Chinese Postman Problem: Find shortest (or lowest weight) closed walk that visits all edges (note some edges may be repeated).

Find optimal Eulerization:

1.) Find all odd degree vertices (note: $\exists$ an even number of odd degree vertices).

2.) For each possible pairing (perfect matching) of odd degree vertices, v_i, v_j,

Find the shortest path or minimum weight path, $P_{i,j}$, between each pair v_i, v_j.

3.) Take optimal pairing (perfect matching), and

duplicate each edge in the path $P_{i,j}$ for each i, j in the optimal pairing
Note this creates an Eulerian graph G'

4.) Find Eulerian circuit in the Eulerian graph G'.

5.) Apply to the original graph by repeating edges that were duplicated to create the Eulerian graph G'.

Konigsberg bridge with made up distances.

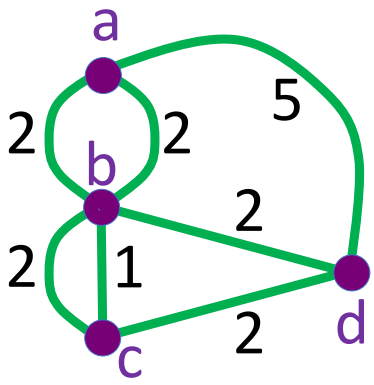
Find shortest tour that visits each bridge and returns to starting point.

I.e. Find lowest weight closed walk.
Thus find optimal Eulerization.

Note all vertices have odd degree.

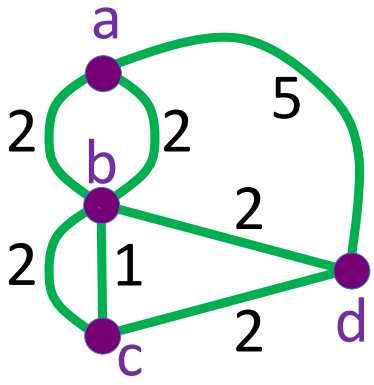
pair: bc

pair: ad



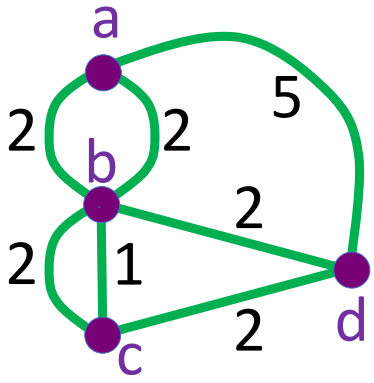
pair: ab

pair: cd



pair: ac

pair: bd



Algorithm 4.2 step 2: Construct a weighted complete graph on $2k$ vertices in which vertex v_i is joined to vertex v_j by an edge with weight $w(P_{i,j})$:

Find an optimal perfect matching.

Defn: A subset, M, of edges of a gragh G is a *matching* if $e, e' \in M$, then e, e' are not incident to the same vertex.

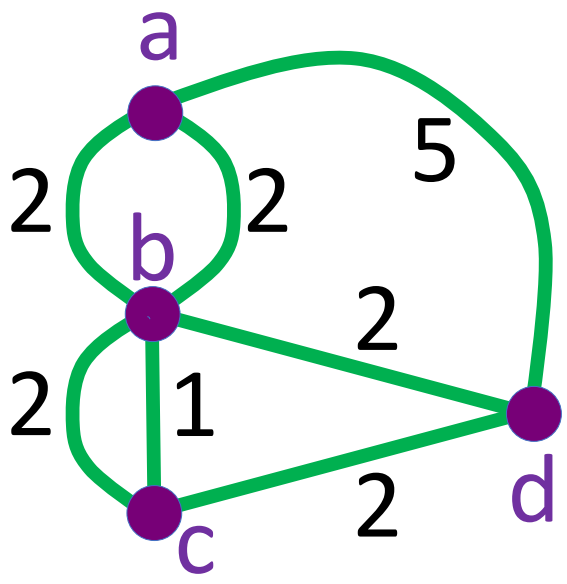
Defn: A matching is *perfect* if every vertex in G is incident to some edge in M.

Some Applications of Matching:

Marriage.

Assignment of tasks to individuals where one task is assigned to each individual.

Eulerizing a graph.



Find shortest tour that visits each bridge and returns to starting point. I.e. Find lowest weight closed walk.

I.e. Find lowest weight closed walk.

If all vertices have even degree, then lowest weight closed walk = Eulerian circuit.

If exactly 2 vertices, u, w, have odd degree, apply Dijkstra's algorithm to u (or w).

If $2k$ vertices have odd degree, form complete graph on $2k$ vertices and use Dijkstra's algorithm $2k - 1$ times to determine edge weights. Then find optimal pairing.