Data Structures & Algorithms
Individual Assignment

Dr. Przemek ('Pshemek') Grabowicz

przemek.grabowicz@ucd.ie

Office: CS B2.12
School of Computer Science
University College Dublin

Assessment and Deadline

Brightspace Submission Deadline: March 16, 11:59PM

- This assignment is worth 35% of the overall grade.
- Your submission should be a **single Python Notebook** with code, analysis and graphs all contained within.
- It should be uploaded to Brightspace by March 16.

Notebook and input file on Brightspace

The Notebook will get you started. Please provide your solutions to the two tasks there.

Learning Outcomes

On successful completion of this project the learner will be able to:

- Understand how to determine the amount of resources necessary to execute a particular algorithm.
- Understand and practice the use of fundamental data structures such as arrays and hash tables.
- Understand how to use curve fitting to identify the function describing how running time depends on input size.
- Successfully write, compile, debug and run programs using these constructs.

Individual Assignment - Task 1

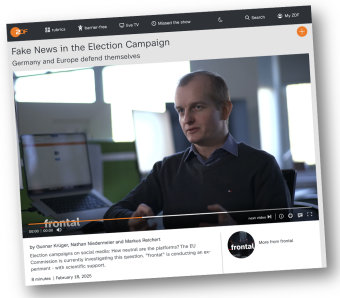
How many times does X feed algorithm show each X user?

1. My research lab examined whether German X users would see politically balanced news feeds if they followed comparable leading politicians from each federal parliamentary party of Germany. We collected tweets that appeared in the feed of such users. **Who appeared most often in their feed?**
2. We ask you to compute the number of times each user appeared in the For You feed using different data structures and algorithms, and compare their performance.

Notebook and input file on Brightspace

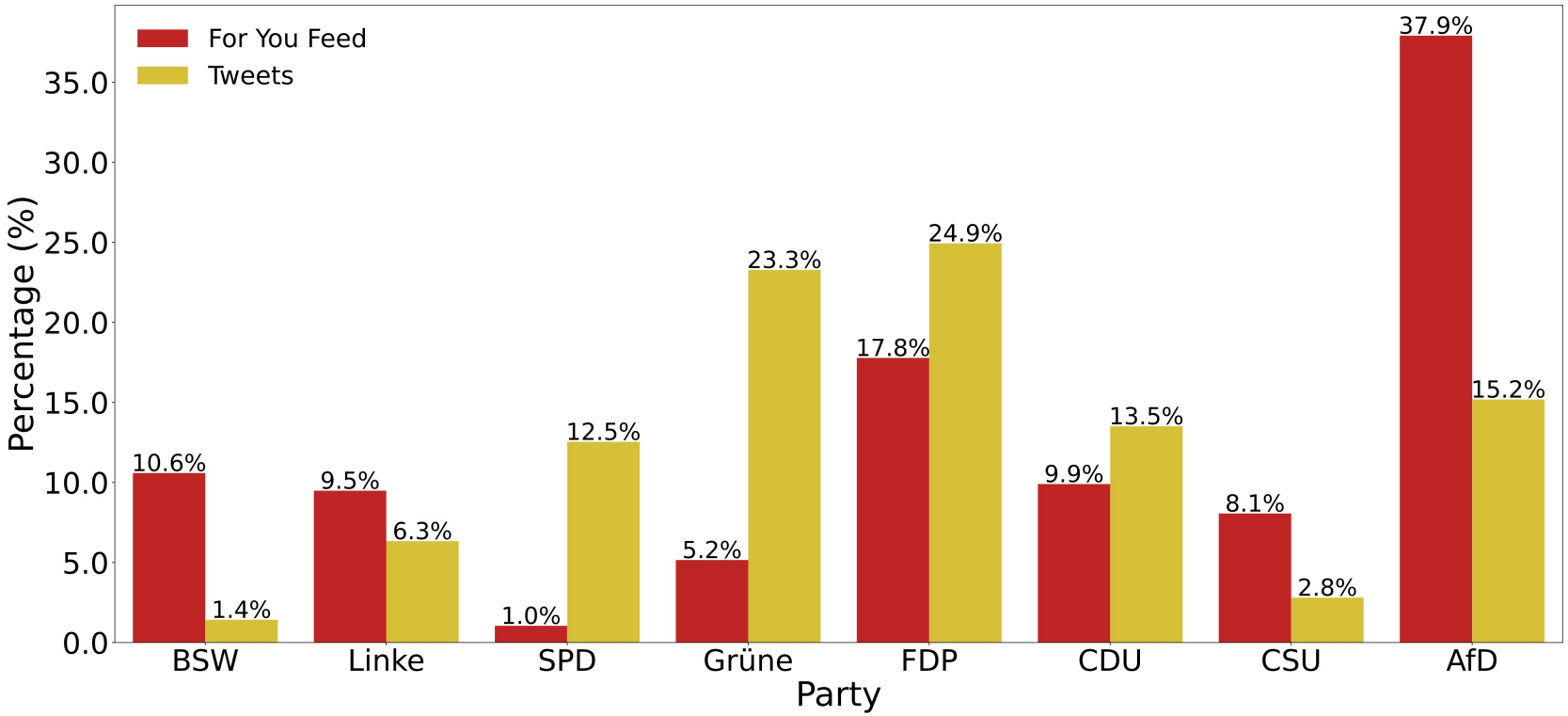
Please follow the instructions provided there.

The Story Behind Task 1



The Story Behind Task 1

https://uncommongood.substack.com



- The default feed of X showed more content from far-right AfD than other parties

Input size described by two values: n and k

In the assignment, we have n appearances of users in X feed, but only $k < n$ unique users.

Input size described by two values: n and k

In the assignment, we have n appearances of users in X feed, but only $k < n$ unique users.

Complexity can be described by two numbers:

Algorithm Return possible products of the numbers in two arrays

Input: an array1 of size n, an array2 of size k
Output: list of all possible products between elements in the arrays

```
products ← []
for i in [0, n-1] do
    for j in [0, k-1] do
        products.append(array1[i] * array2[j])
    endfor
endfor
return products
```

Complexity: $\mathcal{O}(n * k)$

Input size described by two values: n and k

Say, we have worst-case running times (the number of operations) with respect to input size characterized by n and k, where each of them can go to infinity.

`function_a`:	$T(n) = 4n + 1 + \log k$
`function_b`:	$T(n) = n^2 + 2n + k + 2$
`function_c`:	$T(n) = 6n^2 + 3n + 2n \log k + 1$

Easy to express T in Big-$\mathcal{O}$ by dropping constants and lower-order terms, **separately** for n and k.

In Big-$\mathcal{O}$ notation

`function_a`:	$\mathcal{O}(n + \log k)$
`function_b`:	$\mathcal{O}(n^2 + k)$
`function_c`:	$\mathcal{O}(n^2 + n \log k)$

Individual Assignment - Task 2

Search Algorithms

You're asked to implement two search algorithms. Their inputs include a sorted array and a search target value.

1. Implement a linear search algorithm (iterating over an array)
2. Implement a binary search algorithm (recursive or iterative)
3. What is the worst-case time complexity of these two algorithms in Big-$\mathcal{O}$ notation?
4. For each algorithm, measure and plot the average running times as a function of n.
5. Use a curve-fitting function, e.g., from Scipy, to identify how the measured average-case running time depends on n.

Your Report Should:

- Explain your methods and results and discuss any unusual results you find
- Have proper structure with introduction, main body covering the various areas you address, and conclusions
- Have satisfactory technical coverage, balancing breadth of coverage with depth
- Have soundness of argument
- Have good clarity of expression and level of readability
- Clearly reference sources of information, and avoid plagiarism

Assessment: Things to Consider

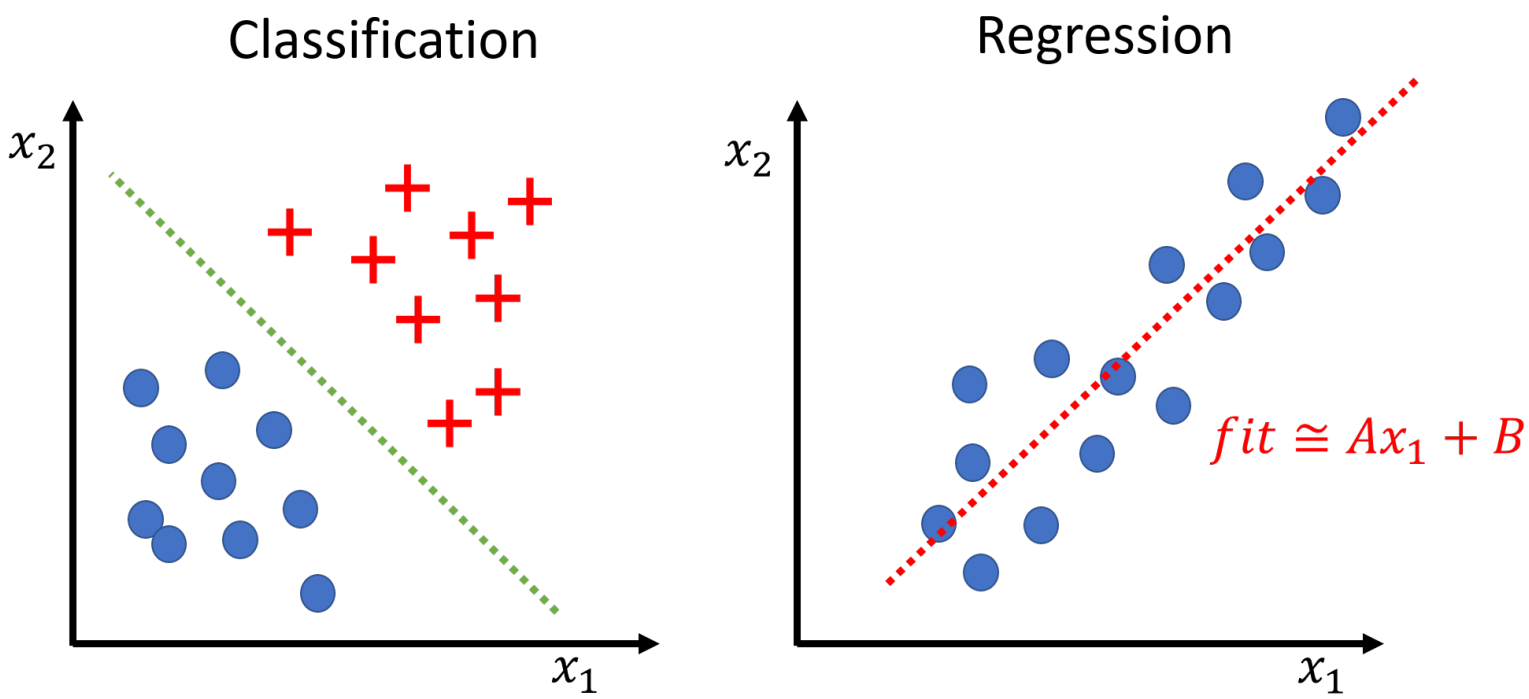
Speed: Test running time as input size grows. Compare running-time vs time complexity
Correctness: Does it do what it is meant to do?
Clarity:

- Are class names, method names, variable names meaningful; is a consistent style used throughout;
- Is code refactored (remove unnecessary or unused variables, loops etc.)?

Maintainability:

- Are appropriate comments included in code?
- Have you (unit) tested the key functionalities

Example



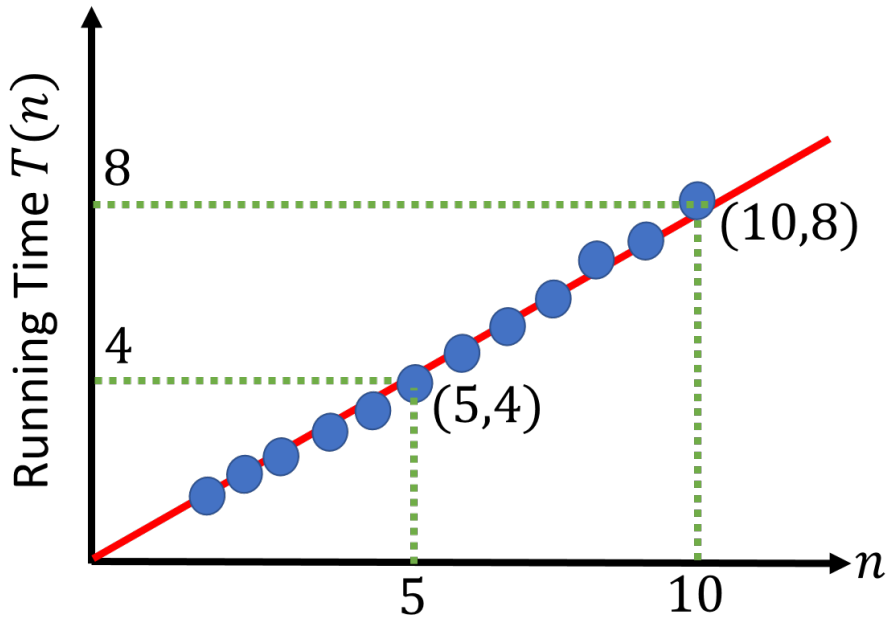
In the field of machine-learning, there are two common problems:

- Classification is to estimate a mapping function that predicts the class or category for a given observation
- Regression is to estimate a mapping function that predicts continuous output

Regression Example

There are many algorithms to predict the fitting function, but let us take the simplest one as an example:

- The general form of linear running time is $T(n) = a * n + b$
- The problem is to find values of a (slope) and b (T-intercept)
- Use two points. Ex: $(n_1, T_1) = (5, 4)$ and $(n_2, T_2) = (10, 8)$
- $4 = 5 * a + b \ldots$ (1) and $8 = 10 * a + b \ldots$ (2)
- By solving the two equations, we get $a = \frac{4}{5}$ and $b = 0$
- $T(n) = \frac{4}{5}n$



Curve Fitting in Python

A common Python library such as SciPy Curve Fitting can be used to fit the worst-case running time function.

See an example here: https://www.askpython.com/python/examples/curve-fitting-in-python.