Dynamic Programming

M269 Tutorial

Contents

Commentary 1

1 Agenda, Aims and Topics

- Overview of aims of tutorial
- Note selection of topics
- Recursion is used throughout the topics
- Points about my own background and preferences
- Adobe Connect slides for reference

ToC

1 Agenda

- Welcome and introductions
- Dynamic Programming — introduction
- Python: List comprehensions, Named Tuples
- Implementations in Structured English, Python and Haskell (Optional)
- Note there is more material here than we can cover — some is for optional interest
- Not covered in this session: section 2 Adobe Connect notes, sections 5.1.1 Fibonacci closed form, 5.1.2 Fibonacci Alternative Calculation, sections 5.7 Edit Distance — Haskell Implementation, 5.10 Edit Distance Diagram Construction
- Slides/Notes are at pmolyneux.co.uk/OU/M269FolderSync/M269TutorialNotes/M269Tutorial2
- **Recording** Meeting > Record Meeting... ✔

Introductions — Phil

- Name Phil Molyneux
- Background
  - Undergraduate: Physics and Maths (Sussex)
  - Postgraduate: Physics (Sussex), Operational Research (Brunel), Computer Science (University College, London)

  - Worked in Operational Research, Business IT, Web technologies, Functional Programming
- First programming languages Fortran, BASIC, Pascal
- Favourite Software
  - Haskell — pure functional programming language
  - Text editors TextMate, Sublime Text — previously Emacs
  - Word processing in LaTeX — all these slides and notes
  - Mac OS X
- Learning style — I read the manual before using the software

Introductions — You

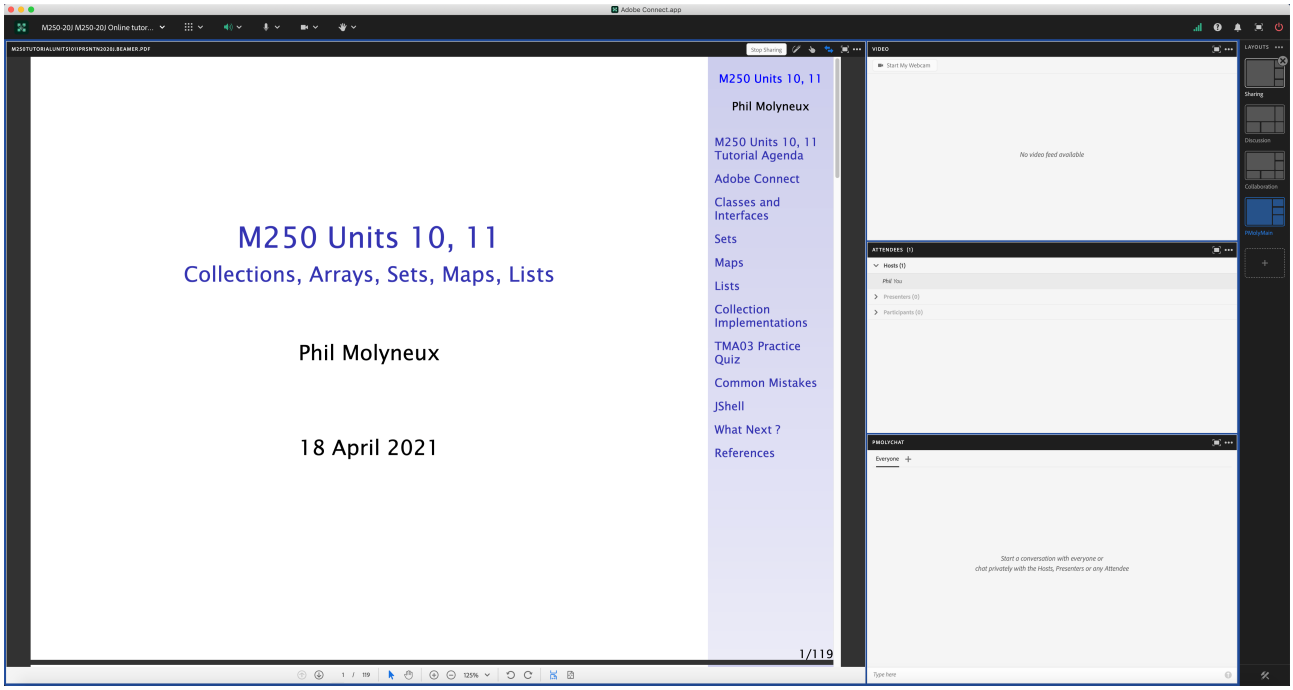
- Name ?
- Favourite software/Programming language ?
- Favourite text editor or integrated development environment (IDE)
- List of text editors, Comparison of text editors and Comparison of integrated development environments
- Other OU courses ?
- Anything else ?

ToC

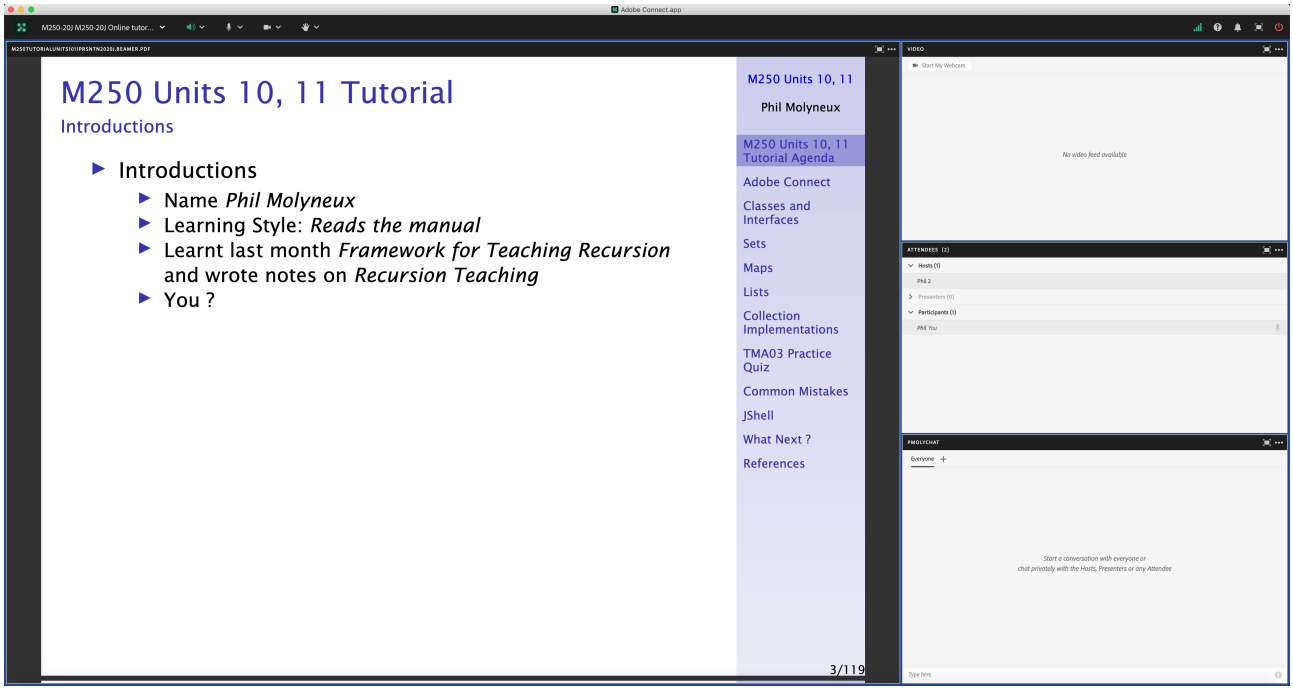
2 Adobe Connect Interface and Settings

2.1 Adobe Connect Interface

Adobe Connect Interface — Host View



Adobe Connect Interface — Participant View



2.2 Adobe Connect Settings

Adobe Connect — Settings

- **Everybody** Menu bar › Meeting › Speaker & Microphone Setup
- Menu bar › Microphone › Allow Participants to Use Microphone ✔
- Check Participants see the entire slide including slide numbers bottom right **Workaround**
  - Disable Draw Share pod › Menu bar › Draw icon
  - Fit Width Share pod › Bottom bar › Fit Width icon ✔

- Meeting > Preferences > General > **Host Cursor** > Show to all attendees
- Menu bar > Video > Enable **Webcam** for Participants ✔
- Do not Enable single speaker mode
- Cancel hand tool
- Do not enable green pointer
- **Recording** Meeting > Record Session ✔
- **Documents** Upload PDF with drag and drop to share pod
- Delete Meeting > Manage Meeting Information > Uploaded Content and check filename > click on delete

Adobe Connect — Access

- **Tutor Access**

  TutorHome > M269 Website > Tutorials

  Cluster Tutorials > M269 Online tutorial room

  Tutor Groups > M269 Online tutor group room

  Module-wide Tutorials > M269 Online module-wide room

- **Attendance**

  TutorHome > Students > View your tutorial timetables

- **Beamer Slide Scaling** 440% (422 x 563 mm)
- **Clear Everyone's Status**

  Attendee Pod > Menu > Clear Everyone's Status

- **Grant Access** and send link via email

  Meeting > Manage Access & Entry > Invite Participants...

- **Presenter Only Area**

  Meeting > Enable/Disable Presenter Only Area

Adobe Connect — Keystroke Shortcuts

- Keyboard shortcuts in Adobe Connect
- **Toggle Mic** ⌘ + M (Mac), Ctrl + M (Win) (On/Disconnect)
- **Toggle Raise-Hand status** ⌘ + E
- **Close dialog box** ⎋ (Mac), Esc (Win)
- **End meeting** ⌘ + \

2.3 Adobe Connect — Sharing Screen & Applications

- Share My Screen > Application tab > Terminal for Terminal
- Share menu > Change View > Zoom in for mismatch of screen size/resolution (Participants)

- (Presenter) Change to 75% and back to 100% (solves participants with smaller screen image overlap)
- Leave the application on the original display
- Beware blued hatched rectangles — from other (hidden) windows or contextual menus
- Presenter screen pointer affects viewer display — beware of moving the pointer away from the application
- First time: System Preferences › Security & Privacy › Privacy › Accessibility

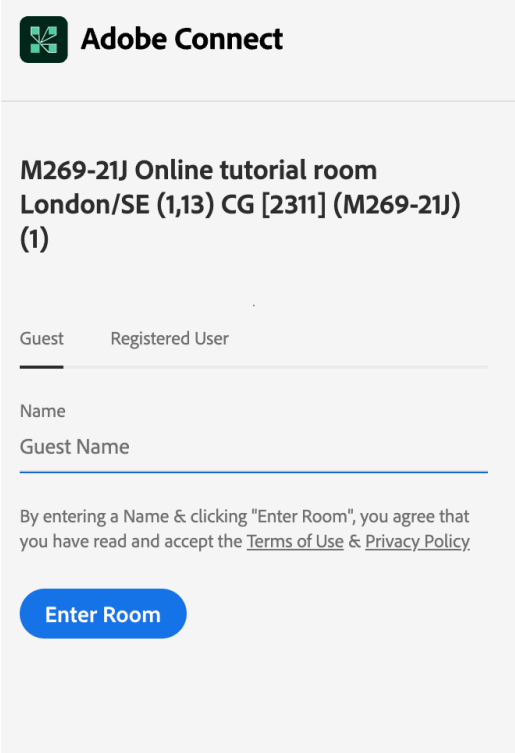
2.4 Adobe Connect — Ending a Meeting

- Notes for the tutor only
- **Student:** Meeting › Exit Adobe Connect
- **Tutor:**
- **Recording** Meeting › Stop Recording ✔
- **Remove Participants** Meeting › End Meeting... ✔
  - Dialog box allows for message with default message:
  - The host has ended this meeting. Thank you for attending.
- **Recording availability** In course Web site for joining the room, click on the eye icon in the list of recordings under your recording — edit description and name
- **Meeting Information** Meeting › Manage Meeting Information — can access a range of information in Web page.
- **Delete File Upload** Meeting › Manage Meeting Information › Uploaded Content tab select file(s) and click Delete
- **Attendance Report** see course Web site for joining room

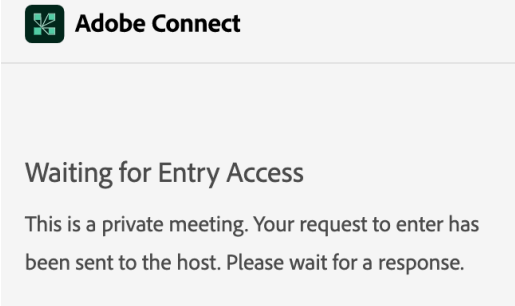
2.5 Adobe Connect — Invite Attendees

- **Provide Meeting URL** Menu › Meeting › Manage Access & Entry › Invite Participants...
- **Allow Access without Dialog** Menu › Meeting › Manage Meeting Information provides new browser window with Meeting Information Tab bar › Edit Information
- Check Anyone who has the URL for the meeting can enter the room
- Default Only registered users and accepted guests may enter the room
- **Reverts to default next session but URL is fixed**
- Guests have blue icon top, registered participants have yellow icon top — same icon if URL is open
- See Start, attend, and manage Adobe Connect meetings and sessions
- Click on the link sent in email from the Host
- Get the following on a Web page

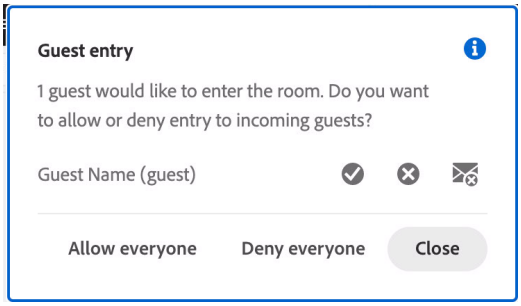
- As Guest enter your name and click on Enter Room



- See the Waiting for Entry Access for Host to give permission



- Host sees the following dialog in Adobe Connect and grants access



2.6 Layouts

- **Creating new layouts** example Sharing layout
- Menu > Layouts > Create New Layout... > Create a New Layout dialog > Create a new blank layout and name it PMolyMain
- New layout has no Pods but does have Layouts Bar open (see Layouts menu)

- **Pods**
- Menu > Pods > Share > Add New Share and resize/position — initial name is Share n — rename PMolyShare
- **Rename Pod** Menu > Pods > Manage Pods... > Manage Pods > Select > Rename or Double-click & rename
- Add Video pod and resize/reposition
- Add Attendance pod and resize/reposition
- Add Chat pod — rename it PMolyChat — and resize/reposition
- Dimensions of **Sharing** layout (on 27-inch iMac)
  - Width of Video, Attendees, Chat column 14 cm
  - Height of Video pod 9 cm
  - Height of Attendees pod 12 cm
  - Height of Chat pod 8 cm
- **Duplicating Layouts** does not give new instances of the Pods and is probably not a good idea (apart from local use to avoid delay in reloading Pods)
- **Auxiliary Layouts** name PMolyAux0n
  - Create new Share pod
  - Use existing Chat pod
  - Use same Video and Attendance pods

2.7 Chat Pods

- **Format Chat text**
- Chat Pod > menu icon > My Chat Color
- Choices: Red, Orange, Green, Brown, Purple, Pink, Blue, Black
- Note: Color reverts to Black if you switch layouts
- Chat Pod > menu icon > Show Timestamps

2.8 Graphics Conversion for Web

- Conversion of the screen snaps for the installation of Anaconda on 1 May 2020
- Using GraphicConverter 11
- File > Convert & Modify > Conversion > Convert
- Select files to convert and destination folder
- Click on Start selected Function or ⌘ + ⏎

2.9 Adobe Connect Recordings

- Menu bar › Meeting › Preferences › Video
- Aspect ratio › Standard (4:3) (not Wide screen (16:9) default)
- Video quality › Full HD (1080p not High default 480p)
- **Recording** Menu bar › Meeting › Record Session ✔
- **Export Recording**
- Menu bar › Meeting › Manage Meeting Information
- New window › Recordings › check Tutorial › Access Type button
- check Public › check Allow viewers to download
- **Download Recording**
- New window › Recordings › check Tutorial › Actions › Download File

ToC

Commentary 2

2 DP Introduction, List Comprehensions

- Overview of Dynamic Programming
- Obtain recursive algorithm for problem but implement bottom up
- Introduction to list comprehensions as a concise way of iterating over lists (or other iterables)
- List comprehension exercises with solutions

ToC

3 Dynamic Programming — Introduction

- Dynamic Programming invented by Richard Bellman in the 1950s
- Programming meant planning the sequence of decisions
- Dynamic suggested evolution of the system over time
- Attributes: (1) optimal substructure (2) overlapping subproblems
- Divide and conquer has non-overlapping subproblems — a tree-structure
- DP has an acyclic graph structure
- Dynamic Programming process:
- Obtain a recursive solution
- Two ways to avoid subproblems being calculated more than once:
- Memoization uses the top-down recursive structure but preserves subproblems in a table for subsequent retrieval

- Tabulation works bottom-up which orders the computations so that simplest results calculated first and dependent problems follow on in some order

ToC

4 List Comprehensions

4.1 List Comprehensions

List Comprehensions — Python

- **List Comprehensions** provide a concise way of performing calculations over lists (or other iterables)
- Example: Square the even numbers between 0 and 9

```
Python3>>> [x ** 2 for x in range(10) if x % 2 == 0]
[0, 4, 16, 36, 64]
Python3>>> [(x,y) for x in range(4)
...               for y in range(4)
...               if x % 2 == 0
...                  and y % 3 == 0]
[(0, 0), (0, 3), (2, 0), (2, 3)]
Python3>>>
```

- In general

```
  [expr for target1 in iterable1 if cond1
        for target2 in iterable2 if cond2 ...
        for targetN in iterableN if condN ]
```

- Lots example usage in the algorithms below

List Comprehensions — Haskell

- **List Comprehensions** provide a concise way of performing calculations over lists
- Example: Square the even numbers between 0 and 9

```
GHCi> [x^2 | x <- [0..9], x `mod` 2 == 0]
[0,4,16,36,64]
GHCi>
```

- In general

```
  [expr | qual1, qual2,..., qualN]
```

- The qualifiers `qual` can be
  - Generators `pattern <- list`
  - Boolean guards — acting as filters
  - Local declarations with `let decls` for use in `expr` and later generators and boolean guards

Activity 1 (a) Stop Words Filter

- Stop words are the most common words that most search engines avoid: `'a','an','the','tha`
- Using list comprehensions, write a function `filterStopWords` that takes a list of words and filters out the stop words

- Here is the initial code

```
sentence \
 = "the␣quick␣brown␣fox␣jumps␣over␣the␣lazy␣dog"

words = sentence.split()

wordsTest \
 = (words == ['the', 'quick', 'brown'
              , 'fox', 'jumps', 'over'
              , 'the', 'lazy', 'dog'])

stopWords \
 = ['a','an','the','that']
```

Go to Answer

Activity 1 (a) Stop Words Filter

```
sentence \
 = "the␣quick␣brown␣fox␣jumps␣over␣the␣lazy␣dog"

words = sentence.split()

wordsTest \
 = (words == ['the', 'quick', 'brown'
              , 'fox', 'jumps', 'over'
              , 'the', 'lazy', 'dog'])

stopWords \
 = ['a','an','the','that']
```

- Notice the Python Explicit line joining with (\<nl>) and Python Implicit line joining with (`(...)`)
- The backslash (\) must be followed by an end of line character (<nl>)
- The ('␣') symbol represents a space (see Unicode U+2423 Open Box)

Go to Answer

Activity 1 (b) Transpose Matrix

- A matrix can be represented as a list of rows of numbers
- We transpose a matrix by swapping columns and rows
- Here is an example

```
matrixA \
 = [[1, 2, 3, 4]
   ,[5, 6, 7 ,8]
   ,[9, 10, 11, 12]]

matATr \
 = [[1, 5, 9]
   ,[2, 6, 10]
   ,[3, 7, 11]
   ,[4, 8, 12]]
```

- Using list comprehensions, write a function `transMat`, to transpose a matrix

Go to Answer

Activity 1 (c) List Pairs in Fair Order

- Write a function which takes a pair of positive integers and outputs a list of all possible pairs in those ranges
- If we do this in the simplest way we get a bias to one argument

- Here is an example of a bias to the second argument

```
yBiasLstTest \
 = (yBiasListing(5,5)
     == [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4)
        , (1, 0), (1, 1), (1, 2), (1, 3), (1, 4)
        , (2, 0), (2, 1), (2, 2), (2, 3), (2, 4)
        , (3, 0), (3, 1), (3, 2), (3, 3), (3, 4)
        , (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)])
```

Go to Answer

Activity 1 (c) List Pairs in Fair Order

- Rewrite the function which takes a pair of positive integers and outputs a list of all possible pairs in those ranges
- The output should treat each argument fairly — any initial prefix should have roughly the same number of instances of each argument
- Here is an example output

```
fairLstTest \
 = (fairListing(5,5)
     == [(0, 0)
        , (0, 1), (1, 0)
        , (0, 2), (1, 1), (2, 0)
        , (0, 3), (1, 2), (2, 1), (3, 0)
        , (0, 4), (1, 3), (2, 2), (3, 1), (4, 0)])
```

Go to Answer

Activity 1 (c) List Pairs in Fair Order

- Rewrite the function which takes a pair of positive integers and outputs a list of lists of all possible pairs in those ranges
- The output should treat each argument fairly — any initial prefix should have roughly the same number of instances of each argument — further, the output should be segment by each initial prefix (see example below)
- Here is an example output

```
fairLstATest \
 = (fairListingA(5,5)
     == [[(0, 0)]
        , [(0, 1), (1, 0)]
        , [(0, 2), (1, 1), (2, 0)]
        , [(0, 3), (1, 2), (2, 1), (3, 0)]
        , [(0, 4), (1, 3), (2, 2), (3, 1), (4, 0)]])
```

Go to Answer

Answer 1 (a) Stop Words Filter

- Answer 1 (a) Stop Words Filter
- Write here:

Answer 1 (a) Stop Words Filter

- Answer 1 (a) Stop Words Filter

```
def filterStopWords(words) :
  nonStopWords \
   = [word for word in words
           if word not in stopWords]
```

```
28      return nonStopWords


31  filterStopWordsTest \
32   = filterStopWords(words) \
33      == ['quick', 'brown', 'fox'
34        , 'jumps', 'over', 'lazy', 'dog']
```

Go to Activity

Answer 1 (b) Transpose Matrix

- Answer 1 (b) Transpose Matrix
- Write here:

Answer 1 (b) Transpose Matrix

- Answer 1 (b) Transpose Matrix

```
49  def transMat(mat) :
50    rowLen = len(mat[0])
51    matTr \
52     = [[row[i] for row in mat] for i in range(rowLen)]
53    return matTr

55  transMatTestA \
56   = (transMat(matrixA)
57      == matATr)
```

- Note that a list comprehension is a valid expression as a target expression in a list comprehension
- The code assumes every row is of the same length

Go to Activity

Answer 1 (b) Transpose Matrix

- Note the differences in the list comprehensions below

```
38  matrixA \
39   = [[1, 2, 3, 4]
40     ,[5, 6, 7 ,8]
41     ,[9, 10, 11, 12]]
```

```
Python3>>> [[row[i] for row in matrixA]
...                 for i in range(4)]
[[1, 5, 9], [2, 6, 10], [3, 7, 11], [4, 8, 12]]
Python3>>> [row[i] for row in matrixA
...                 for i in range(4)]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Python3>>> [row[i] for i in range(4)
...                 for row in matrixA]
[1, 5, 9, 2, 6, 10, 3, 7, 11, 4, 8, 12]
Python3>>> [[row[i] for i in range(4)]
...                 for row in matrixA]
[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

Go to Activity

Answer 1 (b) Transpose Matrix

- Answer 1 (b) Transpose Matrix
- The Python NumPy package provides functions for N-dimensional array objects
- For transpose see numpy.ndarray.transpose

```
Python3>>> import numpy as np
Python3>>> ar = np.array([[1,2],[3,4]])
Python3>>> ar
array([[1, 2],
       [3, 4]])
Python3>>> arT = ar.transpose()
Python3>>> arT
array([[1, 3],
       [2, 4]])
Python3>>> ar
array([[1, 2],
       [3, 4]])
Python3>>> ar.shape
(2, 2)
```

Go to Activity

Answer 1 (c) List Pairs in Fair Order

- Answer 1 (c) List Pairs in Fair Order — first version
- Write here

```
69 yBiasLstTest \
70  = (yBiasListing(5,5)
71      == [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4)
72         , (1, 0), (1, 1), (1, 2), (1, 3), (1, 4)
73         , (2, 0), (2, 1), (2, 2), (2, 3), (2, 4)
74         , (3, 0), (3, 1), (3, 2), (3, 3), (3, 4)
75         , (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)])
```

Go to Activity

Answer 1 (c) List Pairs in Fair Order

- Answer 1 (c) List Pairs in Fair Order
- This is the obvious but biased version

```
63 def yBiasListing(xRng,yRng) :
64   yBiasLst \
65    = [(x,y) for x in range(xRng)
66             for y in range(yRng)]
67   return yBiasLst

69 yBiasLstTest \
70  = (yBiasListing(5,5)
71      == [(0, 0), (0, 1), (0, 2), (0, 3), (0, 4)
72         , (1, 0), (1, 1), (1, 2), (1, 3), (1, 4)
73         , (2, 0), (2, 1), (2, 2), (2, 3), (2, 4)
74         , (3, 0), (3, 1), (3, 2), (3, 3), (3, 4)
75         , (4, 0), (4, 1), (4, 2), (4, 3), (4, 4)])
```

Go to Activity

Answer 1 (c) List Pairs in Fair Order

- Answer 1 (c) List Pairs in Fair Order — second version
- Write here

```
83 fairLstTest \
84  = (fairListing(5,5)
85      == [(0, 0)
86         , (0, 1), (1, 0)
87         , (0, 2), (1, 1), (2, 0)
88         , (0, 3), (1, 2), (2, 1), (3, 0)
89         , (0, 4), (1, 3), (2, 2), (3, 1), (4, 0)])
```

Go to Activity

Answer 1 (c) List Pairs in Fair Order

- Answer 1 (c) List Pairs in Fair Order — second version
- This works by making the sum of the coordinates the same for each prefix

```
77  def fairListing(xRng,yRng) :
78    fairLst \
79     = [(x,d-x) for d in range(yRng)
80                for x in range(d+1)]
81    return fairLst

83  fairLstTest \
84   = (fairListing(5,5)
85       == [(0, 0)
86          , (0, 1), (1, 0)
87          , (0, 2), (1, 1), (2, 0)
88          , (0, 3), (1, 2), (2, 1), (3, 0)
89          , (0, 4), (1, 3), (2, 2), (3, 1), (4, 0)]
```

Go to Activity

Answer 1 (c) List Pairs in Fair Order

- Answer 1 (c) List Pairs in Fair Order — third version
- Write here

```
97  fairLstATest \
98   = (fairListingA(5,5)
99       == [[(0, 0)]
100         , [(0, 1), (1, 0)]
101         , [(0, 2), (1, 1), (2, 0)]
102         , [(0, 3), (1, 2), (2, 1), (3, 0)]
103         , [(0, 4), (1, 3), (2, 2), (3, 1), (4, 0)]])
```

Go to Activity

Answer 1 (c) List Pairs in Fair Order

- Answer 1 (c) List Pairs in Fair Order — third version
- The inner loop is placed into its own list comprehension

```
91  def fairListingA(xRng,yRng) :
92    fairLstA \
93     = [[(x,d-x) for x in range(d+1)]
94                 for d in range(yRng)]
95    return fairLstA

97  fairLstATest \
98   = (fairListingA(5,5)
99       == [[(0, 0)]
100         , [(0, 1), (1, 0)]
101         , [(0, 2), (1, 1), (2, 0)]
102         , [(0, 3), (1, 2), (2, 1), (3, 0)]
103         , [(0, 4), (1, 3), (2, 2), (3, 1), (4, 0)]])
```

Go to Activity

ToC

Python & Haskell Tutorials

- Python tutorials:
  - Beginner's Python Tutorial

  - Python Programming
  - Non-Programmer's Tutorial for Python 3
  - Non-Programmer's Tutorial for Python 2.6
- Haskell Tutorials:
  - Haskell Wikibook
  - What I Wish I Knew When Learning Haskell
  - Haskell Meta-tutorial
  - Learn You a Haskell for Great Good
  - Real World Haskell

ToC

Commentary 3

3 DP Introduction, List Comprehensions

- Fibonacci sequence
- Recursive definition and simple recursive program
- Recursive but more efficient program
- Original implementation in Haskell (optional)
- Implementation in Python
- Edit Distance examples
- Edit Distance diagram construction — generating the LaTeX for the diagrams (optional)

ToC

5 Dynamic Programming

5.1 Fibonacci Sequence

- The Fibonacci Numbers or Sequence invented by Leonardo Fibonacci Pisano, who also popularised the Hindu-Arabic numeral system via his 1202 book Liber Abaci (Book of Calculations)
- Defined by the recurrence relation
- $F_n = F_{n-1} + F_{n-2}$
- $F_0 = 0, \quad F_1 = 1$ (or originally, $F_1 = 1, \quad F_2 = 1$)
- The Fibonacci numbers have lots of interesting properties
- Dr Ron Knott's Fibonacci Numbers
- In programming, often used to illustrate ideas about recurrence relations and recursion

Fibonacci Sequence — Python Recursive (1)

- Recursive definitions are recursive algorithms

```
def fibs1(n):
  return fibs1indent(n,0)

def fibs1indent(n,indent):
  indentStr = '␣'*indent
  print(indentStr + 'fibs(' + str(n) + ')')

  if n == 0 :
    return 0
  elif n == 1 :
    return 1
  else:
    return (fibs1indent(n - 2, indent + 2)
          + fibs1indent(n - 1, indent + 2))
```

```
Python3>>> fibs1(5)
fibs(5)
  fibs(3)
    fibs(1)
    fibs(2)
      fibs(0)
      fibs(1)
  fibs(4)
    fibs(2)
      fibs(0)
      fibs(1)
    fibs(3)
      fibs(1)
      fibs(2)
        fibs(0)
        fibs(1)
5
Python3>>>
```

- This take 15 calls to `fibs1()`, 5 calls to `fibs(1)`, 3 calls to `fibs(0)`

Fibonacci Sequence — Python Recursive (2)

- Recursion but remembering enough to avoid most

```
def fibs2(n):
  return fibs2indent(n,0)

def fibs2indent(n,indent,first=0,second=1):
  indentStr = '␣'*indent
  print(indentStr + 'fibs(' + str(n) + ')')

  if n == 0 :
    return [first]
  else:
    return ([first]
            + fibs2indent(n - 1, indent + 2
                          , second, first + second))
```

```
Python3>>> fibs2(5)
fibs(5)
  fibs(4)
    fibs(3)
      fibs(2)
        fibs(1)
          fibs(0)
[0, 1, 1, 2, 3, 5]
Python3>>>
```

- This takes 6 calls to `fibs2()`
- `fibs2()` $T(n) = T(n-1) + 1 = n + 1$

- `fibs1()` $T(n) = T(n-1) + T(n-2) + 1 = 2F_{n+1} - 1$
- `fibs1(1)` gets called F_n times
- `fibs1(0)` gets called F_{n-1} times
- So the recursion tree has $F_n + F_{n-1} = F_{n+1}$ leaves
- Since the tree is full it must have $2F_{n+1} - 1$ nodes
- (it is the sum of a geometric series)

5.1.1 Fibonacci Closed Form

- $F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}$
- Euler-Binet Formula
- A formula for Fib(n)
- ϕ is the Golden mean
- $\phi = \frac{1}{\phi - 1} = \frac{1+\sqrt{5}}{2}$
- Also known as the Golden ratio
- $\phi \stackrel{def}{=} \frac{a}{b} = \frac{a+b}{a}$

Proofs of Euler-Binet Formula

- ProofWiki: Euler-Binet Formula gives 4 proofs:

(1) Proof by induction — straightforward but doesn't really tell you how to get the formula in the first place

(2) Proof by matrix algebra — find the eigenvalues and eigenvectors of a matrix that generates the Fibnacci sequence

(3) Proof from the ProofWiki: Binet Form

$U_n = mU_{n-1} + U_{n-2}$ where $U_0 = 0$ and $U_1 = 1$

(4) Proof from the ProofWiki: Generating Function for Fibonacci Numbers

$G(z) = \frac{z}{1 - z - z^2}$

Matrix Algebra Proof of Euler-Binet Formula

- Let $A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$
- Then $A^n = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix}$
- Proof by induction
- $A^1 = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_2 & F_1 \\ F_1 & F_0 \end{pmatrix}$

- Assume $A^k = \begin{pmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{pmatrix}$
- Then $A \times A^k = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{pmatrix}$

  $= \begin{pmatrix} F_{k+1} + F_k & F_k + F_{k-1} \\ F_{k+1} & F_k \end{pmatrix} = \begin{pmatrix} F_{k+2} & F_{k+1} \\ F_{k+1} & F_k \end{pmatrix}$
- Demonstrate the eigenvalues of A are ϕ and $\hat{\phi} = 1 - \phi$
- Solve $\det(\lambda I - A) = 0$
- $\det\left(\lambda \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}\right) = 0$
- $\det \begin{pmatrix} \lambda - 1 & -1 \\ -1 & \lambda \end{pmatrix} = 0$
- $\lambda^2 - \lambda - 1 = 0$
- $ax^2 + bx + c = 0$ has roots $= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$
- Hence $\phi = \frac{1+\sqrt{5}}{2}$ and $\hat{\phi} = 1 - \phi = \frac{1-\sqrt{5}}{2}$
- and $A \begin{pmatrix} \phi \\ 1 \end{pmatrix} = \begin{pmatrix} \phi + 1 \\ \phi \end{pmatrix} = \phi \begin{pmatrix} \phi \\ 1 \end{pmatrix}$ as $\phi^2 - \phi - 1 = 0$
- and $A \begin{pmatrix} \hat{\phi} \\ 1 \end{pmatrix} = \begin{pmatrix} \hat{\phi} + 1 \\ \hat{\phi} \end{pmatrix} = \hat{\phi} \begin{pmatrix} \hat{\phi} \\ 1 \end{pmatrix}$ as $\hat{\phi}^2 - \hat{\phi} - 1 = 0$
- Hence $\begin{pmatrix} \phi \\ 1 \end{pmatrix}$ is an eigenvector of A with eigenvalue ϕ
- and $\begin{pmatrix} \hat{\phi} \\ 1 \end{pmatrix}$ is an eigenvector of A with eigenvalue $\hat{\phi}$
- We have to have $\det(\lambda I - A) = 0$ nonninvertible, singular
- Proof: assume $X \neq 0$, $AX = \lambda X$ and $(\lambda I - A)$ is invertible
- Then $X = IX = \left((\lambda I - A)^{-1}(\lambda I - A)\right) X$

  $= (\lambda I - A)^{-1} ((\lambda I - A)X)$

  $= (\lambda I - A)^{-1} 0$

  $= 0$ contradiction
- Hence $A^n \begin{pmatrix} \frac{\phi}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{pmatrix} = \phi^n \begin{pmatrix} \frac{\phi}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{pmatrix}$
- and $A^n \begin{pmatrix} \frac{\hat{\phi}}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{pmatrix} = \hat{\phi}^n \begin{pmatrix} \frac{\hat{\phi}}{\sqrt{5}} \\ \frac{1}{\sqrt{5}} \end{pmatrix}$
- Also $A^n \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} F_{n+1} \\ F_n \end{pmatrix}$

- Also $\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}\frac{\phi}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix} - \begin{pmatrix}\frac{\hat{\phi}}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix}$
- Hence $\begin{pmatrix}F_{n+1}\\F_n\end{pmatrix} = A^n\left(\begin{pmatrix}\frac{\phi}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix} - \begin{pmatrix}\frac{\hat{\phi}}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix}\right)$

  $= \phi^n\begin{pmatrix}\frac{\phi}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix} - \hat{\phi}^n\begin{pmatrix}\frac{\hat{\phi}}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix}$
- Hence $A^n\begin{pmatrix}\frac{\phi}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix} = \phi^n\begin{pmatrix}\frac{\phi}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix}$
- and $A^n\begin{pmatrix}\frac{\hat{\phi}}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix} = \hat{\phi}^n\begin{pmatrix}\frac{\hat{\phi}}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix}$
- Also $A^n\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}F_{n+1} & F_n\\ F_n & F_{n-1}\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}F_{n+1}\\F_n\end{pmatrix}$
- Also $\begin{pmatrix}1\\0\end{pmatrix} = \begin{pmatrix}\frac{\phi}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix} - \begin{pmatrix}\frac{\hat{\phi}}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix}$
- Hence $\begin{pmatrix}F_{n+1}\\F_n\end{pmatrix} = A^n\left(\begin{pmatrix}\frac{\phi}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix} - \begin{pmatrix}\frac{\hat{\phi}}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix}\right)$

  $= \phi^n\begin{pmatrix}\frac{\phi}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix} - \hat{\phi}^n\begin{pmatrix}\frac{\hat{\phi}}{\sqrt{5}}\\ \frac{1}{\sqrt{5}}\end{pmatrix}$

  $= \frac{1}{\sqrt{5}}\begin{pmatrix}\phi^n\cdot\phi - \hat{\phi}^n\cdot\hat{\phi}\\ \phi^n - \hat{\phi}^n\end{pmatrix}$

  $= \frac{1}{\sqrt{5}}\begin{pmatrix}\phi^{n+1} - \hat{\phi}^{n+1}\\ \phi^n - \hat{\phi}^n\end{pmatrix}$
- Hence $F_n = \frac{1}{\sqrt{5}}\left(\phi^n - \hat{\phi}^n\right)$

ToC

5.1.2 Fibonacci Alternative Calculation

- From How to Compute Fibonacci Numbers? and Fibonacci Formulae

```
fib (n + k) = fib (n-1) * fib k + fib n * fib (k+1)
```

- Proof by induction

```
fib (n + 2 + k)
  = fib (n+k) + fib (n+k+1)                    -- by fib
  = fib (n-1) * fib k + fib n * fib (k+1)
    + fib n * fib k + fib (n+1) * fib (k+1)    -- by hyp
  = fib (n+1) * fib k + fib (n+2) * fib (k+1)  -- by fib
```

- We now derive a method to compute `fib` in $O(\log n)$

```
fib (2n+1) = (fib n)^2 + (fib (n+1))^2   -- (1)
fib (2n+2) = fib n * fib (n+1) + fib (n+1) * fib (n+2)
  = fib n * fib (n+1) + fib (n+1) * (fib n + fib (n+1))
  = 2 * fib n * fib (n+1) + (fib (n+1))^2  -- (2)
fib 2n = 2 * fib n * fib (n+1) - (fib n)^2 -- (3)
  -- (3) = (2) - (1)
```

```
fib2v :: Int -> (Int,Int)
fib2v 0 = (0,1)
fib2v n
  | n `mod` 2 == 0  = (c,d)
  | otherwise       = (d,c+d)
  where
  (a,b) = fib2v (n `div` 2)
  c     = 2 * a * b - a * a
  d     = a * a + b * b
```

ToC

5.2 DP Formulation

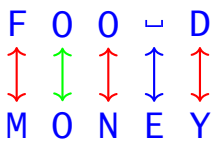
- Write a recursive algorithm for the problem
- Build solutions to the recurrence from the bottom up
  - Identify subproblems
  - Choose a data structure to memoize intermediate results
  - Identify dependencies between subproblems
  - Find an evaluation order so that each subproblem comes after the subproblems it depends on.
  - Implement the algorithm for the evaluation order

ToC

5.3 Edit Distance

5.4 Edit Distance Definitions

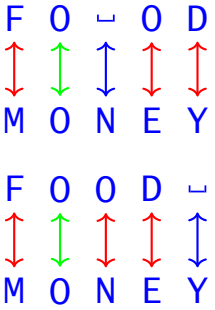
- **Edit Distance** or **Levenshtein Distance** is the minimum number of letter insertions, deletions and substitutions required to transform one word into another.
- Example FOOD → MOOD → MOND → MONED → MONEY
- A better way of displaying the transformation is in the next diagram



- **Exercise** find two more 4 step transformations. Are there more ?

Edit Distance — Example 1

- Two further transformations of 4 step transformations of FOOD into MONEY are



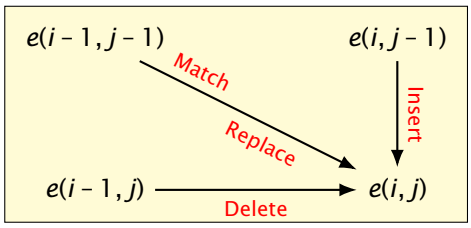
- How do we find such transformations in general ?

ToC

5.5 Edit Distance — Recursive Algorithm

- **Notation**
- s and t are names for the start and target strings
- Let $s(i), t(j)$ denote the prefixes of s and t of lengths i and j
- Let $s[i], t[j]$ denote the letters at index i and j of s and t — remember that Python and Haskell index from 0 not 1
- Let $e(i, j)$ denote the edit distance between prefixes $s(i)$ and $t(j)$
- We now describe recursively how to transform $s(i)$ to $t(j)$

Edit Distance Table Dependencies



- **Deletion** — transform $s(i-1)$ to $t(j)$ and delete the last character of $s(i)$ which is $s[i-1]$
- **Insertion** — transform $s(i)$ to $t(j-1)$ and insert the next character required for $t(j)$ which is $t[j-1]$
- **Match/Replace** — transform $s(i-1)$ to $t(j-1)$ and match or replace the last character of $s(i)$ with the last character of $t(j)$ that is, $s[i-1]$ with $t[j-1]$
- Note: you have to be careful whether you are using 0 or 1 based indexing
- The edit distance will be the minimum of the three possibilities
- This also determines the predecessor nodes (1 to 3)

Edit Distance — Base cases & Table Display

- The shortest way to convert $s(i)$ to an empty string ϵ is by i deletions
- The shortest way to convert an empty string ϵ to $t(j)$ is by j insertions
- **Table Display** — note that in the Haskell (but not the Python), the edit distance table is calculated with (i, j) indexing rows and columns but table is displayed with (j, i) indexing rows and columns
- That means that in the table, the row characters are the source and the column characters are the target, while in the display the column characters are the source.
- This seems to be the convention is most texts but may lead to some tricky thinking when formatting the diagrams
- **Note** The next version will change this design decision.

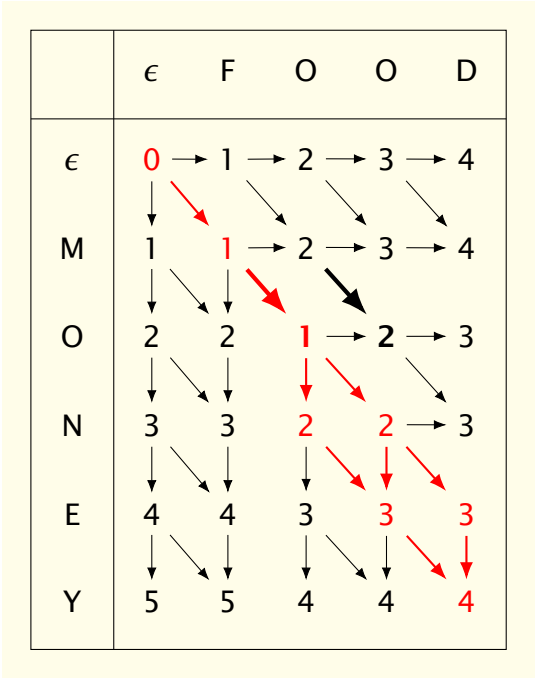
Edit Distance — Recursive Definition

$$e(i,j) = \begin{cases} i & \text{if } j = 0 \\ j & \text{if } i = 0 \\ \min\begin{cases} e(i-1,j)+1 \\ e(i,j-1)+1 \\ e(i-1,j-1)+ \\ \quad \text{if } s[i-1] \neq t[j-1] \text{ then 1 else 0} \end{cases} & \text{otherwise} \end{cases}$$

- For simplicity we have assumed the cost of deletion, insertion, or replacement is 1 and the cost of a match is 0 (made more general in the implementation below)
- the running time of the algorithm is exponential in the length of s and t — but we do not care since are going to implement it bottom up.
- The implementation below also provides the graph of the transformations by calculating the predecessors to each node in the table.

Edit Distance — Example 1 Table

edDistTblFoodMoney



ToC

5.6 Edit Distance — Implementation

- If we calculate the edit distance table in the standard row-major order — row by row, each row from left to right
- then when we reach an entry, the entries it depends on are already available
- We develop the algorithm below in both Haskell and Python
- We have used list comprehensions in both for iterations — so they should look fairly similar
- Note both implementations could be optimised further
- Health Warning you are not expected to write the code for this problem — see the comments on Python_activity_5.11.py — this is more of a reading/comprehension exercise

5.7 Edit Distance — Haskell Implementation

```haskell
module M269TutorialDynamicProgCmntry2023 where
  import Data.List
  import Data.Maybe
```

1. A Haskell script starts with a module header which starts with the reserved identifier, module followed by the module name, M269TutorialDynamicProgCmntry2023
2. The module name must start with an upper case letter and is the same as the file name (without its extension of .lhs)
3. Haskell uses layout (or the off-side rule) to determine scope of definitions, similar to Python

4. The body of the module follows the reserved identifier `where` and starts with two `import` declarations
5. These import the built-in libraries `Data.List` and `Data.Maybe`
6. We use the `sort` function from `Data.List`.
7. The `Maybe` datatype from `Data.Maybe` will be used at the end of this script to implement the graphs with data.

Edit Distance — Haskell Implementation — Type Declarations

```
5  type Dist = Int
6  type Pred = (Int,Int)
7  type EditDistCell = (Dist, [Pred])
8  type EditDistTable = [[EditDistCell]]
```

- The reserved identifier `type` introduces type synonym declarations to improve readability
- The finite-precision integer type `Int` covers the range [-2^{63} = -9223372036854775808, $2^{63}-1$ = 9223372036854775807] (machine dependent)
- `(t1,t2)` is the type of a pair of `t1` and `t2`
- `[t3]` is the type of a list of elements all of type `t3`
- `[[t3]]` type of a list of lists of elements of type `t3`

Edit Distance — Haskell Implementation — Service Functions

```
10 edCellDist :: EditDistCell -> Dist
11 edCellDist  (x, ps) = x

13 edCellPreds :: EditDistCell -> [Pred]
14 edCellPreds (x, ps) = ps
```

- `edCellDist` and `edCellPreds` take apart an `EditDistCell`

```
f :: t1 -> t2
```

- The above is a type signature for the variable `f`
- This specifies `f` has a function type which takes an element of type `t1` and returns an element of type `t2`
- The definition of `f` may be given without a type signature, in which case the system will infer the most general type
- Note that every function takes exactly one input and returns one output

```
16 getEdCell :: EditDistTable -> (Int,Int) -> EditDistCell
17 getEdCell edTbl (i,j) = edTbl!!i!!j

19 getEdDist :: EditDistTable -> (Int,Int) -> Dist
20 getEdDist  edTbl (i,j)
21   = edCellDist  (getEdCell edTbl (i,j))

23 getEdPreds :: EditDistTable -> (Int,Int) -> [Pred]
24 getEdPreds edTbl (i,j)
25   = edCellPreds (getEdCell edTbl (i,j))
```

- Function application is denoted by juxtaposition, is left associative and is more tightly binding than (almost) anything else

  - write `f x` and not `f (x)`
  - say `f x` as f applied to x
  - `f x y` means `(f x) y`

  This notational convention has huge advantages — discuss and also see Currying and Functional Programming in 5 Minutes

- To be consistent, the function type arrow `(->)` is right associative
  - `f :: a -> b -> c` means `f :: a -> (b -> c)`
- Lists are denoted `[1,2,3]`, the empty list `[]`
- `(:)` prefixes an element to a list, `1:[2,3] == [1,2,3]`
- Parentheses over-ride precedence
- `(!!)` is the list indexing operator, 0-origin
  - `[1,2,3]!!1 == 2`

Edit Distance — Haskell Implementation — Costs

```
27  -- Cost of deletion, insertion, substitution
28  delC, insC, subC :: Dist
29  delC = 1
30  insC = 1
31  subC = 1
```

- The cost of deletion, insertion and replacement are defined here
- The cost of a match is assumed to be 0

Edit Distance — Haskell Implementation — Algorithm

```
33  editDistTblDP :: String -> String -> EditDistTable
34  editDistTblDP s t
35    = edTbl
36      where
37      edTbl = [ [ edTblCell (i,j) | j <- [0..length t] ]
38                | i <- [0..length s] ]
39      edTblCell (0,0) = (0,[])
40      edTblCell (i,0) = (i * delC, [(i-1,0)])
41      edTblCell (0,j) = (j * insC, [(0,j-1)])
42      edTblCell (i,j)
43        = (minD, preds)
44          where
45          possPreds = [(delD, (i-1,j))
46                      ,(insD, (i,j-1))
47                      ,(subD, (i-1,j-1))
48                      ]
49          delD = getEdDist edTbl (i-1,j) + delC
50          insD = getEdDist edTbl (i,j-1) + insC
51          subD = getEdDist edTbl (i-1,j-1)
52                 + (if s!!(i-1) == t!!(j-1) then 0 else subC)
53          minD = minimum [delD, insD, subD]
54          preds = [(i,j) | (x,(i,j)) <- possPreds, x == minD]
```

Edit Distance — Haskell Implementation — Examples

```
56  edTblTF = editDistTblDP "TREES" "FOREST"
57  edTblTrTF
58    = transpose (editDistTblDP "TREES" "FOREST")

60  edTblAA = editDistTblDP "ALGORITHM" "ALTRUISTIC"
```

```
61  edTblTrAA
62    = transpose (editDistTblDP "ALGORITHM" "ALTRUISTIC")

64  edTblFM = editDistTblDP "FOOD" "MONEY"
65  edTblTrFM
66    = transpose (editDistTblDP "FOOD" "MONEY")
```

- We use `transpose` since the edit distance table is displayed with the characters of the start string in columns and the target string in the rows with *i* indexing the columns and *j* the rows
- This may lead to some tricky thinking when formatting the diagram but it seems to be the convention.

Edit Distance — Haskell Implementation (2)

```
67  editDistTblDP01 :: String -> String -> EditDistTable
68  editDistTblDP01 s t
69    = edTbl
70      where
71      edTbl = [ [ setEdTblCell (edTbl,s,t) (i,j)
72                | j <- [0..length t] ]
73                | i <- [0..length s] ]
```

- Defining a subsidiary function `setEdTblCell` avoids having nested `where` clauses
- `setEdTblCell` takes the edit distance table `edTbl`, the start and target strings, and a pair of table indices and returns the table cell.
- The list comprehension definition here is recursive and works because the entries are calculated in row-major order.

```
75  setEdTblCell :: (EditDistTable, String, String)
76                  -> (Int, Int) -> EditDistCell
77  setEdTblCell (edTbl,s,t) (0,0) = (0,[])
78  setEdTblCell (edTbl,s,t) (i,0) = (i * delC, [(i-1,0)])
79  setEdTblCell (edTbl,s,t) (0,j) = (j * insC, [(0,j-1)])
80  setEdTblCell (edTbl,s,t) (i,j)
81    = (minD, preds)
82      where
83      possPreds = [(delD, (i-1,j))
84                  ,(insD, (i,j-1))
85                  ,(subD, (i-1,j-1))
86                  ]
87      delD = getEdDist edTbl (i-1,j) + delC
88      insD = getEdDist edTbl (i,j-1) + insC
89      subD = getEdDist edTbl (i-1,j-1)
90             + (if s!!(i-1) == t!!(j-1) then 0 else subC)
91      minD = minimum [delD, insD, subD]
92      preds = [(i,j) | (x,(i,j)) <- possPreds, x == minD]
```

```
94  edTblTF01 = editDistTblDP01 "TREES" "FOREST"
95  edTblTrTF01
96    = transpose (editDistTblDP01 "TREES" "FOREST")

98  edTblAA01 = editDistTblDP "ALGORITHM" "ALTRUISTIC"
99  edTblTrAA01
100   = transpose (editDistTblDP01 "ALGORITHM" "ALTRUISTIC")

102 edTblFM01 = editDistTblDP "FOOD" "MONEY"
103 edTblTrFM01
104   = transpose (editDistTblDP01 "FOOD" "MONEY")
```

- `transpose` is used to switch the rows and columns for display
- This may lead to some tricky thinking about formatting the diagram

ToC

5.8 Edit Distance — Python Implementation

- The Python file Python_activity_5.11.py for Python activity 5.11 contains the code for the Dynamic Programming solution for the Levenshtein Edit Distance problem.
- It uses a double for loop to do the equivalent calculations of the edit distance table
- TODO rewrite using Python list comprehensions and compare with the Haskell version
- Note that the bulk of the code in 5.11 is to format the output and display one path.
- The Haskell to generate the edit table diagrams in these notes is available but not given here since it generates LaTeX TikZ/PGF which is not part of this course.
- The Python code below is based on the Haskell code (elsewhere in this document)
- **Health Warning** the code here should be regarded as near pseudo-code since a few of the Haskell features do not translate directly
- In particular, the Haskell code depended to some extent on the default lazy evaluation strategy which is a particular way of implementing non-strict evaluation
- Strict evaluation evaluates arguments of functions (except in special cases or by special constructs)
- Non-strict evaluation evaluates argument to functions at most once and if possible not at all — it evaluates on need
- Python does struct evaluation unless you use yield or other generators
- Haskell is non-strict unless you use strictness annotations
- This means the Python code below ~~may~~ need modifying
- Service functions
- Edit Distance Table is an array of cells
- Edit Distance cell is a pair of distance and a list of predecessors

```
21  def edCellDist(cell) :
22    return cell[0]
23
24  def edCellPreds(cell) :
25    return cell[1]
```

- functions to get specific cells
- edTbl is a list of list of cells

```
30  def getEdCell(edTbl, i, j) :
31    return edTbl[i][j]
32
33  def getEdDist(edTbl, i, j) :
34    return edCellDist(getEdCell(edTbl, i, j))
35
36  def getEdPreds(edTbl, i, j) :
37    return edCellPreds(getEdCell(edTbl, i, j))
```

- Cost of deletion, insertion, substitution

```
41  delC = 1
42  insC = 1
43  subC = 1
```

- Calculating the Edit Distance Table
- Calculate the entry in each cell
- setEdTblCell takes the from string, s, the to string, t, the cell indices,i, j, and returns a cell

```
52  def setEdTblCell(edTbl, s, t, i, j) :
53    if i == 0 and j == 0 :
54      return (0,[])
55    elif j == 0 :
56      return (i * delC, [(i-1,0)])
57    elif i == 0 :
58      return (j * insC, [(0,j-1)])
59    else :
60      possPreds = [(delD, (i-1,j))
61                   ,(insD, (i,j-1))
62                   ,(subD, (i-1,j-1))
63                  ]
64      delD = getEdDist(edTbl, i-1,j  ) + delC
65      insD = getEdDist(edTbl, i,  j-1) + insC
66      subD = (getEdDist(edTbl, i-1,j-1)
67             + (0 if s[i-1] == t[j-1] else subC))
68      minD = min([delD, insD, subD])
69      preds = [(i,j) for (x,(i,j)) in possPreds if x == minD]
```

- Calculate the Edit Distance Table in row-major order

```
73  def editDistTblDP (s, t) :
74    edTbl = [ [ setEdTblCell (edTbl, s, t, i, j)
75              for j in range(len(t) + 1) ]
76              for i in range(len(s) + 1) ]
77
78    return edTbl
```

- Note that `setEdTblCell` and `editDistTblDP` are not valid Python since the evaluation strategy of Python (eager or strict evaluation) does not permit that style of referencing
- See either `Python_activity_5.11` or `EditDistance2025J.py` and below

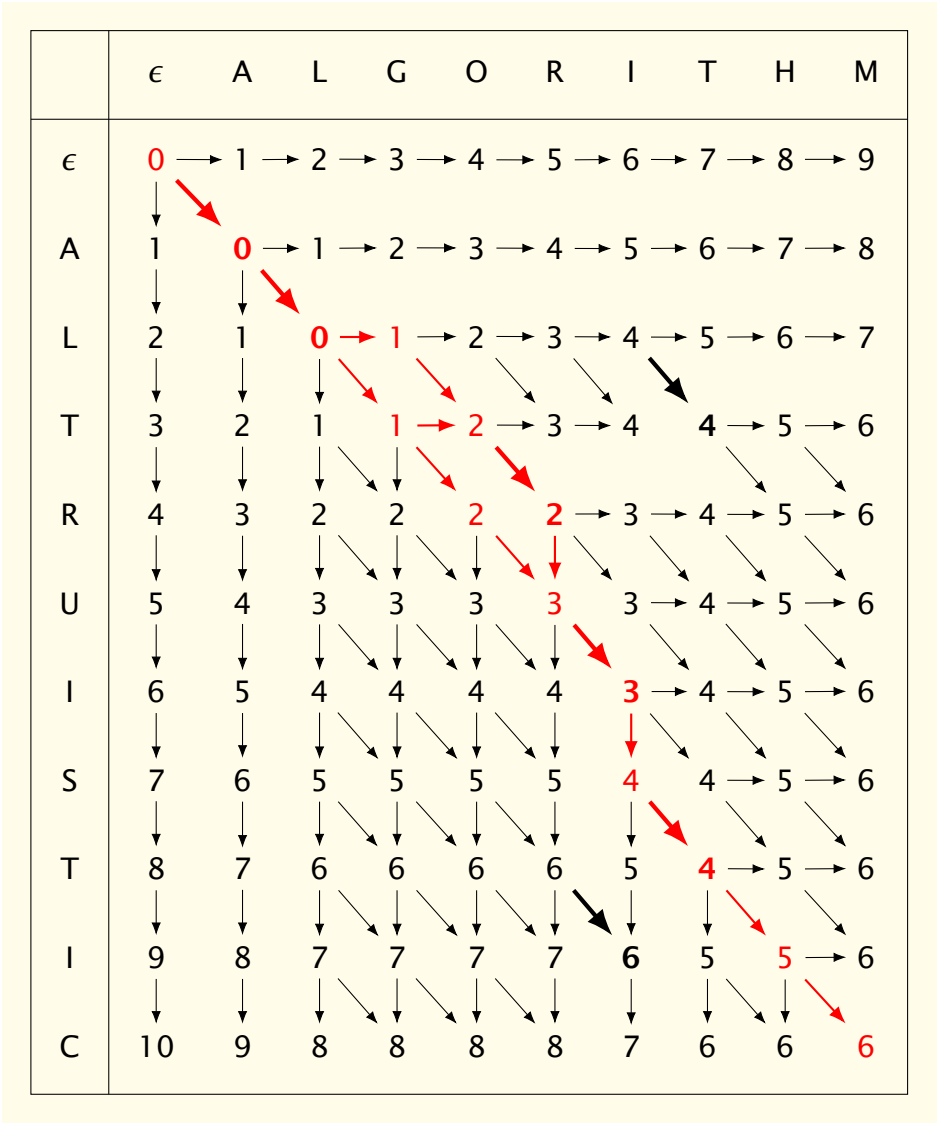
ToC

5.9 Edit Distance Examples

- For `editDistance "ALGORITHM" "ALTRUISTIC"`
  - What is the edit distance ?
  - How many different routes are there ?
  - Give two examples laid out as in the `editDistance "FOOD" "MONEY"` example
- repeat the above with `editDistance "TREES" "FOREST"` (harder!)

Memoization Table for `editDistance "ALGORITHM" "ALTRUISTIC"`

edDistTblAlgorithmAltruistic

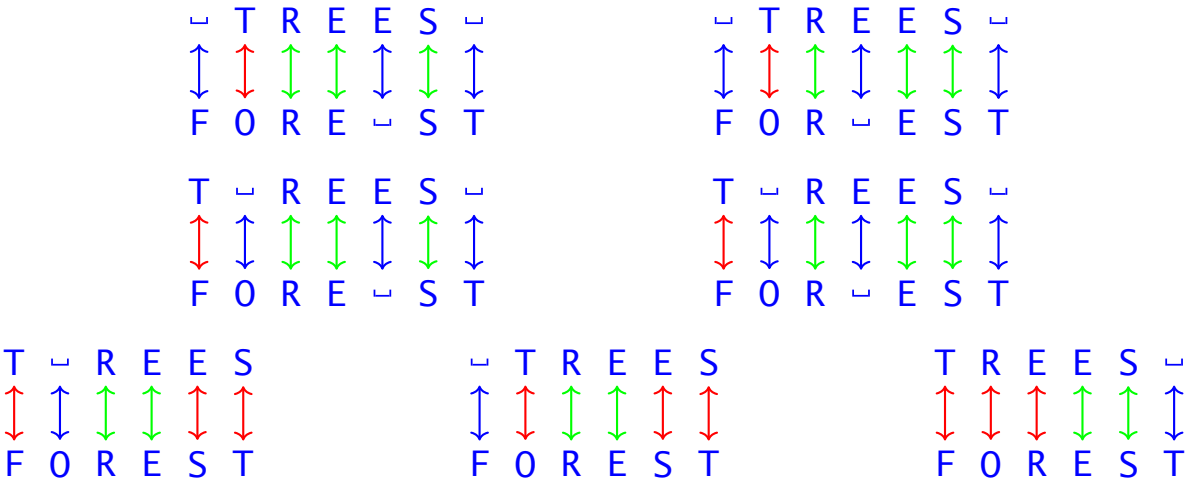


Memoization Table for `editDistance "TREES" "FOREST"`

edDistTblTreesForest

	ϵ	T	R	E	E	S
ϵ	0	1	2	3	4	5
F	1	1	2	3	4	5
O	2	2	2	3	4	5
R	3	3	2	3	4	5
E	4	4	3	2	3	4
S	5	5	4	3	3	3
T	6	5	5	4	4	4

- Alternative ways of representing the collection of transformation operations were given in subsection 5.4
- Here is a representation of the edit distance graph with match, delete/insert, and replace color arrows



- The transformations can also be represented as a sequence of operations
- Note that the collection of operations could be used in any order — the end result is the same

TREES
↓insert
FTREES
↓replace
FOREES
↓delete
FORES
↓insert
FOREST

TREES
↓delete
TRES
↓replace
ORES
↓insert
FORES
↓insert
FOREST

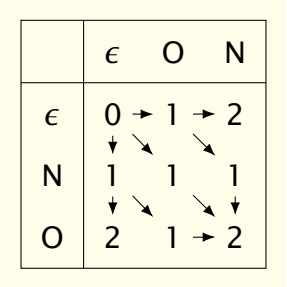
ToC

5.10 Edit Distance Diagram Construction

- `editDistTblDP` takes two strings and returns an `EditDistTable`
- The diagrams illustrate `EditDistTable` with the cells laid out in a table with arrows
- The diagrams use the markup language LaTeX with the PGF/TikZ package
- Note: the Haskell to LaTeX and TikZ/PGF diagram code is not part of M269 and is only here for interest
- Here are several small examples

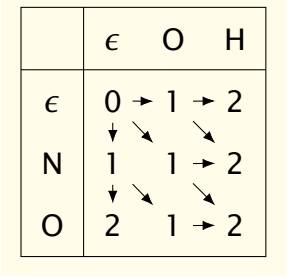
Memoization Table for `editDistTblDP "ON" "NO"`

edDistTblOnNo



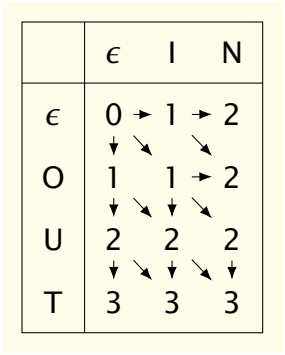
Memoization Table for `editDistTblDP "OH" "NO"`

edDistTblOhNo



Memoization Table for `editDistTblDP "IN" "OUT"`

edDistTblInOut



LaTeX Code for `editDistTblDP "OH" "NO"`

- The LaTeX code is in a `tikzpicture` environment
- LaTeX TikZ style definitions
- Styles for nodes and arrows
- Edit path nodes and arrows are red and thick
- Free substitutions (matches) have bold nodes and ultra thick arrows

```
    [ePath/.style={red,thick}
    ,frSub/.style={font=\bfseries}
    ,efrSb/.style={ePath,frSub}
    ,orArr/.style={-Latex}
    ,frArr/.style={orArr,ultra thick}
    ,eoArr/.style={ePath,orArr}
    ,efArr/.style={ePath,frArr}
    ]
```

- TikZ Matrix code
- The double rows/columns is a hack to ease the frame placement
- See discussion at end

```
% TikZ Matrix code
\matrix (edMat) [matrix of nodes
                ,nodes in empty cells
                ,ampersand replacement=\&
                ,nodes={minimum size=\STtextNodeWidth}]
{
  \&            \&  \&             \&  \&           \&  \&           \& \\
  \&            \&  \&  $\epsilon$ \&  \&         O \&  \&         H \& \\
  \&            \&  \&             \&  \&           \&  \&           \& \\
  \&  $\epsilon$ \&  \&           0 \&  \&         1 \&  \&         2 \& \\
  \&            \&  \&             \&  \&           \&  \&           \& \\
  \&          N \&  \&           1 \&  \&         1 \&  \&         2 \& \\
  \&            \&  \&             \&  \&           \&  \&           \& \\
  \&          O \&  \&           2 \&  \&         1 \&  \&         2 \& \\
  \&            \&  \&             \&  \&           \&  \&           \& \\
};
```

- LaTeX TikZ Arrows code
- All arrows are given the `orArr` style
- TODO: modify the data structure and code to add the edit paths and free substitutions

```
% TikZ Arrows code
\draw[orArr] (edMat-4-4) -- (edMat-4-6);
\draw[orArr] (edMat-4-6) -- (edMat-4-8);
```

```
\draw[orArr] (edMat-4-4) -- (edMat-6-4);
\draw[orArr] (edMat-4-4) -- (edMat-6-6);
\draw[orArr] (edMat-6-6) -- (edMat-6-8);
\draw[orArr] (edMat-4-6) -- (edMat-6-8);

\draw[orArr] (edMat-6-4) -- (edMat-8-4);
\draw[orArr] (edMat-6-4) -- (edMat-8-6);
\draw[orArr] (edMat-8-6) -- (edMat-8-8);
\draw[orArr] (edMat-6-6) -- (edMat-8-8);
```

- LaTeX TikZ Frame code

```
% TikZ frame code
\draw (edMat-1-1.center) -- (edMat-1-9.center)
       -- (edMat-9-9.center) -- (edMat-9-1.center)
       -- cycle;
\draw (edMat-3-1.center) -- (edMat-3-9.center);
\draw (edMat-1-3.center) -- (edMat-9-3.center);
```

Generating PGF/TikZ from the Edit Table

- We generate the PGF/TikZ from `editDistTblDP`
- By convention we display the transpose of the table
- Remember that the transpose edit cells have lists of predecessors indexed in the original table
- This makes the computation more awkward but it appears to be the convention
- This computation does not include the edit paths nor the free substitutions
- That would be better included in the edit table computation
- TODO: Include edit paths and free substitutions in the edit table computation

Offsets, Scales and Constants

```
iOffset = 4  -- offset for original string
jOffset = 4  -- offset for target string
iScale  = 2  -- scale for original string
jScale  = 2  -- scale for target string

spc = ' '  -- space char
nl  = '\n' -- new line char
```

TikZ Arrow Code (1)

- The code for drawing arrows between nodes is built on the code for one arrow.
- `edCellTrPredToArrow` takes a pair of style name and matrix name,
- a pair of offset and scale data for original and target string,
- the coordinates of the destination node and a predecessor
- and returns the TikZ code for the arrow

```
edCellTrPredToArrow (styleName,matName)
  ((iOff,iScl),(jOff,jScl)) (j,i) (a,b)
 = "\\draw" ++ "[" ++ styleName ++ "]" ++ [spc]
   ++ "(" ++ matName ++ "-"
   ++ show (jOff + jScl*b)
     ++ "-" ++ show (iOff + iScl*a) ++ ")"
   ++ "␣--␣"
   ++ "(" ++ matName ++ "-"
```

```
    ++ show (jOff + jScl*j)
      ++ "-" ++ show (iOff + iScl*i) ++ ");"
    ++ "\n"
```

TikZ Arrow Code (2)

- We use edCellTrPredToArrow to build a function to take a transposed edit table and return all the arrows as a TikZ string
- edCellTrPredToArrow01 is a helper function to avoid repeat typing
- edCellTrPredsToArrows01 takes the list of predecessors in a cell and returns the arrows (as a string)
- edCellTrToArrows01 takes a cell and returns the arrows
- edRowTrToArrows01 takes a row and returns the arrows
- edTblTrToArrows01 takes the complete table and returns the arrows

```
edCellTrPredToArrow01
 = edCellTrPredToArrow
     ("orArr","edMat")
     ((iOffset,iScale),(jOffset,jScale))

edCellTrPredsToArrows01  (j,i) ps
 = concat (map (edCellTrPredToArrow01 (j,i)) ps)

edCellTrToArrows01  (j,i) (d,ps)
 = edCellTrPredsToArrows01 (j,i) ps

edRowTrToArrows01 j cs
 = concat [edCellTrToArrows01 (j,i) c
          | (i,c) <- zip [0..length cs - 1] cs]
          ++ "\n"

edTblTrToArrows01 edTblTr
 = concat [edRowTrToArrows01 j cs
          | (j,cs) <- zip [0..length edTblTr - 1]
                         edTblTr]
```

```
-- Test Edit Table to Arrows
edTblTrToArrows01TF
 = putStr (edTblTrToArrows01 edTblTrTF)

edTblTrToArrows01AA
 = putStr (edTblTrToArrows01 edTblTrAA)

edTblTrToArrows01FM
 = putStr (edTblTrToArrows01 edTblTrFM)
```

TikZ Matrix Code (1)

- edCellTrDistToNode takes a distance and returns a node string
- strToNode takes a string and returns a node string
- **Health Warning** the code below needs rewriting to make it easier to read

```
ndWidth = 13 -- string width of a node
-- nWidth = (length zStr + 1) + 2
-- spc = ' ' -- spc is defined above

ampRepStr = "\\&"

zStr = "$\\epsilon$"

matrixPreamble nm ampRepStr
```

```
  = "\\matrix␣(" ++ nm ++ ")␣" ++ "[matrix␣of␣nodes"
    ++ "\n" ++ nSpcs
    ++ ",nodes␣in␣empty␣cells"
    ++ "\n" ++ nSpcs
    ++ ",ampersand␣replacement=" ++ ampRepStr
    ++ "\n" ++ nSpcs
    ++ ",nodes={minimum␣size=\\STtextNodeWidth}]"
    ++ "\n"
    where
    nSpcs = replicate (11 + length nm) spc
```

```
-- PGF/TikZ Matrix functions

edCellTrDistToNode :: Int -> String
edCellTrDistToNode d
 = replicate n spc ++ dStr ++ [spc]
   where
   dStr = show d
   n    = ndWidth - 1 - length dStr

strToNode str
 = replicate n spc ++ str ++ [spc]
   where
   n = ndWidth - 1 - length str
```

TikZ Matrix Code (2)

```
strToPrefixRow str
 = "␣␣" ++ ampRepStr
   ++ replicate ndWidth spc ++ ampRepStr
   ++ "␣␣" ++ ampRepStr
   ++ replicate ndWidth spc ++ ampRepStr
   ++ concat ["␣␣" ++ ampRepStr
              ++ replicate ndWidth spc ++ ampRepStr
             | c <- str]
   ++ "␣\\\\\n"
   ++ "␣␣" ++ ampRepStr
   ++ replicate ndWidth spc ++ ampRepStr
   ++ "␣␣" ++ ampRepStr
   ++ replicate (ndWidth - 11) spc
      ++ "$\\epsilon$␣"++ ampRepStr
   ++ concat ["␣␣" ++ ampRepStr
              ++ replicate (ndWidth - 2) spc
                 ++ [c] ++ [spc] ++ ampRepStr
             | c <- str]
   ++ "␣\\\\\n"
```

```
strToSuffixRow str
 = "␣␣" ++ ampRepStr
   ++ replicate ndWidth spc ++ ampRepStr
   ++ "␣␣" ++ ampRepStr
   ++ replicate ndWidth spc ++ ampRepStr
   ++ concat ["␣␣" ++ ampRepStr
              ++ replicate ndWidth spc ++ ampRepStr
             | c <- str]
   ++ "␣\\\\\n"

colNoTrToPrefixCol zStr str j
 = "␣␣" ++ ampRepStr ++
     (if j == 0
     then replicate (ndWidth - 1 - length zStr) spc
          ++ zStr ++ [spc]
     else replicate (ndWidth - 2) spc
          ++ [str!!(j - 1)] ++ [spc])
   ++ ampRepStr
```

```
edCellTrToNodes (d,ps)
 = "␣␣" ++ ampRepStr
   ++ edCellTrDistToNode d ++ ampRepStr

edRowTrToNodes zStr str j cs
 = "␣␣" ++ ampRepStr
```

```
    ++ replicate ndWidth spc ++ ampRepStr
    ++ concat ["␣␣" ++ ampRepStr
               ++ replicate ndWidth spc ++ ampRepStr
              | c <- cs]
    ++ "␣\\\\\n"
    ++ colNoTrToPrefixCol zStr str j
    ++ concat [edCellTrToNodes c | c <- cs]
    ++ "␣\\\\\n"

edTblTrToNodes zStr str edTblTr
 = concat [edRowTrToNodes zStr str j cs
          | (j,cs) <- zip [0..length edTblTr - 1]
                          edTblTr]
```

```
-- Test usage
edTblTrToNodesTF
 = putStr (edTblTrToNodes zStr "FOREST" edTblTrTF)
```

TikZ Matrix Code (3)

```
edTblTrToMatrix nm zStr s t
 = matrixPreamble nm ampRepStr
   ++ "{\n"
   ++ strToPrefixRow s
   ++ edTblTrToNodes zStr t edTblTr
   ++ strToSuffixRow s
   ++ "};\n"
   where
   edTblTr = transpose (editDistTblDP s t)
```

```
-- Test code for entire matrix
edTblTrToMatrixTF
 = putStr (edTblTrToMatrix "edMat" zStr "TREES" "FOREST")

edTblTrToMatrixAA
 = putStr (edTblTrToMatrix "edMat" zStr "ALGORITHM" "ALTRUISTIC")

edTblTrToMatrixFM
 = putStr (edTblTrToMatrix "edMat" zStr "FOOD" "MONEY")
```

TikZ Picture Code (1)

```
-- Construction of tikzpicture

tikzPreamble
 = "\\begin{tikzpicture}" ++ [nl]
   ++ "␣␣[ePath/.style={red,thick}" ++ [nl]
   ++ "␣␣,frSub/.style={font=\\bfseries}" ++ [nl]
   ++ "␣␣,efrSb/.style={ePath,frSub}" ++ [nl]
   ++ "␣␣,orArr/.style={-Latex}" ++ [nl]
   ++ "␣␣,frArr/.style={orArr,ultra␣thick}" ++ [nl]
   ++ "␣␣,eoArr/.style={ePath,orArr}" ++ [nl]
   ++ "␣␣,efArr/.style={ePath,frArr}" ++ [nl]
   ++ "␣␣]"

tikzEnd
 = "\\end{tikzpicture}" ++ [nl]
```

TikZ Picture Code (2)

```
tikzFrmCoord nm loc i j
 = "(" ++ nm ++ "-" ++ show j ++ "-"
    ++ show i ++ "." ++ loc ++ ")"

tikzFrame nm loc s t
 = "\\draw␣" ++ (tikzFrmCoord nm loc 1 1) ++ "␣--␣"
      ++ (tikzFrmCoord nm loc xMax 1)
      ++ [nl] ++ dSpcs ++ "␣--␣"
      ++ (tikzFrmCoord nm loc xMax yMax) ++ "␣--␣"
```

```
299         ++ (tikzFrmCoord nm loc 1 yMax)
300         ++ [nl] ++ dSpcs ++ " -- cycle;" ++ [nl]
301       ++ "\\draw " ++ (tikzFrmCoord nm loc 1 (jScale + 1))
302         ++ " -- " ++ (tikzFrmCoord nm loc xMax (jScale + 1))
303         ++ ";" ++ [nl]
304       ++ "\\draw " ++ (tikzFrmCoord nm loc (iScale + 1) 1)
305         ++ " -- " ++ (tikzFrmCoord nm loc (iScale + 1) yMax)
306         ++ ";" ++ [nl]
307       where
308         xMax = iOffset + 1 + iScale * (length s)
309         yMax = jOffset + 1 + jScale * (length t)
310         dSpcs = replicate 6 spc
```

TikZ Picture Code (3)

```
312 edTblTrToTikz nm loc zStr s t
313  = tikzPreamble ++ [nl]
314    ++ [nl] ++ "% TikZ Matrix code" ++ [nl]
315    ++ edTblTrToMatrix nm zStr s t ++ [nl]
316    ++ [nl] ++ "% TikZ Arrows code" ++ [nl]
317    ++ edTblTrToArrows01 edTblTr ++ [nl]
318    ++ [nl] ++ "% TikZ frame code" ++ [nl]
319    ++ tikzFrame nm loc s t ++ [nl]
320    ++ tikzEnd
321    where
322      edTblTr = transpose (editDistTblDP s t)
```

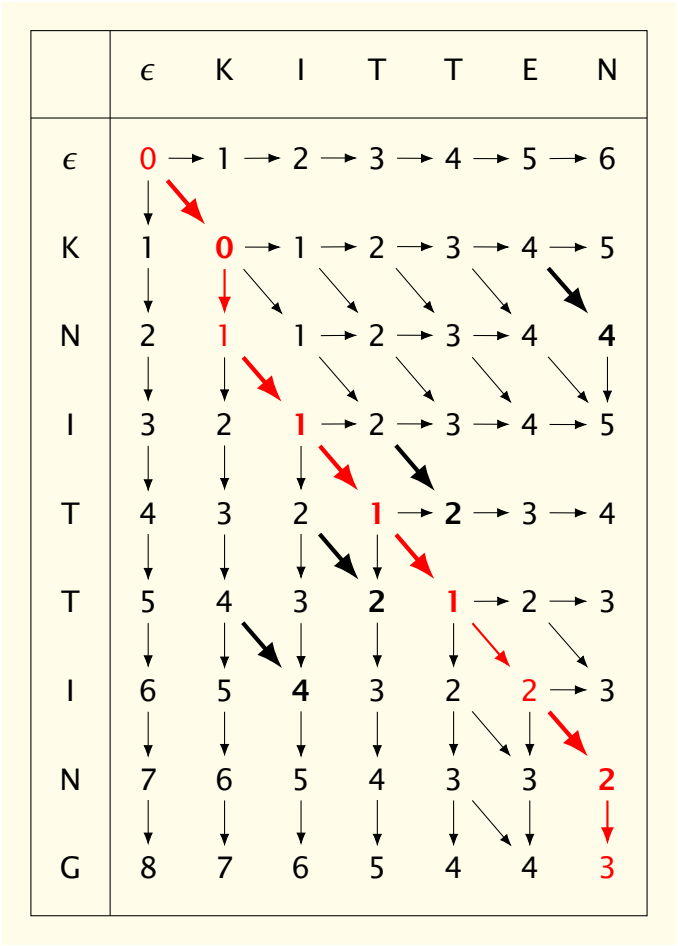
Test Code Matrix Frame

```
324 -- Test code for matrix frame lines

326 tikzFrameFM
327  = putStr (tikzFrame "edMat" "center"
328            "FOOD" "MONEY")

330 tikzFrameAA
331  = putStr (tikzFrame "edMat" "center"
332            "ALGORITHM" "ALTRUISTIC")

334 tikzFrameTF
335  = putStr (tikzFrame "edMat" "center"
336            "TREES" "FOREST")
```

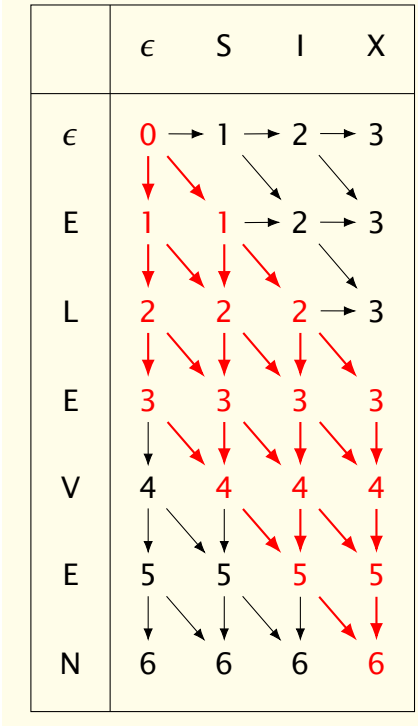
Test Code TikZ Picture

```
338 -- Text code for tikzpicture

340 edTblTrToTikzFM
341  = putStr (edTblTrToTikz "edMat" "center" zStr
342            "FOOD" "MONEY")

344 edTblTrToTikzAA
345  = putStr (edTblTrToTikz "edMat" "center" zStr
346            "ALGORITHM" "ALTRUISTIC")

348 edTblTrToTikzTF
349  = putStr (edTblTrToTikz "edMat" "center" zStr
350            "TREES" "FOREST")

352 edTblTrToTikzKK
353  = putStr (edTblTrToTikz "edMat" "center" zStr
354            "KITTEN" "KNITTING")

356 edTblTrToTikzSE
357  = putStr (edTblTrToTikz "edMat" "center" zStr
358            "SIX" "ELEVEN")
```

edDistTblKittenKnitting



edDistTblSixEleven



ToC

Commentary 4

4 Future Work, Other Examples, References

- Future work — Tutorials and TMAs
- Other examples
- References and other sources
- **Colophon**
- LaTeX with Beamer, Listings and other packages
- Index of Python code and diagrams
- PGF/TikZ for the diagrams
- External copies of the diagrams as PDF with tight bounding boxes are available

ToC

6 Future Work

- Future tutorial and TMA dates
- Sunday 26 April 2026 Tutorial Computability, Complexity Online Module Wide
- Sunday 3 May 2026 Tutorial End of Module Online
- TMA03 Thursday 28 May 2026
- **Other DP examples**
- Longest common subsequence
- Knapsack
- TSP

ToC

7 Web Sites & References

7.1 Edit Distance Sources

- Jeff Erickson Algorithms — basis of these notes
- Wikibooks Levenshtein (12 February 2016)
- Wikipedia: Levenshtein (12 February 2016)
- Linux Magazine March 2016 Issue 184 Better Finds — article on egrep which uses the Levenshtein algorithm
- Rosetta Code: Levenshtein Distance
- Peter Norvig How to Write a Spelling Corrector (9 February 2016)
- Stack Overflow: Edit Distance in Haskell (9 February 2016) — main source of Haskell code in this section

- Levenshtein Algorithm www.levenshtein.net (10 August 2023)
- Levenshtein Demo let.rug.nl/~kleiweg/lev/ (10 August 2023)
- Edit distance (Levenshtein-Distance) algorithm explanation (10 August 2023)
- Wikipedia: Damerau-Levenshtein distance (12 February 2016) allows for transposition of two adjacent characters

ToC

7.2 Code Sources

- **Haskell code**
- /Users/molyneux/MyData/Documents /OU/Courses/Computing/M269/M269Presentations /M269Prsntn2021J/M269Prsntn2021JT /M269Tutorial05Prsntn2021JGraphGreedDP /M269Tutorial05Prsntn2021JGraphGreedDP.lhs

ToC

7.3 Algorithm Texts & Web Sites

- Jeff Erickson Algorithms
- Cormen et al. (2022, 2009) Introduction to Algorithms
- Sedgewick and Wayne (2011) Algorithms see Algorithms, 4th edition
- Bird and Gibbons (2020) Algorithm Design with Haskell

7.4 Analysis of Algorithms

- Big O notation (2 April 2022)
- The Algebra of Big-O (2 April 2022)
- Python Time Complexity — also summarised in Software Summaries
- Computational Complexity of a List Comprehension (10 August 2023) — as it says: The structure of a list comprehension makes it easy to determine computational complexity.

  This is from An Introduction to Functional Programming, Lazy Evaluation, and Streams in Python (10 August 2023)
- Big-O Cheat Sheet (10 August 2023)

7.5 Graph Algorithms

- Graph Theory: Trees
- Graph Theory
- Dekai Wu Algorithms course
- http://jeffe.cs.illinois.edu/teaching/algorithms/Jeff Erickson Algorithms

ToC

7.6 Shortest Paths

- Dijkstra's Algorithm
  - Dijkstra's shortest path algorithm
- Bellman-Ford
  - Bellman–Ford Algorithm
  - Bellman Ford Algorithm (Simple Implementation)
  - Haskell Hackage Data.BellmanFord
  - Data.IGraph

    from igraph
  - igraph Reference Manual Chapter 13 Structural Properties of Graphs
  - R igraph manual pages: Shortest (directed or undirected) paths between vertices
  - IGraph/M: a Mathematica interface for igraph

ToC

7.7 Web Sites for Dynamic Programming

- Stack Exchange http://cs.stackexchange.com/questions/645/deciding-on-sub-problems-for-dynamic-programming
- Jeff Erickson Algorithms http://jeffe.cs.illinois.edu/teaching/algorithms/
- Cormen et al. (2009, page 407) has same example as Jeff Erickson
- Edit Distance HaskellWiki https://wiki.haskell.org/Edit_distance
- Edit Distance in Haskell http://stackoverflow.com/questions/5515025/edit-distance-algorithm-in-haskell-performance-tuning
- Wikibooks Algorithm implementation — Levenshtein Distance http://en.wikibooks.org/wiki/Algorithm_Implementation/Strings/Levenshtein_distance
- Wikipedia Levenshtein Distance http://en.wikipedia.org/wiki/Levenshtein_distance
- Andrew McCallum http://bioinfo.ict.ac.cn/~dbu/AlgorithmCourses/Lectures/Lec6-EditDistance.pdf — beware inconsistent (i,j)
- https://web.stanford.edu/class/cs124/lec/med.pdf
- Needleman-Wunsch algorithm http://en.wikipedia.org/wiki/Needleman\T1\textendashalgorithm
- Peter Norvig Spell Checkerhttp://norvig.com/spell-correct.html (from http://stackoverflow.com/questions/2460177/edit-distance-in-python)
- LaTeX and PGF/TikZ code for table
  - LaTeX Stack Exchange — How do I draw horizontal and vertical lines for a TikZ matrix http://tex.stackexchange.com/questions/134209/how-do-i-draw-horizontal-and-vertical-lines-for-a-tikz-matrix

- LaTeX Stack Exchange — Table-like lines in TikZ matrix http://tex.stackexchange.com/questions/204358/table-like-lines-in-tikz-matrix — but requires adjustment for non-zero column sep and row sep and for pgflinewidth

ToC

References

Bird, Richard (1998). Introduction to Functional Programming using Haskell. Prentice Hall, second edition. ISBN 0134843460.

Bird, Richard (2014). Thinking Functionally with Haskell. Cambridge University Press. ISBN 1107452643. URL https://www.cs.ox.ac.uk/publications/books/functional/.

Bird, Richard and Jeremy Gibbons (2020). Algorithm Design with Haskell. Cambridge University Press. ISBN 9781108869041. URL https://www.cs.ox.ac.uk/publications/books/adwh/. 41

Bird, Richard and Phil Wadler (1988). Introduction to Functional Programming. Prentice Hall, first edition. ISBN 0134841972.

Cormen, Thomas H.; Charles E. Leiserson; Ronald L. Rivest; and Clifford Stein (2009). Introduction to Algorithms. MIT Press, third edition. ISBN 0262533057. URL http://mitpress.mit.edu/books/introduction-algorithms. 41, 42

Cormen, Thomas H.; Charles E. Leiserson; Ronald L. Rivest; and Clifford Stein (2022). Introduction to Algorithms. MIT Press, fourth edition. ISBN 9780262046305. URL https://mitpress.mit.edu/books/introduction-algorithms-fourth-edition. 41

Dijkstra, Edsger W (1959). A note on two problems in connexion with graphs. Numerische mathematik, 1(1):269–271.

Hudak, Paul; John Hughes; Simon Peyton Jones; and Phil Wadler (2007). A History of Haskell: Being Lazy with Class. In Proceedings of the third ACM SIGPLAN conference on History of programming languages, pages 12–1–12–55. ACM New York, NY, USA.

Lee, Gias Kay (2013). Functional Programming in 5 Minutes. Web. http://gsklee.im, URL http://slid.es/gsklee/functional-programming-in-5-minutes.

Lutz, Mark (2011). Programming Python. O'Reilly, fourth edition. ISBN 0596158106. URL http://learning-python.com/books/about-pp4e.html.

Lutz, Mark (2013). Learning Python. O'Reilly, fifth edition. ISBN 1449355730. URL http://learning-python.com/books/about-lp5e.html.

Marlow, Simon and Simon Peyton Jones (2010). Haskell Language and Library Specification. Web. URL http://www.haskell.org/haskellwiki/Language_and_library_specification.

Miller, Bradley W. and David L. Ranum (2011). Problem Solving with Algorithms and Data Structures Using Python. Franklin, Beedle Associates Inc, second edition. ISBN 1590282574. URL https://runestone.academy/ns/books/published/pythonds/index.html.

O'Donnell, John; Cordelia Hall; and Rex Page (2006). Discrete Mathematics Using a Computer. Springer, second edition. ISBN 1846282411. URL http://www.dcs.gla.ac.uk/~jtod/discrete-mathematics/.

Okasaki, Chris (1998). Purely Functional Data Structures. Cambridge University Press. ISBN 0-521-63124-6.

O'Sullivan, Bryan; John Goerzen; and Donald Stewart (2008). Real World Haskell. O'Reilly, first edition. ISBN 0596514980. URL http://book.realworldhaskell.org/.

Rabhi, Fethi and Guy Lapalme (1999). Algorithms: A Functional Programming Approach. Addison-Wesley, second edition. ISBN 0201596040. URL http://www.iro.umontreal.ca/~lapalme/Algorithms-functional.html.

Sedgewick, Robert and Kevin Wayne (2011). Algorithms. Addison Wesley, fourth edition. ISBN 032157351X. URL http://algs4.cs.princeton.edu/home/. 41

Thompson, Simon (2011). Haskell the Craft of Functional Programming. Addison Wesley, third edition. ISBN 0201882957. URL http://www.haskellcraft.com/craft3e/Home.html.

van Rossum, Guido and Fred Drake (2011a). An Introduction to Python. Network Theory Limited, revised edition. ISBN 1906966133.

van Rossum, Guido and Fred Drake (2011b). The Python Language Reference Manual. Network Theory Limited, revised edition. ISBN 1906966141.

ToC

Python Code Index

Index for some (but not all) of the Python code. Note that the index commands are placed after any Python environment containing the code to be indexed.

ToC

Diagrams Index

Index for some of the PGF/TikZ diagrams. Note that the indexing commands are placed after the diagram code

ToC