ZDDのリンクパズルへの応用

川原 純[†]，斎藤寿樹[†]，鶴間浩二[†]，湊 真一[‡†]，**吉仲 亮**[†]

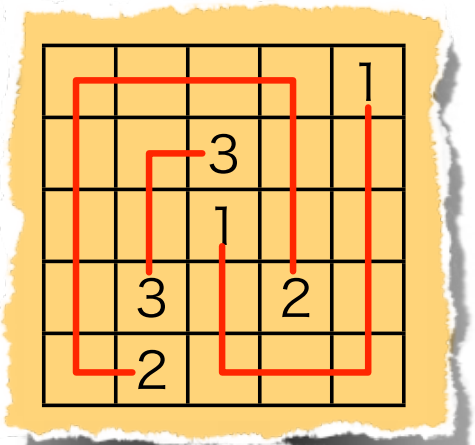
([†]科学技術振興機構ERATO湊離散構造処理系プロジェクト，
[‡]北海道大学大学院情報科学研究科)

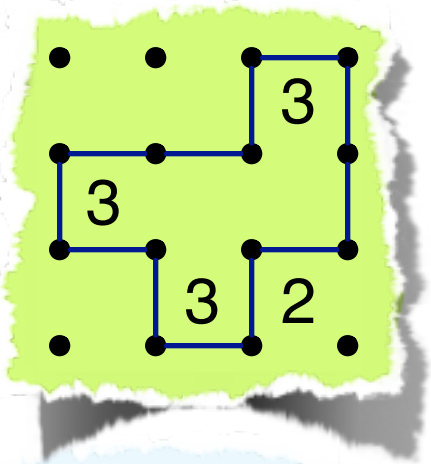
組合せゲーム・パズル ミニプロジェクト
第６回ミニ研究集会
2011年3月10日

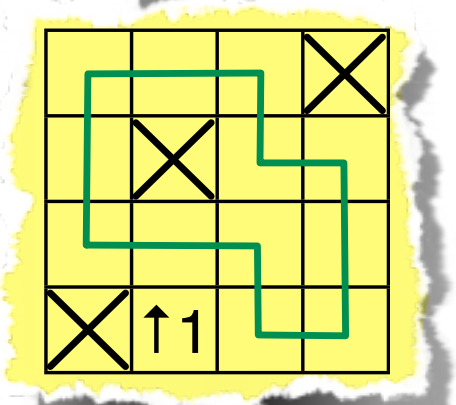
リンクパズル

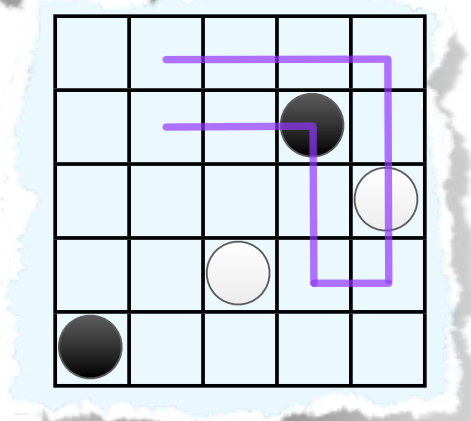
- グラフから次数２以下の部分グラフを抽出
  - ナンバーリンク
  - スリザーリンク
  - ヤジリン
  - ましゅ

    など



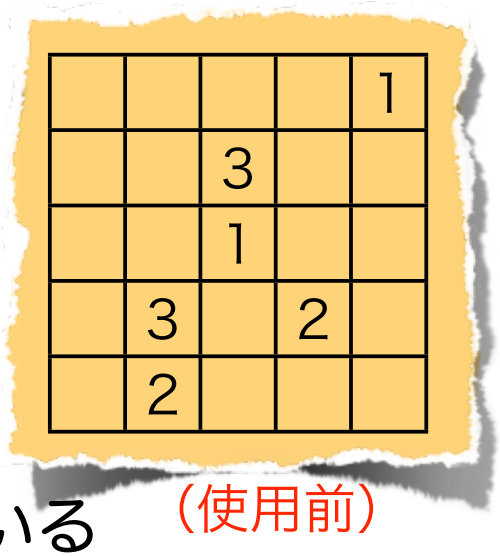




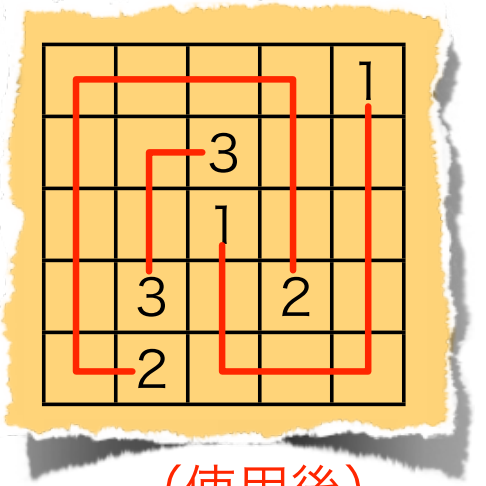


ナンバーリンク

- 問題：
  - 長方形のテーブル
  - 各セルは空白もしくは 1~p の数が入っている
  - 各数はテーブル中にちょうど２回現れる
- 解答：
  - 各セルを縦横に結んで互いに素な
    p 組のパスを作る（パスマッチング）
  - 各パスの２端点が同じ数になる
- NP完全



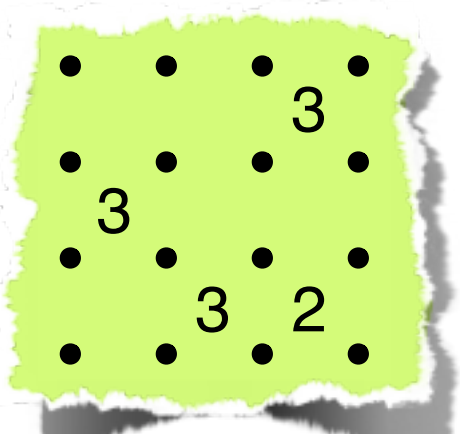
（使用前）



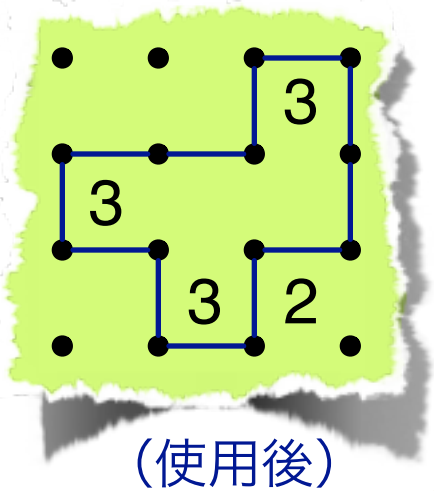
（使用後）

スリザーリンク

- 問題：
  - 長方形のテーブル
  - 各セルは空白もしくは 0~3 の数が入っている
- 解答：
  - 罫線を縦横に結んでサイクルを作る
  - k と書いてあるセルの周りには
    k 本の線が引かれるべき
- NP完全

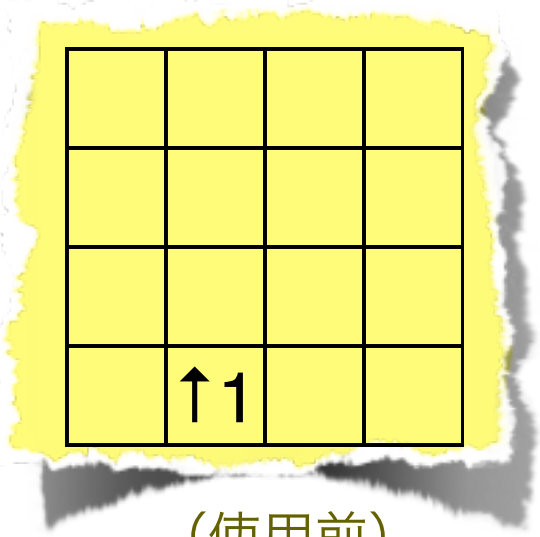


（使用前）

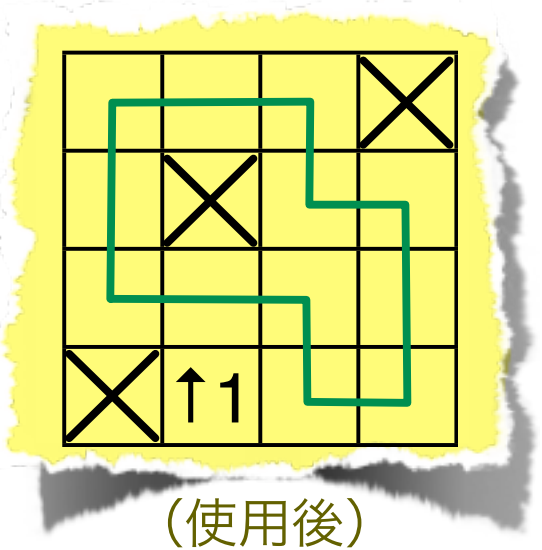


（使用後）

ヤジリン

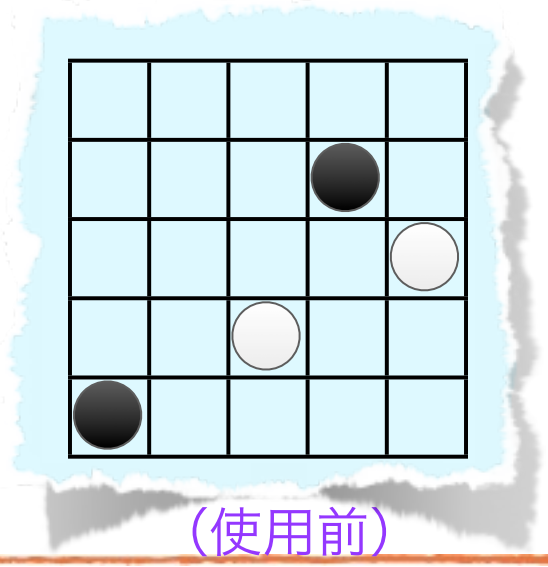


（使用前）

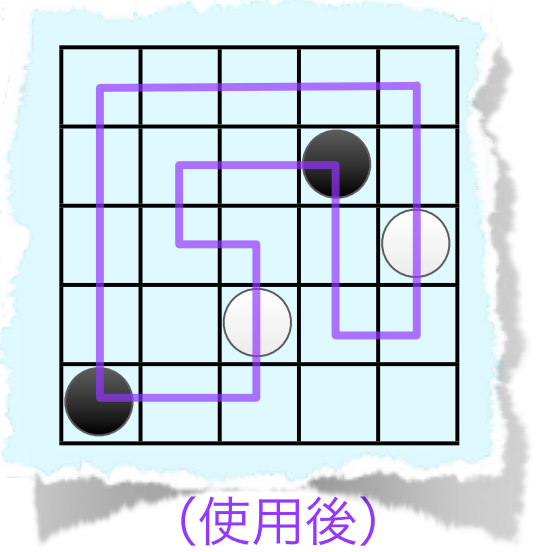


（使用後）

ましゅ



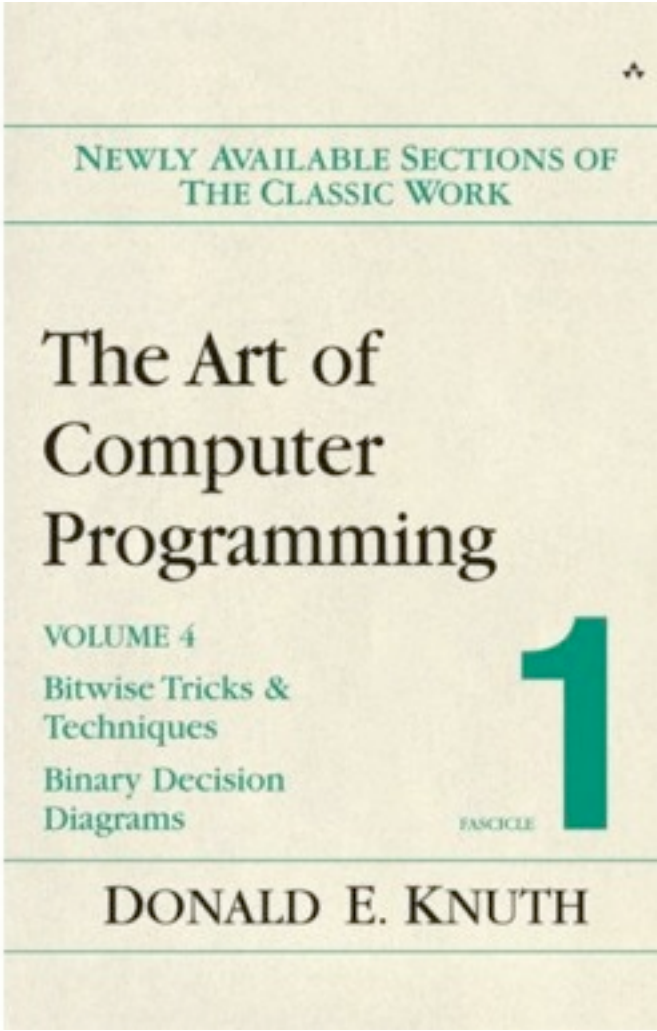
（使用前）



（使用後）

ZDDによるパスの列挙

- D. Knuth によって ZDD を用いたパスやサイクルの列挙手法が提案される（演習問題）
- 多項式回の ZDD の基本演算によるパスの列挙
  （川原ら，冬のLAシンポジウム'10）
- 本発表
  - リンクパズルの解の列挙
  - リンクパズルの問題生成



関連研究

- 古妻浩一・武永康彦
  「OBDDによるナンバーリンクの解法」
  組合せゲーム・パズル第３回ミニ研究集会 (2008)
- 古妻浩一・武永康彦
  「ナンバーリンクのNP完全性と問題の列挙」
  信学技報, vol.109, no.465, COMP2009-49, pp. 1-7, 2010年3月

Zero-suppressed Binary Decision Diagram (ZDD)

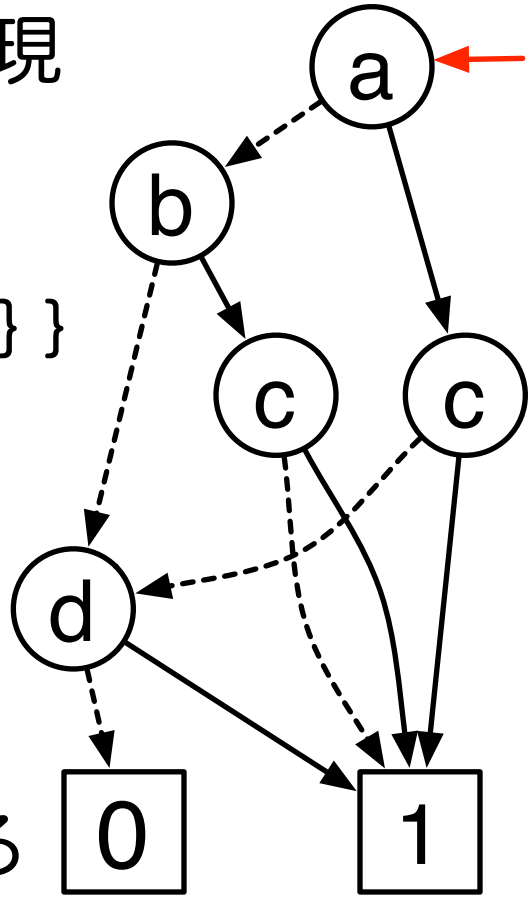
- 組合せ集合をグラフで表現

例：ac＋ad＋b＋bc＋d

{ {a,c}, {a,d}, {b}, {b,c}, {d} }

——→ Yes

- - - - - → No



a<b<c<d の順番で取調べる

Zero-suppressed Binary Decision Diagram (ZDD)

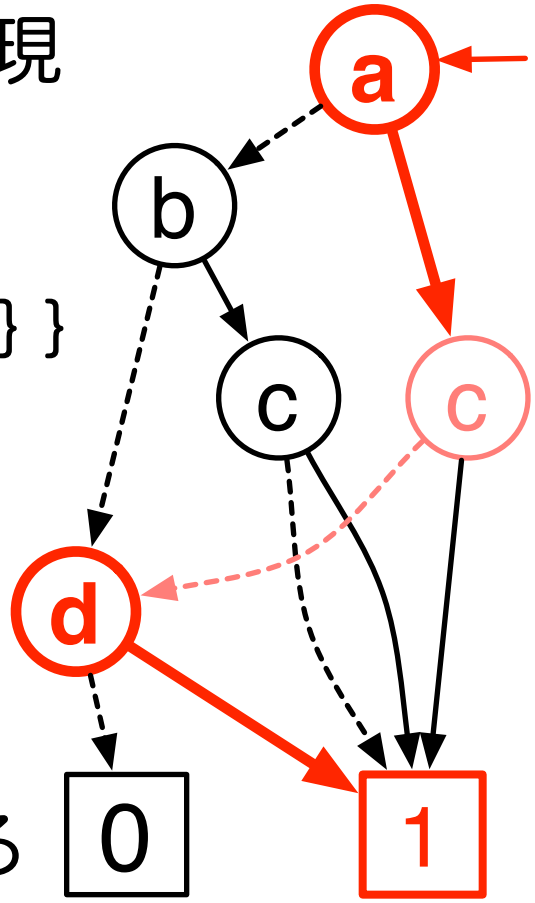
- 組合せ集合をグラフで表現

例：ac＋**ad**＋b＋bc＋d

{ {a,c}, {a,d}, {b}, {b,c}, {d} }

——→ **Yes**

- - - - → No



a<b<c<d の順番で取調べる

Zero-suppressed Binary Decision Diagram (ZDD)

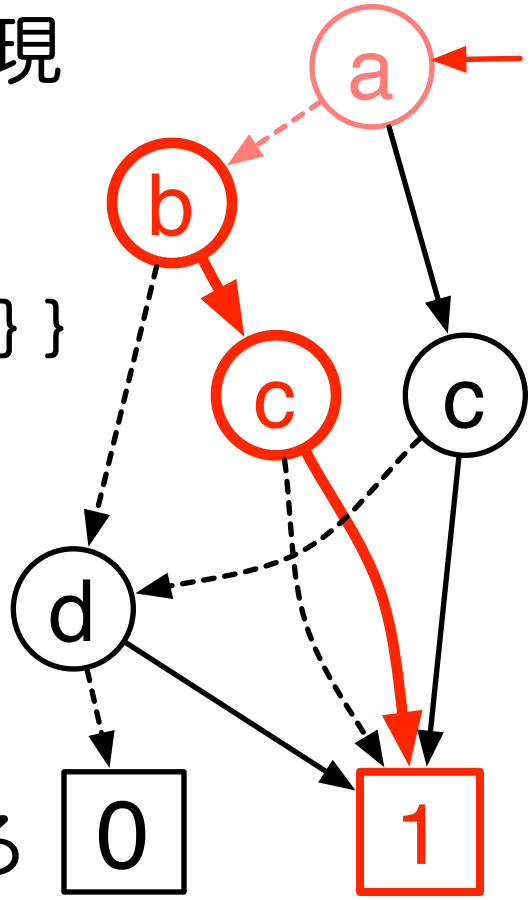
- 組合せ集合をグラフで表現

例：ac＋ad＋b＋**bc**＋d

{ {a,c}, {a,d}, {b}, {b,c}, {d} }

→ **Yes**

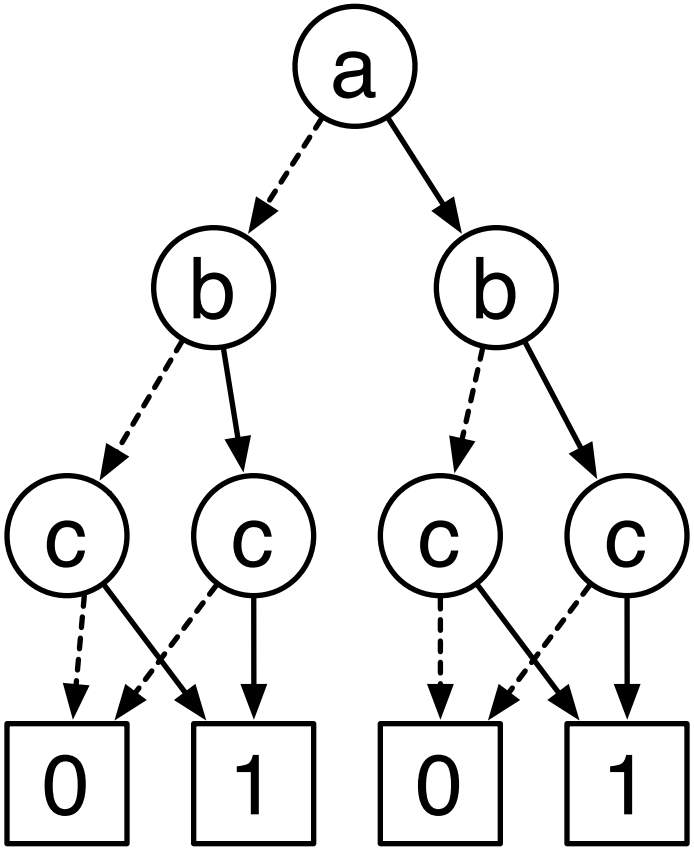
⇢ No



a<b<c<d の順番で取調べる

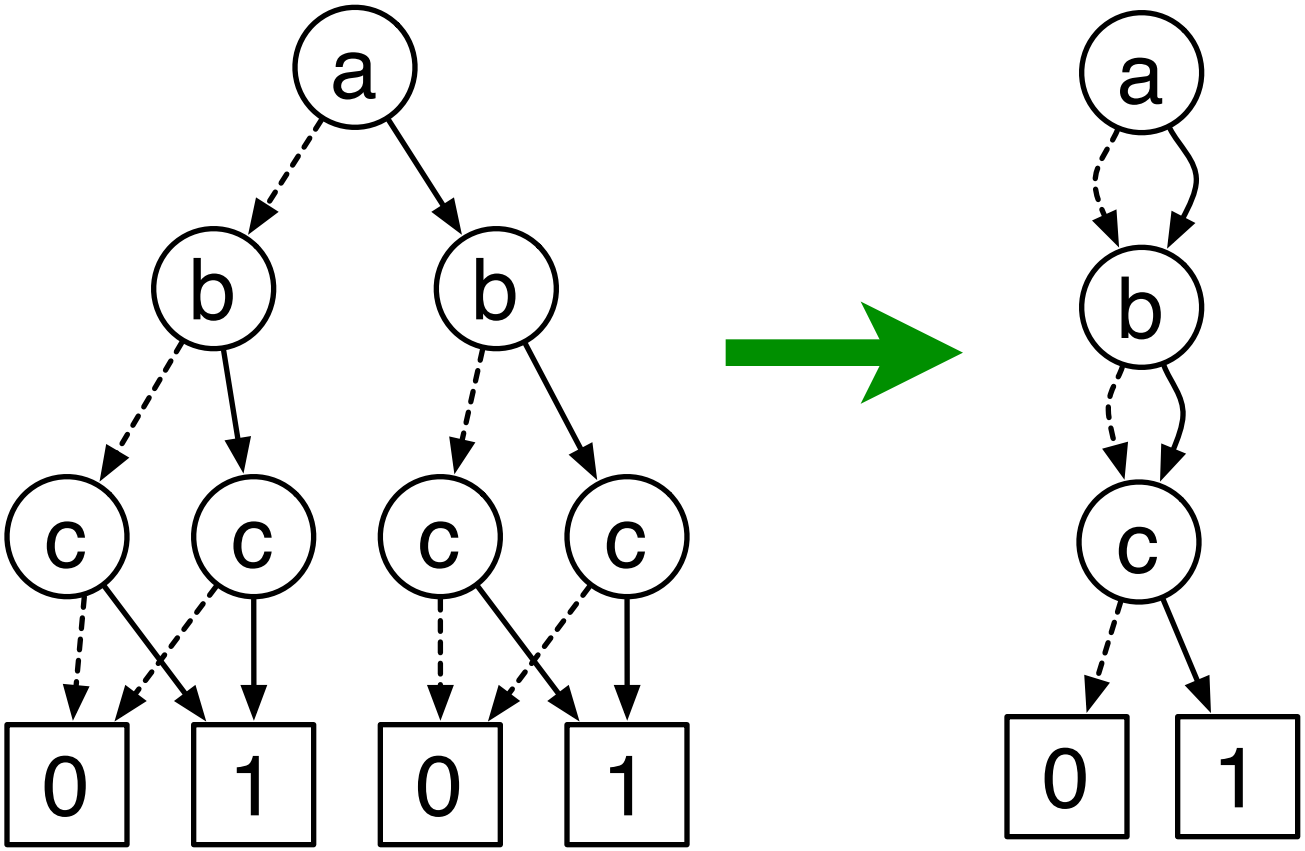
Zero-suppressed Binary Decision Diagram (ZDD)

・同じ働きの節点は２つとない



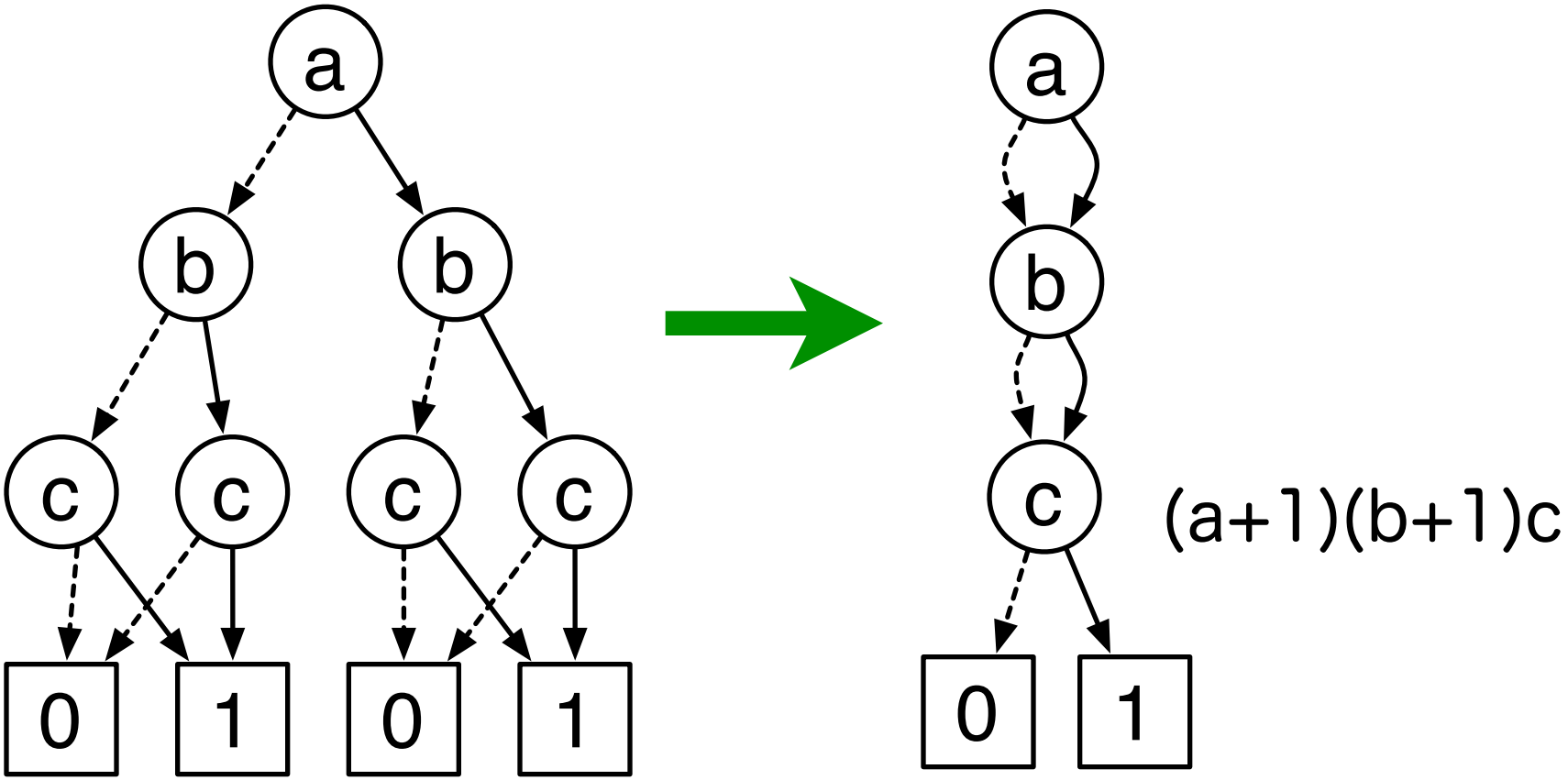
Zero-suppressed Binary Decision Diagram (ZDD)

・同じ働きの節点は２つとない



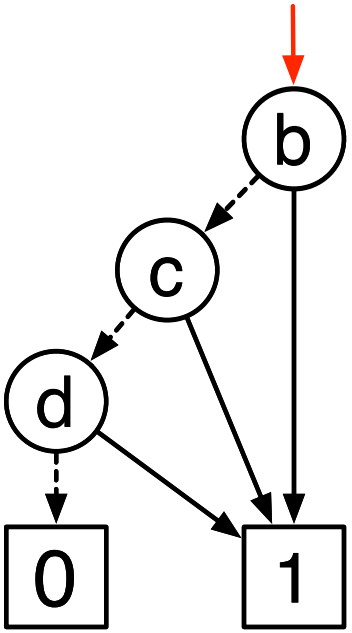
Zero-suppressed Binary Decision Diagram (ZDD)

・同じ働きの節点は２つとない



ZDD の基本代数演算 (1)

$F = b + c + d$



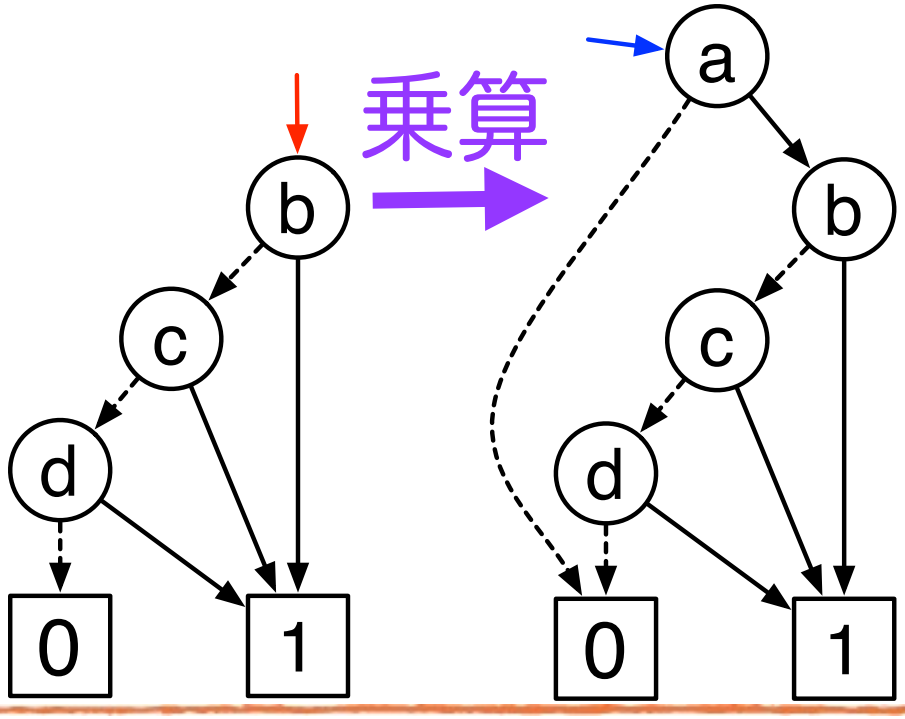
ZDD の基本代数演算 (1)

$F = b + c + d$

$G = aF$

$= ab + ac + ad$

$= a(b + c + d)$

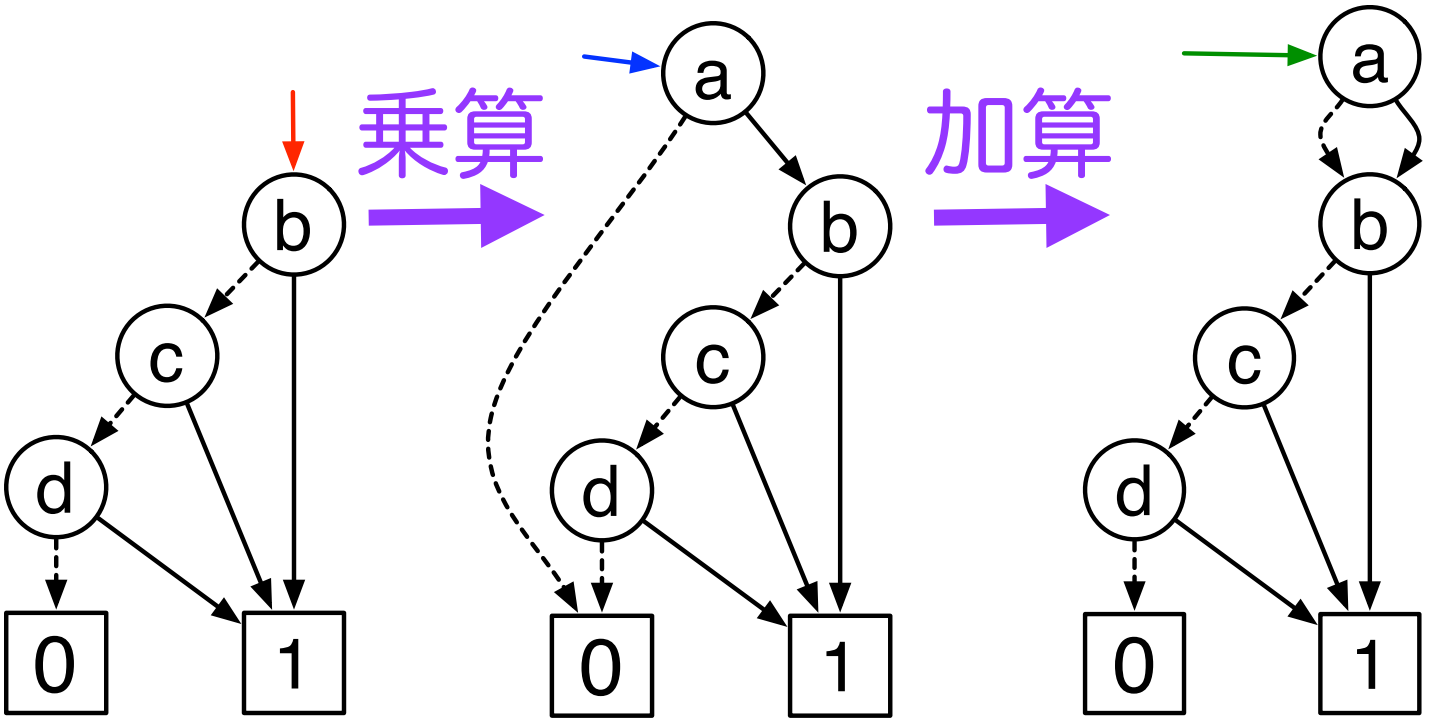


ZDD の基本代数演算 (1)

F = b + c + d

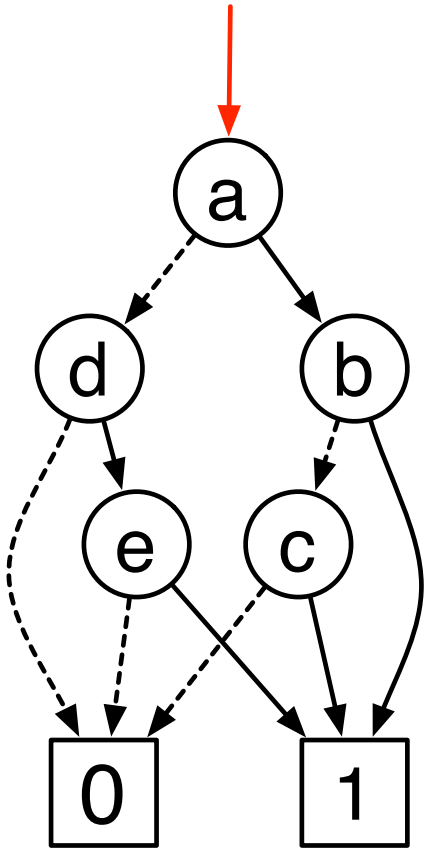
G = aF
= ab + ac + ad
= a(b + c + d)

H = F + G
= ab+ac+ad+b+c+d
= (a+1)(b + c + d)



ZDD の基本代数演算 (2)

$F = a(b + c) + de$

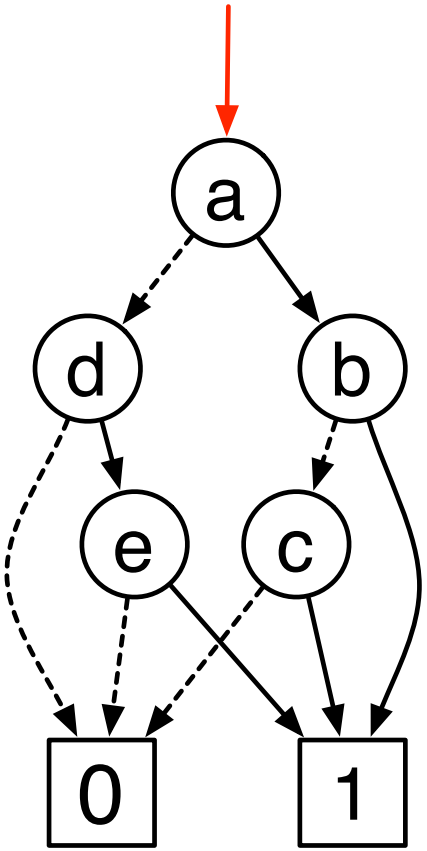


ZDD の基本代数演算 (2)

$F = a(b + c) + de$

除算

$G = F / a = b + c$

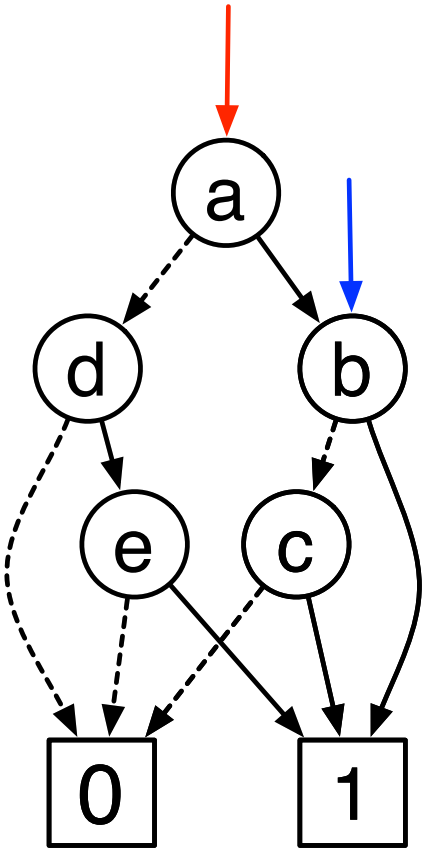


ZDD の基本代数演算 (2)

$F = a(b + c) + de$

除算

$G = F / a = b + c$

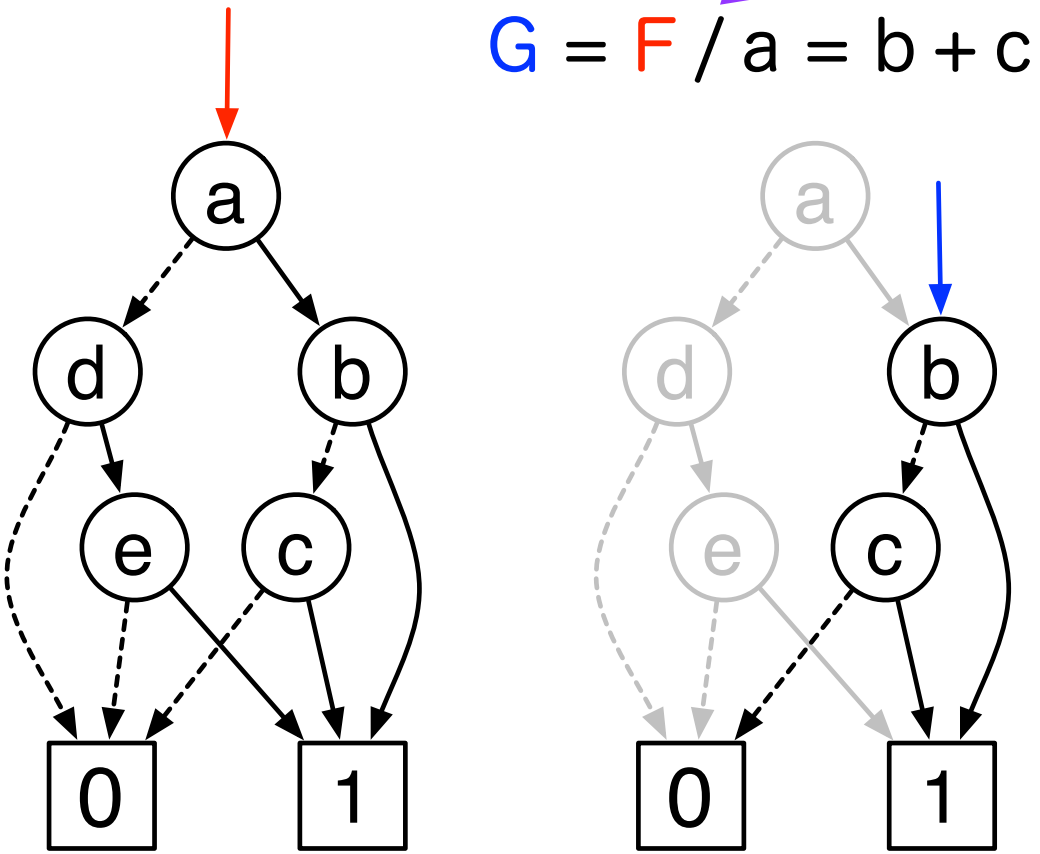


ZDD の基本代数演算 (2)

$F = a(b + c) + de$

除算

$G = F / a = b + c$



ZDD の基本代数演算 (2)

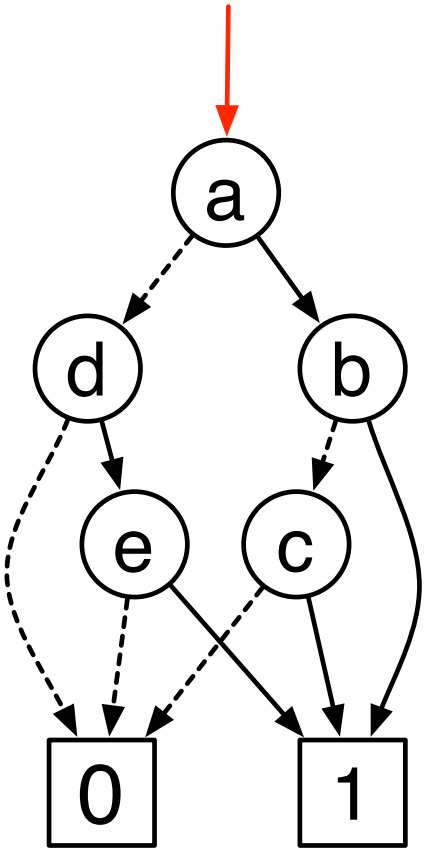
剰余算

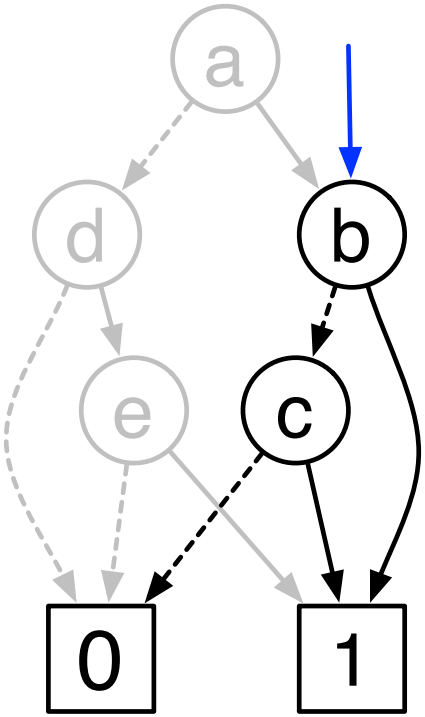
F = a(b + c) + de

除算

H = F % a = de

G = F / a = b + c





ZDD の基本代数演算 (2)

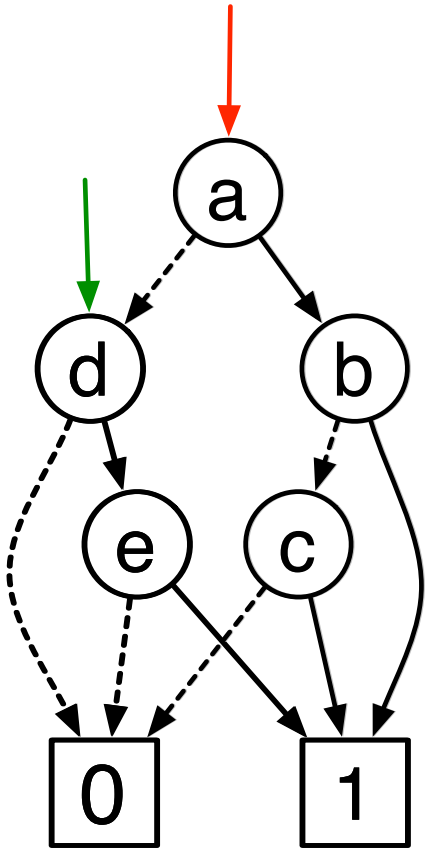
剰余算

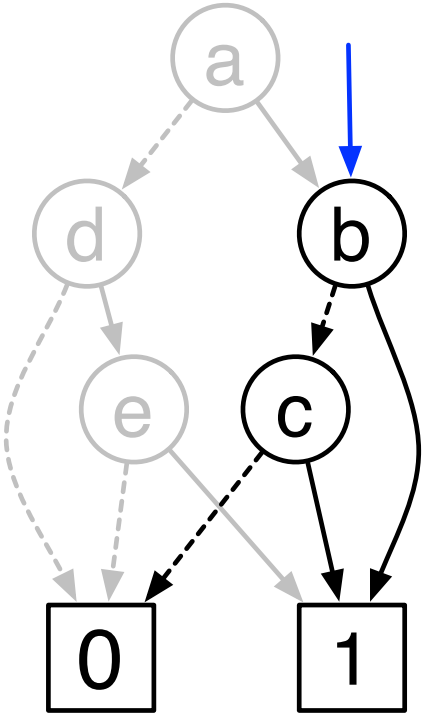
F = a(b + c) + de

除算

H = F % a = de

G = F / a = b + c





ZDD の基本代数演算 (2)

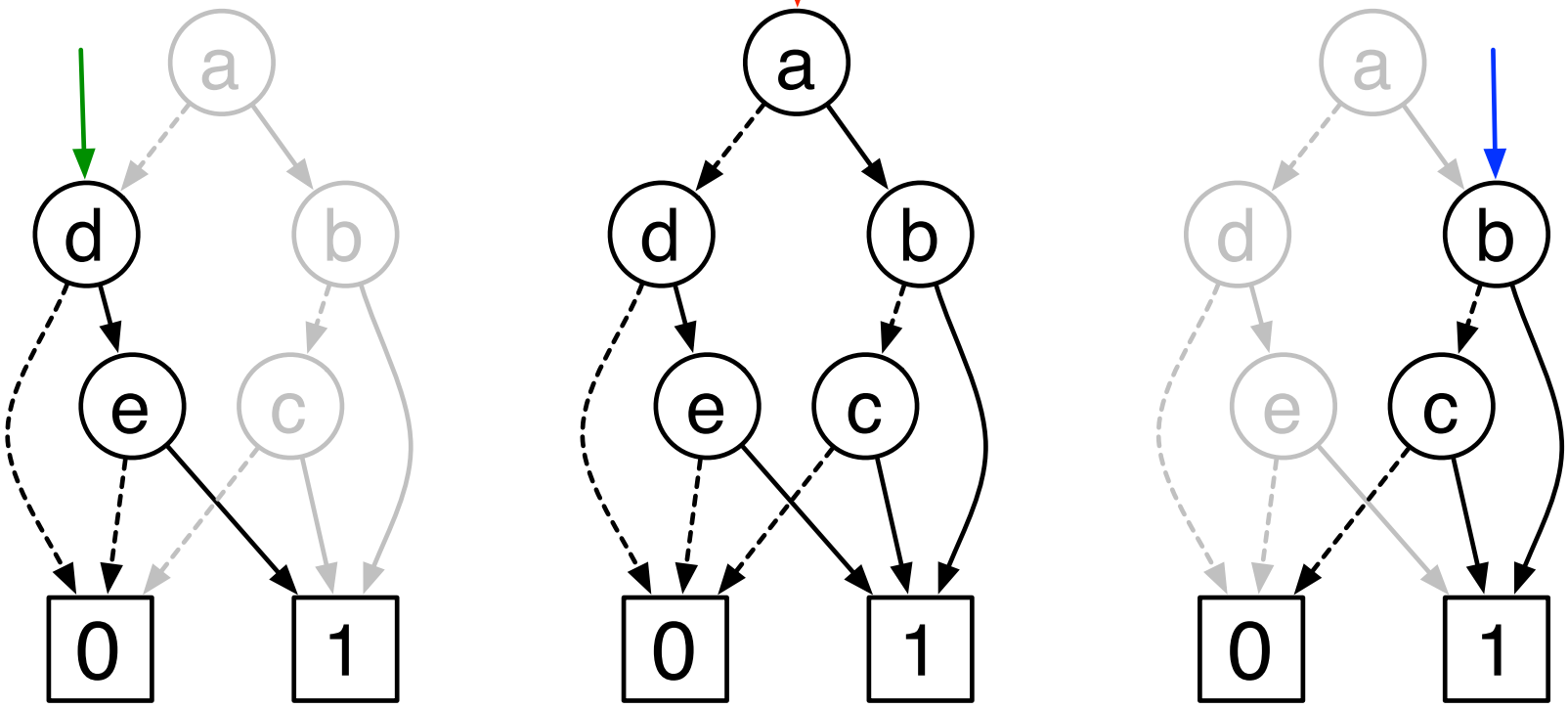
剰余算

F = a(b + c) + de

除算

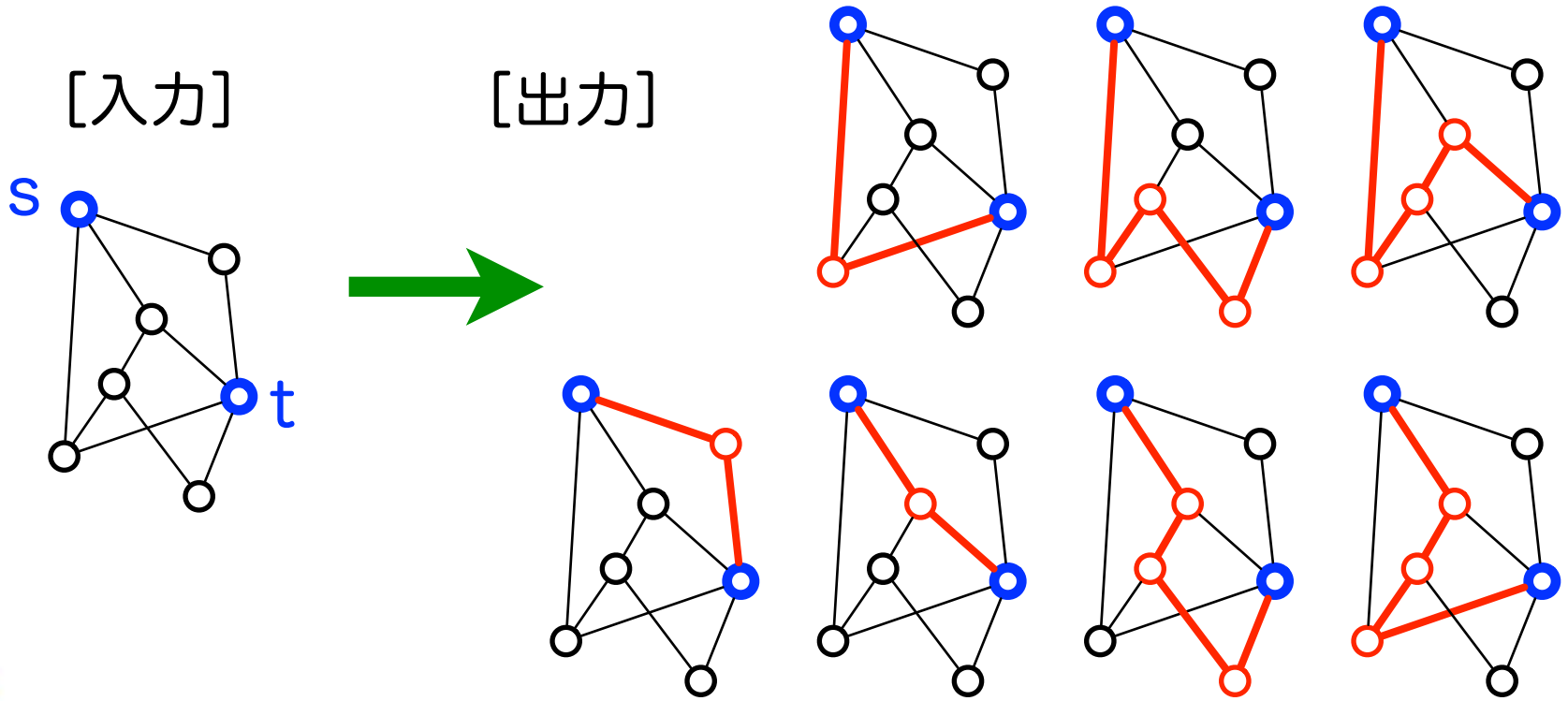
H = F % a = de

G = F / a = b + c

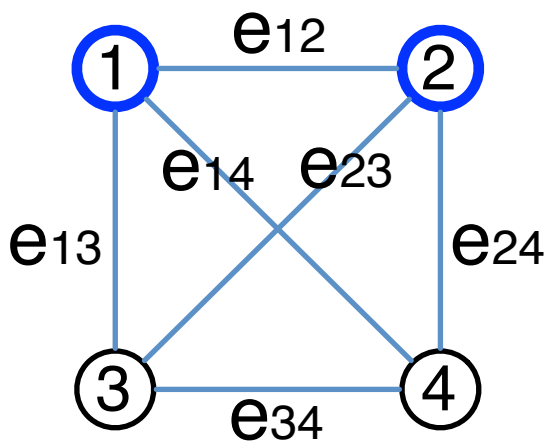


パスの列挙

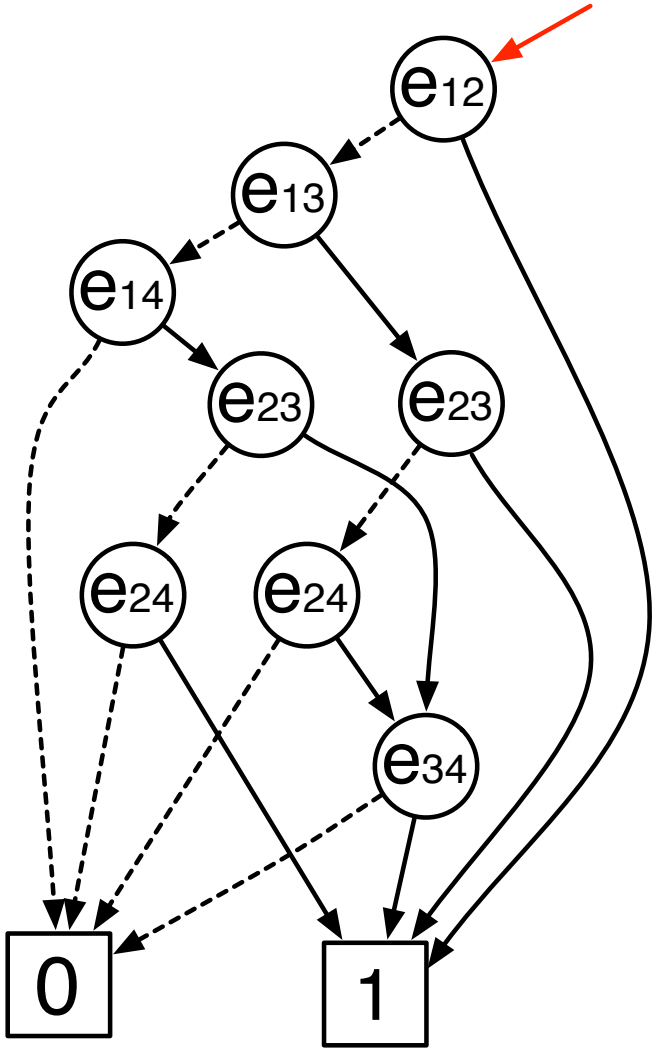
- 入力：グラフ G=(V, E)，2頂点 s, t
- 出力：s から t までのパスを列挙（保持したZDD）



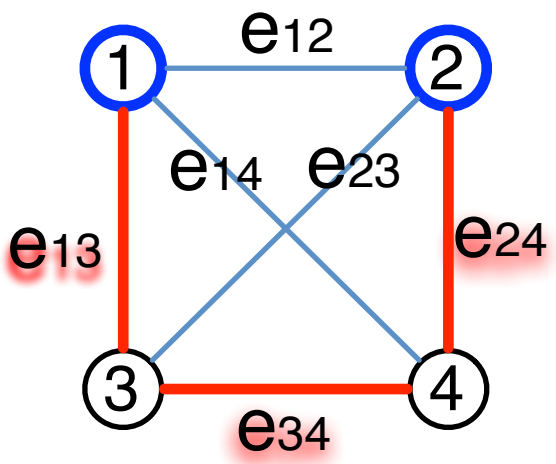
パスは辺の組合せである



$$e_{12} + e_{13}e_{23} + e_{13}e_{24}e_{34} + e_{14}e_{23}e_{34} + e_{14}e_{24}$$

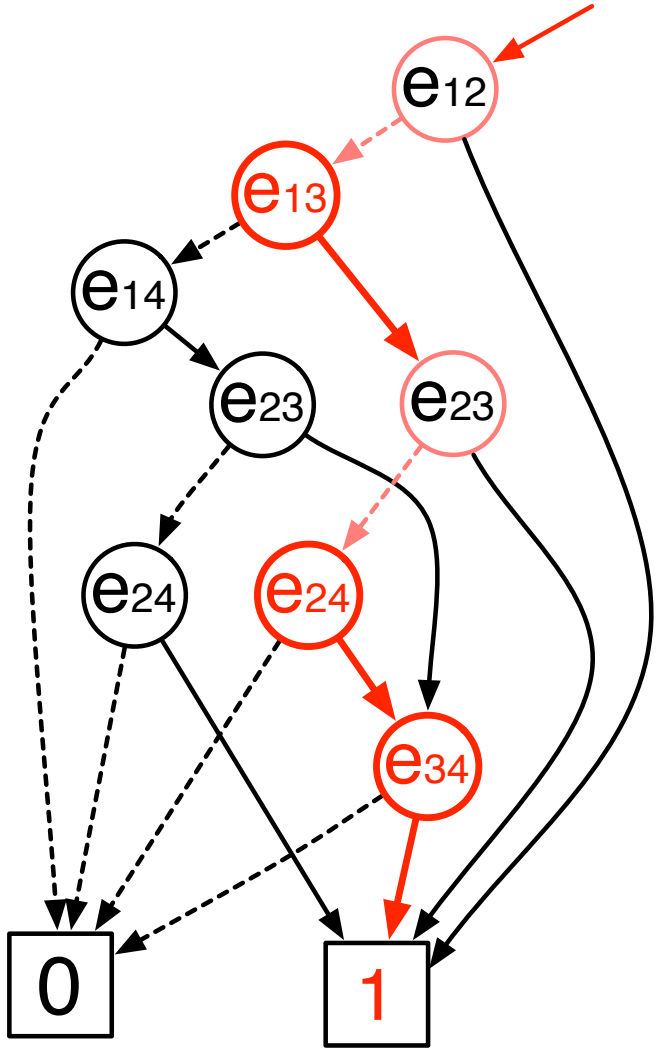


パスは辺の組合せである

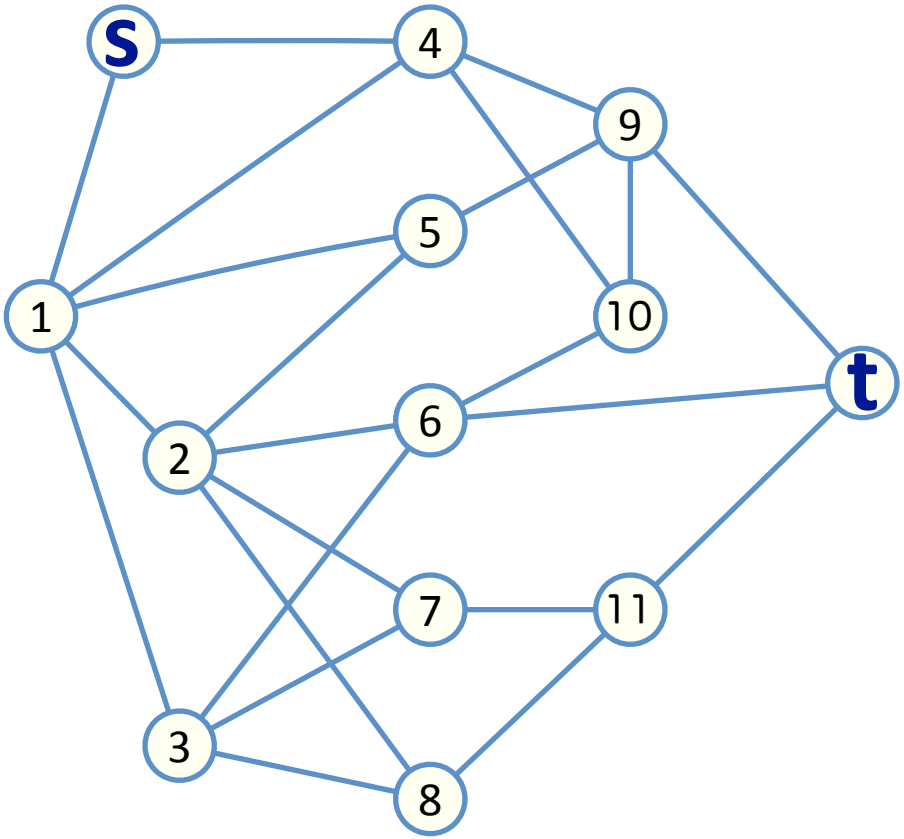


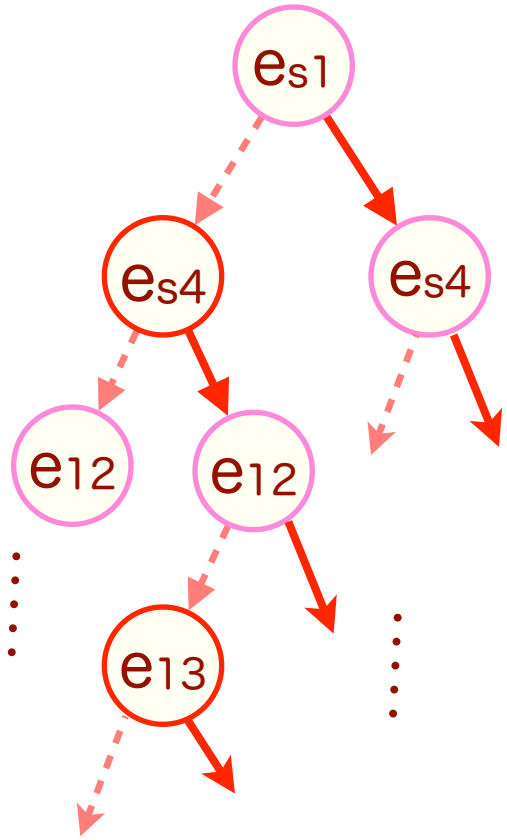
$e_{12} + e_{13}e_{23} + e_{13}e_{24}e_{34}$

$+ e_{14}e_{23}e_{34} + e_{14}e_{24}$

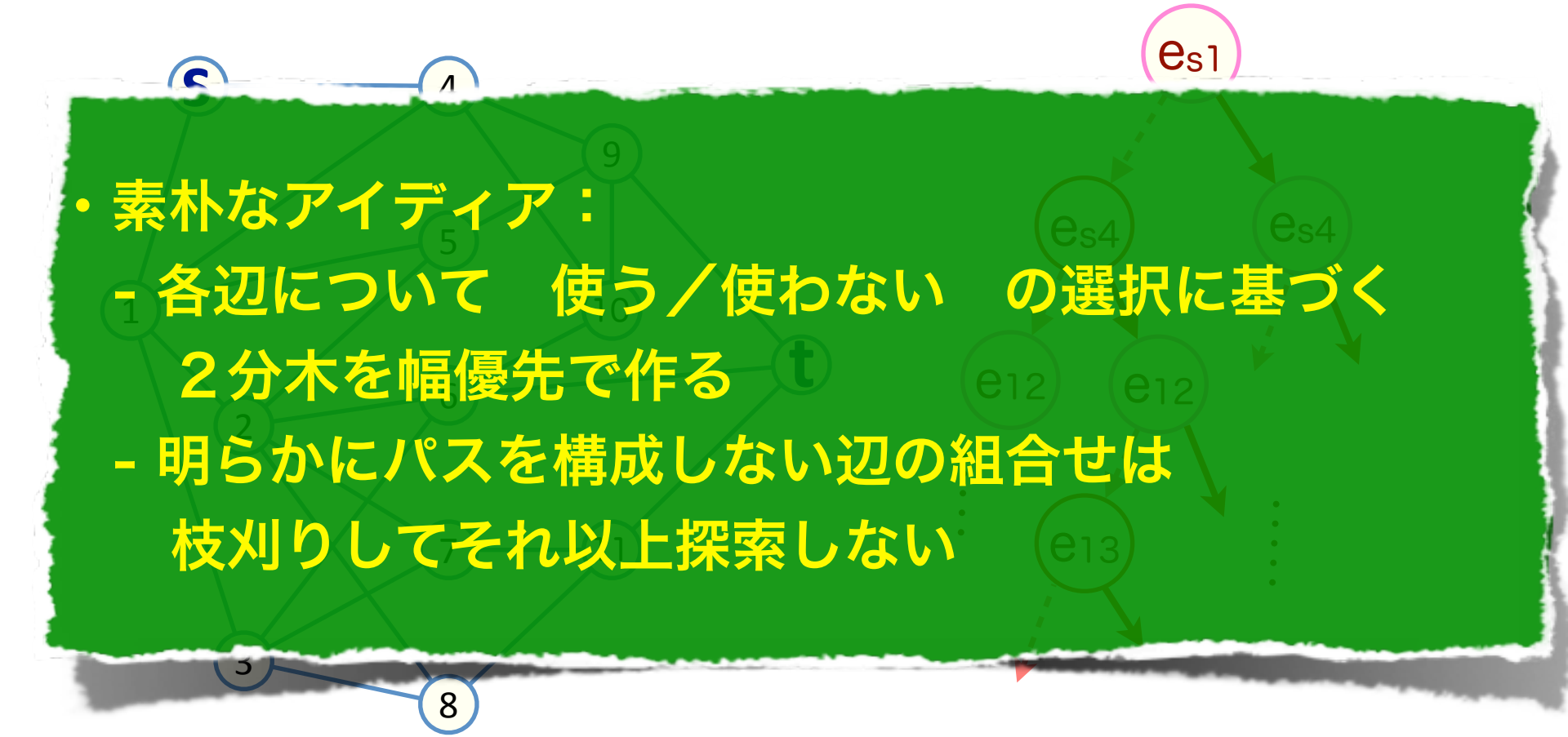


Simpath by D. Knuth



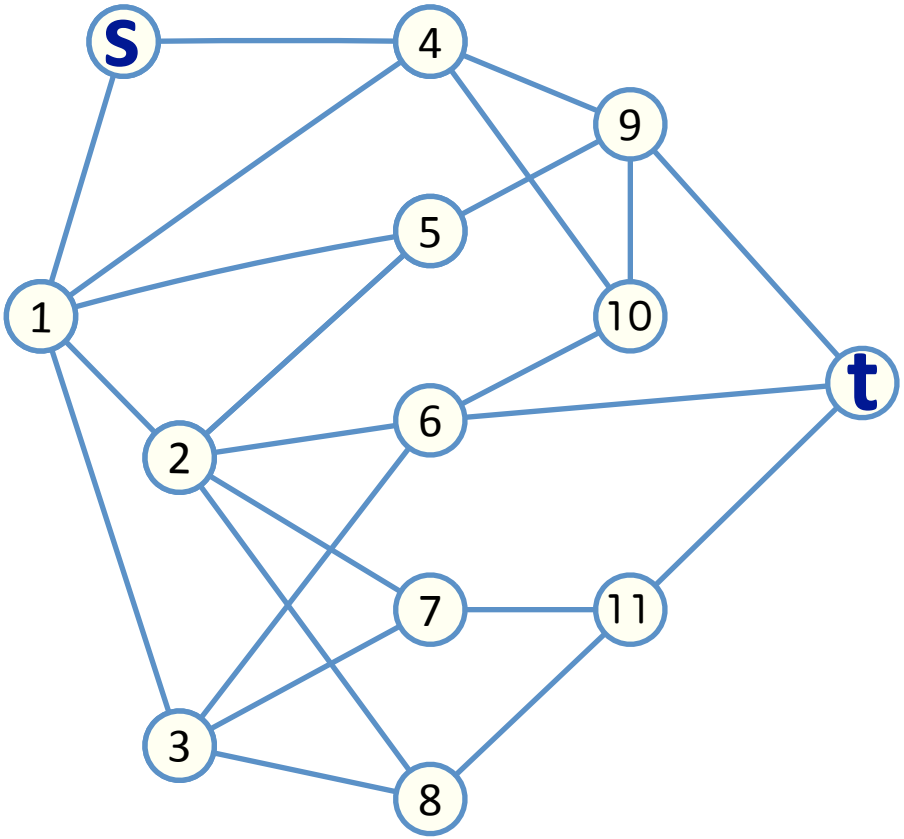


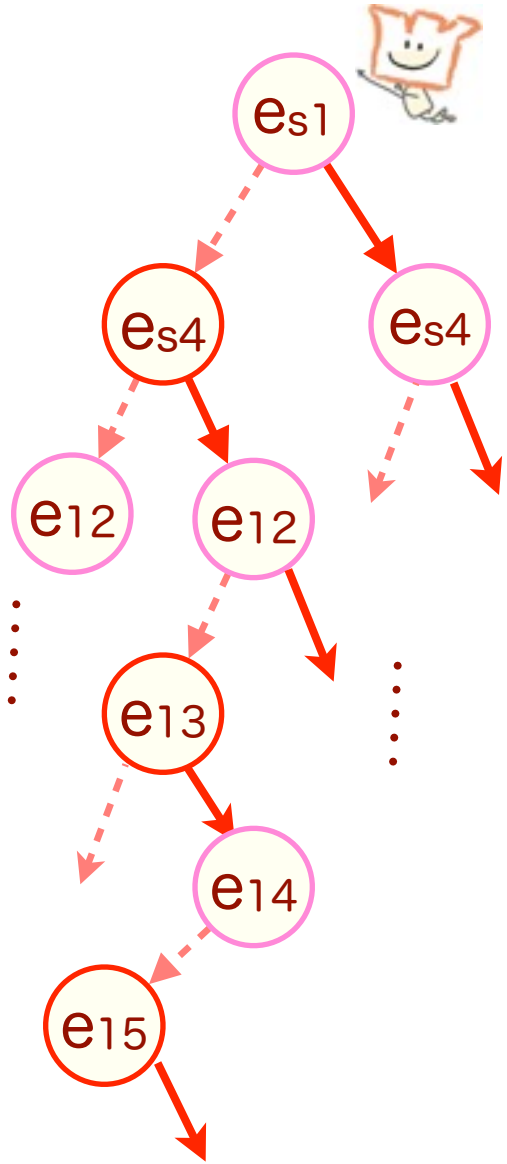
SIMPATH by D. Knuth



- 素朴なアイディア：
  - 各辺について　使う／使わない　の選択に基づく
    ２分木を幅優先で作る
  - 明らかにパスを構成しない辺の組合せは
    枝刈りしてそれ以上探索しない

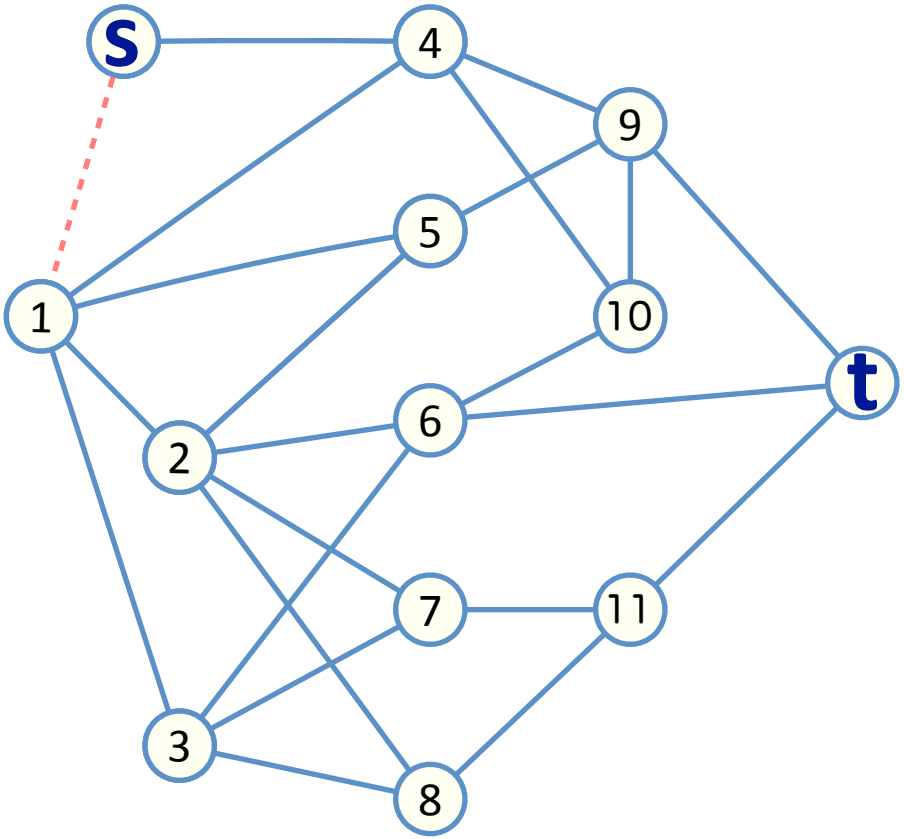
SIMPATH by D. Knuth

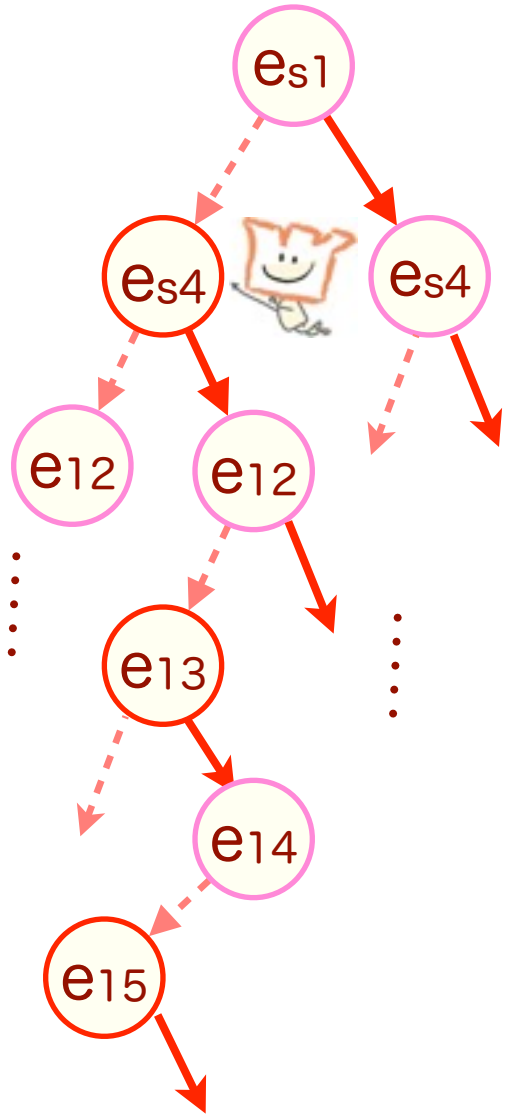




※実際には幅優先で探索

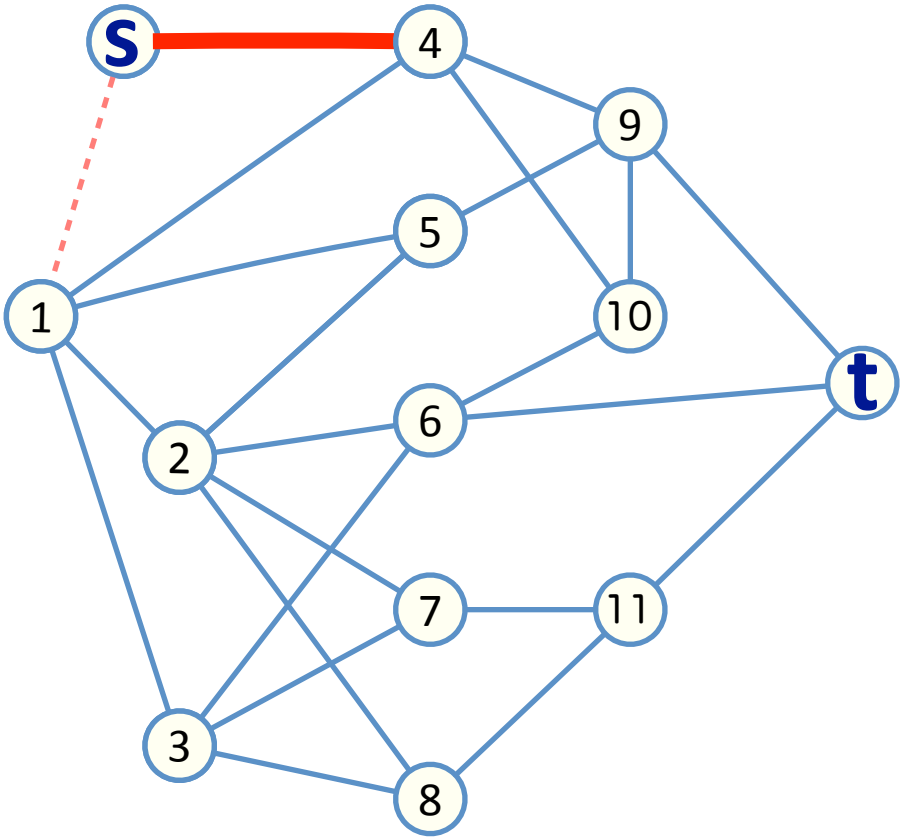
SIMPATH by D. Knuth

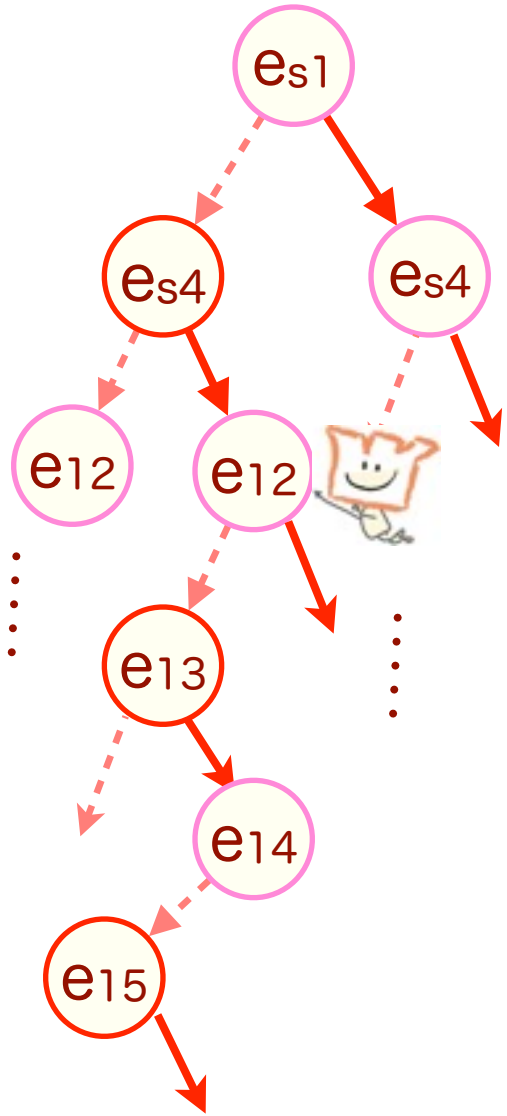




※実際には幅優先で探索

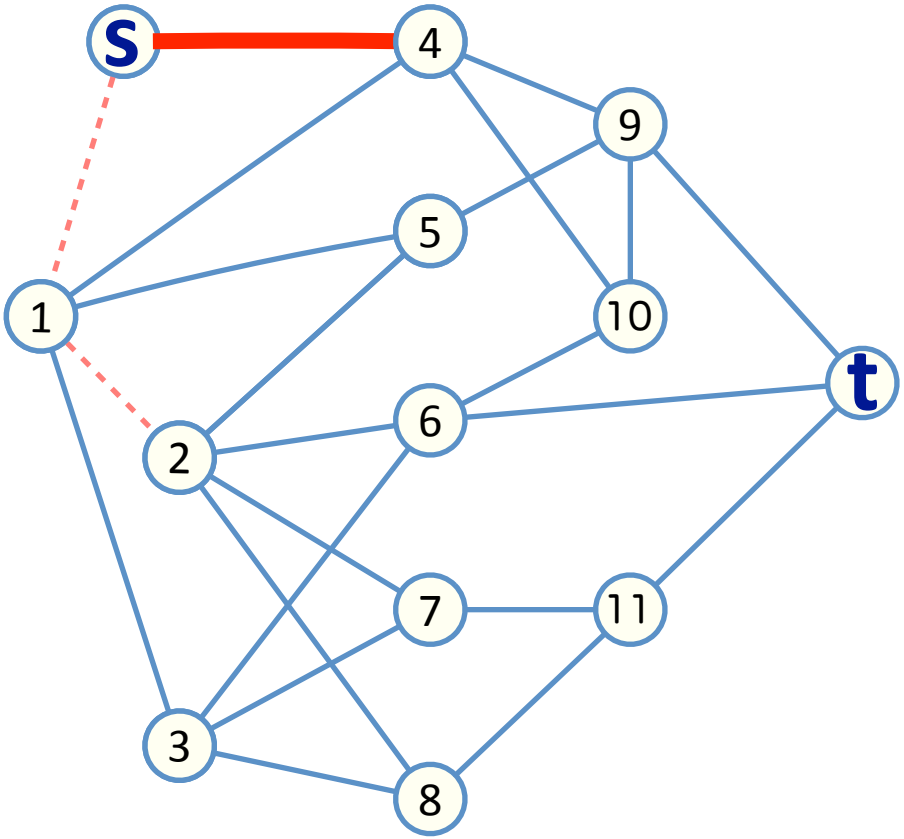
SIMPATH by D. Knuth

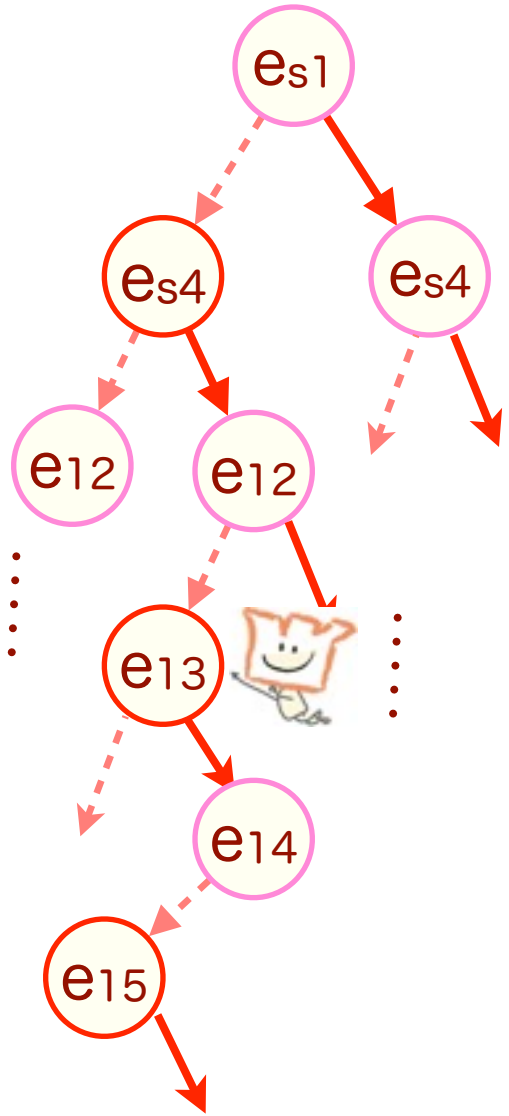




※実際には幅優先で探索

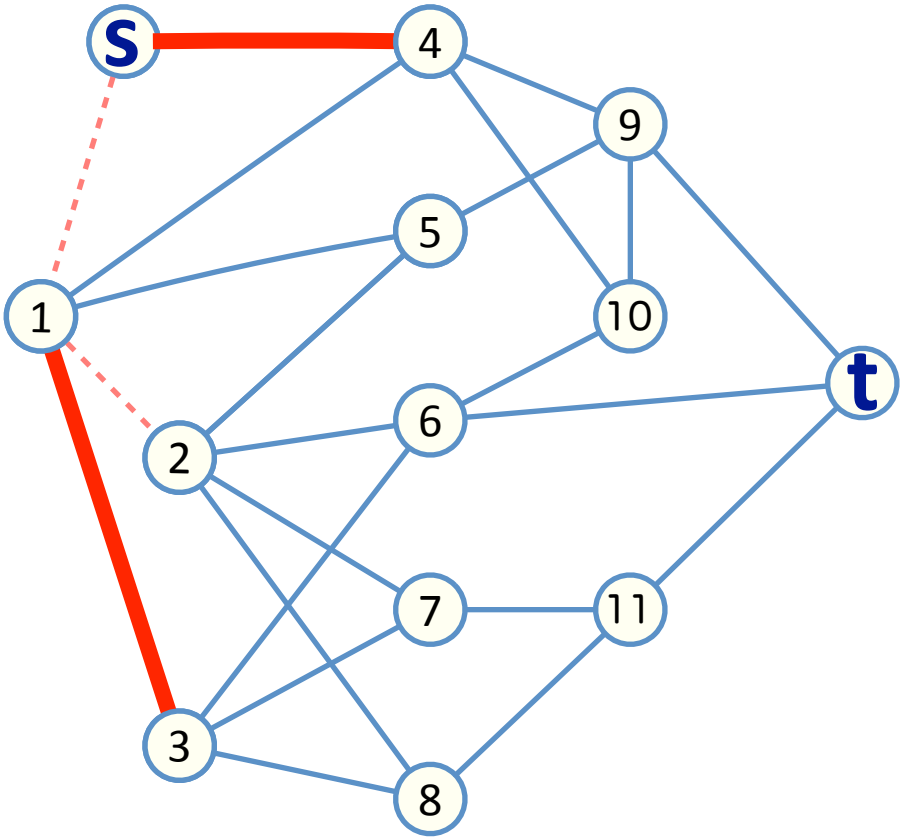
SIMPATH by D. Knuth

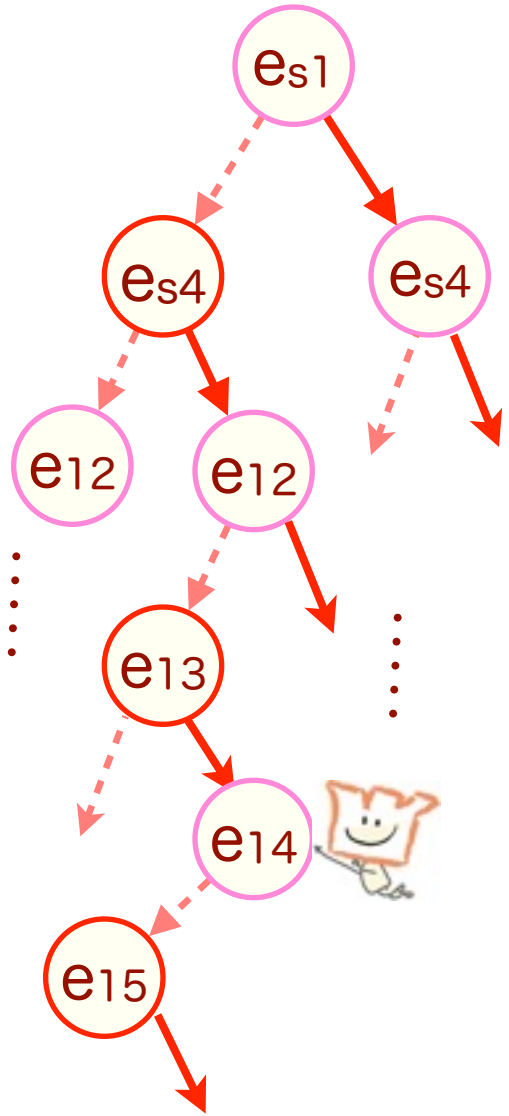




※実際には幅優先で探索

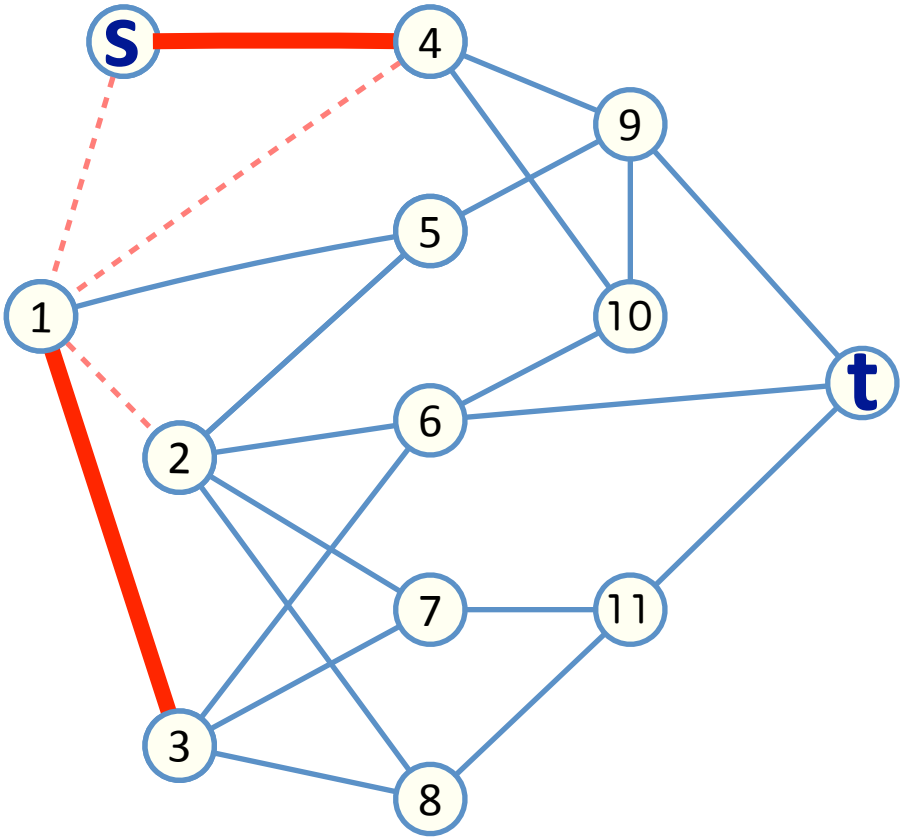
SIMPATH by D. Knuth

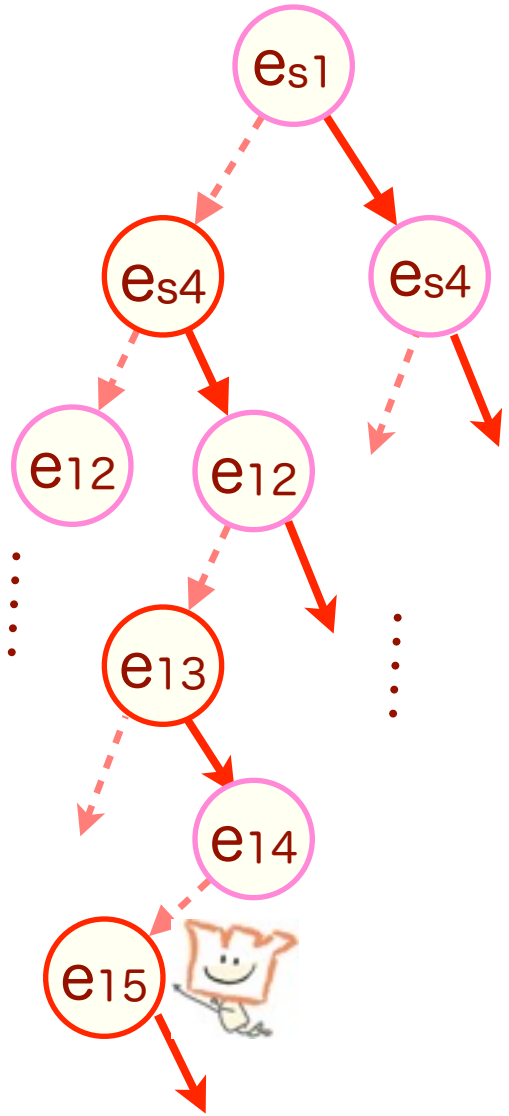




※実際には幅優先で探索

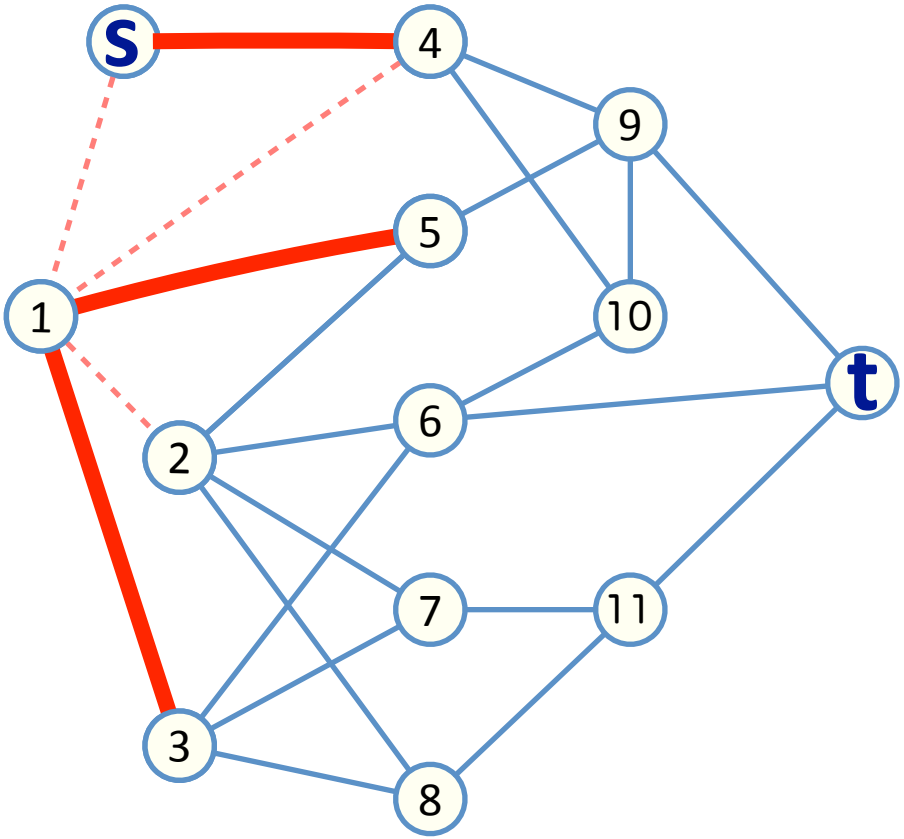
SIMPATH by D. Knuth

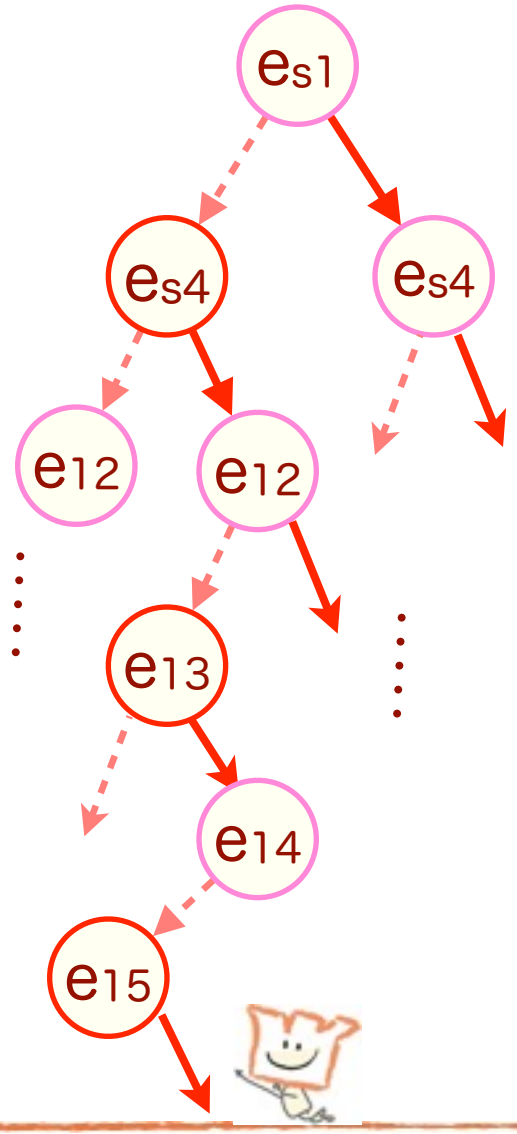




※実際には幅優先で探索

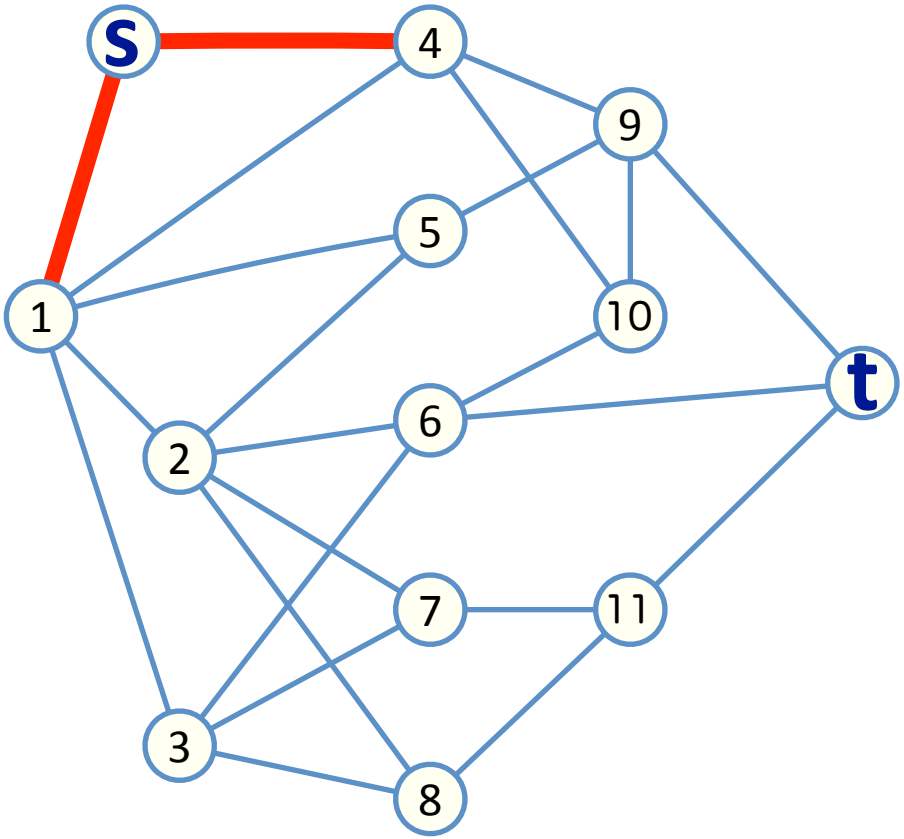
SIMPATH by D. Knuth

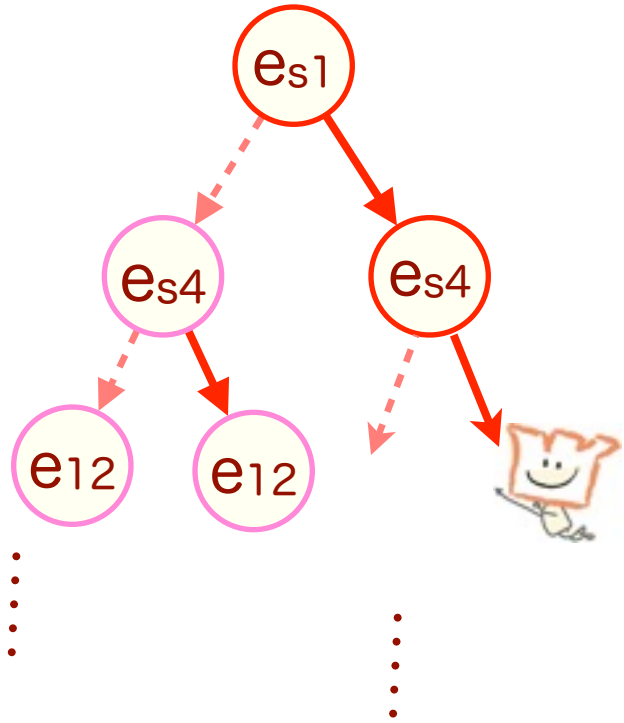




※実際には幅優先で探索

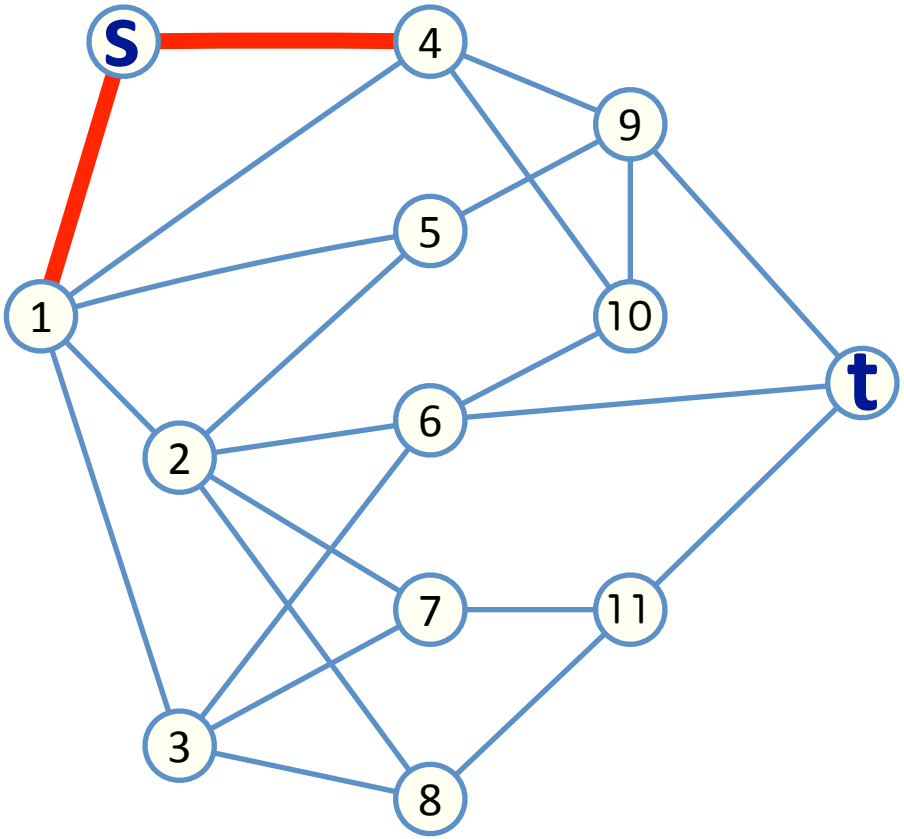
SIMPATH by D. Knuth

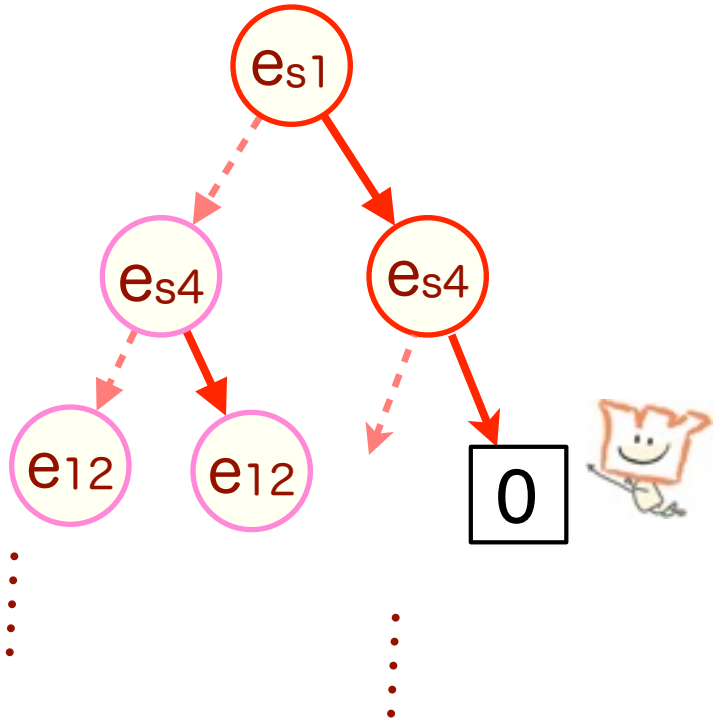




※実際には幅優先で探索

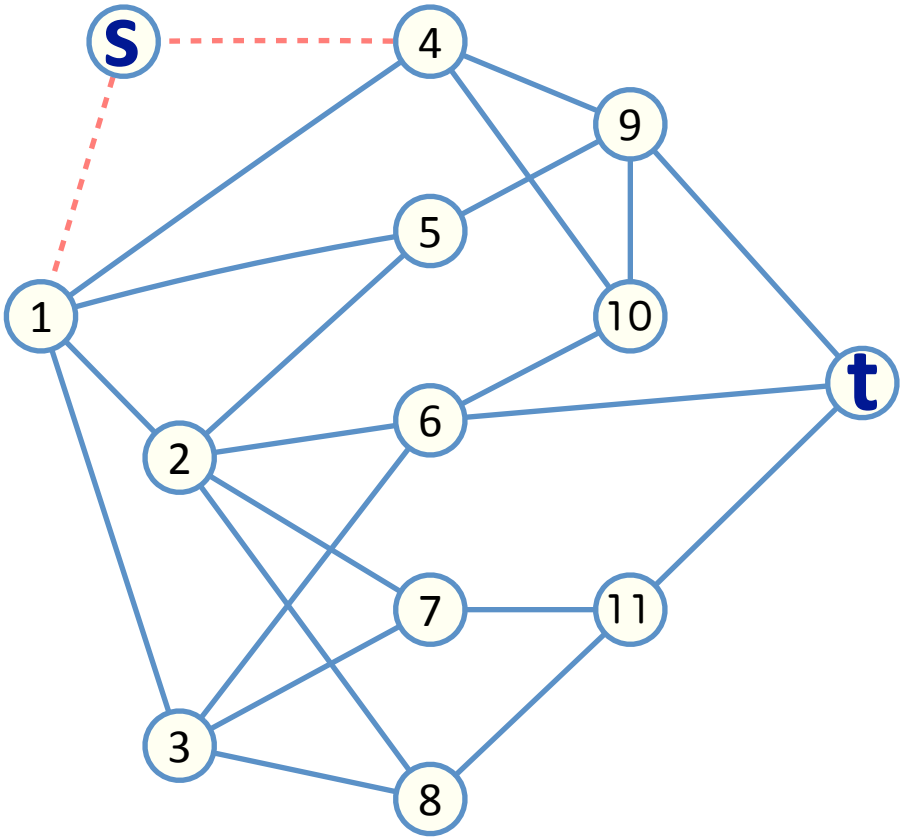
SIMPATH by D. Knuth

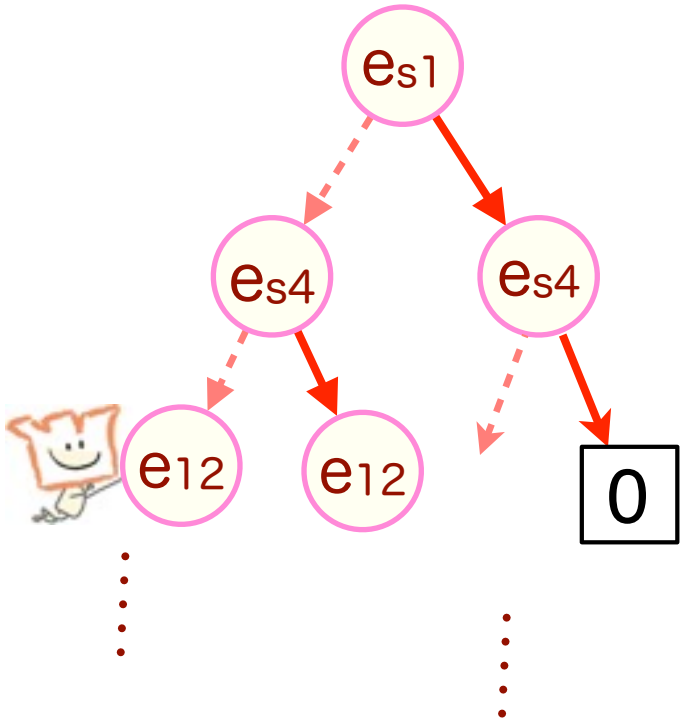




※実際には幅優先で探索

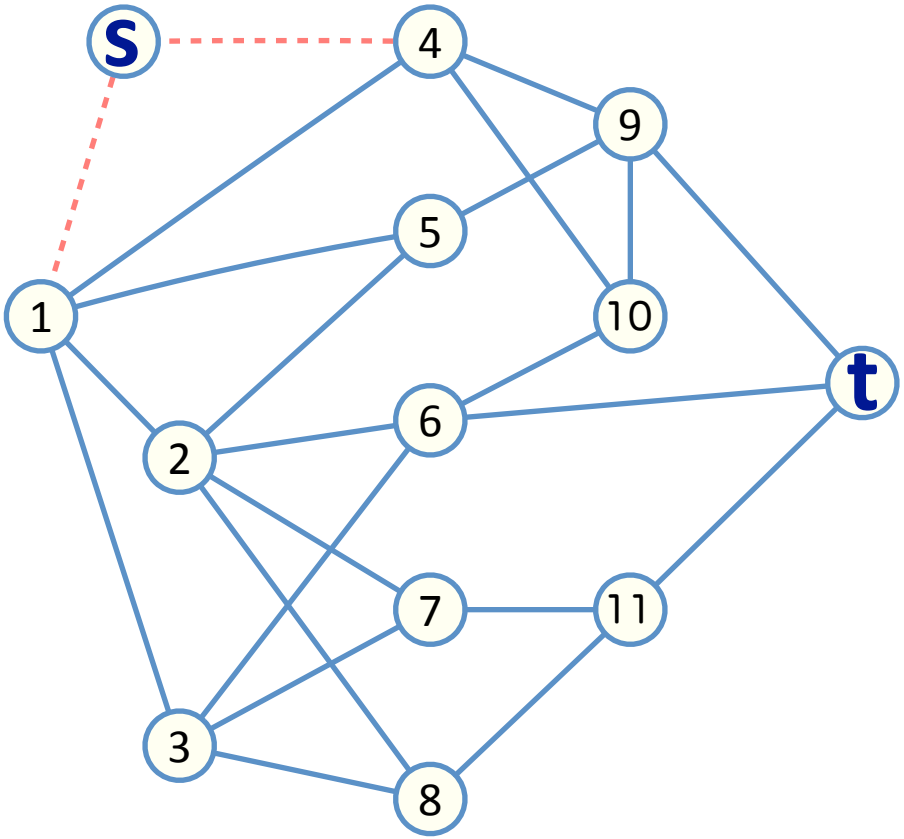
SIMPATH by D. Knuth

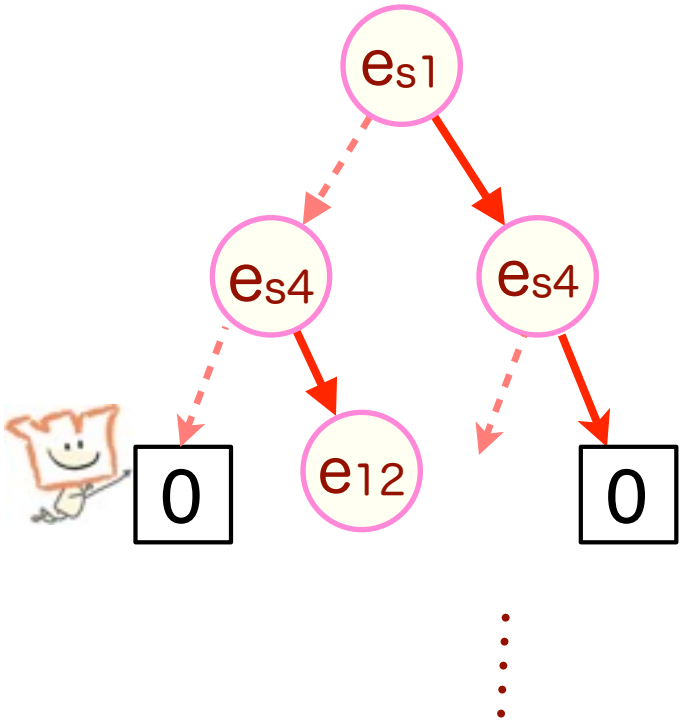




※実際には幅優先で探索

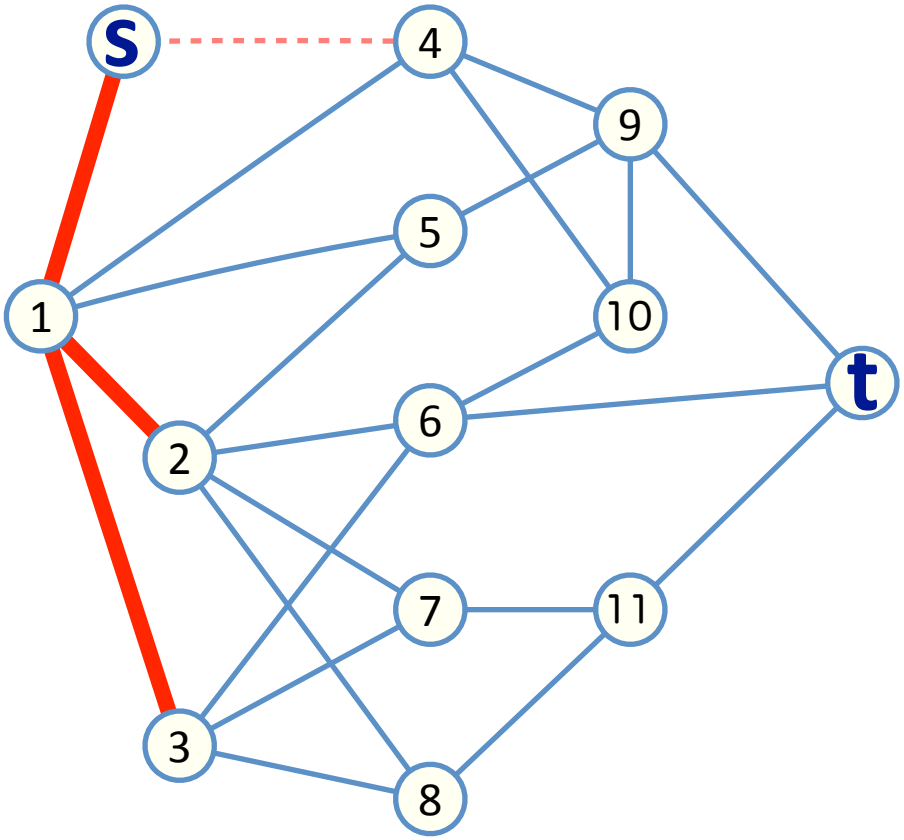
SIMPATH by D. Knuth

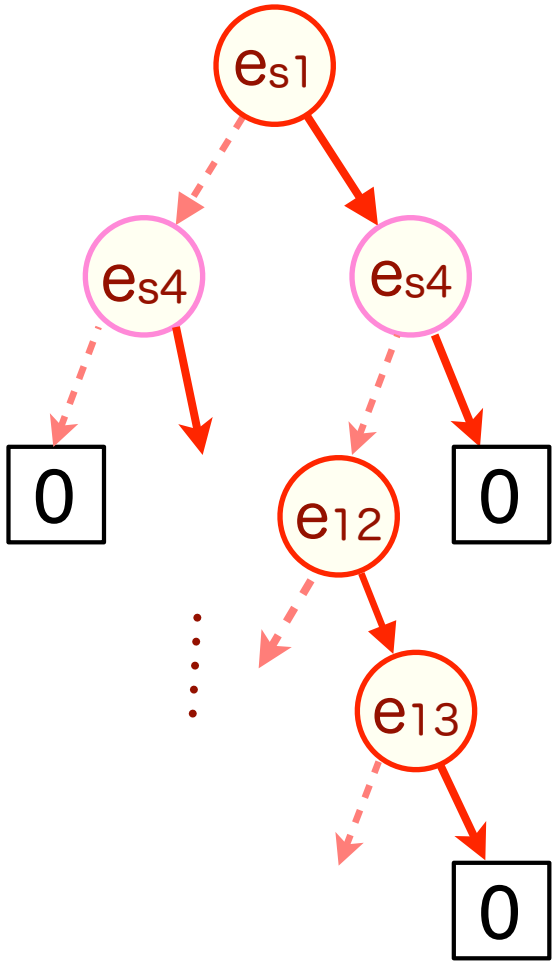




※実際には幅優先で探索

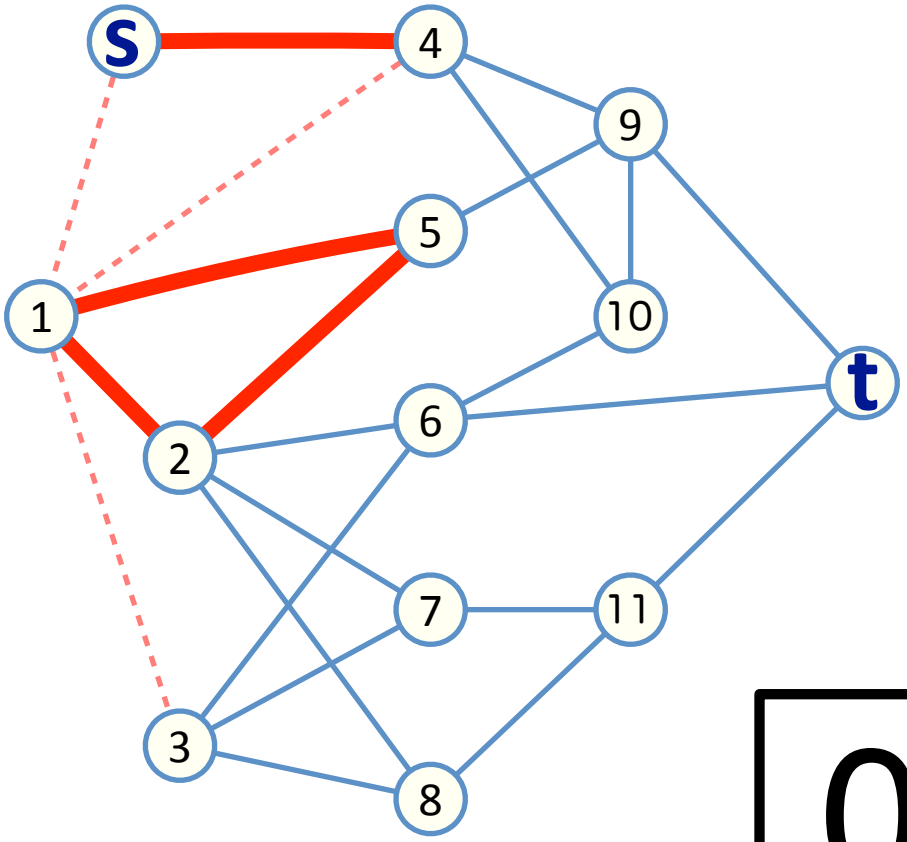
SIMPATH by D. Knuth

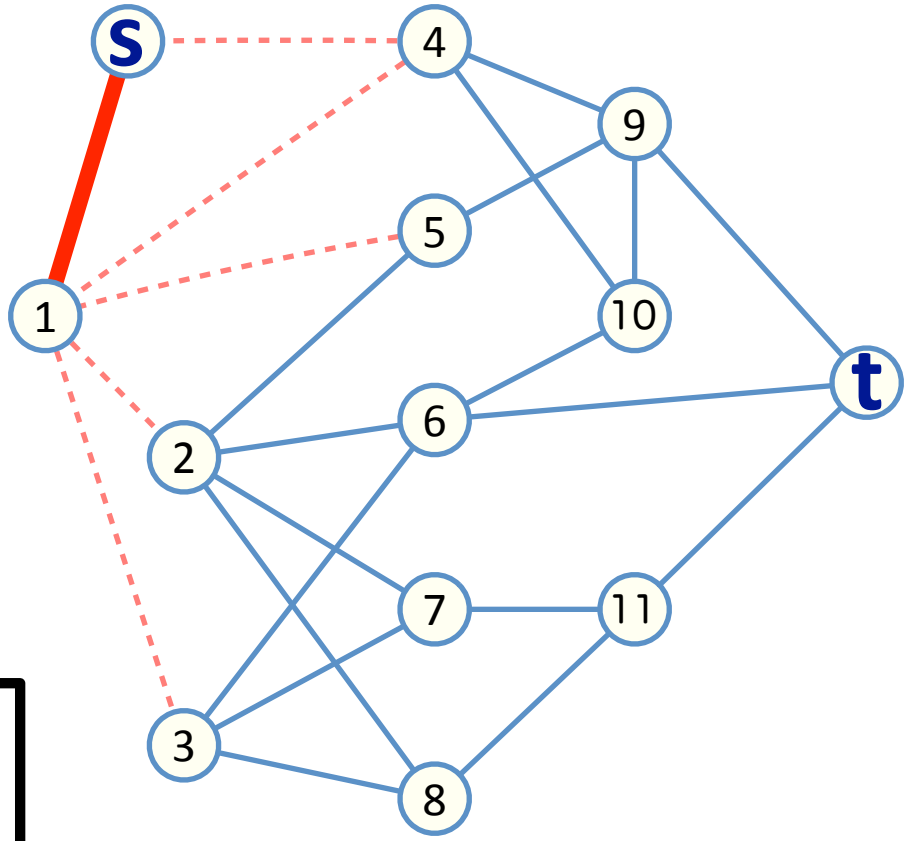




※実際には幅優先で探索

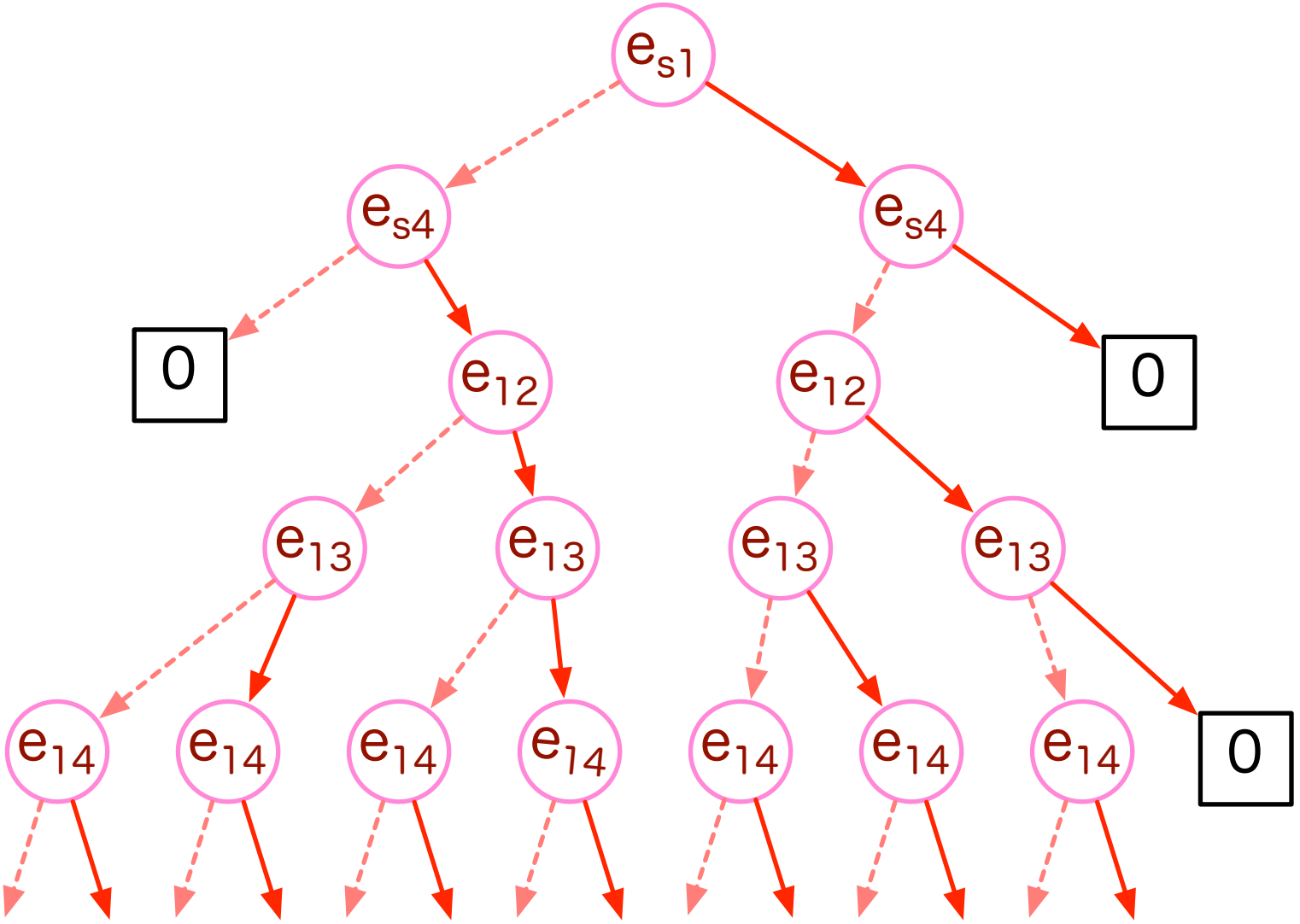
Simpath by D. Knuth



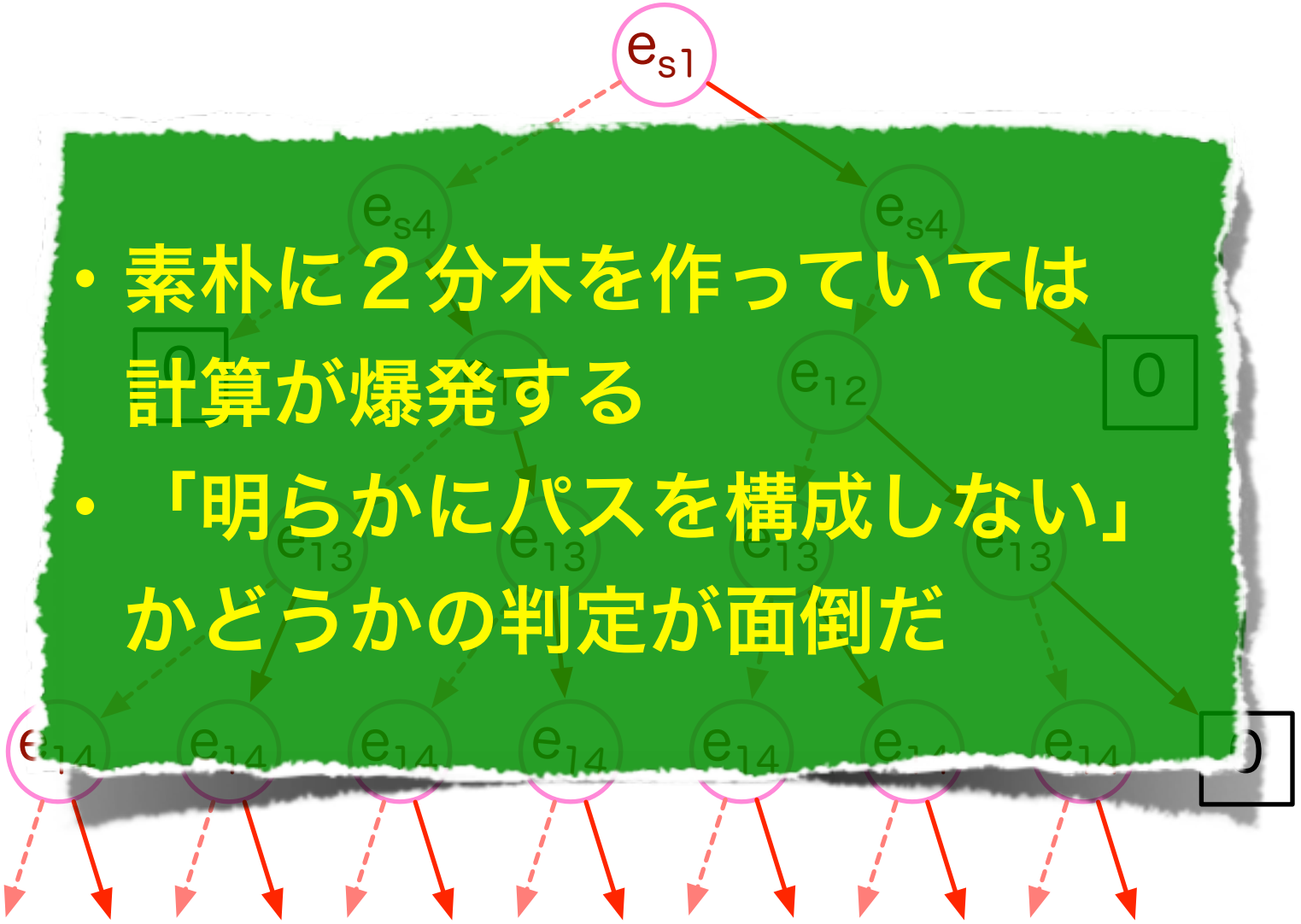


0

SIMPATH by D. Knuth

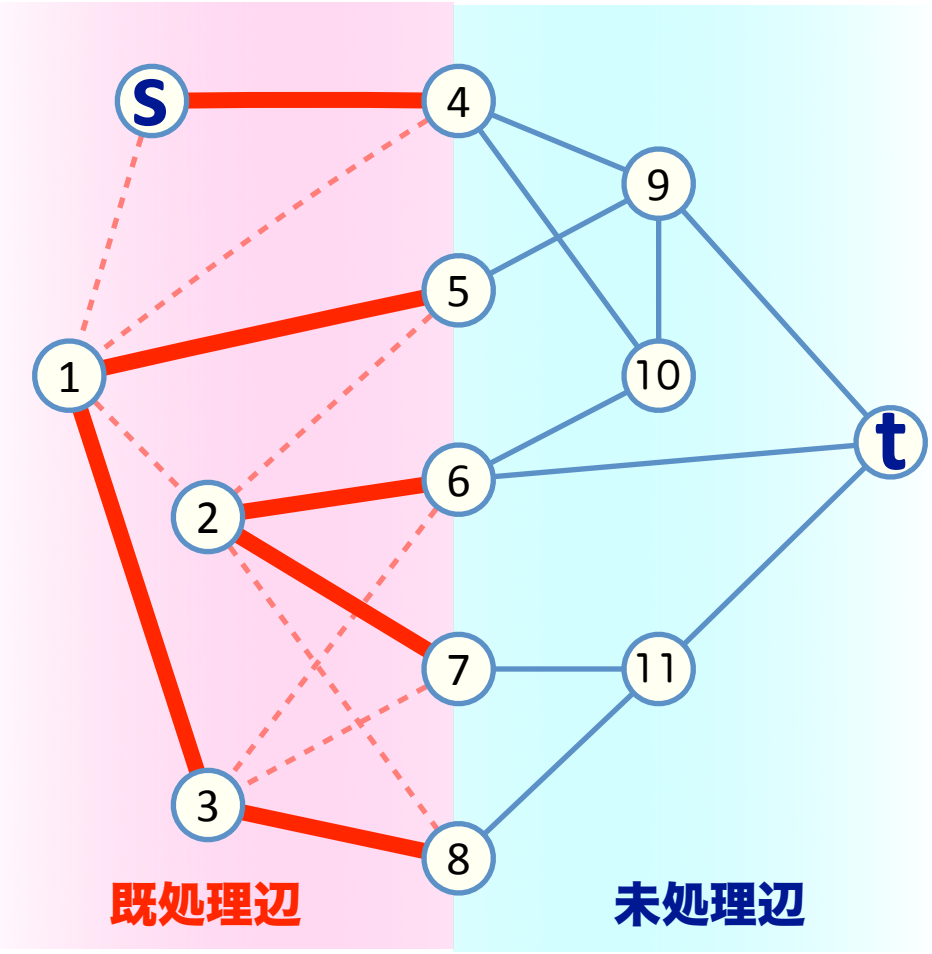


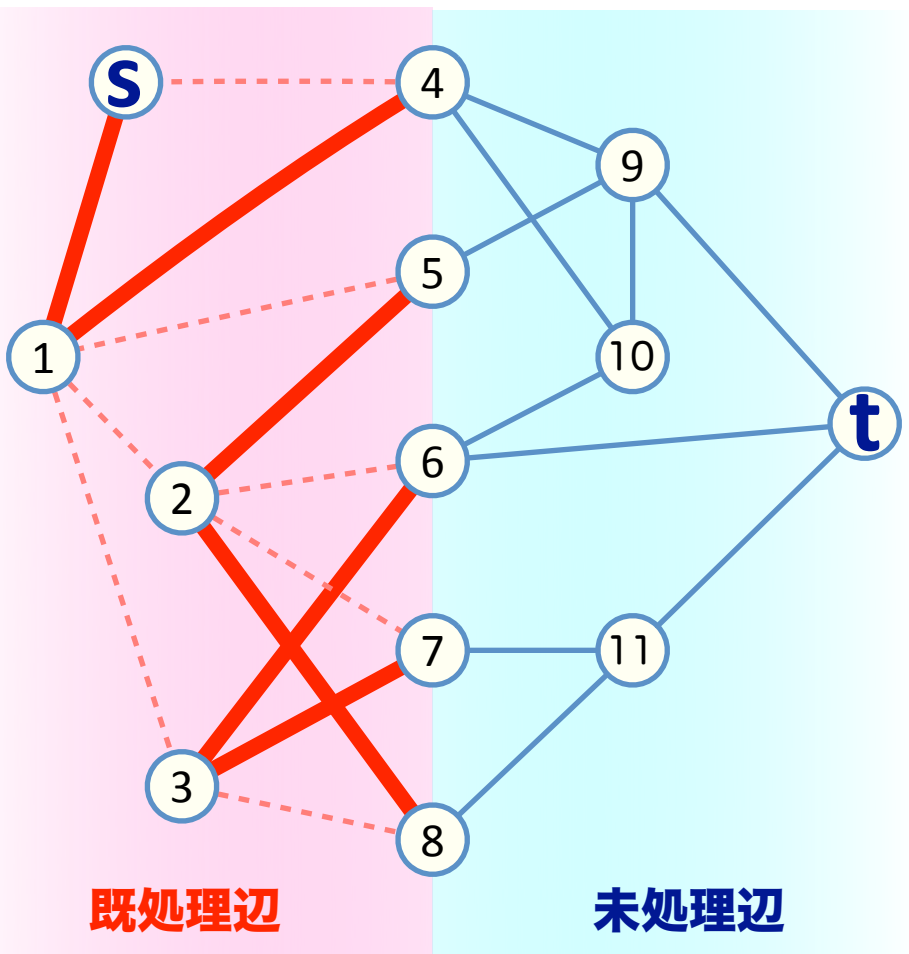
SIMPATH by D. Knuth



- 素朴に２分木を作っていては計算が爆発する
- 「明らかにパスを構成しない」かどうかの判定が面倒だ

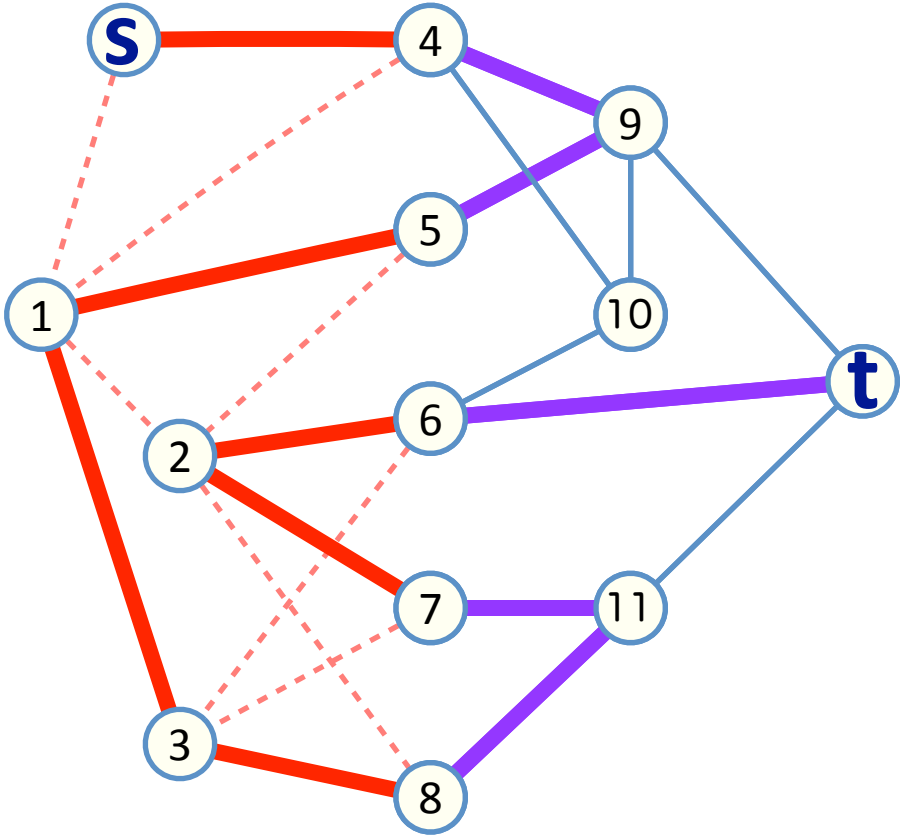
☆ mate ☆ ～ パスの端点対の組合せ ～

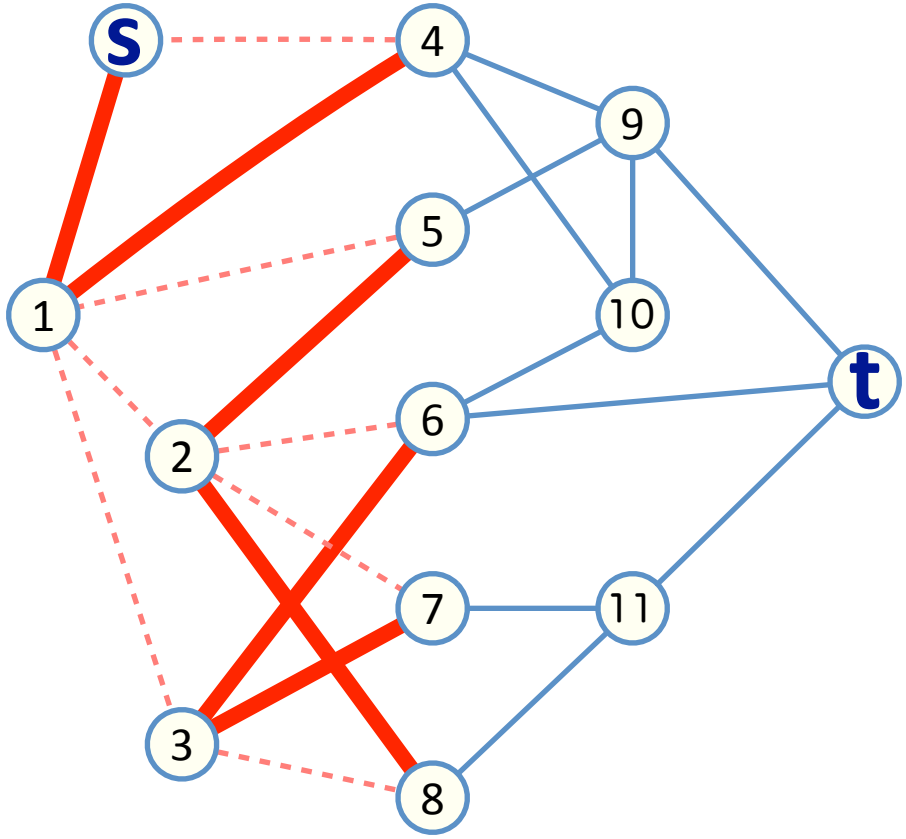




どちらも端点対 (S—④), (⑤—⑧), (⑥—⑦) のパスをもつ

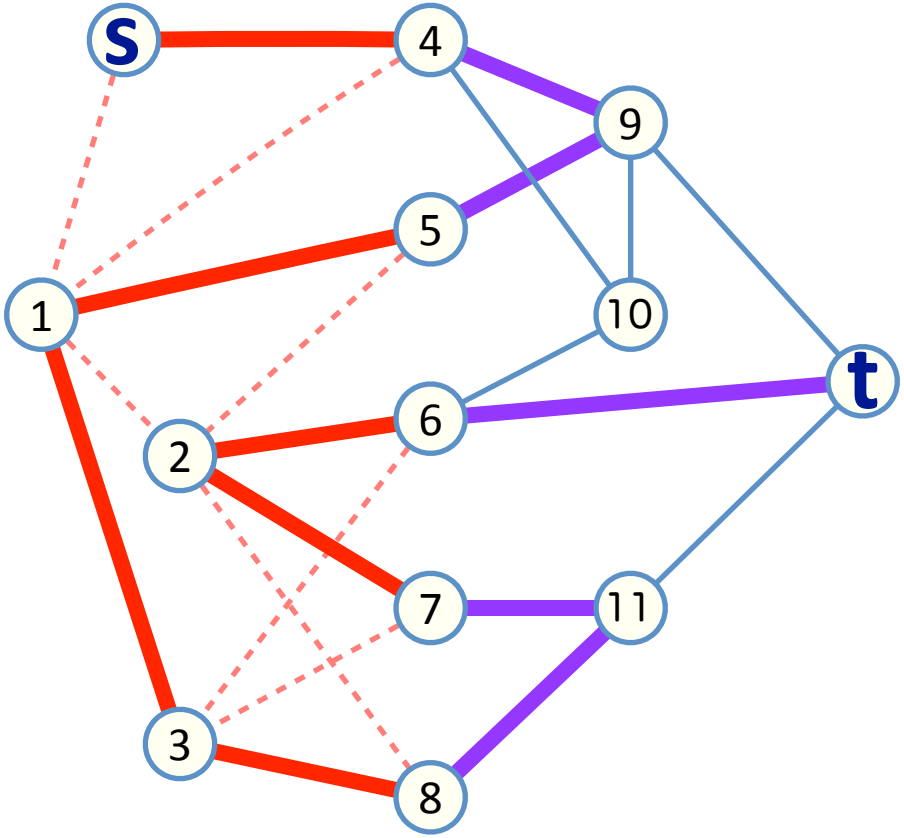
☆ mate ☆ ～ パスの端点対の組合せ ～

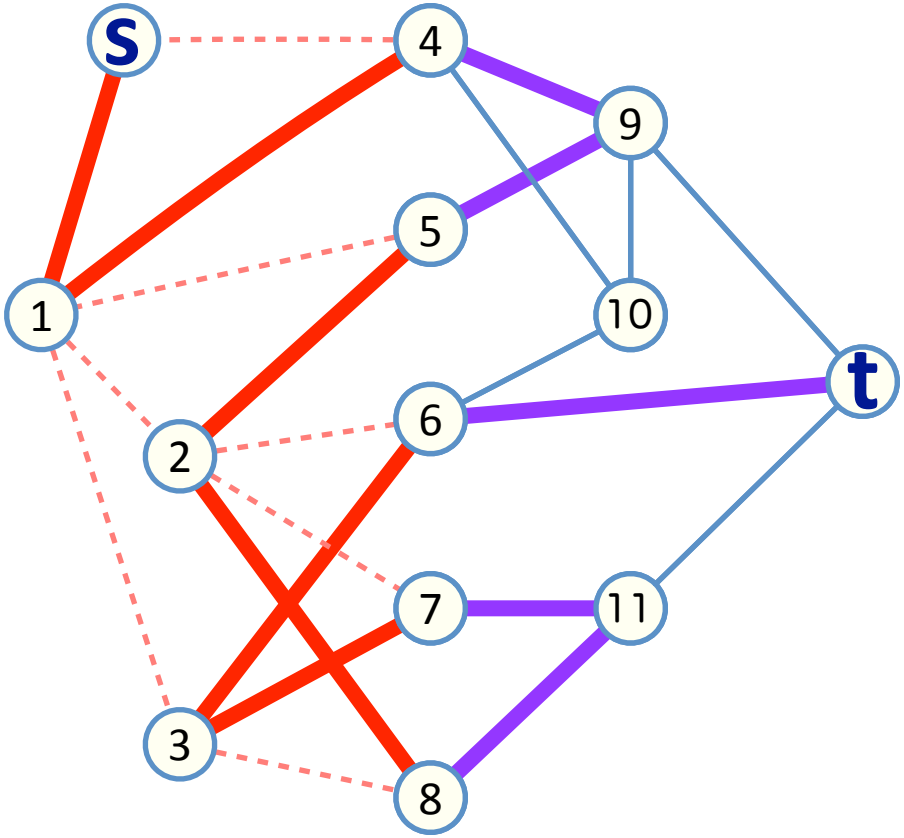




どちらも端点対 (s—④), (⑤—⑧), (⑥—⑦) のパスをもつ

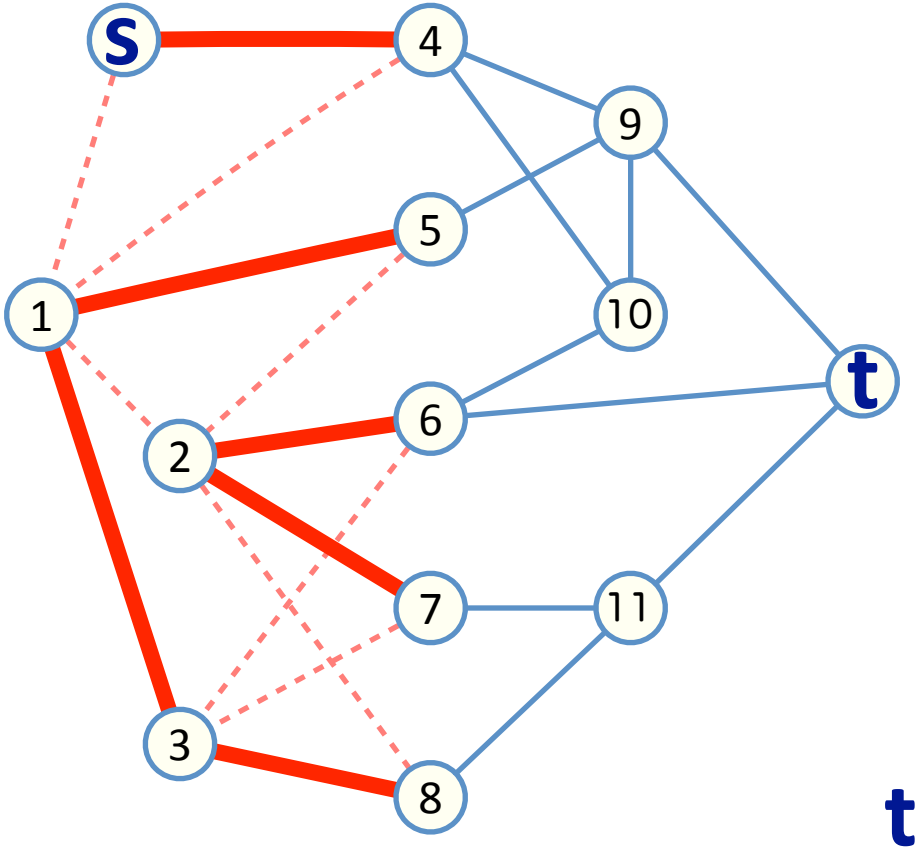
☆ mate ☆ ～ パスの端点対の組合せ ～

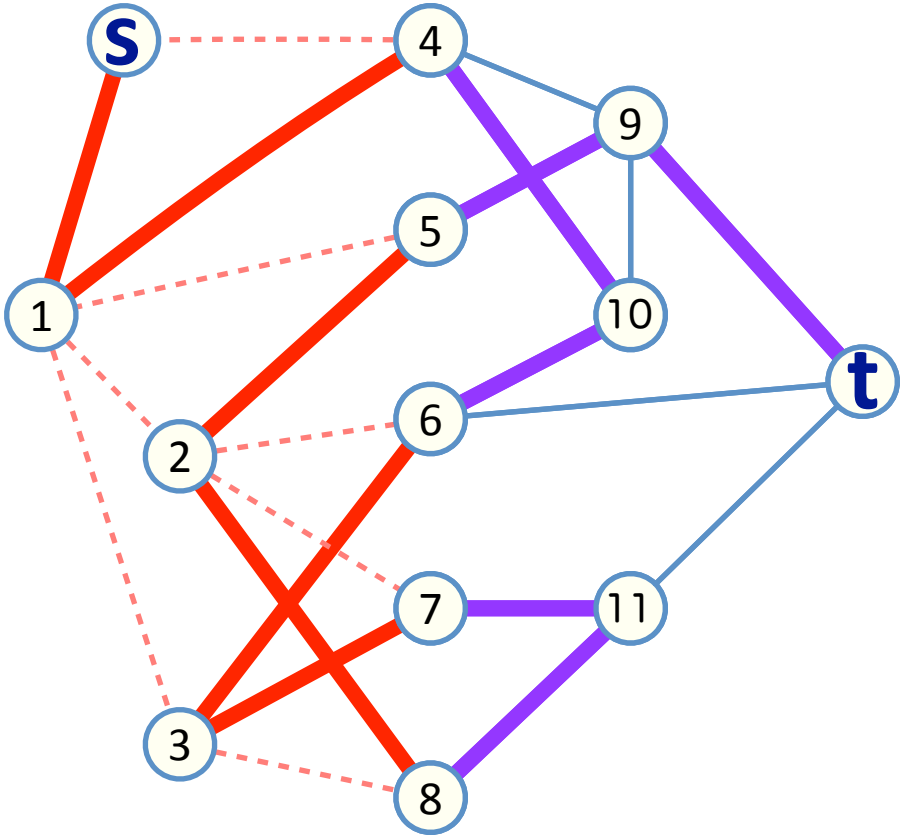


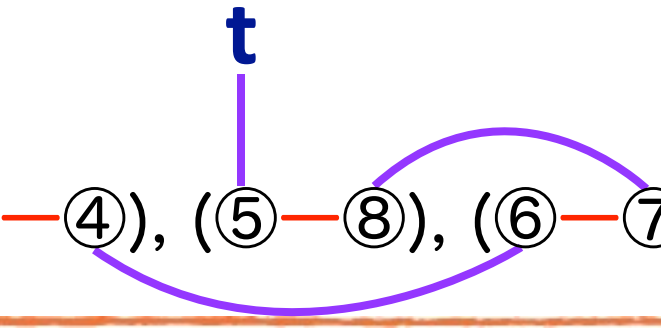


どちらも端点対 (s—④), (⑤—⑧), (⑥—⑦) のパスをもつ

☆ mate ☆ ～ パスの端点対の組合せ ～

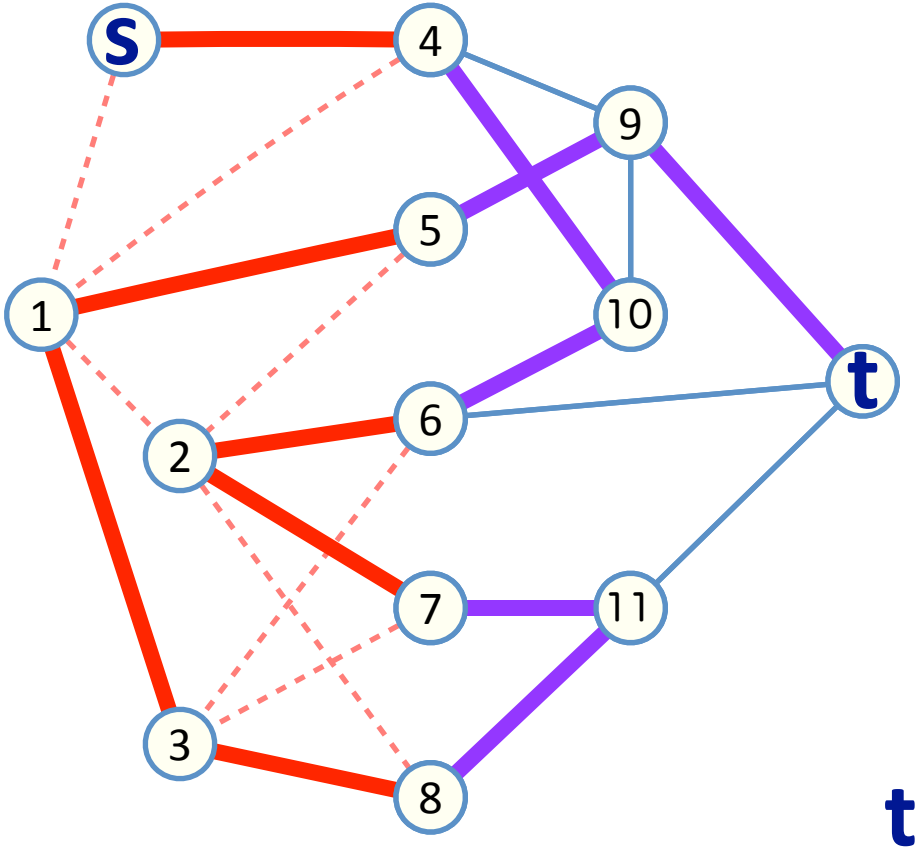


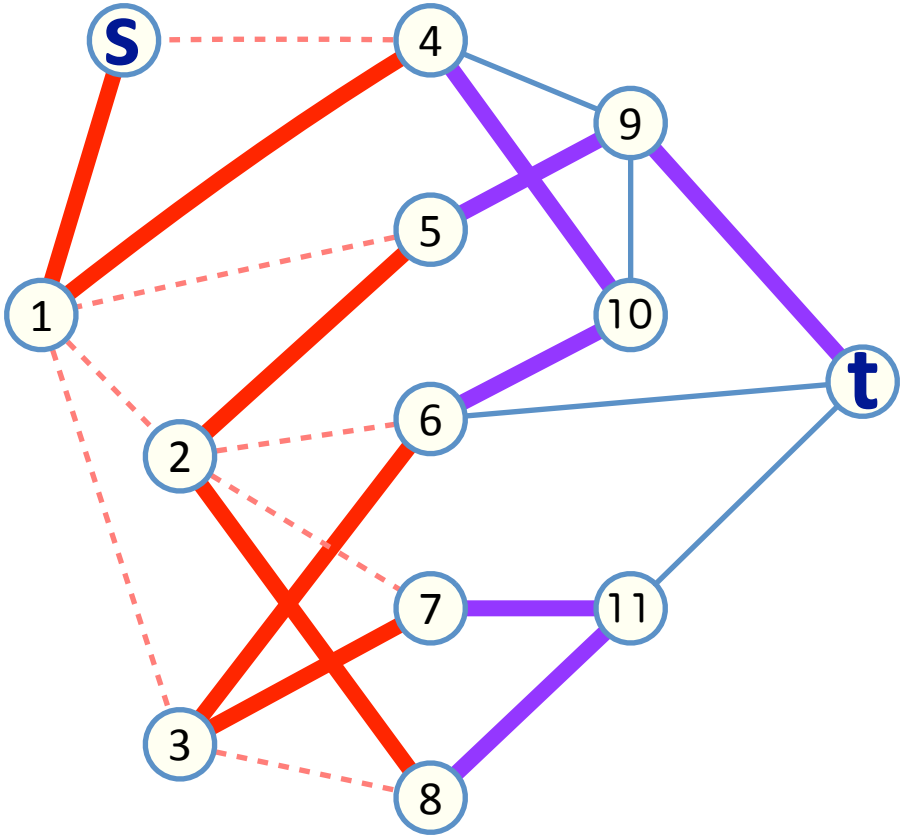


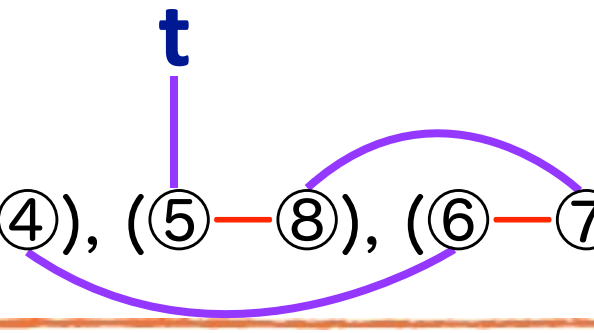


どちらも端点対 (s—④), (⑤—⑧), (⑥—⑦) のパスをもつ

☆ mate ☆ ～ パスの端点対の組合せ ～

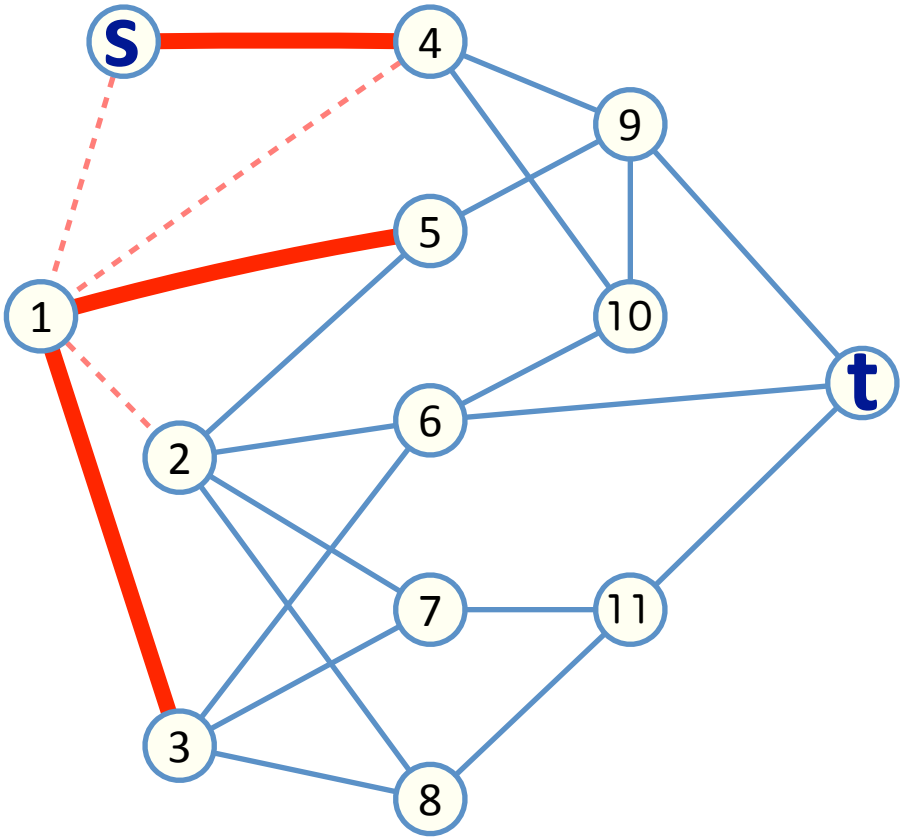


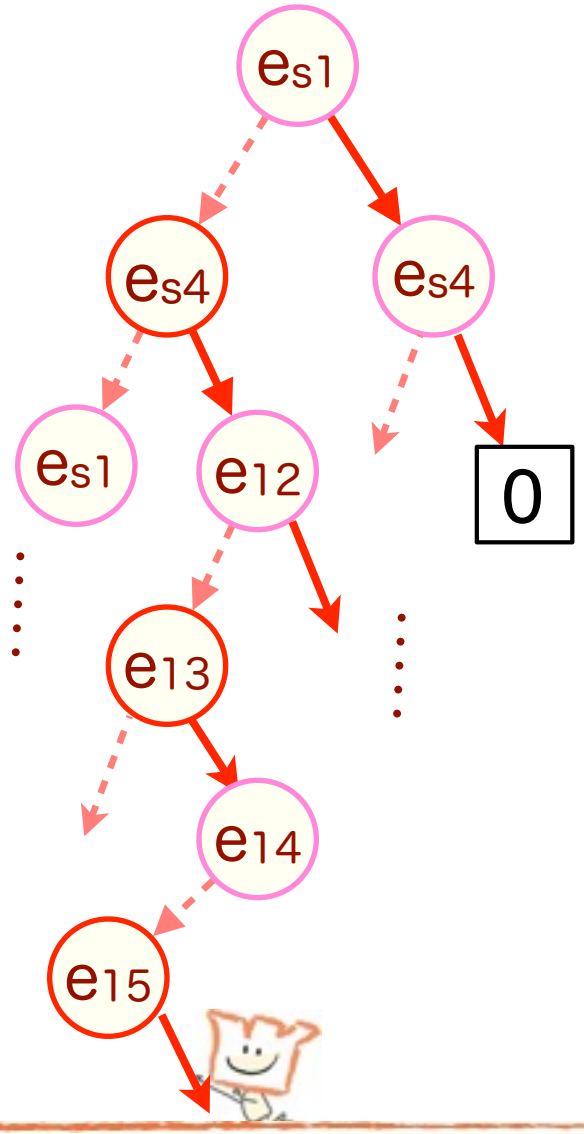




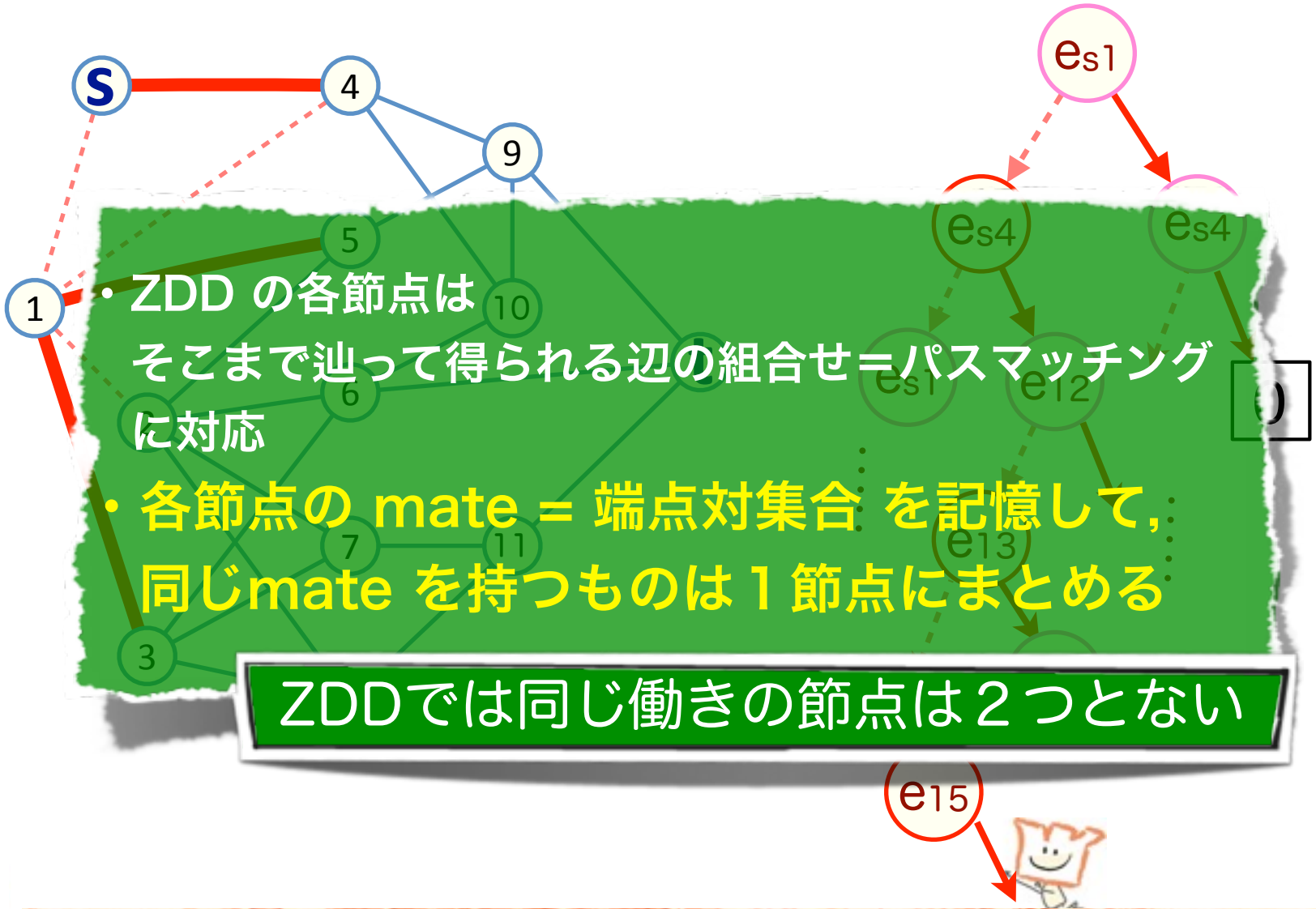
どちらも端点対 (s—④), (⑤—⑧), (⑥—⑦) のパスをもつ

SIMPATH by D. Knuth

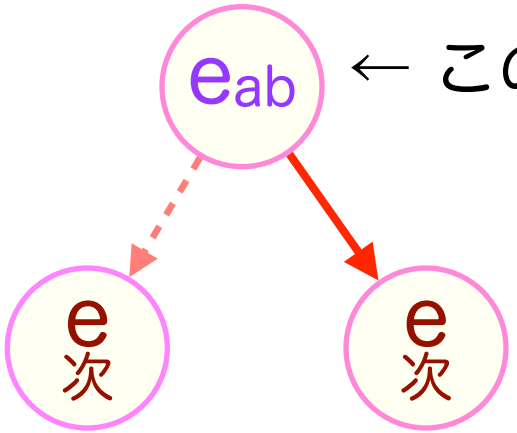




SIMPATH by D. Knuth



mate の更新



← この mate が [...,(a–c),(b–d),...] のとき

← この mate は [...,(c–d),...] になる

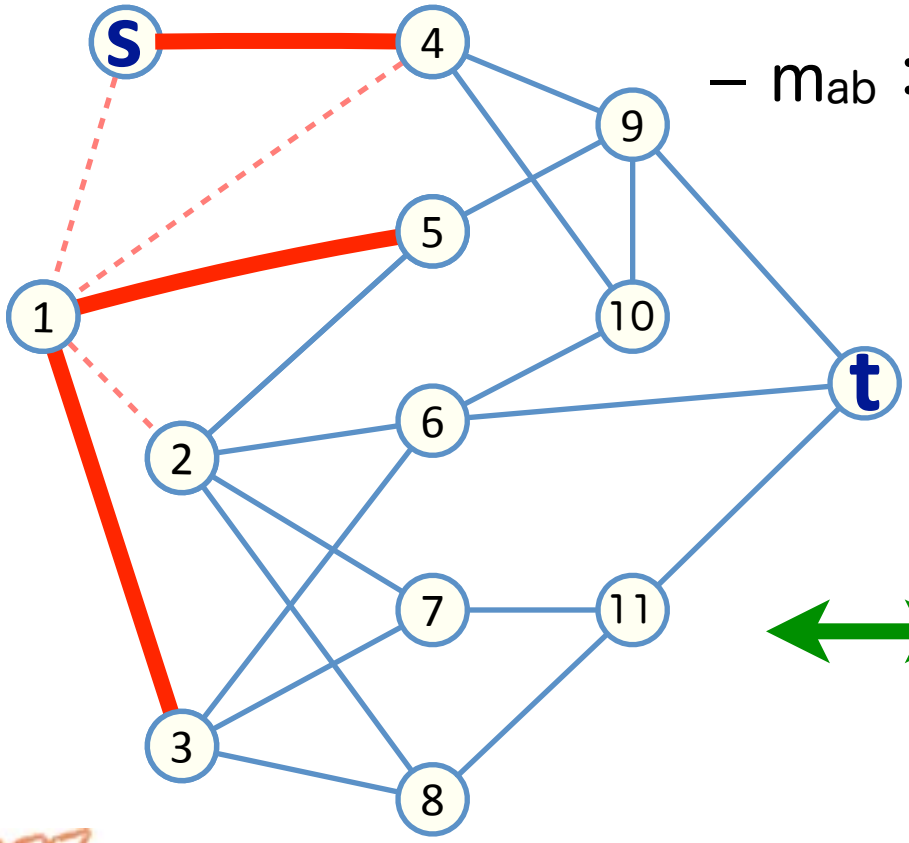
- 未使用の節点 v については (v–v) と約束すると便利
  （長さ0のパス）
- v に接続する全ての辺について処理が終わったら
  - (v–v) は忘れる
  - (v–u) のように v が内点なのに次数１をもつなら枝刈り
- 最後の mate は [(**s**, **t**)] になりますよ！

mate-ZDD 法（提案手法）

- Knuth 法
  - トップダウンにZDDを作った
  - 作成途中のグラフ構造はZDDではない
- mate-ZDD 法
  - ZDD をボトムアップに作る
  - mate 端点対の組合せも ZDD で管理
  - 多項式回の基本代数演算でZDDを完成

Mate-ZDD 法

- mate-ZDD 法での変数
  - e_{ab}：辺 ab を表す変数
  - m_{ab}：mate中の端点対 (a–b) を表す



対応する単項式

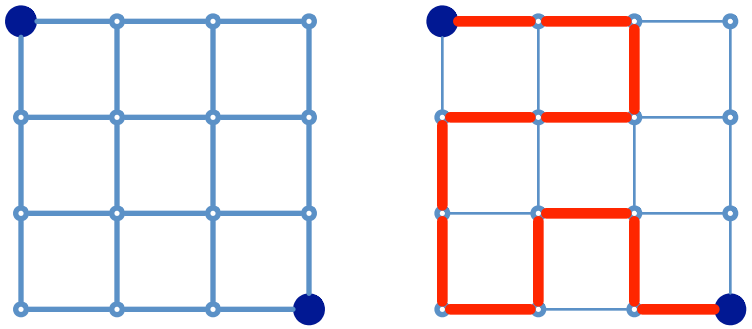
$\longleftrightarrow$ $m_{s4}\ m_{35}\ e_{s4}\ e_{13}\ e_{15}$

ZDD式の更新

- 初期値
  - $F = m_{11}\ m_{22}\ \ldots\ m_{tt}$
- 辺 ab を処理するとき
  それぞれの節点 c,d に対して
  $F = F + (F/m_{ac}/m_{bd}) \times e_{ab} \times m_{cd}$
  - [...,(a–c),(b–d),...] → [...,(c–d),...]
- 節点 a に接続する辺の処理が全て終わったとき
  - それぞれの節点 b ≠ a に対して
    $F = F \ \%\ m_{ab}$ （剰余算）
    m_{ab} で割り切れる＝ a の次数が1
  - $F = F / m_{aa} + F \ \%\ m_{aa}$

SIMPATH vs mate-ZDD 法

n x n 正方格子グラフ上の対角パスの列挙　　所要時間（秒）



n	SIMPATH	Mate-ZDD
8	0.03	0.44
9	0.15	1.92
10	0.63	6.00
11	1.88	23.75
12	6.25	148.11

- Knuth法 は速い
- mate-ZDD法 は
  - コーディングが易しい
  - 可塑性に優れる
- 両アルゴリズムについてパズルへの応用を試す

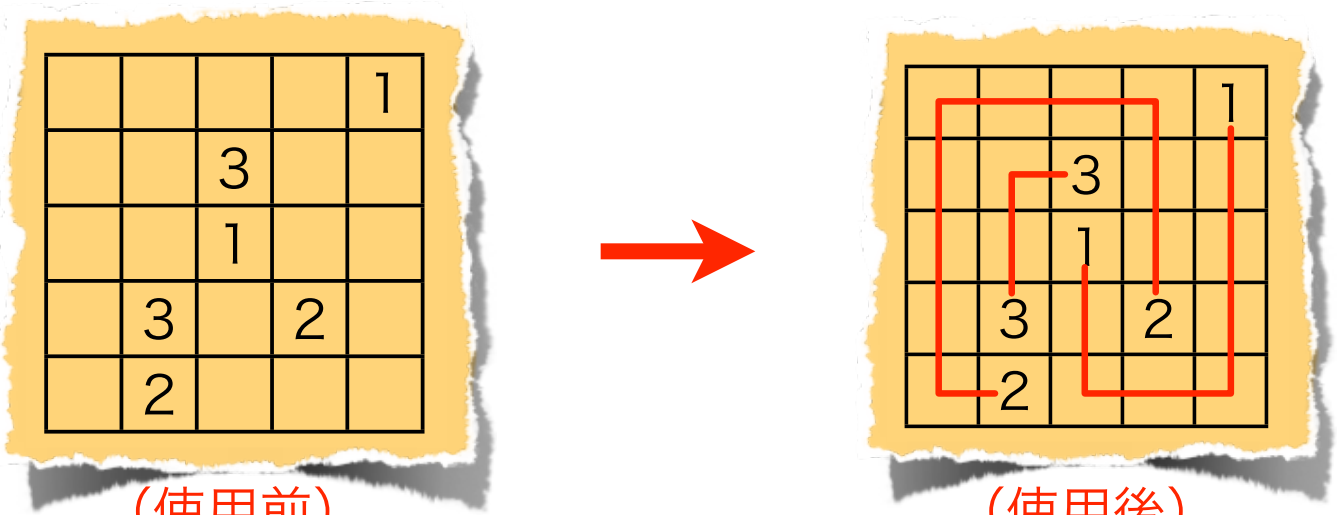
リンクパズルへの応用

	自動解答器	問題生成器
ナンバーリンク	① mate-ZDD base	③ Simpath base
スリザーリンク	② mate-ZDD base	④ mate-ZDD base

① ナンバーリンクソルバー

- 特定２頂点間のパスの列挙

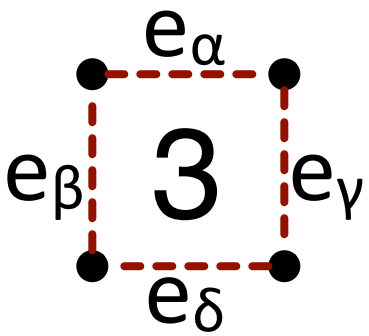
- 特定 p 組２頂点間のパスマッチング列挙



（使用前）　（使用後）

② スリザーリンクソルヴァー

- サイクルの列挙に制約をかます
- mate-ZDD法では辺 e_{ij} を保持しているので,
  各セルの条件と罫線の数の一致を確かめられる



$$F = (F/e_\alpha/e_\beta/e_\gamma\%e_\delta)\, e_\alpha e_\beta e_\gamma$$
$$+ (F/e_\alpha/e_\beta\%e_\gamma/e_\delta)\, e_\alpha e_\beta e_\delta$$
$$+ (F/e_\alpha\%e_\beta/e_\gamma/e_\delta)\, e_\alpha e_\gamma e_\delta$$
$$+ (F\%e_\alpha/e_\beta/e_\gamma/e_\delta)\, e_\beta e_\gamma e_\delta$$

実験

- 対照：
  汎用SATソルバー Sugar（田村直之）
  - パズルから帰着も用意済み

ナンバーリンク対決

	level	size	Solver based on Mate-ZDD	Sugar
1	easy	8 x 8	0.76	1.04
15	easy	8 x 8	0.54	1.07
30	easy	10x10	2.09	1.80
43	medium	10x10	8.41	1.80
52	medium	10x10	1.54	1.15
64	medium	10x10	3.74	1.18
72	hard	10x10	1.47	1.28
79	hard	10x10	6.81	2.09
85	hard	20x15	> 24 hours	> 24 hours
99	hard	20x15	> 24 hours	> 24 hours

Solver based on Mate-ZDD

- 処理する辺の順序に性能が大きく依存

例：

http://www.pori2.net/puzzle/numberlink/03/index.html

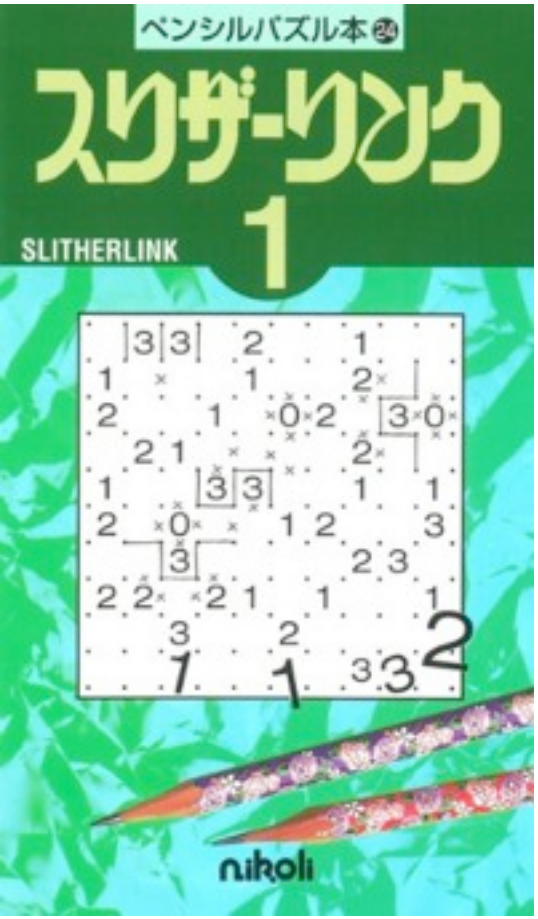
Mate-ZDD

左上から右下へ：2.5 秒

右下から左上へ：866 秒

Sugar：約 2 秒

スリザーリンク対決



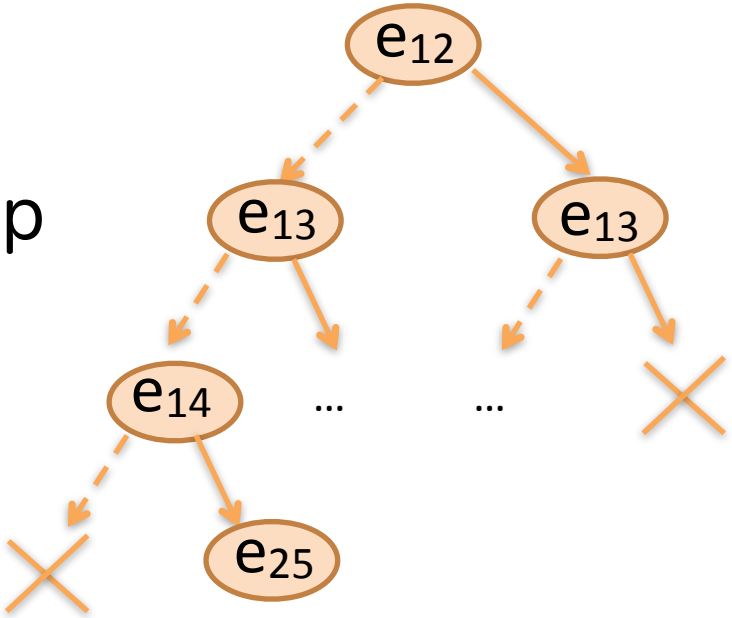
	size	Solver based on Mate-ZDD (sec)	Sugar (sec)
1	10x10	1.00	2.92
12	10x10	3.72	3.62
25	10x10	4.47	2.89
37	10x10	2.19	2.64
43	18x10	21.58	3.74
54	18x10	125.39	3.73
68	24x14	5711.49	6.20
77	24x14	2311.87	5.46
89	36x20	49 hours	35.24
96	36x20	> 24 hours	120.78

- ZDD の十八番は全列挙
- １個だけ解を求めればいいだなんて・・・

そこで　問題生成（列挙）

③ ナンバーリンク問題生成

- 入力：
  盤面サイズ m x n と端点対数 p
- 出力：
  唯一解を持つ出題の全列挙
- 方法：
  Knuth法.
  ZDDでの合流　→　複数解　→　棄却.
  全辺の処理を終えて p 組の端点対を与える mate が欲しい出題そのもの.



③ ナンバーリンク問題生成

4x4 格子

p	生成問題数	計算時間 (秒)
1	0	0.74
2	0	0.92
3	496	1.81
4	3454	3.52
5	6820	4.43
6	4652	4.59
7	952	4.57
8	36	4.56
計	16410	

5x5 格子

p	生成問題数	計算時間 (秒)
1	0	0.89
2	0	119.9
>2	?	N/A
計		

③ ナンバーリンク問題生成

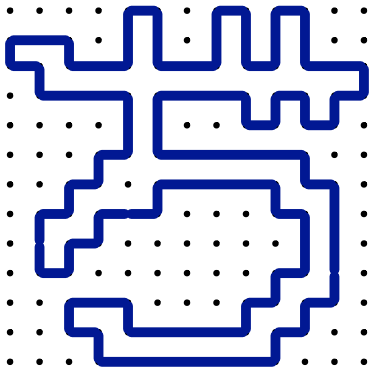
- 小さい盤面でしか全列挙できない
- 適当に問題を作らせると
  くだらない†出題が多い

(†個人の感想です)

④ スリザーリンク問題生成

- 入力：

  m x n 格子グラフ上のサイクル

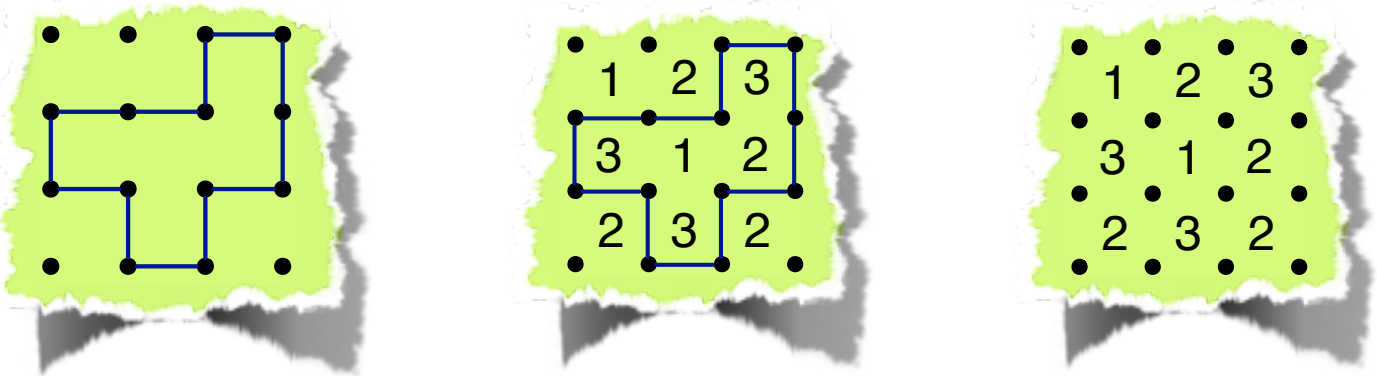
  (絵画的スリザァリンク)

- 出力：

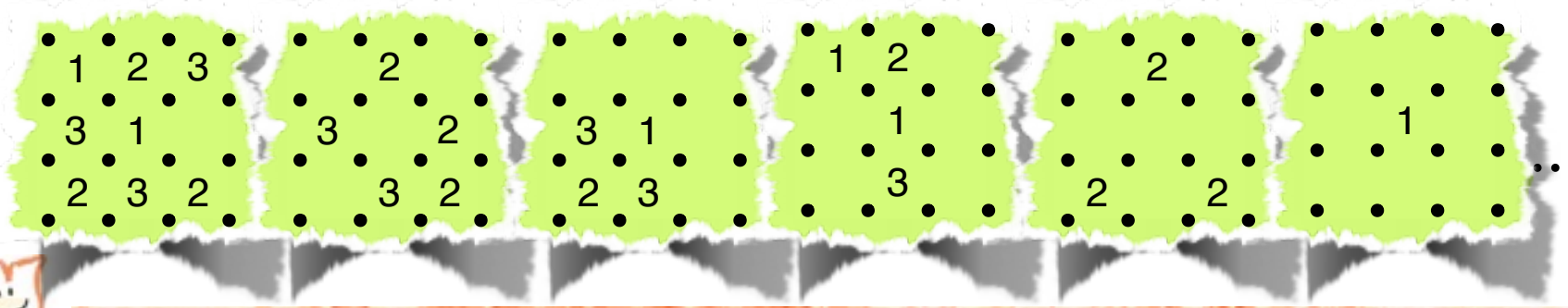
  唯一解を与えるヒントの組合せを全列挙

④ スリザーリンク問題生成

- mate-ZDD 法ベース
- 与えられた解に対して，完全ヒントを生成



- ヒントアリ／ナシのパタンごとに別解の有無をチェック



④ スリザーリンク問題生成

- 完全ヒントなども含まれる
- 問題のランダム生成が可能
- 最小ヒント数のもののみを抽出したりできる
- 小さい盤面でもいやらしい†問題ができる

(†個人の感想です)

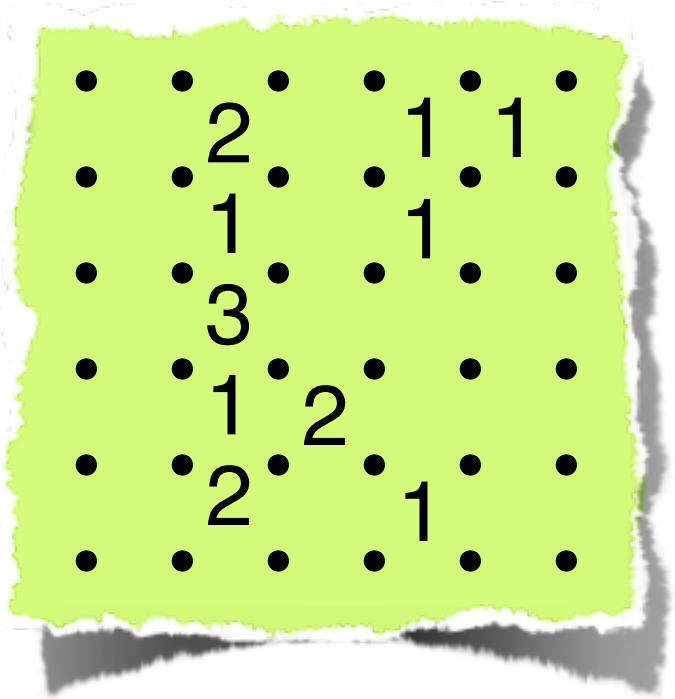
まとめ

- パス／サイクルを列挙するアルゴリズム
  - Simpath （Knuth）
  - Mate-ZDD （提案手法）

  をリンクパズルに応用した
- ソルバーはあまり強力ではないが改良の余地あり
  - 辺・ヒント変数の順序など
- 問題生成も時間がかかって大きな盤面には不向き
- ときどき面白いパズルが出来て楽しかった†

(†個人の感想です)

（演習問題１）次のスリザーリンクを解け



（問題生成器による出題）
極端に難しくはないものの，
通常の定石はあまり役に立たず，
ただ単に深読みが要求される．
解けてもそれが唯一解だという
確信を持ちにくい．

（演習問題２）

６ｘ６ 方眼上で，０ヒントのみを10個使って
解のないスリザーリンク問題を作れ

（問題生成器の教えるところによると10個が最小）