

BMI/CS 576 – Day 7

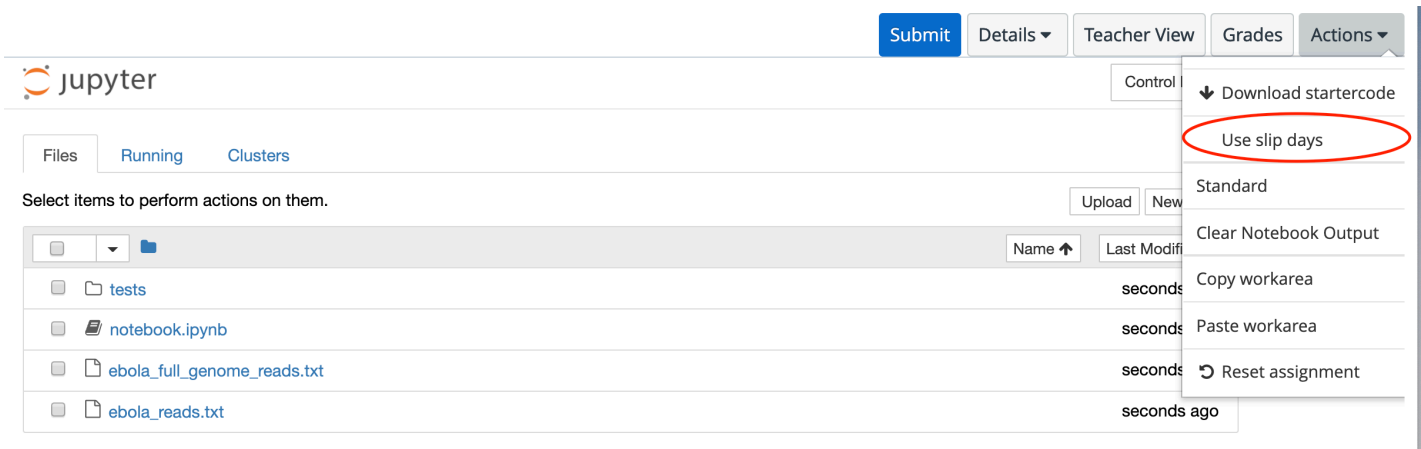
- Today
 - Pairwise sequence alignments
 - Dot plots
 - Scoring alignments
- Thursday
 - Dynamic programming
 - Global pairwise alignment

Purchase your Vocareum account

- Grace period has expired
- Don't let this interrupt your coursework

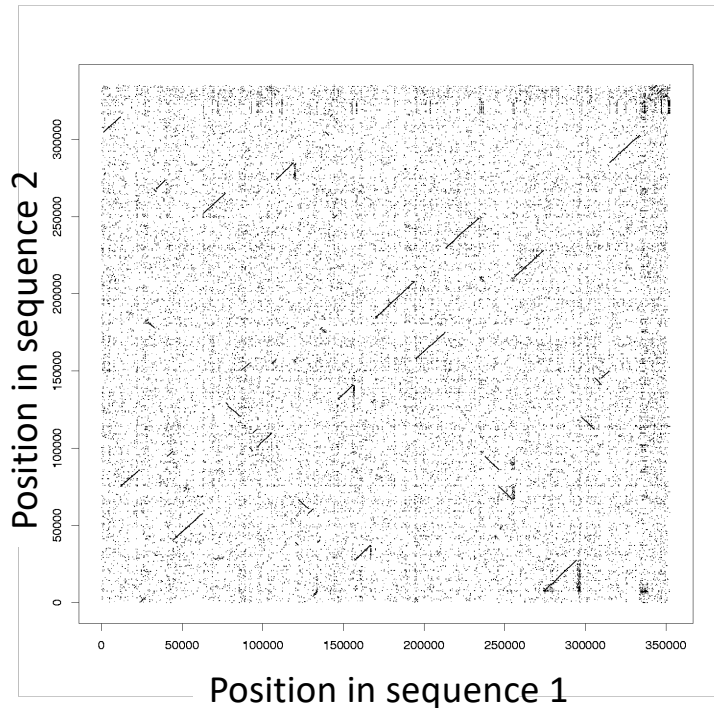
Using late (slip) days

- You are allowed 4 late/slip days total across semester for HW (there will be six HWs)
- To use slip days on your HW:



The screenshot displays the JupyterLab user interface. At the top, there is a navigation bar with buttons for 'Submit', 'Details', 'Teacher View', 'Grades', and 'Actions'. Below this, the 'jupyter' logo is visible on the left, and a 'Control' button is on the right. The main area is divided into tabs: 'Files', 'Running', and 'Clusters'. Below the tabs, a message states 'Select items to perform actions on them.' A file browser shows a list of items: a folder named 'tests', a file 'notebook.ipynb', and two text files 'ebola_full_genome_reads.txt' and 'ebola_reads.txt'. On the right side, an 'Actions' dropdown menu is open, showing options: 'Download startercode', 'Use slip days' (which is circled in red), 'Standard', 'Clear Notebook Output', 'Copy workarea', 'Paste workarea', and 'Reset assignment'.

Dot plots



- Dot indicates sequence similarity between two positions
- Diagonal lines formed by nearby dots indicate consecutive pairs of similar positions -> evidence of true homology relationship

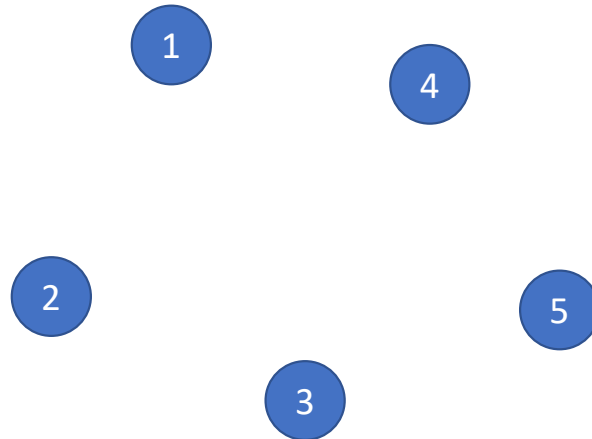
How do we handle...

- Insertions and deletions?
 - Will need to consider all possible alignments
 - Can do so efficiently via a dynamic programming algorithm (see Thursday's material)
- Small inversions
 - In this class, we will not consider events that disrupt the order and/or orientation of positions in the sequence
 - Handling such cases requires more specialized approaches
 - Whole-genome alignment (with rearrangement events) covered in BMI/CS 776

Another Greedy Algorithm Example (for fragment assembly)


Q:

| <u>edge</u> | <u>weight</u> |
|-------------|---------------|
| 1 → 4 | -5 |
| 2 → 3 | -4 |
| 1 → 3 | -3 |
| 5 → 2 | -2 |
| 3 → 1 | -1 |
| . | |
| . | |
| . | |

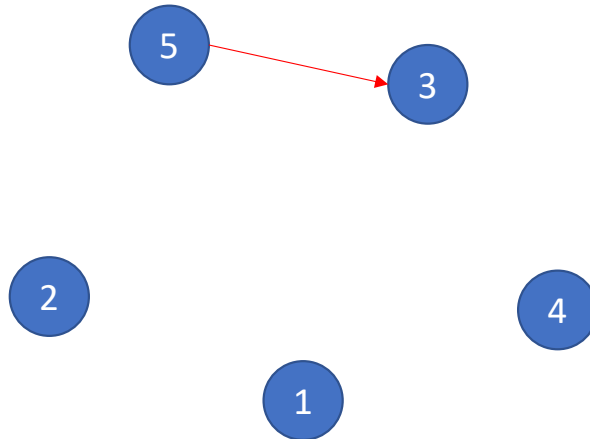


Sorted by edge weights

Another Greedy Algorithm Example – Iteration 1

Q: 

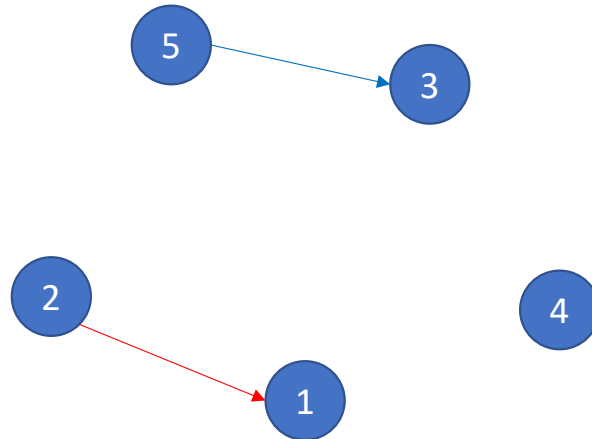
| <u>edge</u> | <u>weight</u> |
|-------------|---------------|
| 5 → 3 | -5 |
| 2 → 1 | -4 |
| 5 → 1 | -3 |
| 4 → 2 | -2 |
| 1 → 5 | -1 |
| . | |
| . | |
| . | |



Another Greedy Algorithm Example – Iteration 2

Q:

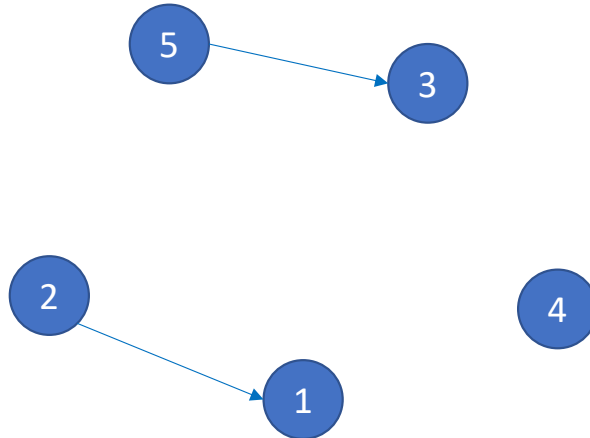
| <u>edge</u> | <u>weight</u> |
|-------------|---------------|
| 5 → 3 | -5 |
| 2 → 1 | -4 |
| 5 → 1 | -3 |
| 4 → 2 | -2 |
| 1 → 5 | -1 |
| . | |
| . | |
| . | |



Another Greedy Algorithm Example – Iteration 3

Q:

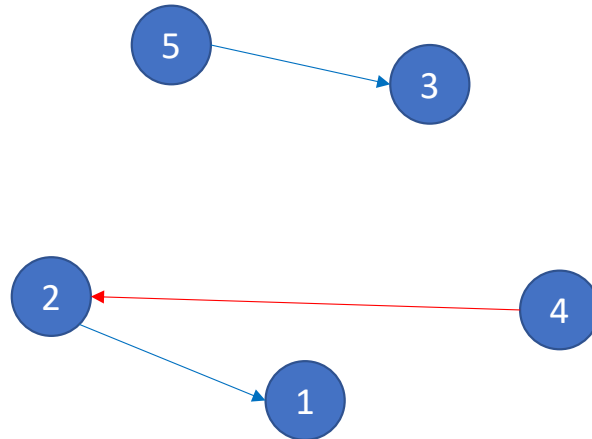
| <u>edge</u> | <u>weight</u> |
|-------------|---------------|
| 5 → 3 | -5 |
| 2 → 1 | -4 |
| 5 → 1 | -3 |
| 4 → 2 | -2 |
| 1 → 5 | -1 |
| . | |
| . | |
| . | |



Another Greedy Algorithm Example – Iteration 4

Q:

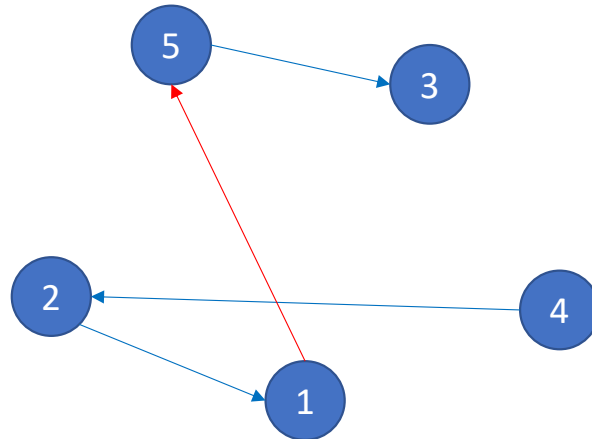
| <u>edge</u> | <u>weight</u> |
|-------------|---------------|
| 5 → 3 | -5 |
| 2 → 1 | -4 |
| 5 → 1 | -3 |
| 4 → 2 | -2 |
| 1 → 5 | -1 |
| . | |
| . | |
| . | |



Another Greedy Algorithm Example – Iteration 5

Q:

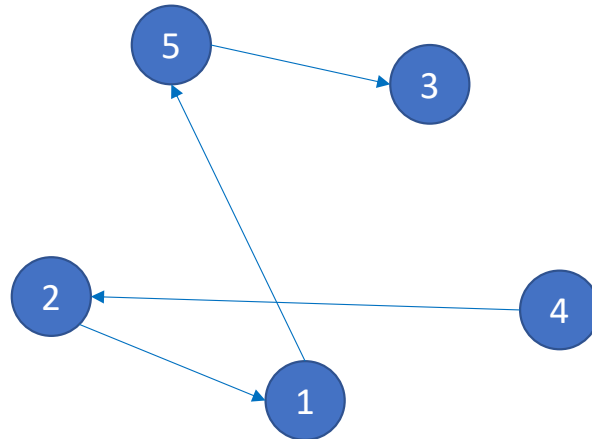
| <u>edge</u> | <u>weight</u> |
|-------------|---------------|
| 5 → 3 | -5 |
| 2 → 1 | -4 |
| 5 → 1 | -3 |
| 4 → 2 | -2 |
| 1 → 5 | -1 |
| . | |
| . | |
| . | |



Another Greedy Algorithm Example – Termination

Q:

| <u>edge</u> | <u>weight</u> |
|-------------|---------------|
| 5 → 3 | -5 |
| 2 → 1 | -4 |
| 5 → 1 | -3 |
| 4 → 2 | -2 |
| 1 → 5 | -1 |
| . | |
| . | |
| . | |



Path completed: [4, 2, 1, 5, 3]

For assembly, convert to
superstring, given vertex labels