はじめに

- ネットワークが不安定なので突然授業がおちる可能性があります
- その場合、そのままお待ちください

教養としての
アルゴリズムとデータ構造
「イントロダクション」

小林浩二

kojikoba@mi.u-tokyo.ac.jp

授業の基本情報

- 科目名:
  - 「教養としてのアルゴリズムとデータ構造」
    - 理学部
    - 後期教養教育科目
      - https://www.u-tokyo.ac.jp/ja/students/special-activities/koukikyouyou.html
  - 「知能社会情報学特別講義IV」
    - 情報理工学系研究科
- 数理データサイエンス教育プログラム:
  - 数理・情報教育研究センター（MIセンター）
  - http://www.mi.u-tokyo.ac.jp/mds-oudan/

自己紹介

- 小林浩二
- 所属:
  - 数理・情報教育研究センター（MIセンター）
  - 総合文化研究科広域科学専攻広域システム科学系
  - 教養学部
    - 大学組織としては、教養学部の情報・図形科学部会
    - 「情報」、「アルゴリズム入門」の授業を担当している部局
- 専門:
  - アルゴリズム

授業の基本情報

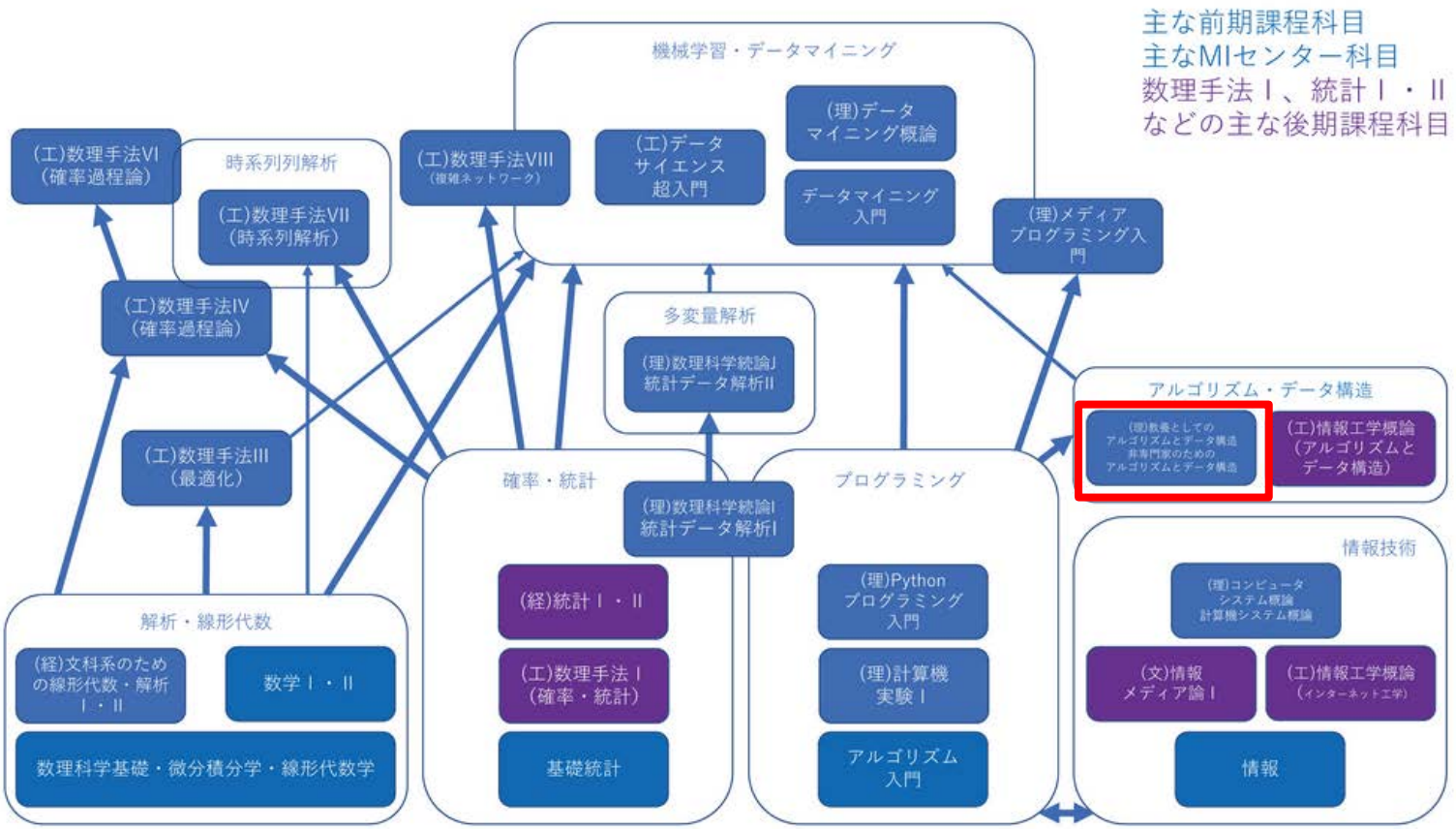
- 期間:
  - S2ターム
- 単位:
  - 1単位
- 対象:
  - 後期課程(学部を問いません)
  - 大学院生
    - 単位取得は可能。修了要件に含められるかは所属に確認して下さい。聴講のみも可能です
    - 情報理工学系研究科所属の人は「知能社会情報学特別講義IV」を受講できません
  - 前期課程
    - 単位取得不可。聴講は可能です

授業の基本情報

- 授業形式:
  - オンライン（Zoom）
    - URL: ITC-LMSを参照して下さい
  - 授業の動画は録画しています。
    - 動画のアップロードは概ね授業の翌日の夜までに行います
    - 視聴にはECCSのアカウントが必要です
    - https://drive.google.com/drive/folders/1IKyi5-0sLSwmW5GOBUgqHAuTZ8lAo09V?usp=sharing

授業の位置づけ

- データサイエンスを学ぶには、そのための基礎として、数理系科目（解析・線形代数）、統計系科目（確率・統計）、情報技術系科目、プログラミング系科目を履修する必要があります。（MIセンターのウェブより）
  - http://www.mi.u-tokyo.ac.jp/mds-guide/index.html



授業の位置づけ

- Python（プログラミング）関係でみると
  - この授業は、プログラムでより効率的（速さ、記憶領域）にプログラムを書きたい方向け
    - Pythonを始めたい方：「Pythonプログラミング入門」（SS, A1）
    - Pythonの流儀で効率的にプログラムを書きたい方：「Pythonプログラミング作法」（SS）
    - Pythonで色々なことがやりたい方：「メディアプログラミング入門」（S2火5,A2）
    - Pythonで直接的にデータサイエンス関係をやりたい方：「データマイニング概論」（S）、「データマイニング入門」(A)
      - Rなら、「データサイエンス超入門」(S1)

授業の紹介

- 教養としてのアルゴリズムとデータ構造
  - アルゴリズムとは、「問題」を解くための手順であり、データ構造は計算機（コンピュータ）上においてデータを保持するための方法である。これらは計算機科学の分野において基本的かつ重要な役割を果たしている。本授業ではアルゴリズム及びデータ構造の基礎について学ぶとともに、Python言語を通じてその機能を体験することを主眼とする。（シラバスから抜粋）

授業の紹介

- 教養としての アルゴリズムとデータ構造
  - アルゴリズムとは、「問題」を解くための手順
    - 我々の身の回りには様々な「問題」が存在する
    - 「問題」の例：
      - 数を昇順に並べ替える
      - データの集まりの中から、あるデータを探し出す
      - 長大な文字列の中から、特定の文字列を抜き出す
      - 地図上の2つの地点間の最短距離・経路を求める
      - 家系図の中から2者の関係を求める
      - コンビニで客をどのレジに並ぶか指示する
      - 1つのケーキを誰も不満を感じずに分け合う
      - 4人でタクシーに乗った時に1人が料金を立て替えた後、その料金を効率よく立て替えた人に支払う
  - この授業では、計算機上の「問題」を主に扱います

授業の紹介

- 教養としてのアルゴリズムとデータ構造
  - データ構造とは、問題を解くアルゴリズムが扱うデータを格納する為の方法
    - Pythonにおける基本的なデータ構造の例:
      - 文字列
      - リスト
      - 辞書

      一般にこれらを組み合わせるなどしてより複雑なデータ構造を構成します

授業の紹介

- 教養としてのアルゴリズムとデータ構造
  - 情報学分野以外でも色々使われています
    - 身の回りで役に立つ（立ちそうな）アルゴリズムやデータ構造の仕組みと働きについて学ぶ
  - “Computational Thinking”, J.M.Wing (CACM, 49(3), 2006)（「計算論的思考」中島秀之、情報処理, 56(6), 2015年）
    - 「計算論的思考は、コンピュータ科学者だけではなく、すべての人にとって基本的な技術である。すべての子供の分析的思考能力として、「読み、書き、そろばん（算術）」のほかに計算論的思考を加えるべきである。」

授業の概要

- 座学でアルゴリズムやデータ構造の仕組み・働きについて学ぶとともに、実際に手を動かして（Pythonで）それらについて体験します
  - コンセプトは、「Pythonプログラミング入門」の延長でアルゴリズムとデータ構造について学習する授業
    - ×「複雑なアルゴリズムやデータ構造をPythonで頑張って書いて『問題』を解く」
    - ○「仕組み・働きを学んだアルゴリズムやデータ構造について、出来合いのモジュールなどを利用して（ある程度効率的に）『問題』を解く」
    - できるだけ身の回りの実データを扱います
      - 官公庁やSNS,インターネット上で取得できる様なデータ

授業の紹介

- 教養としてのアルゴリズムとデータ構造
  - アルゴリズムとは、「問題」を解くための手順
    - 我々の身の回りには様々な「問題」が存在する
    - 「問題」の例：
      - 数を昇順に並べ替える
      - データの集まりの中から、あるデータを探し出す
      - 長大な文字列の中から、特定の文字列を抜き出す
      - 地図上の2つの地点間の最短距離・経路を求める
      - 家系図の中から2者の関係を求める
      - コンビニで客をどのレジに並ぶか指示する
      - 1つのケーキを誰も不満を感じずに分け合う
      - 4人でタクシーに乗った時に1人が料金を立て替えた後、その料金を効率よく立て替えた人に支払う
  - この授業では、計算機上の「問題」を主に扱います

授業の概要

- 探索とは、「データの集まり（集合）の中からあるデータを探し出す」こと
- Pythonにおける探索：

```
list1 = [24, 48, 26, 2, 16, 42, 31, 25, 50, 19, 30, 22, 37, 13]
print(16 in list1)
print(29 in list1)
```

```
True
False
```

  - この探索がどの様に行われているか
  - 探索にかかる時間はどれくらいか
  - 他にどの様な探索方法があるのか
  - 探索を使って色々やってみる

到達目標

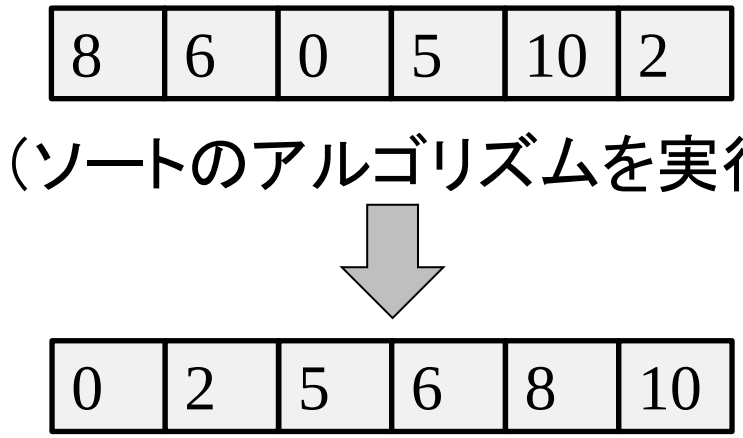
- プログラムで「問題」を解く際に、適切なアルゴリズム・データ構造を選択し、自分が書いているプログラムがどの程度の時間・計算機（パソコンなど）の能力を使用して動くのか見積もれる様になる（為の素地を身に付ける）
  - 「問題」を解く
    - 実データを扱う様な「問題」を解く場合、「問題」の解き方（アルゴリズム・データ構造）が重要になります
    - 答えが正しくても非常に時間がかかったり、計算機の記憶領域を大量に使ったりする様なアルゴリズムは、あまり良いアルゴリズムとは言えません

授業の紹介

- 教養としてのアルゴリズムとデータ構造
  - アルゴリズムとは、「問題」を解くための手順
    - 我々の身の回りには様々な「問題」が存在する
    - 「問題」の例：
      - 数を昇順に並べ替える
      - データの集まりの中から、あるデータを探し出す
      - 長大な文字列の中から、特定の文字列を抜き出す
      - 地図上の2つの地点間の最短距離・経路を求める
      - 家系図の中から2者の関係を求める
      - コンビニで客をどのレジに並ぶか指示する
      - 1つのケーキを誰も不満を感じずに分け合う
      - 4人でタクシーに乗った時に1人が料金を立て替えた後、その料金を効率よく立て替えた人に支払う
  - この授業では、計算機上の「問題」を主に扱います

ソート

- ソート（並べ替え、整列）とは、「順序付けられるデータを、一定の規則に従って並べ替える」こと



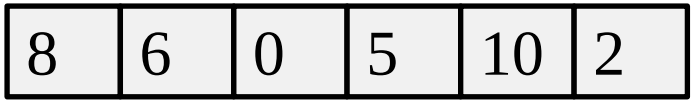

```
list_test = [8, 6, 0, 5, 10, 2]

# list_testをソート

print(list_test)
```

```
[0, 2, 5, 6, 8, 10]
```

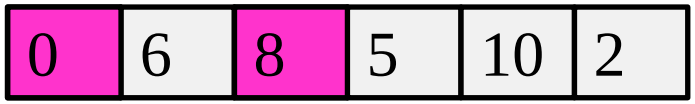
- どの様なアルゴリズムでソートするか？

選択ソート

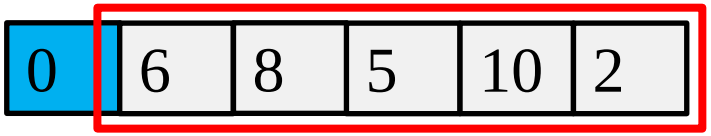
- リストの中から最小値を探す



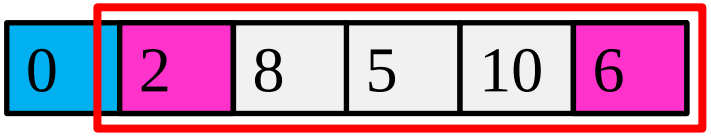
- 最小値と左端の値を入れ替える



- 左端以外のリスト（部分リスト）の中から最小値を探す

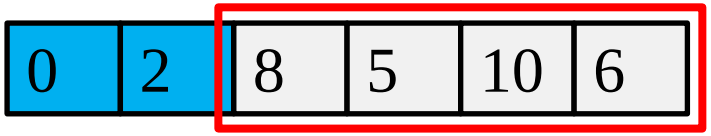


- 最小値と部分リストの左端の値を入れ替える

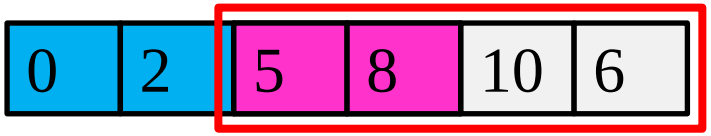


選択ソート

- 左端2つ以外のリストの中から最小値を探す

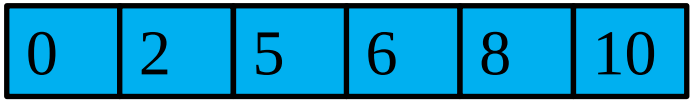


- 最小値と部分リストの左端の値を入れ替える



- これを繰り返す

⋮　(以下略)



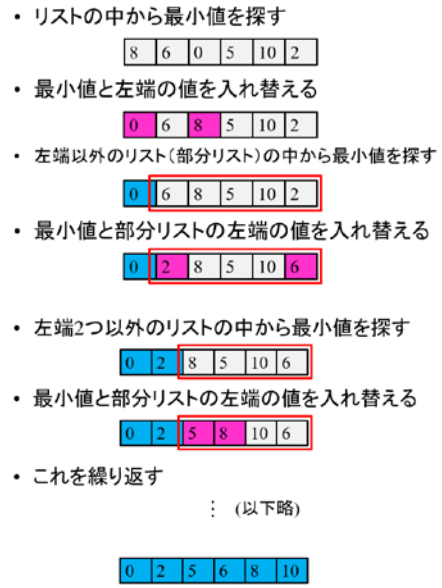
選択ソート

```
def SelectionSort(list1):
    for stindex in range(0, len(list1)-1):
        minval = list1[stindex]
        minvalindex = stindex
        for index1 in range(stindex, len(list1)):
            if list1[index1] < minval:
                minval = list1[index1]
                minvalindex = index1
        list1[minvalindex], list1[stindex] = list1[stindex], list1[minvalindex]
    return list1
list_test = [8, 6, 0, 5, 10, 2]
SelectionSort(list_test)
```

```
[0, 2, 5, 6, 8, 10]
```

選択ソートにかかる時間

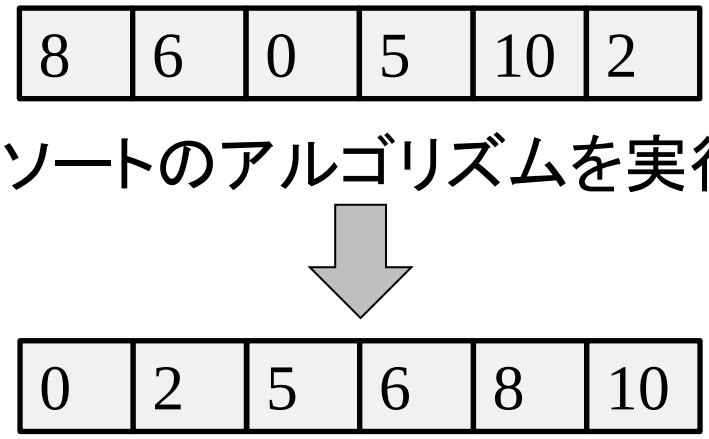
- 長さkの（部分）リストから最小値を探す時間：
  - 部分リストの長さkに比例する
    - n=リストの長さ（要素数）とすると、k=n, n-1, …, 2, 1
- 最小値を探す回数：
  - リストの長さ（≦ n）
- 最小値と左端の値を入れ替える時間：
  - 殆どかからない



- 実行時間≒最小値を探す時間×最小値を探す回数≦ n × n
  - つまり、ほぼ n^2（リストの長さの2乗）に比例
  - 厳密には、n+n-1+ … + 2 + 1 = n(n+1)/2に比例

ソート

- ソート（並べ替え、整列）とは、「順序付けられるデータを、一定の規則に従って並べ替える」こと

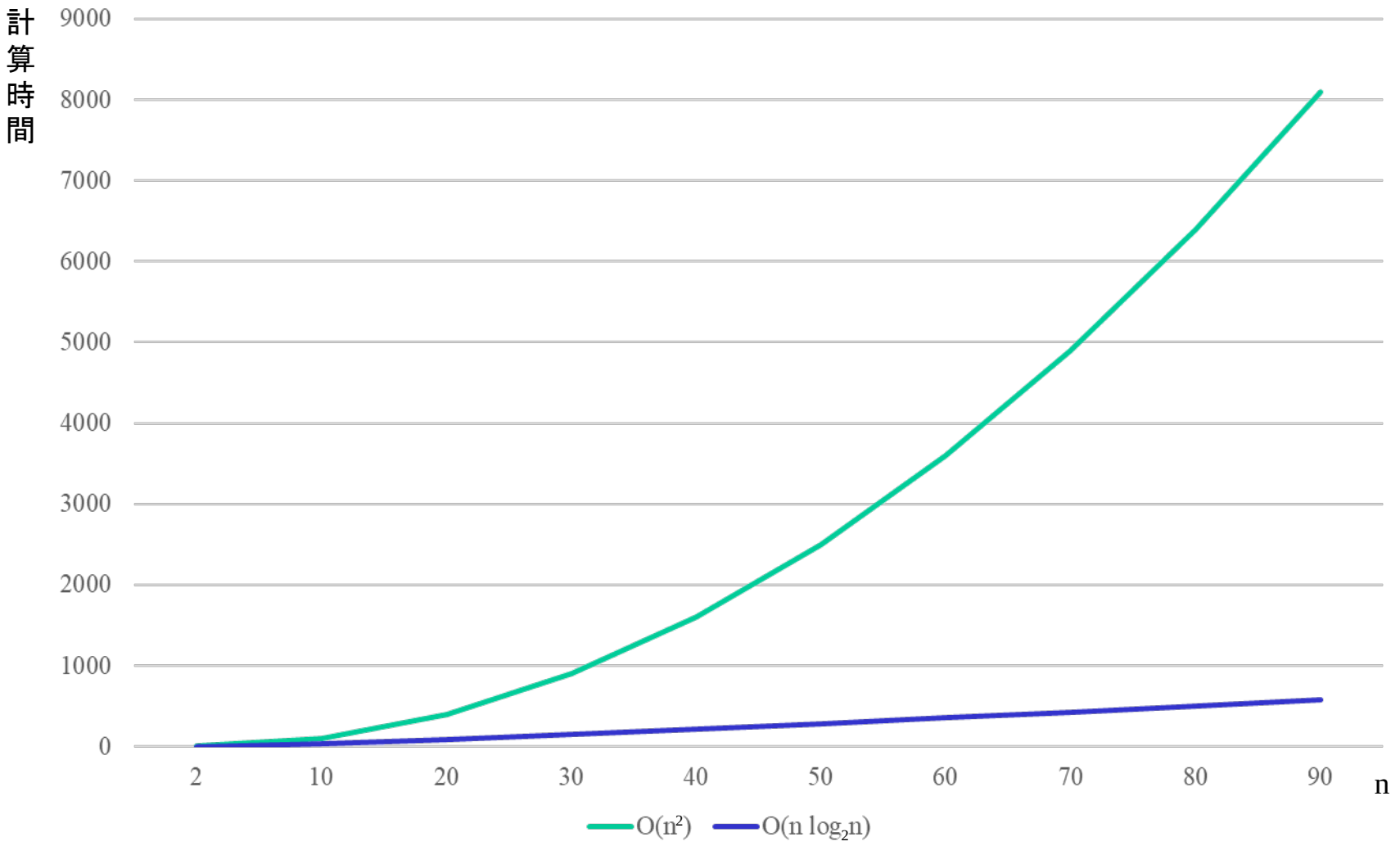



```
: list_test = [8, 6, 0, 5, 10, 2]

list_test.sort()
#もしくは list_test = sorted(list_test)

: print(list_test)

[0, 2, 5, 6, 8, 10]
```

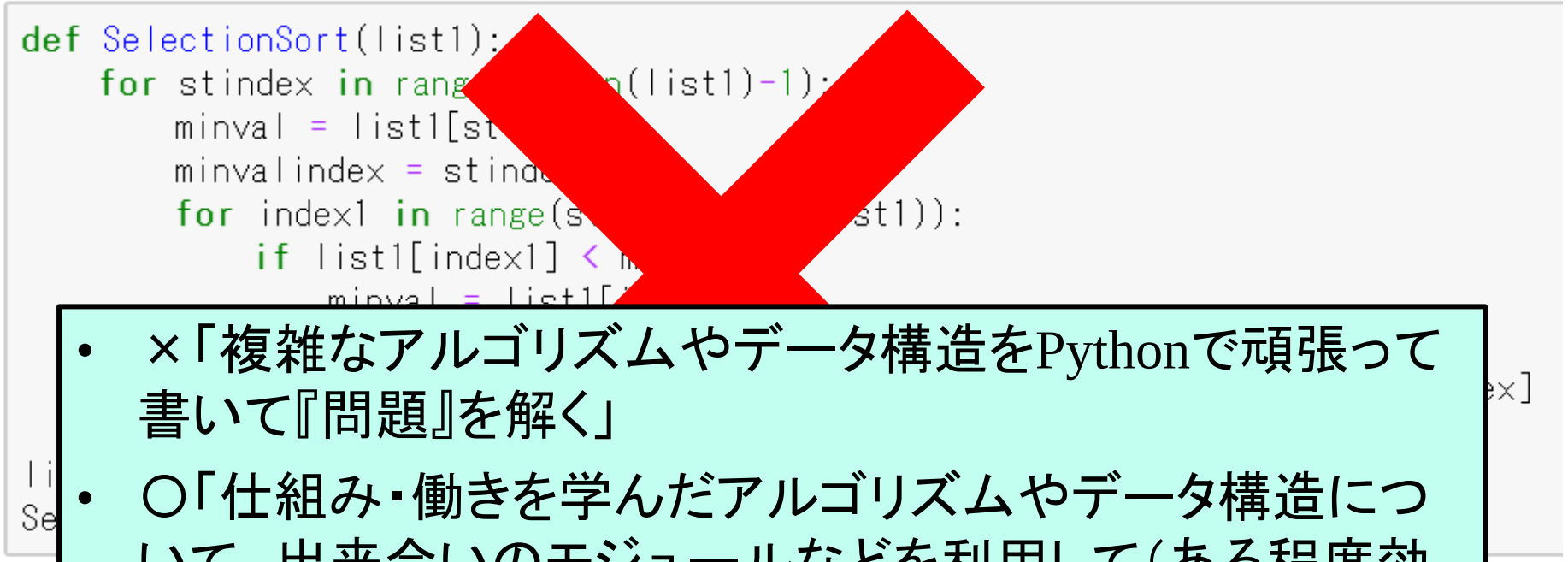
- sort, sortedは最も時間がかかる場合でも $n \log_2 n$に比例
  - n = リストの長さ
- 自作した選択ソートより圧倒的に高速

ソート

- 選択ソートの計算時間 (n^2) とPythonのsortの計算時間($n \log_2 n$) を比較してみます。



選択ソート



- ×「複雑なアルゴリズムやデータ構造をPythonで頑張って書いて『問題』を解く」
- ○「仕組み・働きを学んだアルゴリズムやデータ構造について、出来合いのモジュールなどを利用して（ある程度効率的に）『問題』を解く」
- 答えが正しくても非常に時間がかかったり、計算機の記憶領域を大量に使ったりする様なアルゴリズムは、あまり良いアルゴリズムとは言えません

授業の進め方

- スライドを用いた講義 （概ね授業前半）
  - アルゴリズムやデータ構造の座学
- 前回授業の課題の解説と講評 （概ね授業後半）
  - 模範解答は授業中に公開します
- 演習 （時間があれば）
  - 座学に関係する課題を解答
  - 質問は授業時間内はZoom、時間外はSlack
    - https://join.slack.com/t/ut-aad/shared_invite/zt-p42o4rgg-FJDDqy8vG_75Jq~bPd4ZRw
    - 基礎課題：授業日2日後までに提出（予定）
    - 本課題：次回授業日までに提出（予定）
      - 最終回の本課題は、少し大きめの課題
      - 各課題の提出はITC-LMS経由
    - 課題はITC-LMSで配布します

授業の計画

- 6/9: イントロダクション
- 6/16: 探索
- 6/23: 木構造とその探索
- 6/30: グラフ構造
- 7/7: ソートと再帰
- 7/14: ヒープ
- 7/21: 列挙アルゴリズム（？）

授業の計画（課題）

- 6/9: 第1回本課題（オプション）（-6/15）
- 6/16:第2回基礎課題(-6/18)、第2回本課題(-6/23)
- 6/23:第3回基礎課題(-6/25)、第3回本課題(-6/30)
- 6/30 :第4回基礎課題(-7/2)、第4回本課題(-7/7)
- 7/7 :第5回基礎課題(-7/9)、第5回本課題(-7/14)
- 7/14 :第6回基礎課題(-7/16)、第6回本課題(-7/21)
- 7/21 :第7回基礎課題(-7/23)、最終本課題(-8/4)

履修上の注意

- S1開講の「Pythonプログラミング入門」の前半（1-3回）において学習する程度のPythonの知識が必要
  - https://sites.google.com/view/ut-python
  - 2021年度S1の当該授業の教材
    - 第1回: 数値演算、変数と関数の基礎、論理・比較演算と条件分岐の基礎、デバッグ
    - 第2回: 文字列、リスト、条件分岐
    - 第3回: 辞書、繰り返し、関数

成績評価方法

- 各回の授業において出題する基礎課題と本課題に対する結果によって評価する。
  - 演習課題:（ほぼ100%）
    - 基礎課題
    - 本課題
  - 出席:
    - 取る予定ですが、成績にそこまで影響しません
  - 最終の本課題を提出しない場合、評価は「未受験」になります

昨年の実績(？)

- 最終課題を提出した人数 /「優」取得人数：
  - 40 / 36 (うち優上3)
  - 50 / 43 (うち優上1),

課題の評価

- 提出後、解答を添削して評価・コメントを返します
  - 返すタイミング未定（提出者数依存）
  - 評価・コメントの内容は適宜確認して下さい
- 評価
  - 基礎課題：0点-2点の3段階
  - 本課題： 0点-2点の3段階
    - 2点 > 1点 >締切後（再提出）の2点 >0点>締切後（再提出）の1点
    - 「正解です」：模範解答とほぼ同じ
    - 「正解ですが…」・「正解にしていますが…」：正しいが計算時間などの観点から模範解答に劣る

教材

- ITC-LMSの授業のページに座学で用いるppt（pdf）と演習課題を公開します
  - 2019年度
  - 2020年度
- 次回以降も教材はITC-LMSから取得して下さい

よくある質問と回答

- 初回の授業に出席できなかった
  - このスライドに書いてあることしか基本話してません
- 課題の模範解答はどこかで公開される？
  - 各回の授業で公開しますが、配布はしません
- 課題が解けない
  - 授業中に十分な成績が得られる程度のヒントは毎回出します
  - 分からない場合はできたところまでの解答（ソースコード）を見せてもらえれば何かしらヒントを出します
- 出張などで授業に出席できない（課題が提出できない）
  - 個別に相談して下さい
- 単位は簡単に取れる？
  - 毎回課題を提出すればそれなりの成績になります

よくある質問と回答

- 教材はいつ公開される？
  - 概ね授業の直前です
- 課題はいつ公開される？
  - 授業の直前・授業中・授業後
- 課題はいつ提出可能になる？
  - 課題が公開されてから
- 課題が間違っている様な気がする
  - 間違っていそうだと思ったら、直ちに連絡して下さい

Pythonの導入

- Pythonの実行環境がない人:
  - 自前のPCにインストール
    - Anaconda でググって下さい
      - 標準的なモジュール類しか使いません
      - https://utokyo-ipp.github.io/0_guidance/guidance1.html
    - Colaboratoryでも実行可能です

課題

- ちょっとした「問題」を解いてみましょう
  - ex1.ipynb
    - 本課題（15日締め切り）
    - とりあえず自分なりに解いてみましょう