Discrete Structures: CMPSC 102 A Review of the Class

Oliver BONHAM-CARTER

Fall 2018
Week 15

Final Exam Topics

- Given on Tuesday 18^{th} December at 9:00am, Alden 101
- Online format
- Three hours to complete
- Fifteen questions: Multi-choice, True/False and short answer
- Material covered since week 12

What types of things to study

- **Slides, notes, with chapters to add detail to class material**
- Main ideas behind mathematical subjects in class (again, study your slides)
- Python basics and code
- Sets (main ideas)
- Basic stats: meaning and understanding of how to use a measurement
- Differences between sets, dictionaries, lists
  - Determining the correct data-type for a task.
- Conceptual questions: What did each week teach you in terms of general concepts in discrete structures and computation?

Course Overview: Academic Bulletin Description

An introduction to the foundations of computer science with an emphasis on understanding the abstract structures used to represent discrete objects. Participating in hands-on activities that often require teamwork, students learn the computational methods and logical principles that they need to create and manipulate discrete objects in a programming environment. Students also learn how to write, organize, and document a programs source code so that it is easily accessible to intended users of varied backgrounds. During a weekly laboratory session students use state-of-the-art technology to complete projects, reporting on their results through both written documents and oral presentations. Prerequisite: Knowledge of elementary algebra. Distribution Requirements: QR, SP.

What did I learn here?

> *"An introduction to the foundations of computer science with an emphasis on understanding the abstract structures used to represent discrete objects."*

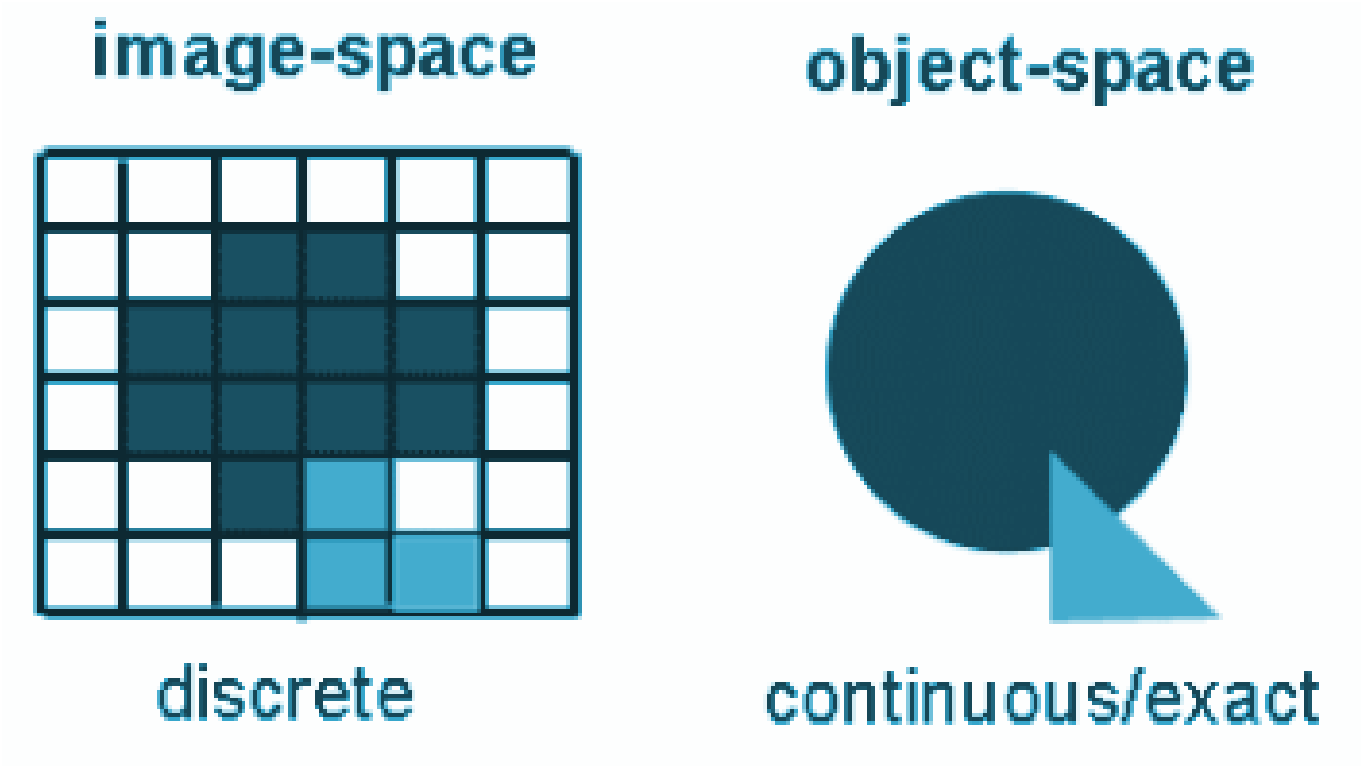
Wait! What?

What is do you mean by, **discrete**?

Discreet or *Discrete*

- **Discreet** means *unobtrusive* or *unnoticeable* (not this course!)
- **Discrete** means *separate*, not continuous or *not sharing any common space*

So, Discrete then?



- Discrete mathematics involves *countable* things.

Discrete objects

- *Discrete* means "countable"
- We can count the number of animals.

About Python...

- Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance.
- Python supports modules and packages, which encourages program modularity and code reuse.
- The Python interpreter and the extensive standard library are open source and freely available in all major platforms

About Python...

Topics

- Using the interactive shell
- Mathematical operators
- If statements
- Conditional statements
- Reading from files
- Data structures
- Applying mathematical reasoning, logic to your code.

Practicals

- Calculating factorials
- Approximating square roots
- Fibonacci sequences

Finding Factorial (not Nemo!)

Practical work: Putting mathematics to computation


```
Enter a number : 10
 The number you entered is the following : 10

Testing the number for Odd or Even polarity ...
 The number << 10 >> is EVEN:

Determing the factorial of the number ...
 Current value of fact_int :  1
 Current value of fact_int :  1
 Current value of fact_int :  2
 Current value of fact_int :  6
 Current value of fact_int :  24
 Current value of fact_int :  120
 Current value of fact_int :  720
 Current value of fact_int :  5040
 Current value of fact_int :  40320
 Current value of fact_int :  362880

 * Factorial for 10 is : 3628800
```

$$n! = \prod_{k=1}^{n} k = n * (n-1) * (n-2) \cdots (3) * (2) * (1)$$

Finding Factorial (not Nemo!)

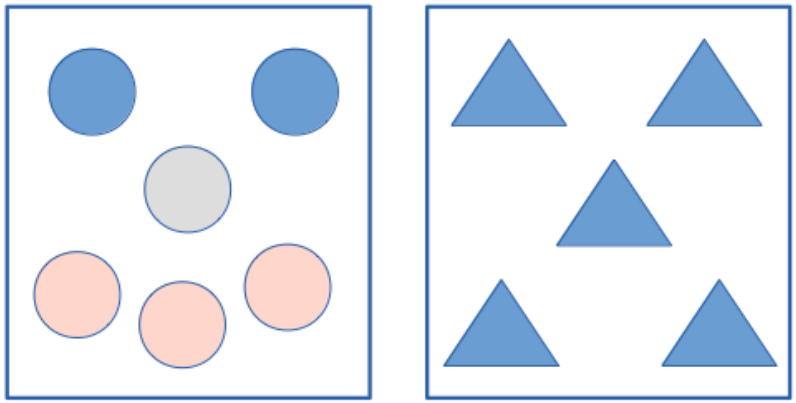
Practical work: The steps for approximation.


```
Finding square root of :  2
 Initial values:  n =  2 and guess =  1.0
 Square root result :  1.4142156862745099

Finding cube root of :  175616
Approx guess :  1.0
 Initial values:  n =  175616 and guess =  1.0
 Cube root result :  56.00000000040617

Finding forth root of :  9834496
 Initial values:  n =  9834496 and guess =  1.0
 Forth root result :  56.0
```

Guess			Approx. root
x_n	$f(x) = x_n^2 - 2$	$f'(x_n) = 2x$	$x_n - \frac{f(x_n)}{f'(x_n)}$
1	-1	2	$1 - \frac{-1}{2} = \frac{3}{2} = 1.5$
$\frac{3}{2}$	$\frac{1}{4} = 0.25$	3.0	$\frac{3}{2} - \frac{(\frac{1}{4})}{3} = \frac{17}{12} = 1.4167$
$\frac{17}{12}$	$\frac{1}{144}$	$\frac{17}{6}$	$\frac{17}{6} - \frac{\frac{1}{144}}{\frac{17}{6}} = \frac{577}{408} = 1.4142$

Types of Sets

One decides which elements make up a set



Intentional definition of sets

- A_1 is the set whose members are the first four positive integers.
- B_1 is the set of colors of the Union Jack (i.e., the British flag)

Types of Sets

Sets of members in curly brackets

Extensional definition of sets

- $A_2 = \{4, 2, 1, 3\}$
  - The first four positive numbers
- $B_2 = \{\text{Blue, Red and White}\}$
  - The set of colors of the Union Jack (the British flag)
- $F = \{n^2 - 4 : n \text{ is an integer; and } 0 \leq n \leq 19\}$
  - The set of all values gained from plugging in n between 0 and 19 into the equation $n^2 - 4$

Types of Sets

Practical: Used to make a secret writing program

- **Intentional Definition:**
  - A_1 is the set are the first four positive integers.
  - B_1 is the set of colors of the Union Jack
- **Extensional Definition:**
  - $A_2 = \{4, 2, 1, 3\}$
  - $B_2 = \{\text{Blue, Red and White}\}$

Prepositional Logic

- A proposition statement:
  - Denoted by a capital letter (i.e., "A")
  - A negation of a proposition statement
  - $\sim A$: "**not** A"
  - Two proposition statements joined by a *connective*
  - $A \wedge B$: "A **and** B"
  - $A \vee B$: "A **or** B"
  - If a connective joins complex statements, parenthesis are added
  - $A \wedge (B \vee C)$: "A and (B or C)"

Compound Truth tables

A	B	∼A	$A \vee B$	$(\sim A) \wedge (A \vee B)$
0	0	1	0	0
0	1	1	1	1
1	0	0	1	0
1	1	0	1	0

Legend

- AND is denoted by : $\wedge$
- OR is denoted by : $\vee$
- Contradiction is denoted by : $\sim$
- Equivalency is denoted by : $\equiv$

Generator Functions For Fibonacci Sequences

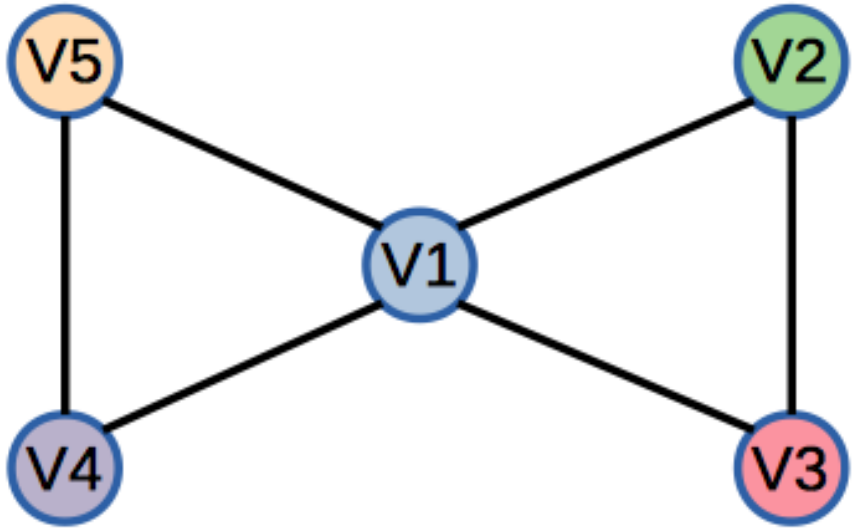
Creating sequences dynamically with *yield*

- Functions having *yield*-statement are generator
- This function works as a generator or otherwise

A generator function for the Fibonacci sequence

```
def fibs(n):
    a=1
    b=1
    for i in range(n):
        yield a
        a, b = b, a + b
print([x for x in fibs(6)])
print("  My type is:",type(fibs))
f = fibs(6)
for i in f: print(i)
print("  My type is: ",type(fibs(6)))
```

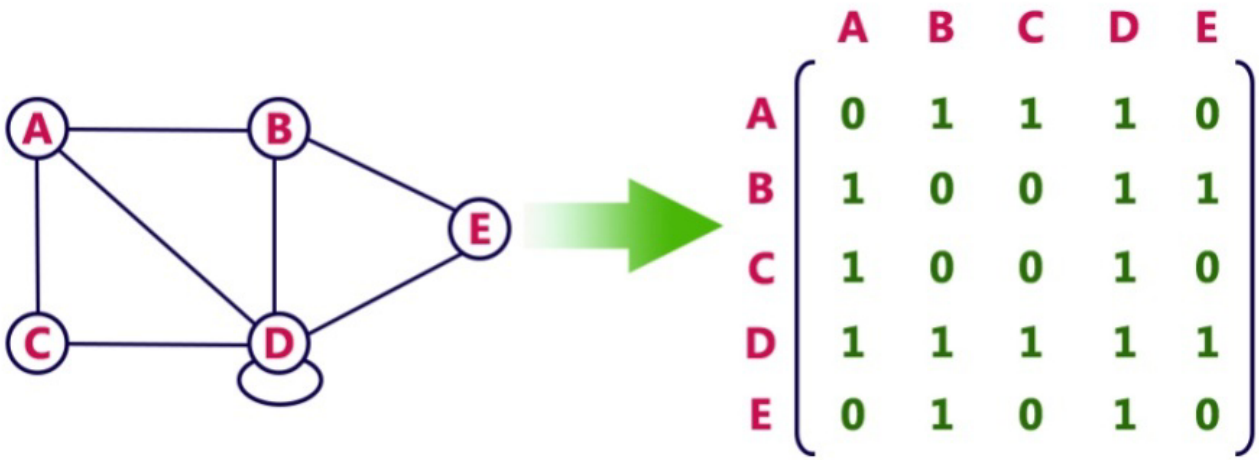
Define a Graph



A Bowtie Graph

- We define a graph by its vertices and edges: G = (V, E)
  - Vertices: V(G) = $\{V_1, V_2, V_3, V_4, V_5\}$
  - Edges: E(G) = $\{V_1V_2,\ V_2V_3,\ V_3V_1,\ V_4V_1,\ V_5V_1,\ V_4V_5\}$

Adjacency Matrices



A matrix is used describe adjacent vertices

- A matrix contains rows and columns
- Vertices are labelled with a 1 or 0 in position (v_i, v_j) according to whether v_i and v_j are adjacent vertices

Creating Plots with Matplotlib

- We first need to know that the library is installed on your machine.

python3

```
from pylab import plot, show
```

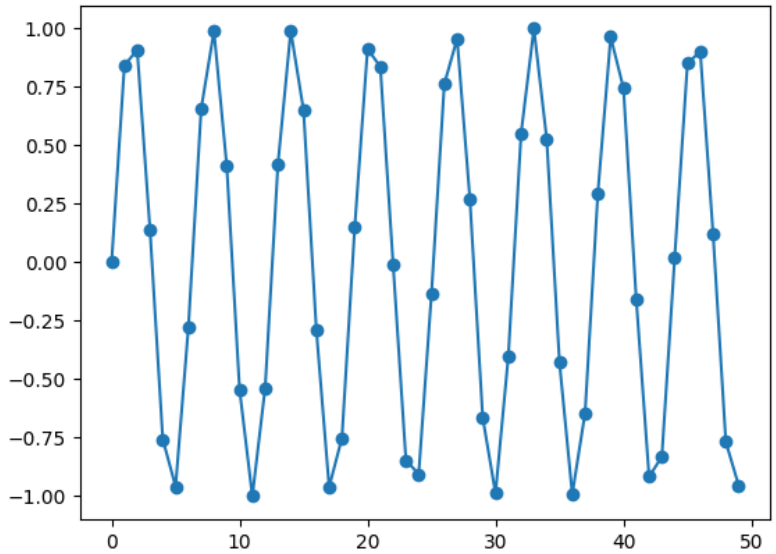
- https://matplotlib.org/index.html
- https://matplotlib.org/3.0.0/users/installing.html

Another Amazing Example!

Plot the sin wave

Place in python3 or in a python3 program file

```
from pylab import plot, show #get the library
import math
x_num = [i for i in range(50)]
y_num = [math.sin(i) for i in x_num]
plot(x_num, y_num, marker ='o')
# also including 'o', '*', 'x', and '+' as points
show() # draw the plot on canvas
```



Frequency Fingerprints of Famous Writers!

Text Analysis

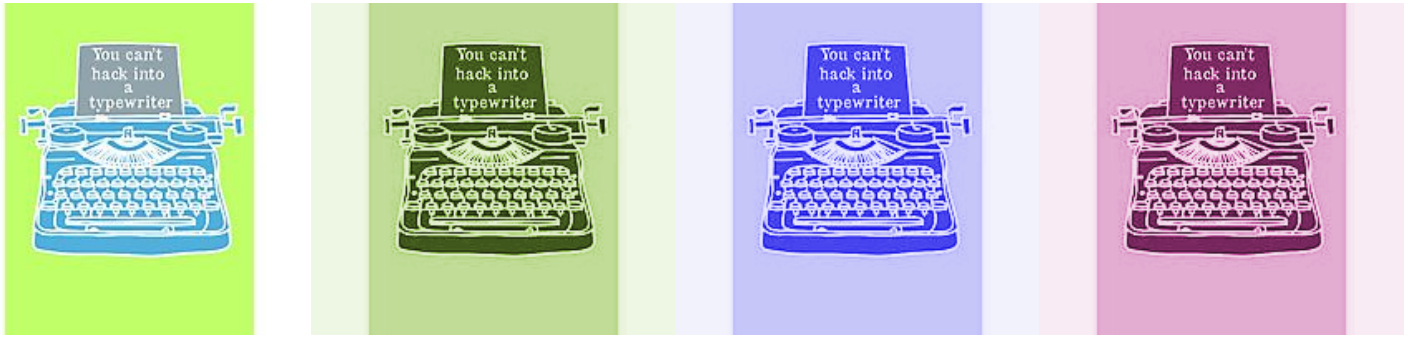


Figure: Maybe we cannot hack into a typewriter but we can still hack the text that typewriters have produced. For this type of hacking, we collect frequency information to determine the distribution of frequencies.

Frequency Fingerprints of Famous Writers!

Text Analysis

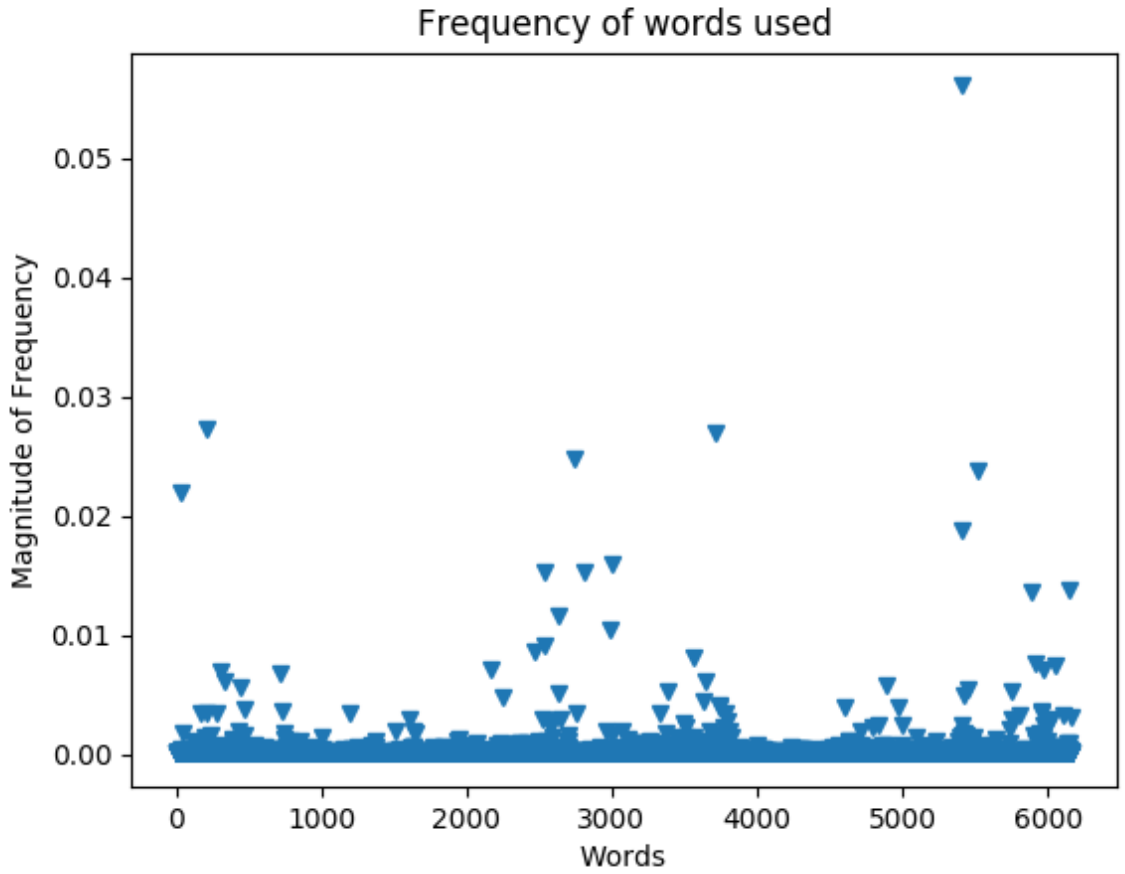
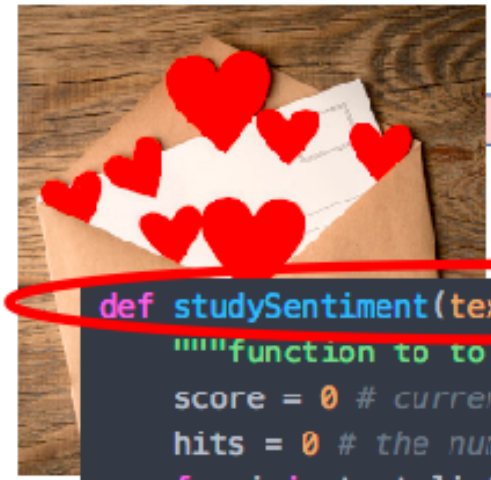


Figure: Authur Conan Doyle's fingerprint from word frequencies.

Sentiment Analyzer!

Text Analysis




```
def studySentiment(text_list, sentiments_dict):
    """function to to determine the sentiment score from the text."""
    score = 0 # current score of the sentimental words
    hits = 0 # the number of words which have a sentiment value
    for i in text_list:
        #print("   Current word: ",i)
        try:
            wordScore_int = int(sentiments_dict[i])
            print("    <<", i,">> : score: ",wordScore_int)
            score =  score + wordScore_int
            hits = hits + 1
        except KeyError:
            #print("  <<",i,">> Word not found in sentiments_dict...")
            pass
```

What else did we learn?!

Much more!!