GRAPH TRAVERSAL
PATH FINDING AND GRAPH TRAVERSAL

- **Path finding** refers to determining the shortest path between two vertices in a graph.

- We discussed the Floyd–Warshall algorithm previously, but you may achieve similar results with the Dijkstra or Bellman-Ford algorithms.

- **Graph traversal** refers to the problem of visiting a set of nodes in a graph in a certain way.

- You can use depth- or breadth-first search algorithms to traverse an entire graph starting from a given "root" (source vertex).
Today we will discuss some heuristics for finding circuits (tours) on graphs. We are particularly interested in finding circuits that also have minimum cost. Remember that circuits can include the same vertex twice, but arcs only once. You may want to perform some pre-processing on the graph to handle such restrictions.
TERMINOLOGY: HAMILTONIAN CIRCUIT

- A Hamiltonian circuit is a path that visits every vertex exactly once and starts at ends at the same vertex.
- NB: Not all graphs may have a Hamiltonian circuit.
MINIMUM COST HAMILTONIAN CIRCUIT

- In a weighted graph, the minimum cost Hamiltonian circuit is that where the sum of the arc weights is the smallest.

- Finding the minimum cost Hamiltonian circuit on your bin service graphs is one option for route planning.

- Finding a Hamiltonian circuit can be difficult. This is a known NP-complete problem. Simply put, the problem may not be solvable in polynomial time and the complexity increases with the number of vertices.
HEURISTIC ALGORITHMS

- Heuristics work quite well for finding (no always optimal, but good) solutions, most of the time.

- Work relatively fast.

- Popular heuristics for finding minimum cost Hamiltonian circuits:
  - Nearest Neighbour Algorithm
  - Sorted Edges Algorithm
NEAREST NEIGHBOUR ALGORITHM

- Nearest Neighbour is a greedy algorithm – at every step it chooses as the next vertex the one connected to the current through the arc with the smallest weight.
- Only searches locally.
- Nodes already visited are ignored.
- Due to its greedy nature it may not find a solution.
- Finding a solution and its total cost depends on the start vertex chosen.
EXAMPLE, STARTING AT '3'
EXAMPLE, STARTING AT '3' (NO SOLUTION)

Arrived back at start point, but vertex 5 not included
EXAMPLE, STARTING AT '5'
EXAMPLE, STARTING AT '5' (CONT'D)

Solution: 5-0-1-2-3-4-5
Total cost: 20
SORTED EDGES ALGORITHM

- Also greedy, but has a more global view → takes slightly more time to find a solution.
- First sorts all the (directed) edges in ascending order of their weights.
- Adds sorted edges one at the time, unless adding a new one leads to three arcs entering/leaving a node, or creates a circuit that does not include all vertices.
- Skips the arcs that violate these rules.
- Keeps adding arcs until finding a Hamiltonian circuit (or no more arcs left).
- Stops when a solution is found, even if there are arcs left.
EXAMPLE
EXAMPLE (CONT'D)

4-0  1  0-5  4
4-1  1  4-5  4
2-3  2  5-0  4
3-4  2  5-4  4
3-5  2  1-2  5
0-1  3  5-3  5
1-0  3

4-0  1
4-1  1
2-3  2
3-5  2
0-1  3
1-0  3 (0-1-0 cycle)

0-5  4 (3 arcs at '0')
4-5  4 (3 arcs at '4')
4-0  1
4-1  1
2-3  2
3-5  2
0-1  3
5-0  4 (3 arcs at '0')
5-4  4 (3 arcs at '4')
EXAMPLE (CONT'D)

No more arcs left

4-0 1
4-1 1
2-3 2
3-4 2
3-5 2
0-1 3
1-0 3
0-5 4
4-5 4
5-0 4
5-4 4
1-2 5
5-3 5

5 (3 arcs at ‘1’)
5 (5-3-3 cycle)
No solution found.
BRUTE FORCE ALGORITHM

- When the number of vertices is small, a 'brute force' approach could be feasible.
- Find all paths that visit all vertices once and pick the one with the lowest cost.
- Guaranteed to find a solution (if there exists one), and this will be optimal.
OTHER APPROACHES

- A Hamiltonian circuit may **not always** be the path that visits all the vertices and **has the lowest cost**.

- Sometimes visiting a node more than once can give circuits that are more cost effective.

- The problem could also be seen as an instance of finding an Eulerian circuit (cycle).
EULERIAN CIRCUITS

- An Eulerian path visits every edge (arc) exactly once.
- The idea dates back to birth of Graph Theory, when Leonhard Euler was asked to determine a path that traverses all of the 7 bridges in Koenigsberg exactly once.

Source: Nature Biotechnology
EULERIAN CIRCUITS

- An Eulerian circuit begins and ends at the same vertex.
- Euler realised a solution to the original problem did not exist, but found conditions that a graph must fulfil to have Eulerian paths.
- For your problem (directed graphs), it is important to know Eulerian circuits exist if and only if every vertex has equal in degree and out degree.
EXAMPLES

No solution

0-1-2-3-4-0-5-3-5-0

Total cost: 28
HIERHOLZER'S ALGORITHM

- Assuming you have verified the necessary condition the graph must meet to have an Eulerian circuit.
- Pick any vertex as a starting point.
- Add arcs to build trail, until returning to the start. You cannot get stuck at other vertices.
- If there are vertices on this tour but have adjacent arcs not included, start new trail from these.
- Join the tours formed this way.
EXAMPLE
THE PROBLEM YOU ARE TRYING TO SOLVE

- The graphs you will be given may not have Hamiltonian or Eulerian circuits.
- Even if they do, at start of service in an area you may not need to visit ALL the bins (vertices).
- It is likely you will need to do some pre-processing, e.g.
  - Build complete graphs starting from the original ones.
  - Remove some arcs.
- And work only with a subset of vertices (the bins that require service).
THE PROBLEM YOU ARE TRYING TO SOLVE

- There is no BEST solution.

- Depending on the graph structure/size, some may return more cost effective circuits than others.

- But may also take more time.

- Try to experiment with several, but aim to have at least ONE decent implementation.