Processing Data with pandas and numpy

CS 165, Project in Algorithms and Data Structures
UC Irvine
Spring 2020

Presented by Rob Gevorkyan

Tools we'll use

- pandas : loading csv data into a data structure we can manipulate in Python
- numpy: scientific computation package for regression line coefficient calculation and various vectorized computations.

Installing the tools

- Using the python package manager ‘pip’, use the following command from the command line to get all of the required packages. They are not installed in the standard Python library.

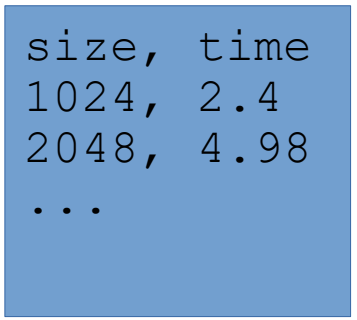
```
pip install numpy pandas
```

- If you do not have pip installed, you can get it from the command line with these commands:

```
curl https://bootstrap.pypa.io/get-pip.py -o get-
pip.py
python get-pip.py1
```

pandas

- A pandas data frame is analogous to a relational database table or excel spreadsheet. It consists of columns and rows.
- Each row has a data entry for each of one or more columns. Each column has a value for every row.
- For project 1, our data frames will consist of at least columns for the size of the input and the time (however you choose to define it) the execution took.



shell_sort1_timings.csv

size	time
1024	2.4
2048	4.98
...	...

shell_sort1_df

Loading pandas dataframes

- To create a pandas dataframe from a csv file, you can use the pandas function read_csv.
- An example is shown below. Note that you must specify a separator of ‘,’ explicitly because sometimes csv files are delimited with other characters since data can sometimes (but not in our case) contain ‘,’ characters. By default, this function assumes the first line is a header line containing the column names.

```
import pandas as pd
df = pd.read_csv('shell_sort1_timings.csv', sep=',')
```

Working with data frames

- Individual columns can be extracted by indexing with [ ] and specifying the name of the desired column.
- Example:
  - Suppose we have a data frame df with columns **size** and **time**

```
# extract all size values in one variable
sizes = df['size']

# extract all times in one variable
times = df['time']
```

Working with data frames

- To select only those rows satisfying a certain condition, you can use a boolean expression in the index operator [ ]. For those familiar with basic SQL queries, this is very similar to using a WHERE clause. Note that this condition is logically evaluated for each row and does not require a loop over the rows to work.
- Example
  - Suppose we have a dataframe df that unifies all algorithm data and contains columns **algorithm**, **size**, **time**

```
merge_sort_rows = df[df['algorithm'] == 'merge_sort']
```

Regression line fitting

- We can perform linear regression in order to approximate a line that closely fits the data.
- For the mathematically/statistically inclined, you can read more about linear regression here
- To compute the necessary slope and intercept for plotting the line, we need two columns from a data frame to serve as our x and y values.
- The syntax is shown below

```
import numpy as np
x = df['size']
y = df['time']
# the third argument is the degree of the polynomial
# we use 1 for linear
m, b = np.polyfit(x, y, 1)
```

Additional Resources

- pandas
  - https://bitbucket.org/hrojas/learn-pandas/src/master/
  - https://pandas.pydata.org/pandas-docs/stable/getting_started/10min.html
- numpy
  - https://docs.scipy.org/doc/numpy/user/quickstart.html
  - https://docs.scipy.org/doc/numpy/user/basics.html