Genome 540 Discussion

Conor Camplisson

February 7th, 2023

Genome Sciences
UNIVERSITY OF WASHINGTON

Outline

- Homework 4 wrap-up
- Future discussion topics
- Homework 5 questions

Outline

- Homework 4 wrap-up
- Future discussion topics
- Homework 5 questions

Homework 4 Wrap-up

genome.fa

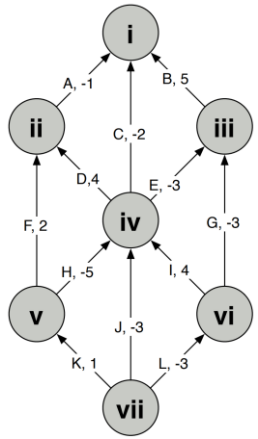
5’-ATCTG-3’

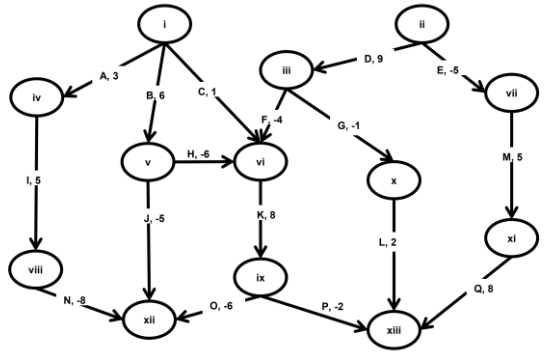
soring_scheme.txt

```
A -1.49
C 0.74
G 0.74
T -1.49
N 0
```

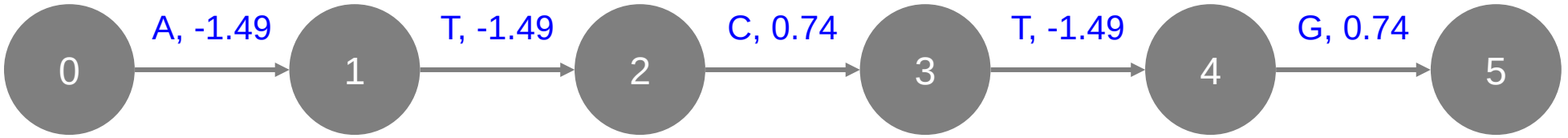
Program 2

```
V 0
V 1
V 2
V 3
V 4
V 5
E A 0 1 -1.49
E T 1 2 -1.49
E C 2 3 0.74
E T 3 4 -1.49
E G 4 5 0.74
```





Program 1

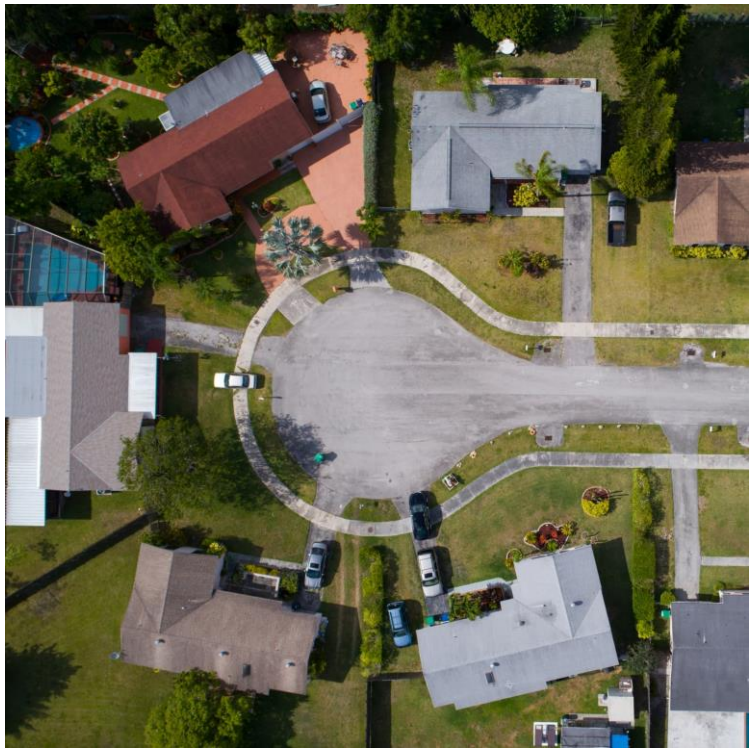


Outline

- Homework 4 wrap-up
- Future discussion topics
- Homework 5 questions

Pointers: conceptual introduction

5 houses



5 pointers to houses

```
1   777 Brockton Avenue, Abington MA 2351
2   30 Memorial Drive, Avon MA 2322
3   250 Hartford Avenue, Bellingham MA 2019
4   700 Oak Street, Brockton MA 2301
5   66-4 Parkhurst Rd, Chelmsford MA 1824
```

Information Content

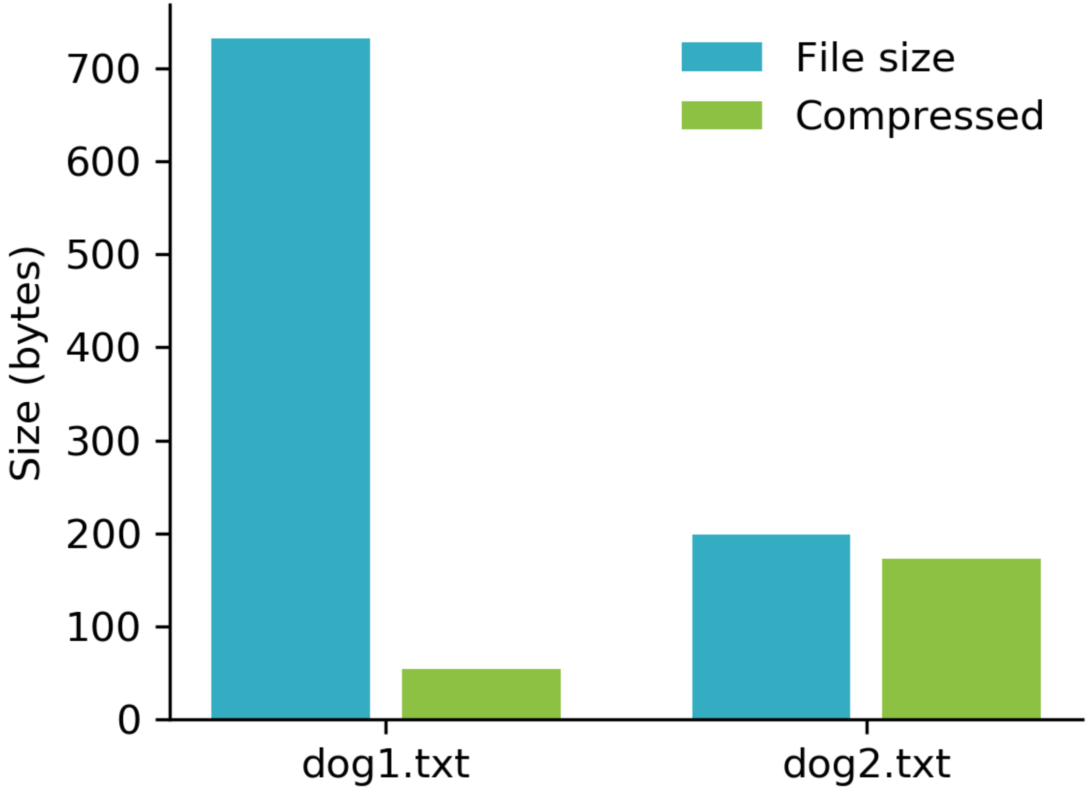
file	size	compressed
dog1.txt	732	54
dog2.txt	199	173

In [61]:
```
print_contents(file_1)
```

```
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG DOG
```

In [62]:
```
print_contents(file_2)
```

```
Lassie is a fictional character created by Eric Knight.

She is a female Rough Collie dog, and is featured in a short story
that was later expanded to a full-length novel called Lassie Come-Home.
```

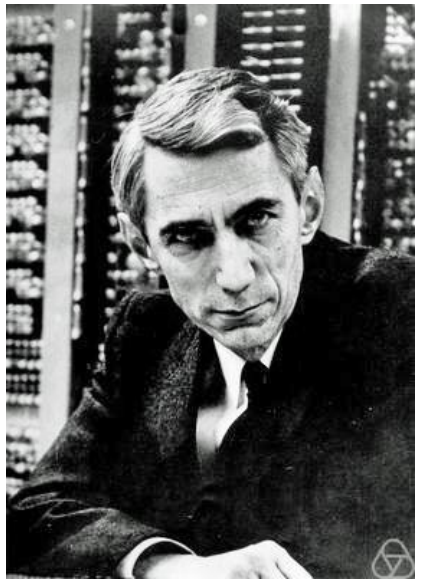


Information Theory

Quantifying information

Heads ✗

Tails ✓



Claude Shannon

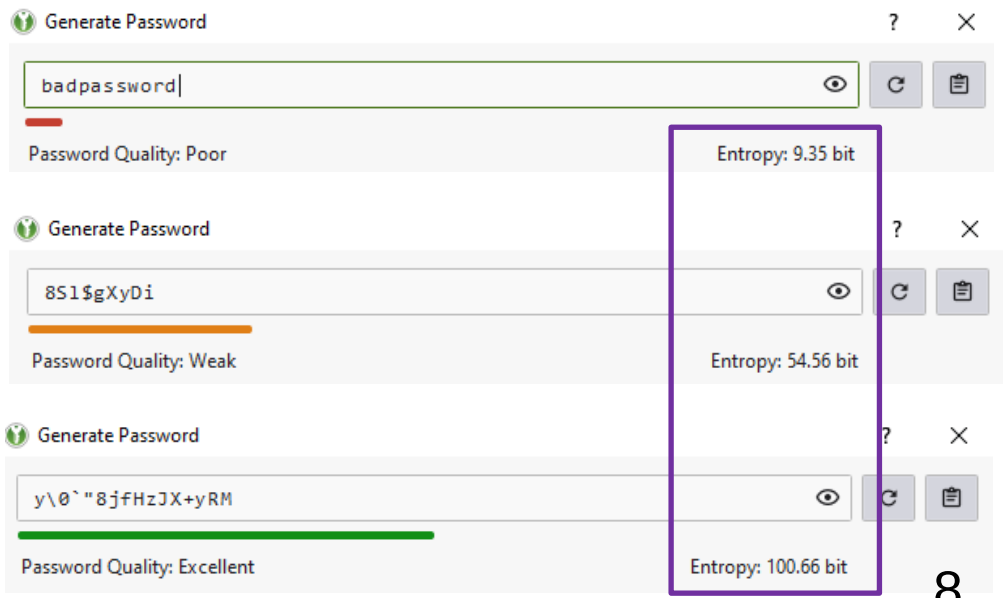
Information entropy

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \mathrm{P}(x_i) \log \mathrm{P}(x_i)$$

H(toss) = -(p(heads) * log2 (p(heads) + p(tails) * log2(p(tails))

H(toss) = -(0.5*log2(0.5) + 0.5*log2(0.5)) = **1.0 bit**

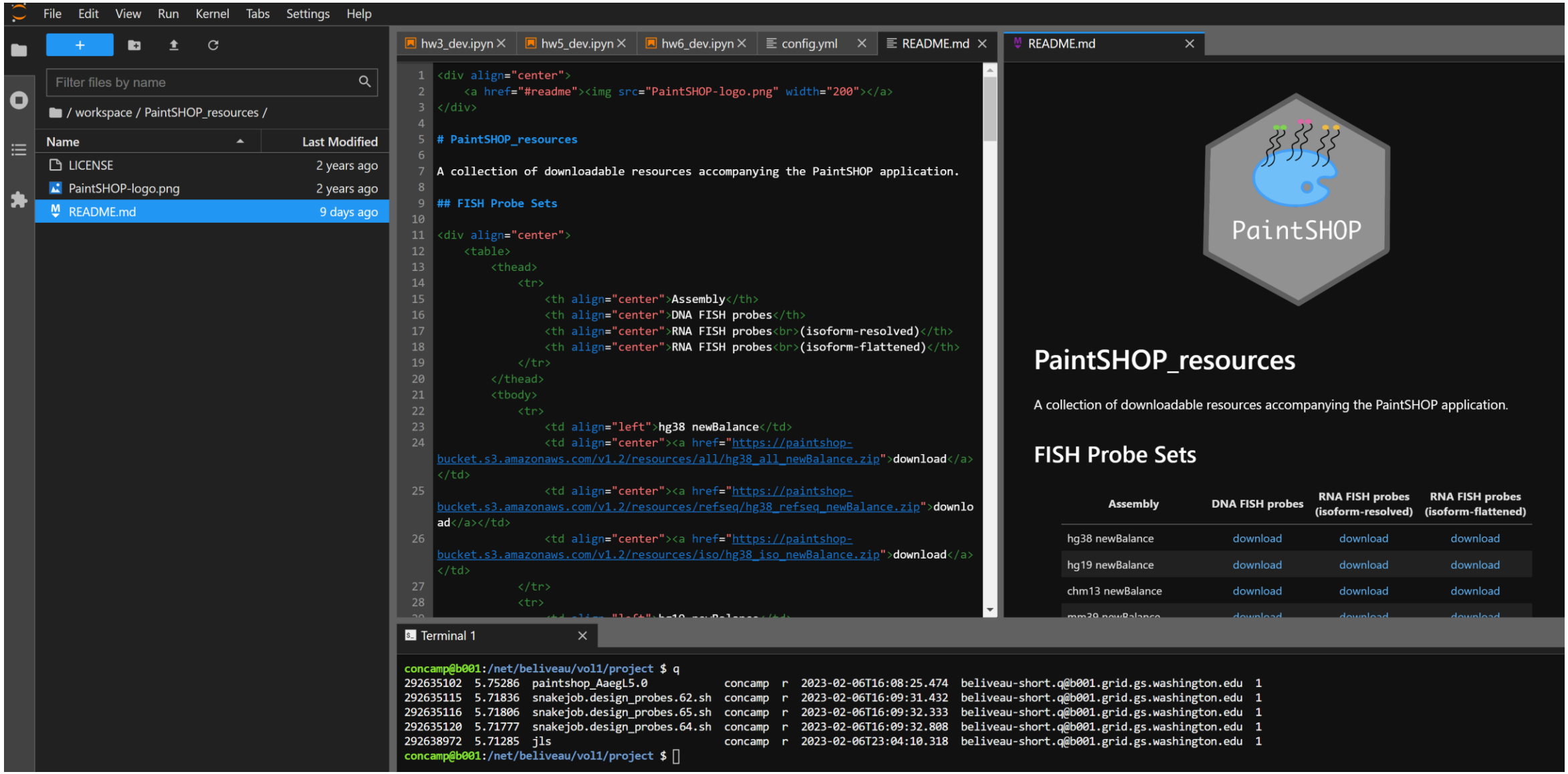
"binary digit" → "bit"



Topics for future discussion sections?

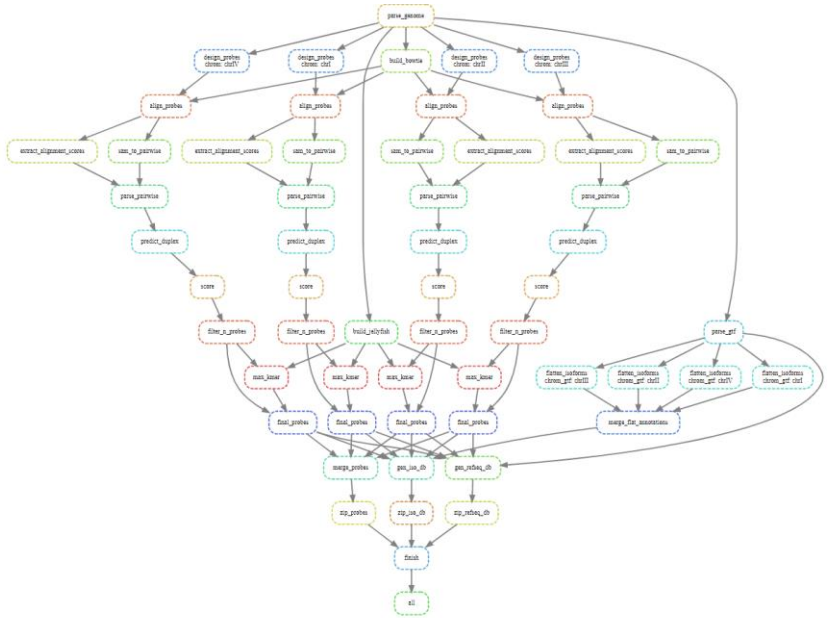
- Scalable and reproducible bioinformatics pipelines (Snakemake)
- Parallel computing: threading, multiprocessing, cluster computing
- General programming tips, languages (Python, C++, Unix tools)
- Version Control / Github
- Jupyter Notebooks / JupyterLab
- Recent DNA sequence algorithms from literature
- Virtual environments: how and why
- Fast array computing in python
- Relational database design

Jupyterlab IDE on GS Cluster



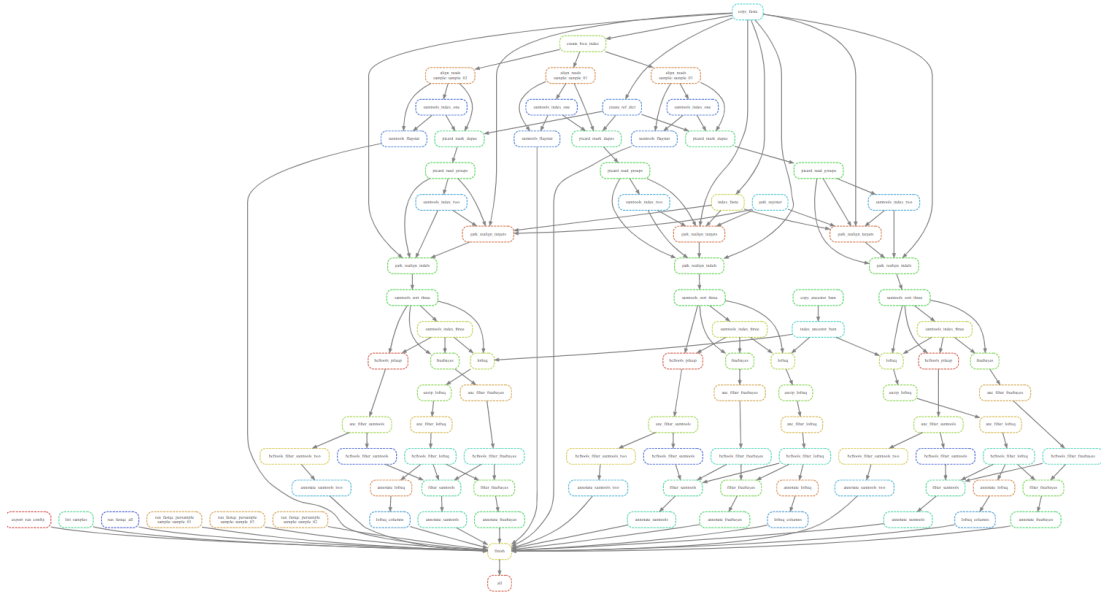
Web browser interface to GS cluster

Simple pipeline in Snakemake



PaintSHOP Pipeline

Snakemake pipeline for genome-scale mining of optimal homology sequences for PaintSHOP



yEvo Pipeline

Variant calling Snakemake pipeline for yEvo sequencing data

GitHub

CONDA

snakemake

Topics for future discussion sections?

- Scalable and reproducible bioinformatics pipelines (Snakemake)
- Parallel computing: threading, multiprocessing, cluster computing
- General programming tips, languages (Python, C++, Unix tools)
- Version Control / Github
- Jupyter Notebooks / JupyterLab
- Recent DNA sequence algorithms from literature
- Virtual environments: how and why
- Fast array computing in python
- Relational database design

Vectorized array operations in python with numpy

```
import random

x = [random.randint(0, 10) for i in range(5)]
x
```

```
[2, 4, 4, 10, 4]
```

```
seq = [random.choice('ACGT') for i in range(10)]
seq
```

```
['A', 'A', 'A', 'T', 'T', 'G', 'A', 'C', 'T', 'C']
```

```
import numpy as np

x = np.random.randint(0, 10, size=5)
x
```

```
array([8, 8, 1, 0, 3])
```

```
seq = np.random.choice(['A', 'C', 'G', 'T'], size=10)
seq
```

```
array(['T', 'G', 'G', 'A', 'T', 'A', 'A', 'C', 'A', 'G'], dtype='<U1')
```

```
seqs = np.random.choice(['A', 'C', 'G', 'T'], size=(5, 10))
seqs
```

```
array([['T', 'C', 'A', 'C', 'G', 'T', 'C', 'A', 'T', 'C'],
       ['C', 'A', 'A', 'T', 'A', 'T', 'T', 'A', 'G', 'C'],
       ['T', 'T', 'T', 'A', 'A', 'A', 'A', 'T', 'T', 'G'],
       ['T', 'C', 'G', 'C', 'T', 'G', 'C', 'T', 'A', 'C'],
       ['T', 'A', 'A', 'T', 'A', 'T', 'T', 'T', 'C', 'G']], dtype='<U1')
```

https://numpy.org/install/

Vectorized array operations in python with numpy

```python
import numpy as np

# generate random data
x = np.random.randint(0, 10, size=(10, 3))
print(x)
```

```
[[9 1 8]
 [0 3 7]
 [4 7 4]
 [1 8 4]
 [2 7 2]
 [3 0 1]
 [7 5 7]
 [0 5 3]
 [0 8 1]
 [5 6 5]]
```

```python
# sum down columns
np.sum(x, axis=0)
```

```
array([31, 50, 42])
```

```python
# sum across rows
np.sum(x, axis=1)
```

```
array([18, 10, 15, 13, 11,  4, 19,  8,  9, 16])
```

```python
# compute x^2
x ** 2
```

```
array([[81,  1, 64],
       [ 0,  9, 49],
       [16, 49, 16],
       [ 1, 64, 16],
       [ 4, 49,  4],
       [ 9,  0,  1],
       [49, 25, 49],
       [ 0, 25,  9],
       [ 0, 64,  1],
       [25, 36, 25]])
```

```python
# make a binary mask for x == 7
(x == 7).astype(int)
```

```
array([[0, 0, 0],
       [0, 0, 1],
       [0, 1, 0],
       [0, 0, 0],
       [0, 1, 0],
       [0, 0, 0],
       [1, 0, 1],
       [0, 0, 0],
       [0, 0, 0],
       [0, 0, 0]])
```

```python
# transpose x
x.T
```

```
array([[9, 0, 4, 1, 2, 3, 7, 0, 0, 5],
       [1, 3, 7, 8, 7, 0, 5, 5, 8, 6],
       [8, 7, 4, 4, 2, 1, 7, 3, 1, 5]])
```

https://numpy.org/install/

Vectorized array operations in python with numpy

```
import numpy as np

def vector_mult(x, c):
    '''Multiply all values in the input vector x by constant c.'''
    for i in range(len(x)):
        x[i] *= c
    return x

def vector_mult_np(x, c):
    '''Vectorized version of vector_mult using numpy.'''
    return x * c


# create a vector with some data
n = 10000 # length of vector to be multiplied
c = 25    # constant used in multiplication
x = list(range(n))
x_np = np.array(x)

# benchmark python list with iteration
%timeit vector_mult(list(x), 10)

# benchmark numpy vectorized calc
%timeit vector_mult_np(x_np.copy(), 10)
```

```
947 µs ± 272 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
10.4 µs ± 175 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)
```

```
# example data
x = [0, 2, 4, 6, 8]
x_np = np.array([0, 2, 4, 6, 8])

vector_mult(list(x), 10)
```

```
[0, 20, 40, 60, 80]
```

```
vector_mult(list(x), 100)
```

```
[0, 200, 400, 600, 800]
```

```
vector_mult_np(x_np.copy(), 10)
```

```
array([ 0, 20, 40, 60, 80])
```

```
vector_mult_np(x_np.copy(), 100)
```

```
array([  0, 200, 400, 600, 800])
```

https://numpy.org/install/

Vectorized array operations in python with numpy

```python
import numpy as np

# generate 8 random 12-mers
seqs = np.random.choice(('A', 'C', 'G', 'T'), size=(8, 10))

print(seqs)
```

```
[['A' 'A' 'T' 'T' 'T' 'T' 'T' 'A' 'G' 'T']
 ['T' 'A' 'C' 'T' 'T' 'T' 'C' 'C' 'C' 'C']
 ['A' 'T' 'G' 'T' 'G' 'G' 'G' 'T' 'G' 'G']
 ['A' 'G' 'C' 'A' 'G' 'T' 'C' 'A' 'G' 'T']
 ['G' 'T' 'G' 'G' 'A' 'T' 'C' 'C' 'T' 'T']
 ['T' 'T' 'A' 'T' 'T' 'A' 'A' 'T' 'T' 'A']
 ['A' 'C' 'T' 'G' 'C' 'C' 'C' 'A' 'G' 'C']
 ['A' 'T' 'T' 'T' 'T' 'G' 'T' 'A' 'T' 'A']]
```

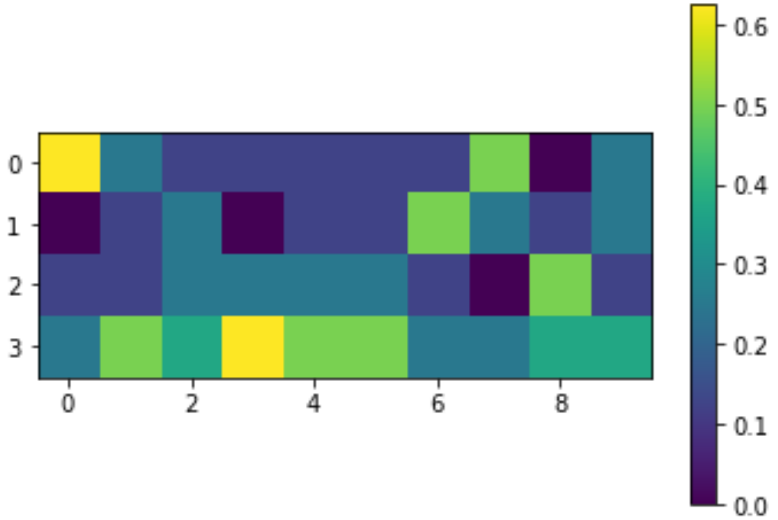
```python
# store nucleotides
NUCS = np.array(['A', 'C', 'G', 'T'])

# compute counts matrix
counts = np.sum(seqs[:,:,np.newaxis] == NUCS, axis=0).T

print(counts)
```

```
[[5 2 1 1 1 1 1 4 0 2]
 [0 1 2 0 1 1 4 2 1 2]
 [1 1 2 2 2 2 1 0 4 1]
 [2 4 3 5 4 4 2 2 3 3]]
```

```python
# compute frequency matrix
freqs = counts / np.sum(counts, axis=0)

print(freqs.round(3))
```

```
[[0.625 0.25  0.125 0.125 0.125 0.125 0.125 0.5   0.    0.25 ]
 [0.    0.125 0.25  0.    0.125 0.125 0.5   0.25  0.125 0.25 ]
 [0.125 0.125 0.25  0.25  0.25  0.25  0.125 0.    0.5   0.125]
 [0.25  0.5   0.375 0.625 0.5   0.5   0.25  0.25  0.375 0.375]]
```

```python
import matplotlib.pyplot as plt

plt.imshow(freqs)
plt.colorbar()
plt.show()
```



https://numpy.org/install/

Vectorized array operations in python with numpy

```python
# store nucleotides
NUCS = np.array(['A', 'C', 'G', 'T'])

# generate two sets of 10 x 10-mers
n = 10
seqs_a = np.random.choice(NUCS, size=(n, 10), p=[0.19, 0.31, 0.31, 0.19]) # GC-rich
seqs_b = np.random.choice(NUCS, size=(n, 10), p=[0.25, 0.25, 0.25, 0.25]) # Equal freq

# compute counts matrices
counts_a = np.sum(seqs_a[:,:,np.newaxis] == NUCS, axis=0).T
counts_b = np.sum(seqs_b[:,:,np.newaxis] == NUCS, axis=0).T

# compute frequency matrices
freqs_a = counts_a / np.sum(counts_a, axis=0)
freqs_b = counts_b / np.sum(counts_b, axis=0)

# compute weight matrix using LLR
weights = np.log2(freqs_a / freqs_b)

print(weights.round(3))
```

```
[[ 1.    -1.       inf  2.    -2.807   -inf  2.807  0.    -1.     0.585]
 [-1.     1.    -1.    -1.     2.     0.415 -2.     1.585  1.    -0.415]
 [ 2.       inf  1.585  0.     0.     0.585 -0.585  1.585  2.     0.585]
 [-1.    -2.    -0.322 -1.     2.      -inf   -inf -1.585 -2.322 -1.   ]]
```

```
/net/gs/vol1/home/concamp/miniconda3/envs/genome_sciences_env/lib/python3.7/site-packages/ipykernel_launcher.py:21:
RuntimeWarning: divide by zero encountered in true_divide
/net/gs/vol1/home/concamp/miniconda3/envs/genome_sciences_env/lib/python3.7/site-packages/ipykernel_launcher.py:21:
RuntimeWarning: divide by zero encountered in log2
```

https://numpy.org/install/

Vectorized array operations in python with numpy

Zero division error:

+1 to bg counts

Log(zero) error:

- Mask with (e.g.) 1.0
- Set to -99.0 after log

```python
# store nucleotides
NUCS = np.array(['A', 'C', 'G', 'T'])

# generate two sets of 1 million random 10-mers
n = int(1e6)
seqs_a = np.random.choice(NUCS, size=(n, 10), p=[0.19, 0.31, 0.31, 0.19]) # GC-rich
seqs_b = np.random.choice(NUCS, size=(n, 10), p=[0.25, 0.25, 0.25, 0.25]) # Equal freq

# compute counts matrices
counts_a = np.sum(seqs_a[:,:,np.newaxis] == NUCS, axis=0).T
counts_b = np.sum(seqs_b[:,:,np.newaxis] == NUCS, axis=0).T

# add pseudocounts to denominator
counts_b[counts_b == 0] = 1

# compute frequency matrices
freqs_a = counts_a / np.sum(counts_a, axis=0)
freqs_b = counts_b / np.sum(counts_b, axis=0)

# prevent log zero errors by masking with 1
zero_idx = np.where(counts_a == 0)
freqs_a[zero_idx] = 1.0

# compute weight matrix using LLR
weights = np.log2(freqs_a / freqs_b)

# assign -99.0 to zero positions
weights[zero_idx] = -99.0
```

https://numpy.org/install/

Vectorized array operations in python with numpy

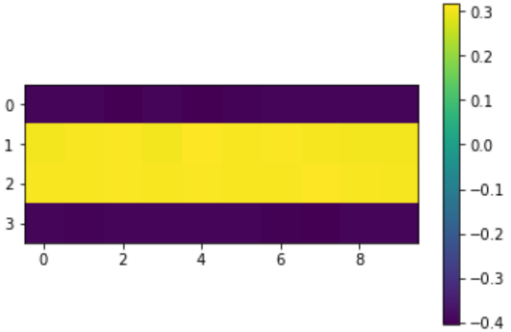
```
import time
start = time.time()

# store nucleotides
NUCS = np.array(['A', 'C', 'G', 'T'])

# generate two sets of 1 million random 10-mers
n = int(1e6)
seqs_a = np.random.choice(NUCS, size=(n, 10), p=[0.19, 0.31, 0.31, 0.19]) # GC-rich
seqs_b = np.random.choice(NUCS, size=(n, 10), p=[0.25, 0.25, 0.25, 0.25]) # Equal freq
```

```
print(weights.round(3))
print(f'Elapsed: {time.time() - start:.3f} sec')
```

```
[[-0.395 -0.391 -0.405 -0.391 -0.403 -0.397 -0.393 -0.393 -0.394 -0.393]
 [ 0.306  0.309  0.313  0.304  0.315  0.311  0.312  0.307  0.305  0.305]
 [ 0.311  0.309  0.313  0.311  0.313  0.311  0.31   0.318  0.31   0.309]
 [-0.392 -0.399 -0.394 -0.395 -0.396 -0.394 -0.401 -0.404 -0.392 -0.392]]
Elapsed: 2.047 sec
```

```
print(f'AT:\tcomputed = {np.log2(0.19 / 0.25):.3f}\t\tsimulated = {np.mean(weights[[0, 3],:]):.3f}')
print(f'GC:\tcomputed = {np.log2(0.31 / 0.25):.3f}\t\tsimulated = {np.mean(weights[1:2,:]):.3f}')
```

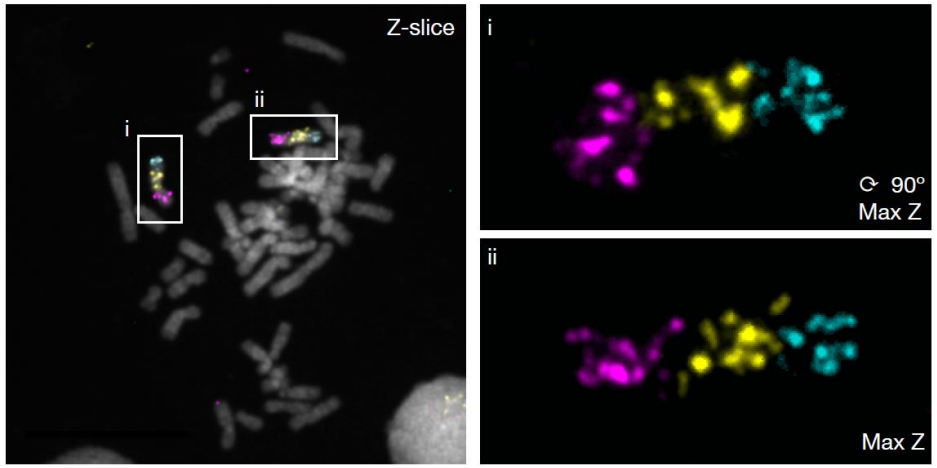
```
AT:     computed = -0.396               simulated = -0.396
GC:     computed = 0.310                simulated = 0.309
```

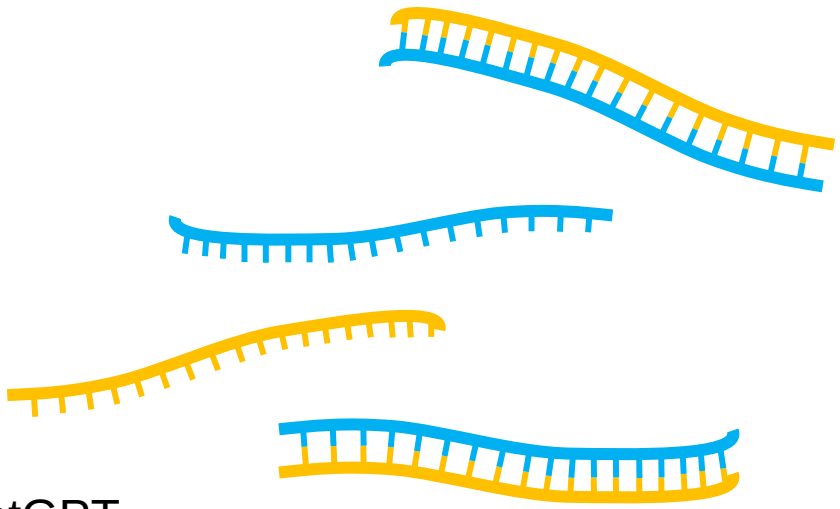


https://numpy.org/install/

Preview: 8bit encoding DNA sequences

Pattern 1: 3-color side-by-side





write python code to efficiently load a single DNA sequence contained in a fasta file and encode it as an 8-bit integer array using numpy

```python
import numpy as np

def load_dna_sequence(filename):
    """
    Loads a single DNA sequence from a fasta file and encodes it as an 8-bit integer array using numpy
    :param filename: the name of the fasta file
    :return: an 8-bit integer numpy array representing the DNA sequence
    """
    # Create a dictionary mapping each nucleotide to its encoding
    encoding = {'A': 0, 'C': 1, 'G': 2, 'T': 3}

    # Initialize an empty list to store the encoded sequence
    sequence = []

    # Read the fasta file line by line
    with open(filename, 'r') as file:
        for line in file:
            # Skip the header line starting with '>'
            if line.startswith('>'):
                continue

            # Iterate over each nucleotide in the line and encode it
            for nucleotide in line.strip():
                sequence.append(encoding[nucleotide])

    # Convert the list to a numpy array and return it
    return np.array(sequence, np.uint8)
```

chatGPT

Outline

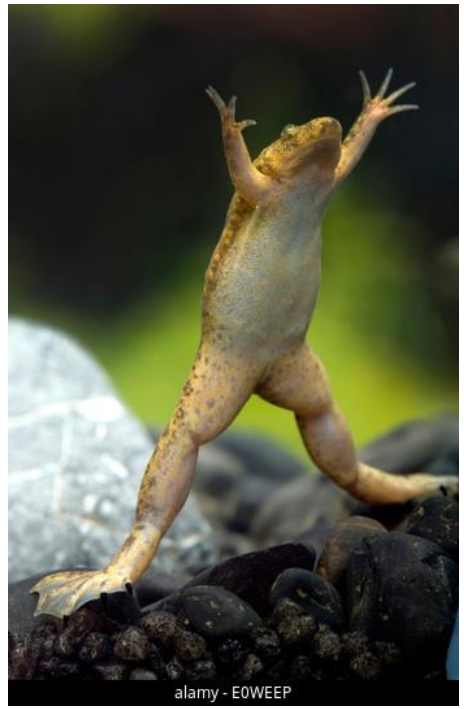
- Homework 4 wrap-up
- Future discussion topics
- Homework 5 questions

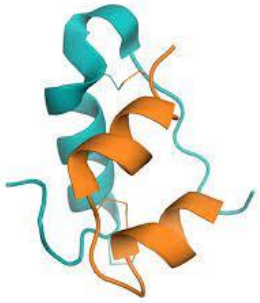
Homework 5 Overview

Homo sapiens

Bubalus bubalis

Xenopus tropcalis





Insulin

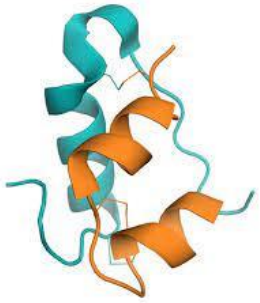
```
>sp|P01308|INS_HUMAN Insulin OS=Homo sapiens GN=INS PE=1 SV=1
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN

>tr|Q25C78|Q25C78_BUBBU Insulin (Fragment) OS=Bubalus bubalis OX=89462 GN=bpi PE=2 SV=1
FVNQHLCGSHLVEALYLVCGERGFFYTPKARREVEGPQVGALELAGGPGAGGLEGPPQKRGIVEQCCASVCSLYQLENYCN

>tr|F6QRS1|F6QRS1_XENTR Insulin OS=Xenopus tropicalis OX=8364 GN=fbxw8 PE=3 SV=3
MLPLSSIMALWMQCLPLVLVLLFSTPNTEALANQHLCGSHLVEALYLVCGDRGFFYYPKIKRDIEQAMVNGPQDNELDGMQLQPQEYQKMKRGIVEQCCHSTCSLFQLESYCN
```

Homework 5 Overview

Goal: find the best (local) multiple sequence alignment of three insulin proteins using the BLOSUM62 score matrix

Approach:

- Write a program that builds a weighted, 3D edit graph for the three sequences and exports it to .txt representation
- Determine the max weight path using your HW4 program



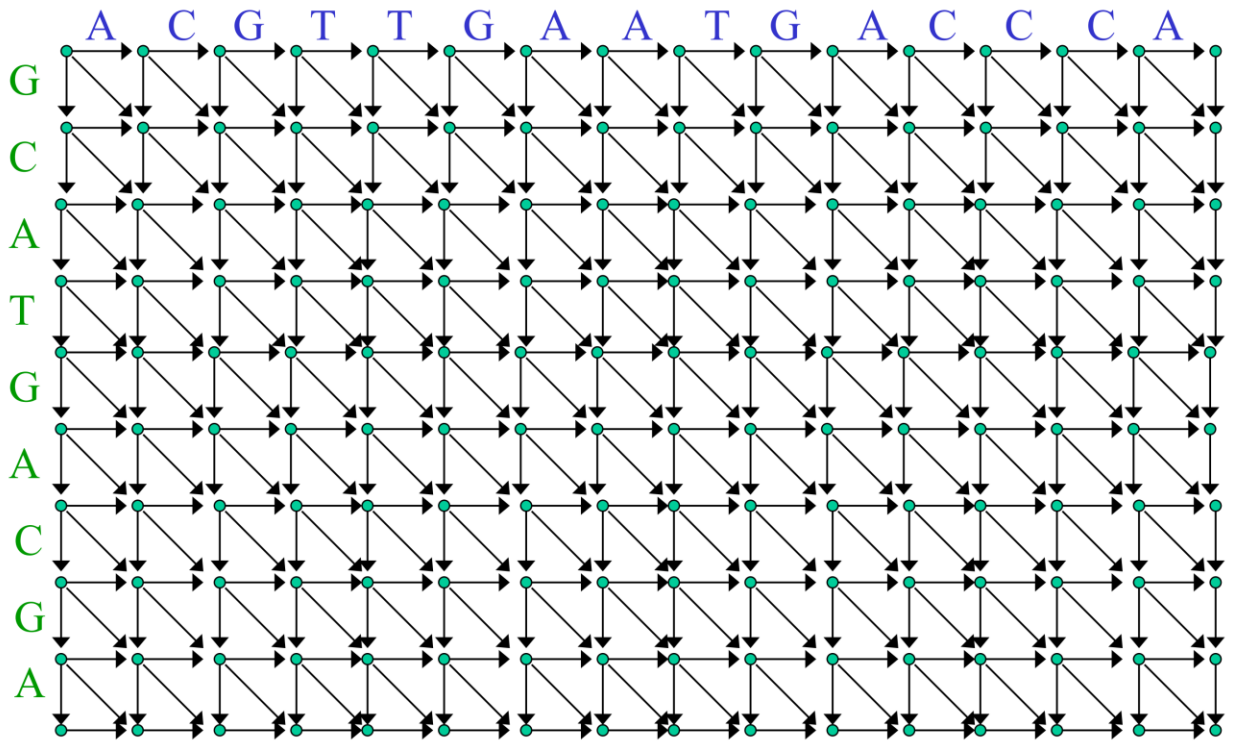
Insulin

```
>sp|P01308|INS_HUMAN Insulin OS=Homo sapiens GN=INS PE=1 SV=1
MALWMRLLPLLALLALWGPDPAAAFVNQHLCGSHLVEALYLVCGERGFFYTPKTRREAEDLQVGQVELGGGPGAGSLQPLALEGSLQKRGIVEQCCTSICSLYQLENYCN

>tr|Q25C78|Q25C78_BUBBU Insulin (Fragment) OS=Bubalus bubalis OX=89462 GN=bpi PE=2 SV=1
FVNQHLCGSHLVEALYLVCGERGFFYTPKARREVEGPQVGALELAGGPGAGGLEGPPQKRGIVEQCCASVCSLYQLENYCN

>tr|F6QRS1|F6QRS1_XENTR Insulin OS=Xenopus tropicalis OX=8364 GN=fbxw8 PE=3 SV=3
MLPLSSIMALWMQCLPLVLVLLFSTPNTEALANQHLCGSHLVEALYLVCGDRGFFYYPKIKRDIEQAMVNGPQDNELDGMQLQPQEYQKMKRGIVEQCCHSTCSLFQLESYCN
```

Homework 5 Overview

1 seq → 1-D sequence graph

2 seqs → 2-D Edit graph(pairwise alignment)

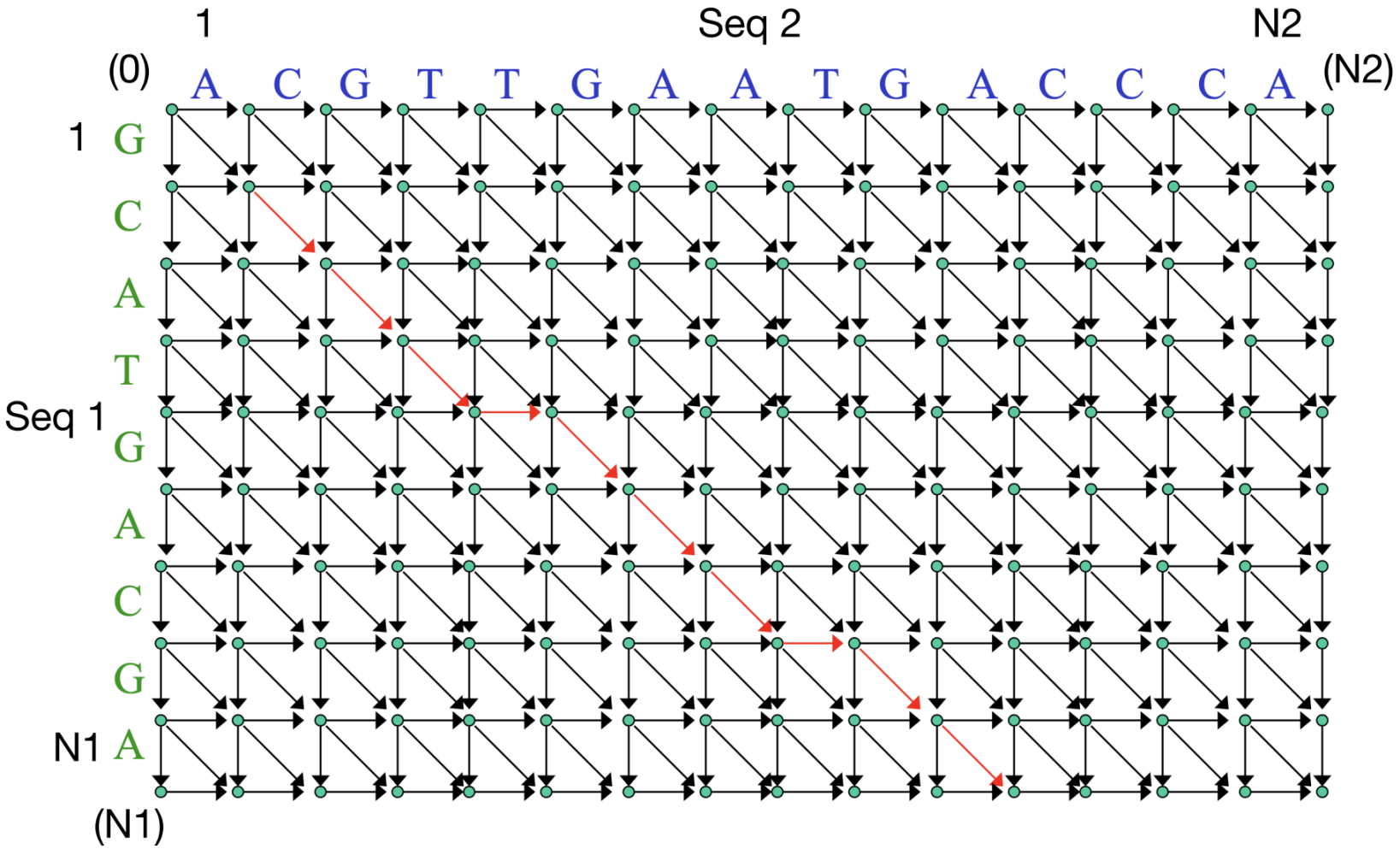


http://bozeman.mbt.washington.edu/compbio/mbt599_2022/Lec8.pdf

Homework 5 Overview

Sequence 1: from 1 to N1
Sequence 2: from 1 to N2

Pairwise alignment



Vertice: (0,0) (0,1) (0,2) ⋯ (0,N2)
(1,0) (1,1)
(2,0) ⋱
⋯
(N1,0) (N1,N2)

Vertice: two *for* loops

Edges: (0,0) (0,1) weight (_A)

(0,0) (1,0) weight (G_)

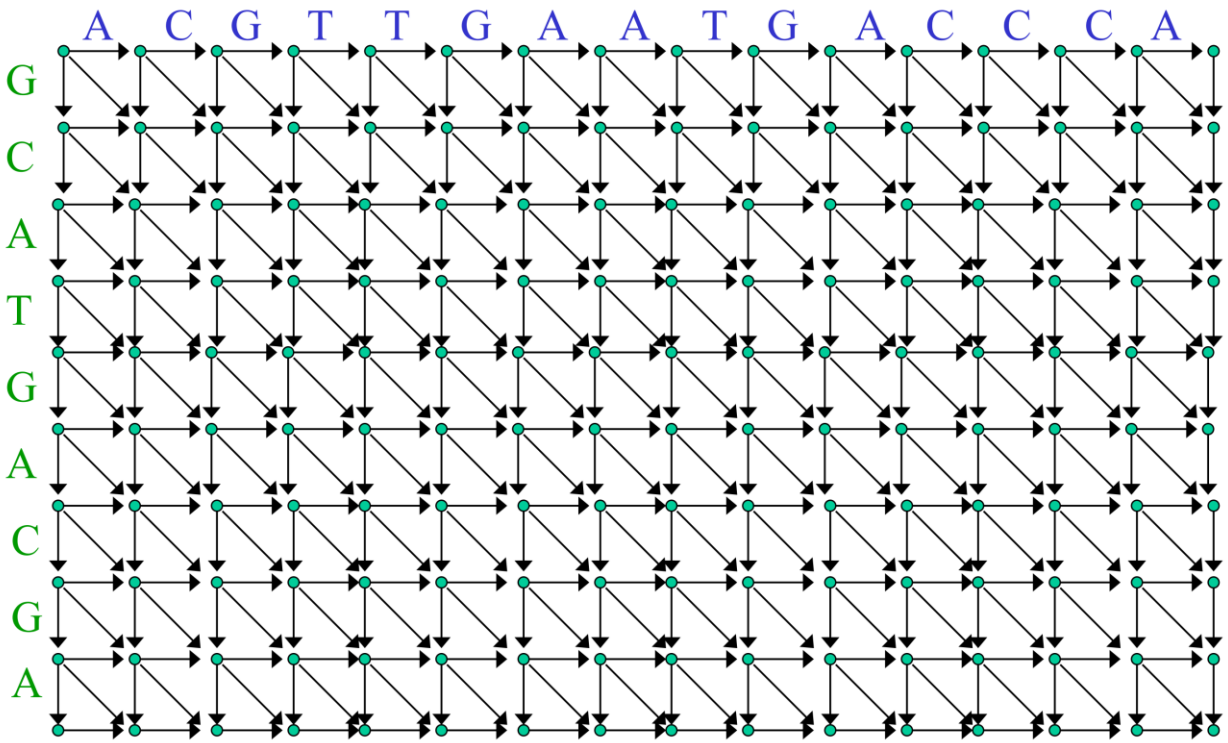
(0,0) (1,1) weight (GA)

…

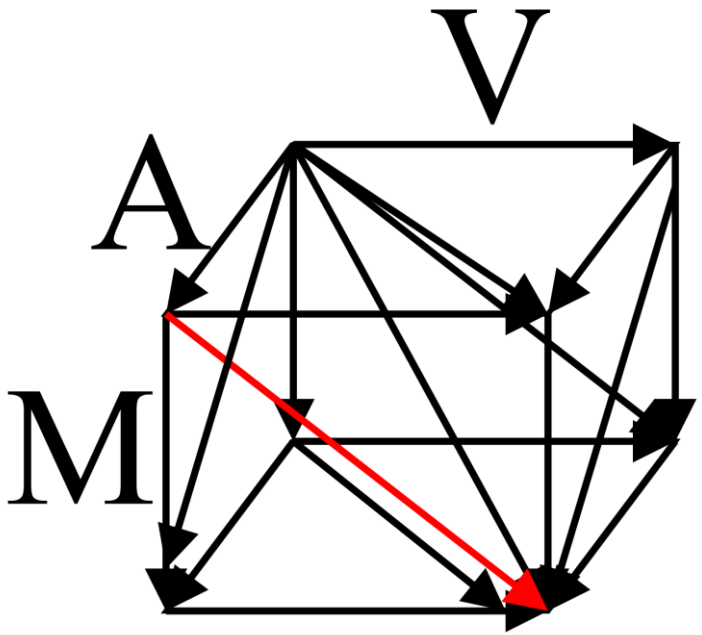
Edges: for any node (i, j)
(i, j) -> (i+1, j)
(i, j) -> (i, j+1)
(i, j) -> (i+1, j+1)

Homework 5 Overview

2 sequences → 2-D (pairwise alignment)



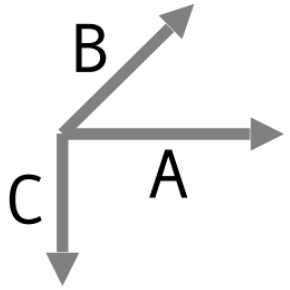
3 sequences → 3-D (multiple alignment)

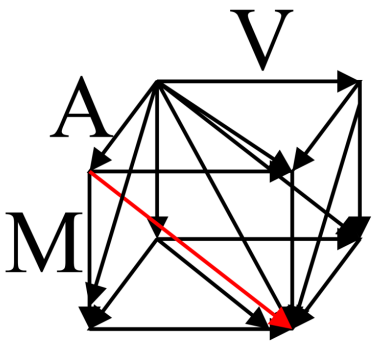


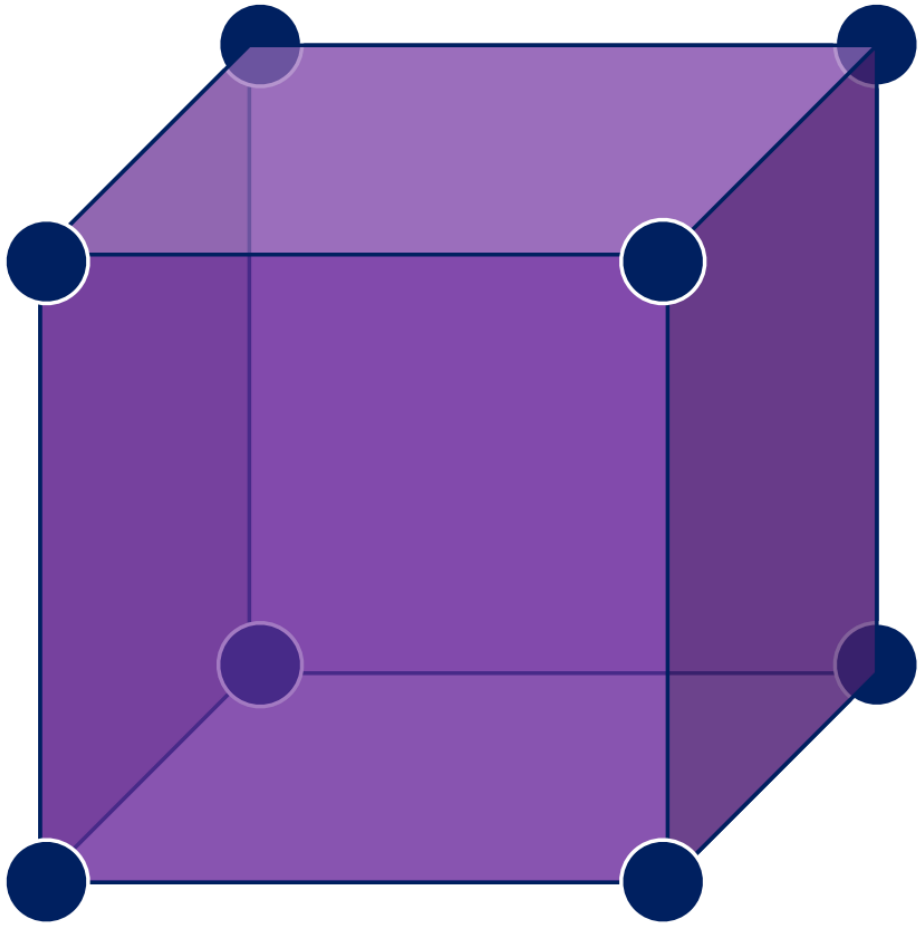
(each cell ^)

http://bozeman.mbt.washington.edu/compbio/mbt599_2022/Lec9.pdf

Homework 5 Overview

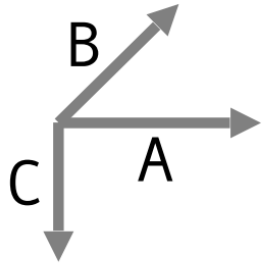


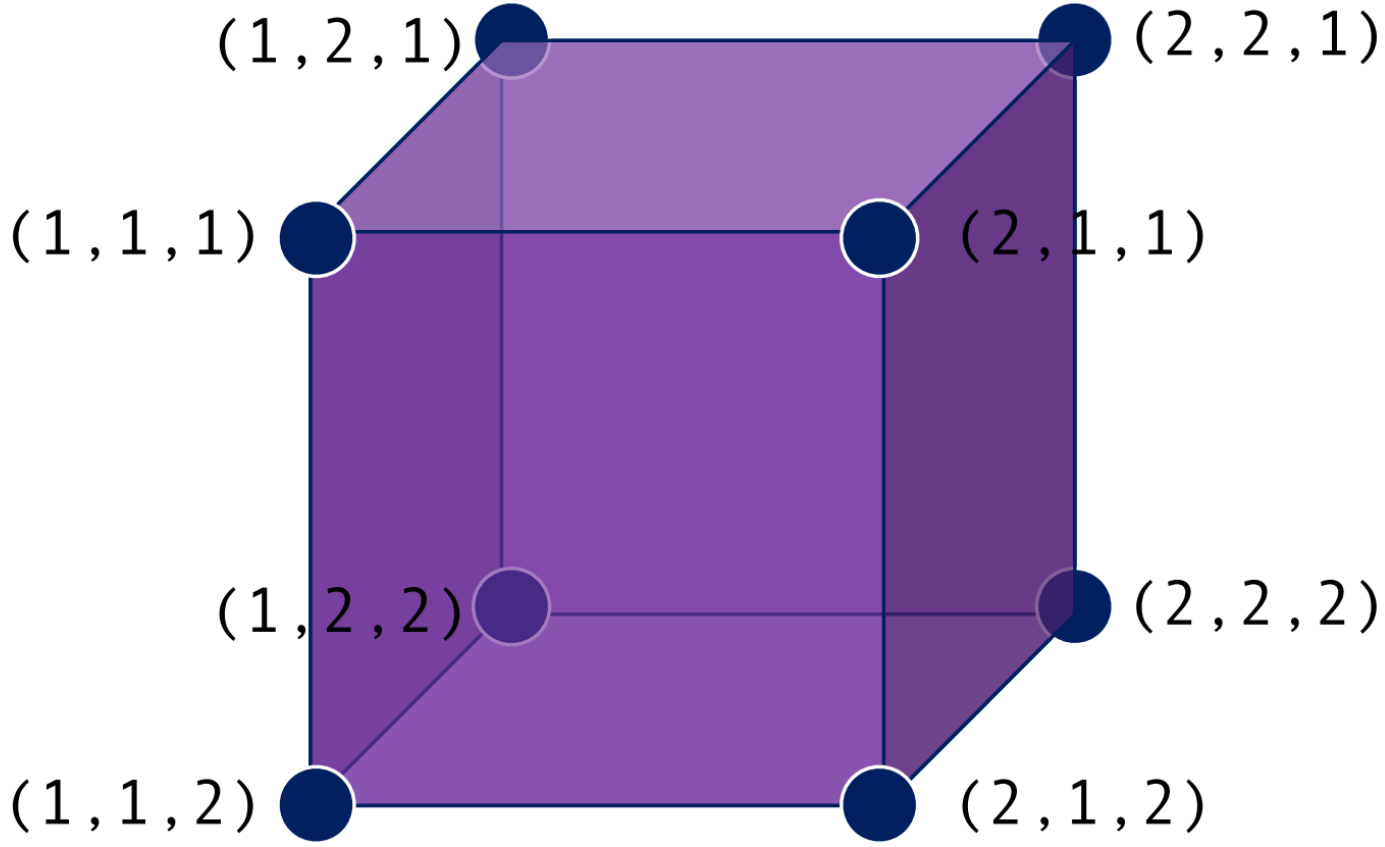




```
A = •M•C•D•R•...
B = •M•S•D•E•...
C = •M•V•D•R•...
```

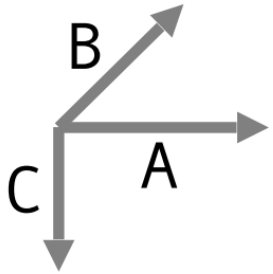
Homework 5 Overview

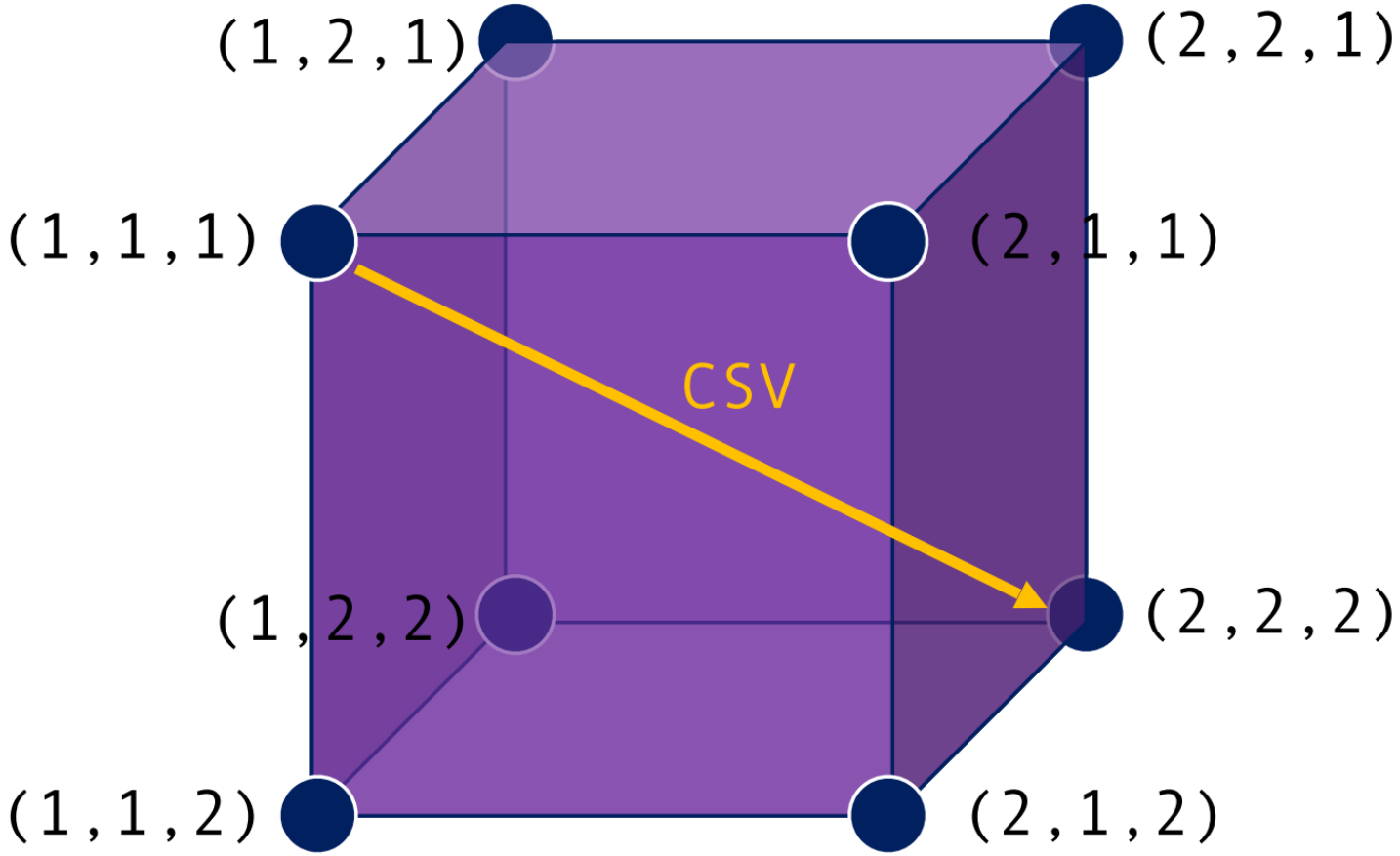





```
A = •M•C•D•R•...
B = •M•S•D•E•...
C = •M•V•D•R•...
```

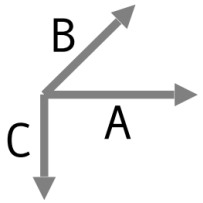
Homework 5 Overview

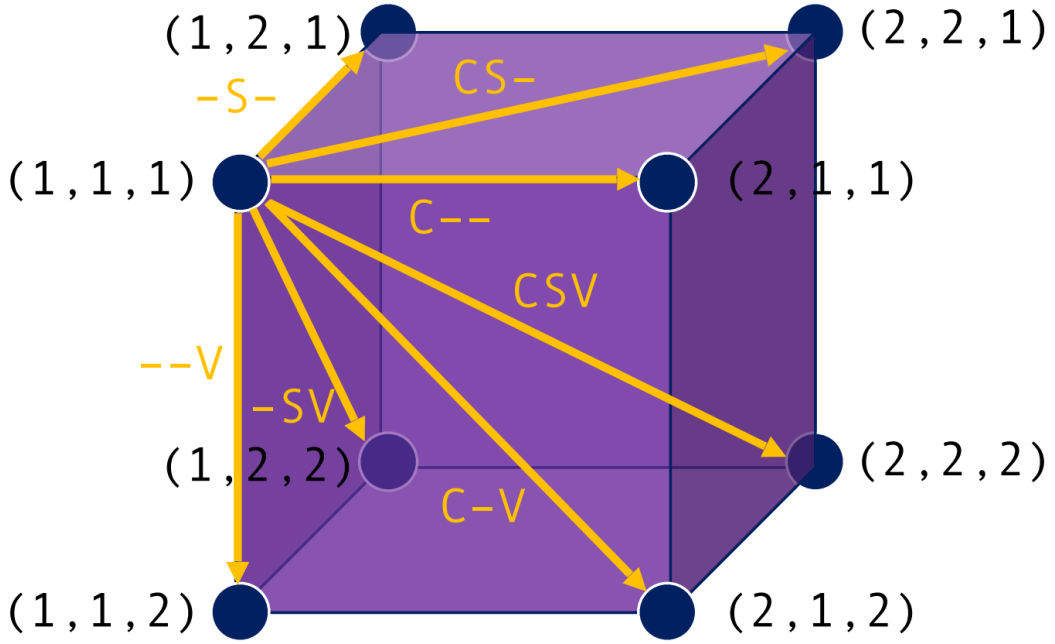





```
.
A = •M•C•D•R•...
B = •M•S•D•E•...
C = •M•V•D•R•...
```

Homework 5 Overview






```
.
A = •M•C•D•R•...
B = •M•S•D•E•...
C = •M•V•D•R•...
```

```
weight(CSV) = score(CS) + score(CV) + score(SV)

weight(CS-) = score(CS) + gap_penalty + gap_penalty
```

Homework 5 Overview

the BLOSUM62 score matrix for the pairwise scores:

	A	R	N	D	C	Q	E	G	H	I	L	K	M	F	P	S	T	W	Y	V	B	Z	X	*
A	4	-1	-2	-2	0	-1	-1	0	-2	-1	-1	-1	-1	-2	-1	1	0	-3	-2	0	-2	-1	0	-4
R	-1	5	0	-2	-3	1	0	-2	0	-3	-2	2	-1	-3	-2	-1	-1	-3	-2	-3	-1	0	-1	-4
N	-2	0	6	1	-3	0	0	0	1	-3	-3	0	-2	-3	-2	1	0	-4	-2	-3	3	0	-1	-4
D	-2	-2	1	6	-3	0	2	-1	-1	-3	-4	-1	-3	-3	-1	0	-1	-4	-3	-3	4	1	-1	-4
C	0	-3	-3	-3	9	-3	-4	-3	-3	-1	-1	-3	-1	-2	-3	-1	-1	-2	-2	-1	-3	-3	-2	-4
Q	-1	1	0	0	-3	5	2	-2	0	-3	-2	1	0	-3	-1	0	-1	-2	-1	-2	0	3	-1	-4
E	-1	0	0	2	-4	2	5	-2	0	-3	-3	1	-2	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
G	0	-2	0	-1	-3	-2	-2	6	-2	-4	-4	-2	-3	-3	-2	0	-2	-2	-3	-3	-1	-2	-1	-4
H	-2	0	1	-1	-3	0	0	-2	8	-3	-3	-1	-2	-1	-2	-1	-2	-2	2	-3	0	0	-1	-4
I	-1	-3	-3	-3	-1	-3	-3	-4	-3	4	2	-3	1	0	-3	-2	-1	-3	-1	3	-3	-3	-1	-4
L	-1	-2	-3	-4	-1	-2	-3	-4	-3	2	4	-2	2	0	-3	-2	-1	-2	-1	1	-4	-3	-1	-4
K	-1	2	0	-1	-3	1	1	-2	-1	-3	-2	5	-1	-3	-1	0	-1	-3	-2	-2	0	1	-1	-4
M	-1	-1	-2	-3	-1	0	-2	-3	-2	1	2	-1	5	0	-2	-1	-1	-1	-1	1	-3	-1	-1	-4
F	-2	-3	-3	-3	-2	-3	-3	-3	-1	0	0	-3	0	6	-4	-2	-2	1	3	-1	-3	-3	-1	-4
P	-1	-2	-2	-1	-3	-1	-1	-2	-2	-3	-3	-1	-2	-4	7	-1	-1	-4	-3	-2	-2	-1	-2	-4
S	1	-1	1	0	-1	0	0	0	-1	-2	-2	0	-1	-2	-1	4	1	-3	-2	-2	0	0	0	-4
T	0	-1	0	-1	-1	-1	-1	-2	-2	-1	-1	-1	-1	-2	-1	1	5	-2	-2	0	-1	-1	0	-4
W	-3	-3	-4	-4	-2	-2	-3	-2	-2	-3	-2	-3	-1	1	-4	-3	-2	11	2	-3	-4	-3	-2	-4
Y	-2	-2	-2	-3	-2	-1	-2	-3	2	-1	-1	-2	-1	3	-3	-2	-2	2	7	-1	-3	-2	-1	-4
V	0	-3	-3	-3	-1	-2	-2	-3	-3	3	1	-2	1	-1	-2	-2	0	-3	-1	4	-3	-2	-1	-4
B	-2	-1	3	4	-3	0	1	-1	0	-3	-4	0	-3	-3	-2	0	-1	-4	-3	-3	4	1	-1	-4
Z	-1	0	0	1	-3	3	4	-2	0	-3	-3	1	-1	-3	-1	0	-1	-3	-2	-2	1	4	-1	-4
X	0	-1	-1	-1	-2	-1	-1	-1	-1	-1	-1	-1	-1	-1	-2	0	0	-2	-1	-1	-1	-1	-1	-4
*	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	-4	1

Gap penalty: -6

Sum of pairs scoring

Edge:
x1, x2, x3

Edge weight:
sum((x1, x2), (x1, x3), (x2, x3))

- the corresponding score matrix entry if x_i and x_j are both residues
- the gap penalty if one of x_i and x_j is a residue, and the other is a gap character
- 0 if both x_i and x_j are gap characters

Homework 5 Overview

Your program should output the following:

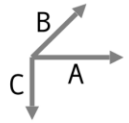
1. The maximum path score
2. A list of all edge weights (sorted alphabetically by edge name)
3. A histogram of edge counts (again, sorted alphabetically by edge name)
4. The highest-scoring alignment, formatted vertically (as described above)

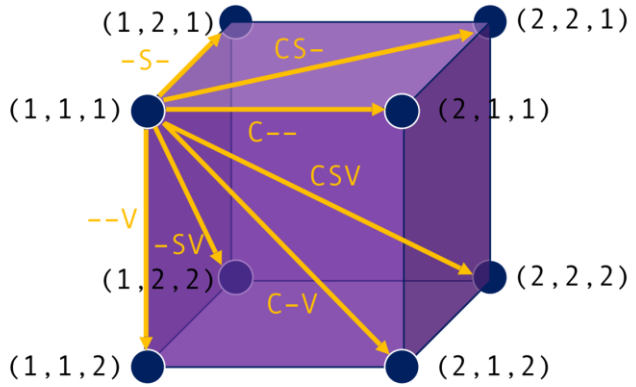
```
Assignment: GS540 HW5
Name: {YOURNAME}
Email: {YOUREMAIL}
Language: {YOURLANGUAGE}
Runtime: {YOURRUNTIME}

Score: 82.0

Edge weights:
--A = -12
--C = -12
--D = -12
--E = -12
--F = -12
.
.
.
list all edge weights in alphabetical order
(only first/last 5 shown here)
.
.
.
YYS = 3
YYT = 3
YYV = 5
YYW = 11
YYY = 21
```

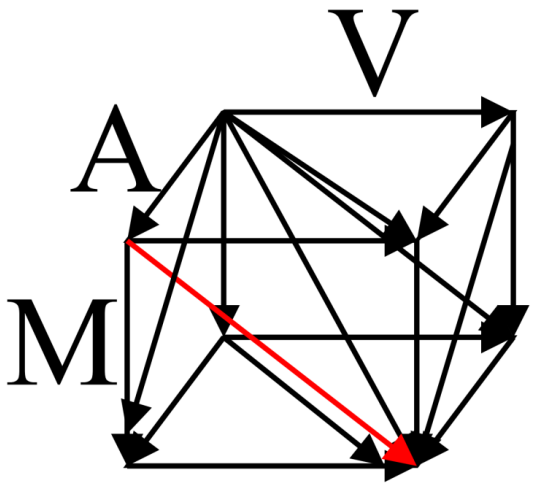
```
Edge counts:
--A = 8832
--C = 17664
--D = 52992
--E = 70656
--F = 44160
.
.
.
list all the edge counts in alphabetical order
(only first/last 5 shown here)
.
.
.
YYS = 48
YYT = 24
YYV = 72
YYW = 24
YYY = 60
```

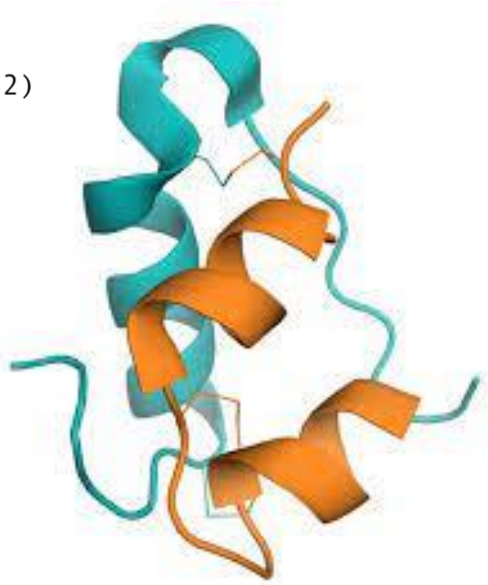
```
Local alignment:
KKK
DLK
YWY
G--
LFL
KVN
REH
IPI
QRQ
KPN
SNS
AVV
FFF
MVE
GSG
SGE
LVI
KKS
KDE
HSP
AVQ
LAI
NQL
STN
LVL
LVK
TDY
DYE
LLL
RLN
RQK
MYY
VTI
--R
--K
NPD
VVI
DED
SSS
-G-
VLV
IMI
VLL
FFF
```

Homework 5 Questions ?



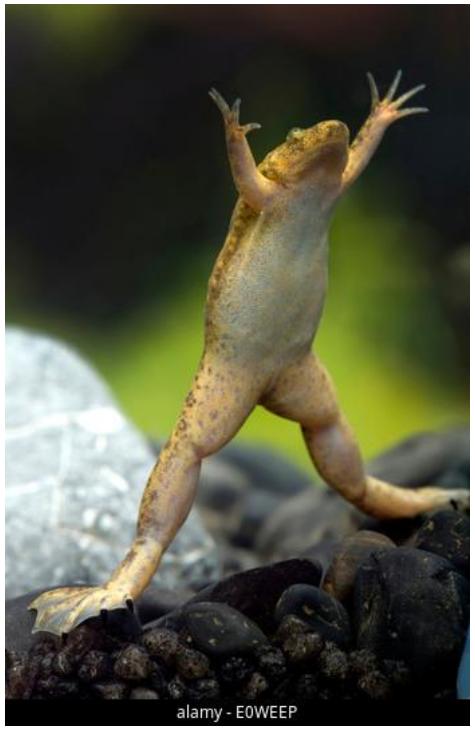


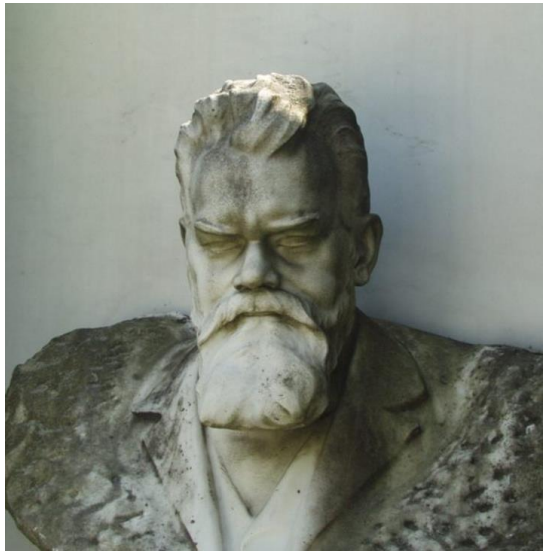




Insulin

Local alignment:
KKK
DLK
YWY
G--
LFL
KVN
REH
IPI
QRQ
KPN
SNS
AVV
FFF
MVE
GSG
SGE
LVI
KKS
KDE
HSP





. . .