HOKKAIDO UNIVERSITY

最大マッチングアルゴリズム

北海道大学 情報理工学博士2年 情報知識ネットワーク研究室
栗田 和宏

参考資料

通称，黄色本

B.コルテ，J.フィーゲン (著)，浅野孝夫，浅野泰仁，小野孝男，平田富夫 (訳)，『組合せ最適化 第 2 版』，丸善出版，2012 年.

一般グラフの最大マッチング，yosupoさん作のスライド

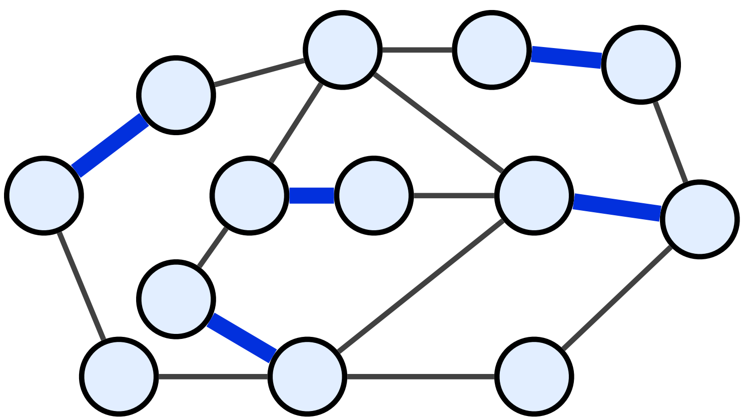
今回の内容はかなり難しいです！！！

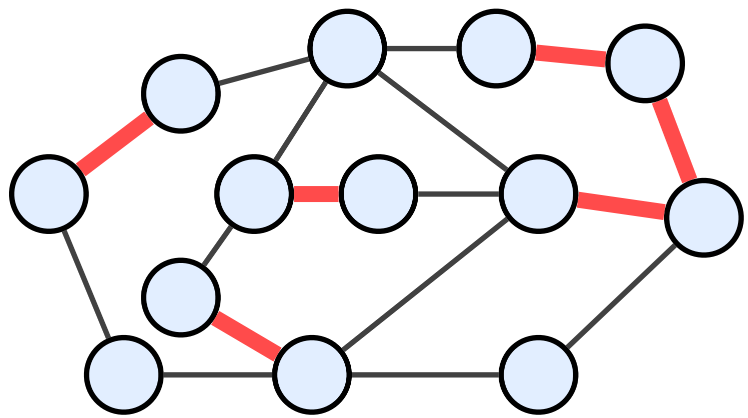
途中でどんどん質問してください.

今回の発表内容（最大マッチング）

今回扱う問題: グラフGが与えられるので，Ｇ中の要素数最大のマッチングを求めよ.

マッチングM: どんな2辺e, f ∈ Mも同じ端点を持たない辺集合



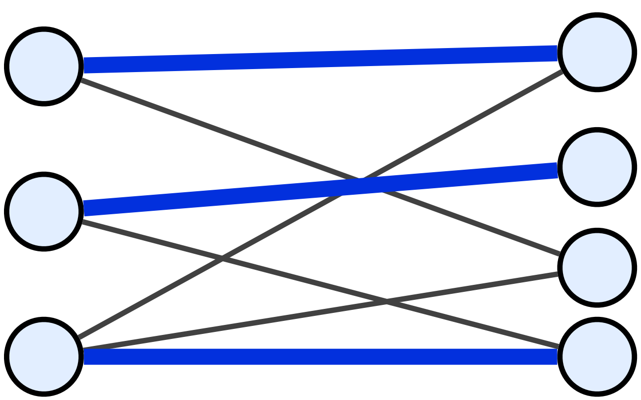


よくあるマッチング系の問題

競プロでは，グラフが二部グラフに制約された問題をよく見る．

現実問題でも二部グラフのマッチング問題はよく出るらしい．（drken マッチングで検索）

二部グラフの最大マッチングはフローを使って求められるが，それは割愛します．（ちょっとだけ今回の話とも関係あり）



最大マッチングアルゴリズムの概要

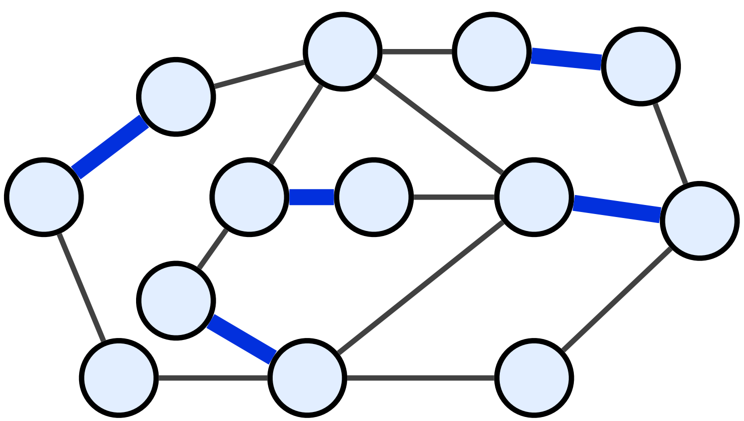
マッチングに関する性質

性質1: マッチングMが最大である ⇔ Gは**M-増加パスP**を持たない.

M-交互パス: Mに対して，パスPの辺が，Mに含まれる場合とそうでない場合が交互になるパス.

M-増加パス: 両端点がMにカバーされないM-交互パス

Mがvを端点に持つ辺を持つ時，**Mはvをカバーする**といい，そうでない場合，**Mはvをカバーしない**という.



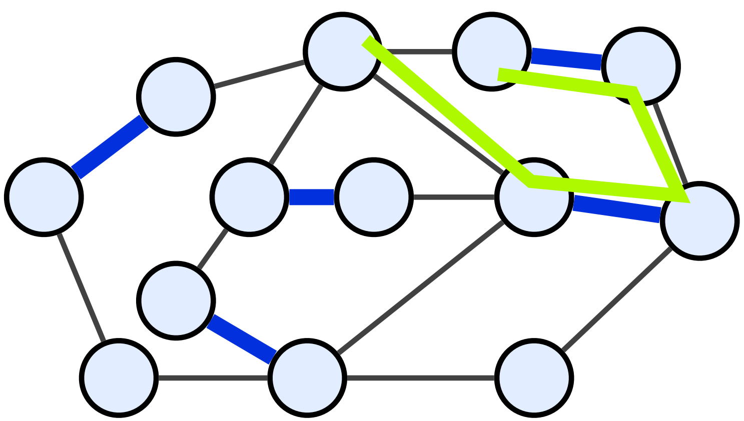
マッチングに関する性質

性質1: マッチングMが最大である ⇔ Gは**M-増加パスP**を持たない.

M-交互パス: Mに対して，パスPの辺が，Mに含まれる場合とそうでない場合が交互になるパス.

M-増加パス: 両端点がMにカバーされないM-交互パス

Mがvを端点に持つ辺を持つ時，**Mはvをカバーする**といい，そうでない場合，**Mはvをカバーしない**という.



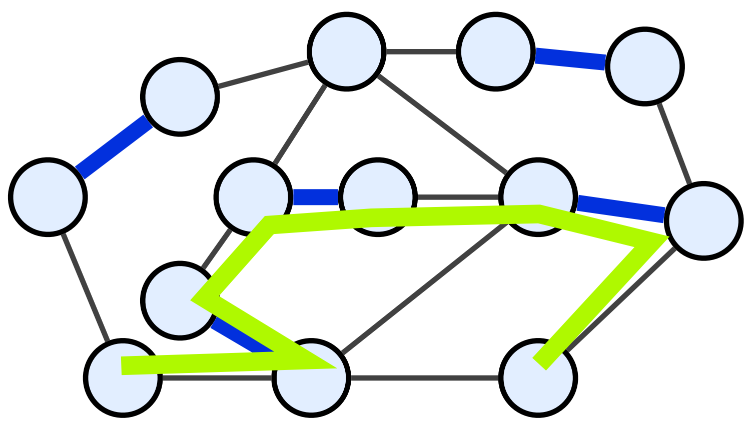
マッチングに関する性質

性質1: マッチングMが最大である ⇔ Gは**M-増加パスP**を持たない.

M-交互パス: Mに対して，パスPの辺が，Mに含まれる場合とそうでない場合が交互になるパス.

M-増加パス: 両端点がMにカバーされないM-交互パス

Mがvを端点に持つ辺を持つ時，**Mはvをカバーする**といい，そうでない場合，**Mはvをカバーしない**という.

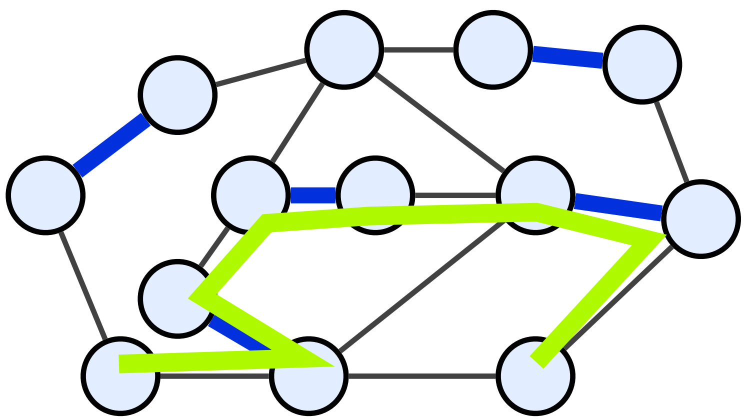


性質1の証明(A → B)

Gは M-増加パスPを持つならばマッチングMは最大でないを示す（対偶）．

Pは M-増加パスなので，M△Pはマッチングであり，Mより多くの辺を持つ．

- 2つの集合AとBに対して，A△B = A ∪ B \ (A ∩ B)とする．



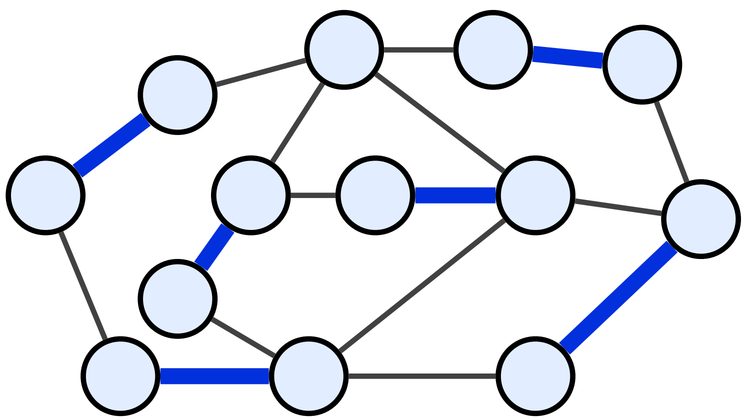
性質1: マッチングMが最大である ⇔ GはM-増加パスPを持たない．

性質1の証明(A → B)

GはM-増加パスPを持つならばマッチングMは最大でないを示す（対偶）．

PはM-増加パスなので，M△PはマッチングでありMより多くの辺を持つ.

- 2つの集合AとBに対して，A△B = A ∪ B \ (A ∩ B)とする.



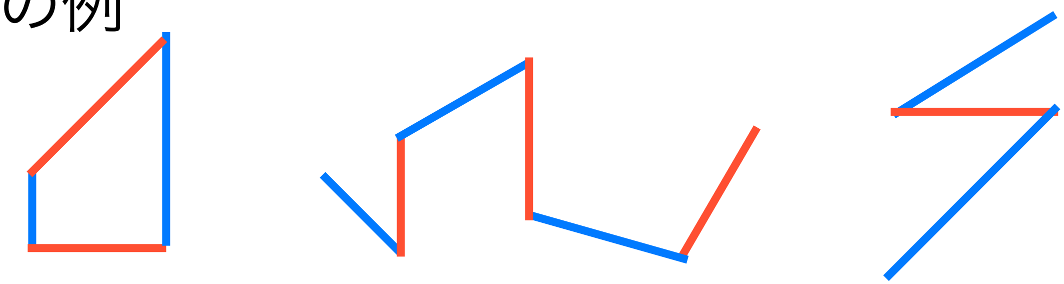
性質1: マッチングMが最大である ⇔ GはM-増加パスPを持たない.

性質1の証明(B → A)

マッチングMが最大でないならばGはM-増加パスPを持つことを示す（対偶）．

最大マッチングをM’とし，M△M’を考える．M△M’は高々次数が2のグラフになるので，パスとサイクルの集まりである．さらに全てのパスはMとM’の辺を交互に使うので，必ずM-増加パスを持つ．

M△M’の例



性質1: マッチングMが最大である ⇔ GはM-増加パスPを持たない.

最大マッチングを求めるアルゴリズム概要

1. M を空集合とする.
2. M-増加パスPを見つける.
3. MをM△Pに更新する.
4. M-増加パスPがなくなるまで2と3を繰り返す.

どうやってM-増加パスを見つけるのか？

- yosupotアルゴリズム？
- Edmondsアルゴリズム

性質1: マッチングMが最大である ⇔ GはM-増加パスPを持たない.

M-増加パスの発見法

M-増加パスの発見

問題:

入力: グラフGとマッチングM

タスク: G中のM-増加パスを1つ出力せよ

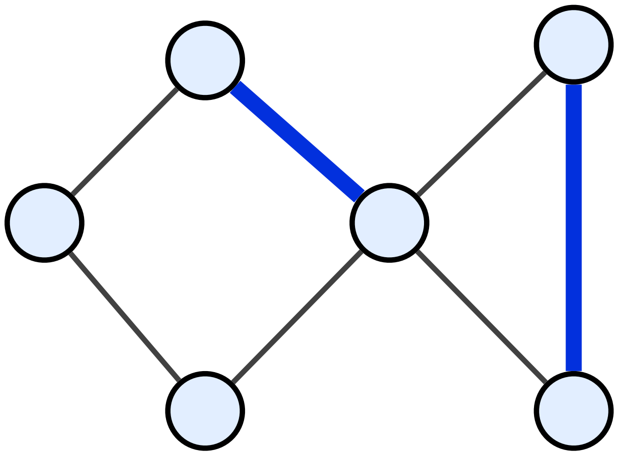
なんかdfsしたくなってくる問題…

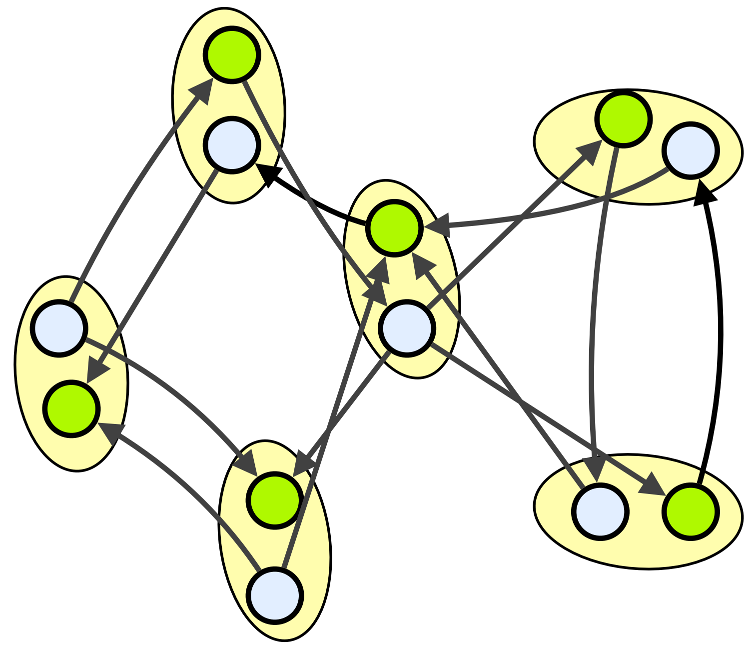
普通のdfsではダメなので，次のdfsを考える.

- 頂点を2倍にしたグラフ上のdfs.

辺のつけ方はちょっとめんどくさいので，具体例を見てみよう.

グラフの生成方法とM-増加パスの発見

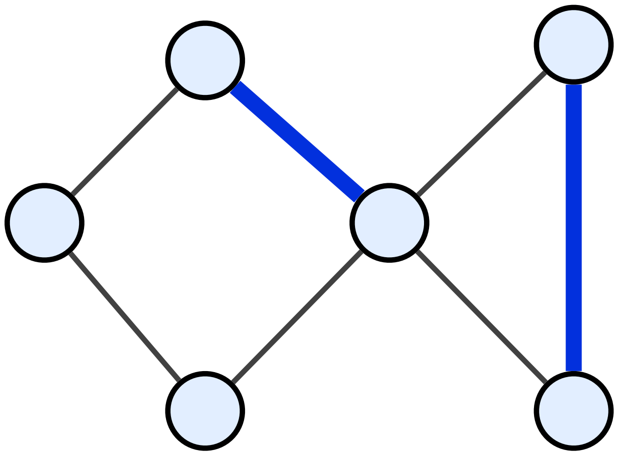


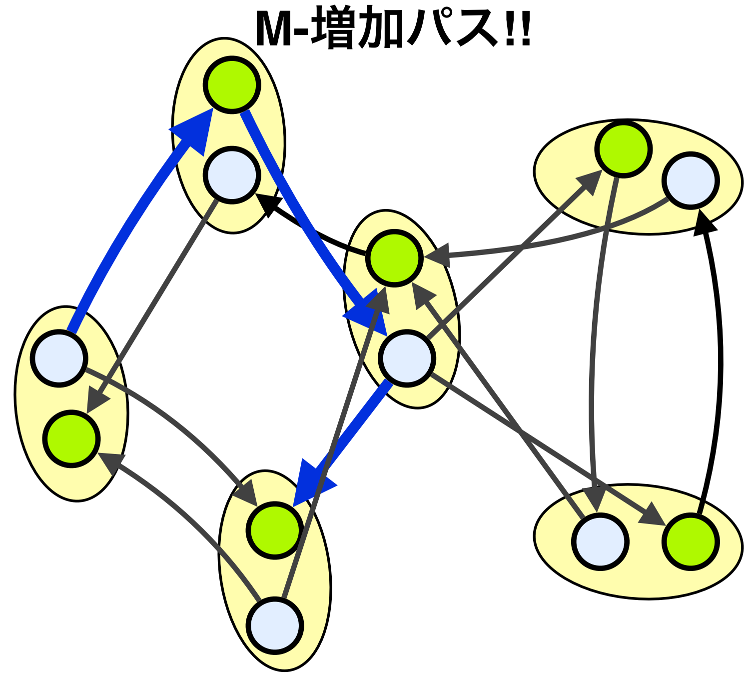


前使った辺がマッチングに含まれる

前使った辺がマッチングに含まれない

グラフの生成方法とM-増加パスの発見

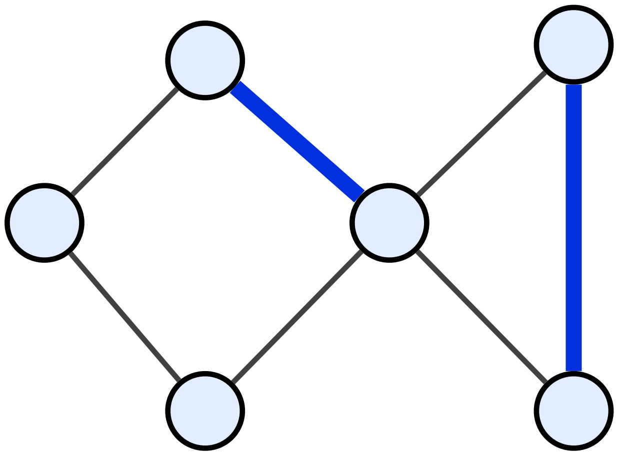


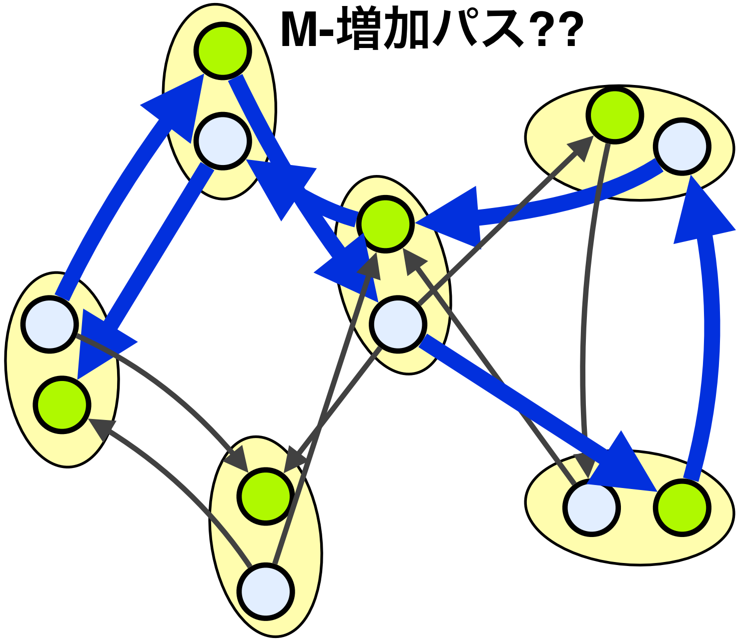


前使った辺がマッチングに含まれる

前使った辺がマッチングに含まれない

グラフの生成方法とM-増加パスの発見



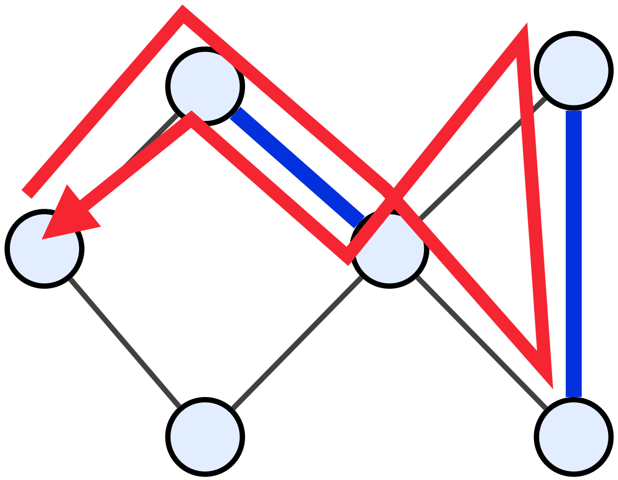


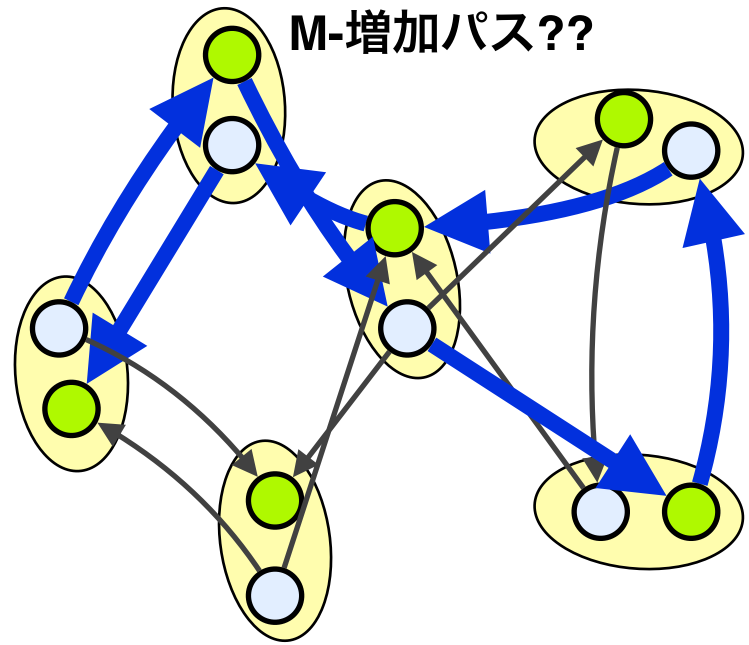
前使った辺がマッチングに含まれる

前使った辺がマッチングに含まれない

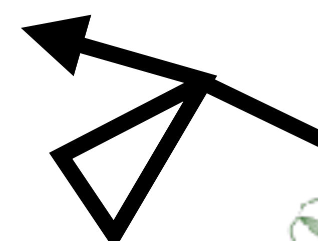
グラフの生成方法とM-増加パスの発見

このパスは閉路を持つ





こんなパスを見つける場合がある.



- 前使った辺がマッチングに含まれる
- 前使った辺がマッチングに含まれない

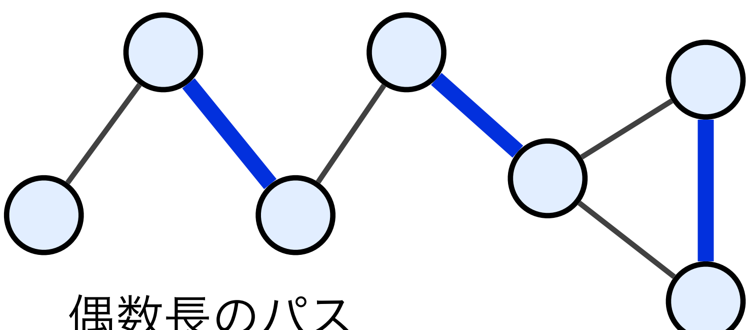
見つけたパスっぽいものの利用

このパスっぽいものも有用である. （下図）

→ 発見したサイクルCを縮約して得られたグラフG’の最大マッチングとGの最大マッチングのサイズは等しい.

これを示すために次の条件を示す.

- GがM-増加パスを持つ ↔ G’がM’-増加パスを持つ.



見つけたパスっぽいものの利用

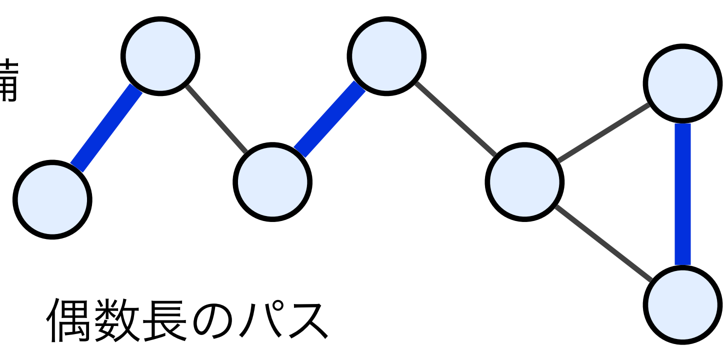
このパスっぽいものも有用である．（下図）

→ 発見したサイクルCを縮約して得られたグラフG’の最大マッチングとGの最大マッチングのサイズは等しい．

これを示すために次の条件を示す．

- GがM-増加パスを持つ ↔ G’がM’-増加パスを持つ．

ちょっと前準備



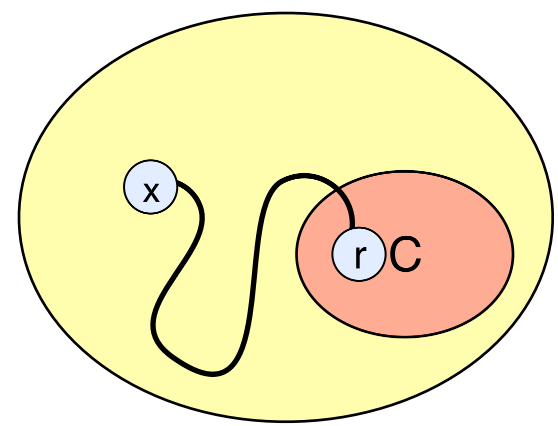
奇数長のサイクルC

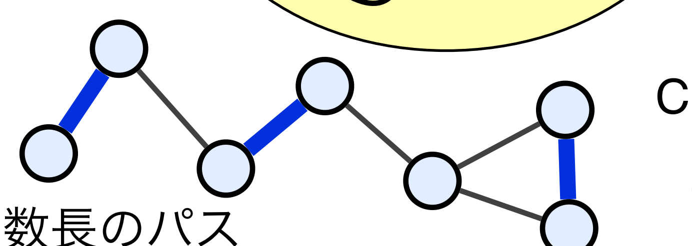
偶数長のパス

証明: A → B

GがM-増加パスPを持ち，PがC中の頂点を含む場合

CはMにカバーされない頂点を1つだけ持つ．これをrとする．Pは端点を2つ持ち，1つはrと異なる．これをxとする．xはCに含まれない．Pのx-r部分パスP’は両端点がMにカバーされないので，M-増加パスであり，これはM’-増加パスになる．





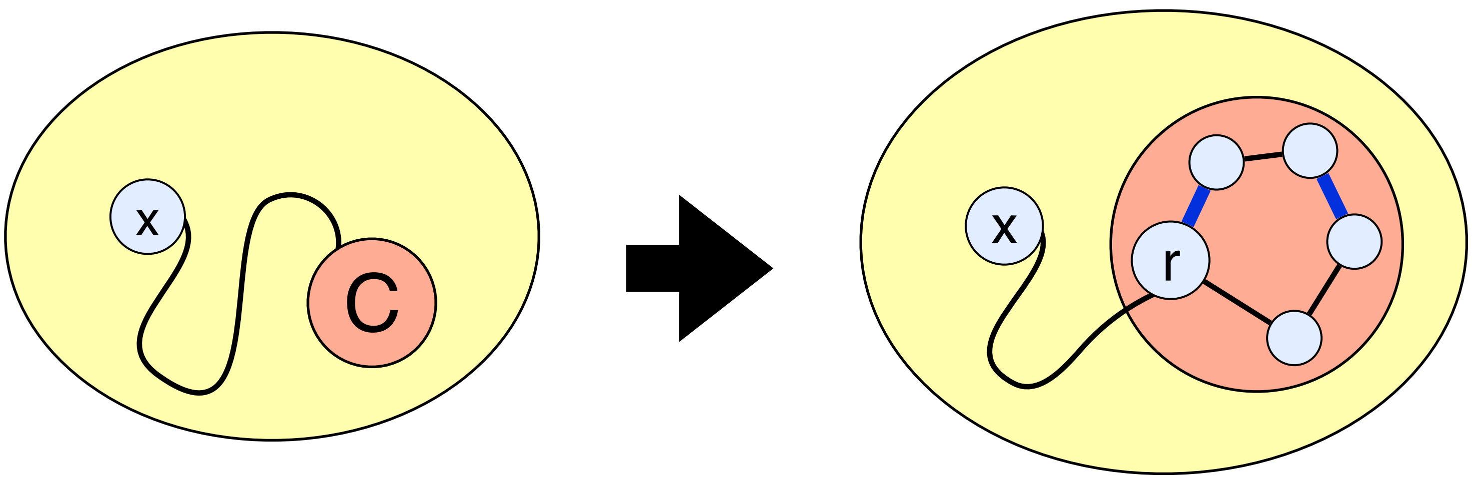
偶数長のパス

GがM-増加パスを持つ ↔ G’がM’-増加パスを持つ.

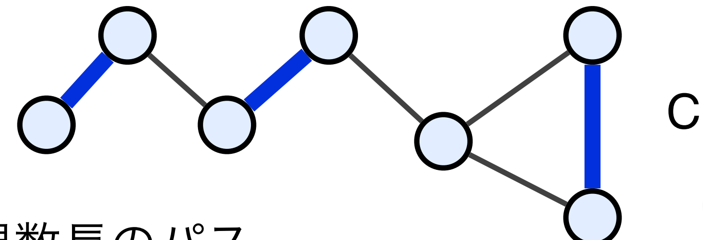
C: 潰した頂点の集合

証明: B → A

P’をG’中のM’-増加パスとする．これはちょっと工夫するとG中のM-増加パスになる．



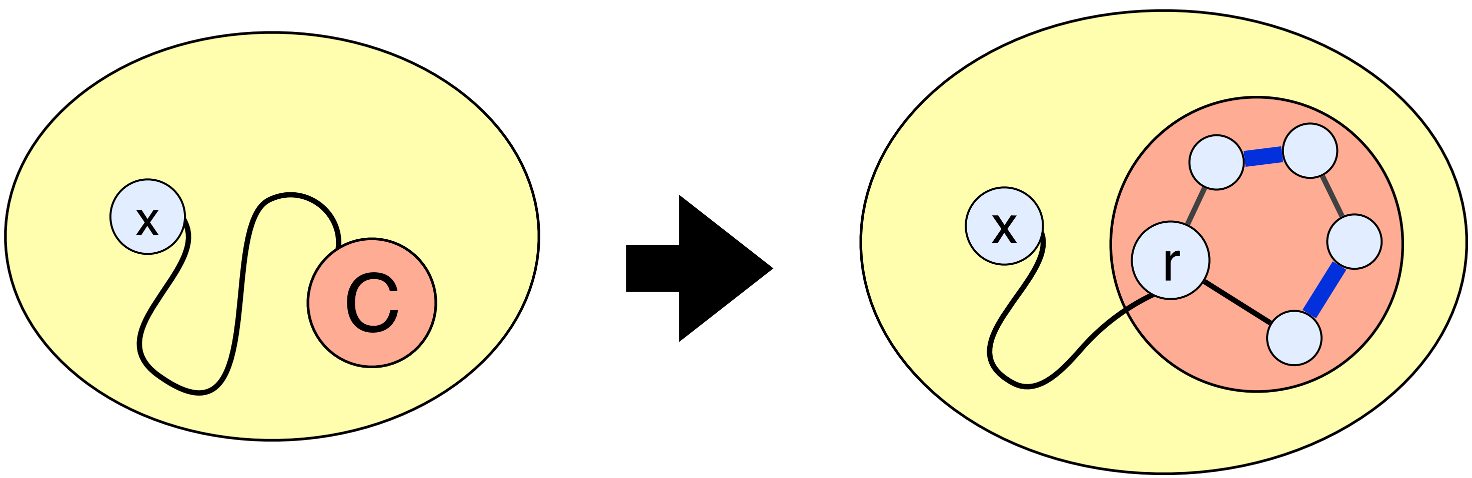
GがM-増加パスを持つ ↔ G’がM’-増加パスを持つ.
C: 潰した頂点の集合

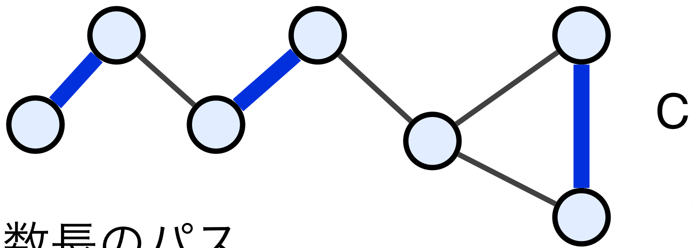


偶数長のパス

証明: B → A

P’をG’中のM’-増加パスとする．これはちょっと工夫するとG中のM-増加パスになる．





偶数長のパス

GがM-増加パスを持つ ↔ G’がM’-増加パスを持つ．
C: 潰した頂点の集合

アルゴリズムの概要

1. カバーされていないそれぞれの点からdfsする.
2. 閉路を見つけたら縮約して，1に戻る.
3. M-増加パスを見つけたらMを増加させる．さらに，縮約した閉路を全て復元して，1に戻る.

計算量解析

1つのM-増加パスの発見には,

M-増加パス発見の始点の候補が$O(n)$

毎回のdfsに$O(n + m)$

グラフの縮約の回数が$O(n)$

の計算が必要.

全体で$O(n^3(n + m))$時間アルゴリズムになる.

yosupoさんスライドでは$O(n^4)$と言っているので
もっと頭が良いことをしてそう

Edomondsのアルゴリズム

Edmonds アルゴリズムの概要

yosupoアルゴリズムと同様に，M-増加パスを発見して，貪欲にマッチングのサイズを大きくする

改善案:

それぞれの頂点から毎回dfsする必要あるのか？

縮約後に1から探索し直す必要があるのか？

交互森

交互森: グラフGとマッチングMに対して，森Fが次の3つの条件を満たす時，Fを交互森という．

(a) V(F)は**Mでカバーされない点を全て含み**，Fのどの連結成分も**Mでカバーされない点をちょうど1個含む**．この頂点を根という．

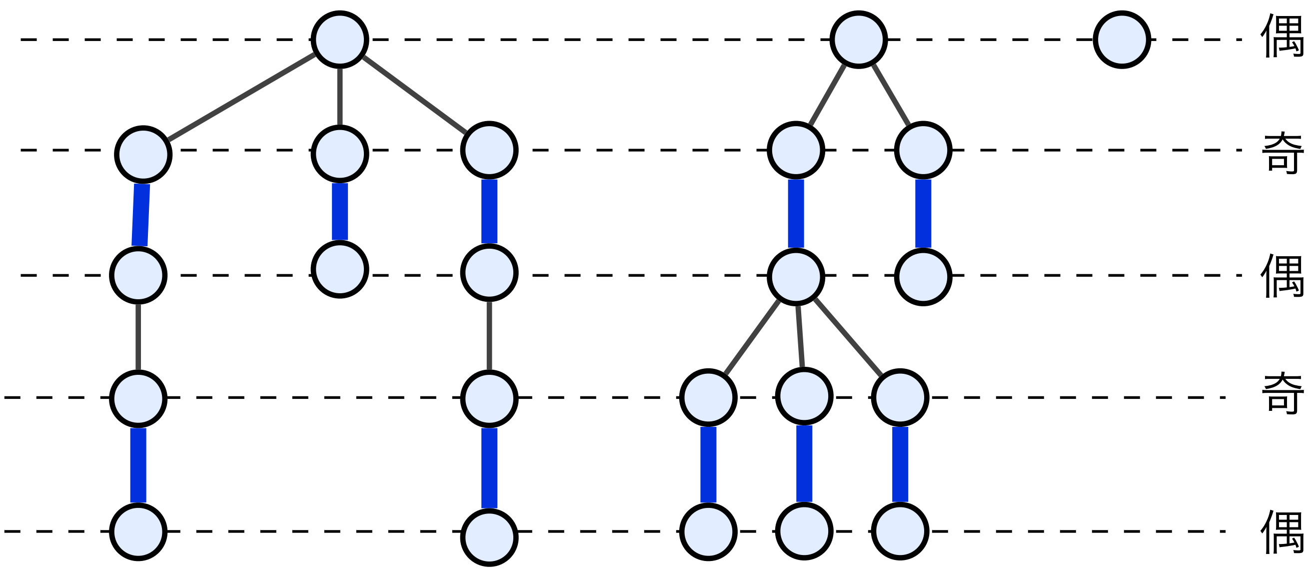
(b) Fのどの奇点も**次数が2**である．

(c)任意の頂点vに対して，vからvを含む連結成分の根までのパスは**M-交互パス**である．

vから根までの距離が偶数の時，vを偶点，そうでない時vを奇点という．

交互森の例

交互森: グラフGとマッチングMに対して，森Fが次の3つの条件を満たす時，Fを交互森という.



(a) V(F)はMでカバーされない点を全て含み，FのどのIの連結成分もMでカバーされない点をちょうど1個含む.

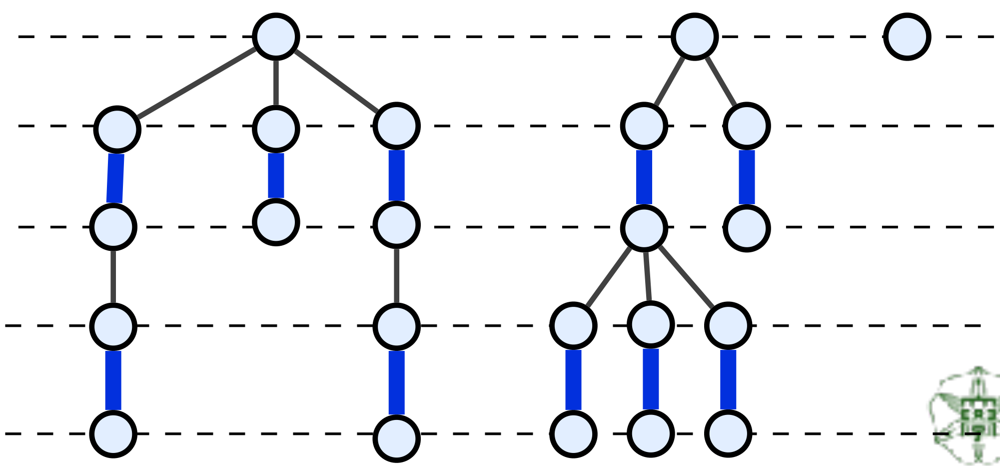
(b) Fのどの奇点も次数が2である.

(c)任意の頂点vに対して，vからvを含む連結成分の根までのパスはM-交互パスである.

交互森の拡張

偶点から辺を探索し，次に見つけた頂点が次の4つの場合について場合分けして考える.

1. 森に現れていない点
2. 奇点
3. 異なる交互木中の偶点
4. 同じ交互木中の偶点

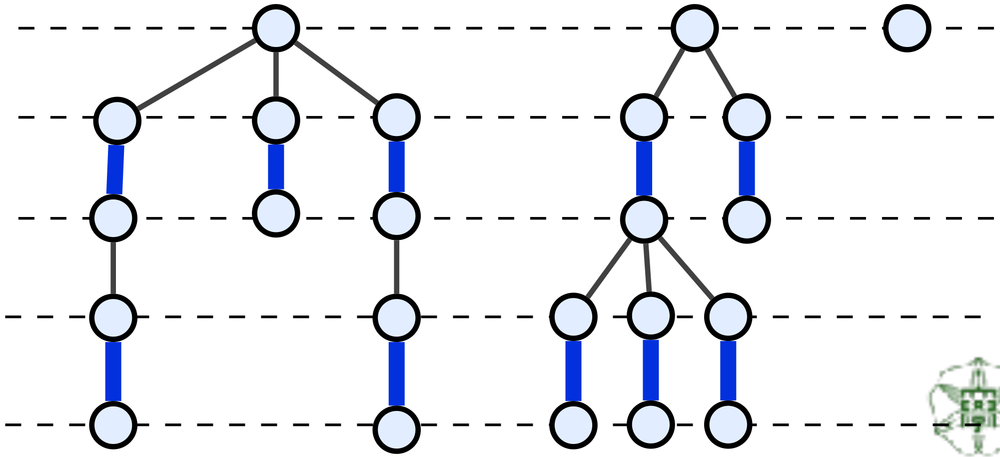


交互森の拡張

偶点から辺を探索し，次に見つけた頂点が次の4つの場合について場合分けして考える.

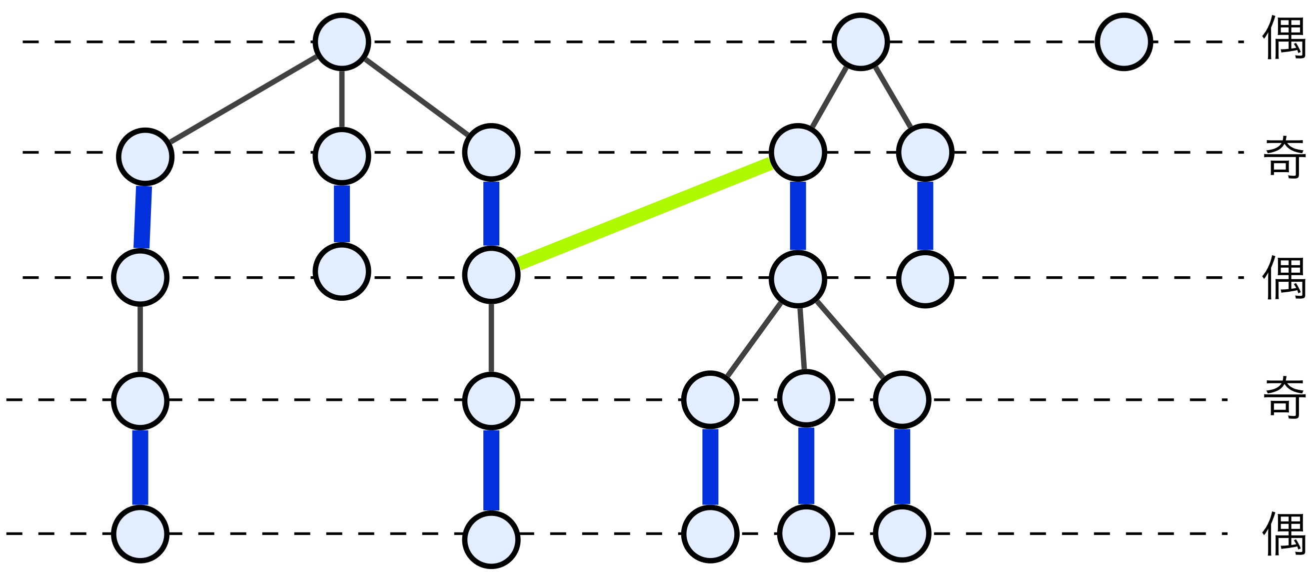
1. 森に現れていない点
2. 奇点
3. 異なる交互木中の偶点
4. 同じ交互木中の偶点

1はただ木に加えるだけなので割愛



2.奇点

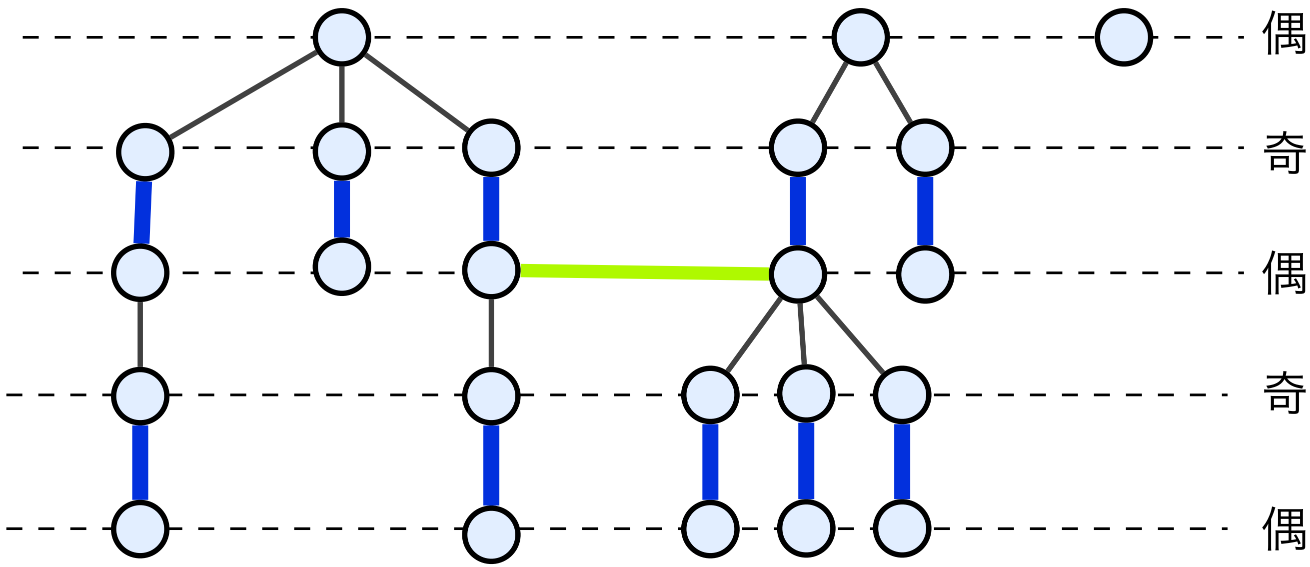
この場合，この辺は無視して探索を続ける.



3.異なる交互木中の偶点

この場合，交互木はM-増加パスPを持つ.

Pを使ってマッチングを増大させ，また最初から交互木を作る



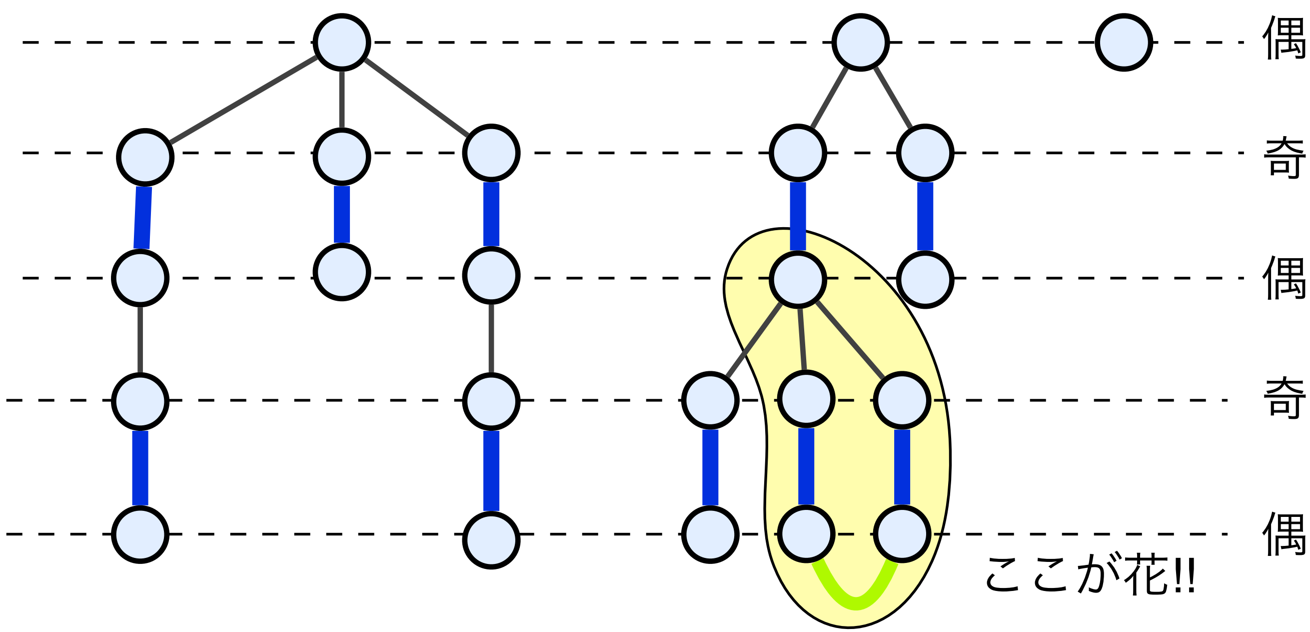
4.同じ交互木中の偶点

この場合，交互木は**花**を持つ.

因子臨界的グラフ: グラフのどんな頂点を削除してもグラフは完全マッチングを持つ.

花: G中で|M∩E(H)| = (|V(H)| - 1)/2を満たす因子臨界的部分グラフH（HはGの誘導部分グラフ）

基点: 花C中でMにカバーされない頂点

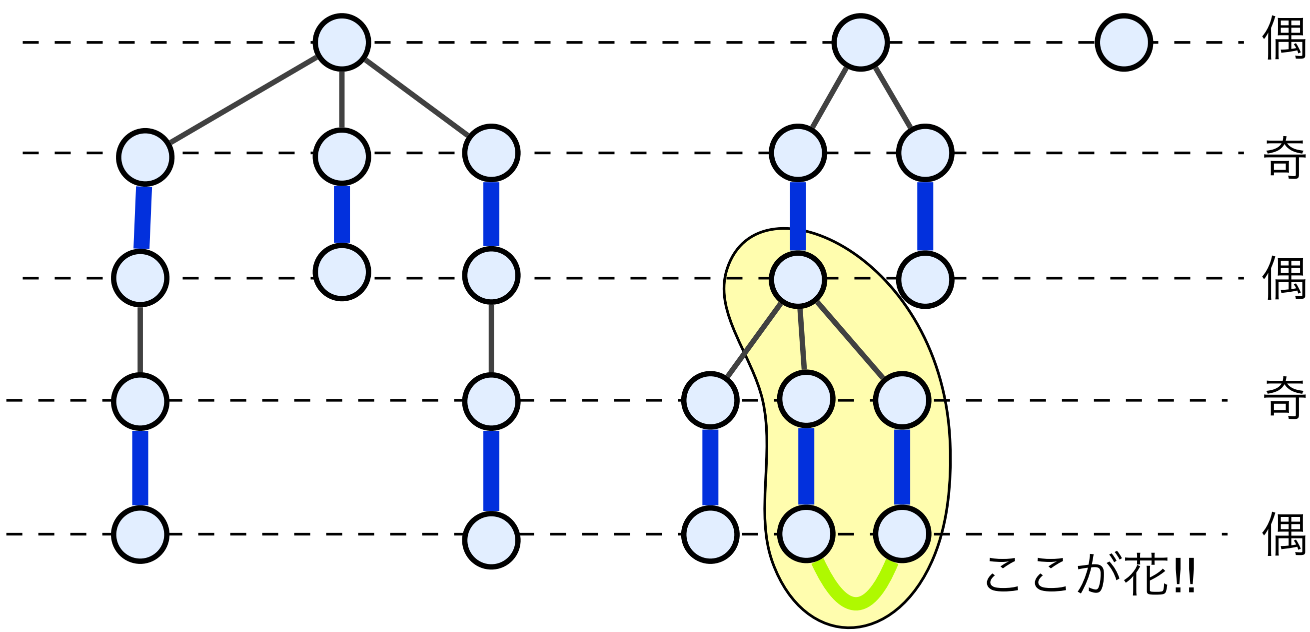


4.同じ交互木中の偶点

グラフ中の花を縮約したグラフをG’とする．この時，Gの最大マッチングのサイズとG’の最大マッチングのサイズが同じになる．

証明は前半のものとほぼ同じ

見つけた花を縮約して，交互木の拡張を続ける．

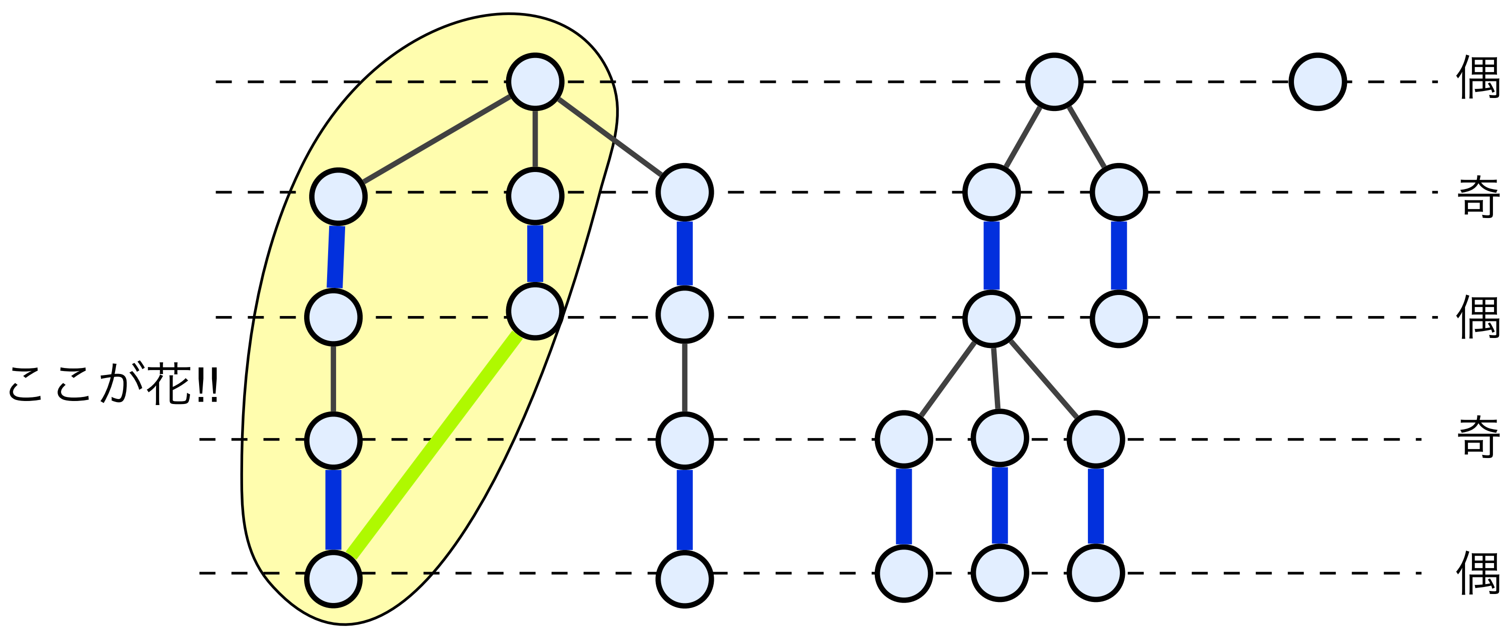


4.同じ交互木中の偶点

グラフ中の花を縮約したグラフをG’とする．この時，Gの最大マッチングのサイズとG’の最大マッチングのサイズが同じになる．

証明は前半のものとほぼ同じ

見つけた花を縮約して，交互木の拡張を続ける．

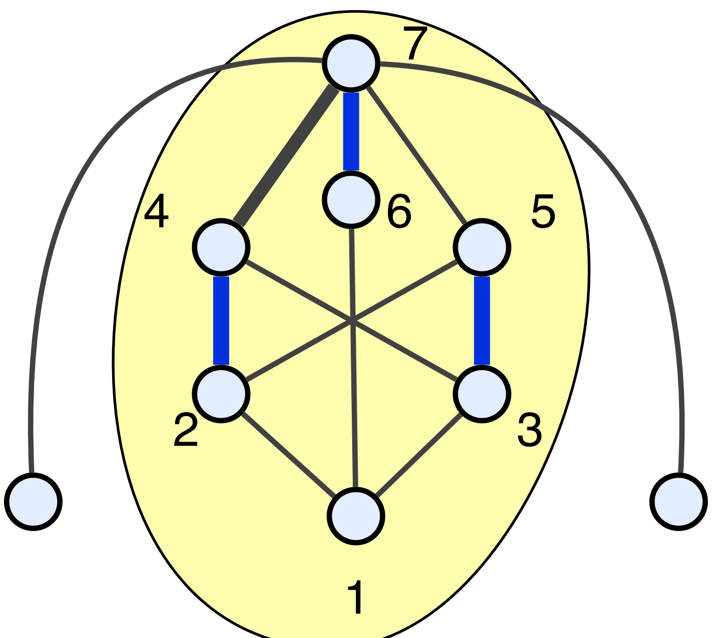


縮約後の交互森中でのM-増加パスの復元

縮約を繰り返すことで，M-増加パスの存在判定ができる．

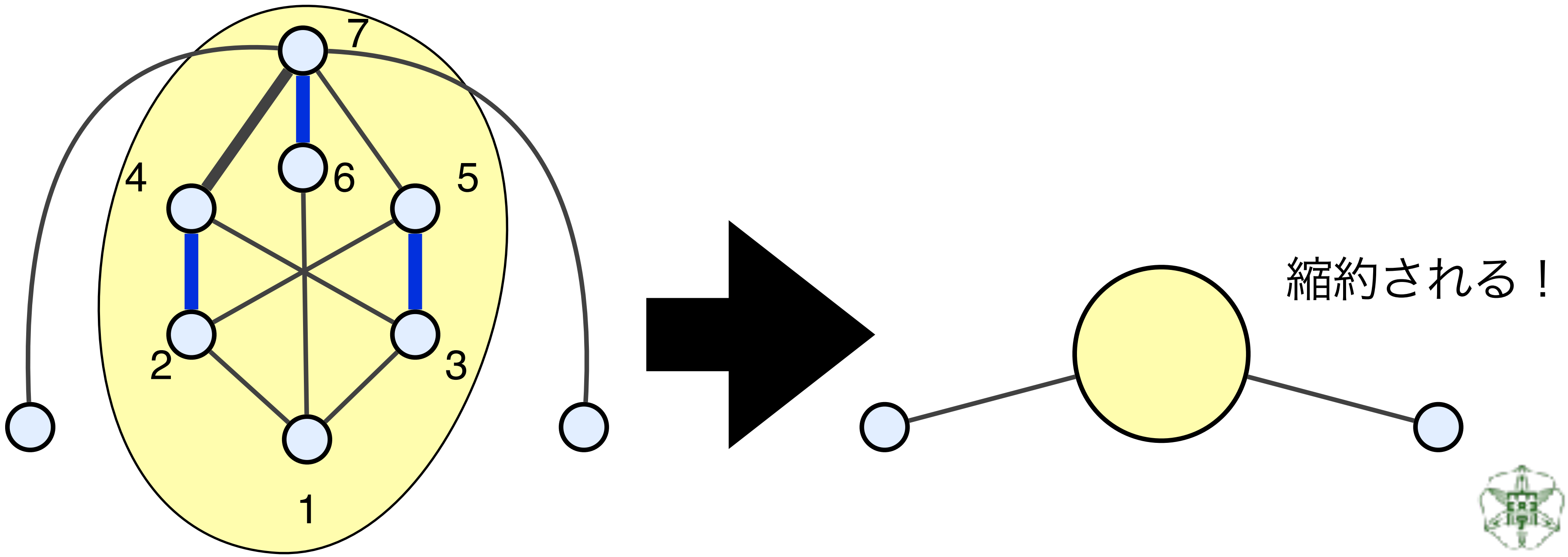
縮約とM-増加パスの発見回数はO(n)回なので，復元が線形時間ならば，全体でO(n(n + m))時間になる．

どうやってO(n + m)時間で復元しよう？



復元の難しい例

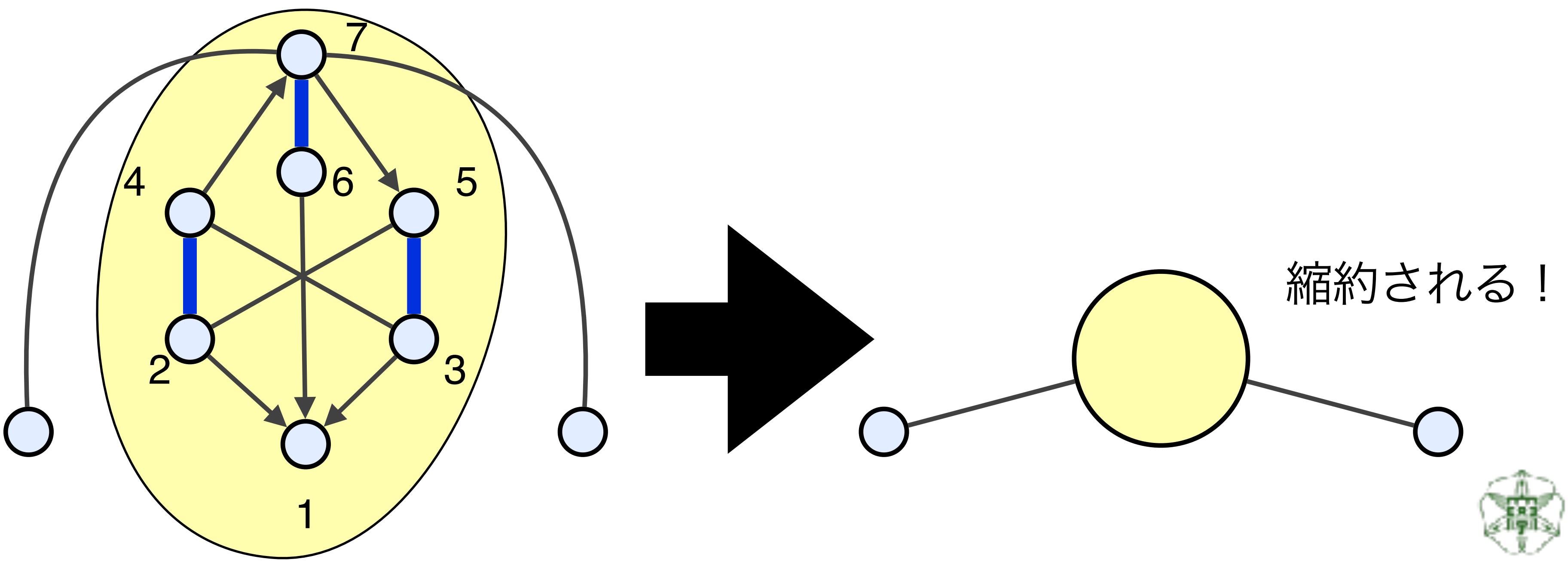
縮約後のグラフG'からどうやってM-増加パスを発見するか？



復元が難しい理由

M-増加パスを復元する場合，マッチングに含まれる辺は選択の余地なく選べる．マッチングに含まれない辺を決定するのが難しい．

したがって，それぞれの頂点に次にどこにいくかを決めて一意性を保つようにする．

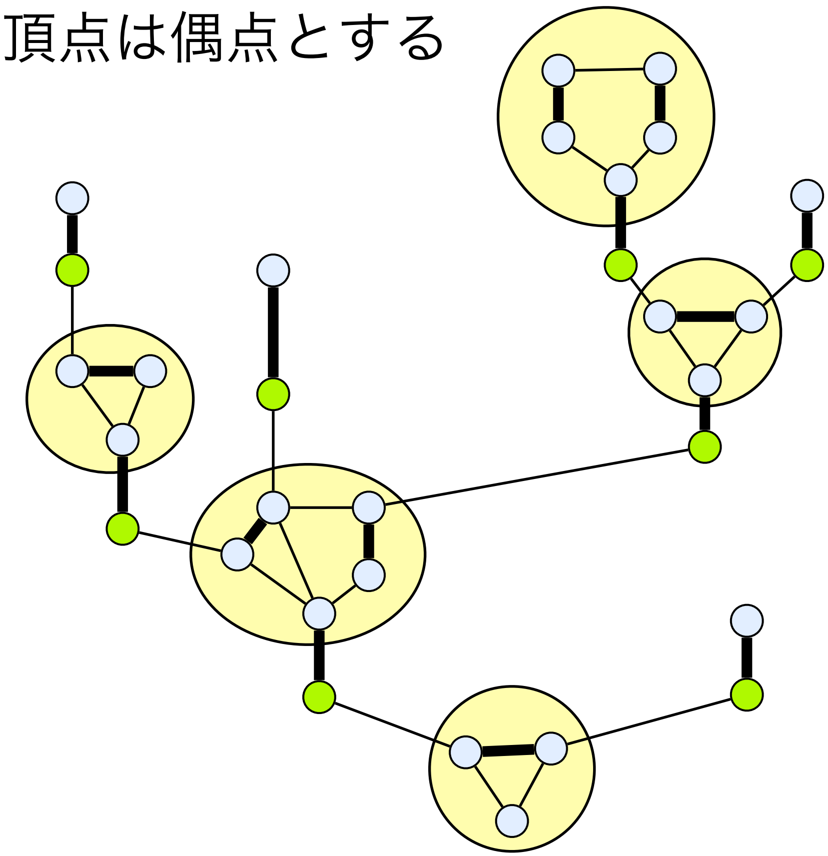


縮約後のグラフ

縮約後のグラフは花Cを縮約したグラフである．この時，**Cの縮約に対応する頂点は偶点**になる．

この**花Cを偶花**という．

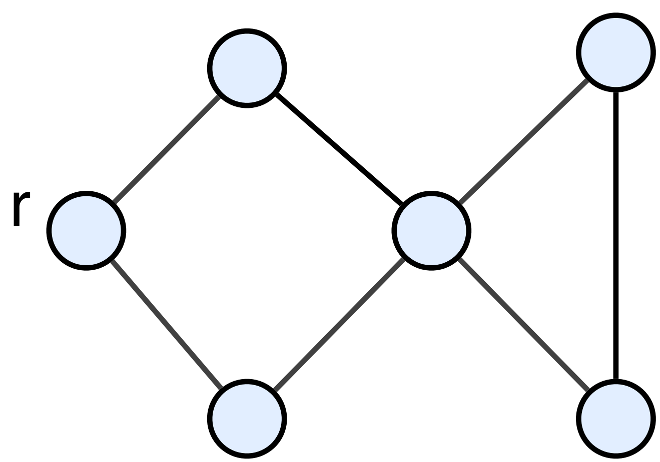
偶花に含まれる頂点は偶点とする



復元のためのデータ構造: M-交互耳分解

M-増加パスを復元できるようにするため，**花をM-交互耳分解**する.

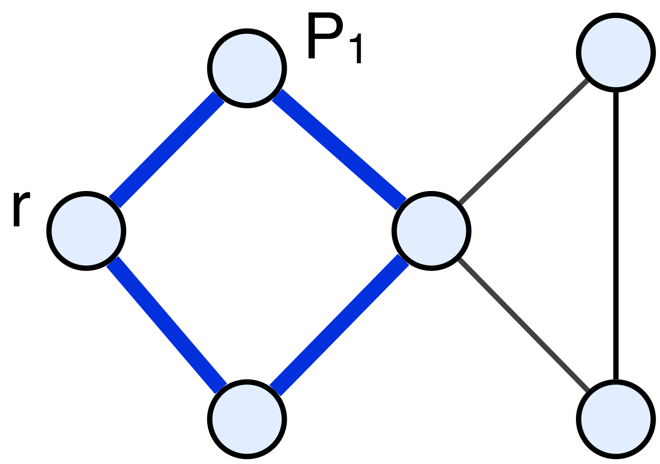
耳分解: グラフを$\{r\} \cup P_1 \cup P_2 \ldots$というように，いくつかのパスに分解すること．ここで，P_iは$\{r\} \cup P_1 \cup \ldots P_{i-1}$を中の頂点を端点にもつ.



復元のためのデータ構造: M-交互耳分解

M-増加パスを復元できるようにするため，**花をM-交互耳分解**する.

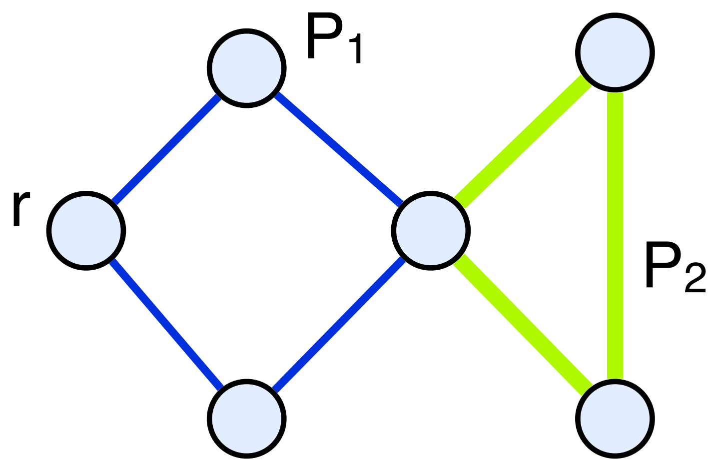
耳分解: グラフを$\{r\} \cup P_1 \cup P_2 \ldots$というように，いくつかのパスに分解すること．ここで，P_iは$\{r\} \cup P_1 \cup \ldots P_{i-1}$を中の頂点を端点にもつ.



復元のためのデータ構造: M-交互耳分解

M-増加パスを復元できるようにするため，**花をM-交互耳分解**する.

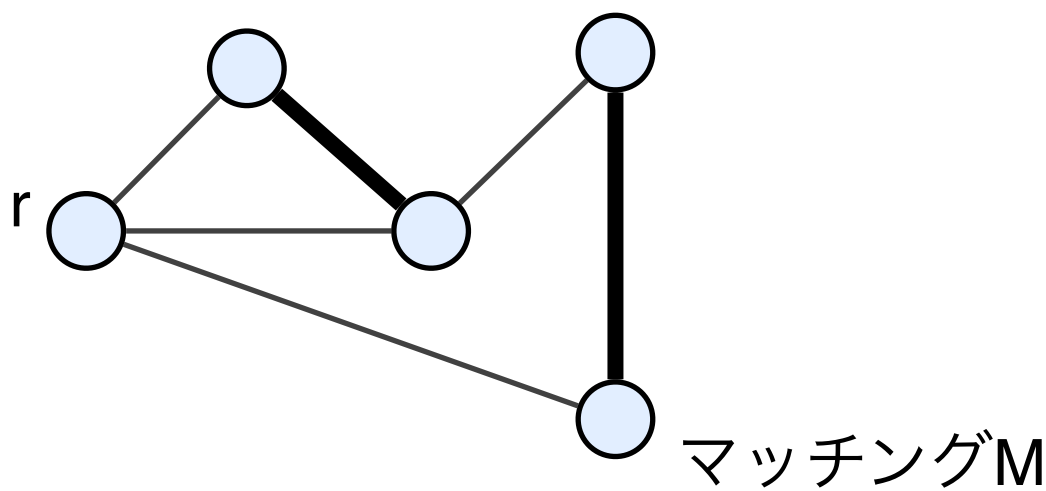
耳分解: グラフを$\{r\} \cup P_1 \cup P_2 \ldots$というように，いくつかのパスに分解すること．ここで，P_iは$\{r\} \cup P_1 \cup \ldots P_{i-1}$を中の頂点を端点にもつ.



復元のためのデータ構造: M-交互耳分解

M-増加パスを復元できるようにするため，**花をM-交互耳分解**する.

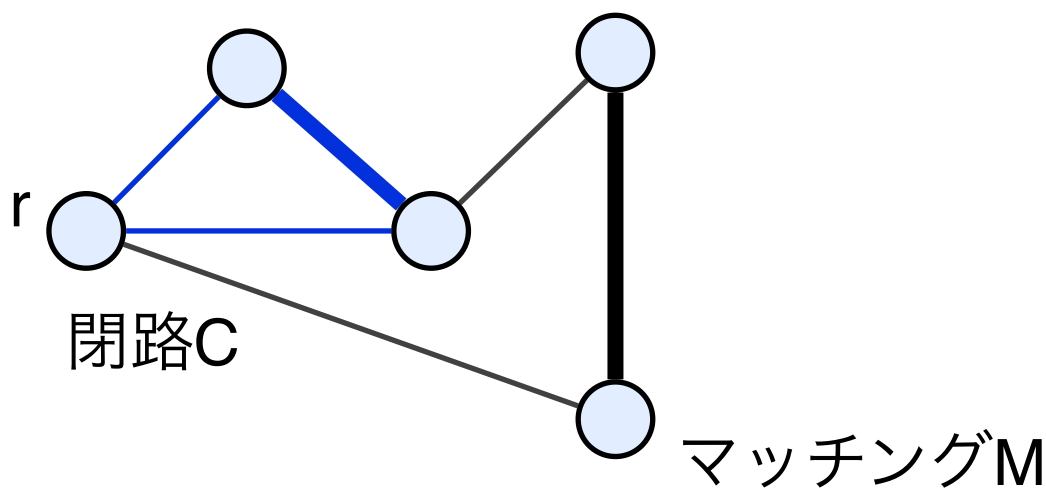
M-交互耳分解: 耳分解の各パスがM-交互パス，または $|E(C) \cap M| + 1 = |E(C) \setminus M|$ になる閉路Cである耳分解



復元のためのデータ構造: M-交互耳分解

M-増加パスを復元できるようにするため，**花をM-交互耳分解**する.

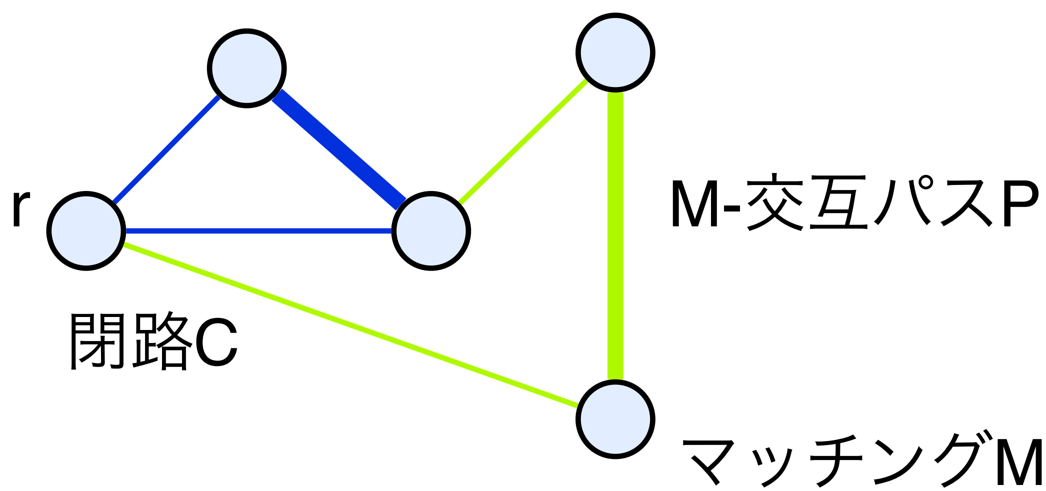
M-交互耳分解: 耳分解の各パスがM-交互パス，または $|E(C) \cap M| + 1 = |E(C) \setminus M|$になる閉路Cである耳分解



復元のためのデータ構造: M-交互耳分解

M-増加パスを復元できるようにするため，**花をM-交互耳分解**する.

M-交互耳分解: 耳分解の各パスがM-交互パス，または $|E(C) \cap M| + 1 = |E(C) \setminus M|$ になる閉路Cである耳分解



復元のためのデータ構造: M-交互耳分解

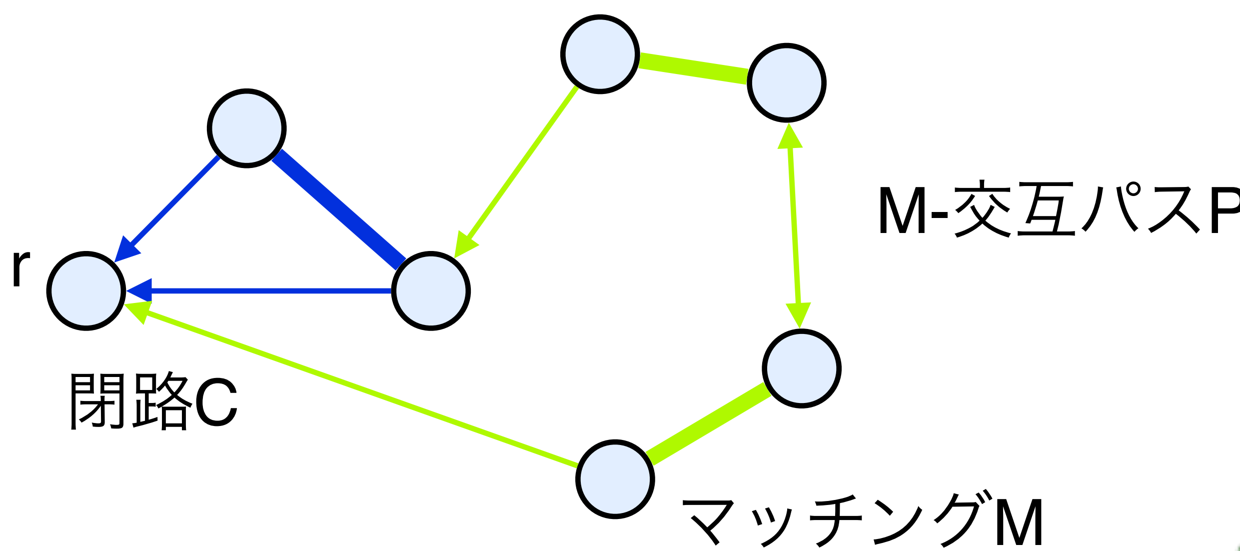
$\mu(x)$: マッチングMに含まれる辺に対応する頂点を返す.

$\psi(x)$: M-交互パスでMに含まれない辺に対応する頂点を返す.

M-交互耳分解は**O(n + m)時間で計算可能.**

性質2:

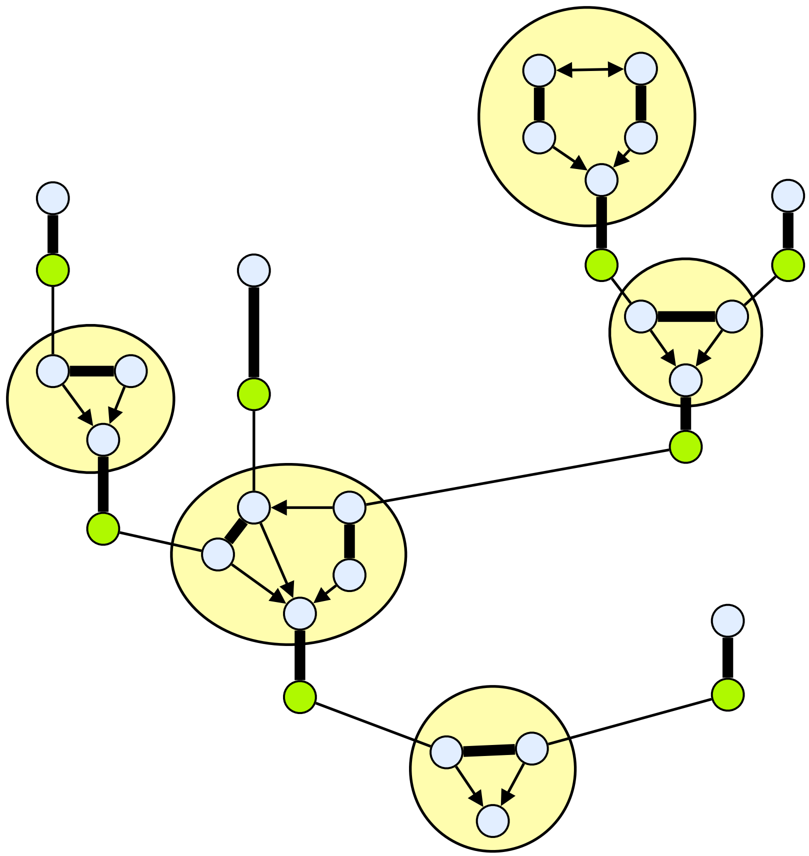
Gが全てのP_iが奇数長のM-交互耳分解を持つ ↔ Gが因子臨界的グラフである.



M-増加パスの復元

縮約前グラフのM-交互耳分解を持っていると簡単にM-増加パスが計算できる.

これで復元も線形時間！



Edmonds アルゴリズムのまとめ

Edmonds アルゴリズムは次のように実行される.

1. Mにカバーされない頂点からなる，辺を持たない交互森を作る.
2. 交互木の偶点の隣接点を見て，交互森を拡張する.
3. 縮約操作，またはM-増加パスを発見した場合は1に戻る.

全体としては，縮約と増加が発生する回数がO(n)回で，毎回の縮約，またはM-増加パスの発見にO(n + m)なので，全体で$O(n^3)$時間アルゴリズムになる.

問題例

[立命館合宿2018 Day2 G問題]

終わり

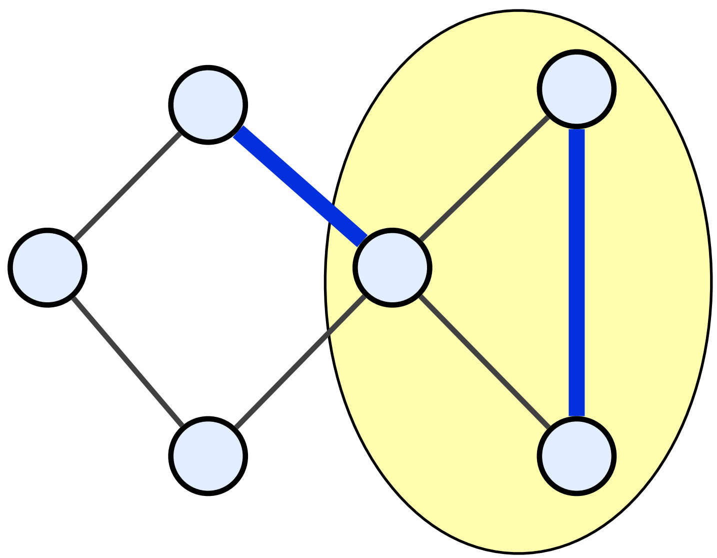
お疲れ様でした！！！！！！！

グラフの因子臨界的部分グラフと花分解

因子臨界的グラフ: グラフのどんな頂点を削除してもグラフは完全マッチングを持つ.

- 先ほどのサイクルCをより一般化した概念

花: G中で$|M \cap E(H)| = (|V(H)| - 1)/2$を満たす因子臨界的部分グラフH（HはGの誘導部分グラフ）



ここはGの花！

グラフの交互森

交互森: グラフGとマッチングMに対して，森Fが次の3つの条件を満たす時，Fを交互森という.

(a) V(F)はMでカバーされない点を全て含み，F喉の連結成分もMでカバーされない点をちょうど1個含む．この頂点を根という.

(b) Fのどの奇点も次数が2である.

(c)任意の頂点vに対して，vからvを含む連結成分の根までのパスはM-交互パスである.

vから根までの距離が偶数の時，vを偶点，そうでない時vを奇点という.