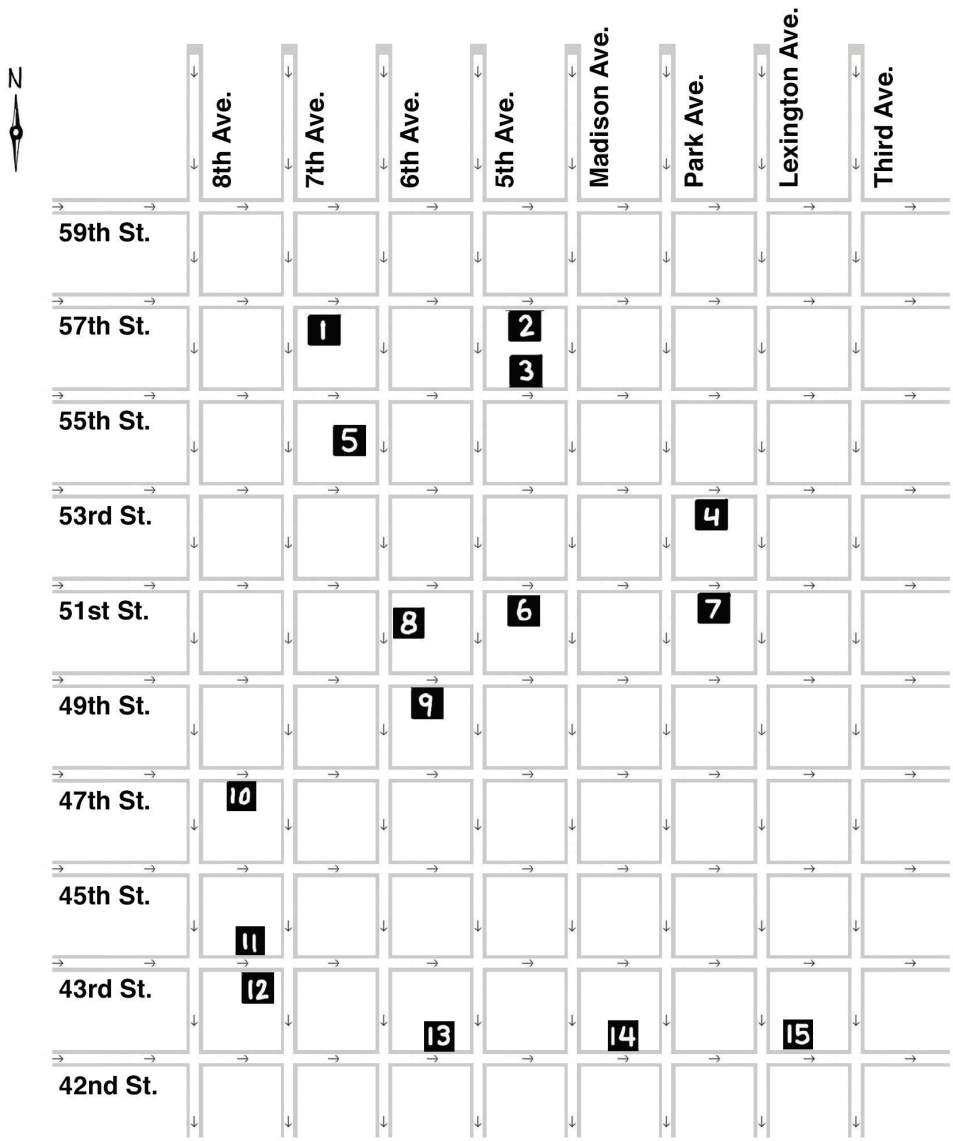
CS4054
Bioinformatics

Spring 2025

Rushda Muneer

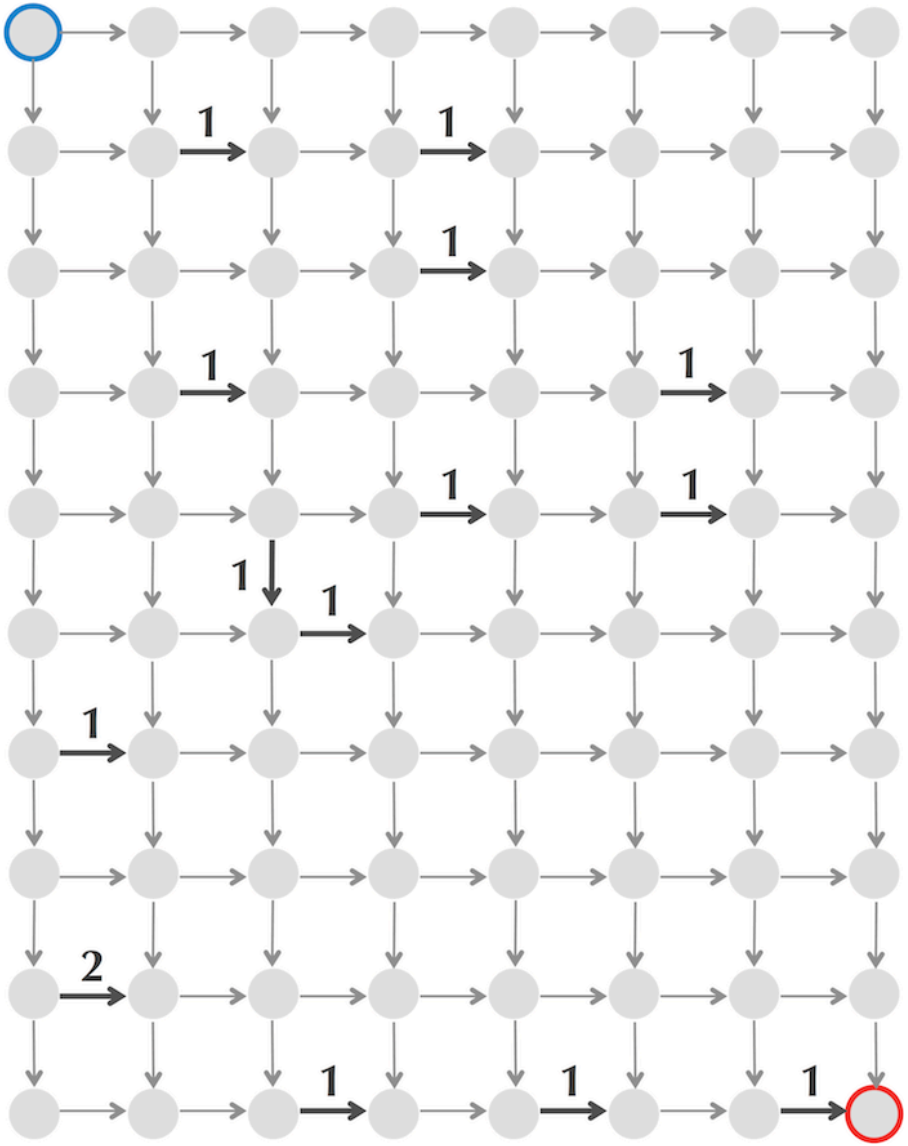
Manhattan Tourist Problem

- Imagine you are a tourist in Midtown Manhattan, and you want to see as many sights as possible on your way from the corner of 59th Street and 8th Avenue to the corner of 42nd Street and 3rd Avenue
- However, you are short on time, and at each intersection, you can only move south (↓) or east (→)
- How can we see the most sites if we move from 59th and 8th to 42nd and 3rd, moving south or east at each step?
- The challenge of finding a legal path through the city that visits the most sights is called the **Manhattan Tourist Problem**



Manhattan Tourist as a Network

- Weight of edge: number of attractions along the edge.
- The starting (blue) node is called the **source node**, and the ending (red) node is called the **sink node**
- Adding the weights along a path from the source to the sink yields the number of attractions along that path
- Goal: Find a longest path from source (top left) to sink (bottom right).
- Therefore, to solve the Manhattan Tourist Problem, we need to find a **maximum-weight path** connecting the source to the sink (also called a **longest path**) in *ManhattanGraph*.

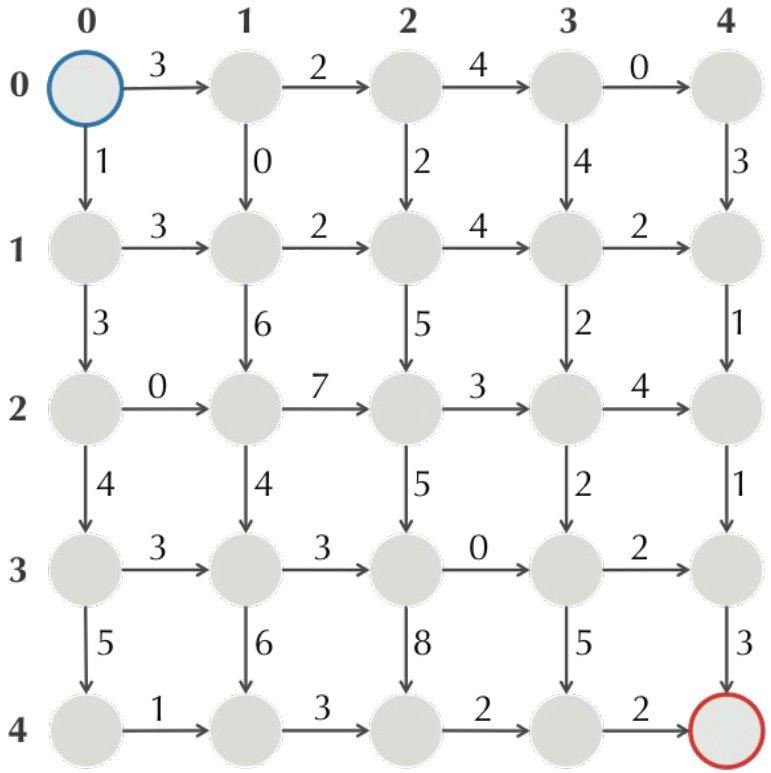


Toward a Computational Problem

- Manhattan Tourist Problem:
  - Input: A weighted n x m rectangular
  - Output: A longest path from source (0, 0) to sink (n-1, m-1) in the grid.

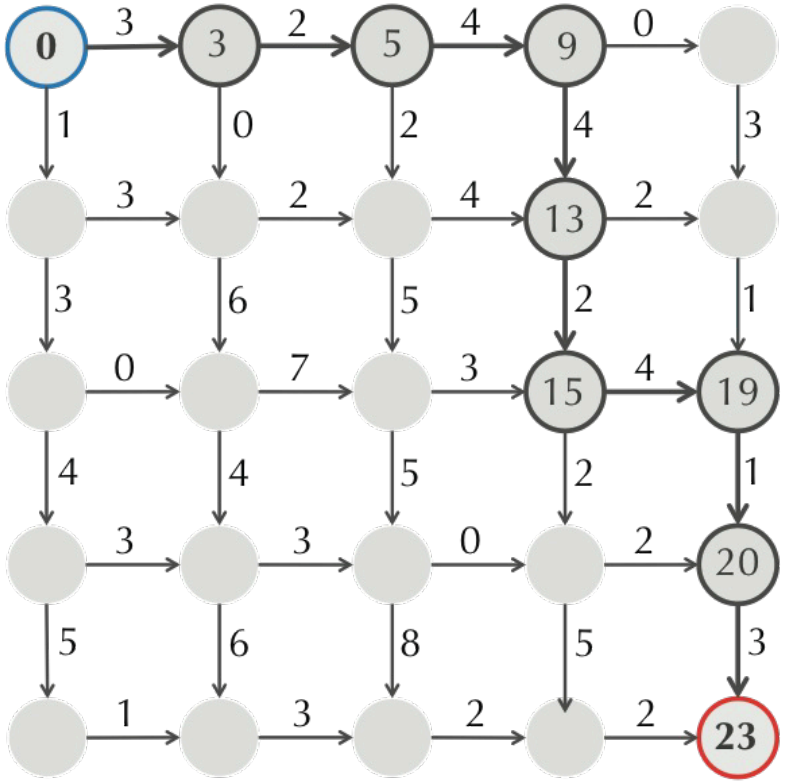
Designing a Manhattan Algorithm

- Exercise:
- What is the longest path in this city?
- What algorithm did you use?



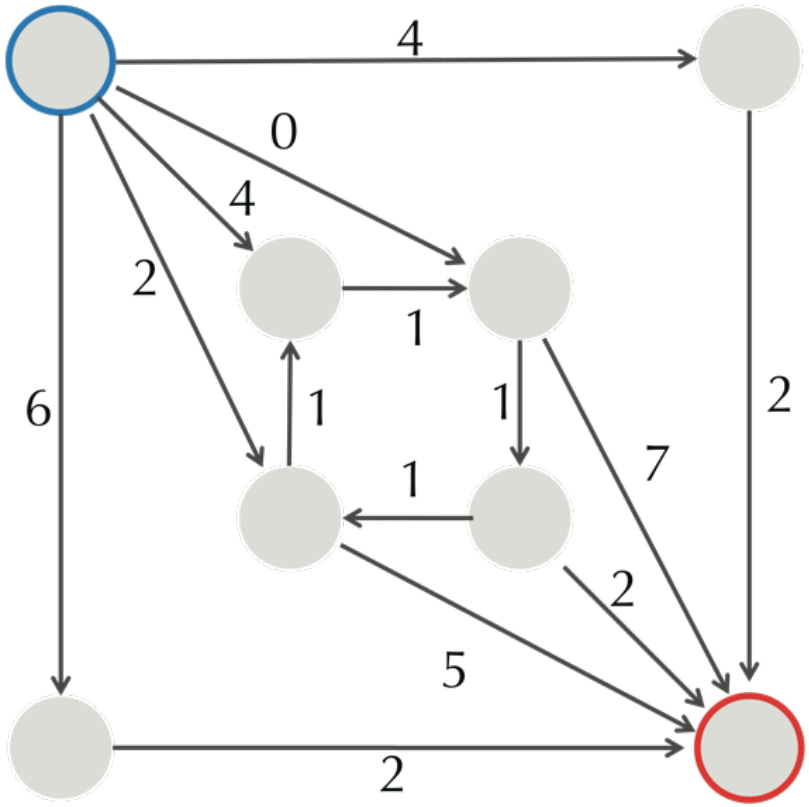
A “Greedy” Manhattan Algorithm Taking the Best Choice in Each Node

- Does the greedy algorithm solve the problem?
- No! Much like with genome assembly, we need a more clever approach.



Manhattan Tourist as a Network Problem

- Longest Path in a Directed Graph:
- Input: An edge-weighted directed graph with source and sink nodes.
- Output: A longest path from source to sink in the graph.
- What is the longest path in this graph?
- Cycles in graphs cause infinite paths



Generalizing Manhattan Tourist

- Directed acyclic graph (DAG): A directed graph that contains no cycles.
- Longest Path in a DAG Problem:
  - Input: An edge-weighted DAG with source and sink nodes.
  - Output: A longest path from source to sink in the DAG.
- But what does finding a longest path in a DAG have to do with sequence comparison?

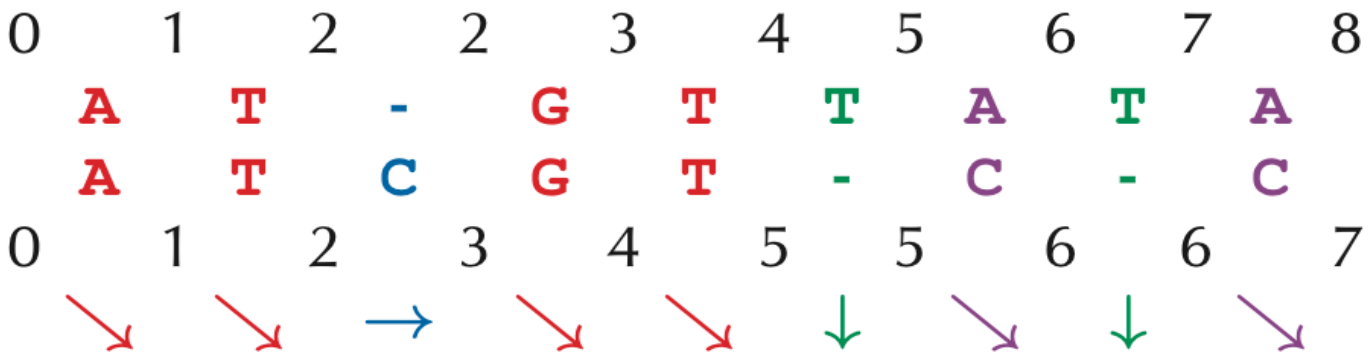
Returning to Sequence Alignment ...

```
A  T  -  G  T  T  A  T  A
A  T  C  G  T  -  C  -  C
```

Returning to Sequence Alignment ...

- We add two arrays of integers to an alignment of "ATGTTATA" and "ATCGTCC".
- The array [0 1 2 2 3 4 5 6 7 8] holds the number of symbols of "ATGTTATA" used up to a given column in the alignment.
- Similarly, the array [0 1 2 3 4 5 5 6 6 7] holds the number of symbols of "ATCGTCC" used up to a given column.

Returning to Sequence Alignment ...



- We add a third array [↘ ↘ → ↘ ↘ ↓ ↘ ↓ ↘] recording whether each column represents a **match**/**mismatch** (↘/↘), an **insertion** (→), or a **deletion** (↓).
- This third array corresponds to a path from source to sink in an 8 × 7 rectangular grid. The *i*-th node of this path is made up of the *i*-th element of [0 1 2 2 3 4 5 6 7 8] and the *i*-th element of [0 1 2 3 4 5 5 6 6 7]:

(0. 0) ↘ (1, 1) ↘ (2, 2) → (2, 3) ↘ (3, 4) ↘ (4, 5) ↓ (5, 5) ↘ (6, 6) ↓ (7, 6) ↘ (8, 7)

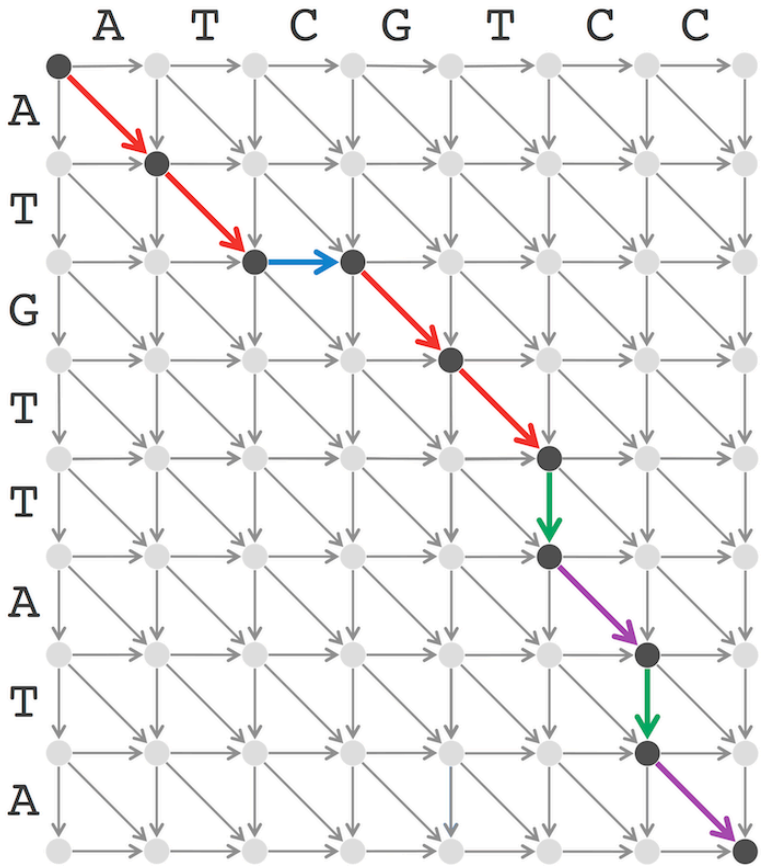
This is a path in a 2-D network!

Representing an Alignment as a Path in a Manhattan-like DAG

A T - G T T A T A

A T C G T - C - C

- We call this DAG the **alignment graph** of strings *v* and *w*, denoted *AlignmentGraph*(*v*, *w*)
- We call a path from source to sink in this DAG an **alignment path**.
- Every alignment of *v* and *w* can be viewed as a set of instructions to construct a unique alignment path in *AlignmentGraph*(*v*, *w*) where,
  - Each **match**/**mismatch**, **insertion**, and **deletion** corresponds to an edge ↘/↘, →, and ↓, respectively.
- Thus, each alignment of strings *v* and *w* corresponds to a path in *AlignmentGraph*(*v*, *w*), and vice-versa.



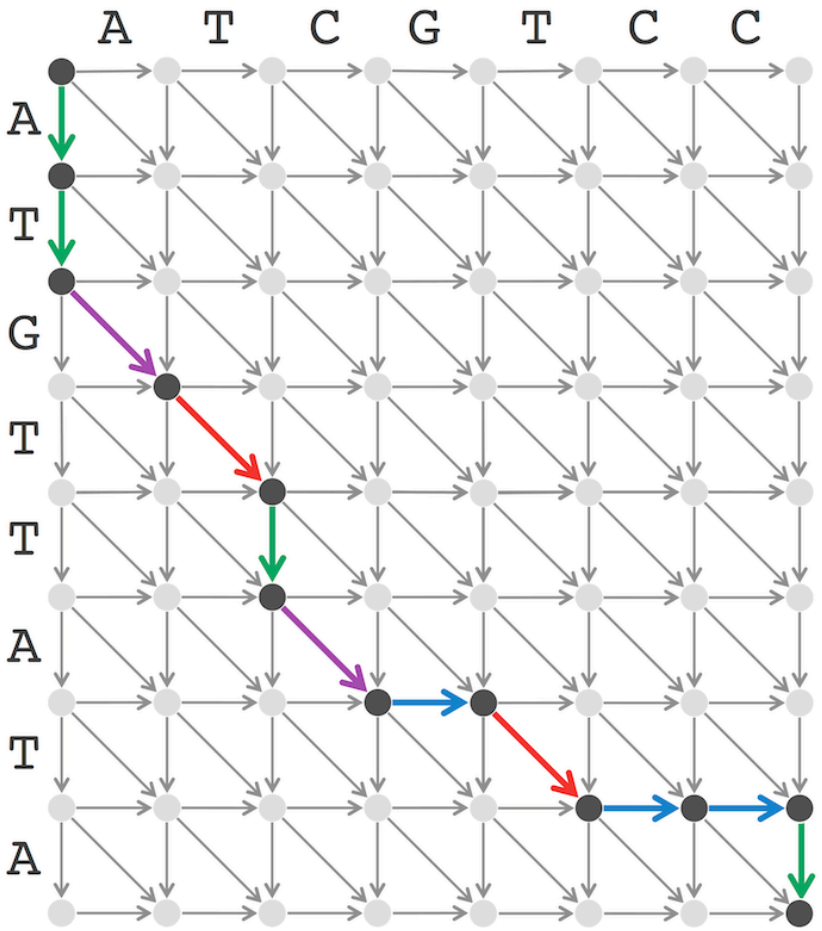
We can also construct an alignment from a path

Exercise: What alignment does this path correspond to?

Answer:

A T G T T A - T - - A

- - A T - C G T C C -



Solving the Symbol Matching Problem

- Symbol Matching Problem:
  - Input: Two strings.
  - Output: The greatest number of matched symbols in any alignment of the two strings.
- How can we use the alignment network to solve this problem?

Counting Matches Only

- Answer: If we weight the red edges as 1 and the other edges as 0, then a maximum-weight path from source to sink solves the Symbol Matching Problem!
- Thus, we need to design an algorithm for the Longest Path in a DAG Problem, but to do so, we need to know more about **dynamic programming.**