Evan Zhang | z5383657

Question 3

3.1

Begin by constructing a bipartite flow network as follows:

- The left-hand side vertices represent boys
- The right-hand side vertices represent girls
- Source s and sink t
- Connect s to each boy vertex with capacity 1
- Connect each girl vertex to t with capacity 1
- Connect each boy vertex to each of their liked girls with capacity 1

From this flow network construction, we run Edmonds-Karp to find the maximum flow and look to see whether the max flow is equal to n. If it is, add 1 to the number of songs able to be played. Reset all source and sink edges back to 1 and remove all boy-girl edges that were used. Repeat until max flow is less than n and you will have the number of songs able to be played with k = 0, as all pairs must be like-pairs for a song to be played.

Each repeat is simulating one song. We remove any pairs formed to simulate pairs existing in the past. They are removed as a pair in the past may need to exist for that song for that song to be played and exchanging that pairing for a different pairing at a different song is incorrect. (e.g., if boy a and girl a are the only pair left and in the next song boy b reverses that pair making it so boy a pairs with girl b instead, it will be as if boy a paired with girl b and girl b also paired with their original boy for the first song).

The time complexity is $O(SVE^2)$ where S = max songs, V = 2n + 2 and $E \leq 2n + n^2$, so the algorithm runs in time polynomial in n as required.

3.2

Do the same as 3.1 except with an addition to the flow network:

- For each boy vertex, there will be a boy-dislike vertex.
- Connect each boy vertex to their corresponding boy-dislike vertex with capacity k
- Connect each boy-dislike vertex to each girl the boy dislikes with capacity 1

If a boy-boy-dislike edge is used, keep its current weight. If a boy-dislike-girl edge is used, remove it, to simulate a pair having existed in the past. When max flow is less than n, you will have the number of songs able to be played with arbitrary k.

Again, each repeat is simulating one song.

The time complexity is $O(SVE^2)$ where S = max songs, V = 3n + 2 and $E \leq 3n + 2n^2$, so the algorithm runs in time polynomial in n as required.