

04/07/12

gSpanを用いた 部分グラフ構造マッチング

岡野原 大輔

參考資料

- ▶ 「gSpan: Graph-Based Substructure Pattern Mining」
Xifeng Yan Jiawei Han

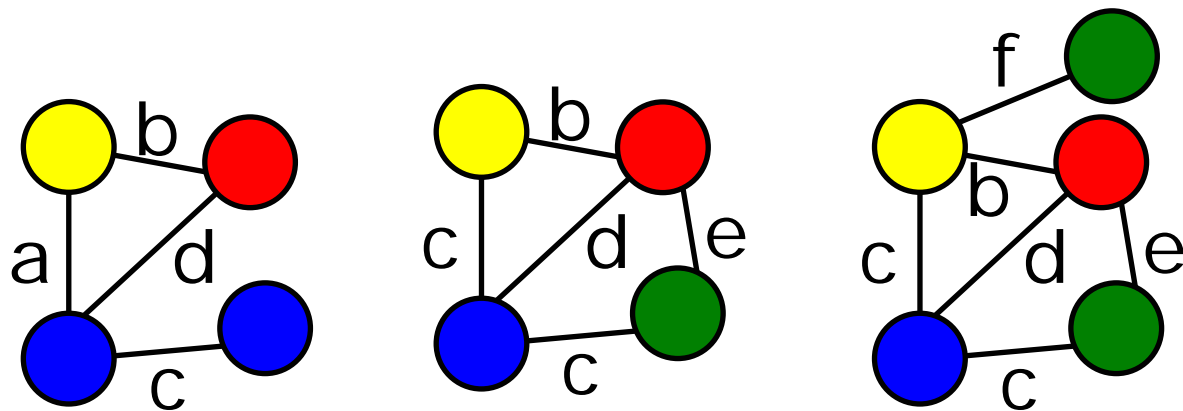
背景と問題設定

- ▶ 複数の無向グラフが与えられた時、頻出する部分グラフを報告する。
- ▶ グラフは根本的なデータ構造なのであらゆる分野で、この頻出する部分グラフ探索は利用される。
 - ▶ 化学、生命科学分野における分子解析
 - ▶ コンピュータービジョンでの認識
 - ▶ ビデオのindex化
 - ▶ テキスト検索

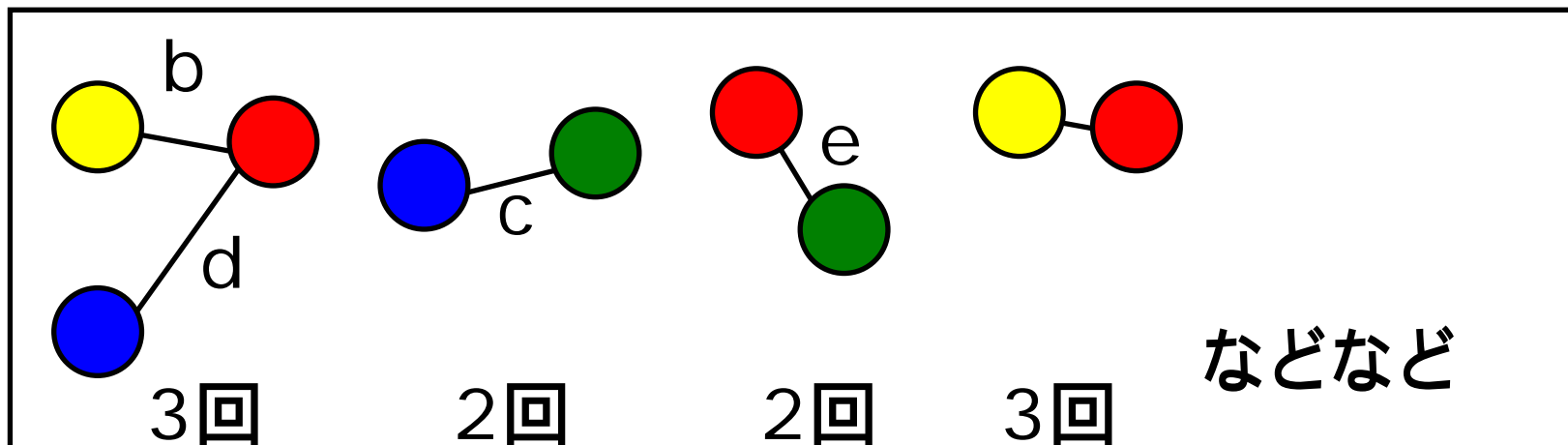
正式な問題設定

- ▶ 最小出現回数 minSup とする。
- ▶ $(g, G) = 1$ (g は G の subGraph)
 0 (g は G の subGraph ではない)
- $(g, GS) = \sum_{G \supseteq GS} (g, G)$
としたとき、 $(g, GS) > \text{minSup}$ となる、 g を全て報告せよ。
- ▶ minSup 回以上出現する部分グラフを frequent と呼び、そうでない部分グラフを infrequent と呼ぶ。

mining frequent subgraph patterns



minSup = 2



問題の難しさと解決への手法

- ▶ 同形判定(NP-complete)は非常に計算量がかかる。
- ▶ 候補となる部分グラフ生成にも計算量がかかる。



- ▶ 候補となる部分グラフを木構造として管理し、DFSを用いて頻出する部分グラフを網羅する。
さらに、無駄な候補グラフは生成しないように適切な枝狩りを行う。この際、枝刈りによって本来なら選ばれるはずの部分グラフが調べられないことのないような工夫が必要(アルゴリズムの完全性)

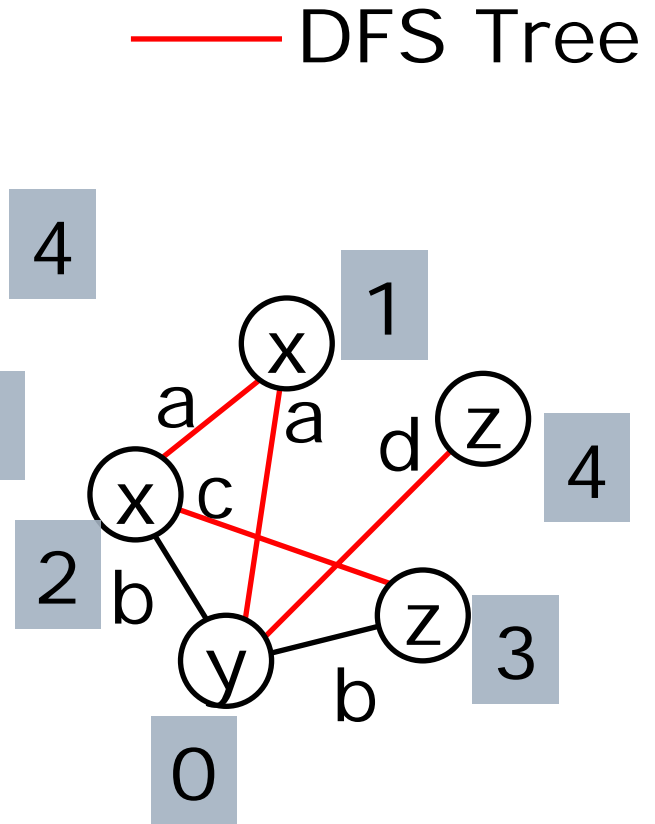
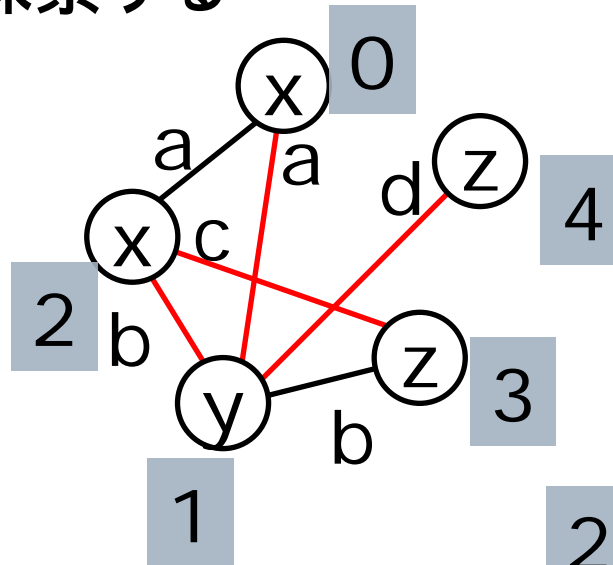
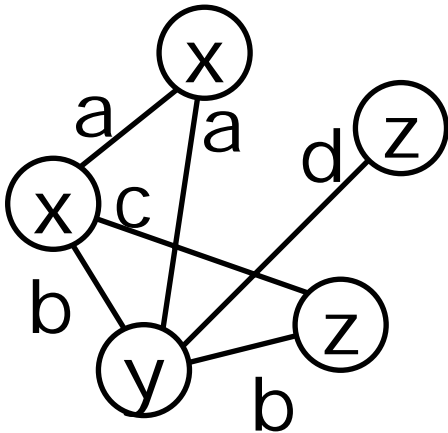
方針

- ▶ グラフをDFSした時に生成された木 (DFSで新しく訪れた節点に対する枝のみから生成される木) を用いる。この木をDFS Treeと呼ぶ。
- ▶ DFS Treeから決定されるDFS Codeをそのグラフを表現するCodeとする。
- ▶ 同一のグラフには複数のDFS Codeが与えられる可能性があるので、最小(定義は後で)のDFS Code、Minimum DFS Codeのみを考える。
- ▶ 全てのグラフに唯一のDFS Codeが決定される。

- ▶ このDFS Code間で決定される木 (子供は親節点が表現するグラフに一本枝と節点を加えたもの) においてDFS行う。この際、Minimum ではないDFS Codeを持つ節点は枝刈りすることにより、無駄な候補の生成を防ぐ。

DFS TreeとDFS Code

▶ グラフをDFSで探索する



— DFS Tree

▶ 開始点及び、枝の伸ばし方で複数のDFS Treeが定義される。

DFS Treeにおける枝の順序

- ▶ DFSで訪問した順で、頂点に番号を付ける (前項の四角内の番号)。
- ▶ DFS Treeに含まれる枝は $e = (i, j)$ ($i < j$)、含まれない枝は $e = (i, j)$ ($i > j$)とする。
- ▶ 全ての枝について、この頂点番号に基づくラベルに基づいて、次のように順序付けを行う ($<_{E,T}$)

$$e_1 = (i_1, j_1) \quad e_2 = (i_2, j_2)$$

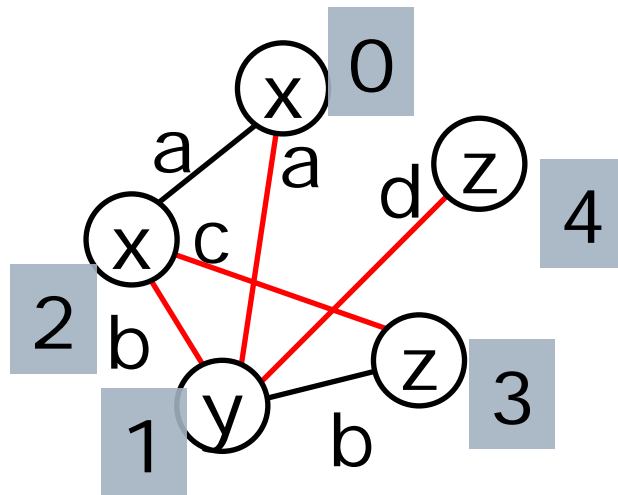
(i) $i_1 = i_2$ かつ $j_1 < j_2$ のとき $e_1 <_{E,T} e_2$

(ii) $i_1 < i_2$ かつ $j_1 = i_2$ のとき $e_1 <_{E,T} e_2$

(iii) $e_1 < e_2$ かつ $e_2 < e_3$ ならば $e_1 <_{E,T} e_3$

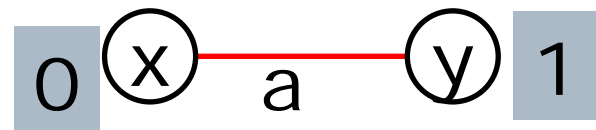
DFS Code

- ▶ DFS Treeにおいて順序付けされた各枝を、 $e_i <_{E,T} e_{i+1}$ ($i = 0 \dots |E| - 1$)を満たすように並べたものをDFS Codeと呼ぶ。
- ▶ 二ページ前のDFS Codeの例
 - ▶ $(0, 1)(1, 2)(2, 0)(2, 3)(3, 1)(1, 4)$



DFS Codeの順序付け

- ▶ 同一のグラフからは複数のDFS Treeが定義され、複数のDFS Codeが生成される。
- ▶ その中から唯一のDFS Codeを選びたいので、DFS Code間に順序付けを行い、最小のものを、そのグラフに対して用いるDFS Codeとする。
- ▶ 各枝に訪問された頂点の番号数のみではなく、両端点のラベル、及び枝のラベルも加える。
- ▶ 例 $(0, 1)$ $(0, 1, x, a, y)$



DFS Codeに対する順序付け(2)

- ▶ $\prec_{E,T}$ 及び、ラベルに関する順序付け \prec_L から辞書式順序 $\prec_e = \prec_{E,T} \times \prec_L \times \prec_L \times \prec_L$ を構成し、DFS Codeの要素間の辞書付き順序を決定する。
- ▶ Z を全てのラベル付き連結グラフから作られたDFS codeの集合とする。
- ▶ Z 中の要素間 $A = \{a_0, a_1, \dots, a_m\}$ $B = \{b_0, b_1, \dots, b_n\}$ に次の(i)または(ii)が成り立つとき、 $A \prec_e B$ とする。
 - (i) $t, 0 \leq t \leq \min(m, n), a_k = b_k$ for $k < t, a_t <_e b_t$
 - (ii) $a_k = b_k$ for $0 \leq k < m$ かつ $n < m$

Minimum DFS Code

- ▶ あるグラフ G が与えられた時、 G より構成される全てのDFS Codeのうち、最小のものを G のMinimum DFS Codeと呼び、これを G の標準のラベル付けとし、 $\min(G)$ と表記する。
- ▶ 二つのグラフ、 G, G' が与えられた時、 $G < G'$
 $\min(G) < \min(G')$ とする。
- ▶ 定理
二つのグラフ、 G, G' で G が G' とisomorphicの時は
 $\min(G) = \min(G')$ の時である。
(証明略)

DFS Code Tree

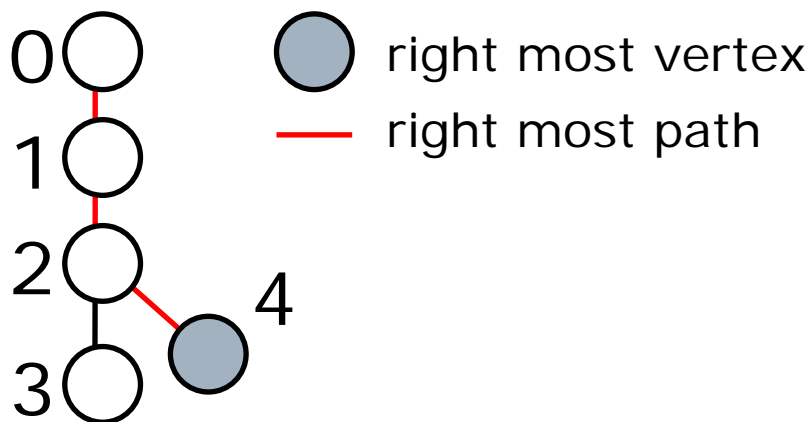
- ▶ Minの話は少し忘れる。
- ▶ 二つのDFS codeが次の場合
$$= \{a_0, a_1, \dots, a_m\}, \quad = \{a_0, a_1, \dots, a_m, b\}$$

は の子と定義し、 は の親と定義する。
- ▶ もし、 のグラフに枝を一本加えて、 の子を作ろうとする場合、任意の場所に枝を加えられない。(不適当な場所に枝を加えると、そのDFS codeは の子ではなくなる)

right most vertex

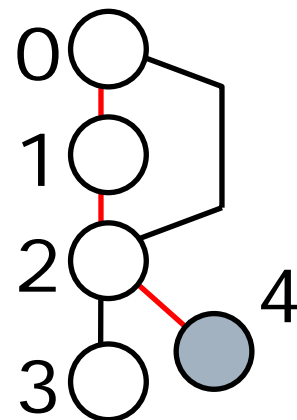
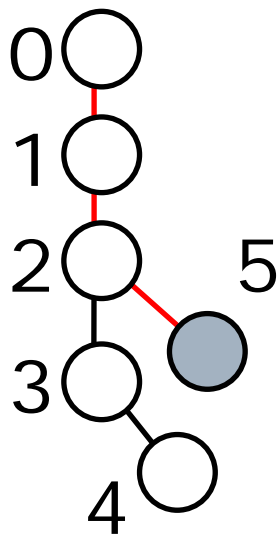
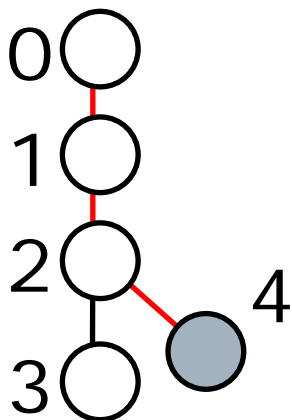
right most path

- ▶ DFS Treeの中で、最も大きい番号をつけられた頂点をrightmost vertex、根からrightmost vertexへのpathをright most pathと呼ぶ



子になる必要条件

▶ 前のグラフに枝を一本加えた時、子になる条件を考える。

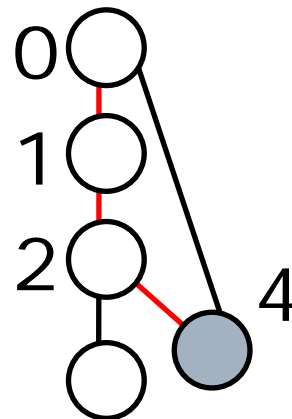
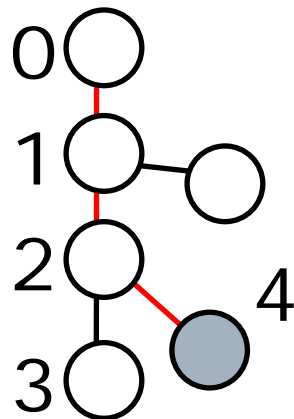


× 3から新しい頂点
へ枝を伸ばす

× 2から1に枝
を伸ばす

子になる必要条件

- ▶ 前向きの枝 (枝の端点の頂点番号 i, j が $i < j$) の場合 right most path から枝を伸ばした場合のみが子になりうる。
- ▶ 後向きの枝 (前向きの枝以外の枝) の場合 right most vertex から伸ばした枝のみが子になりうる。



DFS Code Tree

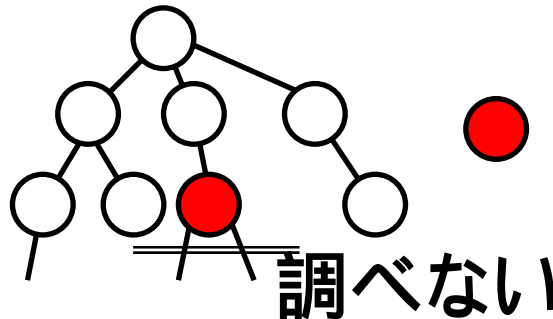
- ▶ DFS Code間に定められた親子関係でDFS Codeによる木が形成される (minとなるDFS Code以外のDFS Codeも含まれる)
- ▶ 兄弟関係はDFS Codeの順序関係で決定される。
- ▶ あるラベルセットが与えられた時、そのラベルセットのみを含むグラフからなるDFS Code Treeが形成される。(問題解決用のDFS Code Tree)

出現頻度に関する非単調性

- ▶ もし、 G の出現回数 $f(G)$ が $f(G) > \text{minSup}$ ならば、 G の全ての部分グラフ $p(G)$ は、 $f(p(G)) > \text{minSup}$ を満たす。
- ▶ もし、 $f(G) < \text{minSup}$ ならば、 G を部分グラフとする全てのグラフ $q(G)$ は $f(q(G)) < \text{minSup}$ を満たす。
- ▶ G のDFS Code Treeに関する祖先は、 G の部分グラフであり、子孫は G を部分グラフとするグラフである。祖先、子孫について上の二つの単調性が成り立つ。

minimum DFS code に関する性質

- ▶ もし、DFS Code Treeのある節点がminimum DFS codeでなければ、その節点の子孫には、minimum DFS codeである節点は存在しない。
- ▶ 全てのminimum DFS codeは、DFS code TreeをDFSで探索してき、もしminimum DFS codeで無い節点に到達したとき、それ以上その節点及び子孫を調べないとしても(枝刈りしても)、全てのminimum DFS codeを含む節点を調べることができる。



● minimum DFS codeではない節点

gSpan Algorithm

- ▶ 入力: グラフの集合GS minSup
----前処理----
- ▶ GSの頂点、枝のラベルを出現頻度順にソートする。
infrequentな頂点、枝を取り除く(出現頻度に関する非単調性)
- ▶ 残りの頂点、枝をソートした結果でラベルを付け直す。
- ▶ $S1 :=$ GS中のfrequentな枝が一つのグラフ
- ▶ S1をDFS codeの順序で小さい順にソート
- ▶ $S := S1$

gSpan Algorithm(2)

```
▶ for each edge e ∈ S1 do
    s := e
    s.GS = {g | g ∈ GS, e ∈ E(g)}
    /* eが含まれるグラフ番号を記録 */
    Subgraph_Mining(GS, S, s)
    GS -= e;
    /* eを含む部分グラフはもう探す
       必要はない */
    if |GS| < minSup
        break;
```

Subgraph_Mining(GS, S, s)

```
▶ if s == min(s)
    return;
/* 既にs、sの子孫のグラフは処理済 */
S := S ∪ {s}
sに一つ枝を加えたsの子供を生成
Enumerate(s);
/* sの子供の部分グラフについてs.GSのうちどれに
含まれているか調べる。support(C)を計算 */
for each c as sの子供 do
    if support(c) >= minSup
        S = C
        Subgraph_Mining(GS, S, s)
```

Enumerate(s)

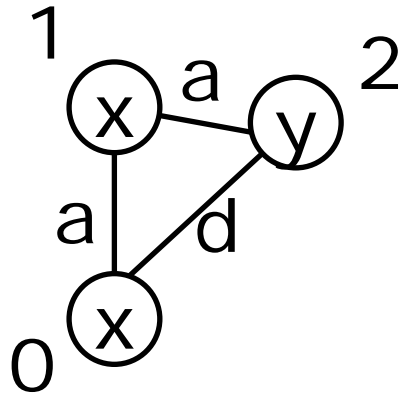
```
▶ for each g ∈ s.GS do
    for each t as g中のsの出現位置 do
        もし、tに枝を一つ加えたものと、sの子
        供(c)が一致する
            c.GS := c.GS ∪ {g};
        if sの全ての子供をgがカバー
            break;
```

Enumerateの補足

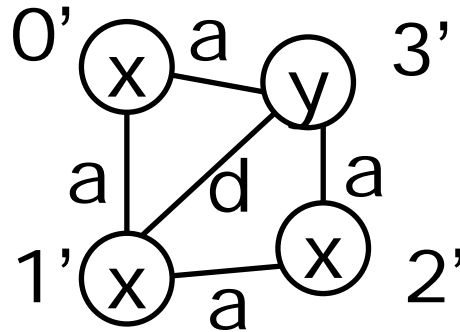
部分グラフの列挙

- ▶ グラフ g の中で部分グラフ s の出現をどのように列挙するか
- ▶ $s.code = s$ のminimum DFS code
 $g.code = g$ のminimum DFS code
- ▶ s の一番上の枝から順に g の枝とマッチングさせる。もしマッチングしたら、次の s の枝のマッチングを調べ、しないなら、バックトラックを行い、 s の次の枝について探す。

部分グラフの列挙

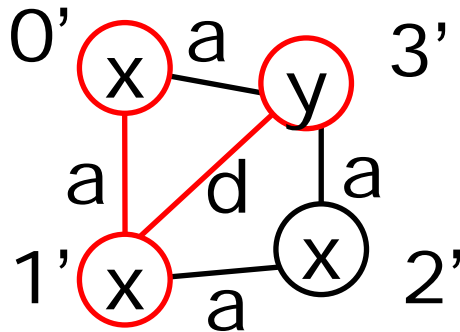
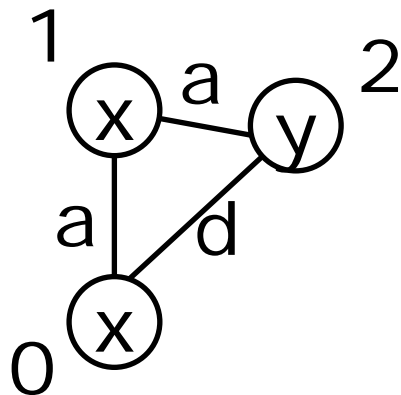


(0, 1, x, a, x)
(1, 2, x, a, y)
(2, 0, y, d, x)



(0', 1', x, a, x)
(1', 2', x, a, x)
(2', 3', x, a, y)
(3', 0', y, a, x)
(3', 1', y, d, x)

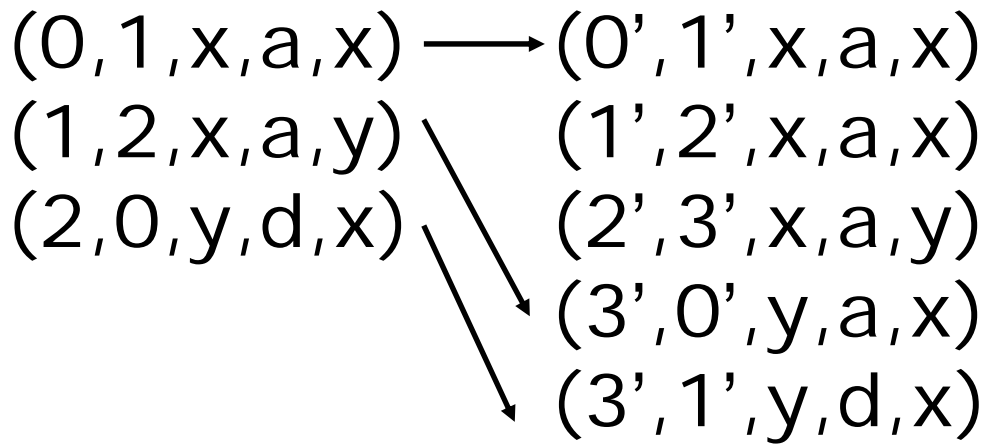
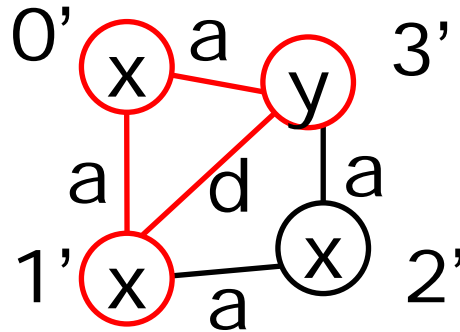
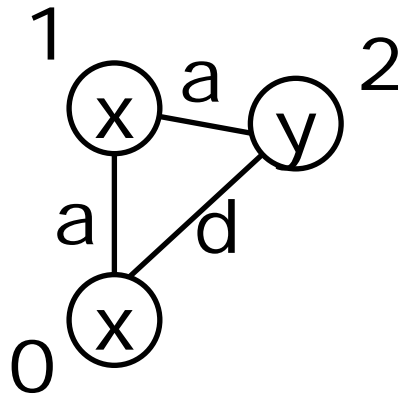
部分グラフの列挙



$(0, 1, x, a, x) \longrightarrow (0', 1', x, a, x)$
 $(1, 2, x, a, y) \longrightarrow (1', 2', x, a, x)$
 $(2, 0, y, d, x) \longrightarrow (2', 3', x, a, y)$
 $(3', 0', y, a, x)$
 $(3', 1', y, d, x)$

$0 = 0'$
 $1 = 1'$
 $2 = 3'$

部分グラフの列挙

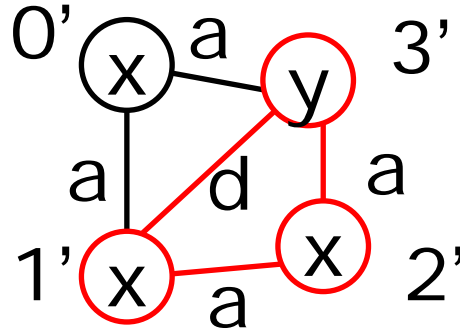
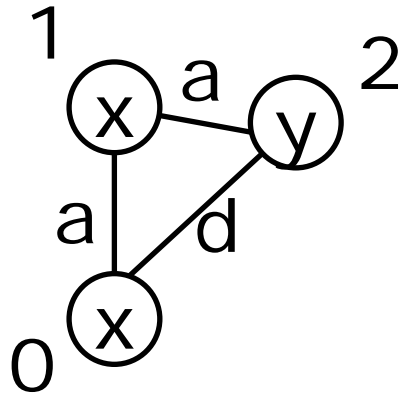


$$0 = 0'$$

$$1 = 1'$$

$$2 = 3'$$

部分グラフの列挙



$(0, 1, x, a, x)$

$(1, 2, x, a, y)$

$(2, 0, y, d, x)$

$(0', 1', x, a, x)$

$(1', 2', x, a, x)$

$(2', 3', x, a, y)$

$(3', 0', y, a, x)$

$(3', 1', y, d, x)$

$0 = 1'$

$1 = 2'$

$2 = 3'$

s min(s)の判定

- ▶ s min(s)はsの全てのDFS codeを生成し、その中でsより小さいDFS codeを持つものが出た時点で判定される。
- ▶ しかし、この手法は非常にコストがかかるために、minとなるための必要条件を用いて、子の候補を生成した直後に判定を行い、大部分のs min(s)となる子の候補を発見し、候補から除外する。

$s = \min(s)$ の必要条件

- ▶ s の最初の枝 e_0 より小さい枝は s のその後には出現しない
- ▶ 新しく付け加えた枝が後ろ向きの枝
 $e = (i, j) (i > j)$ の時、 j に接続する他の枝より小さくはならない。(ラベル順序も含めて)

さらなる子生成の制限

▶ $X = (e_0, e_1, \dots, e_n, e_x)$

▶ $Y = (e_0, e_1, \dots, e_n, e_y)$

▶ どちらかがinfrequentの時

$Z = (e_0, e_1, \dots, e_n, e_x, e_y)$

はinfrequent

(ただし、上のどちらかが正しいIDFS codeではない場合でもZは正しいIDFS codeの時がある)

gSpanの特徴

- ▶ 無駄な候補を作らない
無駄なisomorphismテストも行わない
- ▶ DFSを行うので、メモリ使用量が少ない
(従来のはBFS)
- ▶ 実際のテストデータ、人工的に作ったデータでも10倍から100倍程度速い