Graph Theory

FORMULAS YOU MUST KNOW

$\deg(v) =$ number of edges at v — a loop counts twice
Degree:

no loops and no repeated edges; then each edge just links two distinct vertices
Simple graph:

K_n joins every pair, so it has $\dfrac{n(n-1)}{2}$ edges and every vertex has degree $n-1$
Complete graph:

each edge carries a number (distance, cost, time) to be minimised
Weighted graph:

edges have a direction; each vertex has an in-degree and an out-degree
Directed graph:

a connected graph with no cycles; n vertices are joined by exactly $n-1$ edges
Tree:

$\deg(i) =$ the sum of row i (here $2, 3, 4, 3, 2$) — read straight off the matrix
Degree:

replace each 1 with the edge's weight to get the weighted adjacency table used by the algorithms below
Weighted table:

$\sum_i \deg(i) = 2E$, where E is the number of edges
Handshake lemma:

$E = \frac{1}{2}\sum_i \deg(i)$ — halve the degree sum
Edge count:

$(A^k)_{ij} =$ the number of walks of length k from vertex i to vertex j
Walks:

$(A^2)_{ij}$ counts the common neighbours of i and j; $(A^2)_{ii} = \deg(i)$
Length 2:

KEY METHODS

- Learn the walk vocabulary: a walk is any run of edges (vertices and edges may repeat); a trail repeats no edge; a path repeats no vertex; a cycle is a closed path back to its start. A subgraph is any graph formed from a selection of the vertices and edges.
- For a simple undirected graph A is symmetric with a zero diagonal. That symmetry is a free error-check when you type the matrix into the GDC.
- A neat consequence: the number of odd-degree vertices is always even. Here that pair is B and D — exactly the vertices the Chinese postman (§7) will have to worry about.

EXAM TRAPS TO AVOID

- In a directed graph "degree" splits into in- and out-degree, and the two need not match. Only for a simple undirected graph is a vertex's degree just its number of neighbours.
- For a directed graph A is generally not symmetric: A_{ij} (an arrow $i \to j$) can differ from A_{ji}. Keep the rows as "from" and the columns as "to".
- The degree sum is always even. If yours comes out odd you have miscounted — a valid graph can never have an odd total degree.
- "Length" means the number of edges, so a walk of length n needs A^n, not A^{n-1}. Walks may revisit

- Enter A once, evaluate A^k, and read off the single entry in row i, column j. Sum all the entries of A^k only when the question asks about the whole network.
- Both algorithms give the same total weight. Kruskal sorts edges globally (best by hand when there are few edges); Prim grows one connected blob outward from a start vertex (best on dense graphs).
- Once a vertex holds the smallest un-fixed label it can never improve, so fix it and move on. Back-track the fixed labels at the end to read off the actual route, not just its length.
- Chinese postman is about covering every edge (Eulerian). Contrast the next section, the travelling salesman, which is about visiting every vertex (Hamiltonian) — a completely different problem.

vertices and edges — that is why $A \to B \to A \to B$ is a legitimate length-3 walk.

- Never add an edge whose two ends are already connected — it closes a cycle. Take the next-cheapest instead, and stop the moment you have $n - 1$ edges.
- Dijkstra minimises total weight, not the number of edges. A three-edge path can easily beat a one-edge path if that single edge is heavy — always add the weights, never count the hops.
- The repeated distance is the shortest path between the two odd vertices, which may run through another vertex rather than along the direct edge — here 11 via C beats the direct 12.