離散数理工学　第 2 回
数え上げの基礎：漸化式の立て方

岡本 吉央
okamotoy@uec.ac.jp

電気通信大学

2020 年 10 月 13 日

最終更新：2020 年 10 月 12 日　09:52

スケジュール 前半 (予定)

1	数え上げの基礎：二項係数	(10/6)
2	数え上げの基礎：漸化式の立て方	(10/13)
3	数え上げの基礎：漸化式の解き方 (基礎)	(10/20)
4	数え上げの基礎：漸化式の解き方 (発展)	(10/27)
⋆	休み (祝日)	(11/3)
5	離散代数：行列式とパーマネント	(11/10)
6	離散代数：非交差経路の数え上げ	(11/17)
⋆	休み (調布祭片付け)	(11/24)
7	離散代数：全域木の数え上げ	(12/1)

注意：予定の変更もありうる

スケジュール 後半 (予定)

9	離散確率論：確率的離散システムの解析 (基礎)	(12/8)
⋆	中間レポート出題	(12/15)
10	離散確率論：確率的離散システムの解析 (発展)	(12/22)
11	離散確率論：乱択データ構造とアルゴリズム (基礎)	(1/5)
12	離散確率論：乱択データ構造とアルゴリズム (発展)	(1/12)
13	離散確率論：マルコフ連鎖 (基礎)	(1/19)
14	離散確率論：マルコフ連鎖 (発展)	(1/26)
⋆	予備	(2/2)

注意：予定の変更もありうる

今日の目標

今日の目標
漸化式を立てられるようになる
- 組合せ構造の数え上げ
- アルゴリズムの計算量

格言
アルゴリズムの計算量解析の基礎は数え上げ

目次

1. 組合せ構造の数え上げ
   グラフにおける完全マッチングの数え上げ
2. アルゴリズムの計算量
3. 今日のまとめ

無向グラフ

定義：無向グラフとは？
無向グラフとは，順序対 (V, E) で，
- V は集合
- $E \subseteq 2^V$ は V の 要素数 2 の部分集合の集合

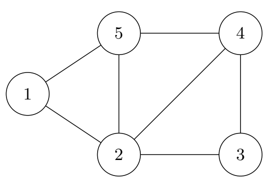
であるもののこと

例：
- $V = \{1, 2, 3, 4, 5\}$
- $E = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\}$

注意
$\{2, 5\} = \{5, 2\}$　(集合では順序を不問)

無向グラフの図示

- $V = \{1, 2, 3, 4, 5\}$
- $E = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{4, 5\}\}$

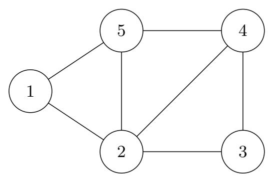


無向グラフの用語

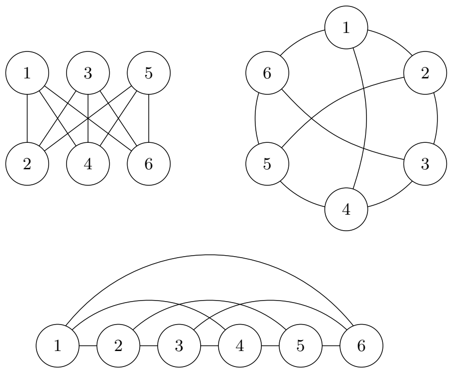
無向グラフ $G = (V, E)$

定義：無向グラフの用語
- V の要素を G の頂点と呼ぶ
- E の要素を G の辺と呼ぶ
- V を G の頂点集合と呼ぶ
- E を G の辺集合と呼ぶ
- 辺 $\{u, v\} \in E$ に対して，u, v をその端点と呼ぶ
- 頂点 v が辺 e の端点であるとき，v は e に接続するという
- 頂点 u と v が辺を成すとき，u と v は隣接するという

- $V = \{1, 2, 3, 4, 5\}$
- $E = \{\{1, 2\}, \{1, 5\}, \{2, 3\}, \{2, 4\},$
  $\{2, 5\}, \{3, 4\}, \{4, 5\}\}$
- 頂点 $2, 3$ は辺 $\{2, 3\}$ の端点
- 頂点 2 は辺 $\{2, 3\}$ に接続する
- 頂点 2 と頂点 3 は隣接する



1つのグラフに対するいろいろな図示



用語に関する注意

無向グラフ

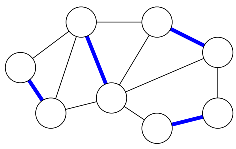
- 「頂点」の別名：「節点」,「ノード」,「点」
- 「辺」の別名：「枝」,「エッジ」

完全マッチング

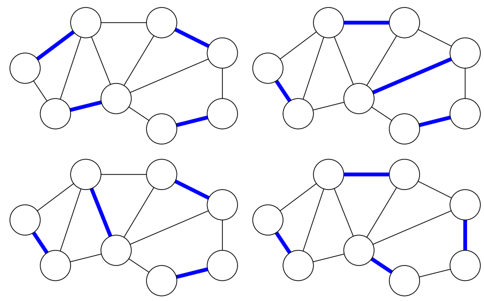
無向グラフ $G = (V, E)$

定義：完全マッチングとは？

G の完全マッチングとは，辺部分集合 $M \subseteq E$ で，
各頂点 $v \in V$ に対して，v に接続する M の辺がただ 1 つ存在するもの



すべての完全マッチング (完全マッチング全体)

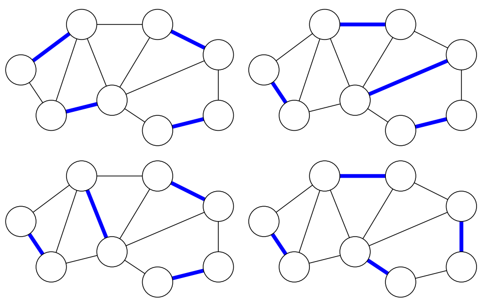


4 個

目標

やりたいこと

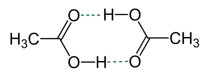
与えられた無向グラフにおける完全マッチングの総数を計算したい

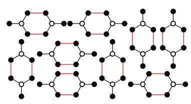


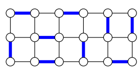
4 個

目標：なぜ計算したい？

統計力学における「二量体モデル」





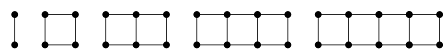


https://commons.wikimedia.org/wiki/File:Acetic_Acid_Hydrogenbridge_V.1.svg

- 系を無向グラフ $G = (V, E)$ としてモデル化する
- 系において許される状態の総数 ＝ 完全マッチングの総数
- $\rightsquigarrow$ 系の分配関数の計算 $\rightsquigarrow$ 系の振舞いのシミュレーション

物理学は，数え上げ組合せ論の重要な応用分野

例：2 段の格子



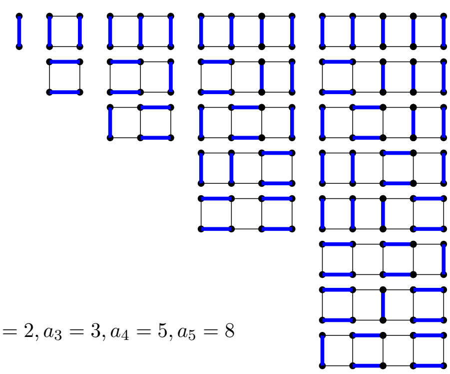
A_1　A_2　A_3　A_4　A_5

目標

グラフ A_n における完全マッチングの総数を計算する

例：2 段の格子 — 手でやってみる

$a_n = A_n$ の完全マッチングの総数

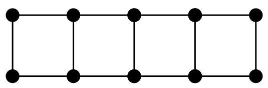


$a_1 = 1, a_2 = 2, a_3 = 3, a_4 = 5, a_5 = 8$

例：2 段の格子 — 系統立ててやってみる

グラフ A_5 を考えると，完全マッチングは次の 2 種類

- (A) 左上の頂点に接続する辺 が 縦の辺 のもの
  $\rightsquigarrow$ 他の辺は 右側に残る A_4 の完全マッチング
- (B) 左上の頂点に接続する辺 が 横の辺 のもの
  $\rightsquigarrow$ 他の辺は 右側に残る A_3 の完全マッチング

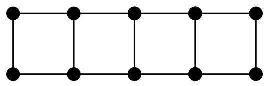


つまり，$a_5 = a_4 + a_3$

例：2 段の格子 — 系統立ててやってみる (一般化)

グラフ A_n を考えると，完全マッチングは次の 2 種類 $(n \geq 3)$

- (A) 左上の頂点に接続する辺 が 縦の辺 のもの
  $\rightsquigarrow$ 他の辺は 右側に残る A_{n-1} の完全マッチング
- (B) 左上の頂点に接続する辺 が 横の辺 のもの
  $\rightsquigarrow$ 他の辺は 右側に残る A_{n-2} の完全マッチング



つまり，$a_n = a_{n-1} + a_{n-2}$ $(n \geq 3)$

例：2 段の格子 — まとめ

$a_n =$ グラフ A_n における完全マッチングの総数　とするとき

漸化式

$$a_n = \begin{cases} 1 & (n = 1 \text{ のとき}) \\ 2 & (n = 2 \text{ のとき}) \\ a_{n-1} + a_{n-2} & (n \geq 3 \text{ のとき}) \end{cases}$$

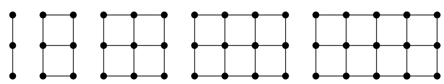
これでプログラムは書ける (解くのは次回)

例：2 段の格子 — プログラム実行結果

n	a_n	n	a_n
1	1	16	1597
2	2	17	2584
3	3	18	4181
4	5	19	6765
5	8	20	10946
6	13	21	17711
7	21	22	28657
8	34	23	46368
9	55	24	75025
10	89	25	121393
11	144	26	196418
12	233	27	317811
13	377	28	514229
14	610	29	832040
15	987	30	1346269

例：3 段の格子

次のグラフを考える (B_n と書くことにする)



B_1　B_2　B_3　B_4　B_5

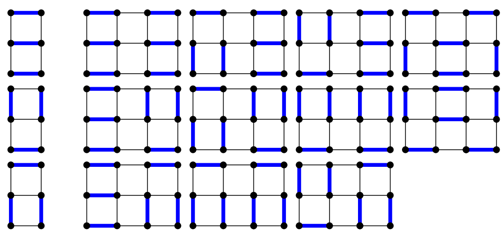
目標

グラフ B_n における完全マッチングの総数 b_n を計算する

注：n が奇数 $\Rightarrow b_n = 0$

例：3 段の格子 — 手でやってみる

$b_n = B_n$ の完全マッチングの総数

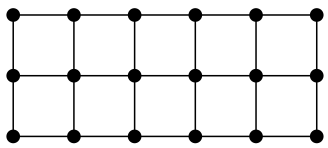


$b_1 = 0, b_2 = 3, b_3 = 0, b_4 = 11, b_5 = 0$

例：3 段の格子 — 系統立ててやってみる

グラフ B_n を考えると，完全マッチングは次の 2 種類 ($n \geq 4$，偶数)

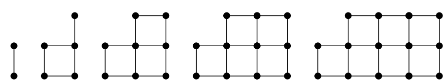
- (A) 左上の頂点に接続する辺 が 縦の辺 のもの
- (B) 左上の頂点に接続する辺 が 横の辺 のもの



問題点：小さくなったグラフが 3 段の格子ではない

例：欠けた 3 段の格子

次のグラフを考える (C_n と書くことにする)



C_1　C_2　C_3　C_4　C_5

目標

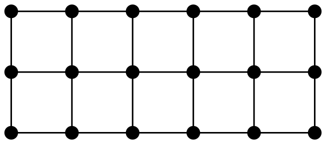
グラフ C_n における完全マッチングの総数 c_n を計算する

注：n が偶数 $\Rightarrow c_n = 0$

例：3 段の格子 — 系統立ててやってみる

グラフ B_n を考えると，完全マッチングは次の 2 種類 ($n \geq 4$，偶数)

- (A) 左上の頂点に接続する辺 が 縦の辺 のもの
  ⇝ 他の辺は 右側に残る C_{n-1} の完全マッチング
- (B) 左上の頂点に接続する辺 が 横の辺 のもの
  ⇝ 他の辺は 右側に残る C_{n-1} の完全マッチング か
  右側に残る B_{n-2} の完全マッチング



つまり，$b_n = b_{n-2} + 2c_{n-1}$ ($n \geq 4$，偶数)

例：3 段の格子 — まとめ

- b_n = グラフ B_n における完全マッチングの総数
- c_n = グラフ C_n における完全マッチングの総数

とするとき

漸化式

$$b_n = \begin{cases} 0 & (n \text{ が奇数のとき}) \\ 3 & (n = 2 \text{ のとき}) \\ b_{n-2} + 2c_{n-1} & (n \geq 4, \text{ 偶数のとき}) \end{cases}$$

$$c_n = \begin{cases} 0 & (n \text{ が偶数のとき}) \\ 1 & (n = 1 \text{ のとき}) \\ b_{n-1} + c_{n-2} & (n \geq 3, \text{ 奇数のとき}) \end{cases}$$

(演習問題)

これでプログラムは書ける (解くのは次回)

例：3 段の格子 — プログラム実行結果

n	$b(n)$	$c(n)$
1	0	1
2	3	0
3	0	4
4	11	0
5	0	15
6	41	0
7	0	56
8	153	0
9	0	209
10	571	0
11	0	780
12	2131	0
13	0	2911
14	7953	0
15	0	10864

n	$b(n)$	$c(n)$
16	29681	0
17	0	40545
18	110771	0
19	0	151316
20	413403	0
21	0	564719
22	1542841	0
23	0	2107560
24	5757961	0
25	0	7865521
26	21489003	0
27	0	29354524
28	80198051	0
29	0	109552575
30	299303201	0

目次

単純な再帰アルゴリズム

アルゴリズム A

```
1: def fnct(n)
2:    print "a"
3:    if n > 2
4:      fnct(n-1)
5:      fnct(n-2)
6:    end
7: end
```

質問

fnct(n) を実行したとき,「a」は何個出力されるか？

単純な再帰アルゴリズム：例

n	a の数	n	a の数	n	a の数	n	a の数
1	1	11	177	21	21891	31	2692537
2	1	12	287	22	35421	32	4356617
3	3	13	465	23	57313	33	7049155
4	5	14	753	24	92735	34	11405773
5	9	15	1219	25	150049	35	18454929
6	15	16	1973	26	242785	36	29860703
7	25	17	3193	27	392835	37	48315633
8	41	18	5167	28	635621	38	78176337
9	67	19	8361	29	1028457	39	126491971
10	109	20	13529	30	1664079	40	204668309

単純な再帰アルゴリズム

アルゴリズム A

```
1: def fnct(n)
2:    print "a"
3:    if n > 2
4:      fnct(n-1)
5:      fnct(n-2)
6:    end
7: end
```

漸化式に向けて

f_n = fnct(n) を実行したときに出力される a の数

単純な再帰アルゴリズム

アルゴリズム A

```
1: def fnct(n)
2:    print "a"
3:    if n > 2
4:      fnct(n-1)
5:      fnct(n-2)
6:    end
7: end
```

漸化式に向けて

- 2 行目：n が何であろうと必ず 1 つは a が出力される
- 4 行目と 5 行目：再帰呼び出し

単純な再帰アルゴリズム

アルゴリズム A

```
def fnct(n)
  print "a"
  if n > 2
    fnct(n-1)
    fnct(n-2)
  end
end
```

漸化式

$$f_n = \begin{cases} 1 & (n \le 2 \text{ のとき}) \\ 1 + f_{n-1} + f_{n-2} & (n \ge 3 \text{ のとき}) \end{cases}$$

ユークリッドのアルゴリズム — 最大公約数の計算

ユークリッドのアルゴリズム (正当性は演習問題)

```
def gcd(a, b) # precondition: a >= b
  print "G"
  if b == 0
    return a
  else
    gcd(b, a % b)
  end
end
```

a % b = a を b で割った余り (数学では $a \bmod b$ と書く)

質問

gcd(a, b) を実行したとき，「G」は何個出力されるか？

厳密に求めるのは難しいので，上界を求めたい

(最悪の場合における保証)

ユークリッドのアルゴリズム：ちょっと観察 (1)

a	b	G の数
14	11	5
143	11	2
1432	11	4
14325	11	5
143259	11	5
1432591	11	5
14325910	11	4
143259106	11	3
1432591067	11	5
14325910676	11	2
143259106765	11	4
1432591067659	11	5
14325910676592	11	5
143259106765923	11	4

ユークリッドのアルゴリズム：ちょっと観察 (2)

a	b	G の数
14	13	3
143	13	2
1432	13	4
14325	13	4
143259	13	4
1432591	13	4
14325910	13	3
143259106	13	4
1432591067	13	5
14325910676	13	4
143259106765	13	7
1432591067659	13	5
14325910676592	13	7
143259106765923	13	6

ユークリッドのアルゴリズム：解析に向けて

ユークリッドのアルゴリズム

```
def gcd(a, b) # precondition: a >= b
  print "G"
  if b == 0
    return a
  else
    gcd(b, a % b)
  end
end
```

考える量

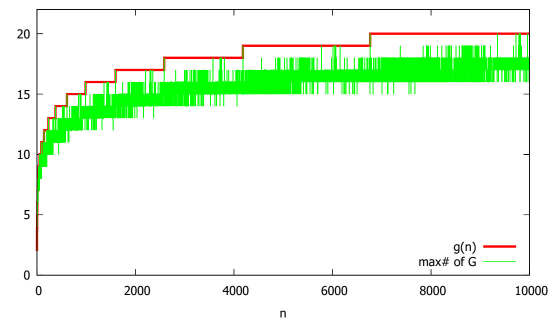
$$g_n = \max_{a \ge 1, b \le n} \{\text{gcd}(a,\ b) \text{ の実行で出力される G の数}\}$$

直感：$g_n =$ 「$b \le n$ に限った場合の最悪時計算量」

欲しいもの

g_n の上界

ユークリッドのアルゴリズム：解析に向けて (図)



$$g_n = \max_{a \ge 1, b \le n} \{\text{gcd}(a,\ b) \text{ の実行で出力される G の数}\}$$

$$\text{細い線} = \max_{a \ge 1} \{\text{gcd}(a,\ n) \text{ の実行で出力される G の数}\}$$

ユークリッドのアルゴリズム：計算量解析 — 補題 A

考える量

$$g_n = \max_{a \ge 1, b \le n} \{\text{gcd}(a,\ b) \text{ の実行で出力される G の数}\}$$

補題 A

任意の自然数 $n \ge 1$ に対して，$g_n \le g_{n+1}$

証明：「$g_n = \text{gcd}(a, b)$ の実行で出力される G の数」となる
$a \ge 1$ と $b \le n$ を考えると，

$$\begin{aligned} g_n &= \text{gcd}(a,b) \text{ の実行で出力される G の数} \\ &\le \max_{a' \ge 1, b' \le n+1} \{\text{gcd}(a',\ b') \text{ の実行で出力される G の数}\} \\ &= g_{n+1} \end{aligned}$$

したがって，$g_n \le g_{n+1}$ □

ユークリッドのアルゴリズム：計算量解析 — 補題 B

補題 B

自然数 $a, b \ge 1$ に対して，$a \ge b$ のとき，

$$a \bmod b \le \left\lfloor \frac{a}{2} \right\rfloor$$

証明：$a = bq + r$ とする (ただし，$0 \le r < b$)

- このとき，$a \bmod b = r$
- $a \ge b$ より，$q \ge 1$
- $b \le \left\lfloor \frac{a}{2} \right\rfloor$ のとき，$r < b \le \left\lfloor \frac{a}{2} \right\rfloor$
- $b > \left\lfloor \frac{a}{2} \right\rfloor$ のとき，$r = a - bq \le a - b < a - \left\lfloor \frac{a}{2} \right\rfloor = \left\lceil \frac{a}{2} \right\rceil$
- したがって，このとき，$r \le \left\lfloor \frac{a}{2} \right\rfloor$ □

注 (演習問題)：任意の自然数 $n \ge 0$ に対して，$n - \left\lfloor \frac{n}{2} \right\rfloor = \left\lceil \frac{n}{2} \right\rceil$

ユークリッドのアルゴリズム：計算量解析に向けて

ユークリッドのアルゴリズム

```
def gcd(a, b) # precondition: a >= b
  print "G"
  if b == 0
    return a
  else
    gcd(b, a % b)
  end
end
```

$$g_n = \texttt{gcd}(a,b) \text{ の実行で出力される G の数}$$

となる a, b を考えると...

ユークリッドのアルゴリズム：計算量解析 (1)

$$\begin{aligned} g_n &= \texttt{gcd}(a,b) \text{ の実行で出力される G の数} \\ &= 1 + \texttt{gcd}(b, a \bmod b) \text{ の実行で出力される G の数} \end{aligned}$$

ここで，場合分け

- $a \bmod b = 0$ のとき，$g_n = 2$
  ($\because$ gcd$(b, a \bmod b)$ はもう再帰呼び出しをしない)
- $a \bmod b \neq 0$ のとき，次のページ

ユークリッドのアルゴリズム：計算量解析 (2)

$$\begin{aligned} g_n &= \texttt{gcd}(a,b) \text{ の実行で出力される G の数} \\ &= 1 + \texttt{gcd}(b, a \bmod b) \text{ の実行で出力される G の数} \\ &= 2 + \texttt{gcd}(a \bmod b, b \bmod (a \bmod b)) \text{ の実行で出力される G の数} \\ &\leq 2 + \max_{a' \geq 1, b' \leq \lfloor b/2 \rfloor} \{\texttt{gcd}(a', b') \text{ の実行で出力される G の数}\} \\ &= 2 + g_{\lfloor b/2 \rfloor} \leq 2 + g_{\lfloor n/2 \rfloor} \end{aligned}$$

注意

補題 B より，$b \bmod (a \bmod b) \leq \left\lfloor \frac{b}{2} \right\rfloor$

つまり，$n \geq 1$ のとき，どちらの場合でも $g_n \leq 2 + g_{\lfloor n/2 \rfloor}$

ユークリッドのアルゴリズム：計算量解析 (結論)

得られた漸化式 (不等式であることに注意)

$$g_n \begin{cases} = 1 & n = 0 \text{ のとき} \\ \leq 2 + g_{\lfloor n/2 \rfloor} & n \geq 1 \text{ のとき} \end{cases}$$

ここからどう進めるかは次回

未解決問題：コラッツ予想

次のアルゴリズムを考える

```
def collatz(n)
  print n
  if n % 2 == 0
    collatz(n/2)
  else
    collatz(3*n+1)
  end
end
```

これは止まらないが…

コラッツ予想 (未解決)

任意の n に対して，collatz(n) は必ずいつか「1」を出力する

$n \leq 2^{68}$ のときは正しいと分かっている (Barina '20)

http://www.ericr.nl/wondrous/

目次

今日の目標

今日の目標

漸化式を立てられるようになる

- 組合せ構造の数え上げ
- アルゴリズムの計算量

格言

アルゴリズムの計算量解析の基礎は数え上げ