情報数理科学特別講義 D ／情報数理科学特別講義 H ／情報数理科学特殊講義 B
第 1 回
組合せ最適化による現実問題へのアプローチ (計算困難問題の対処法)

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学

2012 年 8 月 27 日

最終更新：2012 年 8 月 29 日　12:02

基本情報：講師

- **講師：岡本 吉央 (おかもと よしお)**
- **本務：電気通信大学 大学院情報理工学研究科 情報・通信工学専攻 准教授**
- Email：okamotoy@uec.ac.jp
- Web: http://dopal.cs.uec.ac.jp/okamotoy/

基本情報：講義

- Web: http://dopal.cs.uec.ac.jp/okamotoy/lect/2012/osakafu-u/
- **内容：**

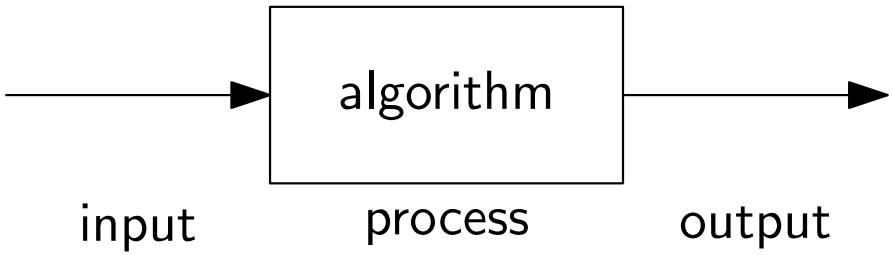
現実世界の問題解決のためには,「問題解決に対する一般的方法論」を基にした「問題固有の構造による方法論の特殊化」が重要である．この授業では特に組合せ最適化に焦点を絞って，この 2 点の活用法，そして，そこへ数理的にアプローチする方法論を解説する．

- **成績評価：レポートによる**

アルゴリズムとは

アルゴリズムがすること (簡単に言うと)

- データを受け取り (入力)
- それを処理し (処理)
- 結果を返す (出力)



アルゴリズムに要求されること

- どんな入力に対しても (入力の普遍性)
- 高速に処理し (処理の高速性)
- 正しい結果を返す (出力の正当性)

要求項目のトレードオフ

あちらをたてれば，こちらがたたず...

- どんな入力に対しても正しい結果を返さなければならない
  ⇒ 処理の高速化に限界があるかもしれない
- 正しい結果を高速に返さなければならない
  ⇒ すべてのデータを処理できないかもしれない
- どんな入力に対しても高速に処理をしなければならない
  ⇒ 正しい結果を返せないかもしれない

計算理論的には「NP 困難性」などの枠組で議論される

直観的な言い方

問題が NP 困難 ⇒ その問題に対して前ページの「3 つの要求」をすべて満たすアルゴリズムは存在しないと思われている

現実世界の付帯条件

スケジューリングのような問題にも様々

- **中学校の時間割作成**
  教員数 30 **名程度，学級数** 15 **程度，**1 **年に** 1 **回作成**
- **銀行** ATM **の現金補充計画作成**
  ATM **数** 200 **個程度，拠点数** 20 **個程度，**1 **年に** 1 **回程度作成**
- **看護師の勤務表作成**
  看護師数 25 **名程度，**3 **交替制，**1**ヵ月に** 1 **回作成**

考察

状況に応じて，次は変わる

- 問題の規模
- 問題の特殊さ
- 計算にかけられる時間
- 解の質

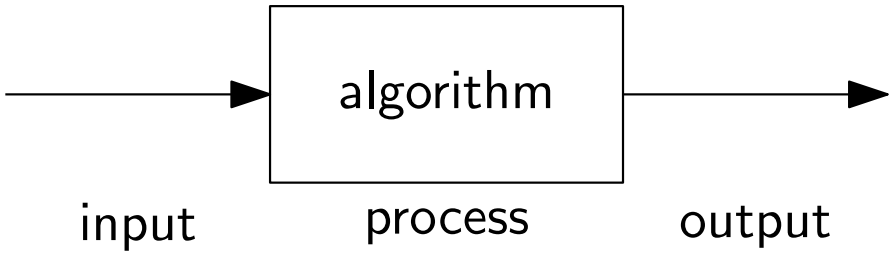
状況に応じて，アルゴリズムも変わる

- 問題が特殊 $\Rightarrow$ 特殊な問題にのみ動くアルゴリズム
- 計算時間が長くてもよい $\Rightarrow$ 厳密なアルゴリズム
- 解の質が悪くてもよい $\Rightarrow$ 近似的なアルゴリズム

アルゴリズムとは (再掲)

アルゴリズムがすること (簡単に言うと)

- データを受け取り (入力)
- それを処理し (処理)
- 結果を返す (出力)



アルゴリズムに要求されること

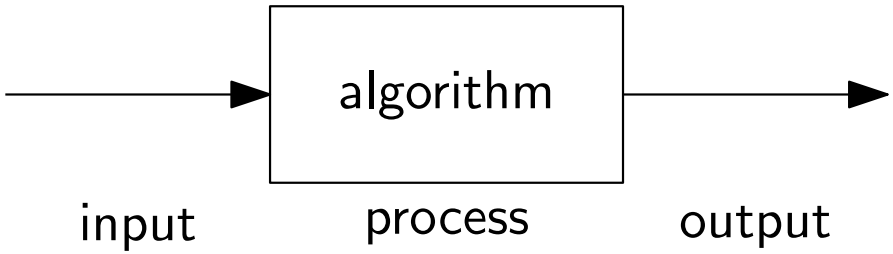
- どんな入力に対しても (入力の普遍性)
- 高速に処理し (処理の高速性)
- 正しい結果を返す (出力の正当性)

この 3 つの要求を緩和できる場合がある

緩和の仕方 (1):「入力の普遍性」の緩和

アルゴリズムに要求されること

- どんな入力に対しても (入力の普遍性)
- 高速に処理し (処理の高速性)
- 正しい結果を返す (出力の正当性)



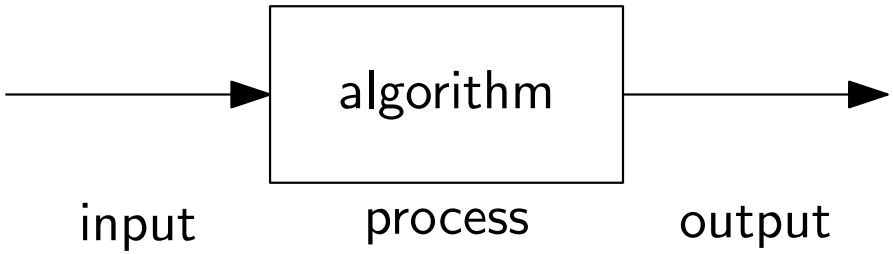
「入力の普遍性」の緩和

- 一部の入力に対して (入力の普遍性の緩和)
- 高速に処理し (処理の高速性)
- 正しい結果を返す (出力の正当性)

緩和の仕方 (2)：「処理の高速性」の緩和

アルゴリズムに要求されること

- どんな入力に対しても (入力の普遍性)
- 高速に処理し (処理の高速性)
- 正しい結果を返す (出力の正当性)



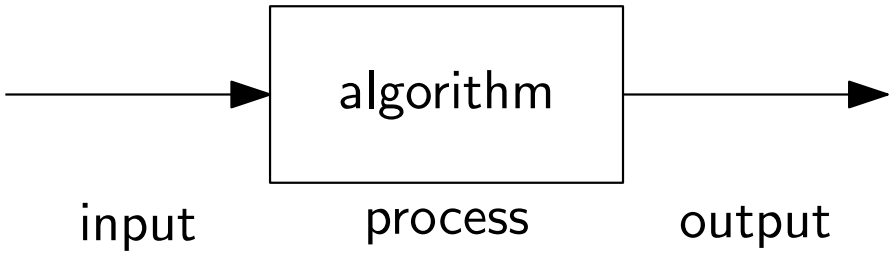
「処理の高速性」の緩和

- どんな入力に対しても (入力の普遍性)
- ある程度の速さで処理し (処理の高速性の緩和)
- 正しい結果を返す (出力の正当性)

緩和の仕方 (3)：「出力の正当性」の緩和

アルゴリズムに要求されること

- どんな入力に対しても (入力の普遍性)
- 高速に処理し (処理の高速性)
- 正しい結果を返す (出力の正当性)



「出力の正当性」の緩和

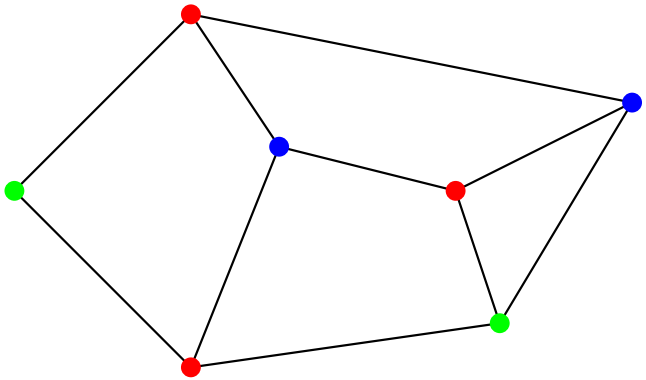
- どんな入力に対しても (入力の普遍性)
- 高速に処理し (処理の高速性)
- 近似的な結果を返す (出力の正当性の緩和)

彩色問題

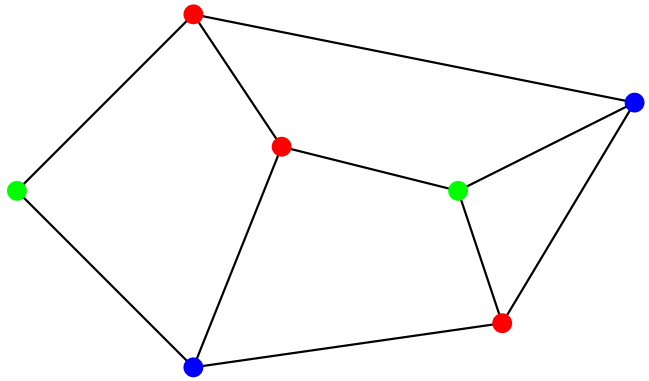
彩色問題

- 入力：無向グラフ $G = (V, E)$
- 出力：G の彩色
- 目的：使用する色数の最小化

G の彩色：G の頂点への色の割当で，各辺の両端点の色が異なるもの

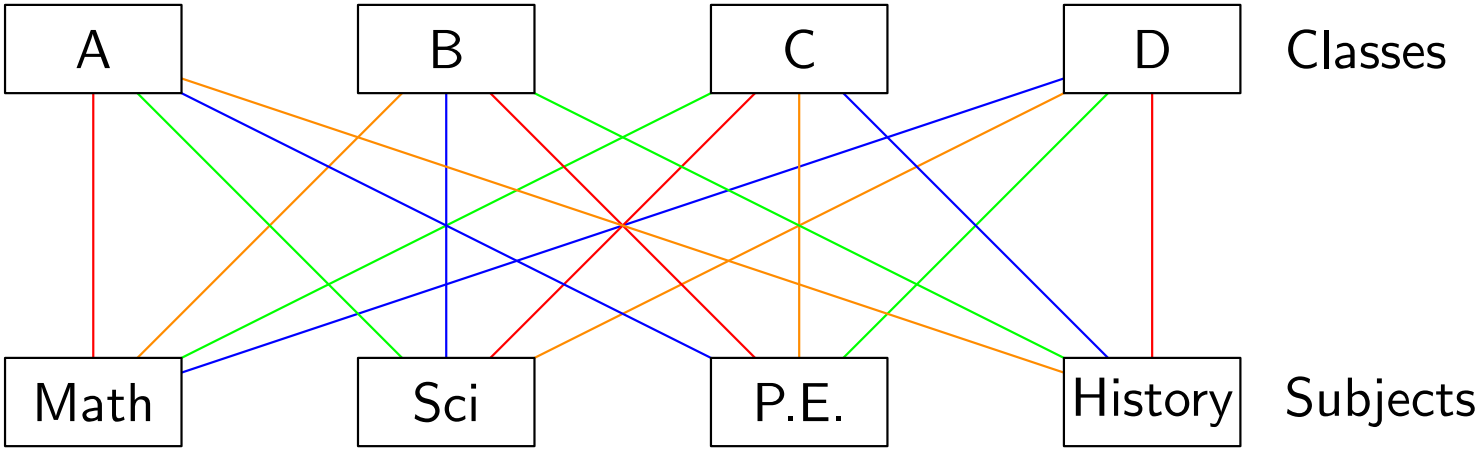


彩色である



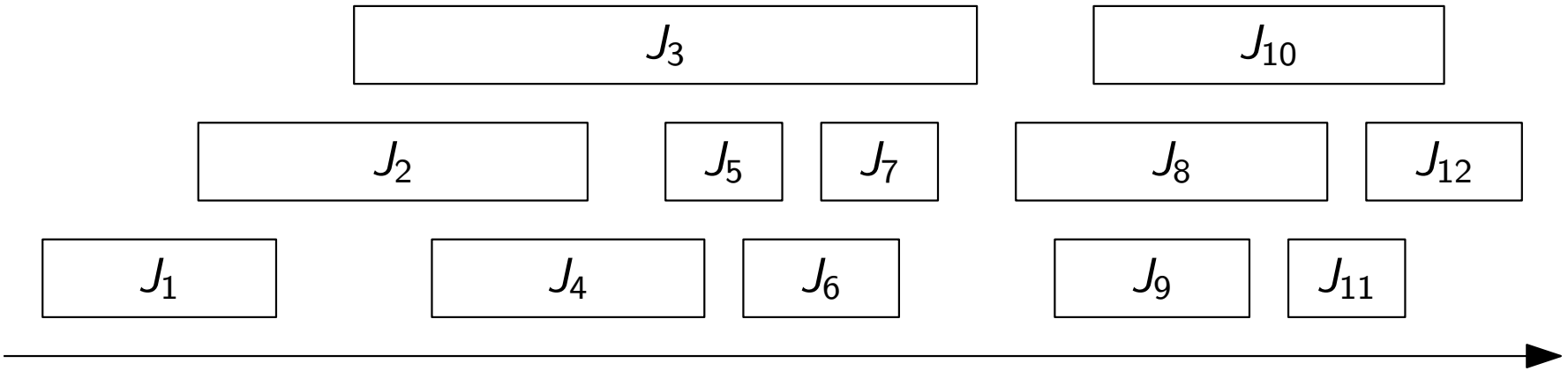
彩色ではない

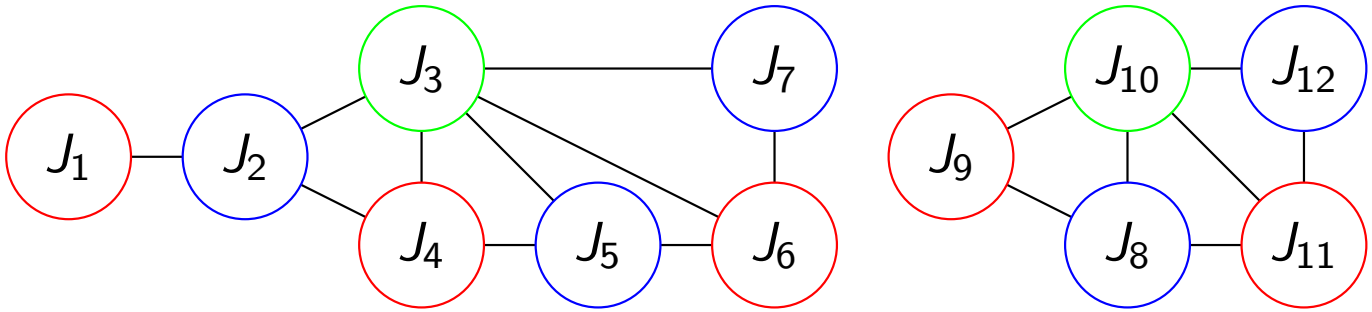
彩色問題が現れる場面 (1)：時間割作成



	A	B	C	D
1	Math	P.E.	Sci	History
2	Sci	History	Math	P.E.
3	P.E.	Sci	History	Math
4	History	Math	P.E.	Sci

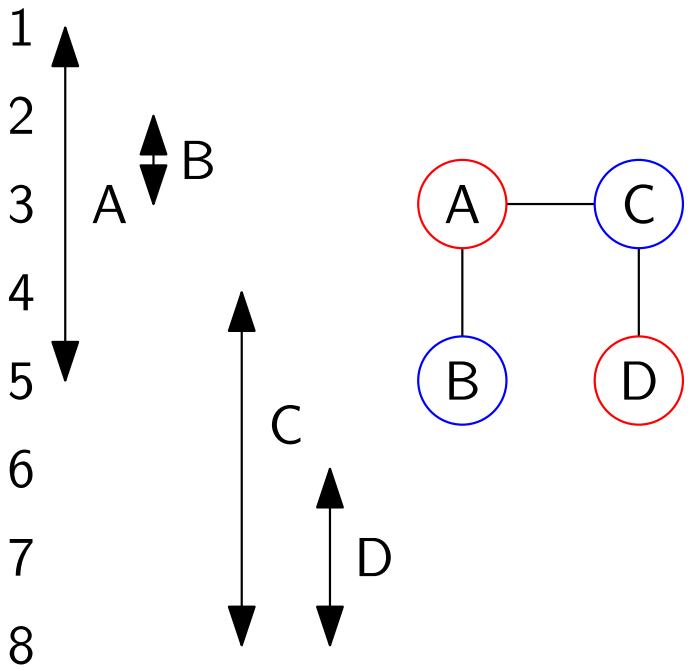
彩色問題が現れる場面 (2)：ジョブスケジューリング





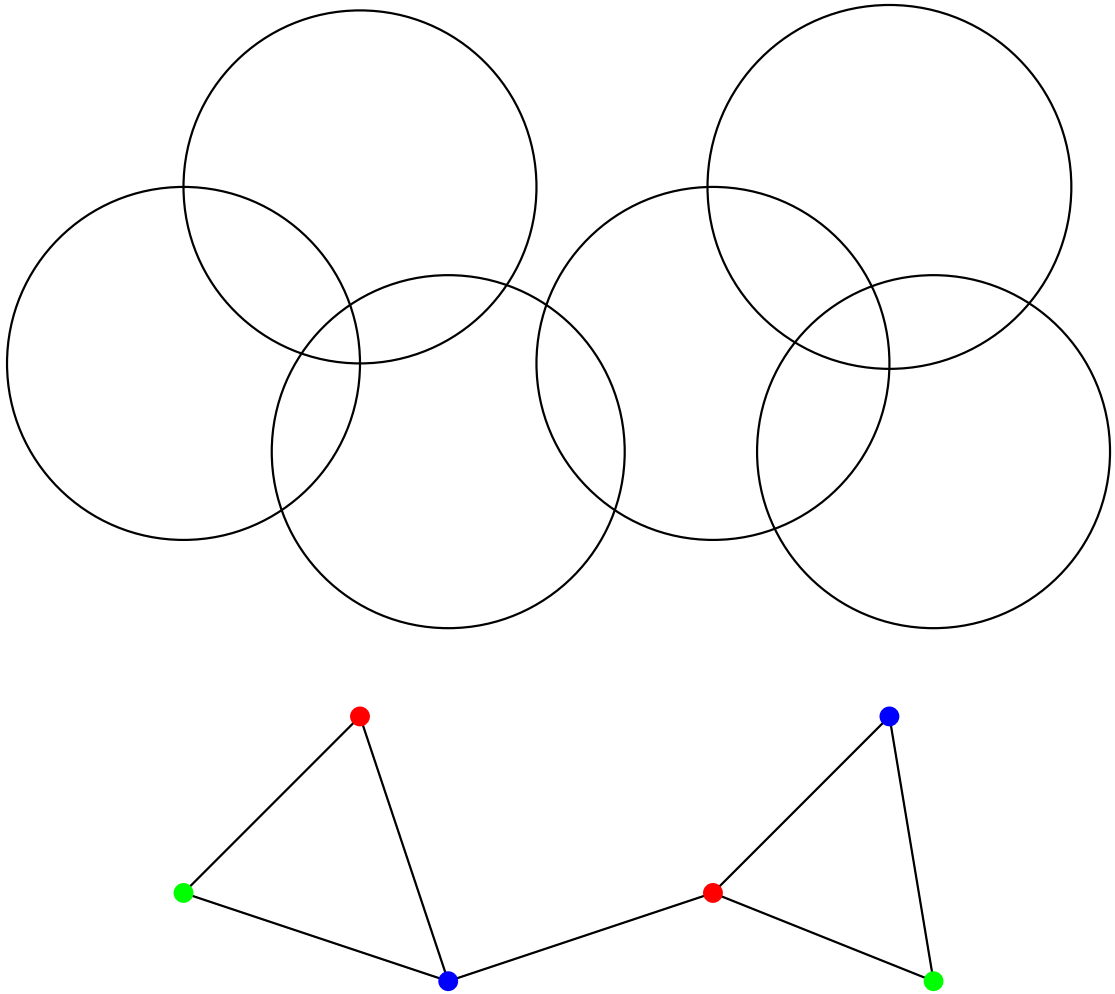
彩色問題が現れる場面 (3)：レジスタ割当

```
1: A = 2
2: B = 3
3: B = B + 2
4: C = A + 1
5: A = C + 3
6: D = 4
7: D = C + 2
8: C = 3
```




```
1: R1 = 2
2: R2 = 3
3: R2 = R2 + 2
4: R2 = R1 + 1
5: R1 = R2 + 3
6: R1 = 4
7: R1 = R2 + 2
8: R2 = 3
```

彩色問題が現れる場面 (4)：移動体通信における周波数割当



彩色問題は NP 困難

事実 (Karp '72)

彩色問題は NP 困難

つまり，

帰結

彩色問題に対して

- **どんな入力に対しても** **(入力の普遍性)**
- **高速に処理し** **(処理の高速性)**
- **正しい結果を返す** **(出力の正当性)**

アルゴリズムは存在しないと思われている

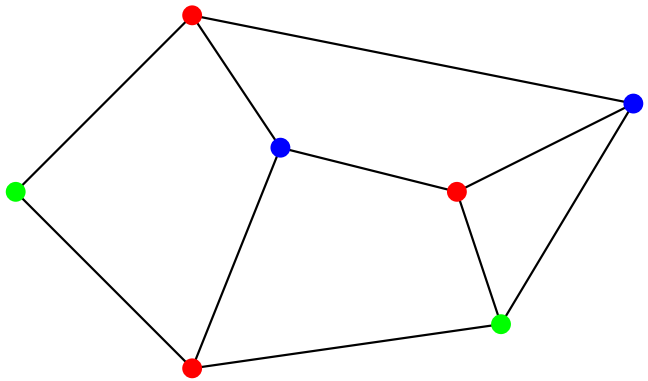
この講義で見ていくこと

- 「妥協」をするようなアルゴリズムをどのように設計するか
- そのようなアルゴリズムの理論的解析をどのようにするか

厳密に解きたい場合

次の判定問題を考える

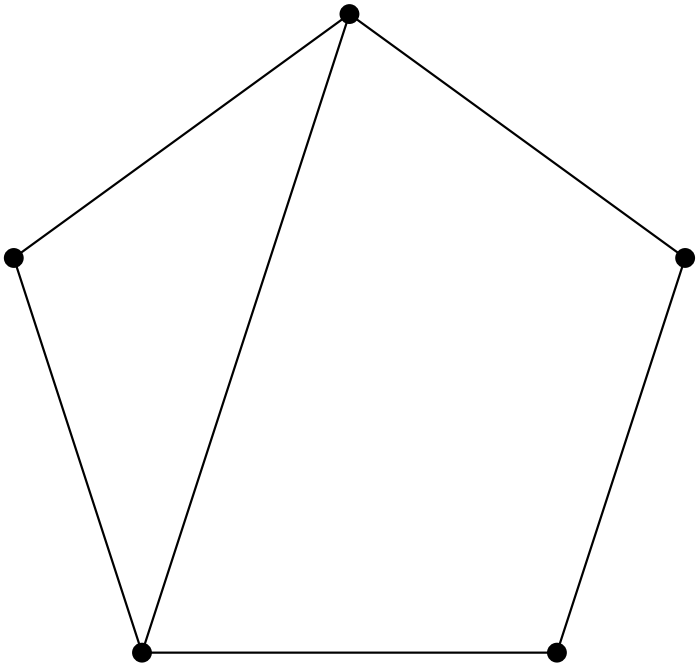
与えられた無向グラフ G と自然数 k に対して，「G は k 色で塗れるか」

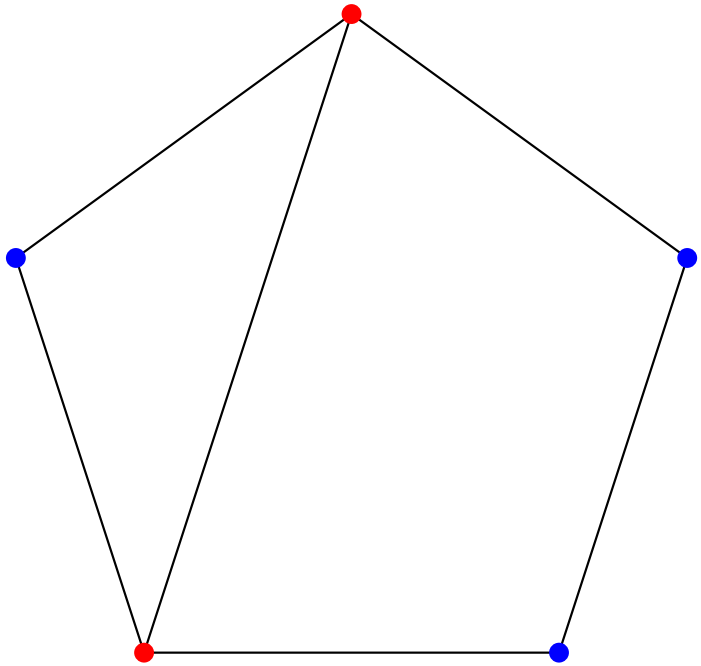


- $k = 2$ **のとき，答えは** No
- $k = 3$ **のとき，答えは** Yes
- $k = 4$ **のとき，答えは** Yes

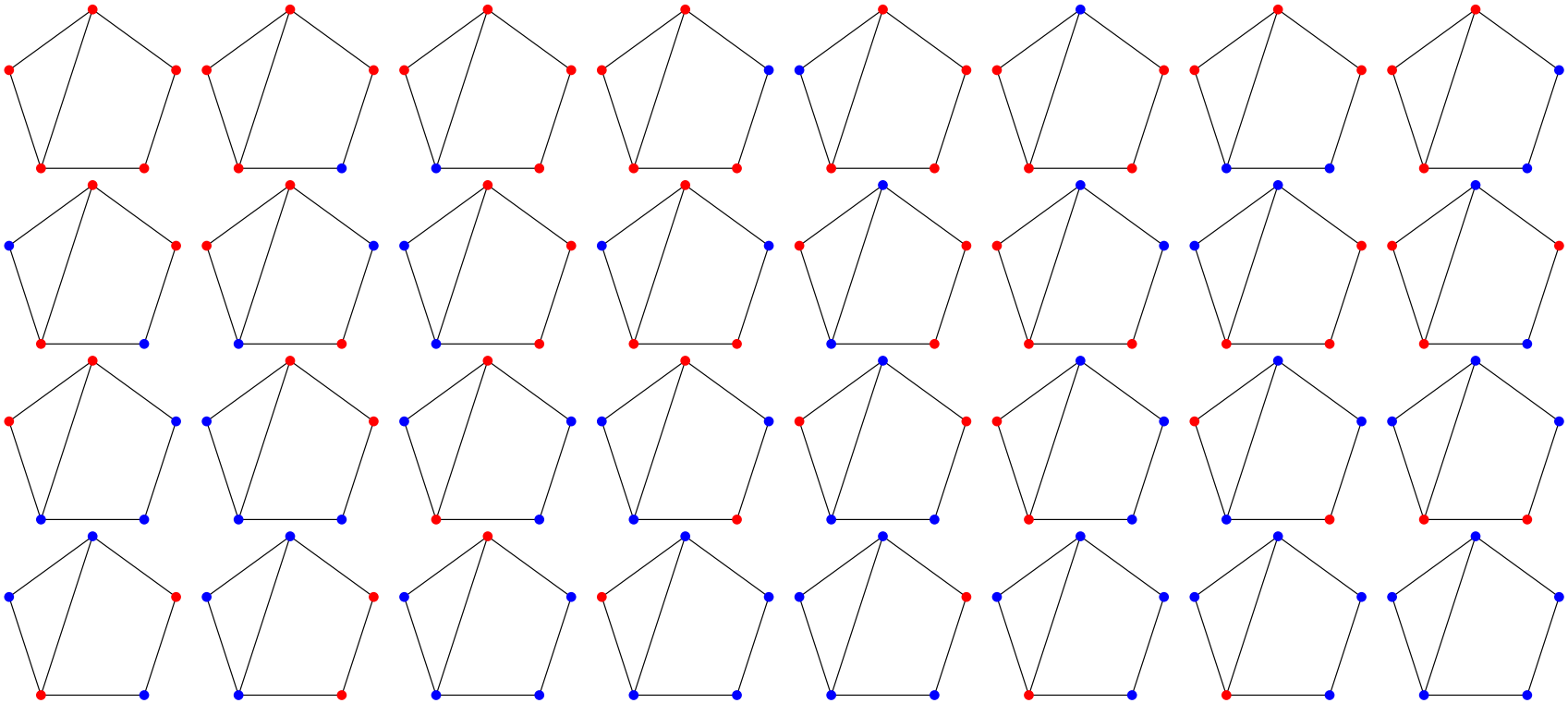
厳密に解きたい場合：単純なアルゴリズム 1

- 頂点への色の割り当て方をすべて試す





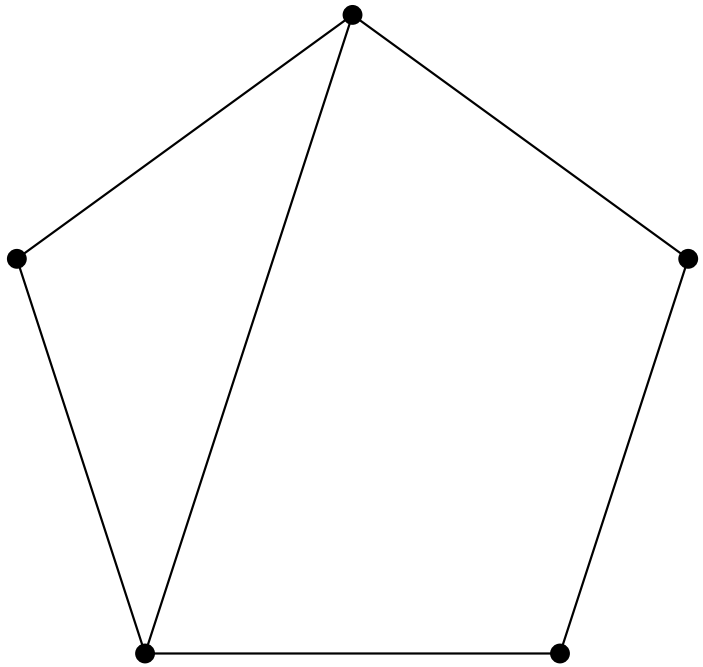
厳密に解きたい場合：単純なアルゴリズム 1 (すべて試した)



- 試す色の割り当て方の総数 $= k^n$　　(n はグラフの頂点数)

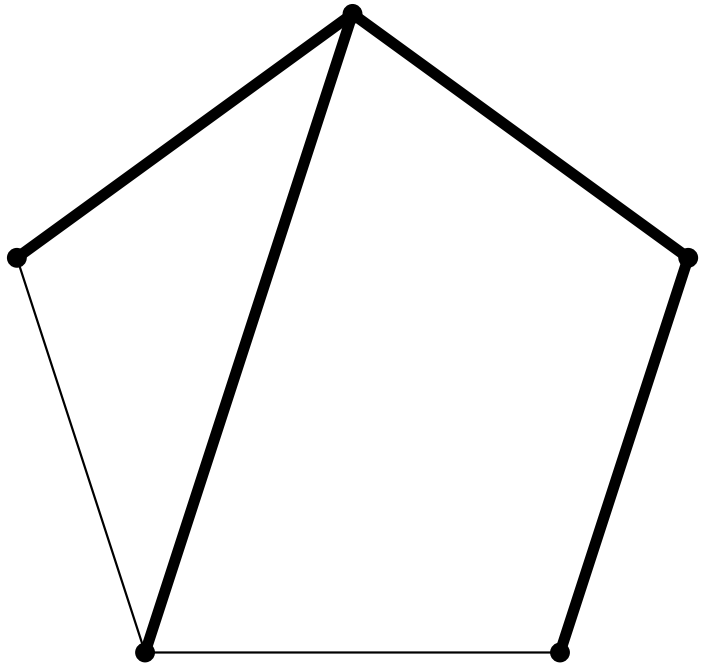
厳密に解きたい場合：少し賢いアルゴリズム 2

1. G の全域木 (全張木) を 1 つ見つける
2. それに沿って色を割り当てる
   - 木の深さ優先探索順に色を割り当て
   - 頂点に色を割り当てるときはその親と違う色を割り当てる



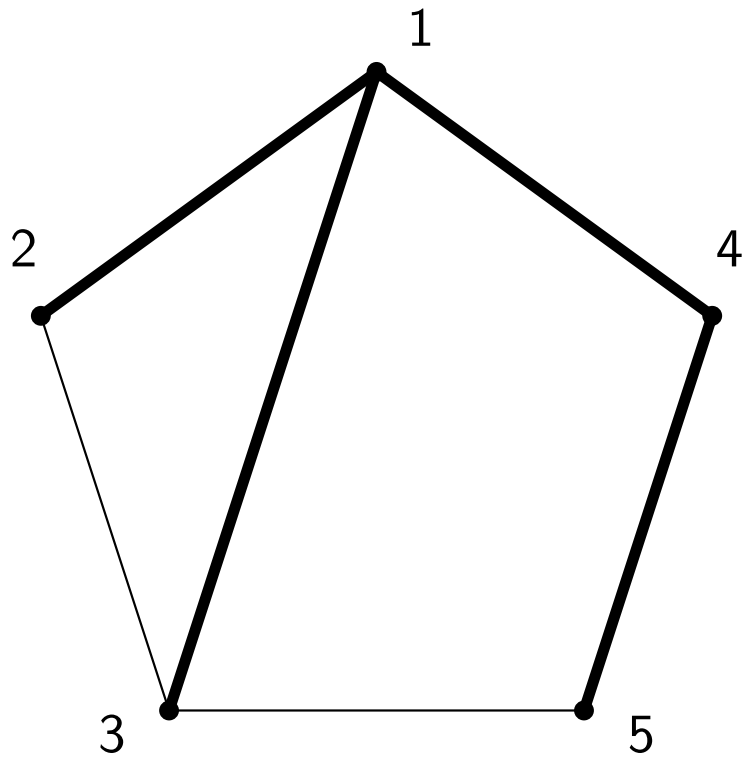
厳密に解きたい場合：少し賢いアルゴリズム 2

1. G の全域木 (全張木) を 1 つ見つける
2. それに沿って色を割り当てる
   - 木の深さ優先探索順に色を割り当て
   - 頂点に色を割り当てるときはその親と違う色を割り当てる



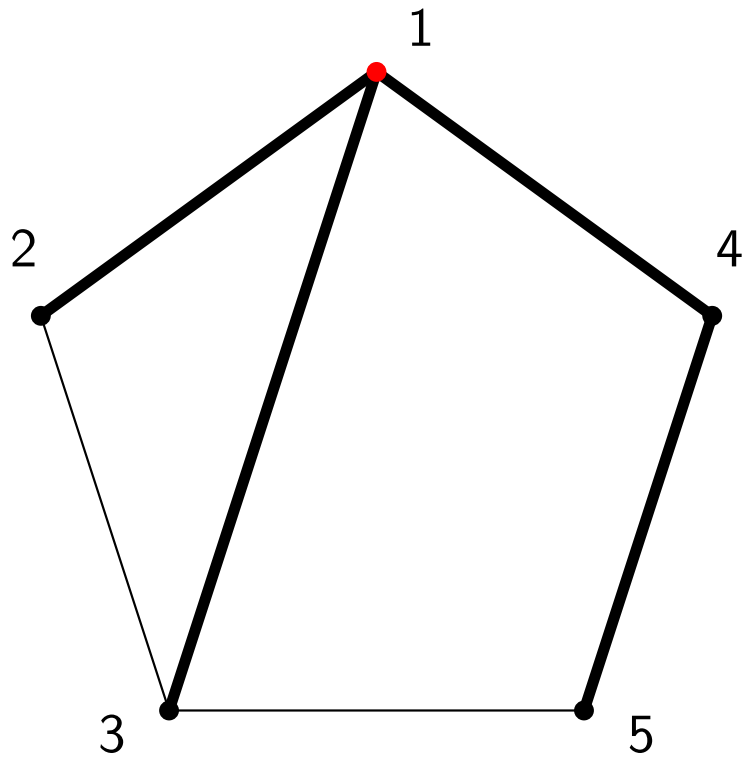
厳密に解きたい場合：少し賢いアルゴリズム 2

1. G の全域木 (全張木) を 1 つ見つける
2. それに沿って色を割り当てる
   - 木の深さ優先探索順に色を割り当て
   - 頂点に色を割り当てるときはその親と違う色を割り当てる



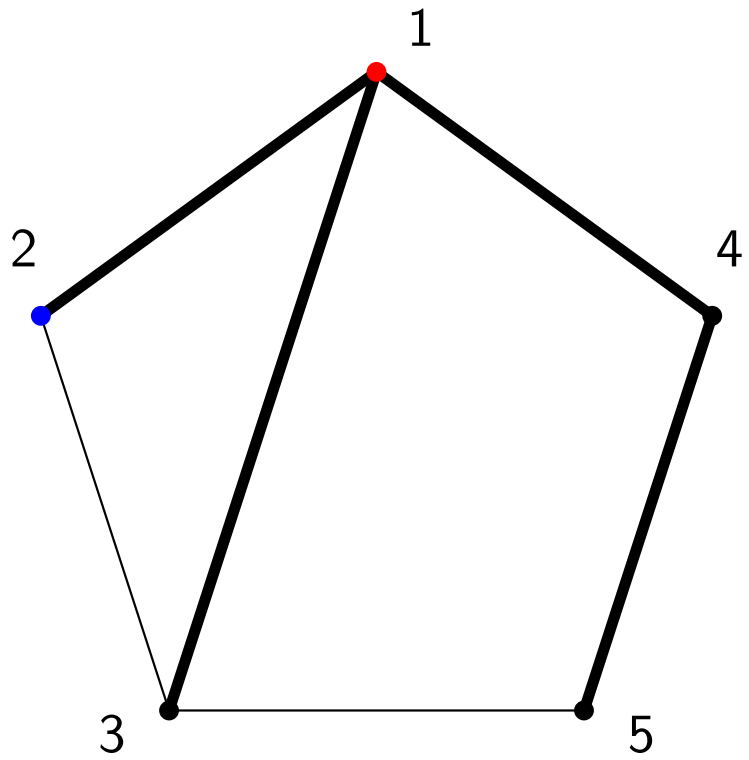
厳密に解きたい場合：少し賢いアルゴリズム 2

1. G の全域木 (全張木) を 1 つ見つける
2. それに沿って色を割り当てる
   - 木の深さ優先探索順に色を割り当て
   - 頂点に色を割り当てるときはその親と違う色を割り当てる



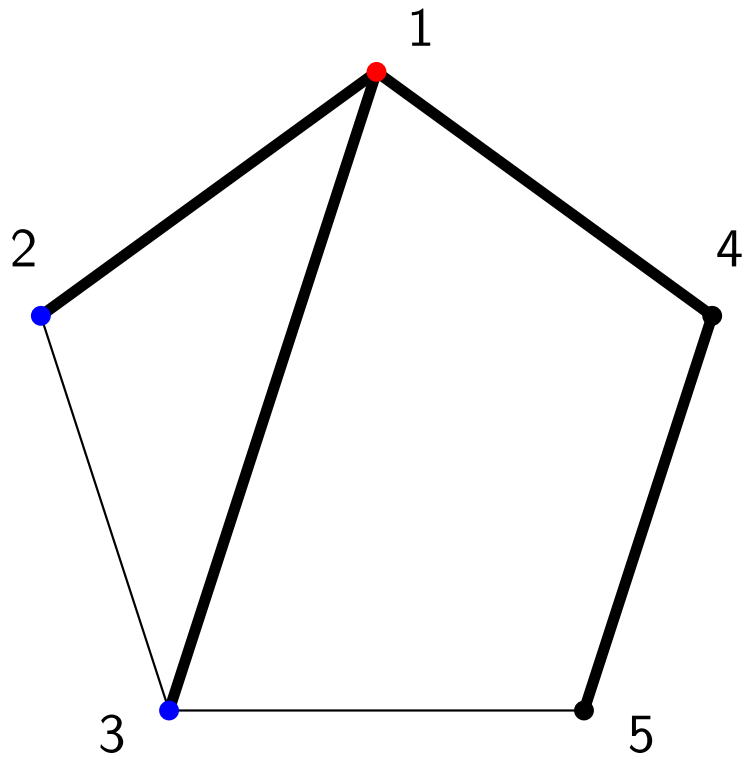
厳密に解きたい場合：少し賢いアルゴリズム 2

1. G の全域木 (全張木) を 1 つ見つける
2. それに沿って色を割り当てる
   - 木の深さ優先探索順に色を割り当て
   - 頂点に色を割り当てるときはその親と違う色を割り当てる



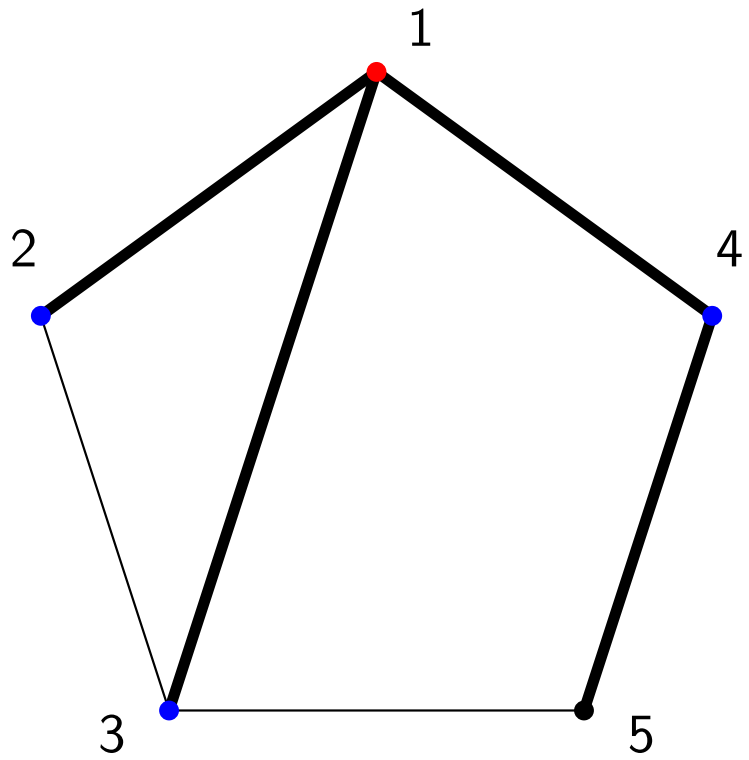
厳密に解きたい場合：少し賢いアルゴリズム 2

1. G の全域木 (全張木) を 1 つ見つける
2. それに沿って色を割り当てる
   - 木の深さ優先探索順に色を割り当て
   - 頂点に色を割り当てるときはその親と違う色を割り当てる



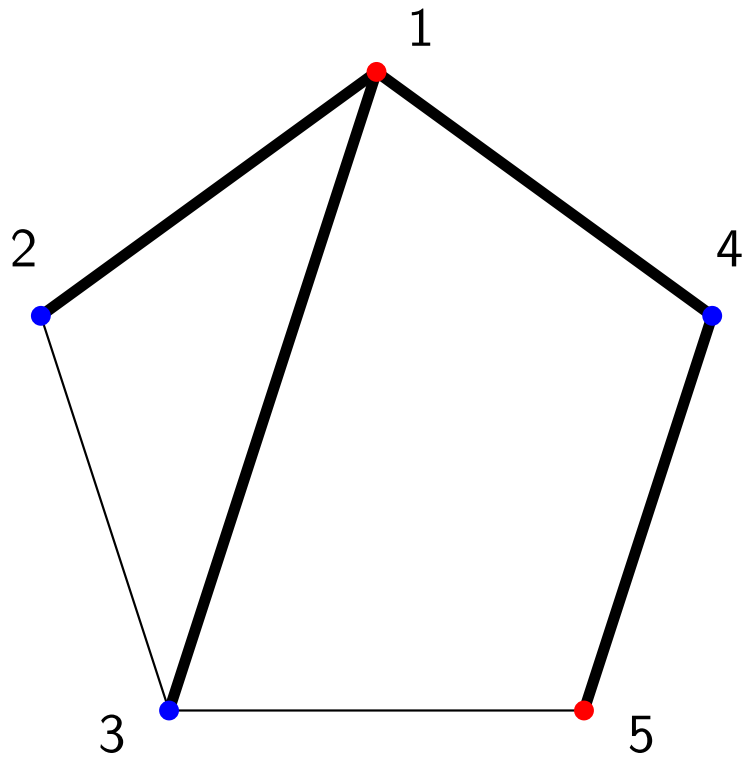
厳密に解きたい場合：少し賢いアルゴリズム 2

1. G の全域木 (全張木) を 1 つ見つける
2. それに沿って色を割り当てる
   - 木の深さ優先探索順に色を割り当て
   - 頂点に色を割り当てるときはその親と違う色を割り当てる

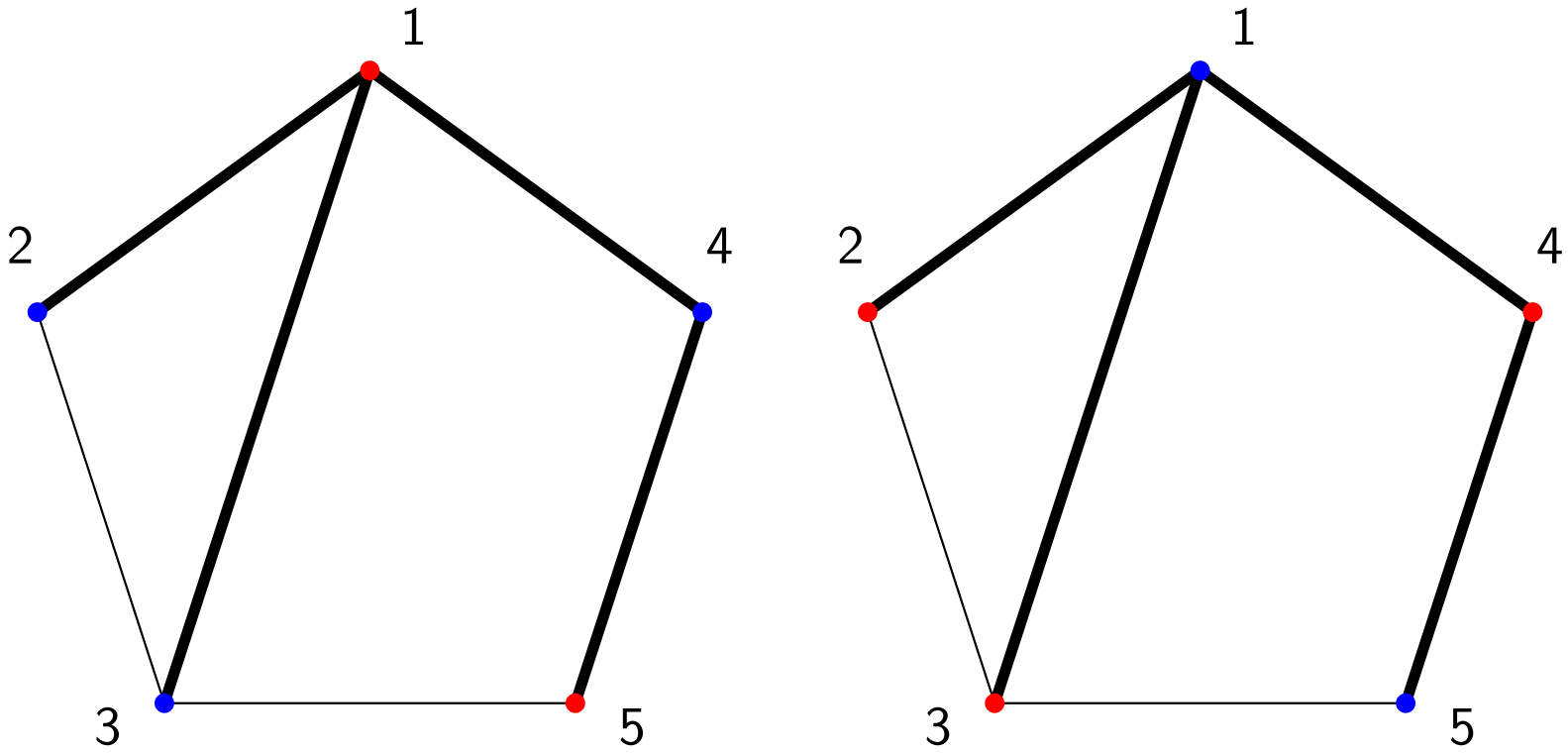


厳密に解きたい場合：少し賢いアルゴリズム 2

1. G の全域木 (全張木) を 1 つ見つける
2. それに沿って色を割り当てる
   - 木の深さ優先探索順に色を割り当て
   - 頂点に色を割り当てるときはその親と違う色を割り当てる



厳密に解きたい場合：少し賢いアルゴリズム 2 (すべて試した)



- 試す色の割り当て方の総数 $= k(k-1)^{n-1}$　　(n はグラフの頂点数)
  - 根以外の頂点には，その親の色を割り当てないから

単純なアルゴリズム 1 と少し賢いアルゴリズム 2 の違い (1)

試す色の割り当て方の総数 ($n = 10$ のとき)

k	アルゴリズム 1	アルゴリズム 2
2	1,024	2
3	59,049	1,536
4	1,048,576	78,732
5	9,765,625	1,310,720
6	60,466,176	11,718,750
7	282,475,249	70,543,872
8	1,073,741,824	322,828,856
9	3,486,784,401	1,207,959,552
10	10,000,000,000	3,874,204,890
11	25,937,424,601	11,000,000,000

単純なアルゴリズム 1 と少し賢いアルゴリズム 2 の違い (2)

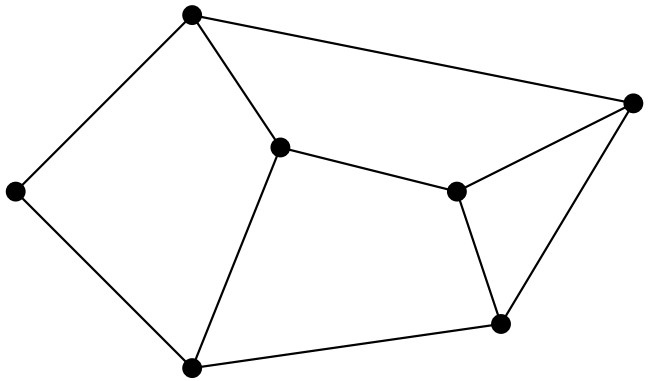
試す色の割り当て方の総数 ($k = 9$ のとき)

n	アルゴリズム 1	アルゴリズム 2
2	81	72
3	729	576
4	6,561	4,608
5	59,049	36,864
6	531,441	294,912
7	4,782,969	2,359,296
8	43,046,721	18,874,368
9	387,420,489	150,994,944
10	3,486,784,401	1,207,959,552
11	31,381,059,609	9,663,676,416

とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

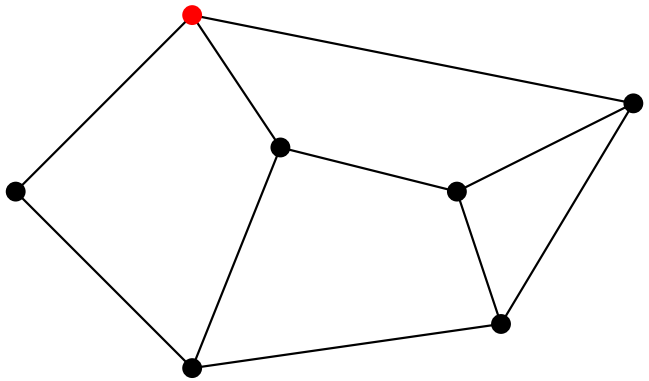
実行例



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

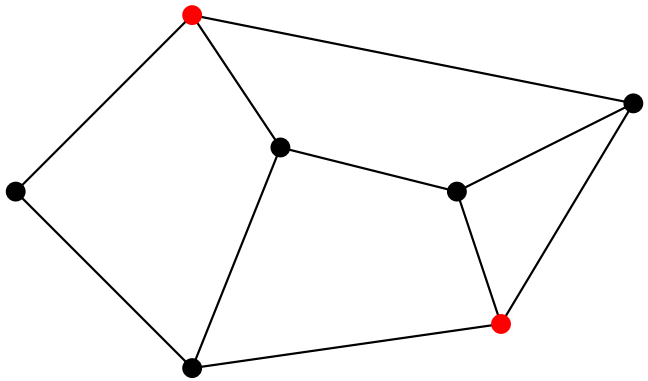
実行例



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

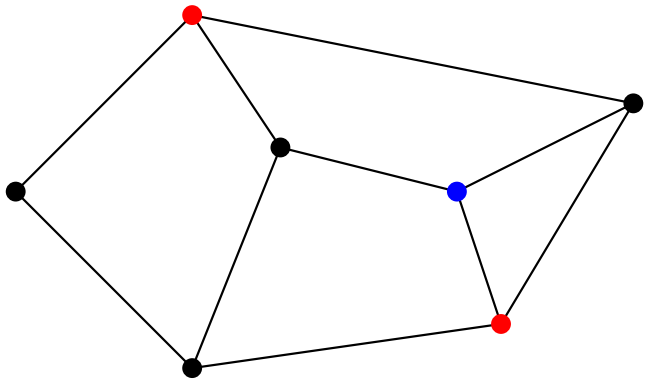
実行例



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

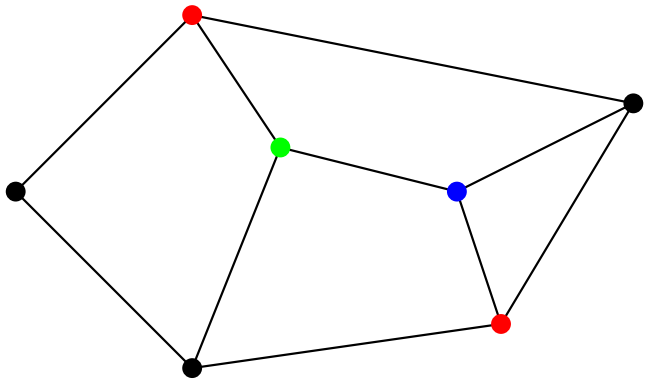
実行例



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

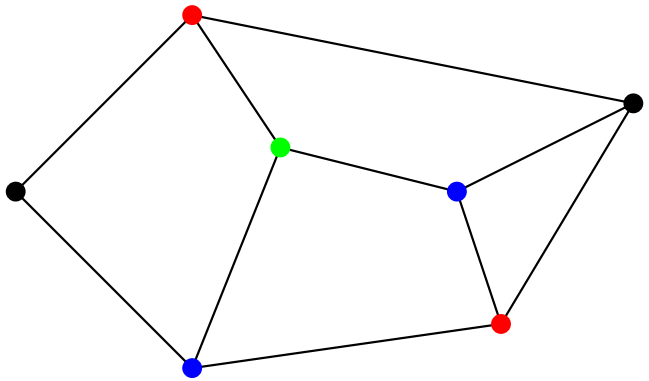
実行例



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

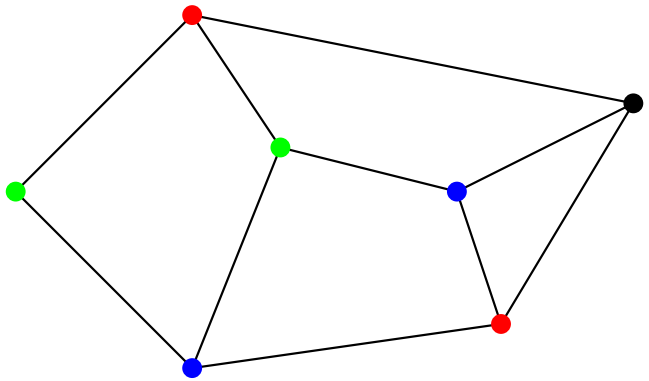
実行例



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

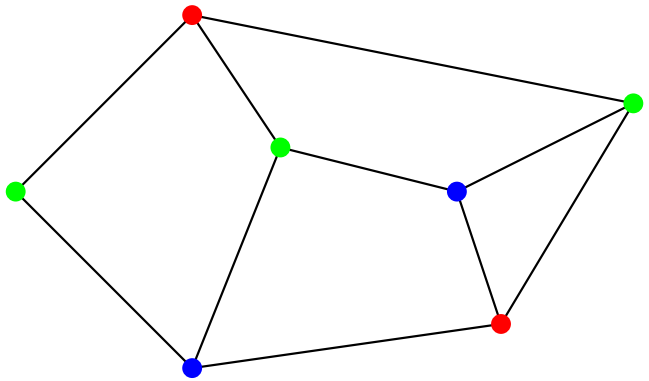
実行例



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

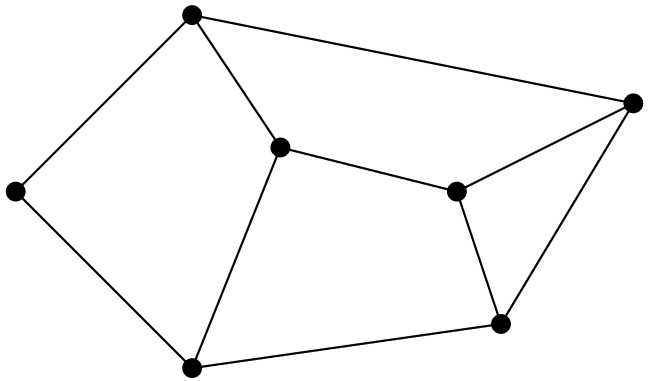
実行例



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

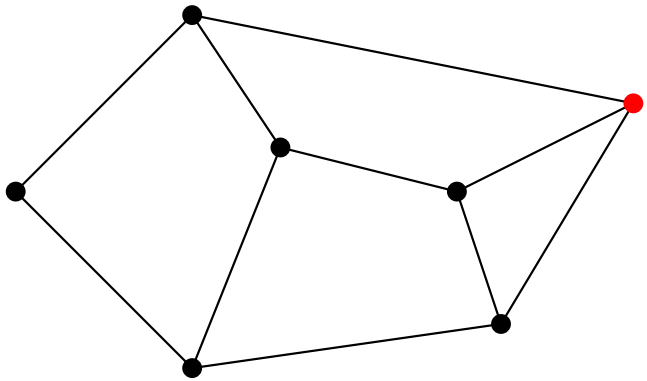
実行例 (別の順序)



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

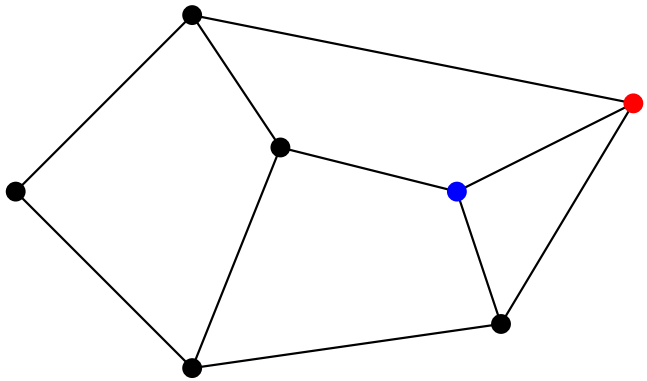
実行例 (別の順序)



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

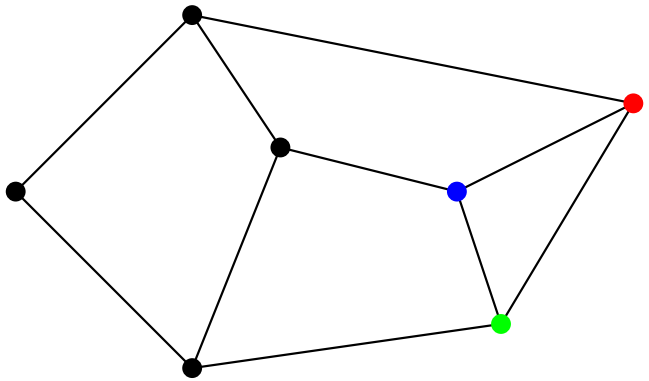
実行例 (別の順序)



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

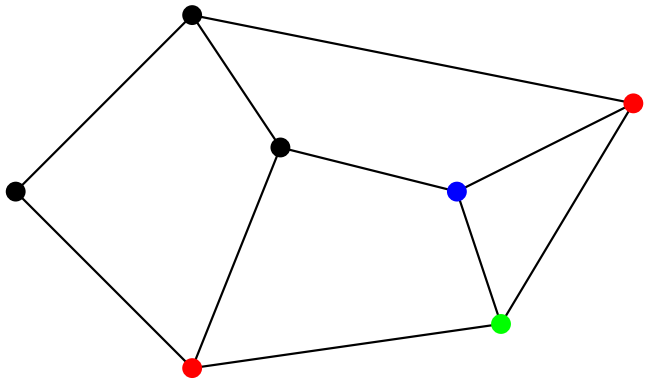
実行例 (別の順序)



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

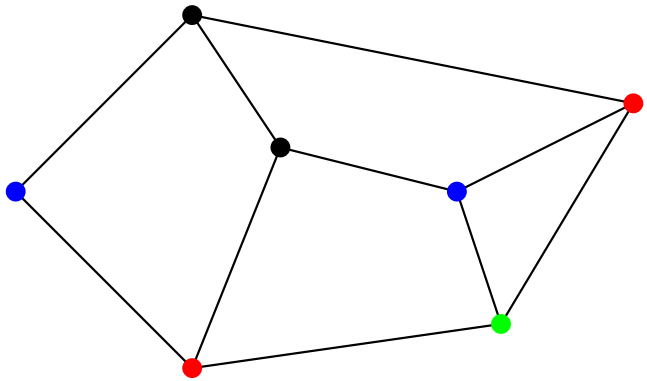
実行例 (別の順序)



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

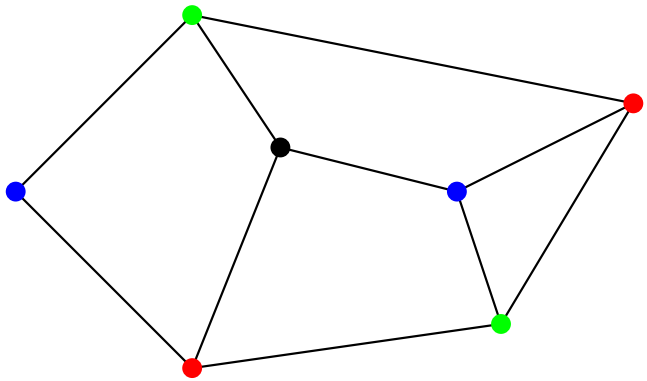
実行例 (別の順序)



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

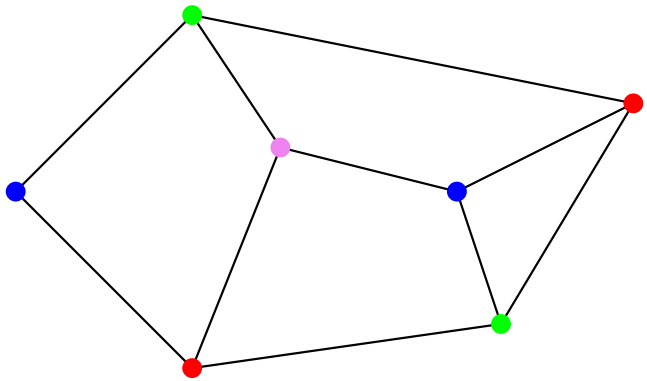
実行例 (別の順序)



とにかく解が欲しい場合：貪欲アルゴリズム 3

- 頂点を任意の順序で 1 つずつ塗る
- その頂点の隣接点の色と違い，既に使った色の中で最も早く使った色で塗る
- 既に使った色で塗れない場合は，新しい色で塗る

実行例 (別の順序)



貪欲アルゴリズム 3 の性能

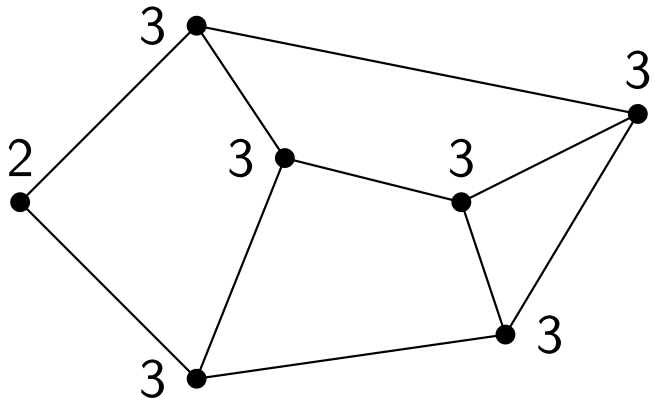
定理 1

任意の無向グラフ $G = (V, E)$ と V 上の任意の頂点順序に対して，

貪欲アルゴリズム 3 が費やす色数 $\mathrm{ALG} \leq \Delta(G) + 1$

定義：次数，最大次数

- **無向グラフ G における頂点 v の次数 $d(v)$ とは，v に接続する辺の数**
- **無向グラフ G の最大次数 $\Delta(G)$ とは，その頂点の次数の最大値**



$\Delta(G) = 3$

貪欲アルゴリズム 3 の性能：証明

使う色に $1, 2, \ldots, \Delta(G)+1$ **というラベルを付けておく (と都合がよい)**

- **任意の頂点** v **に色を塗るときのことを考える**
- v **の隣接頂点とは違う色を** v **に塗る**
- v **の隣接頂点に使われる色の数** $\leq d(v)$ **(次数)**
- $\therefore$ v **を塗るために必要な色は** $1, 2, \ldots, d(v)+1$ **の中にある**
- $\therefore$ **どの頂点も** $1, 2, \ldots, \Delta(G)+1$ **の中の色で塗れる**$(\because d(v) \leq \Delta(G))$

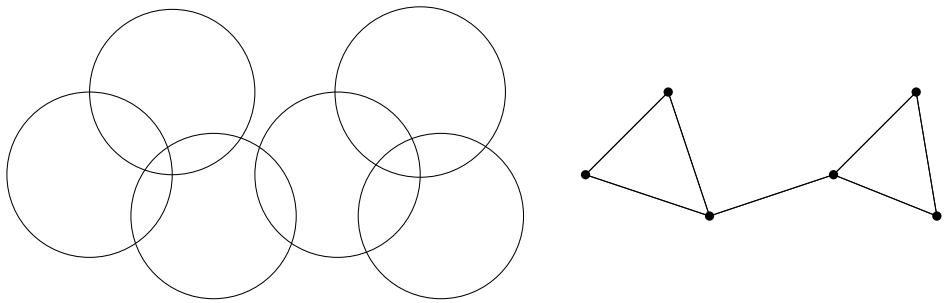
□

周波数割当と単位円グラフ

定義：単位円グラフ

単位円グラフとは次のようにして構成できる無向グラフ G

- G の各頂点は平面上の単位円に対応
- G の各辺は 2 つの交わる単位円に対応



事実 (Clark, Colbourn, Johnson '90)

単位円グラフに対する彩色問題は NP 困難

単位円グラフと貪欲アルゴリズム

定理 2 (Marathe et al. 1994)

任意の単位円グラフ $G = (V, E)$ **と任意の頂点順序に対して，**

$$\text{貪欲アルゴリズム } 3 \text{ が費やす色数 } \mathrm{ALG} \leq 6 \cdot \mathrm{OPT} - 5$$

ただし，OPT **は** G **の彩色における最小色数**

この定理はアルゴリズムの出力の相対誤差に対して理論的保証を与える

$$\left| \frac{\mathrm{ALG} - \mathrm{OPT}}{\mathrm{OPT}} \right| \leq 5 - \frac{5}{\mathrm{OPT}}$$

単位円グラフと貪欲アルゴリズム：証明 (1)

- $\mathrm{ALG} \leq \Delta(G) + 1$ ($\because$ **定理** 1)

今から示すこと

$$\mathrm{OPT} \geq \frac{\Delta(G)}{6} + 1$$

これが示せたと仮定すると...

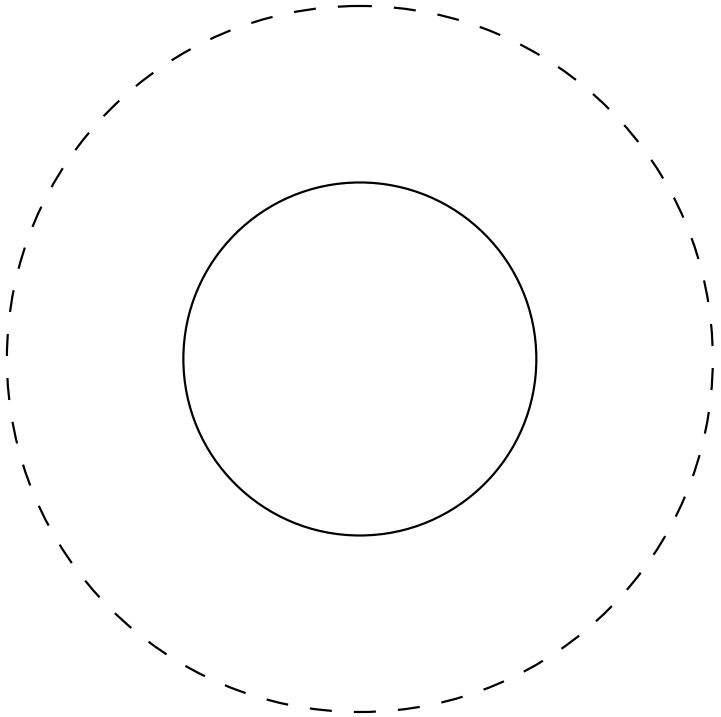
- $\Delta(G) \leq 6\mathrm{OPT} - 6$
- $\therefore \mathrm{ALG} \leq \Delta(G) + 1 \leq 6\mathrm{OPT} - 5$ □

単位円グラフと貪欲アルゴリズム：証明 (2)

今から示すこと (再掲)

$$\mathrm{OPT} \geq \frac{\Delta(G)}{6} + 1$$

- 任意の彩色を考える
- 最大次数の頂点 v を見る
  (v の隣接頂点数 $= \Delta(G)$)
- v の隣接頂点の中で，
  同じ色を持つものの数 ≤ 6
- $\because$ v に対応する円の周りを
  60 度ずつ区切ったとき，
  同じ扇に中心を持つ円は
  同じ色で塗れない

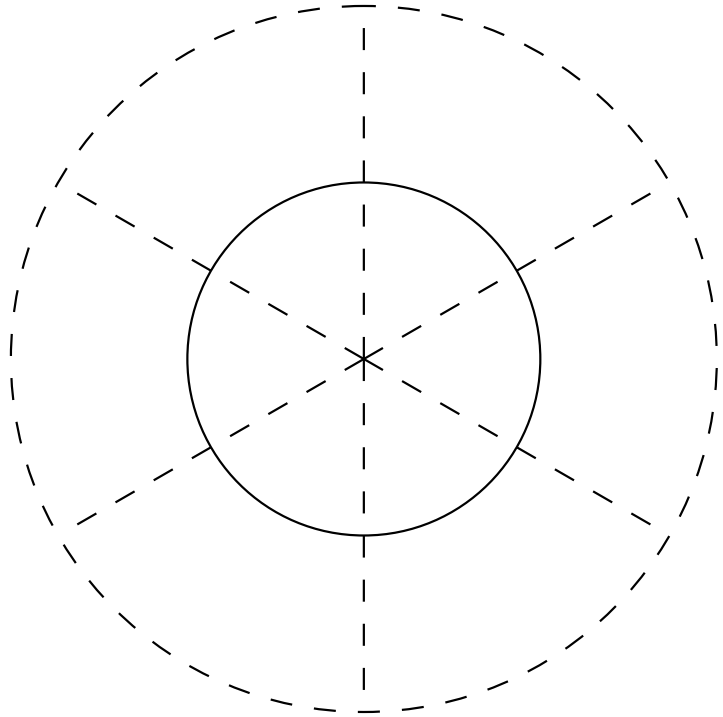


単位円グラフと貪欲アルゴリズム：証明 (2)

今から示すこと (再掲)

$$\mathrm{OPT} \geq \frac{\Delta(G)}{6} + 1$$

- 任意の彩色を考える
- 最大次数の頂点 v を見る
  (v の隣接頂点数 $= \Delta(G)$)
- v の隣接頂点の中で，
  同じ色を持つものの数 ≤ 6
- $\because$ v に対応する円の周りを
  60 度ずつ区切ったとき，
  同じ扇に中心を持つ円は
  同じ色で塗れない



単位円グラフと貪欲アルゴリズム：証明 (3)

今から示すこと (再掲)

$$\mathrm{OPT} \geq \frac{\Delta(G)}{6} + 1$$

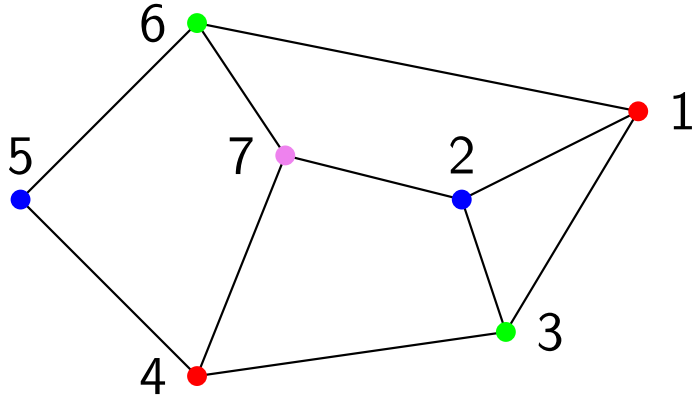
- $\therefore$ v **の隣接頂点全体に使われる色数** $\geq \Delta(G)/6$
- $\therefore$ v **も含めて使われる色数** $\geq \Delta(G)/6 + 1$
- $\therefore$ **この彩色の色数** $\geq \Delta(G)/6 + 1$
- $\therefore$ **最小色数** $\geq \Delta(G)/6 + 1$ ($\because$ **彩色の任意性**)

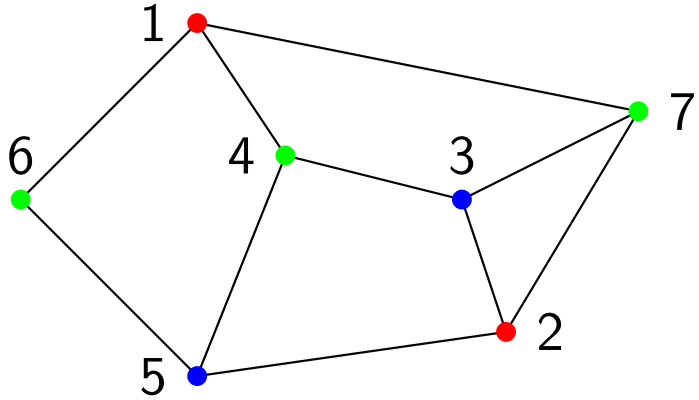
□

貪欲アルゴリズムは便利

貪欲アルゴリズムの柔軟さ

- 頂点順序は何でもよい
- 頂点順序によっては，色数の少ない彩色が得られるかもしれない



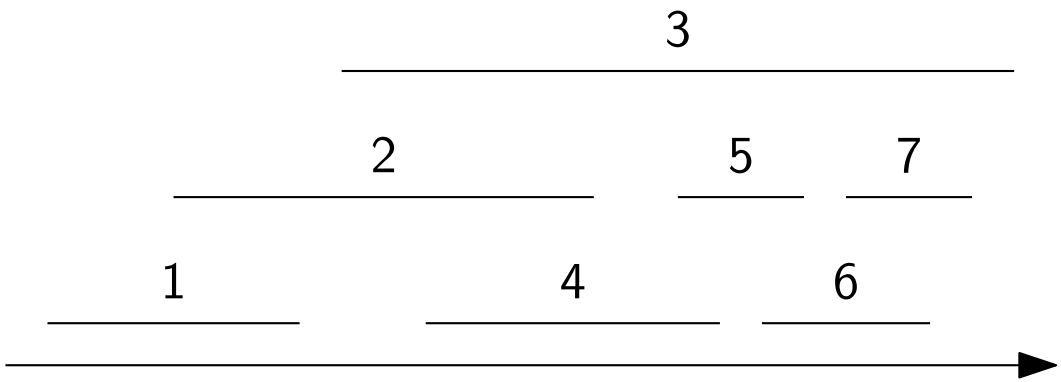


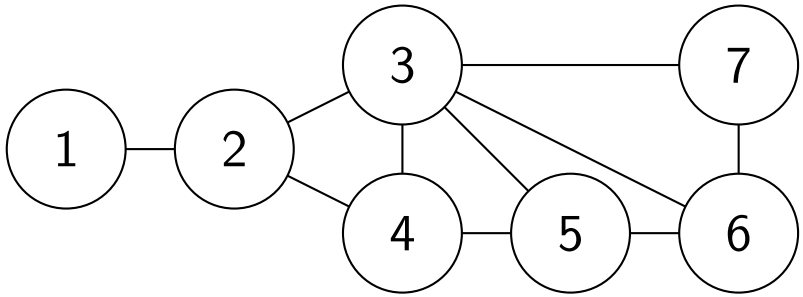
ジョブスケジューリングと区間グラフ

定義：区間グラフ

区間グラフとは次のようにして構成できる無向グラフ G

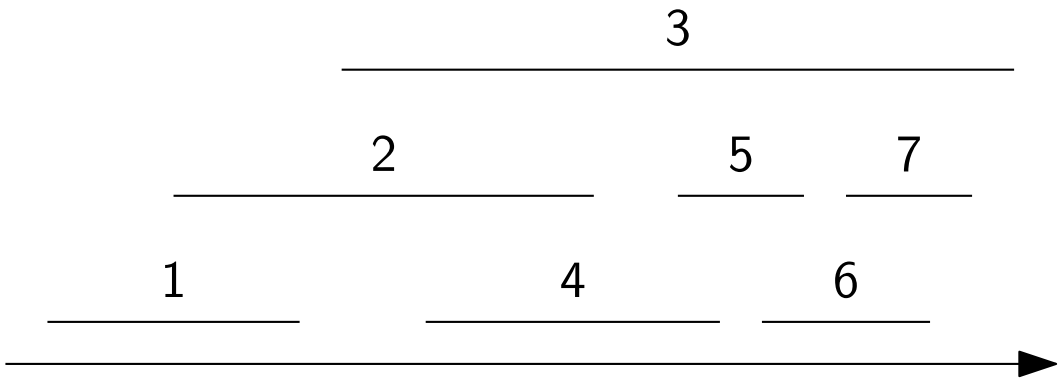
- G の各頂点は数直線上の閉区間に対応
- G の各辺は 2 つの交わる区間に対応

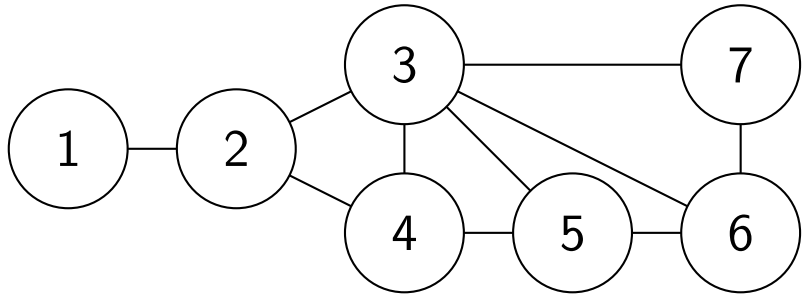




区間グラフと貪欲アルゴリズム：頂点順序

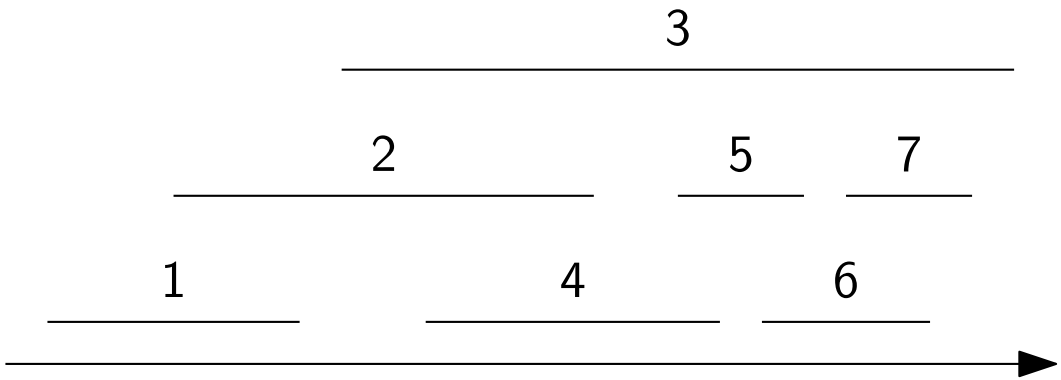
- 頂点に対応する区間の左端を見て，
  それを左から順にならべた順序を考える

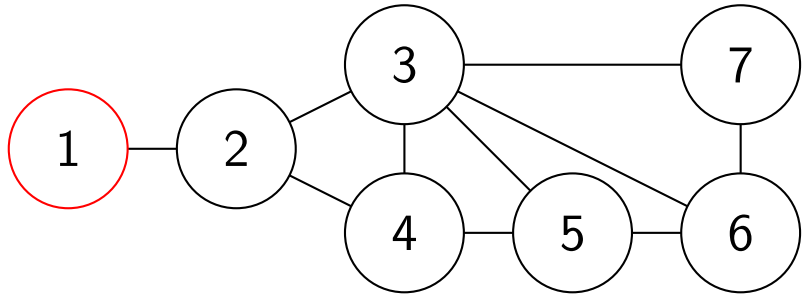




区間グラフと貪欲アルゴリズム：頂点順序

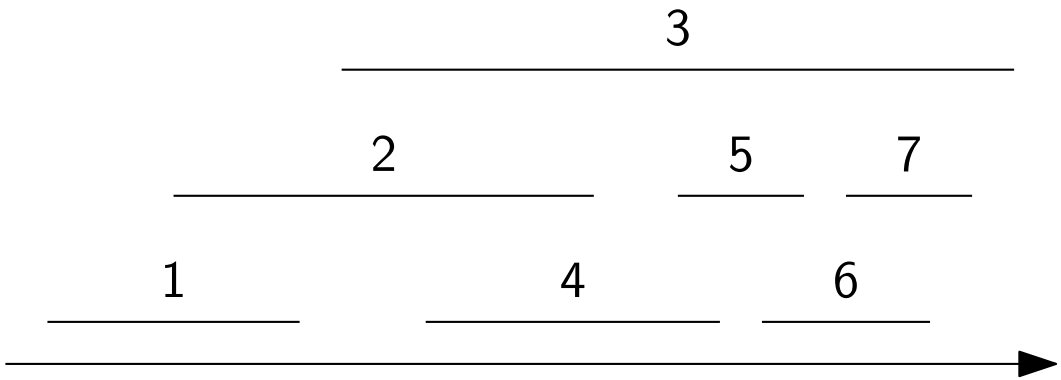
- 頂点に対応する区間の左端を見て，
  それを左から順にならべた順序を考える

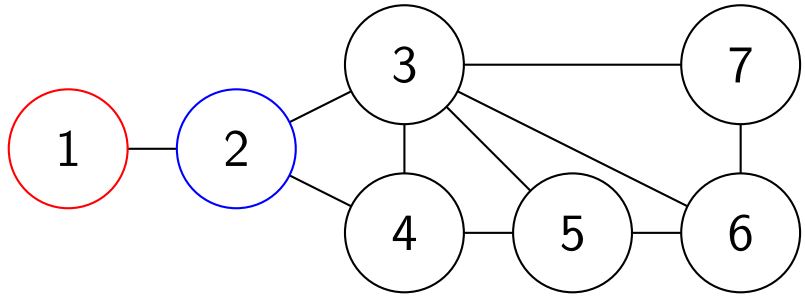




区間グラフと貪欲アルゴリズム：頂点順序

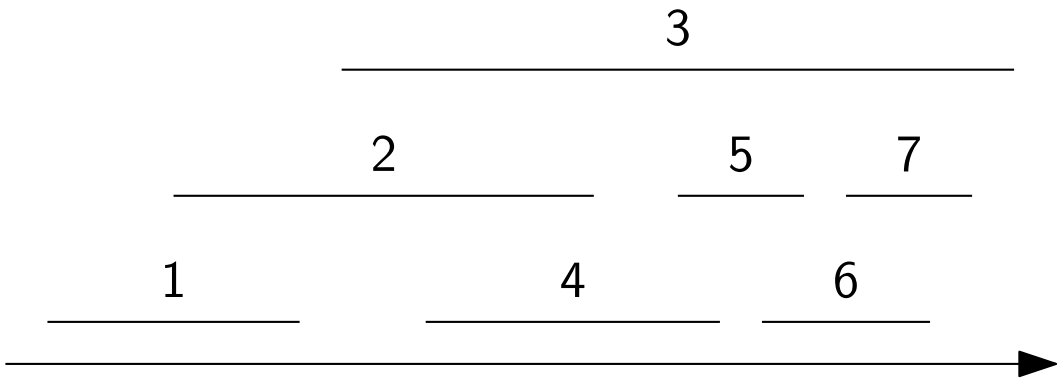
- 頂点に対応する区間の左端を見て，
  それを左から順にならべた順序を考える

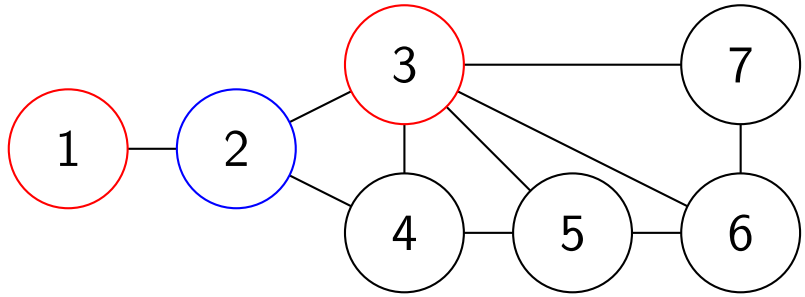




区間グラフと貪欲アルゴリズム：頂点順序

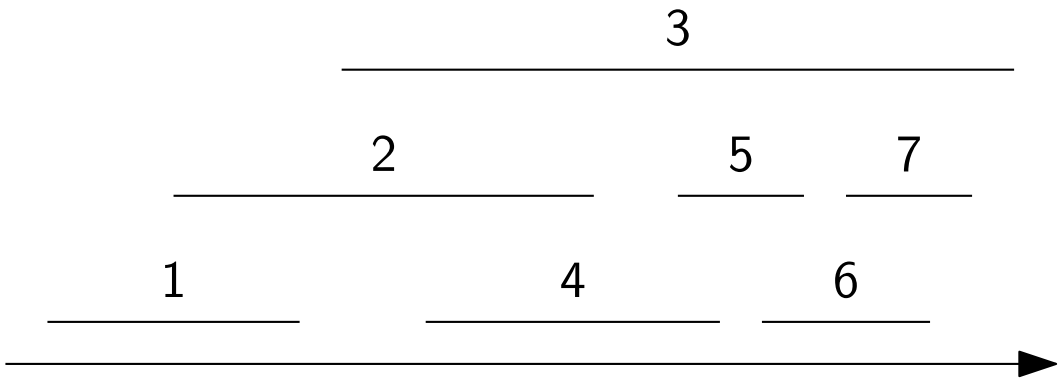
- 頂点に対応する区間の左端を見て，
  それを左から順にならべた順序を考える

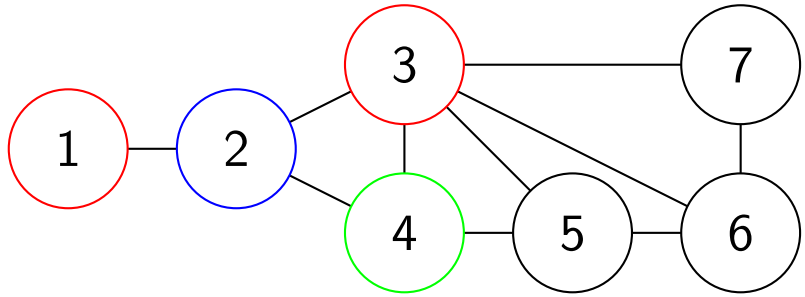




区間グラフと貪欲アルゴリズム：頂点順序

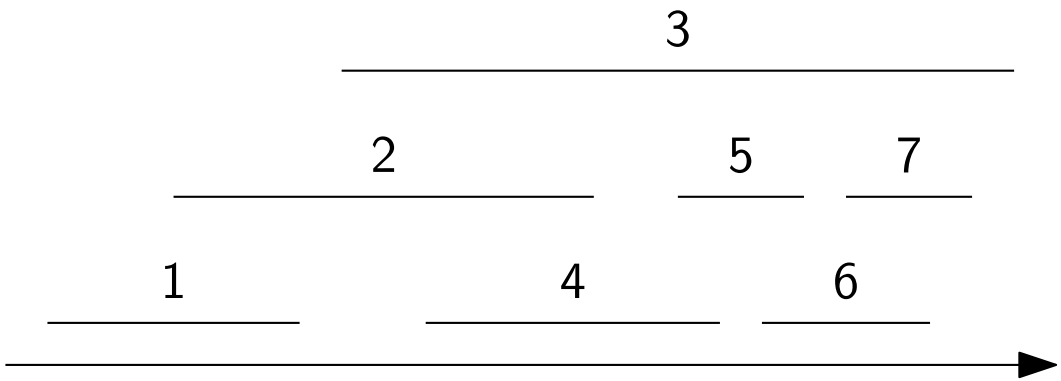
- 頂点に対応する区間の左端を見て，
  それを左から順にならべた順序を考える

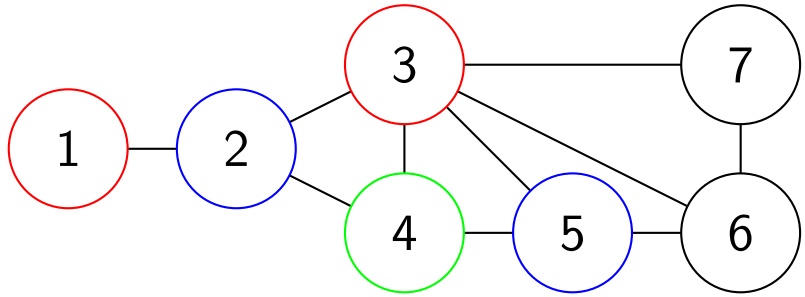




区間グラフと貪欲アルゴリズム：頂点順序

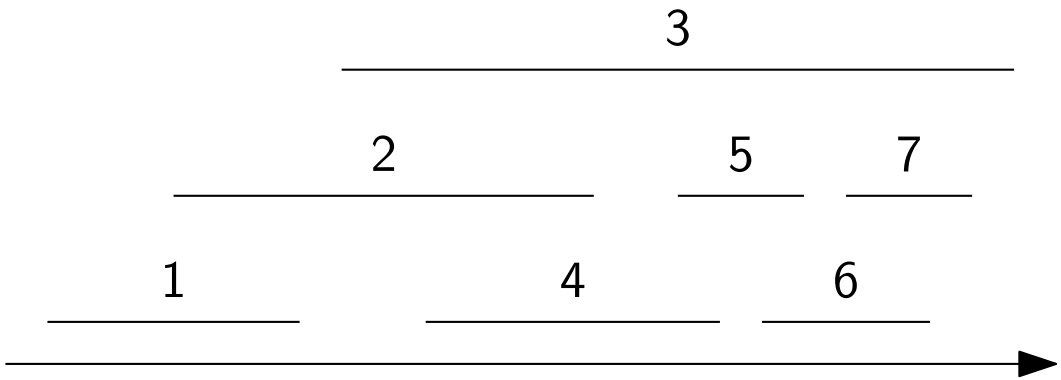
- 頂点に対応する区間の左端を見て，
  それを左から順にならべた順序を考える

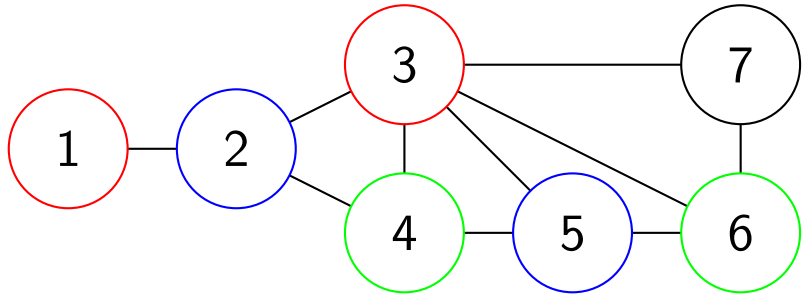




区間グラフと貪欲アルゴリズム：頂点順序

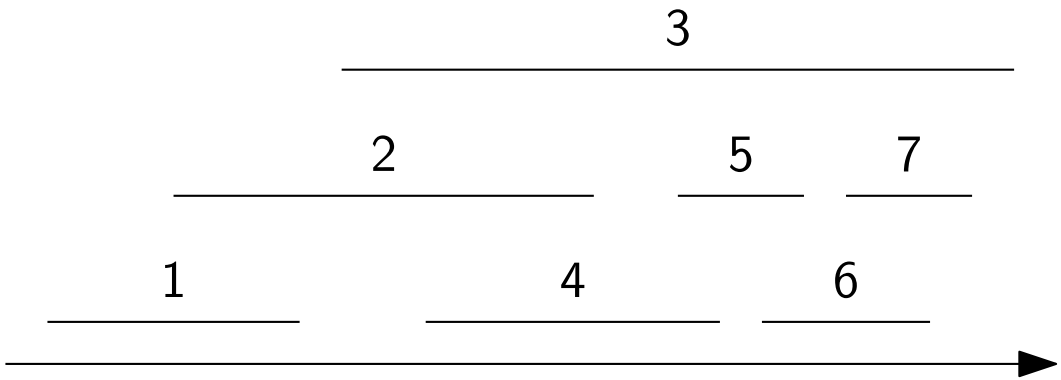
- 頂点に対応する区間の左端を見て，
  それを左から順にならべた順序を考える

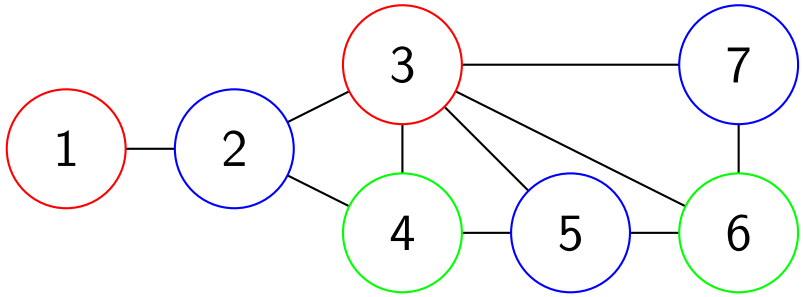




区間グラフと貪欲アルゴリズム：頂点順序

- 頂点に対応する区間の左端を見て，
  それを左から順にならべた順序を考える





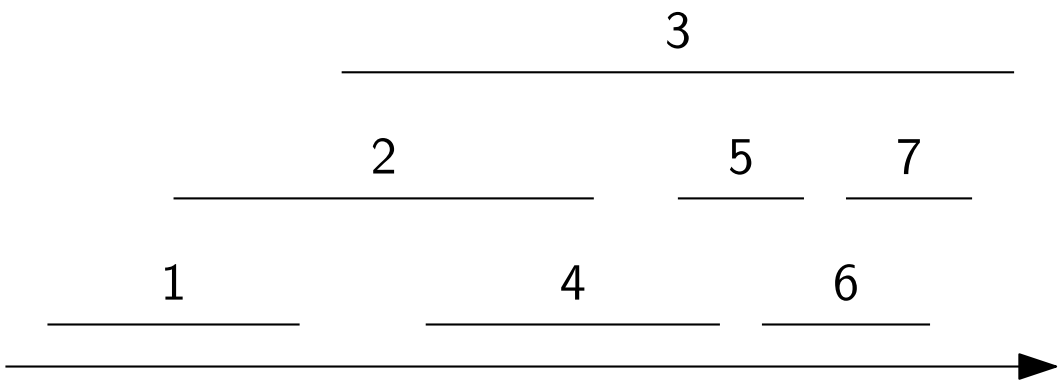
区間グラフと貪欲アルゴリズム：性能解析 (1)

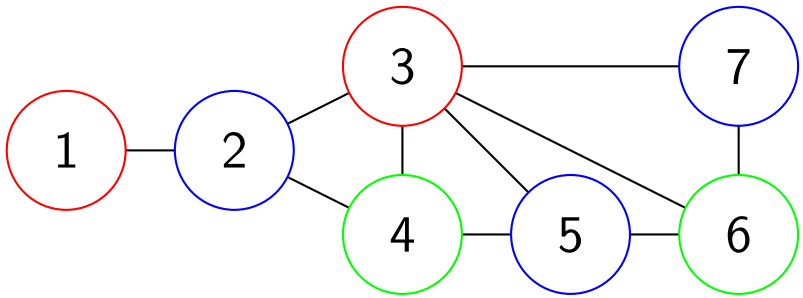
定理 3

任意の区間グラフに対して，前ページの規則で定めた順序で貪欲アルゴリズムを実行すると，色数最小の彩色が得られる

証明：使用した色が $1, 2, \ldots, k$ であるとする

- **観察：数直線上の 1 点 x を含む区間の数 $\leq$ 最小色数**
- **I を色 k で塗られた最初の頂点に対応する区間とする その左端を y とする**
- **y を含む区間の数 $\leq$ 最小色数**





区間グラフと貪欲アルゴリズム：性能解析 (2)

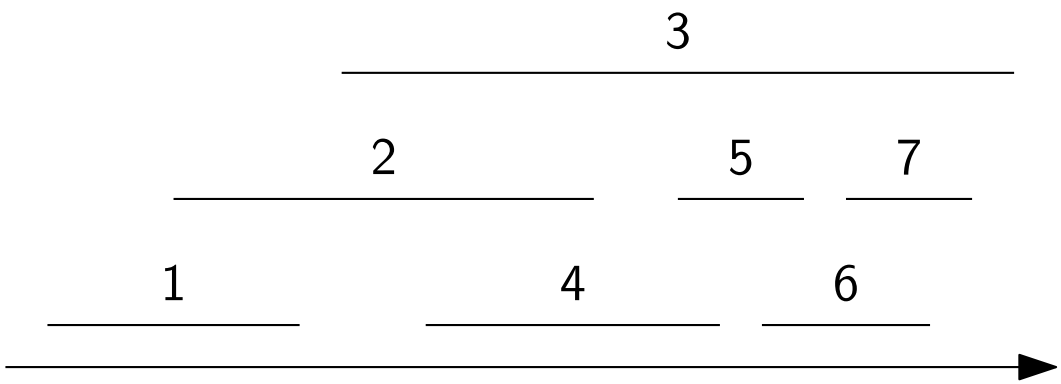
主張

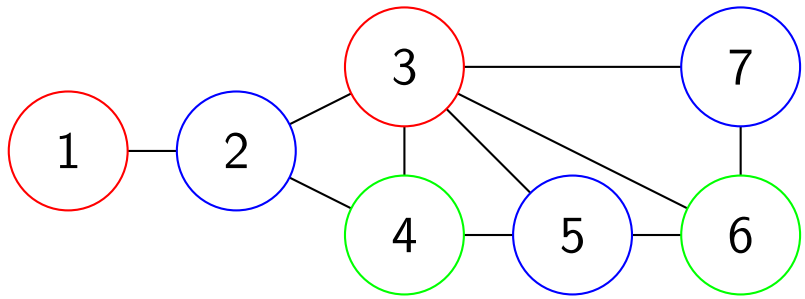
y を含む区間の数 $= k$

主張の証明

- I と交わり，I の左端よりも左端が左にある区間に対応する頂点は $1, 2, ..., k-1$ で塗られている
  ($\because$ I に対応する頂点が色 k で塗られた)
- それらは全部 y を含む
- $\therefore$ そのような区間の数 $= k - 1$
- $\therefore$ y を含む区間の数 $= k$

□





第 1 回のまとめ

アルゴリズムに要求されること と その緩和

- どんな入力に対しても (入力の普遍性)
- 高速に処理し (処理の高速性)
- 正しい結果を返す (出力の正当性)

この 3 つの要求を緩和できる場合がある

例としての彩色問題

- 厳密に解くアルゴリズム
  - 少しの工夫で計算量の改善
- とにかく解を得るアルゴリズム
  - 理論的解析が可能である

参考文献

- 現実問題に対する数理モデル化，最適化手法の適用
  - OR 事典 Wiki．
    http://www.orsj.or.jp/~wiki/wiki/index.php/メインページ
- 3 つの観点からの妥協について
  - 岡本吉央．組合せ最適化理論の三次元描像．第 21 回回路とシステム軽井沢ワークショップ論文集，2008，pp. 659–664.
    http://dopal.cs.uec.ac.jp/okamotoy/PDF/2008/comb3d.pdf
  - J. ホロムコヴィッチ．計算困難問題に対するアルゴリズム理論．和田幸一，増澤利光，元木光雄訳．丸善出版，2005 年．
  - 岩間一雄．アルゴリズム・サイエンス：出口からの超入門．共立出版．2006 年．

参考文献

- **参照文献**

  - B. N. Clark, C. J. Colbourn, and D. S. Johnson. Unit disk graphs. Discrete Mathematics 86 (1990) 165–177.
  - R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher (eds.), Complexity of Computer Computations. Plenum, New York, 1972, pp. 85–103.
  - M. V. Marathe, H. Breu, H. B. Hunt III, B. Harry, S. S. Ravi, D. J. Rosenkrantz. Geometry based heuristics for unit disk graphs. arXiv:math.CO/9409226, 1994.

目次