

# プログラム設計とアルゴリズム

## 第1回 (9/27)

早稲田大学高等研究所 講師  
福永津嵩

# 自己紹介

## 本科目の説明

# 自己紹介

- Twitter ID: @fukunagatsu
- HP: <https://sites.google.com/site/fukunagatsu/>
- 職位: 早稲田大学 高等研究所(講師)
- 研究テーマ: バイオインフォマティクス (BI)  
アルゴリズムや機械学習を活用したゲノム配列解析手法の開発研究
- 電気・生命情報工学科では浜田道昭教授がバイオインフォマティクスを研究されています。  
(私も昔浜田研で研究員をしていました。)



# (BIとは何か?) ヒトゲノム解読



(平成16年度科学技術白書より)

- ヒトゲノム = ヒトの全遺伝情報、設計図
- 2003年にヒトゲノムが**解読**される

# ゲノム配列情報解析

ATGCTGTTTATCAAATTCATTCATCTCGAAGTATTCAGTATGCCAGAAAGGGCTATTGCAAGACACGAAGTCCGAGAAATTGAGCAGCGACAT  
ACAATGGATGGCCCTCGGCAAGATGCCACTTTAGATGAGGAAGAGGACATGGTGATCATTTATAACAGAGTTCCCAAAACGGCAAGCACTTCA  
TTTACCAATATCGCCTATGACCTGTGTGCAAAGAATAAATAACCATGTCCTTCATATCAACACTACCAAAAATAATCCAGTGATGTCATTGCAA  
GATCAGGTGCGCTTTGTAAAGAATATAACTTCCTGGAAAGAGATGAAACCAGGATTTTATCATGGACACGTTTCTTACTTGGATTTTGC AAAA  
TTTGGTGTGAAGAAGAAACCAATTTACATTAATGTCATAAGGGATCCTATTGAGAGGCTAGTTTCTTATTATTACTTTCTGAGATTTTGGAGAT  
GATTATAGACCAGGGTTACGGAGACGAAAACAAGGAGACAAAAAGACCTTTGATGAATGTGTAGCAGAAGGTGGCTCAGACTGTGCTCCAGAG  
AAGCTCTGGCTTCAAATCCCGTTCTTCTGTGGCCATAGCTCCGAATGCTGGAATGTGGGAAGCAGGTGGGCTATGGATCAAGCCAAGTATAAC  
CTAATTAATGAATATTTTCTGGTGGGAGTTACTGAAGA ACTTGAAGATTTTATCATGTTATTGGAGGCAGCATTGCCCCGGTTTTTTCAGGGGT  
GCTACTGAACTCTATCGCACAGCACCAGATACTGCTACACCATCTCATCGGCATGGACCTATATGTGGCAGCAAGTCCATCTCATCGCTTTTG  
GTGAAAGTCAGTCCAGTTTGTAAAGATTCTCATTGTCACTGTAGACGAGGAACTGGGACACCAAAAAGGAGAAACTCTGGCCACGCTTGCACCC  
TGTTCCCAATCCTGGTCCAGTGTCACCCACAGATGGTAAGGAGCTCTAGAGACCTCACCAGCCCCTGGGATTGGTCACCTCACTCTTCTATGG  
ACAGAGATTCCTGCTGGGATCCTTTGAGGGCAAGCAGACCCTTCTTCCAGCTCGGACTGTGAACTCCACTGCAGCCGTAAGGACTGTCTGTGA  
CAGTGAGCCCGAGATGACTGGGCTCTGTGCTCCCTCCCGGCCCTCCAATCCTTGGCCTGCCACAGAGAACTGAGCTCTTTTATTAGCACCATG  
AATGTGACTGATACAGCTAGCCATTCCCTTGTGCGAATGACTCAGTTTATTAATGCTCTGCTAAAGATGGCTTCTTTGCTTGCCAGCAGCCTT  
AAACAGTATTTCAATTA AAAACTGGCTTAATTATTTTGAGAAGACGGCCCAATTAAAAGCTATACACTCCCTCTATGTGAGTGTTTATACATAGA  
GCTGTATATATAATACATATTTGTAAGTGTGTATATATATATGTGTGTATGTATGTGTCTATAAATATATAGGCTTAGCAATTTTCATTACATG  
GGATAAATTGTTGGAAAAAATACCCAGGAGCTGGTCCCCTTTCTGTTGCTAGATT CAGAGTAGAGGCCACCCCTCCACTCTGGGAGAGGCTGG  
TGTTGGTGATCTCTCAATGACTCTGCAATGGAAGTCCCAACTGCACAGAGCCCTGCCCCAGTTTTCAGGAGCCAGCAGCCTCGGAGAGGCGGAT  
CCTGACCTCTGCTCTGCTCTTGGGATAGCCTTTCCCTTCCCAGCAGGGTTGAGATACTTGGGCCGGGAAATGTTGTGGCAAAGTGTTTGCCAA  
AGCTCAGGAGAGACACAGACTTGGGGCTTTTGT TTTCTTGAGCTGGCTGTCTAGCTTTCCTAATGAGCAAATATGTTCTCTTTAAGGAAACAAA  
CAAACAAAGCAAAAACACCAATTCATCTGGATTTTATTCA TTTGTTTTAAATACAAACAAACAAAAGGAGAGTGGTTATTTCTGCACCAACTA  
TTTCAAATGCAAGTTACTCCATCGCTCGGGGTGGTTGGATGGTGCTTGT CACCATAGGACCCACAGGGCTAGTTCCA ACTGTTATTCGGTAAG  
GCTTTTTTCTTTCCAAAATTCCCAGTGTTCTTTAAGGCCCATTTAGCTGCGGGTTTTGTTTATTCTCCCGGCAATCAGCATTTAAAATAAGA

- ゲノムとは遺伝情報を保持するDNA配列の総体で、ATCGの4文字からなる文字列と見ることが出来る。ヒトゲノムはおよそ30億文字である。
- この文字列情報に生物学的な意味を注釈づけたい。

# ゲノム配列情報解析

ATGCTGTTTATCAAATTCATTCATCTCGAAGTATTCAGTATGCCAGAAAGGGCTATTGCAAGACACGAAGTCCGAGAAATTGAGCAGCGACAT  
ACAATGGATGGCCCTCGGCAAGATGCCACTTTAGATGAGGAAGAGGACATGGTGATCATTTATAACAGAGTTCCCAAACGGCAAGCACTTCA  
TTTACCAATATCGCCTATGACCTGTGTGCAAAGAATAAATAACCATGTCCTTCATATCAACACTACCAAAAATAATCCAGTGATGTCATTGCAA  
GATCAGGTGCGCTTTGTAAAGAATATAACTTCCTGGAAAGAGATGAAACCAGGATTTTATCATGGACACGTTTCTTACTTGGATTTTGC AAAA  
TTTGGTGTGAAGAAGAAACCAATTTACATTAATGTCATAAGGGATCCTATTGAGAGGCTAGTTTCTTATTATTACTTTCTGAGATTTGGAGAT  
GATTATAGACCAGGGTTACGGAGACGAAAACAAGGAGACAAAAAGACCTTTGATGAATGTGTAGCAGAAGGTGGCTCAGACTGTGCTCCAGAG  
AAGCTCTGGCTTCAAATCCCGTTCTTCTGTGGCCATAGCTCCGAATGCTGGAATGTGGGAAGCAGGTGGGCTATGGATCAAGCCAAGTATAAC  
CTAATTAATGAATATTTTCTGGTGGGAGTTACTGAAGA ACTTGAAGATTTTATCATGTTATTGGAGGCAGCATTGCCCCGGTTTTTTCAGGGGT  
GCTACTGAACTCTATCGCACAGCACCAGATACTGCTACACCATCTCATCGGCATGGACCTATATGTGGCAGCAAGTCCATCTCATCGCTTTTG  
GTGAAAGTCAGTCCAGTTTGTAAAGATTCTCATTGTCACTGTAGACGAGGA ACTGGGACACCAAAAAGGAGAACTCTGGCCACGCTTGCACCC  
TGTTCCCAATCCTGGTCCAGTGTCACCCACAGATGGTAAGGAGCTCTAGAGACCTCACCAGCCCCTGGGATTGGTCACCTCACTCTTCTATGG  
ACAGAGATTCTGCTGGGATCCTTTGAGGGCAAGCAGACCCTTCTTCCAGCTCGGACTGTGAACTCCACTGCAGCCGTAAGGACTGTCTGTGA  
CAGTGAGCCCGAGATGACTGGGCTCTGTGCTCCCTCCCGGCCCTCCAATCCTTGGCCTGCCACAGAGAACTGAGCTCTTTTATTAGCACCATG  
AATGTGACTGATACAGCTAGCCATTCCCTTGTGCGAATGACTCAGTTTATTAATGCTCTGCTAAAGATGGCTTCTTTGCTTGCCAGCAGCCTT  
AAACAGTATTTCAATAAA ACTGGCTTAATTATTTTGAGAAGACGGCCCAATTAAAAGCTATACACTCCCTCTATGTGAGTGTTTATACATAGA  
GCTGTATATATAATACATATTTGTAAGTGTGTATATATATATGTGTGTATGTATGTGTCTATAAATATATAGGCTTAGCAATTTATTACATG  
GGATAAATTGTTGGAAAAAATACCAGGAGCTGGTCCCCTTTCTGTTGCTAGATTTCAGAGTAGAGGCCACCCCTCCACTCTGGGAGAGGCTGG  
TGTTGGTGATCTCTCAATGACTCTGCAATGGAAGTCCCAACTGCACAGAGCCCTGCCCCAGTTTCAGGAGCCAGCAGCCTCGGAGAGGCGGAT  
CCTGACCTCTGCTCTGCTCTTGGGATAGCCTTTCCCTTCCCAGCAGGGTTGAGATACTTGGGCCGGGAAATGTTGTGGCAAAGTGTTTGCCAA  
AGCTCAGGAGAGACACAGACTTGGGGCTTTTGTCTTCTTGAGCTGGCTGTCTAGCTTICCTAATGAGCAAATATGTTCTCTTTAAGGAAACAAA  
CAAACAAAGCAAAAACACCAATTCATCTGGATTTTATTCATTTGTTTTAAATACAAACAAACAAAAGGAGAGTGGTTATTTCTGCACCAACTA  
TTTCAAATGCAAGTTACTCCATCGCTCGGGGTGGTTGGATGGTGCTTGTACCATAGGACCCACAGGGCTAGTTCCA ACTGTTATTCGGTAAG  
GCTTTTTTCTTTCCAAAATTCCCAGTGTTCTTTAAGGCCCATTTAGCTGCGGGTTTTGTTTATTCTCCCGGCAATCAGCATTTAAAATAAGA

- 遺伝子領域
- transfer RNA領域
- 疾患に関与するゲノム変異

# ゲノム配列情報解析

- 30億文字のゲノム情報を人手で注釈づけることは不可能なので、  
計算機を用いてデータ解析をする必要がある。
- 必要な技術
  1. アルゴリズム (高速文字列解析など)
  2. 機械学習・深層学習
  3. 統計解析
  4. 暗号理論 (プライバシー保護ゲノム解析)
  5. 自然言語処理 (論文からの生物学知識自動解析)
  6. ハードウェア(処理の高速化)

# 本講義の評価

- レポートなどはなし。試験のみでの評価を予定。
- 1/24 16:30-18:00 @ 56-101を予定
- 本講義はZoomによるオンラインで行なっているが、試験は対面を予定している。
- 諸事情により試験の参加が難しい学生は事前に申し出ること。
- COVID-19の状況次第で、予定は変更される可能性がある。



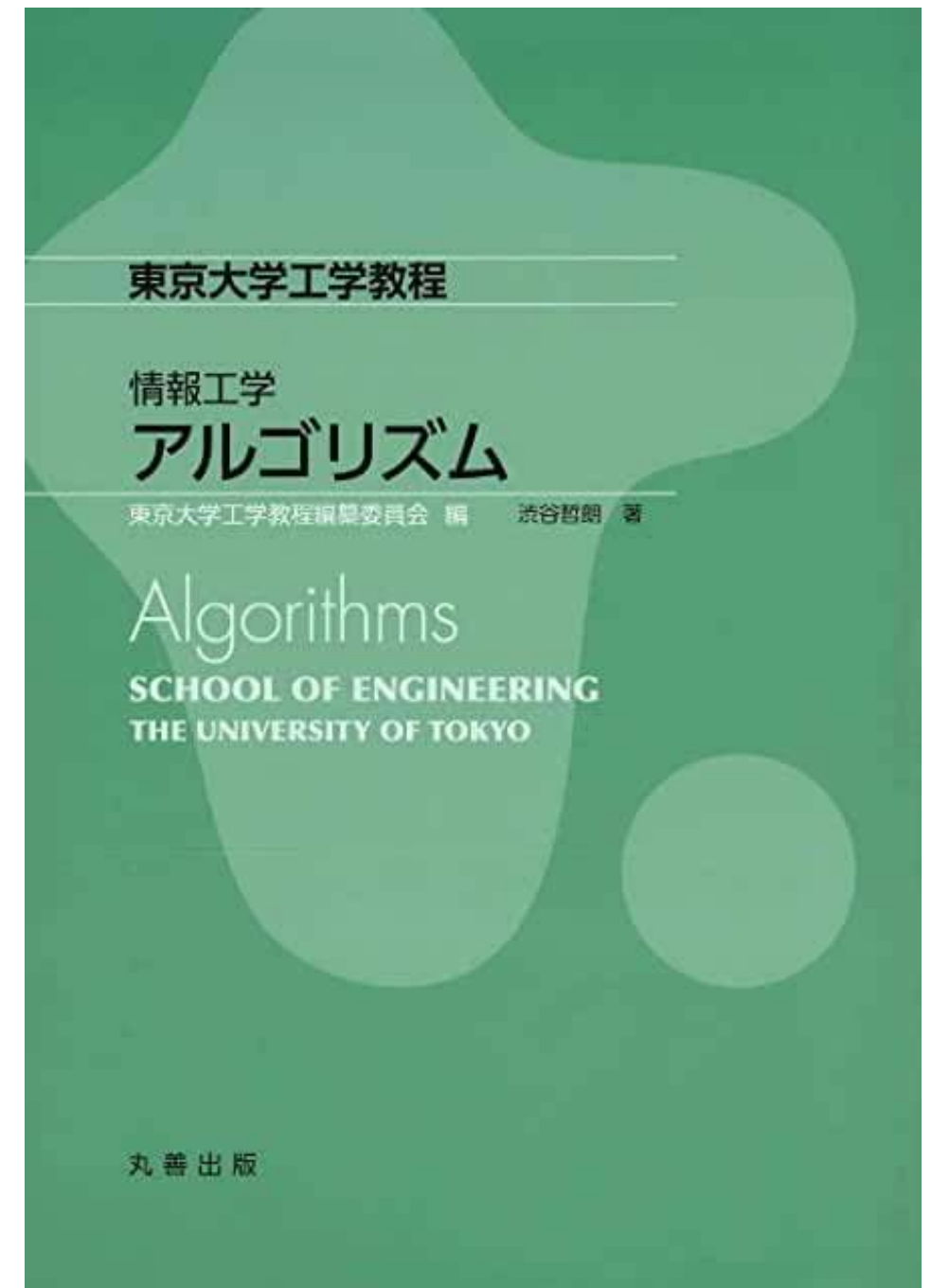
# 教科書

- 「問題解決力を鍛える！  
アルゴリズムとデータ構造」  
(通称:けんちゃん本)をメインの教科書とします。
- 図が豊富、記述が固くない、  
コードが載っているなど  
非常にわかりやすいと思います。
- 特にコメントのない限り、  
スライドの図は本教科書から、  
コードは本教科書のサポートサイトの  
Githubから引用しています。



# 参考書

- 「東京大学工学教程 アルゴリズム」  
(渋谷本)
- ページ数が少なくとっつきやすい。  
ただし記述は厳密で固く、「教科書的」  
なので、初学者向けではない。
- けんちゃん本を読んでイメージを掴んだ  
方が、より数理的に内容を把握したい  
時に読む本。



# 参考書

- 「データ構造とアルゴリズム」(杉原本)
- 渋谷本同様、ページ数が少なくとっつきやすい。渋谷本よりは読みやすいが、けんちゃん本よりはるかに教科書的。
- まずけんちゃん本を理解した方が良い。
- 講義の12回以降は、渋谷本と杉原本から、文字列・計算幾何など、けんちゃん本で扱っていない内容を主に取り扱う。



# 参考書

- 「アルゴリズム・イントロダクション」
- 1000Pを超える教科書。アルゴリズムをしっかりと勉強したい人向け。
- 杉原本を読んで更にアルゴリズムについて知識を深めたくなった場合に読むと良い。



つまり、けんちゃん本→杉原本→アルゴリズムイントロダクション。講義ではこのレベルまでは取り扱わない。


# 本講義の資料公開

- Waseda Moodle上で、本講義の録画と授業資料をアップロードいたします。
- 録画はそのままアップロードしますが、講義資料では教科書から引用した図は、(図番号のみの掲載として)全て削除します。
- 動画や授業資料の流出などがないように気をつけてください。



# 競技プログラミング

- 本講義で解説するアルゴリズムとデータ構造は、競技プログラミングと大きく関係があります。
- 競技プログラミングとは、時間内でプログラミングの問題を解きあってスコアを競争するオンラインイベントです。
- 最近では、AtCoderというサイトが人気があります。  
情報系で就活をする場合、競技プログラミングで成績が良いと有利になることもあるようです。



解けた！を世界に届けたい。

AtCoderは、世界最高峰の競技プログラミングサイトです。  
リアルタイムのオンラインコンテストで競い合うことや、  
3,000以上の過去問にいつでもチャレンジすることができます。

# 第一章

## アルゴリズムとは

# アルゴリズムとは何か？

- 「(数理的な)問題を解くための方法や手続き」
- 具体例：年齢当てゲーム

問題:

初対面のAさんの年齢を当てたいと考えます。

Aさんの年齢が20歳以上36歳未満であるとわかっているとします。

Aさんに「Yes/Noで答えられる質問」を4回まで出来るとした時、あなたはこの年齢当てゲームで勝つ事が出来るでしょうか？



# 最も単純なアルゴリズム

- 年齢を順番に聞いていく方法

Q1: あなたは20歳ですか? →Yesなら20歳、NoならQ2へ

Q2: あなたは21歳ですか? →Yesなら21歳、NoならQ3へ

...

Q4: あなたは23歳ですか? →Yesなら23歳、

Noなら質問回数終了のためゲームに負け

- もし質問回数制限がなかったとするなら質問を続け、

Q15: あなたは34歳ですか? →Yesなら34歳、Noなら35歳  
までして初めて年齢がわかる。

# より効率的なアルゴリズム

- 先の方法も問題を解くための手続きであるため、アルゴリズムであるがゲームには負けてしまう(効率が悪い)
- そこで次のように、候補を半分に絞るアルゴリズムを考える

図1.2

# より効率的なアルゴリズム

- 答えがYesであれNoであれ、年齢の候補を半分(16から8)に減らすことができる
- 同じことを繰り返す。  
たとえば最初の答えがYesであったとすると、年齢は20~27のいずれかなので、「24歳未満ですか？」と尋ねる。
- この答えがYesなら年齢は20~23のいずれか、Noなら24~27のいずれかである。どちらにせよ、年齢の候補を半分(8から4)に減らすことができる。

# より効率的なアルゴリズム

- すると候補の数の変遷は、次のようになる。  
1回目の質問:  $16 \rightarrow 8$   
2回目の質問:  $8 \rightarrow 4$   
3回目の質問:  $4 \rightarrow 2$   
4回目の質問:  $2 \rightarrow 1$  (ゲームに勝ち！)
- 具体例  
Q1: あなたは28歳未満ですか?  $\rightarrow$ No (28~35歳)  
Q2: あなたは32歳未満ですか?  $\rightarrow$ No (32~35歳)  
Q3: あなたは34歳未満ですか?  $\rightarrow$ Yes (32~33歳)  
Q4: あなたは33歳未満ですか?  $\rightarrow$ No (33歳)

あなたは33歳

# 線形探索と二分探索

- 最初に紹介した、選択肢を順番に調べていく単純なアルゴリズムは、線形探索と呼ばれる。
- 次に紹介した、候補を半分に絞っていく効率的なアルゴリズムは、二分探索と呼ばれる。
- どちらの方法も、(質問回数制限を考えなければ)  
相手の年齢が何歳であるかにかかわらず、  
同じ方法で答えを手に入れることができる。

# アルゴリズムの例(1): 深さ優先探索

- 次の虫食い算パズルを解くアルゴリズムを考える。

図1.3

- ここでは、深さ優先探索(depth-first search:DFS)という方法で虫食い算を解く手法を紹介する。深さ優先探索とは、行き詰まるまでどんどん仮定を置いて探索していき、行き詰まったら一步戻って別の選択肢を探す手法である。

# アルゴリズムの例(1): 深さ優先探索

図1.4上

# アルゴリズムの例(1): 深さ優先探索

図1.4下



# アルゴリズムの例(2): 幅優先探索

- 次の迷路を解くアルゴリズムを考える。

図1.5

ここでは、幅優先探索(breadth-first search:BFS)という方法で迷路を解く手法を紹介する。幅優先探索とは、出発点に近いところから順に探索を行う手法である。

# アルゴリズムの例(2): 幅優先探索

図1.6

# アルゴリズムの記述方法

- アルゴリズムの挙動を大雑把に伝えるには、日本語が良い  
(しかし細部が伝わらない)
- アルゴリズムを実際に実行するのはコンピュータであり、  
プログラムとして記述されている。これを踏まえ、if/for/while文などを  
活用してプログラムっぽく挙動を伝える方法を擬似コードという。
- 本講義では、実際に計算機上で動作するプログラムとして  
アルゴリズムを記述する。言語としてはC++を用いる。  
(言語特有の記述については都度解説する。)

# コーヒーブレイク

- 数え上げお姉さん問題

(<https://www.youtube.com/watch?v=Q4gTV4r0zRs>)

- 興味のある方は、

「超高速グラフ列挙アルゴリズム：〈フカシギの数え方〉が拓く  
組合せ問題への新アプローチ」(森北出版)を読むと良い。

(ただし本講義の内容をおよそ理解していからないでないと厳しい)



## 第二章

# 計算量とオーダー記法

# 計算量とは

- アルゴリズムの効率性を見積もる重要な基準として、計算量がある。計算機上での計算時間を大雑把に見積もる事が可能となる。
- まず、年齢当て問題の線形探索と二分探索にかかる計算時間を再考する

問題:

初対面のAさんの年齢を当てたいと考えます。

Aさんの年齢が20歳以上36歳未満であるとわかっているとします。

Aさんに「Yes/Noで答えられる質問」を4回まで出来るとした時、あなたはこの年齢当てゲームで勝つ事が出来るでしょうか？

# 年齢当て問題の質問回数

- 元の問題設定通りであれば、  
線形探索の質問回数：最悪15回  
二分探索の質問回数：4回
- もし年齢が0歳以上65536歳未満であれば、  
線形探索：最悪65535回  
二分探索：16回
- 一般に、年齢が0歳以上 $N$ 歳未満であれば、  
線形探索：最悪 $N-1$ 回  
二分探索：およそ $\log_2 N$ 回
- $N$ が大きいほど差が大きくなる。

# 計算量のオーダー記法

- ・ 「アルゴリズムの計算時間はデータサイズ増加に伴いどう増加するか？」
- ・ 簡単な例として、for文の反復処理の時間を考える。

## コード2.1

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int N;
6      cin >> N;
7
8      int count = 0;
9      for (int i = 0; i < N; ++i) {
10         ++count;
11     }
12 }
```

おまじない

標準入力

count = count + 1



# 計算量のオーダー記法

## コード2.2

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int N;
6      cin >> N;
7
8      int count = 0;
9      for (int i = 0; i < N; ++i) {
10         for (int j = 0; j < N; ++j) {
11             ++count;
12         }
13     }
14 }
```

# 計算量のオーダー記法

表2.1

- コード2.1の計算時間はおおよそ $N$ に比例する
- コード2.2の計算時間はおおよそ $N^2$ に比例する

# 計算量のオーダー記法

- アルゴリズムAの計算時間が、概ね $P(N)$ に比例する場合、アルゴリズムAの計算量は $O(P(N))$ であると呼ぶ。
- つまりコード2.1の計算量は $O(N)$ であり、コード2.2の計算時間はおおよそ $O(N^2)$ である。これをランダウの $O$ 記法と呼ぶ。  
(厳密な定義は後ほど)

# 計算量のオーダー記法

- あるアルゴリズムの計算時間 $T(N)$  が、
$$T(N) = 3N^2 + 5N + 100$$
で表されるとする。この時の計算量は？
- まず高次の項ほどデータサイズ $N$ の影響をうけるので、低次の項を無視する( $3N^2$ )。  
そして係数を無視する( $N^2$ )。
- すなわち、
$$T(N) = O(N^2)$$
となる。

# 計算量を求める例(1):偶数の列挙

- 正の整数Nを受け取って、N以下の偶数を全て出力するアルゴリズム

```
1  #include <iostream>
2  using namespace std;
3
4  int main() {
5      int N;
6      cin >> N;
7
8      for (int i = 2; i <= N; i += 2) {
9          cout << i << endl;
10     }
11 }
```

- for文の反復回数はおおよそ $N/2$ 回なので、計算量は $O(N)$

# 計算量を求める例(2):最近点对問題

問題:

N個の座標値( $x_i, y_i$ ) ( $i = 0, 1, \dots, N-1$ )が与えられた時、最も距離が近い2点間の距離を求めなさい。

- 全ての点对について距離を計算し、そのうち最小のものを出力する

図2.3

# 計算量を求める例(2):最近点对問題

```
1  #include <iostream>
2  #include <vector>
3  #include <cmath>
4  using namespace std;
5
6  // 2 点 (x1, y1) と (x2, y2) との距離を求める関数
7  double calc_dist(double x1, double y1, double x2, double y2) {
8      return sqrt((x1 - x2) * (x1 - x2) + (y1 - y2) * (y1 - y2));
9  }
10
11 int main() {
12     // 入力データを受け取る
13     int N; cin >> N;
14     vector<double> x(N), y(N); | xとyという2つの配列を用意
15     for (int i = 0; i < N; ++i) cin >> x[i] >> y[i];
16
17     // 求める値を、十分大きい値で初期化しておく
18     double minimum_dist = 100000000.0;
```

# 計算量を求める例(2):最近点对問題

```
20      // 探索開始
21      for (int i = 0; i < N; ++i) {
22          for (int j = i + 1; j < N; ++j) {
23              // (x[i], y[i]) と (x[j], y[j]) との距離
24              double dist_i_j = calc_dist(x[i], y[i], x[j], y[j]);
25
26              // 暫定最小値 minimum_dist を dist_i_j と比べる
27              if (dist_i_j < minimum_dist) {
28                  minimum_dist = dist_i_j;
29              }
30          }
31      }
32
33      // 答えを出力する
34      cout << minimum_dist << endl;
```



# 計算量を求める例(2):最近点对問題

- for文のループ回数が重要
- $i = 0$ の時、 $j$ は $N - 1$ 回ループする。  
 $i = 1$ の時、 $j$ は $N - 2$ 回ループする。  
...  
 $i = N - 2$ の時、 $j$ は1回ループする。
- よって、
$$T(N) = (N - 1) + (N - 2) + \cdots + 1 + 0 = \frac{1}{2}N^2 - \frac{1}{2}N$$

すなわち、 $T(N) = O(N^2)$

# 計算量の使い方

- 計算実行時間と解きたい問題のサイズについて、ざっくりとした感覚を掴む事が重要。
- 現在の一般家庭用の計算機の計算速度は、1秒間で処理できる計算ステップ回数が $10^9$ 程度と見積もって良い。

(計算実行環境に強く依存することに注意、あくまでも大雑把な感覚)

# 計算量の使い方

- 入力サイズと計算ステップ数の関係は次の通り ( $\log$ の底は2)  
1秒以上かかるものは-とされている。

表2.2

# 計算量の使い方

- $2^N$ や $N!$ といった計算量を要するアルゴリズムを、指数時間アルゴリズムと呼ぶ。
- また、ある定数 $d > 0$ によって、その計算量が $N^d$ により上から抑えられるアルゴリズムを、多項式時間アルゴリズムと呼ぶ。
- また、データサイズ $N$ に対して計算時間が依存せず、定数時間で処理が終了する場合、定数時間アルゴリズムと呼び、 $O(1)$ と表記する。

# 計算量に関する注釈

- 指数時間アルゴリズムはNが大きい時全く実用的ではないが、Nが小さければ十分に有効。解きたい問題のサイズを確認すべき。
- 計算量解析では係数が無視されるが、現実には大きく効いてくる(Nが小さい場合特に)。計算量が小さいアルゴリズムが実際に必ず高速なわけではない。

例)

1.  $T(N) = N^2 + N + 1$  ( $T(N) = O(N^2)$ )

2.  $T(N) = 1000N$  ( $T(N) = O(N)$ )

N=10の時、1は111だが2は10000であり、2の方が計算時間がかかる。

# 計算量に関する注釈

- これまでの議論は全て計算時間に関するものであった。  
正確には、これらの計算量は時間計算量とも呼ばれる。
- アルゴリズム実行に必要なメモリ使用量についても、同様にランダウの $O$ 記法で計算量を表現することができる。この場合、これらの計算量は領域計算量(空間計算量)と呼ばれる。
- 今後特に注釈のない場合は、計算量とは時間計算量を指すものとする。

# 計算量に関する注釈

- アルゴリズムの実行時間は、入力データの偏りなどで計算時間が変わる事がある。  
(年齢当てゲームの線形探索は1回で終わることもある。)
- 最悪ケースにおける時間計算量は最悪時間計算量と呼び、  
平均的なケースにおける時間計算量は平均時間計算量と呼ぶ。
- 今後単に計算量という場合は、最悪時間計算量を指すものとする。

# ランダウの $O$ 記法の詳細

ランダウの $O$ 記法の定義



# ランダウのO記法の詳細

- $T(N) = 3N^2 + 5N + 100$  を再考する。まず  $T(N)$  を  $N$  で割ってみると、

$$3N + 5 + \frac{100}{N}$$

となり、明らかに定数  $c$  で抑える事は出来ない。

よって  $T(N) = O(N)$  ではない。

- 次に  $T(N)$  を  $N^2$  で割ると、

$$3 + \frac{5}{N} + \frac{100}{N^2}$$

となり、 $N$  が十分に大きい時、これは  $c=4$  で抑えられる。

よって、 $T(N) = O(N^2)$

# ランダウの $O$ 記法の詳細

- 定義から、 $T(N) = O(N^3)$  も正しいが、可能な限りタイトな関数で書く事が一般的である。
- なお、上界ではなく下界を考えた場合は $\Omega$ 記法を用いる。

$\Omega$ 記法の定義

# ランダウの $O$ 記法の詳細

- $T(N) = O(P(N))$  であり、また  $T(N) = \Omega(P(N))$  でもある場合には、 $T(N) = \Theta(P(N))$  と書く。
- これは、アルゴリズムの計算時間が上からも下からもタイトに抑えられていることを意味する。ただし慣習として、 $\Theta$ 記法を使える場合でも  $T(N) = O(P(N))$  と書く事が多く、本講義もそれに倣う。

# 本日のまとめ

- (数理的な)問題を解く方法や手続きをアルゴリズムという。
- アルゴリズムの時間的な効率を評価する方法が計算量であり、計算時間のオーダーを見積もる事が出来る。
- 同一の問題を解くアルゴリズムであっても、計算量には違いがある事があり、データセットが一定より大きい時には計算量が小さいアルゴリズムを利用した方が効率が良い。