Exercise 1

We define $\bar{v}$ as the map that sends the simplices in C to the images given by the map v, i.e. for all simplex $\{p_0, \dots, p_k\} \in C$,

$$\bar{v} : \{p_0, \dots, p_k\} \mapsto \{v(p_0), \dots, v(p_k)\}$$

To show that this is a simplicial map, consider a simplex $\sigma = \{p_0, \dots, p_k\} \in C$. By the definition of C, the vertices $\{p_0, \dots, p_k\}$ of σ in $G(P)$ form a clique. Then, by definition of the graph-induced complex $\mathbb{G}(Q, G(P))$,

$$\bar{v}(\sigma) = \tau \in \mathbb{G}(Q, G(P))$$

Indeed, for every vertex τ_i for τ, $\tau_i = v(p_i)$ for a vertex p_i that is included in a clique $\{p_0, \dots, p_k\}$ of $G(P)$, where v is the map that sends each point of P to its closest point of Q.
Now consider a simplicial complex K with $V(K) = Q$ for which v has a simplicial extension $\bar{v} : C \to K$. Let $\tau \in \mathbb{G}(Q, G(P))$. Then, for each point τ_i of τ, $\tau_i = v(p_i)$ for p_i a point that is included in a clique $\{p_0, \dots, p_k\}$ of $G(P)$. Then, by definition of C, $\sigma = \{p_0, \dots, p_k\} \in C$. Therefore $\bar{v}(\sigma) = \tau$ is well defined as a simplex in K, meaning that $\tau \in K$

Exercise 2

We fix $p \geq 0, \alpha := \|f - g\|_\infty$. Let K be a simplicial complex containing a $(p+1)$-simplex σ such that $K \setminus \{\sigma\}$ has just one p-dimensional hole, which is destroyed by σ. We then define functions f and g as follows:

$$f : K \setminus \{\sigma\} \to \{0\}, f(\sigma) = x$$

$$g : K \setminus \{\sigma\} \to \{\alpha\}, g(\sigma) = x$$

for an $x > \alpha$ to be defined.
Then, $\|f - g\|_\infty = \alpha$, and the diagrams of the sublevel set filtrations $DGM_p(\mathcal{F}_f), DGM_p(\mathcal{F}_g)$ will have just one off-diagonal point corresponding to the birth of the p-dimensional hole in $K \setminus \{\sigma\}$ and its death when σ is added.
For $\mathcal{F}_f$, the coordinates of this point are $(0, x)$, since the hole is born when f reaches value 0, and dies when f reaches value x.
For $\mathcal{F}_g$, the coordinates of this point are (α, x), since the hole is born when g reaches value α, and dies when g reaches value x.
Therefore, it suffices to define x such that the distance between these two points in the "best" bijection as defined in the bottleneck distance is smaller than the distance between either of these points and the diagonal. This would then imply that $d_b(DGM_p(\mathcal{F}_f), DGM_p(\mathcal{F}_g)) = \alpha = \|f - g\|_\infty$, since $\|(0, x) - (\alpha, x)\|_\infty = \alpha$.
To satisfy the above condition, it suffices to have $x > 3\alpha$, since $\|(\alpha, 3\alpha) - diag\|_\infty = \alpha$, thus

$$\|(\alpha, x) - diag\|_\infty > \alpha = \|(0, x) - (\alpha, x)\|_\infty$$

for all $x > 3\alpha$, meaning that the best bijection couldn't map the point (α, x) to the diagonal.

Exercise 3

Definition 1 (Bottleneck cost)**.** *Given a weighted bipartite graph $G = (A \sqcup B, E, w)$, with $|A| = n = |B|$, a weight function $w : E \to \mathbb{R}_+$, and a perfect matching $M \subseteq E$, the bottleneck cost is defined as* $\max\{w(e) \mid e \in M\}$*, the maximal weight of its edges. T*

As we assumed during the lecture, the persistence diagrams X, Y consist of finitely many off-diagonal points with finite multiplicity (and all the diagonal points with infinite multiplicity). In this case, the task of computing $W_\infty(X, Y)$ can be reduced to a bipartite graph matching problem. Let X_0, Y_0 denote the off-diagonal points of X and Y, respectively. If $u = (x, y)$ is an off-diagonal point, then we denote its orthogonal projection on the diagonal $((x + y)/2, (x + y)/2)$ as u', which is the closest point to u on the diagonal. Let X_0' denote the set of all projections of X_0, that is, $X_0' = \{u' \mid u \in X_0\}$. With Y_0' defined analogously as $\{v' \mid v \in Y_0\}$, we define

$U = X_0 \cup Y_0'$ and $V = Y_0 \cup X_0'$; both have the same number of points. We define the weighted complete bipartite graph, $G = (U \sqcup V, U \times V, c)$, whose weights are given by the function

$$c(u,v) = \begin{cases} \|u - v\|_\infty & \text{if } u \in X_0 \text{ or } v \in Y_0 \\ 0 & \text{otherwise} \end{cases}.$$

Lemma 1 (Reduction lemma)**.** *Bottleneck distance between X and Y equals the bottleneck cost of G, [1].*

Lemma 2. *Let $G[r]$ be the subgraph of G that contains the edges with weight at most r. The bottleneck distance of G is the minimal value r such that $G[r]$ contains a perfect matching, [1].*

Therefore we can use the previous lemma do create the following algorithm:

1. Initialize a lower bound l and an upper bound u on the bottleneck distance of G. A natural choice for u is the maximum weight of any edge in G, and a natural choice for l is 0.
2. While $l < u$:
   - Compute the midpoint m of l and u.
   - Compute the subgraph $G[m]$ of G that contains only edges with weight at most m.
   - Check whether $G[m]$ contains a perfect matching using any of the standard algorithms for finding perfect matchings, such as the Hopcroft-Karp algorithm or the Blossom algorithm.
   - If $G[m]$ contains a perfect matching, set u to m. Otherwise, set l to $m + 1$.
3. Return l, which is the bottleneck distance of G.

Algorithm 1 Algorithm for computing Bottleneck distance between two persistence diagrams

Require: A weighted graph $G = (V, E)$ with non-negative edge weights.
Ensure: The bottleneck distance of G.

```
Let l ← 0 and u ← max_{(u,v)∈E} w(u, v)
while l < u do
    Let m ← ⌊(l + u)/2⌋
    Compute the subgraph G[m] of G that contains only edges with weight at most m.
    Check whether G[m] contains a perfect matching using any standard algorithm.
    if G[m] contains a perfect matching then
        Set u ← m
    else
        Set l ← m + 1
    end if
end while
return l
```

Correctness The correctness of the algorithm follows from the fact that the bottleneck distance is a monotonically increasing function of r: if $G[r]$ contains a perfect matching, then so does $G[r']$ for any $r' \geq r$. Therefore, the algorithm maintains the invariant that the bottleneck distance is between l and u at each iteration of the loop, and terminates with $l = u$ equal to the smallest value of r such that $G[r]$ contains a perfect matching.

Runtime Computing $G[m]$ takes $O(|E|)$ time since we can simply iterate over all the edges in E and include only those with weight at most m. Checking whether $G[m]$ contains a perfect matching can be done using any standard algorithm, such as the Hopcroft-Karp algorithm or the Blossom algorithm. The Hopcroft-Karp algorithm has a time complexity of $O(|V|^{2.5})$, while the Blossom algorithm has a time complexity of $O(|E||V|^2)$ in the worst case, but can be improved to $O(|V|^3)$ using a series of optimizations.
The number of iterations of the loop is $O(\log W)$, where W is the maximum weight of any edge in G. This is because at each iteration, the size of the range $[l, u]$ is divided by a factor of 2. Therefore, the overall time complexity of the algorithm is $O(|V|^{2.5} \log W)$ using the Hopcroft-Karp algorithm or $O(|V|^3 \log W)$ using the Blossom algorithm.
Note: If we want the algorithm to be polynomial in the number of input points (and not the "description size" of the diagrams) we cannot do binary search, since logW may be superpolynomial in n. Thus, technically a linear search would be needed. However in practice this would never be an issue and binary search would obviously be better (you could also binary search in the list of weights after sorting all the weights, this would combine both benefits)

References

[1] Michael Kerber, Dmitriy Morozov, and Arnur Nigmetov. Geometry helps to compare persistence diagrams. *ACM J. Exp. Algorithmics*, 22, sep 2017.