持つ

※乱数生成器＝無限長のランダム0-1列

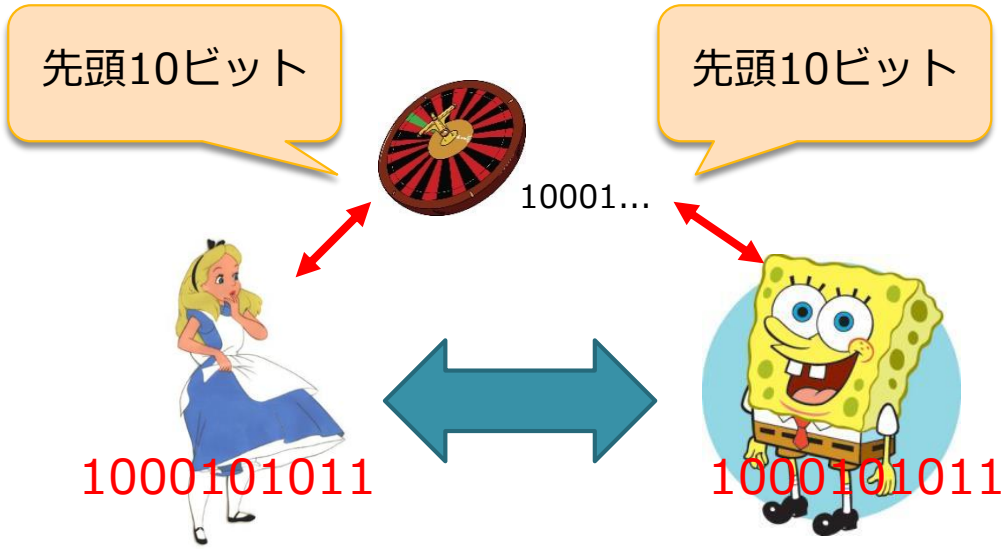
Private vs Public Randomness

- 乱択プロトコルにおける２つのモデル



Private Randomness

AliceとBobがそれぞれ独立に
乱数生成器を持つ



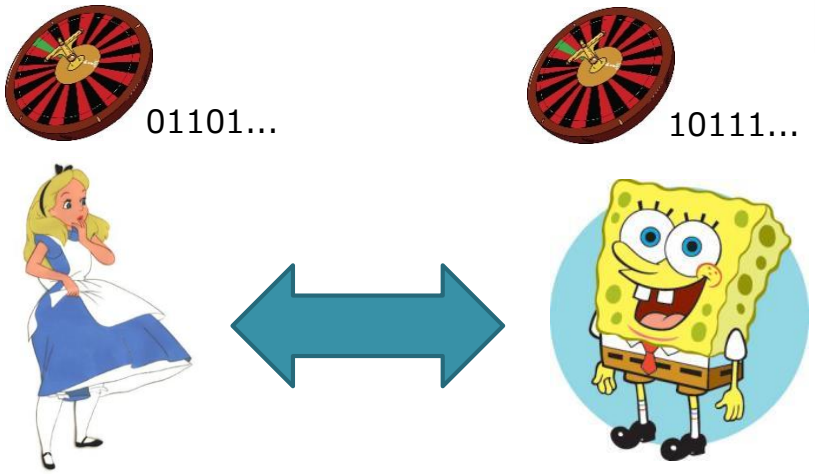
Public Randomness

AliceとBobがどちらも(コストゼロで)
アクセスできる共通の乱数生成器を持つ

※乱数生成器＝無限長のランダム0-1列

Private vs Public Randomness

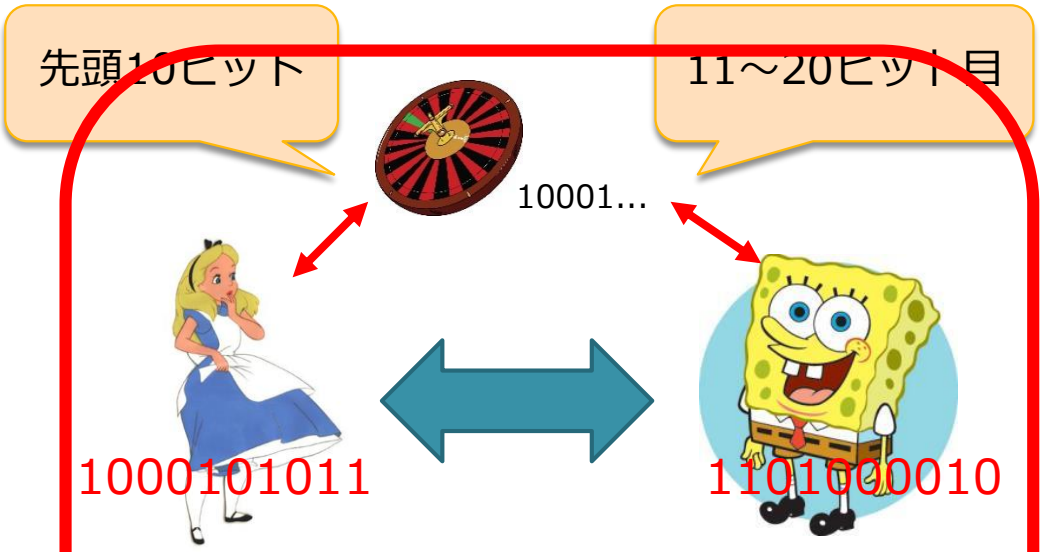
- 乱択プロトコルにおける２つのモデル



Private Randomness

AliceとBobがそれぞれ独立に
乱数生成器を持つ

もちろんこちらのほうが強力
(プロトコルの設計が容易)



Public Randomness

AliceとBobがどちらも(コストゼロで)
アクセスできる共通の乱数生成器を持つ

※乱数生成器＝無限長のランダム0-1列

交叉判定の通信複雑性

- 以降の議論には，とりあえず以下の事実を知っていればOK

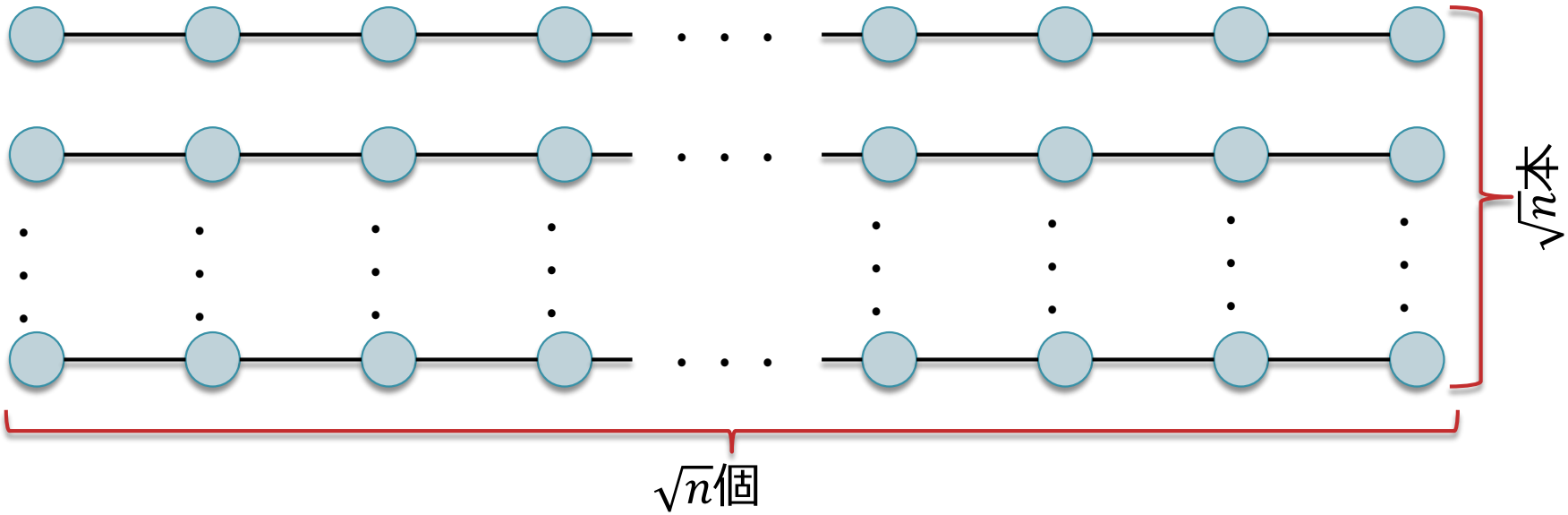
定理 [KC87, Razborov90]

Public Randomnessモデル上でn-ビット交叉判定を行う任意のϵ-誤りプロトコルの通信複雑性が$\Omega(n)$ビットとなるような定数$\epsilon > 0$が存在する

- 交叉判定を有意に高い確率で解こうと思うと(漸近的には)自明なプロトコルが最適

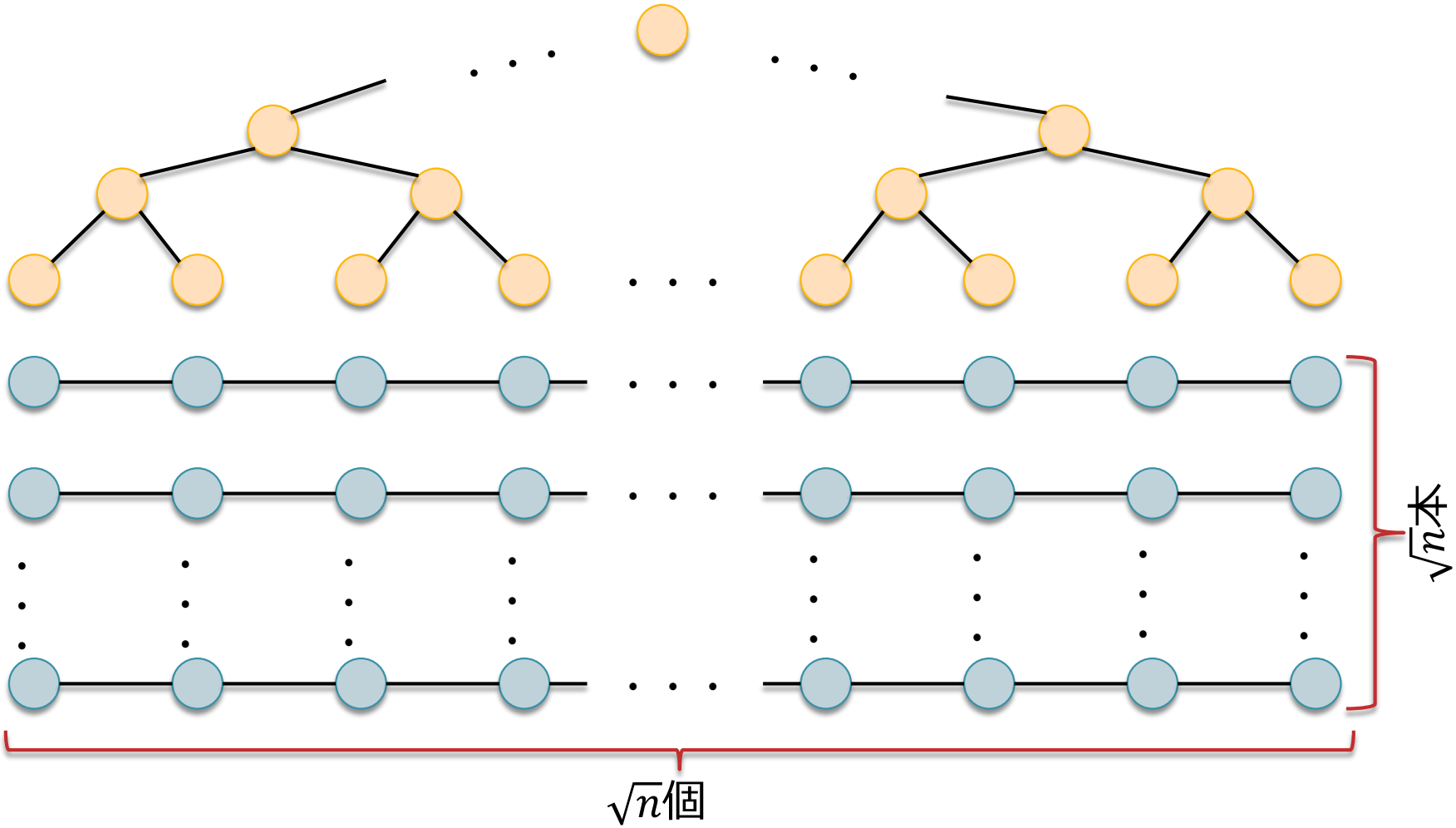
Hard-Core Topology G_{LB}

- ステップ1：長さ$\sqrt{n}$のパスを$\sqrt{n}$本用意する



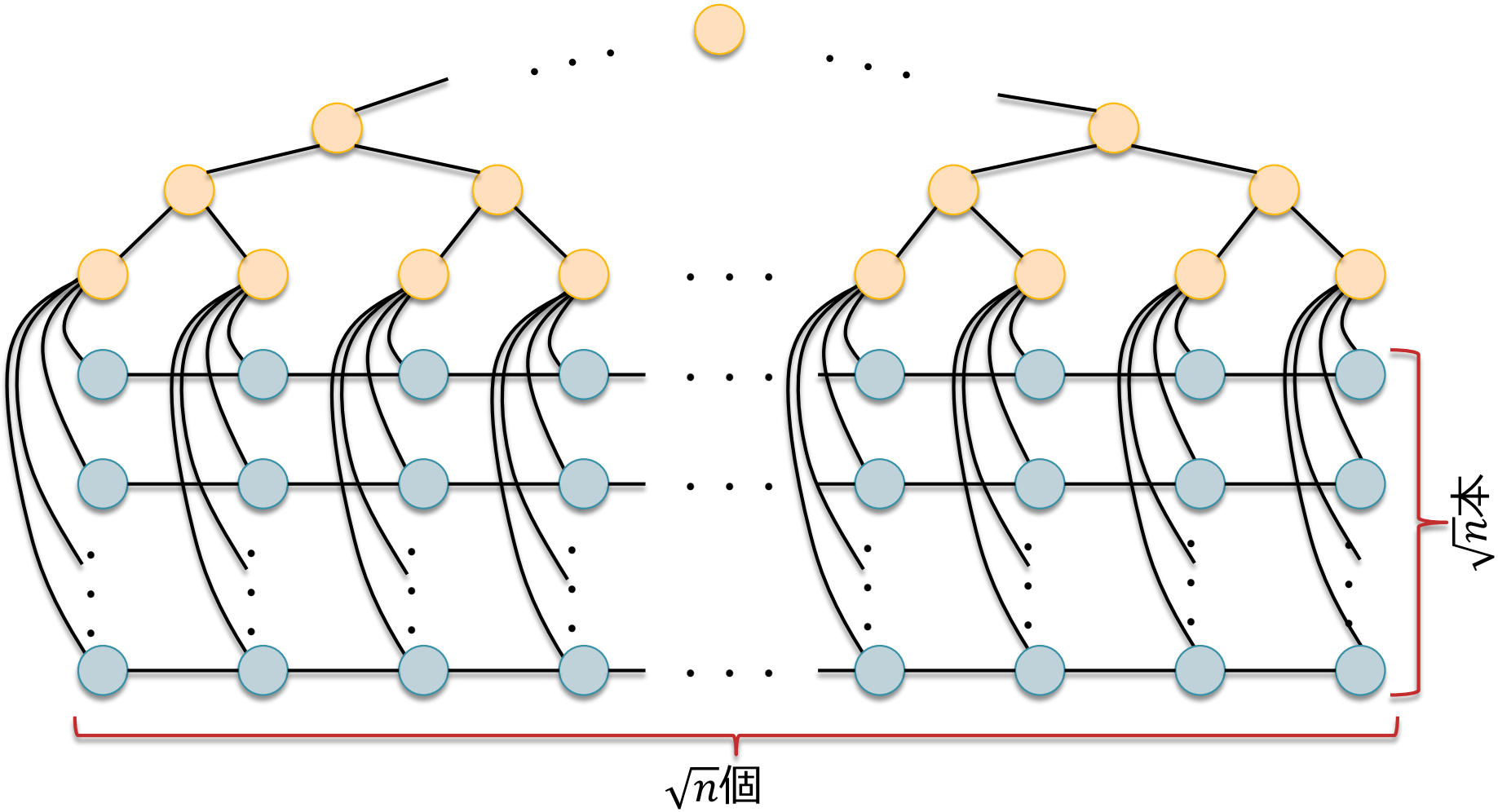
Hard-Core Topology G_{LB}

- ステップ２：葉数$\sqrt{n}$の完全二分木を用意
  (頂点数> nになるが，漸近的には$O(n)$なのであまり気にしないように)



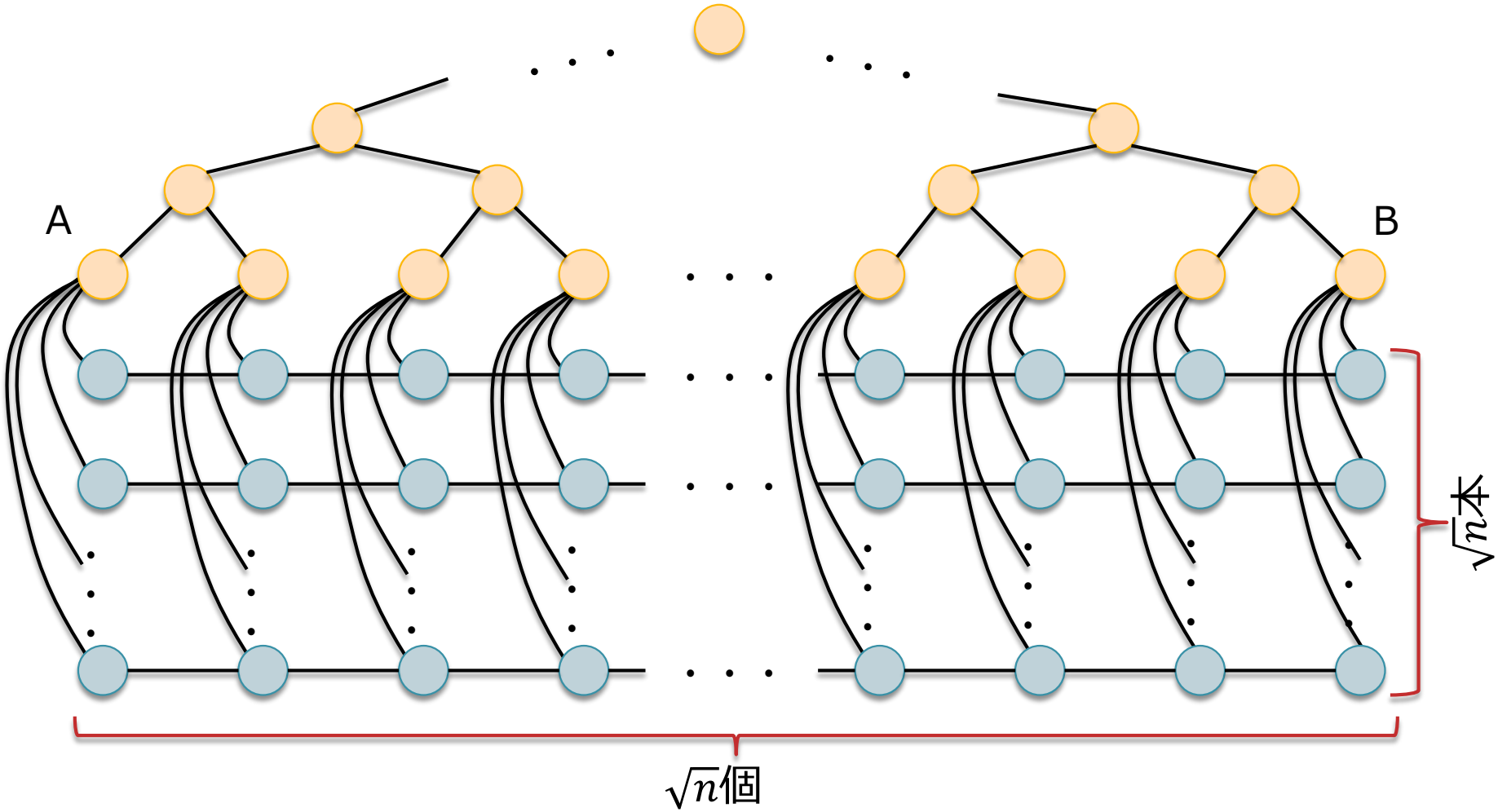
Hard-Core Topology G_{LB}

- ステップ3：パス中のi番目の頂点とi番目の葉を接続



Hard-Core Topology G_{LB}

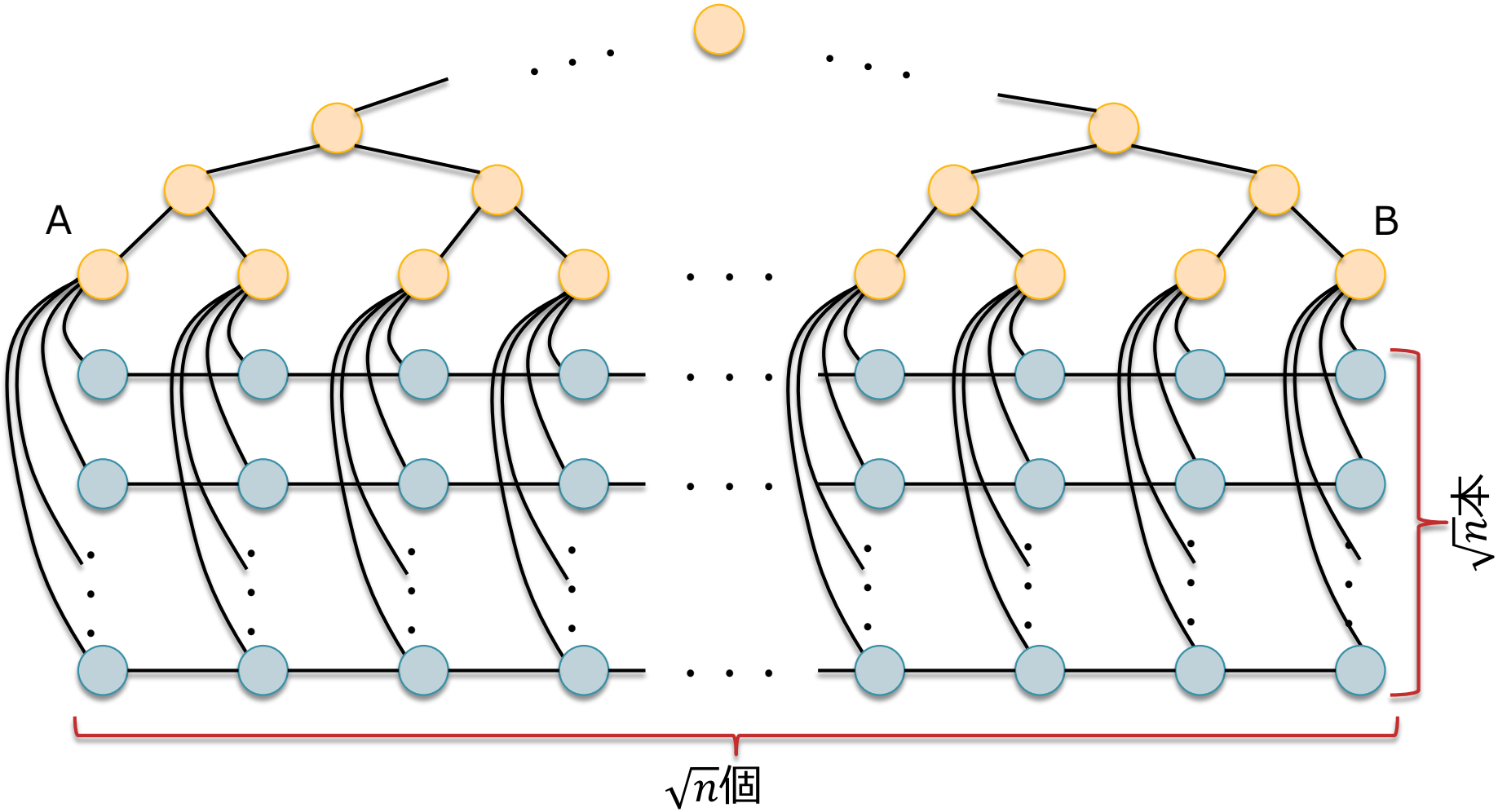
- 一番左の葉をA, 一番右の葉をBとする



H

定理 [PR00, DHK+12]

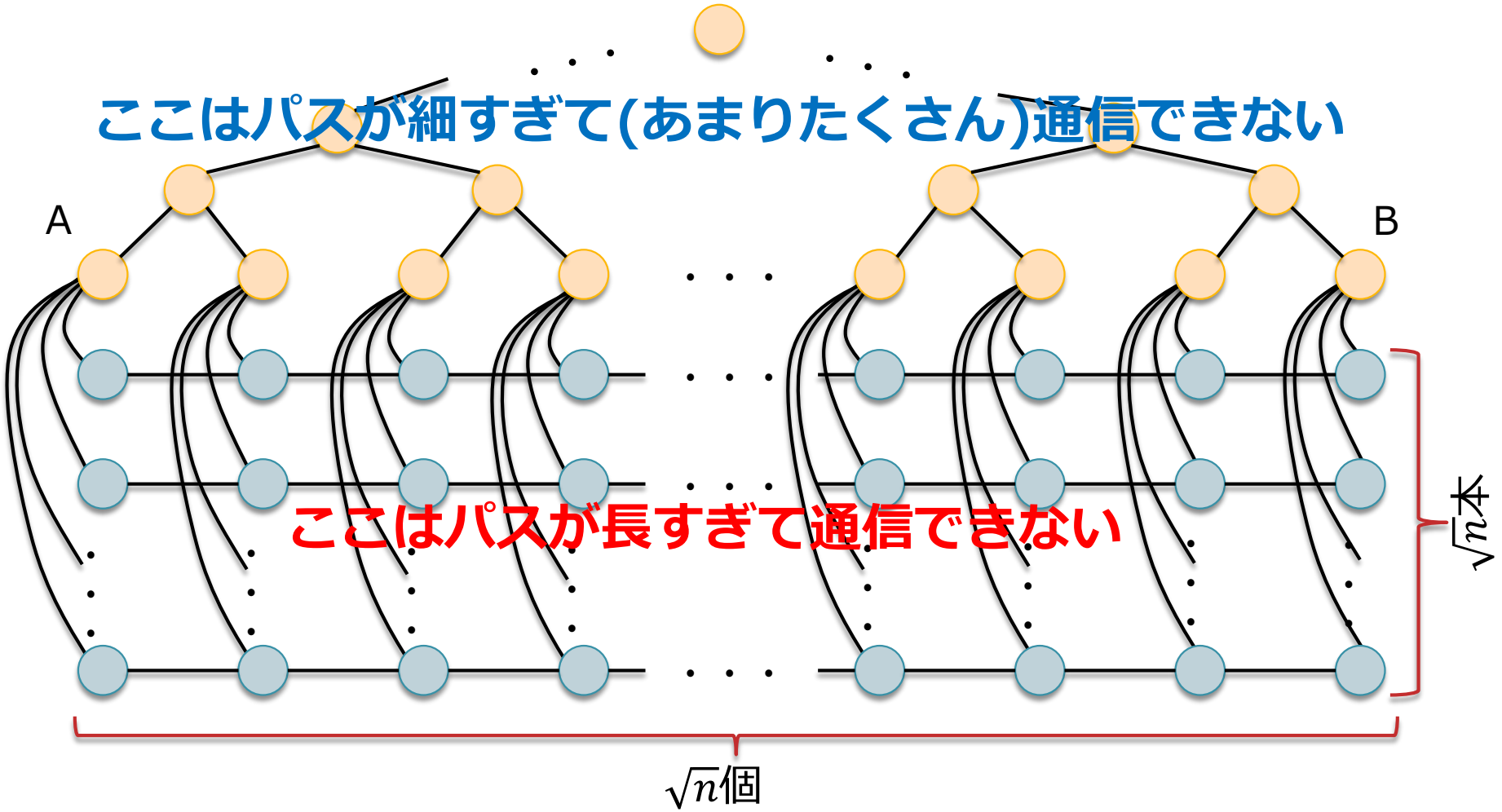
任意のkラウンド($k < \frac{\sqrt{n}}{2}$)のアルゴリズムにおいて
A-B間は高々$k \log^2 n$ ビットしか通信ができない



H

定理 [PR00, DHK+12]

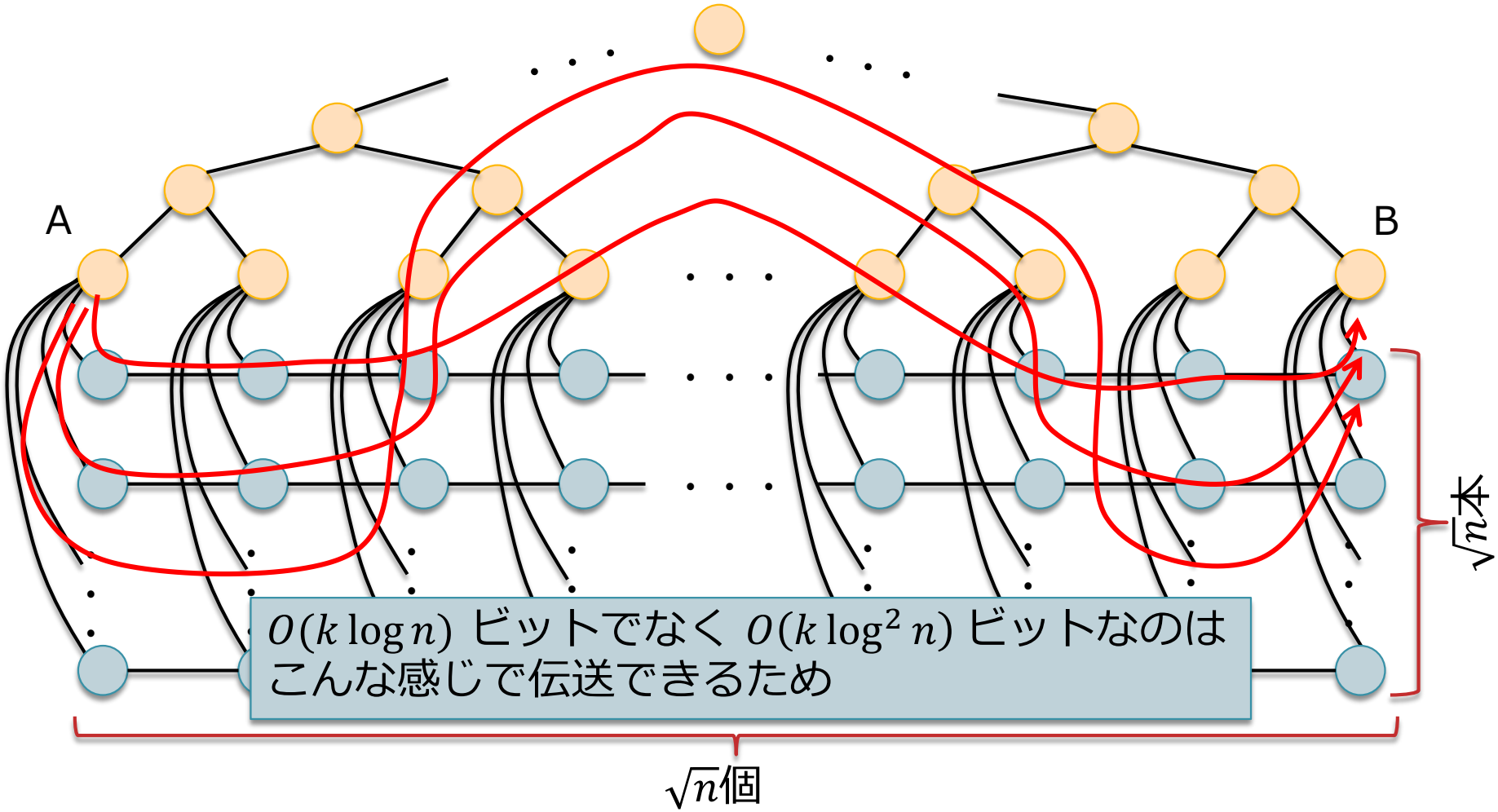
任意のkラウンド($k < \frac{\sqrt{n}}{2}$)のアルゴリズムにおいて
A-B間は高々$k \log^2 n$ ビットしか通信ができない



H

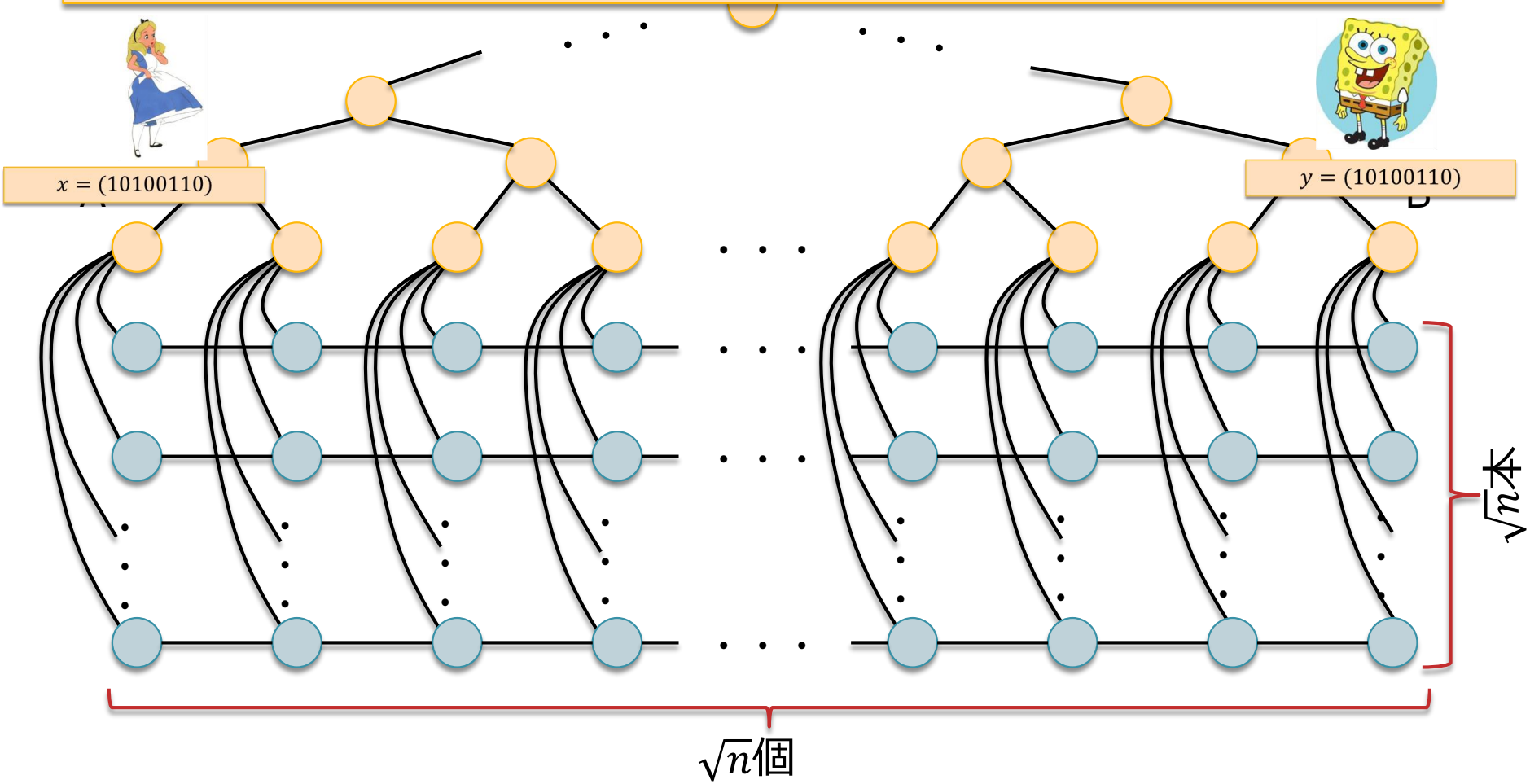
定理 [PR00, DHK+12]

任意のkラウンド($k < \frac{\sqrt{n}}{2}$)のアルゴリズムにおいて
A-B間は高々$O(k \log^2 n)$ ビットしか通信ができない



H系 [DHK+12]

- Aが$\sqrt{n}$ビットの入力xを持ち，Bが$\sqrt{n}$ビットの入力yを持つとするとき，$disj(x,y)$を$\tilde{o}(\sqrt{n})$ラウンドで計算するアルゴリズムは(乱択/決定性問わず)存在しない



系に関する補足

- この系はA, B以外の他ノードはx, yに関する情報以外すべてを知っているとしても成立

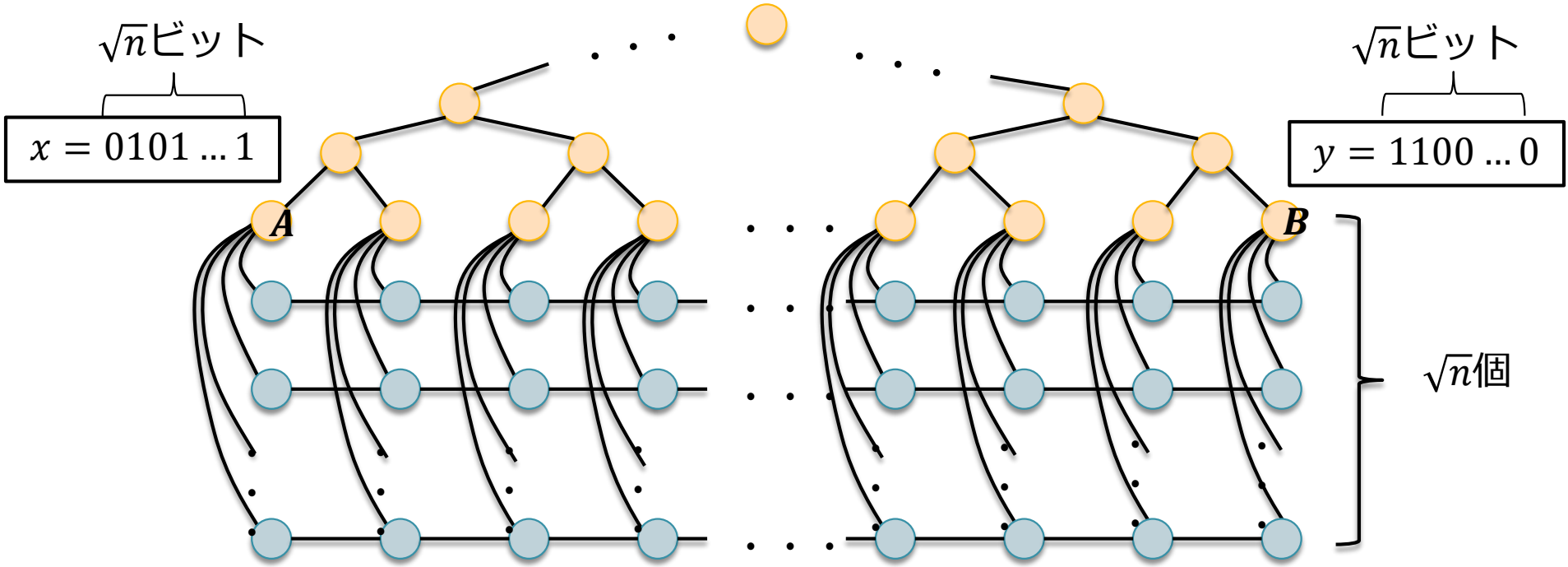
トポロジ全体，その中での自分の位置

⇔ G_{LB}に特化したアルゴリズムをもってしても$\tilde{o}(\sqrt{n})$ラウンドで交叉判定を解くことが不可能

帰着による下界証明

- set-disjointness → (Approx.) MST

やること：MSTを解くAlg.を使って(G_{LB}に特化した)交叉判定Alg.を構成

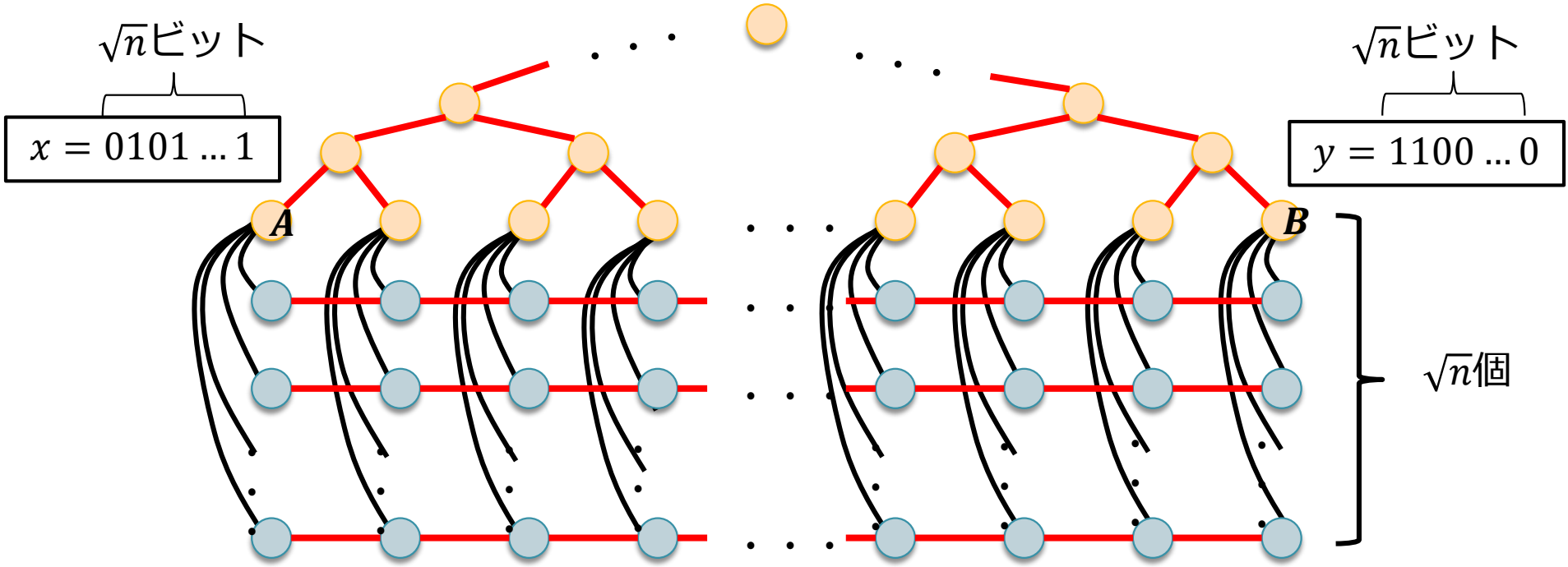


帰着による下界証明

重み1
重みn^{100}

- set-disjointness → (Approx.) MST

やること：MSTを解くAlg.を使って(G_{LB}に特化した)交叉判定Alg.を構成



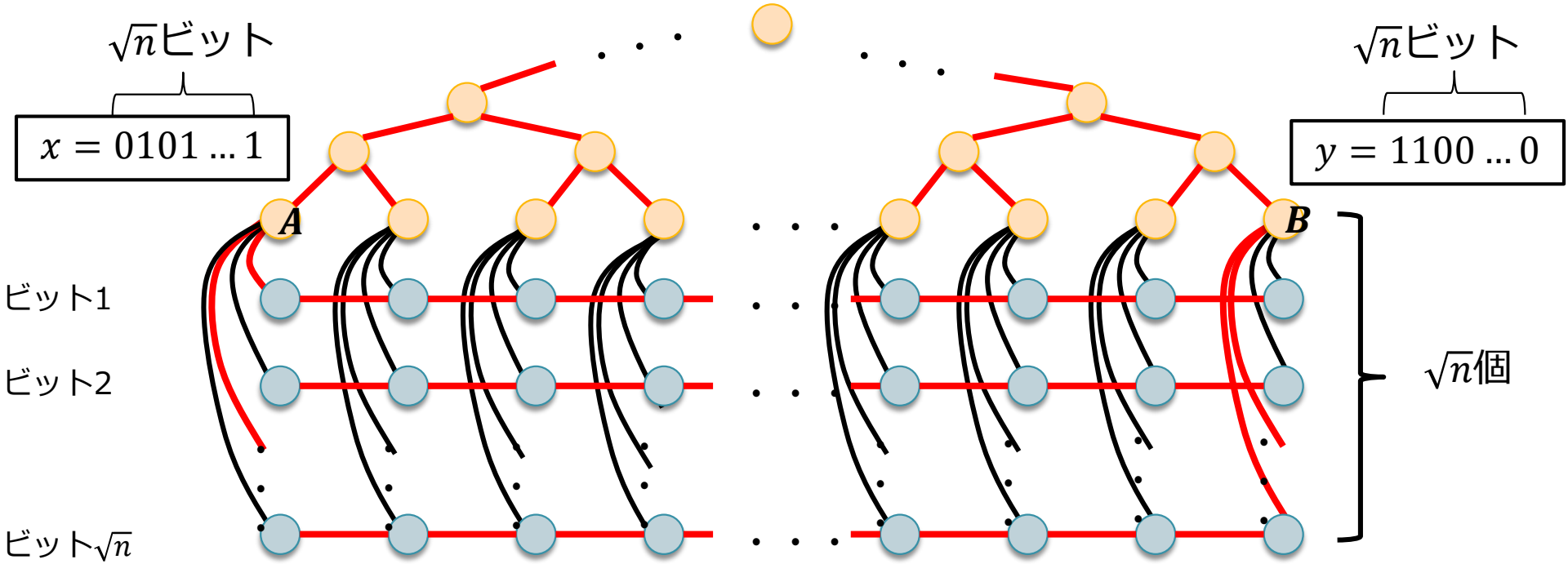
0. 上記黒色の辺の重み=n^{100} 赤色辺の重み=1とする

帰着による下界証明

重み1
重みn^{100}

- set-disjointness → (Approx.) MST

やること：MSTを解くAlg.を使って(G_{LB}に特化した)交叉判定Alg.を構成

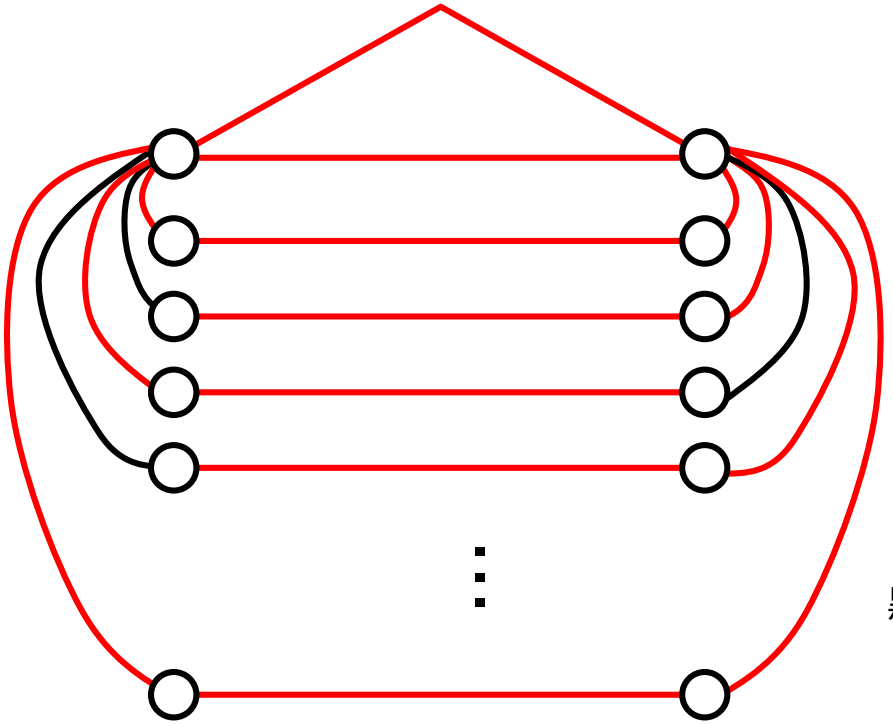


1. パスの各段を交叉判定問題の各ビットに対応付ける
2. A/Bからi段目への辺の重み $= \begin{cases} 1 & (x\text{の}i\text{ビット目} = 0) \\ n^{100} & (\text{それ以外}) \end{cases}$

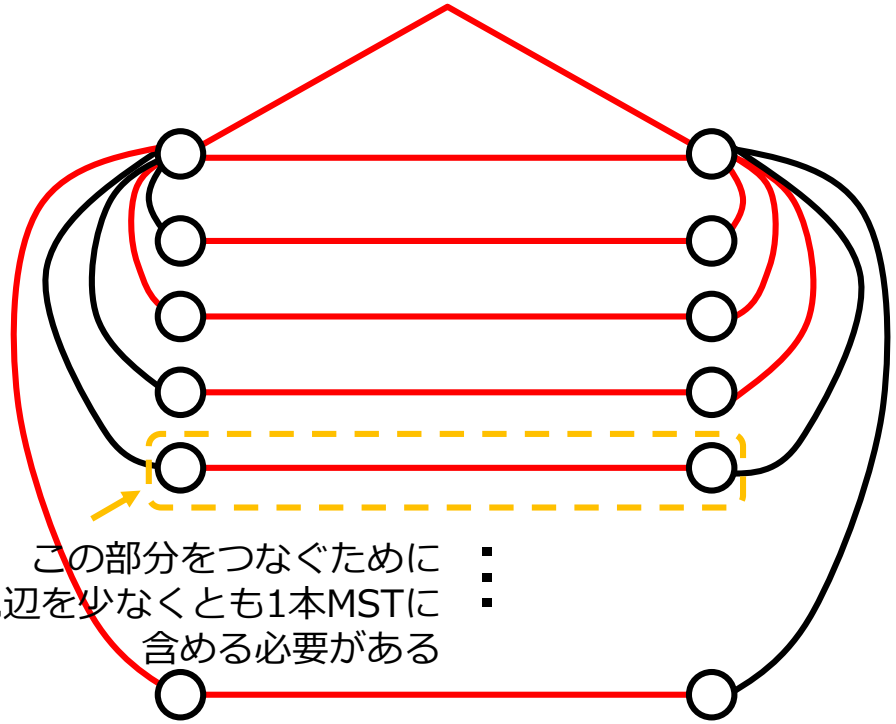
帰着による下界証明

重み1
重みn^{100}

- x, yが交叉⇔MSTがn^{100}重みの辺を含む



互いに素なとき



交叉するとき

ビット0に相当　ビット1に相当

帰着に要するコスト

- 辺重みの設定：1ラウンド
  - A,Bに接続する辺はxの各ビット情報を隣接ノードに送信
  - それ以外の辺の重みは各頂点が初期状態で設定
    - 今はG_{LB}に特化したアルゴリズムを構成しているので各頂点が自身の隣接辺に割り当てるべき重みは既知(=アルゴリズムにハードコーディングする)

定理 [DHK+12]
MSTの重みがn以下 or $n-1+n^{100}$以上を$f(n)$ラウンドで判定するアルゴリズムが存在
⇒ $f(n)+1$ラウンドでG_{LB}上の交叉判定が解ける

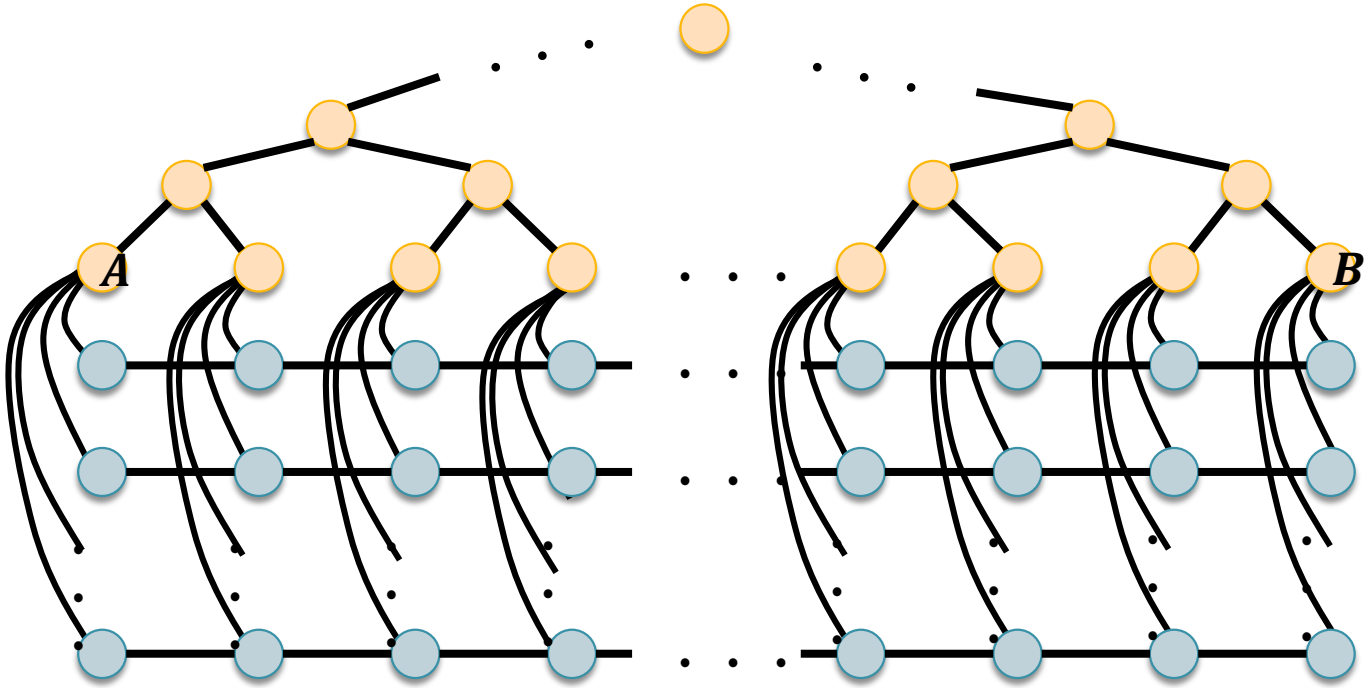
注)もちろんn^{100}は便宜的であり，実際は好きな値を取れる

最悪時インスタンスの回避

- トポロジG_{LB}はある意味で「悪魔的な」インスタンス
- ネットワークトポロジに何らかの制限を
  置くことでより高速に問題が解けないか？
  - 疎なグラフ，密なグラフ
  - 次数が制限されたグラフ
  - 直径が小さいグラフ
  - 連結度が高いグラフ
  - その他，代表的なグラフクラス
    (平面的，部分k-木, etc...)

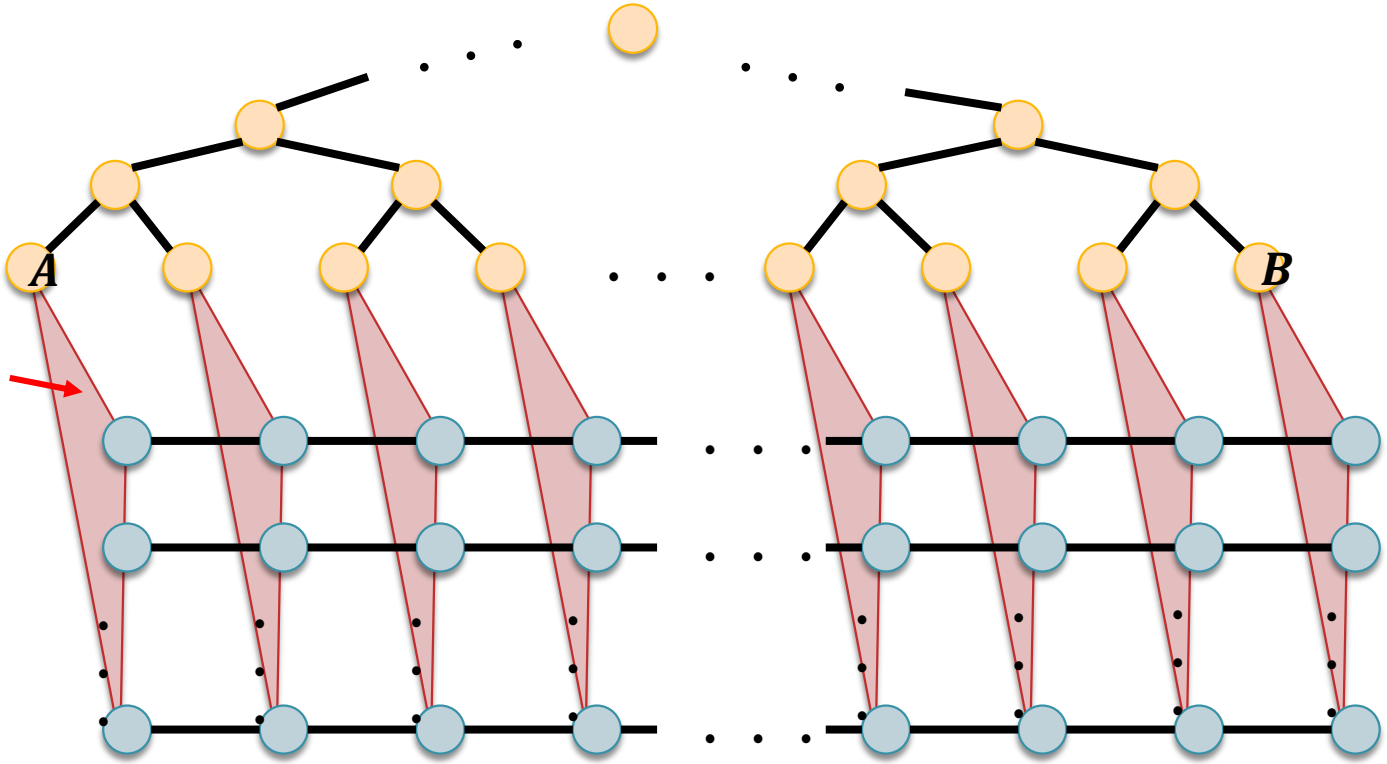
下界の検討

- 疎なグラフ：×
  - G_{LB}は既に疎である



下界の検討

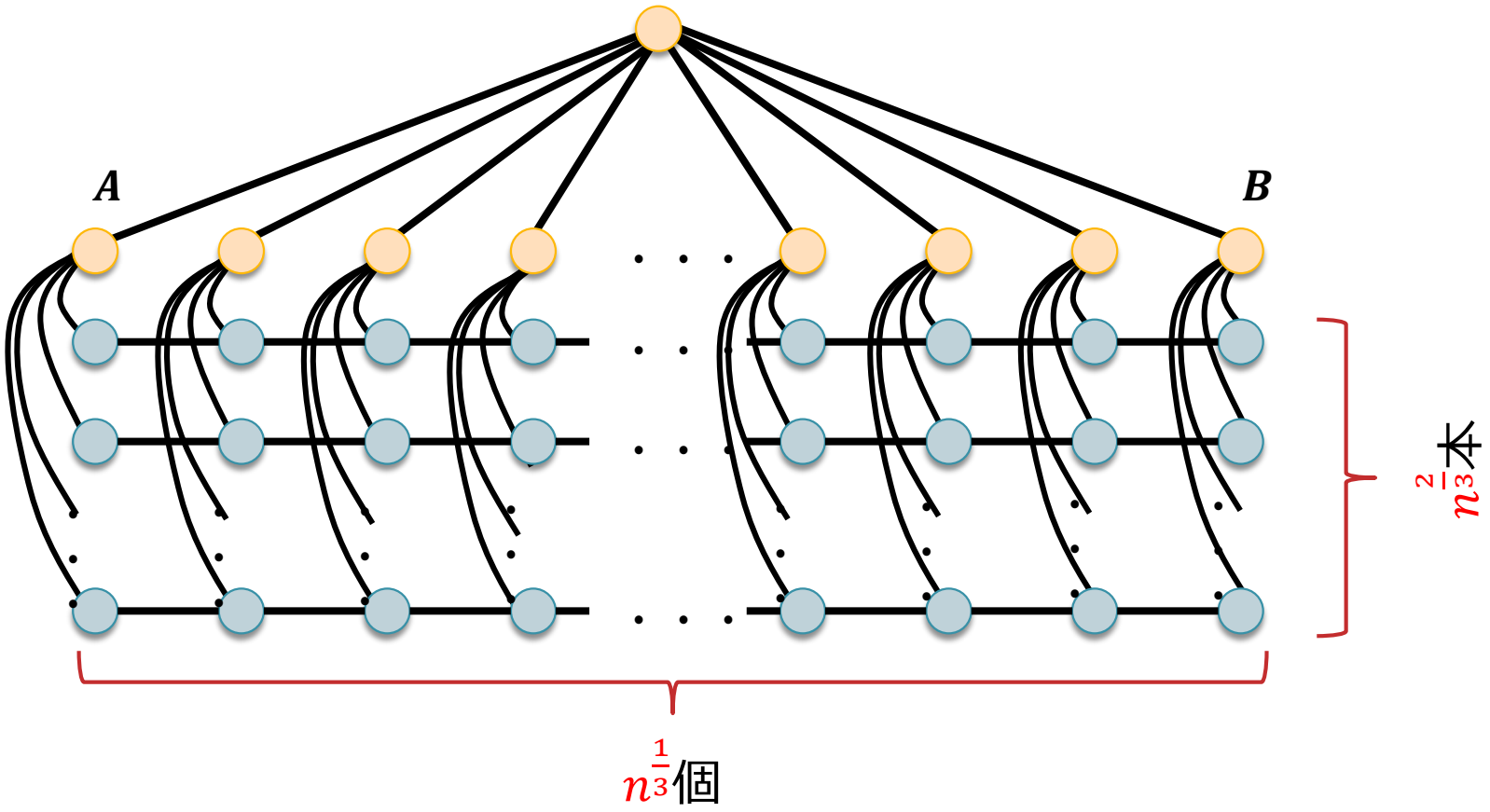
- 次数を制限：×
  - G_{LB}を以下のように変形しても元と同様の下界が得られる



最大次数3

下界の検討

- 直径小：△
  - 以下のようにアレンジすると直径は4になる[LPP06]

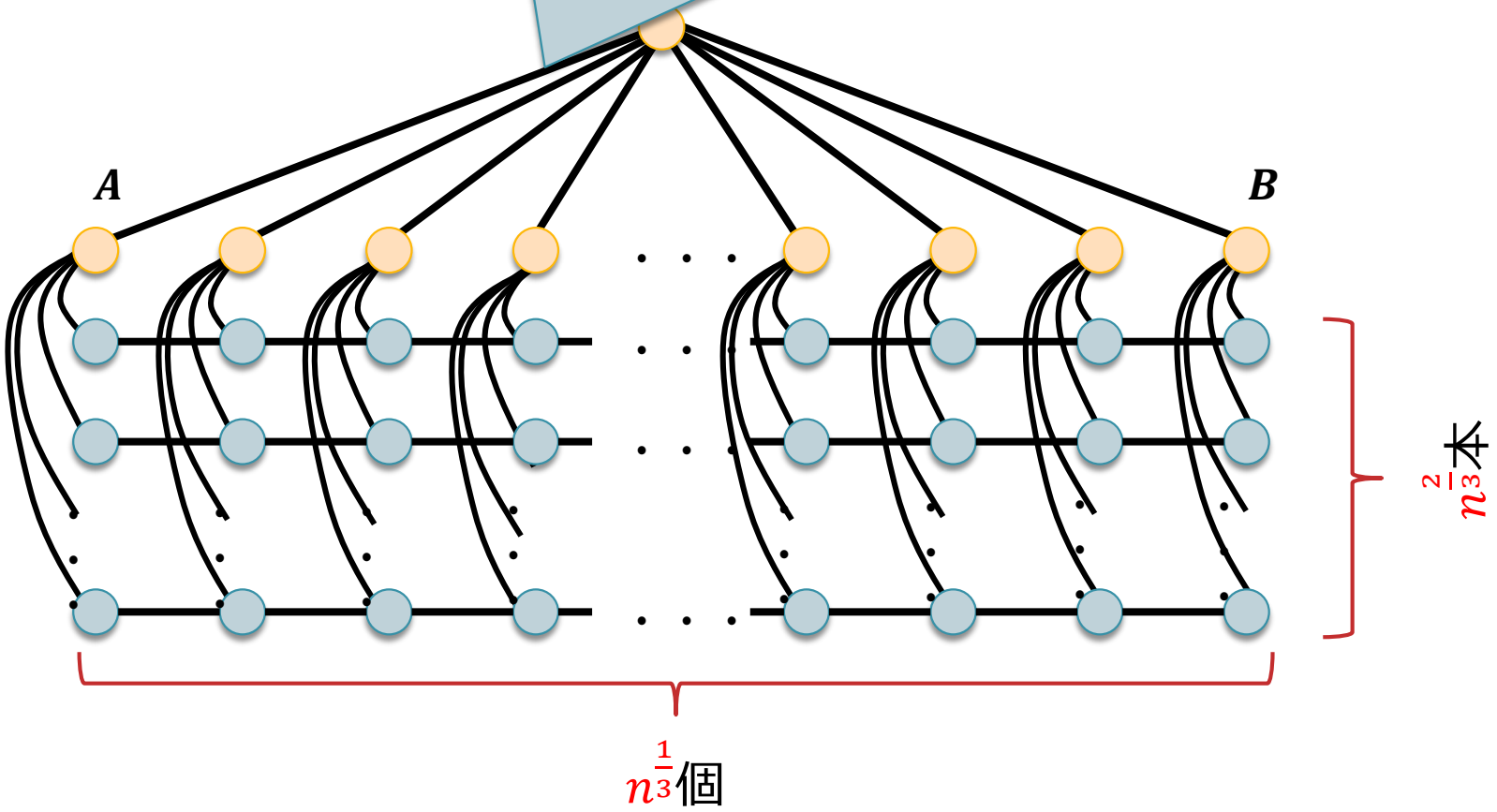


下界の検討

- 直径小：△
  - 以下のように

最初のkラウンド($k < n^{\frac{1}{3}}$)で，A-B間は$\tilde{O}(kn^{\frac{1}{3}})$ビットしかメッセージを伝送できない

前述の問題群に対して$\tilde{\Omega}(n^{\frac{1}{3}})$ ラウンドの下界が得られる



下界の検討

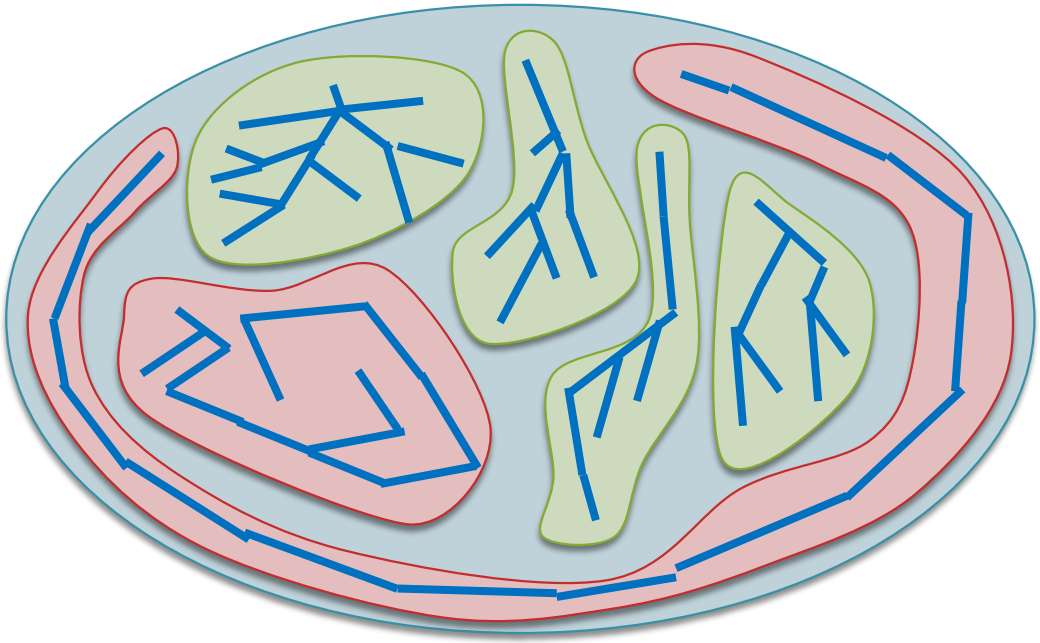
- その他の仮定
  - 平面的グラフ：少なくともG_{LB}はまったく平面的ではない
  - 部分k木：少なくともG_{LB}の木幅は小さくない
  - 密，連結度高い：少なくともG_{LB}は以下略

- これらは基本的には有用な仮定

定理 [GH16, HIZ16-1,HIZ16-2]
平面的グラフ，木幅定数のグラフにおいてMSTは
$\tilde{O}(D)$ラウンドで構成可能

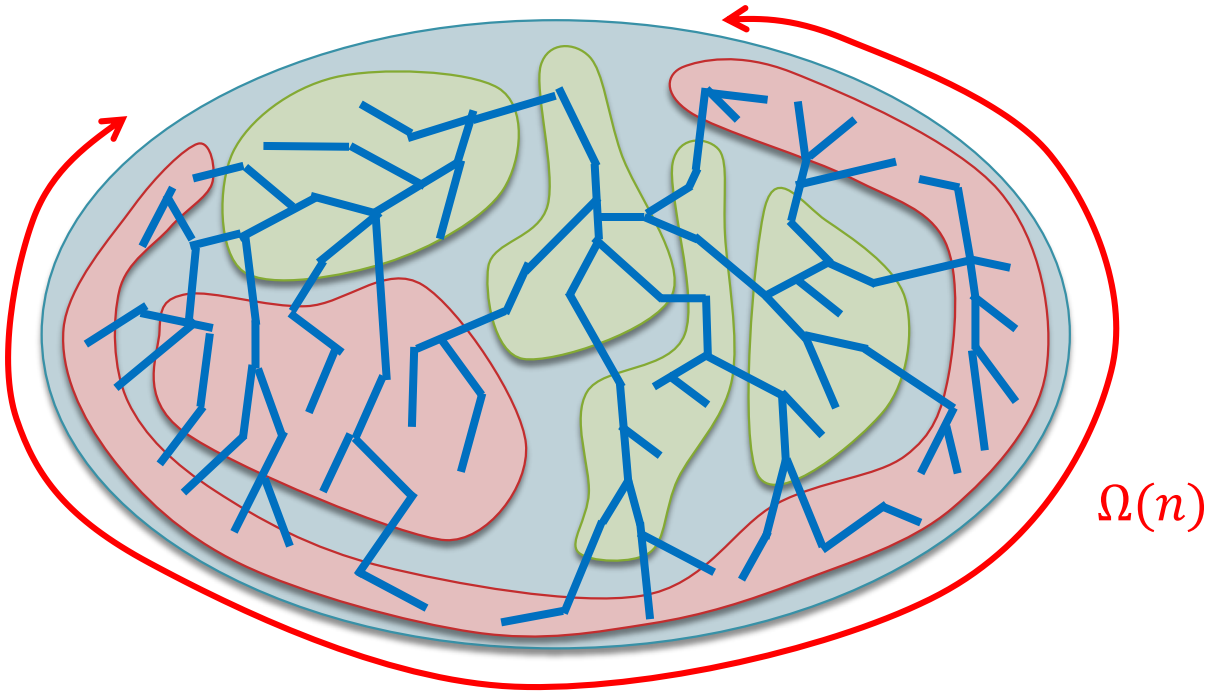
何が問題だったか？

- 問題：直径の大きい部分グラフにおける収集/放送



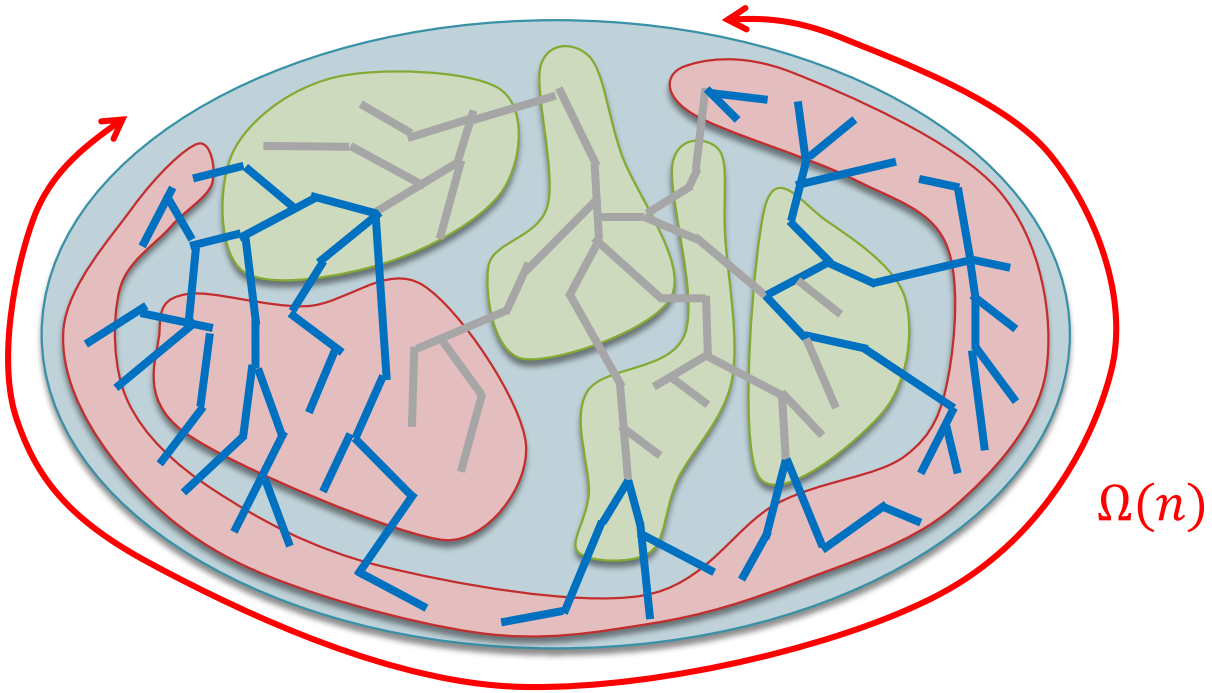
何が問題だったか？

- サイズの大きい($> \sqrt{n}$)部分グラフは
  BFS木を使った(Dilation $O(D)$ / Congestion $O(\sqrt{n})$)



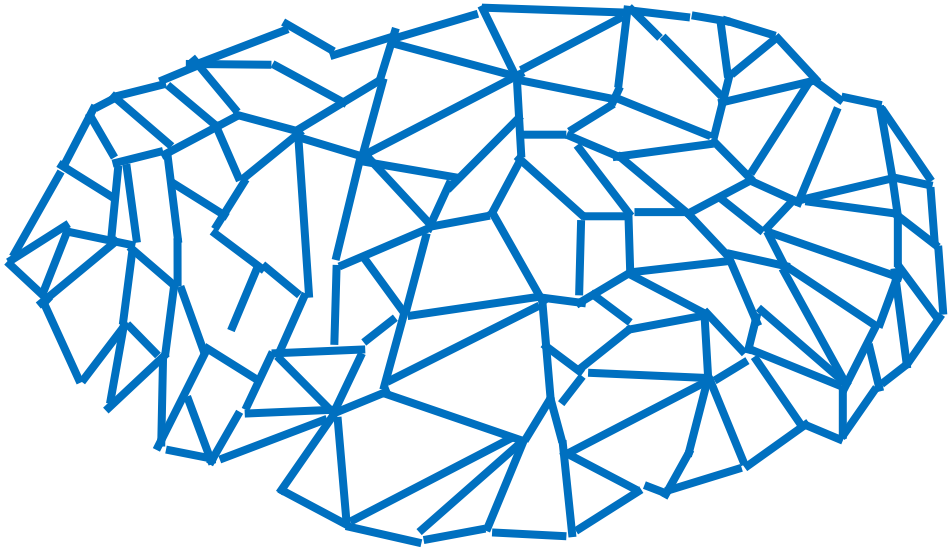
アプローチ

- 注意深く利用辺を割り当てることで(Dilationをあまり上げず)Congestionを下げることができないだろうか？



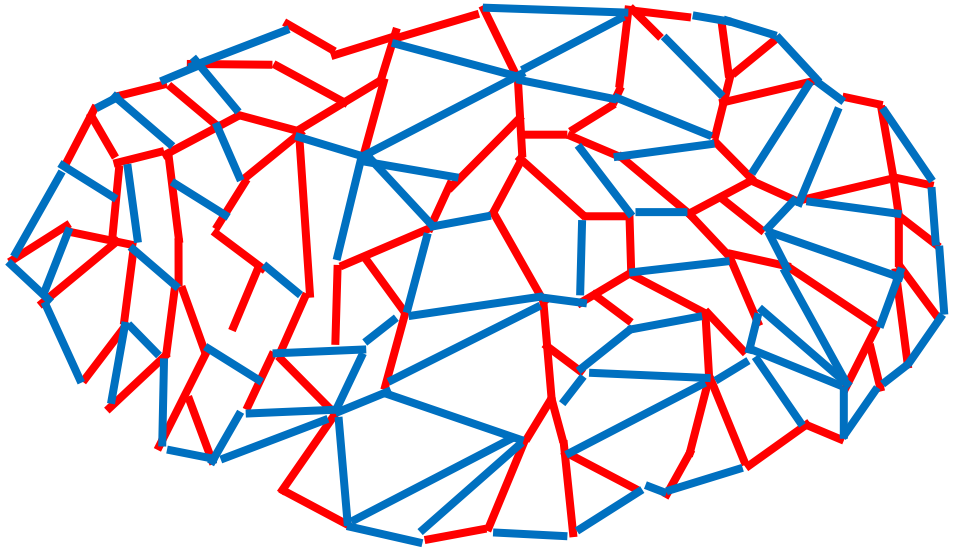
平面的グラフの例

- BFS木を構成



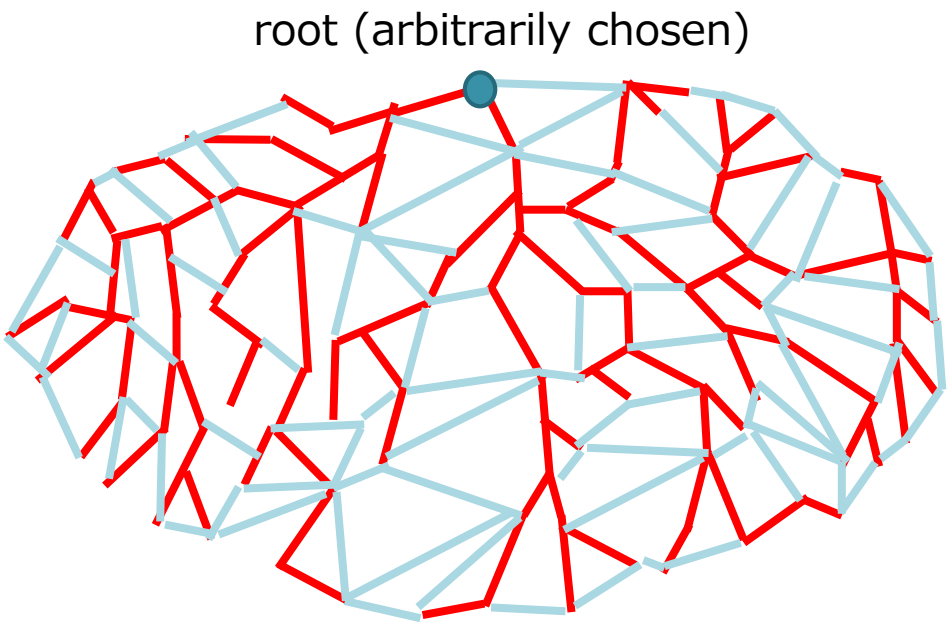
平面的グラフの例

- BFS木を構成



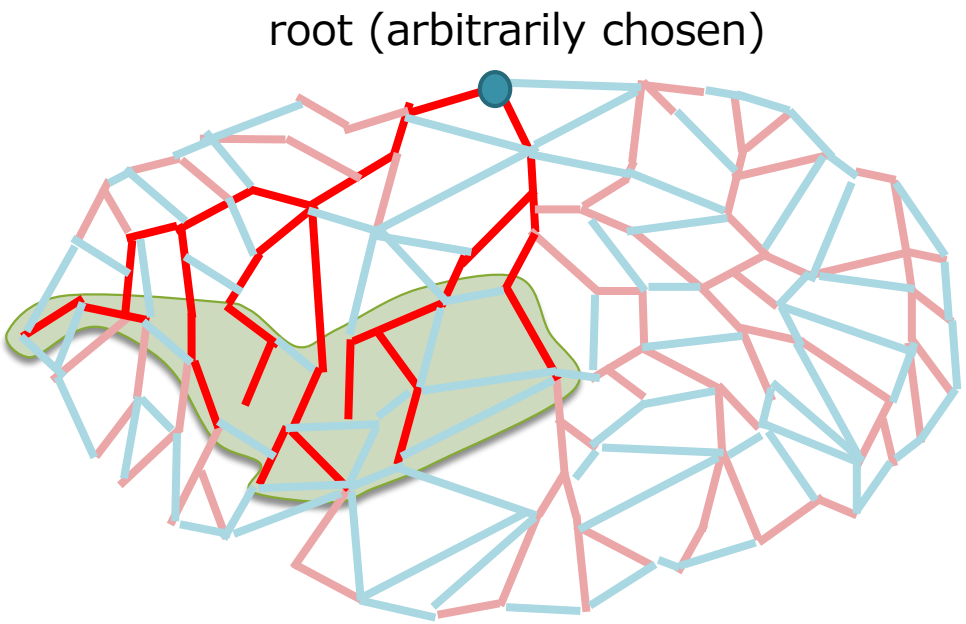
平面的グラフの例

- BFS木を構成
- 部分グラフについて，親へ向かうパスを
  ショートカットとして追加



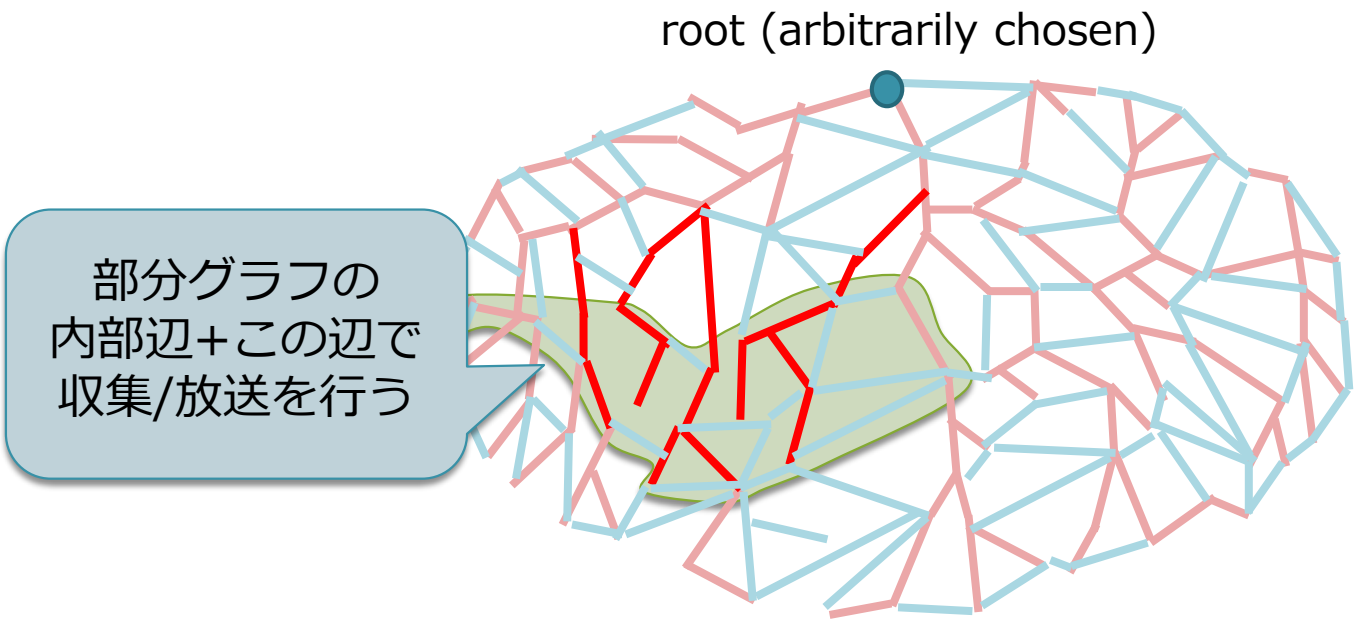
平面的グラフの例

- BFS木を構成
- 部分グラフについて，親へ向かうパスを
  ショートカットとして利用



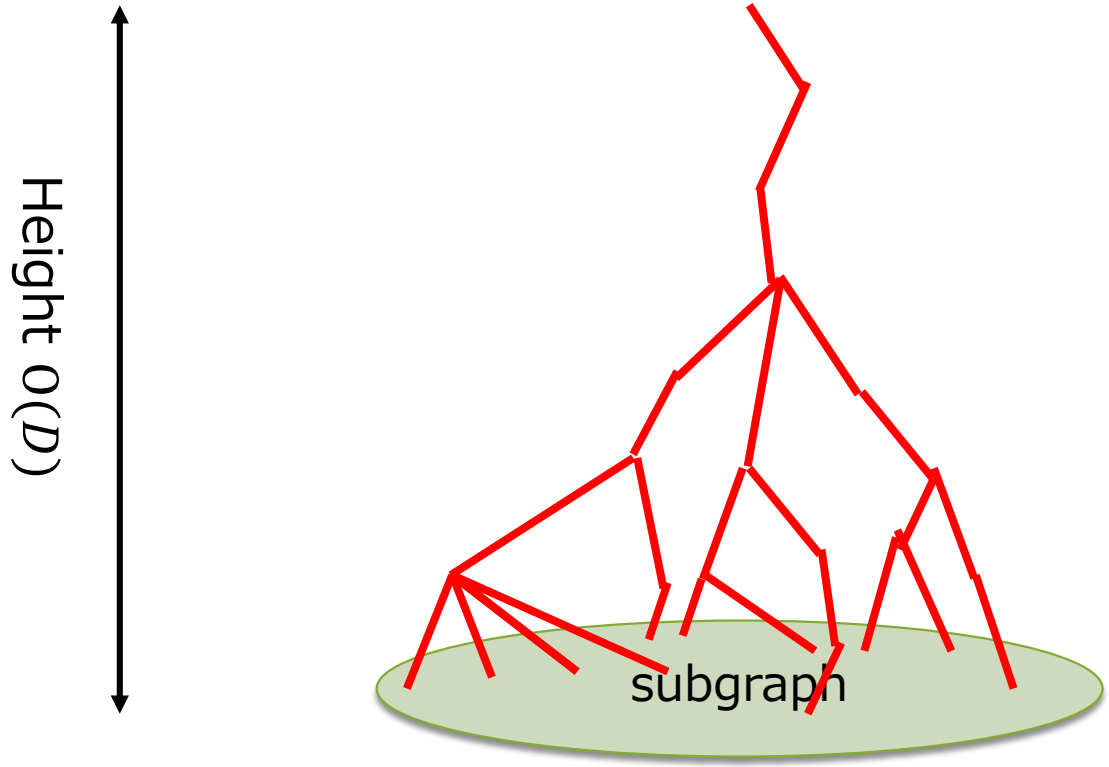
平面的グラフの例

- BFS木を構成
- 部分グラフについて，親へ向かうパスを
  ショートカットとして追加
- ただし最左と最右のパスは除外する



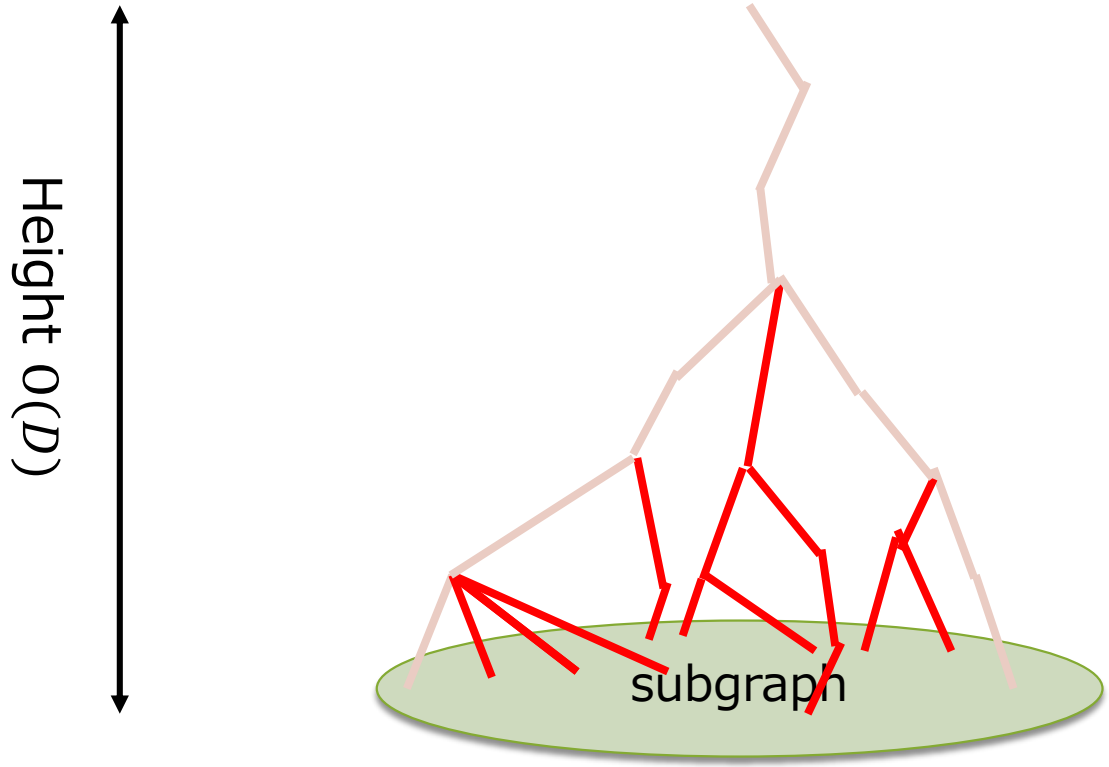
ショートカットの質

- この構成はdialation $O(D^2)$, congestion $O(D)$ を達成



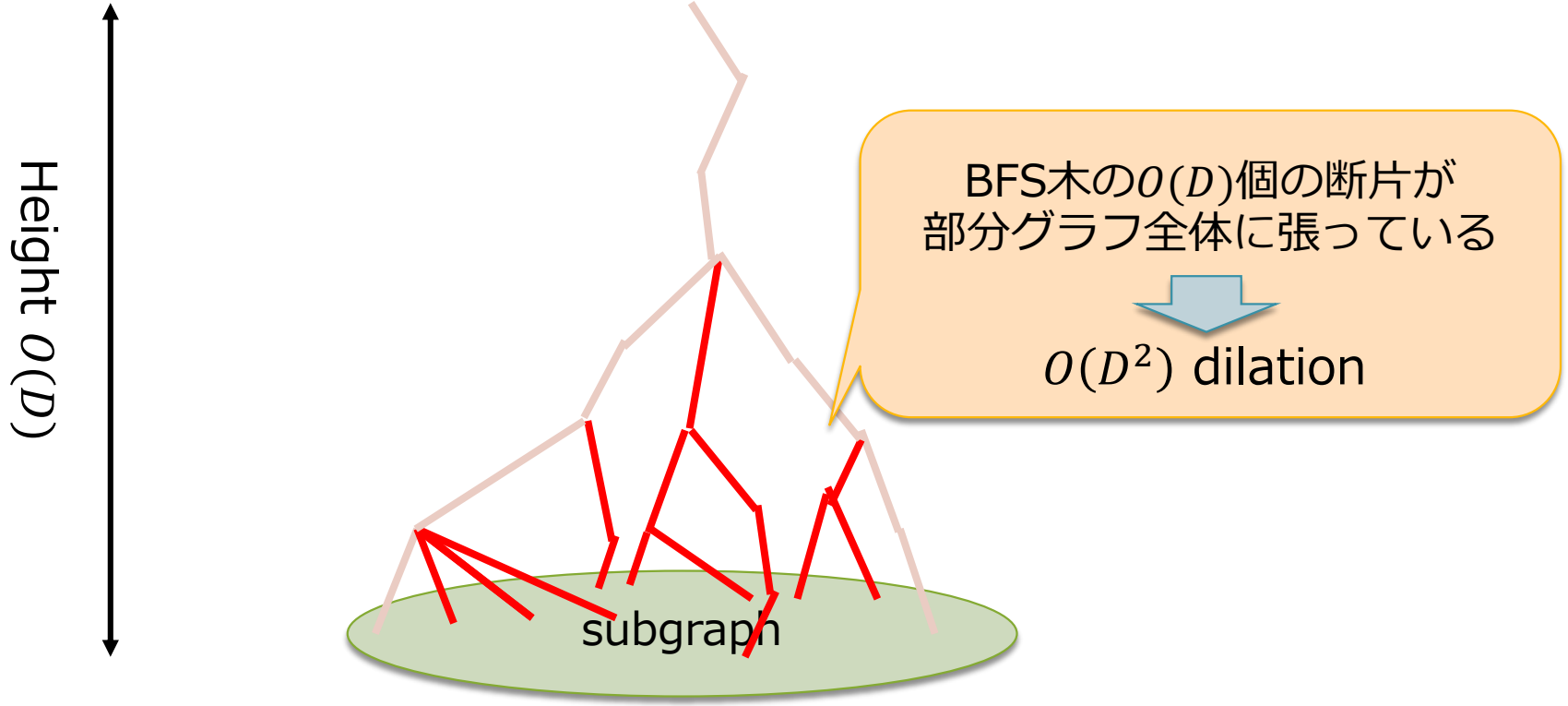
ショートカットの質

- この構成はdialation $O(D^2)$, congestion $O(D)$ を達成



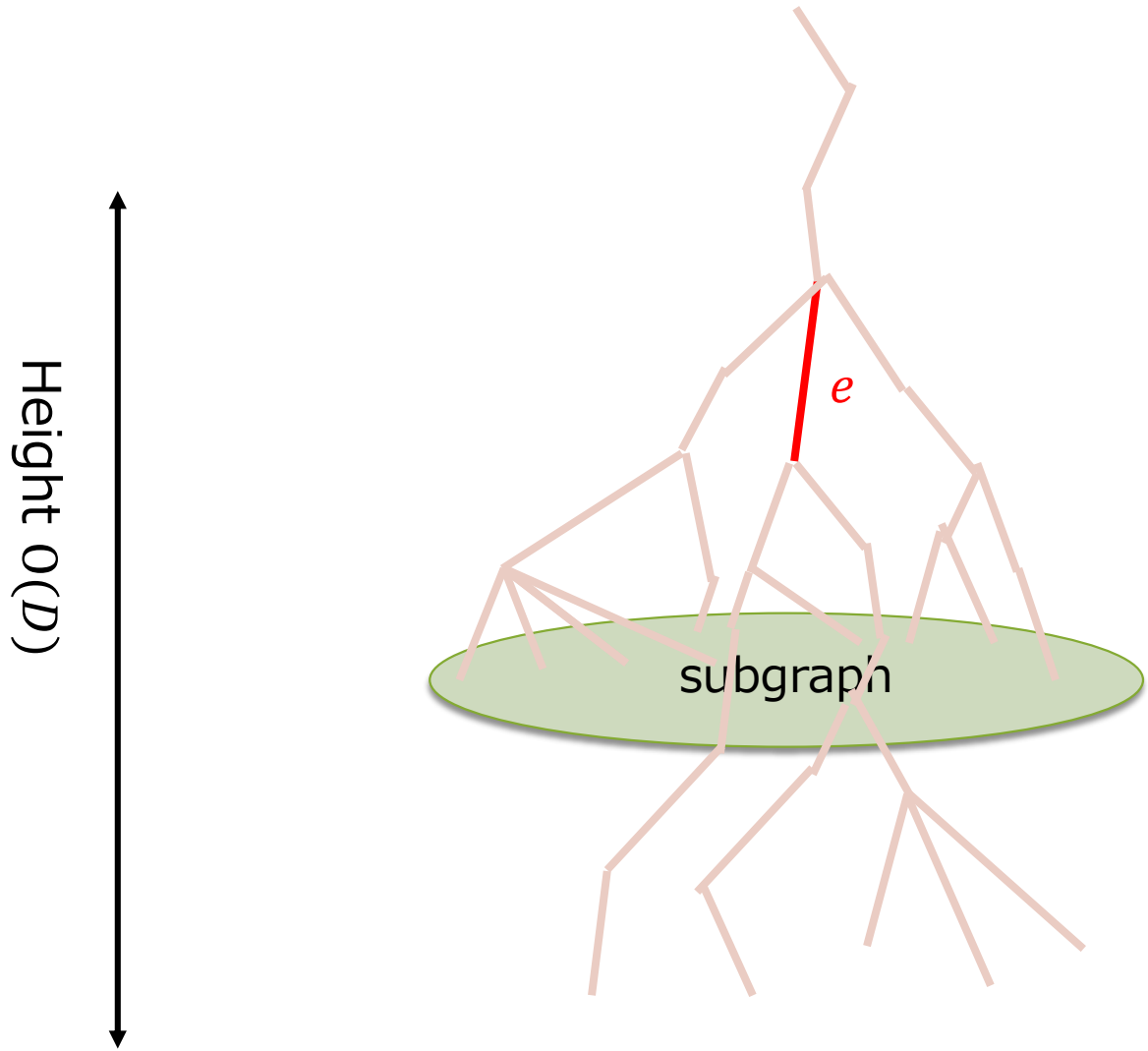
ショートカットの質

- この構成はdialation $O(D^2)$, congestion $O(D)$ を達成



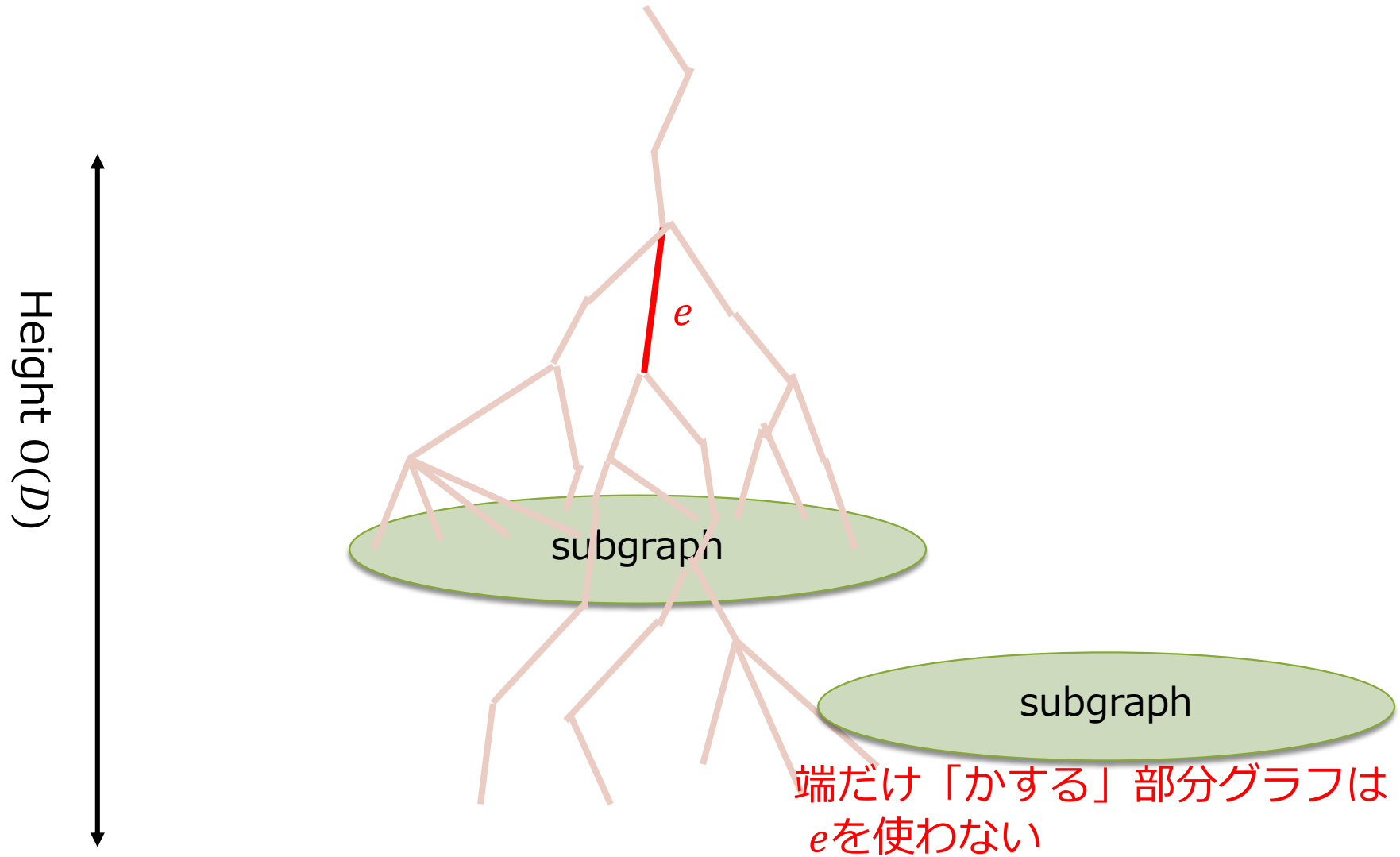
ショートカットの質

- この構成はdialation $O(D^2)$ congestion $O(D)$ を達成



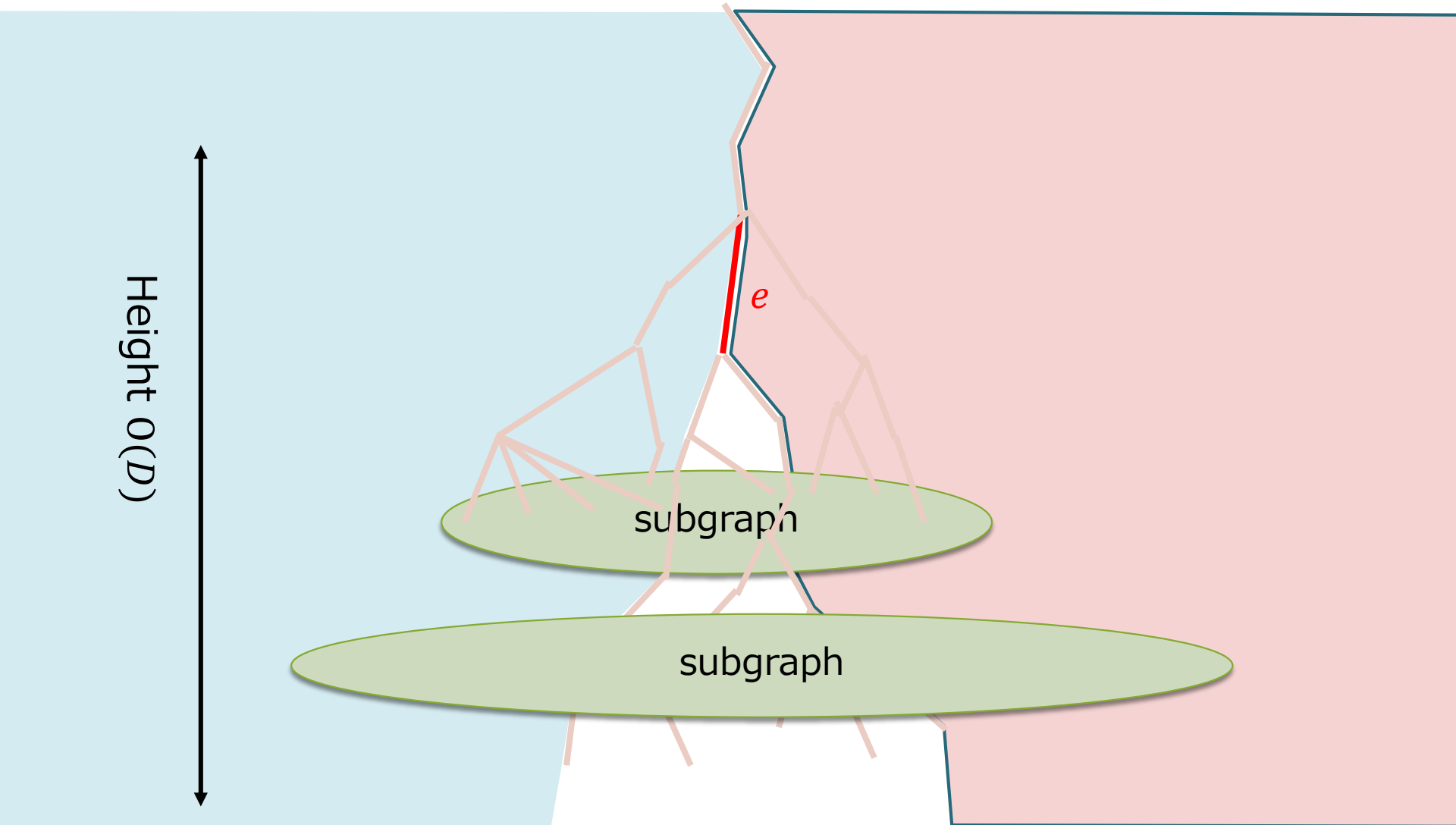
ショートカットの質

- この構成はdialation $O(D^2)$ congestion $O(D)$ を達成



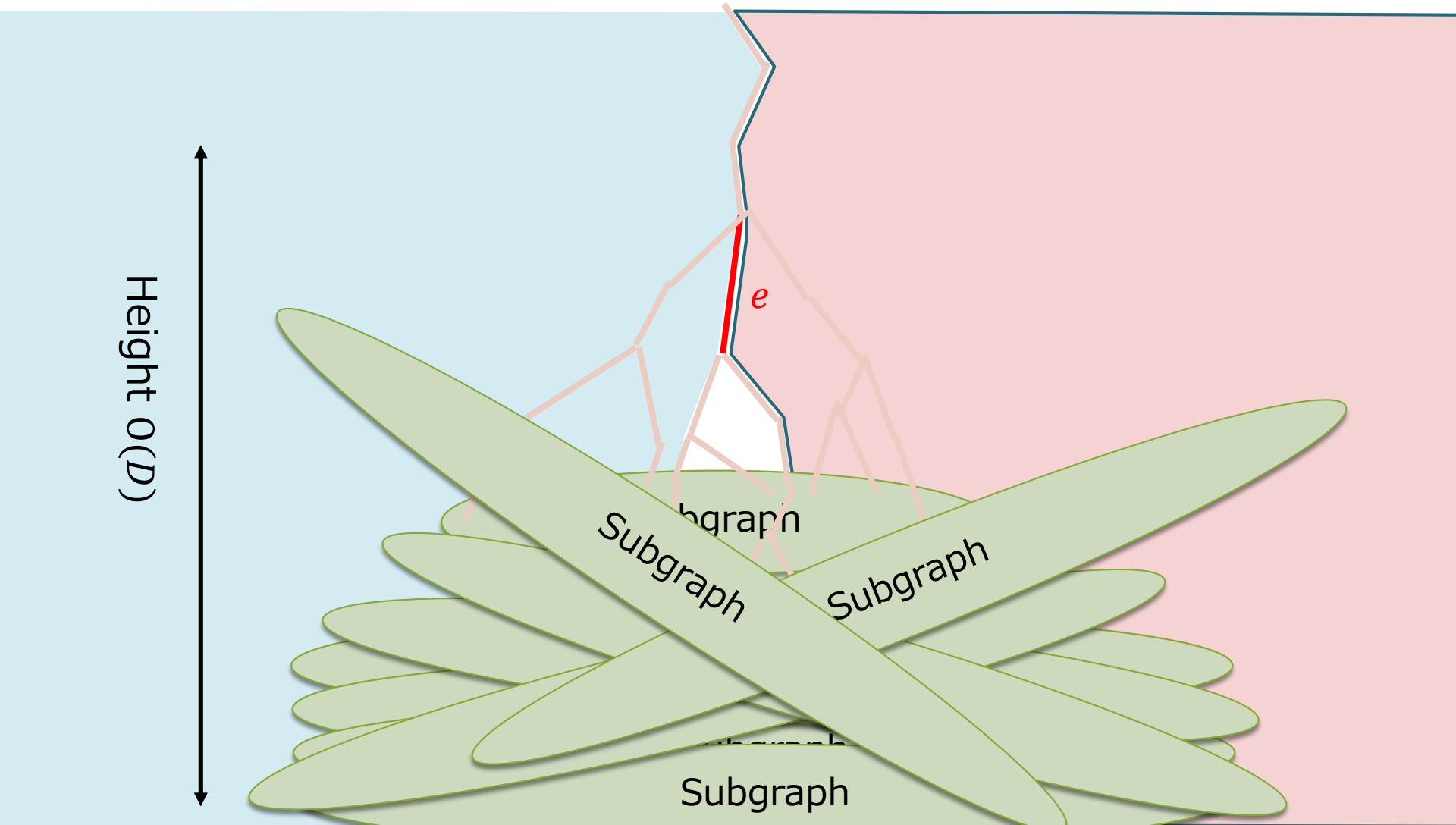
ショートカットの質

- この構成はdialation $O(D^2)$ congestion $O(D)$ を達成



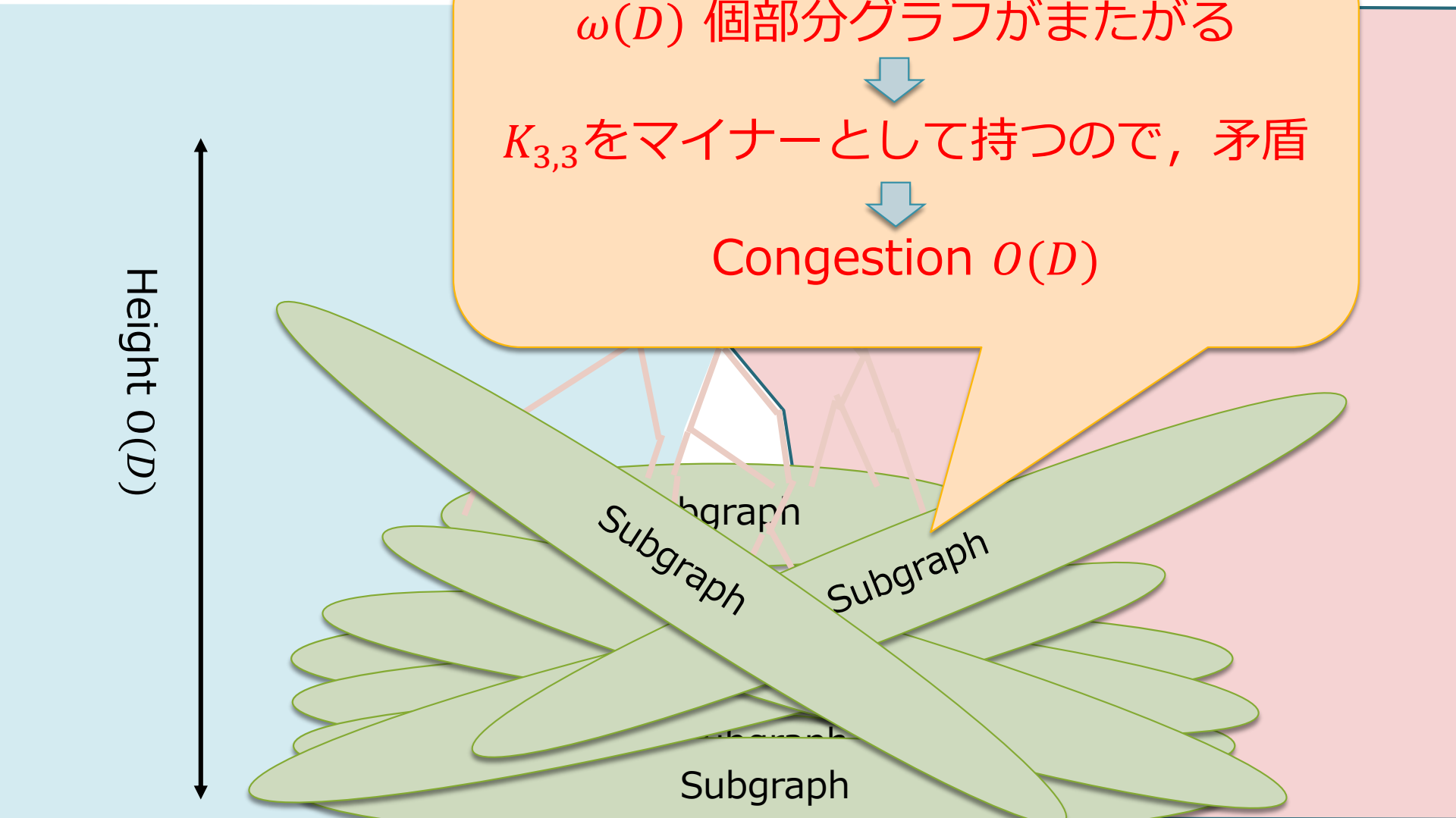
ショートカットの質

- この構成はdialation $O(D^2)$ congestion $O(D)$ を達成



ショートカットの質

- この構成はdialation $O(D^2)$ congestion $O(D)$ を達成



低競合ショートカット

- グラフGの連結部分グラフへの分割$P_1, P_2, \cdots, P_N$
- (c, d)-shortcut：各P_iへの部分グラフ(辺集合) $H_1, H_2, \cdots, H_N$ の割り当てで，以下を満たすもの
  - 任意の辺 $e \in E(G)$ は たかだか c回利用される (congestion)
  - 任意のiについて，$P_i + H_i$ は直径が高々d (dilation)
- 以下のメタ定理が成立する

定理 [GH16]
(c, d)-shortcutを持ち，かつそれを$f(n)$時間で構成可能なグラフに対して，MST，最小カットの$(1+\epsilon)$-近似を$\tilde{O}(c + d + f(n))$時間で計算することが可能

低競合ショートカット

- よいショートカットの存在(と構成)が知られているクラス

いずれも分散的に構成可能

Graph Family	Quality(c+d)	Construction	Lower Bound
Planar [GH16]	$O(D \log D)$	$O(D \log D)$	$\Omega\left(D \frac{\log D}{\log \log D}\right)$
Genus-g [HIZ16-2]	$O(\sqrt{g} D \log D)$	$O(\sqrt{g} D \log D)$	$\Omega\left(\frac{\sqrt{g} D}{\log g}\right)$
Treewidth-k [HIZ16-2]	$O(kD \log n)$	$O(kD \log n)$	$\Omega(kD)$

- 他にも，expander, unit disk graph, chordal graph などで構成できることが分かっている[unpublished]

参考文献

- [GHS82] R. G. Gallager, P. A. Humblet, and P. M. Spira, “A distributed algorithm for minimum-weight spanning trees,” ACM Trans. Prog. Lang. and Sys., 5(1), pp.66–77, 1983.
- [LMR94] T. Leighton, B. M. Maggs, and S. B. Rao, “Packet routing and job-shop scheduling in O(congestion+ dilation) steps”, Combinatorica, 14(2), pp.167–186, 1994.
- [Ghaffari15] M.Ghaffari, “Near-Optimal Scheduling of Distributed Algorithms”, PODC2015, pp.3-12.
- [Topkis85] D. M. Topkis, “Concurrent broadcast for information dissemination”, IEEE Trans. on Software Engineering, (10), pp. 1107–1112, 1985.
- [HIZ16-1] B. Haeupler, T. Izumi, G. Zuzic, “Low-Congestion Shortcuts without Embeddings”, PODC2016, pp.451-460.
- [GH16] M. Ghaffari, B. Haeupler, “Distributed algorithms for planar networks II: Low-congestion shortcuts, MST, and min-cut”, SODA2016, pp. 202–219.

- [KP98] S. Kutten and D. Peleg, “Fast Distributed Construction of Small k-Dominating Sets and Applications”, J. Algorithms, 28(1), pp.40–66, 1998.
- [Nanongkai14] D. Nanongkai. “Distributed Approximation Algorithms for Weighted Shortest Paths”, STOC2014, pp.565– 573.
- [RC87] A. Kalyanasundaram and G. Schnitger, “The probabilistic communication complexity of set intersection”, Conf. on Structure in Complexity Theory, pp.41-49, 1987.
- [Razborov90] A. A. Razborov. On the distributional complexity of disjointness. Theor. Comp. Sci., 106, pp.385–390, 1992.
- [DHK+12] A. Das Sarma, S. Holzer, L. Kor, A. Korman, D. Nanongkai, G. Pandurangan, D. Peleg, R. Wattenhofer, “Distributed verification and hardness of distributed approximation”, SIAM J. Comput., 41(5), pp.1235–1265, 2012.
- [FHW12] S. Frischknecht, S. Holzer, R. Wattenhofer, “Networks cannot compute their diameter in sublinear time”, SODA2012, pp.1150–1162.

- [PR00] D. Peleg and V. Rubinovich, “A Near-Tight Lower Bound on the Time Complexity of Distributed Minimum-Weight Spanning Tree Construction”, SIAM J. Comput., 30(5), pp.1427–1442, 2000.
- [LP13] C. Lenzen, B. Patt-Shamir, “Fast Routing Table Construction Using Small Messages”, STOC2013, pp. 381-390.
- [LPP06] Z. Lotker, B. Patt-Shamir, D. Peleg, “Distributed MST for Constant Diameter Graphs”, Dist. Comp., 18(6), pp.453-460, 2006.
- [HIZ16-2] B. Haeupler, T. Izumi, G. Zuzic, “Near-Optimal Low-Congestion Shortcuts on Bounded Parameter Graphs”, DISC2016, pp. 158-172.

演習問題

問題1

- ある1つのノードにグラフのトポロジ情報がすべて集まるような解法を「集中型の解法」と呼ぶことにする

- いかなる集中型の解法も$\Omega(n^2)$ ラウンドを必要とするようなトポロジを一つ挙げよ

問題2

- GHSアルゴリズムの実行時間評価$\Omega(n \log n)$ラウンドがタイトとなるような入力インスタンスを一つ挙げよ

問題3

(1)重み付きグラフにおいて，始点sから距離d以内(ホップ距離でなく，重み付きの距離)のすべての頂点に対してsからの距離を計算する, congestion=$\tilde{O}(1)$, dilation=$O(d)$ のアルゴリズムを構成せよ

(2)重み付きグラフにおいて，始点sからすべての頂点への$(1+\epsilon)$-近似距離を計算する，congestion=$O(1)$, dilation=$\tilde{\Theta}(n)$ のアルゴリズムを構成せよ(ϵ=$O(1)$とする)

注意：辺重みは正整数である

問題4

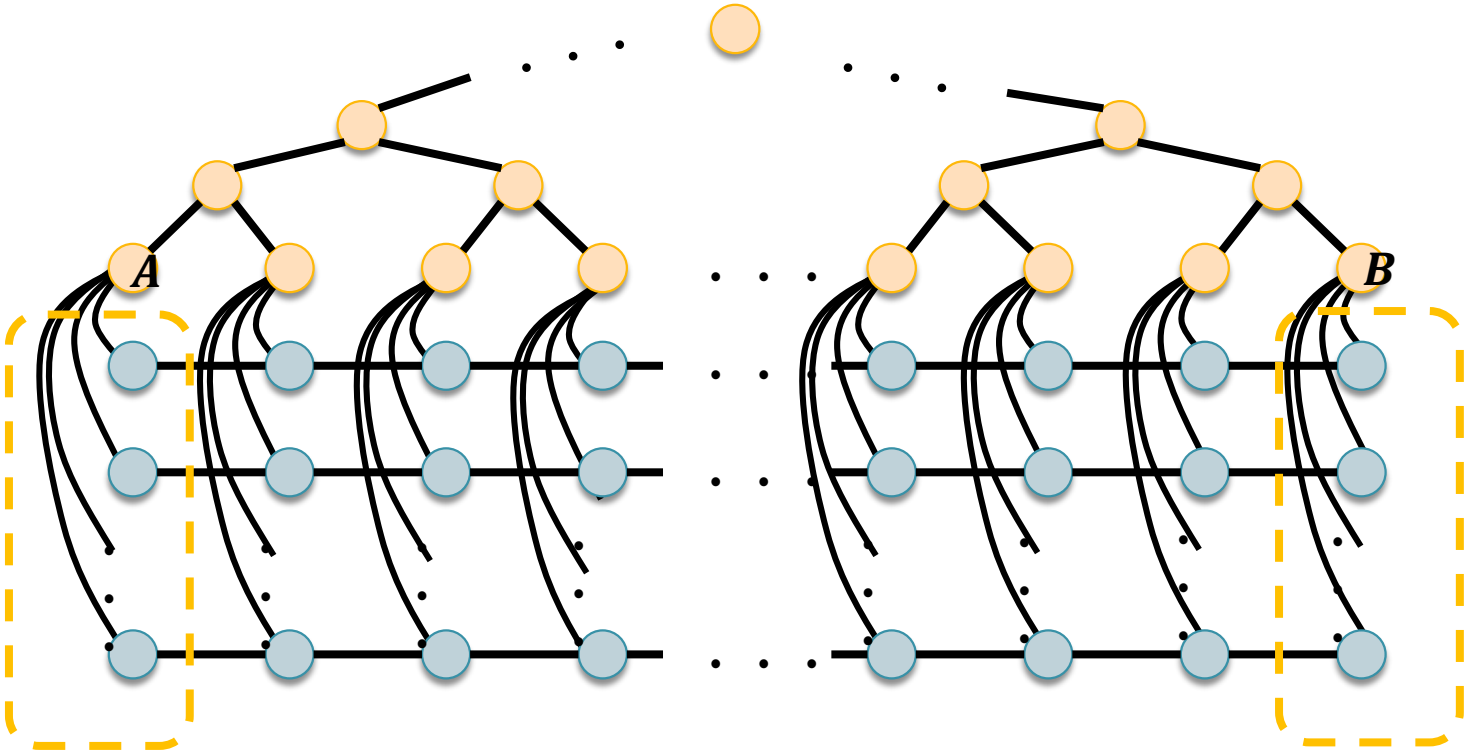
- 任意の近似率$\alpha(n)$に対して，s-t最短距離の$\alpha(n)$-近似値を求める問題が$\tilde{\Omega}(\sqrt{n})$ラウンド必要であることを示せ

問題5

- 入力としてネットワークの各辺eに$\{0,1\}$のラベルが付けられているとする
- 1でラベル付けされた辺集合で誘導される部分グラフが所望の性質を満たしているかどうかを検査する問題（部分グラフ検証問題）を考える．以下の性質の検査はいずれも$\tilde{\Omega}(\sqrt{n})$ラウンド必要であることを示せ
  - 連結である
  - サイクルを含む
  - 全域木である (たぶん難問)

問題4,5 補足&ヒント

- 以下の [ ] で示す部分に自由に辺を追加しても述べた定理の成立には影響しない
(この部分については，帰着のために自由にトポロジを改変してよい)



問題6

- 以下のグラフクラスが (c, d)-shortcutを持つとしたとき，c, dはどの程度良い値が保証できるだろうか？
  - 完全グラフ
  - サイズkの支配集合を持つ