1 Maximum weight perfect matching

As a subroutine to the Chinese postman problem, we want solve this question.

Maximum weight perfect matching
Input: A graph G with weights w on its edges
Output: A perfect matching M of G maximizing $\sum_{e\in M} w_e$. (Or "G has no perfect matching")

Recall the prefect matching polytope is integral.
(PMP)

$$\begin{aligned}
&\sum_{u\in N(v)} x_{uv} = 1 \ \forall v \in V(G)\\
&\sum_{e\in\delta(S)} x_e \geq 1 \quad \forall S \subseteq V, |S| \geq 3, |V-S| \geq 3 \text{ odd}\\
&0 \leq x_e \leq 1 \qquad \forall e \in E(G)
\end{aligned}$$

In fact, vectors in (PMP) are convex combinations of characteristic vectors of matchings (just check no other integer solutions are possible).

So if we could optimize over it, we could set the objective to max $\sum_e w_e x_e$. But there is an exponential number of constraints. If we could decide if a solution is feasible in polynomial time (polynomial in the size of the input graph), we could use the Ellipsoid method. An algorithm which does this is a *separation oracle*.

The first and third set of constraints can simply be checked one by one as there is a polynomial number of them. So it remains to check the second set of constraints.

So given G weighted by x, we have to decide if there is an odd cut of weight less than 1. We now see how we can find a minimum weight odd cut in a weighted graph. Then it is just a matter of testing if the optimum is less than 1.

Minimum weight odd cut
Input: A graph G with weights x
Output: A minimum weight cut $\delta(S)$ with $|S|$ odd.

We can certainly find the minimum weight cut in G (using flows). If we are lucky and this cut is odd.

Turns out if it is even then this cut (or its complement) contains a minimum odd cut.

Lemma 1.1. *For any even cut S, either S or $V(G)-S$ contains a minimum odd cut.*

Notation: $x(F) = \sum_{f\in F} x_f$

Proof. Take any mininum even cut S and mininum odd cut S^*. Without loss of generality, $S \cap S^*$ is odd (or replace S by $V(G)-S$) and so

$$x(\delta(S^*)) + x(\delta(S)) \geq x(\delta(S\cap S^*)) + x(\delta(V-S-S^*))$$

by comparing the multiset of edges in these cuts.
Since S^* is minimum, $x(\delta(V - S - S^*)) \geq x(\delta(S^*))$.
So $x(\delta(S)) \geq x(\delta(S \cap S^*))$. By minimality of S, $x(\delta(S \cap S^*))$ is a minimum odd cut. □

This suggests a recursive algorithm:
Find a minimum cut S and if its even, recurse on both sides.
How can we make sure we only find (minimum) cuts that are subset of S when we recurse? Contract $V(G) - S$ to a single vertex.

Both graphs G/S and $G/(V - S)$ are smaller as both S and $V - S$ are even. Since the cut S^* we are looking for is odd, it contains a vertex of S and is missing a vertex from S so we can look for $s \in S$ to $t \in S$ cuts for all pairs $s, t \in S$.

But the contracted vertex has the wrong (representative) parity (we contracted an even number of vertices to one single vertex (and 1 is odd)). So we keep track of a set V_{even} of even vertices (from these contractions) and look for cuts between the remaining vertices $V(G) - V_{\text{even}}$ (each side of any minimum odd cut contains at least one vertex from this set).

Algorithm: `min_odd_cut`(G, V_{even})

1. Find the minimum cut S that separates vertices in $V(G) - V_{\text{even}}$.
2. If S is odd, return S
3. Else return $\arg\min($`min_odd_cut`$(G/S, V_{\text{even}} \cup \{s\}),$ `min_odd_cut`$(G/(V - S), V_{\text{even}} \cup \{s\})$ where s is the new vertex created from from contraction.

We can find a minimum cut between odd vertices by testing all pairs of odd vertices for a minimum cut between them. But can do it faster by first fixing one point of the pair and iterating over all other odd vertices.

Running time analysis: $T(n) = T(n_1) + T(n + 2 - n_1) + p(n)$ where $p(n)$ is the time needed for $n = |V(G)|$ max flows (in fact $|V(G) - V_{\text{even}}|$ is enough).