BMI3 Lecture 1.1
Introduction to Algorithms

Wanlu Liu

ZJU-UoE Institute

Wanluliu@intl.zju.edu.cn

2022/09/13

BMI3 Week1 2022

WELCOME BACK!!!

General Information - BMI 3

1.1 Overview of Course

- Biomedical Informatics 3 provides students the opportunity to develop their skills and understanding of **algorithms in biomedical informatics**.

- The course will provide a solid theoretical foundation on various algorithms, with an emphasis how to **design algorithms for biomedical problems**, and **how these algorithms can be applied to biomedical problems.**

- To facilitate the students' skills for the application of algorithms for biomedical data analysis and to improve their problem-solving skills, the course also introduces sessions for **advancing communication with biomedical scientists and software engineers.**

Credits for BMI3 and your UoE/ZJE degrees

- You require **90** ZJU credits* to progress to Y4
- You require **360** UoE credits* to progress to Y4
- *credits accumulated across all 3 years

- This course is worth **5** ZJU credits
- **20** UoE credits

- This course contributes **3.3%** of the final ZJU GPA or **3.6%** to the final GPA for international students

- This course contributes **5.6%** of the final UoE degree mark used for degree classification

General Information - BMI 3

Every week: two 3-hours teaching activity sessions

Lectures & Practicals

1. Algorithms and Complexity
2. Intro to data structure
3. Exhaustive Search
4. Greedy Algorithms
5. Graph Algorithms
6. Dynamic Programming
7. String algorithm and Combinatorial Pattern Matching
8. Divide-and-Conquer Algorithms
9. Randomized Algorithms
10. Clustering
11. Dimensionality reduction
12. Hidden Markov Models

Workshops

1. Numpy, Pandas, Sci-kit, advanced coding practices in python
2. Communication and professionalism 1; Live stream programming in python; Intro to LaTeX
3. How to debug
4. Communication and professionalism 2: Towards the next step in your bioinformatics career
5. Communication and professionalism 3: research seminar and career sharing by bioinformatics researcher in Tencent AI Lab
6. Review Sessions & mock exam

Course learning outcomes

After taking this course, students will be able to:

- Describe and discuss the **main principles of algorithms**, and how they can be used to solve biomedical problems.
- Characterize and **render biomedical problems into bioinformatics problems**
- **Develop, implement and apply** bioinformatics algorithm knowledge to solve an omics problem in a group setting.

Course handbook: available on **blackboard** under **BMI3 (2022-2023)**

Course website: https://labw.org/bmi3-2022/
usename: zje pass: 85c701e0

Course Team

Dr. Wanlu LIU
ZJE, Course Organizer

Dr. Hugo SAMANO
ZJE

Dr. Zhaoyuan Fang
ZJE

Teaching Assistant
Ziwei Xue, Ph.D. student, ZJE

Teaching Assistant
Ruonan Tian, Master student, ZJE

Contact info in course handbook

Course Structure

Teaching activities

Teaching activities are from **9:00 – 11:50 on Tuesday** and **14:00-16:50 on Thursday** every week. Lectures, practical, workshops will be included in those teaching activities. Copies of the lecture slides will be given on Blackboard Learn at least 24 hours before the lecture.

Assessment And Examination Information

This course is assessed by in-course assessment (ICA) and final exam.
In order to pass the BMI3 course, you must:

1. Submit all assessments that contribute to the ICA component of the course, AND
2. obtain an overall ICA mark of at least 60% in the ICAs, AND

The ICAs consist of:

1. A **'course-project'** involving choosing and solving a more elaborate problem (**group-based**, assessed based on:
   1. **a reflective written report, 20% (individual mark)**
   2. **source code 10% (group mark)**
   3. **oral presentation, 20% (group mark)**

Computer-based timed exam

You will receive and individual mark for this assignment that is **worth 50% of your final mark**. This assessment will happen under exam conditions during the **semester 1 exam period.**

- This assessment is **open-book,** which means that you are allowed to bring and consult any **notes** you may have. However, you **will not be allowed to access the internet** and you will **not be allowed to work with other students**. Detailed instructions will be distributed prior to the assessment.
- You will receive your provisional marks and feedback for this final exam **within 15 working days**.
- This final exam will be assessed **anonymously by two independent markers** and an average mark of their marks used.

Mini-project ICA

Four to five students will form a group to work on a mini-project related to **biomedical informatics algorithms project**. You will receive a group mark for this assessment that is worth 30% , and 20% of individual mark of your final mark including 20% from a reflective written report (individual mark), 10% from the source code of your project (group mark), and 20% from group oral presentation (group mark).

- The deadline for this ICA presentation is during practical time on **12/15 12:00-16:50**
- The deadline for ICA report and code is **12/16 12:00 p.m. (noon)**
- You will receive your provisional marks and feedback for this ICA within 15 working days

Summary Table of ICA submission and feedback deadlines

ICA	Submission Deadline	Feedback Deadline
Computer-based timed exam (50%)	Semester 1 exam period	Within 15 days
Group mini-project presentation (20%)	2022/12/15	Within 15 days
Group mini-project written report (20%)	2022/12/16	Within 15 days
Group mini-project source code (10%)	2022/12/16	Within 15 days

Your feedback is important

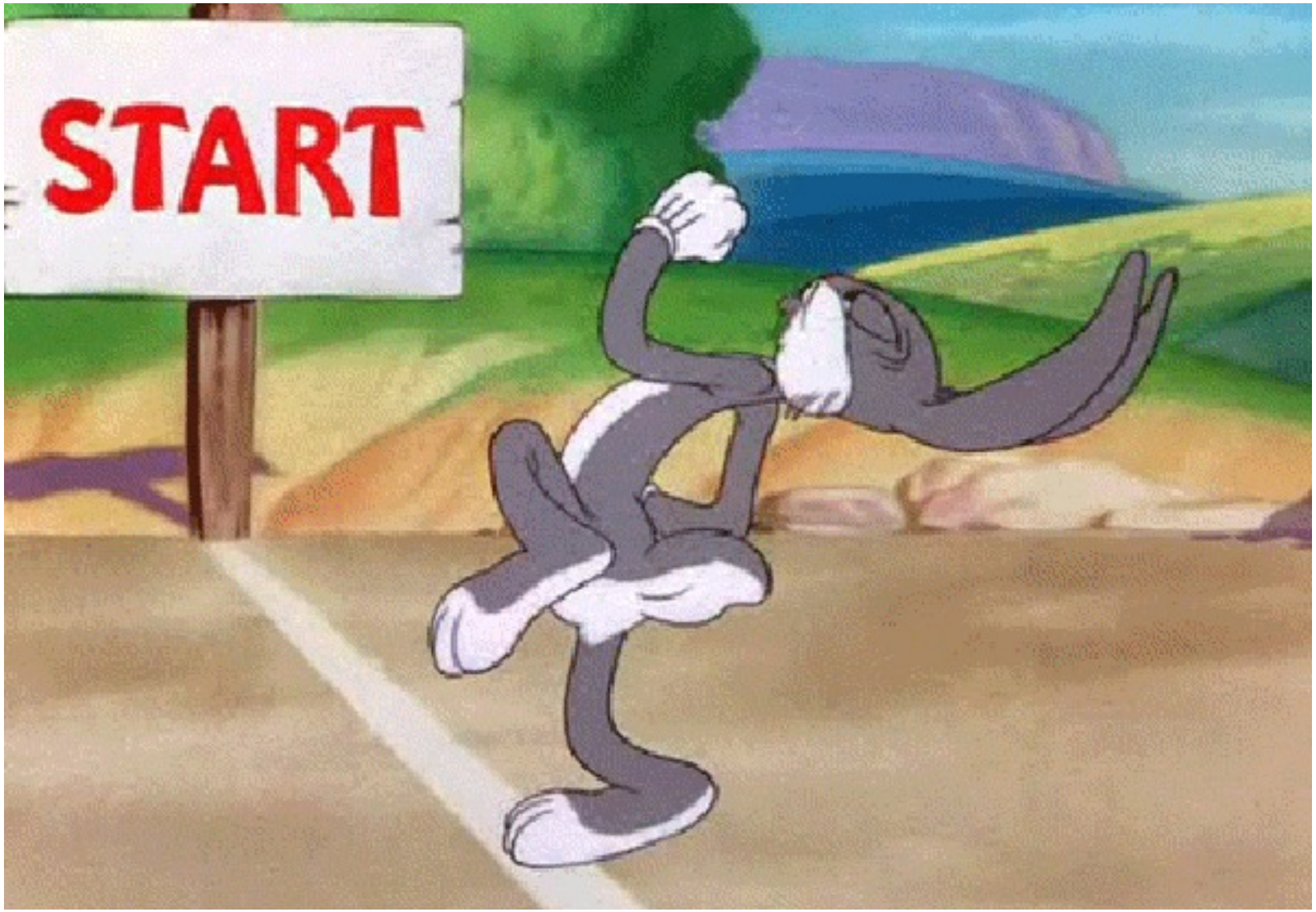
- **Ways to give feedback**
  - Student Rep
  - Mid-term course evaluation
  - End course feedback survey

- **To address last year's feedback, we did:**
  - Add intro to data structure workshop
  - Add intro to linear algebra workshop
  - Add more python programming tutorials workshops
  - Group work add individual reflective reports …..

Student Rep

Any volunteer?

Let’s begin our BMI3 tour!



Learning objectives for this lecture

- Review basic concepts in python3
- Recall python programming basics
- Describe the principles of an algorithm
- Define and apply pseudocode for algorithms
- Distinguish Iterative vs. Recursive algorithms

Let's Set up the 'environment' for this course

- Installation of Python3
- Virtual Environment (VENV)
- Installation/Intro Pip
- Installation/Intro of Conda
- Installation/Intro VSCode
- General recap (from IBI1😀) of Python3 programming
- The PTA system

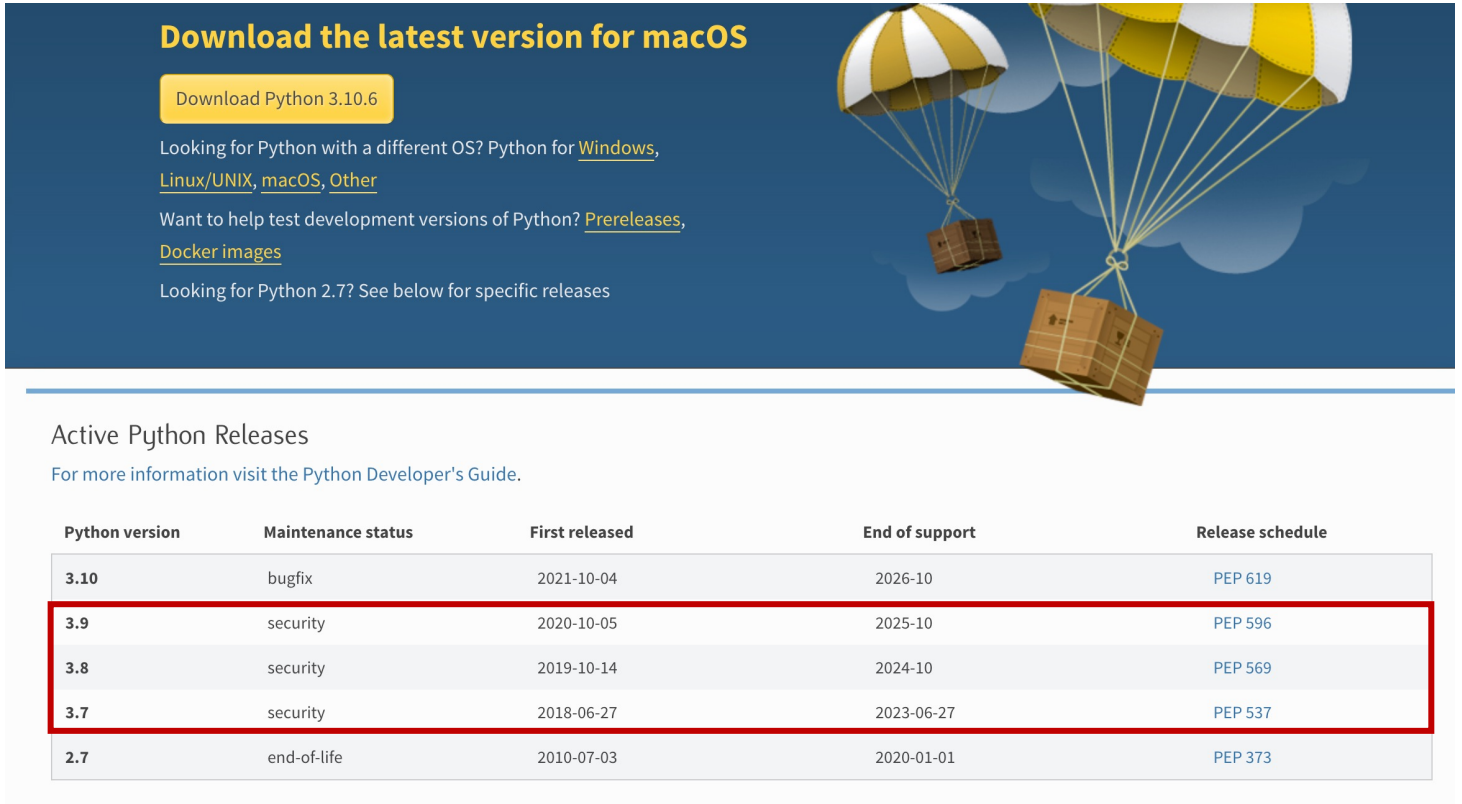
Installation of Python3

- **What and Why Python?**
  - **Easy to use, Great Functionality, Generalization**
    - A real programming language (vs. shell)
    - More error checking (vs. C)
    - very-high-level language (high level data types built in)
    - Adapt to Larger problem domain (vs. Awk, Perl)
  - **Modules** (reusable, modulization)
  - **Interpreted language** (no compilation, no linking, interactively interpreter)
  - **Extensible**
- **Why Python3?**
  - Python2 is not maintained anymore

https://bugs.python.org/file47781/Tutorial_EDIT.pdf

Installation of Python3

- How to install Python3?
  - https://www.python.org/downloads/



Make sure you Have python3 Installed on your PC

Let's Set up the 'environment' for this course

- Installation of Python3
- Virtual Environment (VENV)
- Installation/Intro Pip
- Installation/Intro of Conda
- Installation/Intro VSCode
- General recap (from IBI1😀) of Python3 programming
- The PTA system

Virtual Environment

- **What is VENV?**
  - Virtual Environment are created to provide project level isolation of packages installed to one Python interpreter.
- **Why VENV?**

  "The short answer is that Python isn't great at dependency management."
  - **Avoid System Pollution**
    - Linux and macOS come preinstalled with a version of Python that the operating system uses for internal tasks. If you install packages to your operating system's global Python, these packages will mix with the system-relevant packages.
  - **Sidestep dependency conflicts**
    - One of your projects might require a different version of an external library than another one. If you have only one place to install packages, then you can't work with two different versions of the same library.
  - **Minimize Reproducibility Issues**
    - Can easily share all the dependencies information with others
  - **Dodge installation privilege lockouts**
    - In a corporate work environment, you most likely won't have that level of access to the machine that you're working on. If you use virtual environments, then you create a new installation location within the scope of your user privileges, which allows you to install and work with external packages.

https://realpython.com/python-virtual-environments-a-primer/#how-can-you-work-with-a-python-virtual-environment

Use Virtual Environment

- **Create It**

Any time you're working on a Python project that uses external dependencies that you're installing with pip, it's best to first create a virtual environment:

Windows PowerShell

```
PS> python -m venv venv
```

macOS

```
$ python3 -m venv venv
```

```
# Which creates a virtual environment directory at venv
```

- **Activate it**

Windows PowerShell

```
source .\Scripts\Activate.ps1
```

macOS

```
$source venv/bin/activate
```

```
# Which activate this environment
```

- **Deactivate it**

Windows PowerShell

```
deactivate
```

macOS

```
deactivate
```


https://realpython.com/python-virtual-environments-a-primer/#how-can-you-work-with-a-python-virtual-environment

Task: Set up virtual environment for 2022bmi3

```
$python3 -m venv venv --prompt="2022BMI3"
$source venv/bin/activate
$python -m pip list
 Package    Version
 ---------- -------
 pip        19.0.3
 setuptools 40.8.0
```

Hold on …

Let's Set up the 'environment' for this course

- Installation of Python3
- Virtual Environment (VENV)
- Installation/Intro Pip
- Installation/Intro of Conda
- Installation/Intro VSCode
- General recap (from IBI1😀) of Python3 programming
- The PTA system

Introduction of PIP

- **Why PIP?**
  - PIP is a package manager for Python packages, or modules if you like.
- **What is a Package?**
  - A package contains all the files you need for a module
  - Modules are python code libraries you can include in your project.
- **Do you need to install pip?**
  - pip is already installed if you're using Python 2 >=2.7.9 or Python 3 >=3.4 binaries downloaded from python.org, but you'll need to upgrade pip.
  - Additionally, pip will already be installed if you're working in a Virtual Environment created by virtualenv or pyvenv

https://www.w3schools.com/python/python_pip.asp
https://readthedocs.org/projects/pip-python3/downloads/pdf/stable/

Installing PIP

- **If you don't have PIP installed, Installing with get-pip.py**
  - Step1: securely download get-pip.py

    ```
    wget https://bootstrap.pypa.io/get-pip.py
    ```
  - Step2: execute get-pip.py

    ```
    Python get-pip.py
    ```

For more info, refer to Chp.2.2 *https://readthedocs.org/projects/pip-python3/downloads/pdf/stable/*

Installing Packages using PIP

- **Install a package from PyPI**

```
$pip install SomePackage [someversion]
 [...]
 Successfully installed SomePackage
```

- **show what fiels were installed**

```
$pip show --files SomePackage
 Name: SomePackage
 Version: 1.0
 Location: /my/env/lib/pythonx.x/site-packages
 Files: ../somepackage/__init__.py
 [...]
```

```
$pip install pandas==1.4.3
#Pandas with 1.4.3 version
$pip install pandas>=1.4.3
#Pandas with latter or equal than 1.4.3
$pip install pandas~=1.4.3
#any Padas compatible with 1.4.3
```

https://readthedocs.org/projects/pip-python3/downloads/pdf/stable/

Installing Packages using PIP

- **List what packages are outdated:**

```
$pip list --outdated
 SomePackage (Current: 1.0 Latest: 2.0)
```

- **Upgrade a package:**

```
$pip install --upgrade SomePackage
 [...]
 Found existing installation: SomePackage 1.0
 Uninstalling SomePackage:
    Successfully uninstalled SomePackage
 Running setup.py install for SomePackage
 Successfully installed SomePackage
```

- **Uninstall a package:**

```
$pip uninstall SomePackage
 Uninstalling SomePackage:
   /my/env/lib/pythonx.x/site-packages/somepackage
 Proceed (y/n)? y
 Successfully uninstalled SomePackage
```


https://readthedocs.org/projects/pip-python3/downloads/pdf/stable/

Task: install packages @ VENV ‘2022bmi3’

```
$python3 -m venv venv --prompt="2022BMI3"
$source venv/bin/activate
$python -m pip list
 Package    Version
 ---------- -------
 pip        19.0.3
 setuptools 40.8.0
$deactivate

$python3 -m venv venv --prompt="2022BMI3"
$source venv/bin/activate
$python3 -m pip install numpy
$python -m pip list
 Package    Version
 ---------- --------
 numpy      1.21.6
 pip        19.0.3
 setuptools 40.8.0
```

Then deactivate and reactivate the ‘2022BMI3’ venv, see whether numpy packages is still there (10 mins)

Let's Set up the 'environment' for this course

- Installation of Python3
- Virtual Environment (VENV)
- Installation/Intro Pip
- Installation/Intro of Conda
- Installation/Intro VSCode
- General recap (from IBI1😀) of Python3 programming
- The PTA system

Introduction of Conda

- **What is Conda?**
  - "Package, dependency and environment management for any language—Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, Fortran, and more." (most popular in python!)
- **Why Conda?**
  - Similar to the rationale for venv, but more:
    - Supports environments of different Python intepreter versions
    - Can management of packages written in C, C++, FORTRAN, R, etc..
    - Uses binary packages, meaning a working C/C++/FORTRAN compilers not needed
- **Concepts added by Conda**
  - **Environment**: is something like a virtual environment
  - **Channels**: are online package repositories organized by topics
  - **Distribution**: is a collection of Conda itself and a base environment with some packages

https://docs.conda.io/en/latest/
https://github.com/conda/conda

AnaConda, Miniconda, Conda-Forge or Bioconda

- **Conda**
  - Is a general-purposed user-level package management system which supports environments
- **Anaconda**
  - Anaconda Inc. is a company which provides Conda-related services
  - anaconda.org, a Conda package hosting;
  - `Anaconda` channel, a Conda channel;
  - `Anaconda` distribution, including Conda and lots of packages installed in `base` environment.
- **Miniconda**
  - Is a Conda distribution with minimal packages installed in `base` environment
- **Conda Forge**
  - Is a general-purposed Conda channel
- **BioConda**
  - Is another bioinformatics-related Conda channel

Installing Conda

https://docs.conda.io/en/latest/miniconda.html#linux-installers

Note: These Miniconda installers contain the conda package manager and **Python**.

Latest Miniconda Installer Links

Latest - Conda 4.12.0 Python 3.9.7 released February 15, 2022

Platform	Name	SHA256 hash
Windows	Miniconda3 Windows 64-bit	1acbc2e8277ddd54a5f724896c7edee112d068529588d944702966c867e7e9cc
	Miniconda3 Windows 32-bit	4fb64e6c9c28b88beab16994bfba4829110ea3145baa60bda5344174ab65d462
macOS	Miniconda3 macOS Intel x86 64-bit bash	007bae6f18dc7b6f2ca6209b5a0c9bd2f283154152f82becf787aac709a51633
	Miniconda3 macOS Intel x86 64-bit pkg	cb56184637711685b08f6eba9532cef6985ed7007b38e789613d5dd3f94ccc6b
	Miniconda3 macOS Apple M1 64-bit bash	4bd112168cc33f8a4a60d3ef7e72b52a85972d588cd065be803eb21d73b625ef
	Miniconda3 macOS Apple M1 64-bit pkg	0cb5165ca751e827d91a4ae6823bfda24d22c398a0b3b01213e57377a2c54226
Linux	Miniconda3 Linux 64-bit	78f39f9bae971ec1ae7969f0516017f2413f17796670f7040725dd83fcff5689
	Miniconda3 Linux-aarch64 64-bit	5f4f865812101fdc747cea5b820806f678bb50fe0a61f19dc8aa369c52c4e513
	Miniconda3 Linux-ppc64le 64-bit	1fe3305d0ccc9e55b336b051ae12d82f33af408af4b560625674fa7ad915102b
	Miniconda3 Linux-s390x 64-bit	ff6fdad3068ab5b15939c6f422ac329fa005d56ee0876c985e22e622d930e424

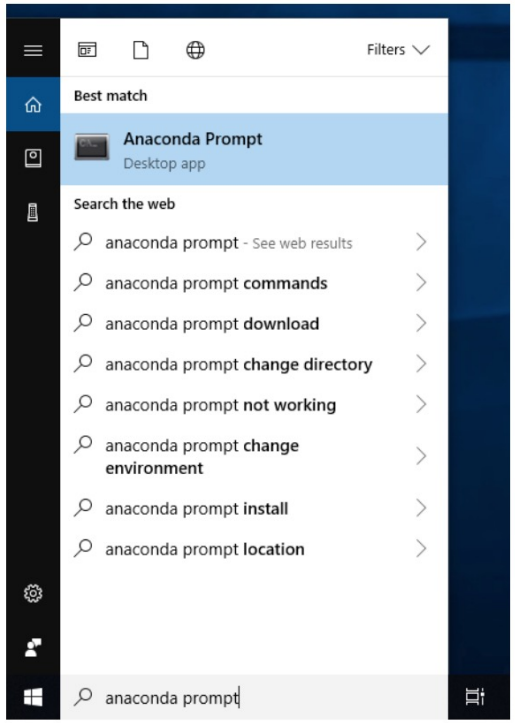
In class task:
Install miniconda for Python3.8

Starting & Managing Conda

Starting conda

Windows

- From the Start menu, search for and open "Anaconda Prompt."



MacOS

- Open Launchpad, then click the terminal icon.

On macOS, all commands below are typed into the terminal window.

Linux

- Open a terminal window.

On Linux, all commands below are typed into the terminal window.

- **Verify conda**

```
$conda --version
 conda 4.13.0
```

- **Update conda**

```
$conda update conda
 Proceed ([y]/n)? y
```

https://conda.io/projects/conda/en/latest/user-guide/getting-started.html#starting-conda

Managing Environments/Python

- **Create a new environment and install a package in it**

```
$ conda create --name miniconda2022BMI3 numpy
  Proceed ([y]/n)? y
$ conda create --name miniconda2022BMI3 python=3.9
                                  #specify python version
```

- **Activate conda**

```
$ conda activate miniconda2022BMI3
 (miniconda2022BMI3) wanlus-MacBook-Air:~ wanluliu$
```

#you may need to conda init bash before you can activate

- **List environment**

```
$ conda info --envs
```

```
# conda environments:
#
                         /Users/wanluliu/Library/r-miniconda
                         /Users/wanluliu/Library/r-miniconda/envs/r-reticulate
base                     /Users/wanluliu/opt/miniconda3
miniconda2022BMI3     *  /Users/wanluliu/opt/miniconda3/envs/miniconda2022BMI3
```

- **Change current environment back to default (base)**

```
$ conda activate
 (base) wanlus-MacBook-Air:~ wanluliu$
```


https://conda.io/projects/conda/en/latest/user-guide/getting-started.html#starting-conda

Managing Packages

- **First activate the environment**
- **Then search for packages**

```
$ conda search numpy
```

- **Install package into the current environment**

```
$ conda install scipy
```

- **List packages**

```
$ conda list
```

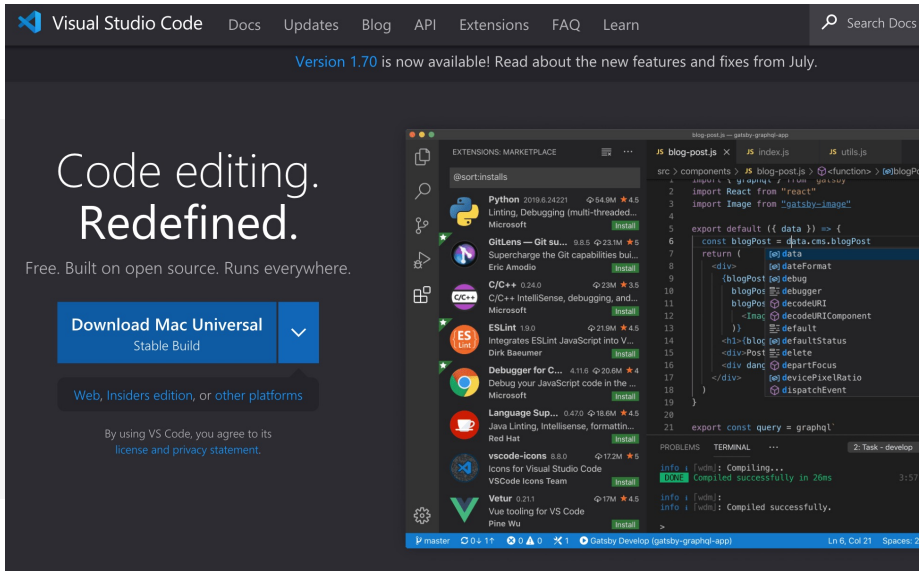
https://conda.io/projects/conda/en/latest/user-guide/getting-started.html#starting-conda

Let's Set up the 'environment' for this course

- Installation of Python3
- Virtual Environment (VENV)
- Installation/Intro Pip
- Installation/Intro of Conda
- Installation/Intro VSCode
- General recap (from IBI1😀) of Python3 programming
- The PTA system

Installation/Intro of VS Code

- **What is VS code?**
  - A open source code editors for various languages
  - https://code.visualstudio.com/



- **Why VS code?**
  - Easy to use
  - Have many extensions, useful for testing, debugging etc.

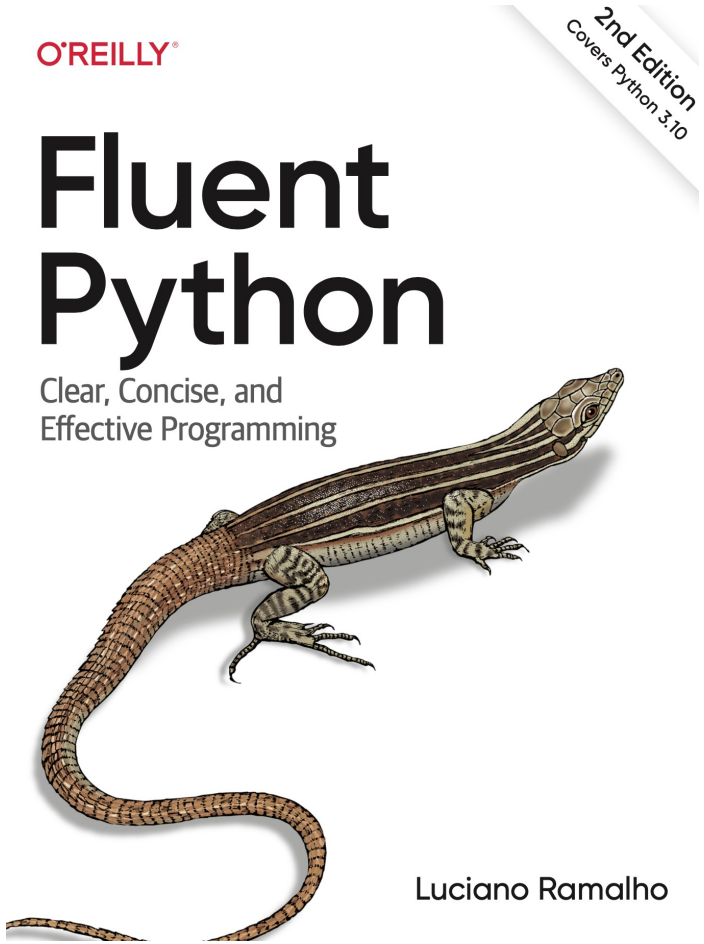
Homework task:
If you haven't install VS code, and python extension, follow this tutorial to set up python on VS code
https://code.visualstudio.com/docs/python/python-tutorial

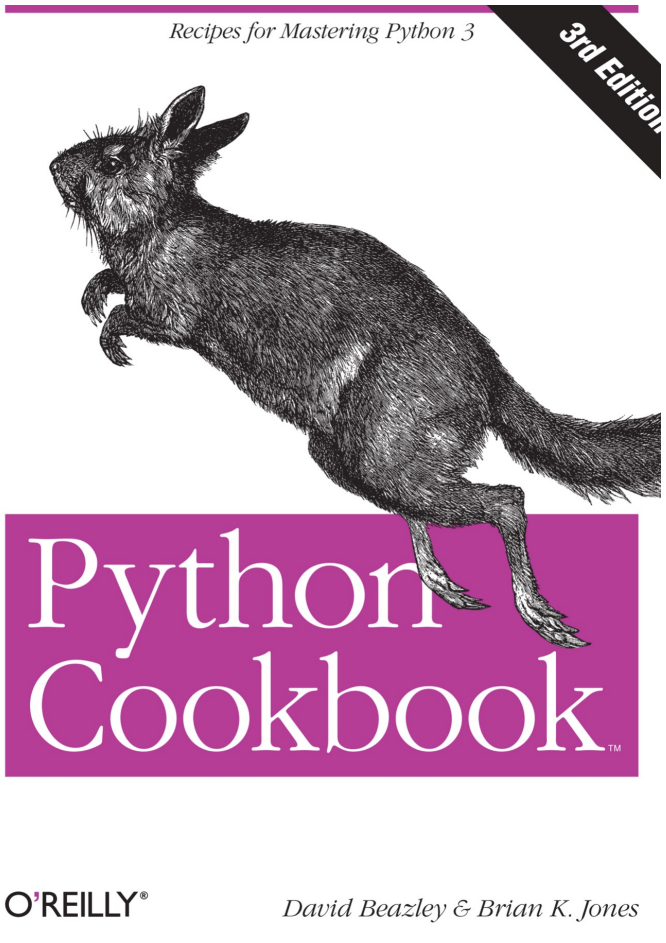
Let's Set up the 'environment' for this course

- Installation of Python3
- Virtual Environment (VENV)
- Installation/Intro Pip
- Installation/Intro of Conda
- Installation/Intro VSCode
- General recap (from IBI1😀) of Python3 programming
- The PTA system

Learn programming in Python

Books with good python snippets





Github….
Google…
Other online sources…

E book can be found on course website: https://labw.org/bmi3-2022/textbooks
usename: zje pass: 85c701e0

This part is largely adapted from Ziwei Xue (ZJE), and Kranthi Varala (Purdue U)

Variables, Objects, Classes

Variables

- Variables are the basic unit of storage for a program.
- Variables can be created and destroyed.
- At a hardware level, a variable is a reference to a location in memory.
- Programs perform operations on variables and alter or fill in their values.

Objects

- Objects are higher level constructs that include one or more variables and the set of operations that work on these variables.
- An object can therefore be considered a more complex variable.

Classes vs Objects

- Every Object belongs to a certain class.
- Classes are abstract descriptions of the structure and functions of an object.
- Objects are created when an instance of the class is created by the program.
- For example, “Fruit” is a class while an “Apple” is an object.
- Every variable you create is either a built-in data type object OR a new class you created.

Data types in Python

These are fundamental data types in Python

Text Type	`str`
Numeric	`int`, `float`
Sequence	`list`, `tuple`
Mapping	`dict`
Set Types	`set`
Boolean	`bool`
Binary	`bytes`, `bytearray`, `memoryview`
None Type	`NoneType`

`str.upper()`

`set.intersection()`

……

You can do most of work with these built-in classes

Data types in Python: Numbers

Integers, float

- Simple assignment creates an object of number type such as:

```
>>> b=2
>>> type(b)
<class 'int'>

>>> c=2.0
>>> type(c)
<class 'float'>

>>> d=2.0123
>>> type(d)
<class 'float'>
```

- Supports simple to complex arithmetic operators;

```
>>> e=d/b
>>> e
1.00615

>>> type(d)
<class 'float'>

>>> type(b)
<class 'int'>

>>> type(e)
<class 'float'>
```

Try in your local python3 2 mins

`dir(object)` *can list the functions available for certain object.*

Data types in Python: Strings

- A string object is a ‘sequence’, i.e., it’s a list of items where each item has a defined position.
- Each character in the string can be referred, retrieved and modified by using its position.
- This order id called the ‘index’ and always starts with 0.

Try in your local python3 2 mins

```
>>> s="2022BMI3"
>>> type(s)
<class 'str'>
>>> len(s)
8
>>> s[0]
'2'
>>> s[4]
'B'
>>> s[9]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
```

```
>>> s[-1]
'3'
>>> s[3:]
'2BMI3'
>>> s[2:6]
'22BM'
```

```
>>> s+'Cool'
'2022BMI3Cool'
>>> s+' Cool'
'2022BMI3 Cool'
>>> s*3
'2022BMI32022BMI32022BMI3'
>>> s+' Cool'*3
'2022BMI3 Cool Cool Cool'
>>> (s+' Cool')*3
'2022BMI3 Cool2022BMI3 Cool2022BMI3 Cool'
```

Data types in Python: Lists & Tuples

- List is a more general sequence object that allows the individual items to be of different types.
- Equivalent to **arrays** in other languages.
- Lists have no fixed size and can be expanded or contracted as needed.
- Items in list can be retrieved using the index.
- Lists can be nested just like arrays, i.e., you can have a list of lists.

Try in your local python3 2 mins

```
>>> l=[123,3.14,'2022BMI3']
>>> type(l)
<class 'list'>
>>> l
[123, 3.14, '2022BMI3']
>>> l[0]
123
```

Tuples:

```
>>> t=(123,3.14,'2022BMI3')
>>> type(t)
<class 'tuple'>
```

Nested list:

```
>>> nestedList=[[1,2,3],[4,5,6],[7,8,9]]
>>> type(nestedList)
<class 'list'>
>>> nestedList
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> nestedList[1][1]
5
```

- Tuples are immutable lists
- Maintain integrity of data during program execution.

Data types in Python: Dictionaries

- Dictionaries are unordered mappings of ’Name : Value’ associations.
- Comparable to hashes and associative arrays in other languages.
- Intended to approximate how humans remember associations.

```
>>> Dict={'name':'apple','color':'red','taste':'sweet','number':'5'}
>>> Dict['name']
'apple'
>>> type(Dict)
<class 'dict'>
```

Try in your local python3 2 mins

Data types in Python: Files

- File objects are built for interacting with files on the system.
- Same object used for any file type.
- User has to interpret file content and maintain integrity.

```
>>> file=open('test.txt','w')
>>> file.write('Hello\t')
6
>>> file.write('World!\n')
7
>>> file.close()
>>> file=open('test.txt')
>>> text=file.read()
>>> text
'Hello\tWorld!\n'
>>> print(text)
Hello World!
```

Mutable vs. Immutable

- Numbers, strings and tuples are immutable i.e. cannot be directly changed
- Lists, dictionaries and sets can be changed in place

```
>>> s
'2022BMI3'
>>> s[0]
'2'
>>> s[0]='3'
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>
TypeError: 'str' object does not
support item assignment
```

```
>>> l
[123, 3.14, '2022BMI3']
>>> l[1]=321
>>> l
[123, 321, '2022BMI3']
```

```
>>> t
(123, 3.14, '2022BMI3')
>>> t[1]
3.14
>>> t[1]=321
Traceback (most recent call last):
  File "<stdin>", line 1, in
<module>
TypeError: 'tuple' object does not
support item assignment
```

Try in your local python3 2 mins

Sets

Sets are used to store multiple items in a single variable.
A set is a collection which is *unordered*, *unchangeable**, and *unindexed*.

- Special data type introduced since Python 2.4 onwards to support mathematical set theory operations.
- Unordered collection of unique items.
- Set itself is mutable, BUT every item in the set has to be an immutable type.
- So, sets can have numbers, strings and tuples as items but cannot have lists or dictionaries as items.

```
>>> thisset={"apple","banana","cherry"}
>>> type(thisset)
<class 'set'>
>>> set[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'type' object is not subscriptable
>>> print(thisset)
{'cherry', 'banana', 'apple'}
```

https://www.w3schools.com/python/python_sets.asp

Python programming Styles

Functions:

```
def divide(a: float, b: int) -> float:
    """Divide two numbers

    Divide a float number by an integer number

    Args:
        a: A float number
        b: An integer number

    Returns:
        A float number

    Raises:
        ValueError: An error occurred when the denomiator is zero
    """
    if b == 0:
        raise ValueError("The denominator is zero")
    return a / b
```

Type hints

Function briefs

Function args

Public returns

Function implementations

Python programming Styles

Functions:

7.1. Writing Functions That Accept Any Number of Arguments

Problem

You want to write a function that accepts any number of input arguments.

Solution

To write a function that accepts any number of positional arguments, use a * argument.
For example:

```
def avg(first, *rest):
    return (first + sum(rest)) / (1 + len(rest))

# Sample use
avg(1, 2)          # 1.5
avg(1, 2, 3, 4)    # 2.5
```

Python programming Styles: Functions

7.2. Writing Functions That Only Accept Keyword Arguments

Problem

You want a function to only accept certain arguments by keyword.

Solution

This feature is easy to implement if you place the keyword arguments after a * argument or a single unnamed *. For example:

```
def recv(maxsize, *, block):
    'Receives a message'
    pass

recv(1024, True)        #  TypeError
recv(1024, block=True)  # Ok
```

Python programming Styles: Classes

```
class SampleClass(object):
    """Summary of class here.

    Longer class information....
    Longer class information....

    Attributes:
        likes_spam: A boolean indicating if we like SPAM or not.
        eggs: An integer count of the eggs we have laid.
    """

    def __init__(self, likes_spam=False):
        """Inits SampleClass with blah."""
        self.likes_spam = likes_spam
        self.eggs = 0

    def public_method(self, b:int):
        """Performs operation blah."""
          return self.eggs + b

    def __private_method(self):
        """Performs operation blah."""
```

Class description

Class attributes

Class initialization

Public method

Private

Python programming Styles: Modules and Packages

Problem

You want to organize your code into a package consisting of a hierarchical collection of modules.

Solution

Making a package structure is simple. Just organize your code as you wish on the file-system and make sure that every directory defines an *__init__.py* file. For example:

```
graphics/
    __init__.py
    primitive/
        __init__.py
        line.py
        fill.py
        text.py
    formats/
        __init__.py
        png.py
        jpg.py
```

```
import graphics.primitive.line
from graphics.primitive import line
import graphics.formats.jpg as jpg
```

Let's Set up the 'environment' for this course

- Installation of Python3
- Virtual Environment (VENV)
- Installation/Intro Pip
- Installation/Intro of Conda
- Installation/Intro VSCode
- General recap (from IBI1😀) of Python3 programming
- The PTA system

PTA

- **What is PTA?**
  - Programming teaching assistant (online programming judgement system)
  - https://pintia.cn/
  - You should all have PTA account from your DST2😉
- **In this course, we will use PTA for:**
  - In course programming challenges
  - Weekly homework challenges
  - Code4Fun competition

Task: in class PTA practice - data types

2022 BMI3 Week 1.1 – Session 1 – Python data types

</> 编程题 6

标号	标题	分数
7-1	Get Input	10
7-2	Numeric Type Conversion	10
7-3	Upper Case	10
7-4	Split Input	10
7-5	List Intersection	10
7-6	Count with a Dictionary	10

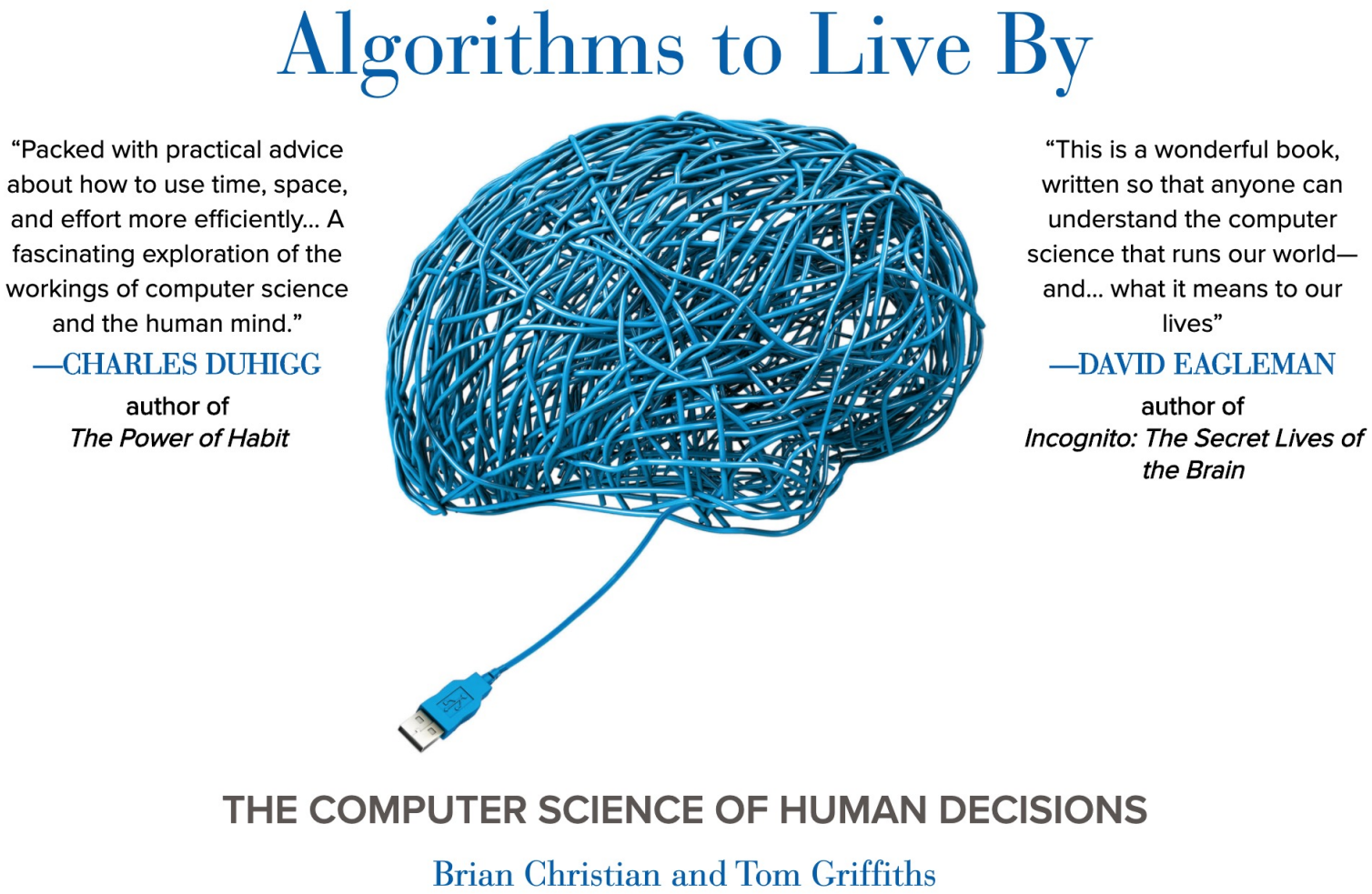
20 mins

Learning objectives for this lecture

- Describe the principles of an algorithm
- Define and apply pseudocode for algorithms
- Distinguish Iterative vs. Recursive algorithms

Algorithms

- Algorithm: a sequence of instructions used to solve a well-formulated problem.
- The problem need to be specified with inputs and outputs.
- A well-formulated problem is unambiguous and precise, leaving no room for misinterpretation.



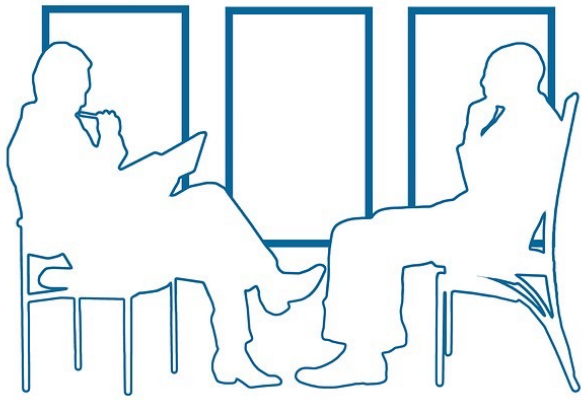
The dating problem

How many people should you date before truly finding the “one” or deciding to settle down with?

- It’s a tricky question, and as with many tricky questions, math has an answer of some sort, which tells you : Its **37%** of the way through your search!

The Secretary Problem – how to well-formulated the problem

- Famous puzzle in mathematics known as '**The Secretary Problem**'.
- The strategy is, say you're interviewing a group of applicants for a position, **how do you maximize the chances of hiring the single best applicant in the pool?**
- This problem became popular when it was first published in a scientific journal (Scientific American) in 1960.



Problem setting:

The process is a manager interviewing applicants for the position of a secretary.

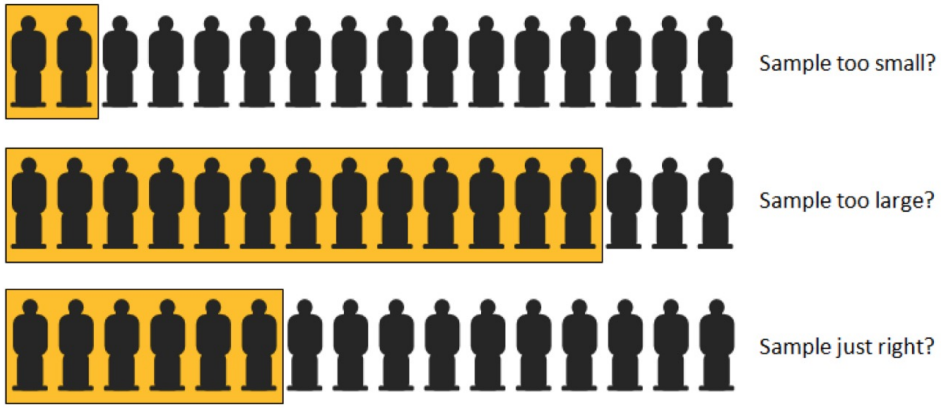
1. The applicants will be interviewed in random order, one at a time.
2. A decision about each applicant is to be made immediately following each interview.
3. If rejected, the applicant leaves the room after the interview, and he or she is not recalled again.
4. Any applicant will accept the job if asked, effectively ending the search.

Optimal Stopping

- The crucial problem here is not who would you choose but **how many options would you consider before choosing.**
- What if you choose too early and miss the best applicant that follows?
- What if you stop looking too late and the best applicant already left?
- At what point do you stop and select the best candidate?
- When do you stop and just make the decision?

The problem has an elegant solution using a method called Optimal Stopping.

The answer with the highest probability of success is to **reject the first 37% of applicants**, then when the next best applicant comes along and found better than the first 37%, you stop and hire! This gives you the best chance of ending the interview with the best candidate.



Why 37%? What's the algorithm behind? You can figure it out if you are interested in it!

Algorithm (Recipe) to make a pumpkin pie

- 1.5 cups canned or cooked pumpkin
- 1 cup brown sugar, firmly packed
- 0.5 teaspoon salt
- 2 teaspoon cinnamon
- 1 teaspoon ginger
- 2 tablespoons molasses
- 3 eggs, slightly beaten
- 12 ounces can of evaporated milk
- 1 unbaked pie crust

- Combine pumpkin, sugar, salt, ginger, cinnamon, and molasses. Add eggs and milk and mix thoroughly . Pour into unbaked pie crust and bake in hot oven (425 degrees Fahrenheit) for 40 – 45 minutes, or until knife inserted comes out clean.

Algorithm to make a pumpkin pie

pseudocode

input

```
MIXFILLING(pumpkin, sugar, salt, spices, eggs, milk)
1       bowl ← Get a bowl from cupboard
2       PUT(pumpkin, bowl)
3       PUT(sugar, bowl)
4       PUT(salt, bowl)
5       PUT(spices, bowl)
6       STIR(bowl)
7       PUT(eggs, bowl)
8       PUT(milk, bowl)
9       STIR(bowl)
10      filling ← Contents of bowl
11      return filling
```

variable

Assign value to variable

output

```
MAKEPUMPKINPIE(pumpkin, sugar, salt, spices, eggs, milk, crust)
1       PREHEATOVEN(425)
2       filling ← MIXFILLING(pumpkin,sugar,salt,spices,eggs,milk)
3       pie ← ASSEMBLE(crust,filling)
4       while knife inserted does not come out clean
5               BAKE(pie)
6       output "Pumpkin pie is complete"
7       return pie
```

subroutine

pseudocode

Assignment

Format: $a \leftarrow b$

Effect: Sets the variable a to the value b.

Example: $b \leftarrow 2$
$a \leftarrow b$

Result: The value of a is 2

Arithmetic

Format: $a+b$, $a-b$, $a \cdot b$, a/b, a^b

Effect: Addition, subtraction, multiplication, division, and exponentiation of numbers.

Example: DIST(x_1,y_1,x_2,y_2)

1 $dx \leftarrow (x_2 - x_1)^2$
2 $dy \leftarrow (y_2 - y_1)^2$
3 **return** $\sqrt{(dx + dy)}$

Result: DIST(x_1, y_1,x_2, y_2) computes the Euclidean distance between points with coordinates (x_1, y_1) and (x_2, y_2).

DIST(0, 0, 3, 4) =

Conditional

Format: **if** A is true
B
else
C

Effect: If statement A is true, executes instructions B, otherwise executes instructions C. Sometimes we will omit "else C," in which case this will either execute B or not, depending on whether A is true.

Example: MAX(a,b)

```
1   if a < b
2       return b
3   else
4       return a
```

Result: MAX(a,b) computes the maximum of the numbers a and b. For example, MAX(1, 99) returns 99.

pseudocode

For loops

Format: for $i \leftarrow a$ **to** b

B

Effect: Sets i to a and executes instructions **B**. Sets i to a + 1 and executes instructions **B** again. Repeats for i = a + 2, a + 3, . . . , b − 1, b.

Example: SUMINTEGERS(n)

```
1   sum ← 0
2   for i ← 1 to n
3         sum ← sum + i
4   return sum
```

Results: SUMINTEGERS(n) computes the sum of integers from 1 to n. SUMINTEGERS(10) returns $1 + 2 + \cdots + 10 = 55$.

While loops

Format: **while** A is true

B

Effect: Checks the condition A. If it is true, then executes instructions B. Checks A again; if it's true, it executes B again. Repeats until A is not true.

Example: ADDUNTIL(b)

```
1  i ← 1
2  total ← i
3  while total ≤ b
4        i ← i+1
5        total ← total + i
6  return i
```

ADDUNTIL(25) =

pseudocode

Array access

Format: a_i

Effect: The ith number of array a = $(a_1, \ldots a_i, \ldots a_n)$. For example, if F=(1,1,2,3,5,8,13), then F_3 =2,andF_4 =3.

Example: FIBONACCI(n)

```
1  F_1 ← 1
2  F_2 ← 1
3  for i ← 3 to n
4        F_i ← F_{i−1} + F_{i−2}
5  return F_n
```

Example: FIBONACCI(n) computes the nth Fibonacci number.

FIBONACCI(8) =

Task: in class PTA practice – Pseudocode & python basics

2022 BMI3 Week 1.1 – Session 2 – Pseudocode & Python basics

</> 编程题 5

标号	标题	分数
7-1	Dist	10
7-2	Max	10
7-3	Sum Integers	20
7-4	Add Until	20
7-5	Fibonacci	20

30 mins

Pseudocode – in class practice

Write pseudo code that reads in three numbers and writes them all in sorted order.

In class PTA practice:
Write the pseudocode,
and implement on PTA

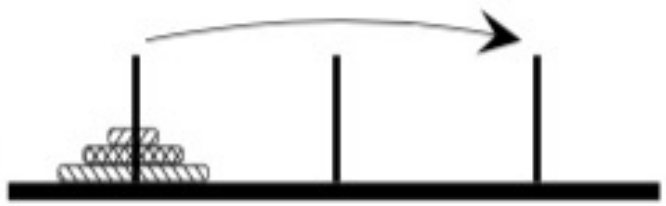
2022 BMI3 Week 1.1 - Session 3 - Sort Number Pseudocode & Python implementation

编程题 1

标号	标题	分数	提交通过率
7-1	Sort Num	30	0/0(0.00%)

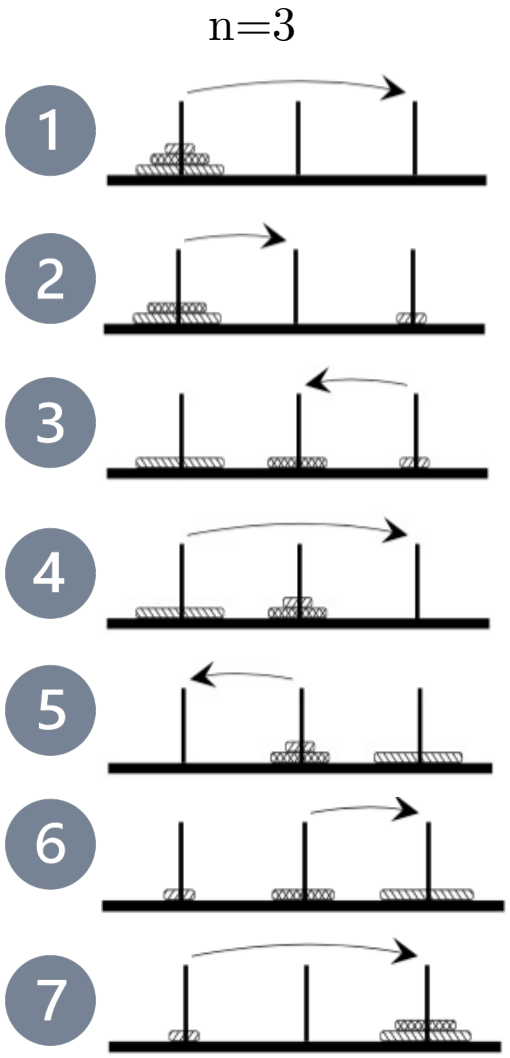
Recursive algorithms – Towers of Hanoi puzzle

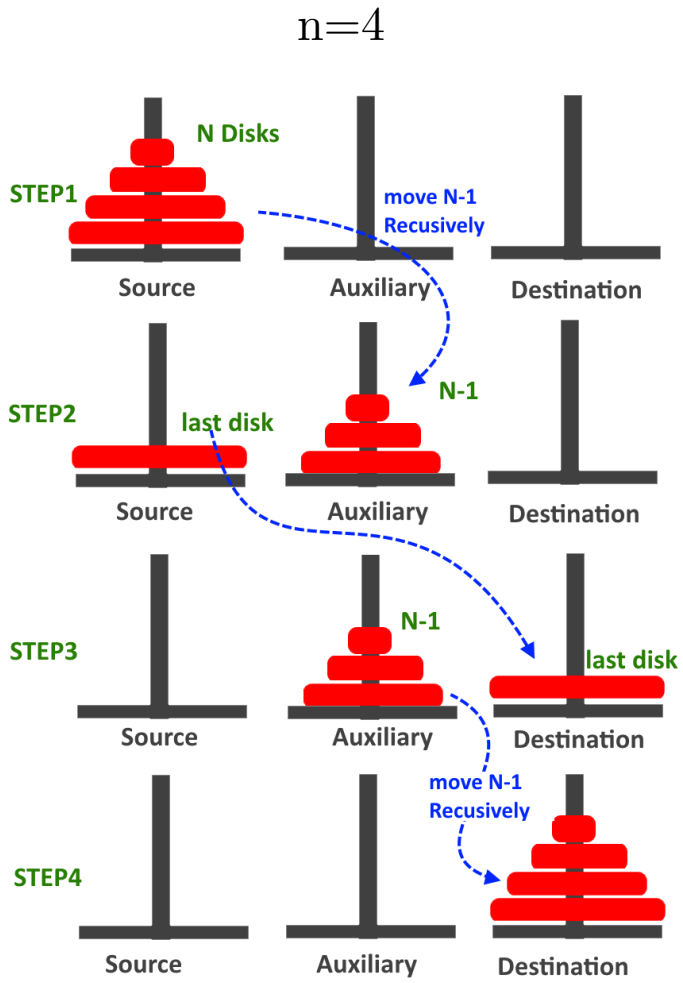
Move disk from peg 1 to peg 3



Tower of Hanoi problem
Output a list of moves that solves the Tower of Hanoi.

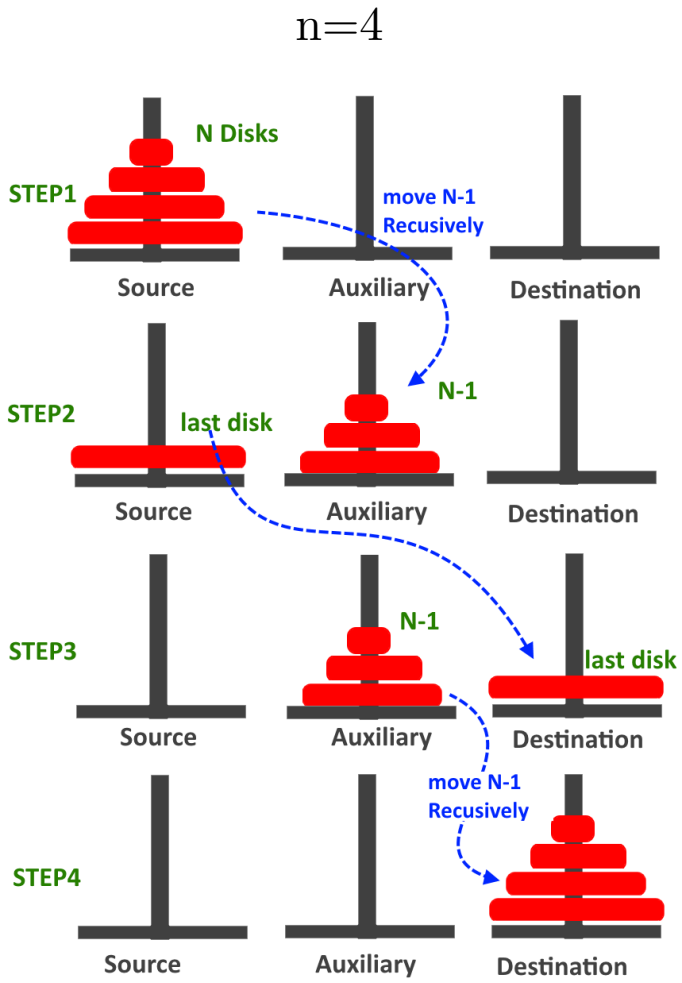
Input: An integer *n*.
Output: A sequence of moves that will solve the *n*-disk Towers of Hanoi puzzle.





Moves: 7+1+7

Recursive algorithms – Towers of Hanoi puzzle

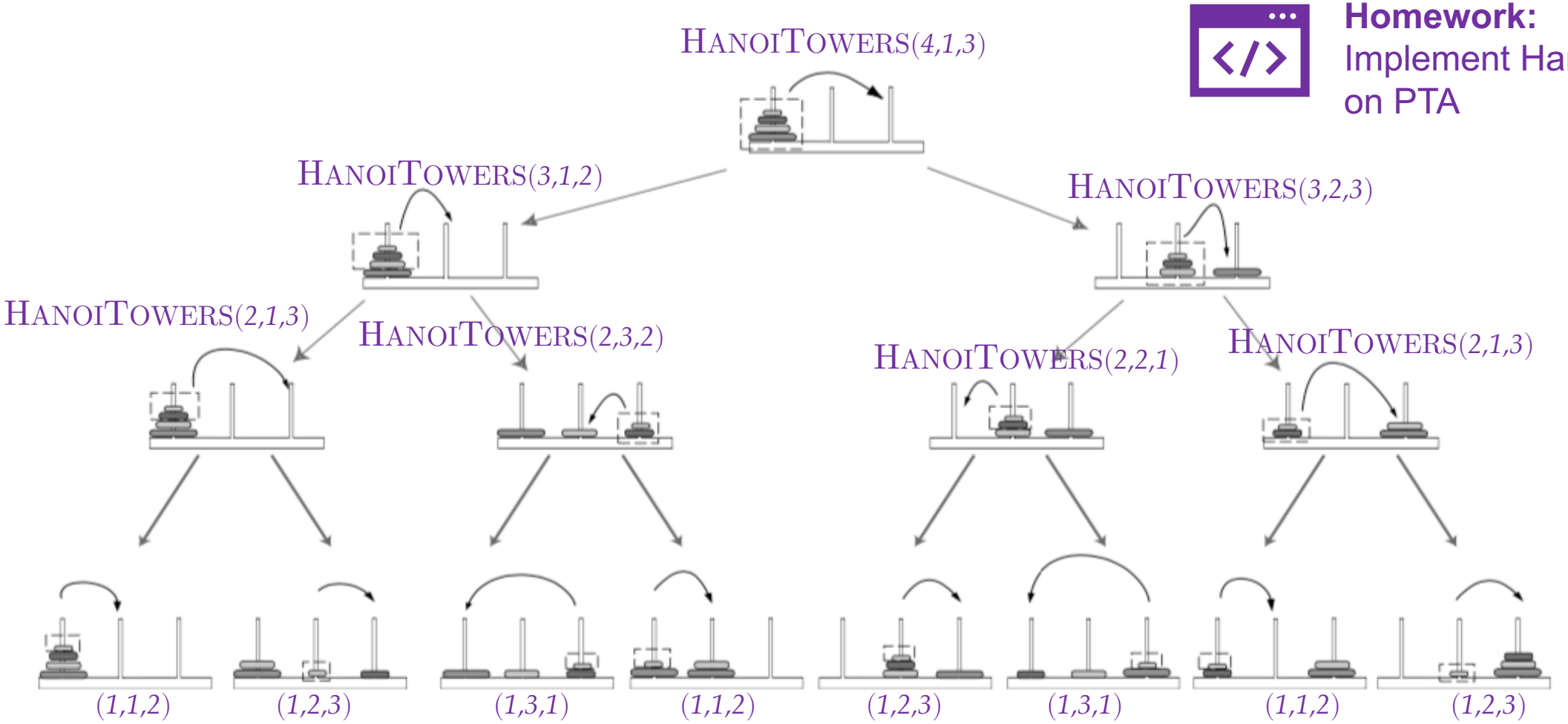


Moves: 7+1+7

Pseudocode *fromPeg/toPeg/unusedPeg=1,2,or3*

```
HANOITOWERS(n, fromPeg, toPeg)
1  if n=1
2    output "Move disk from peg fromPeg to peg toPeg"
3    return
4  unusedPeg ← 6 – fromPeg – toPeg
5  HANOITOWERS(n-1, fromPeg, unusedPeg)
6  Output "Move disk from peg fromPeg to peg toPeg"
7  HANOITOWERS(n-1, unusedPeg, toPeg)
8  return
```

The recursion tree for HANOITOWERS



Homework:
Implement HanoiTower() on PTA

Complexity could be really high for a n=100 disk Hanoi Tower

Task: homework on PTA – HanoiTower()

2022 BMI3 Week 1 - Homework

编辑试卷

</> 编程题 2

标号	标题	分数	提交通过率
7-1	Hanoi Tower	40	0/0(0.00%)
7-2	Selection Sort	60	0/0(0.00%)

Iterative vs. Recursive algorithms

Iterative ⟷ **Recursive**

```
FIBONACCI(n)
1   F_1 ← 1
2   F_2 ← 1
3   for i ← 3 to n
4           F_i ← F_i−1 + F_i−2
5   return F_n
```

```
RECURSIVEFIBONACCI(n)
1   if n=1 or n=2
2      return 1
3   else
4      a ← RECURSIVEFIBONACCI(n-1)
5      b ← RECURSIVEFIBONACCI(n-2)
6      return a+b
```

- Iteration can be used in place of recursion
  - An iterative algorithm uses a **looping construct**
  - A recursive algorithm uses a **branching structure**
- Recursive solutions are often less efficient
  - In terms of both **time** and **space**
- Recursion may simplify the solution
  - **Shorter**, more easily understood source code

Summary

- Review basic concepts in python3
  - **VENV, PIP, conda, VS code**
- Recall python programming basics
  - **Variables, objects, classes, basic data types, functions …**
- Describe the principles of an algorithm
  - **Input, output, well-formulated problem**
- Define and apply pseudocode for algorithms
  - **Assignment, Arithmetic, Conditional, For loops, while loops, array access**
- Distinguish Iterative vs. Recursive algorithms

Homework & Code4Fun

Top 3 on the code for fun ranking board will get a small gift from wanlu ☺

2022 BMI3 Week 1 - Homework

编辑试卷

编程题 2

标号	标题	分数	提交通过率
7-1	Hanoi Tower	40	0/0(0.00%)
7-2	Selection Sort	60	0/0(0.00%)

2022 Biomedical Informatics 3 - Code for Fun

编辑试卷

编程题 8

	标号	标题	分数	提交通过率
✓	7-1	Complementary Sequence	100	0/1(0.00%)
	7-2	Prime Numbers	100	0/0(0.00%)
	7-3	Text Reversed	100	0/0(0.00%)
	7-4	Crawling Version List	100	0/0(0.00%)
	7-5	24 Points	100	2/6(33.33%)
	7-6	Quick Sort Not in Place	60	0/0(0.00%)
	7-7	Merge Sort	70	0/0(0.00%)
	7-8	Tim Sort	70	0/0(0.00%)


THANK YOU FOR YOUR ATTENTION!