1a.

* n^2 doubled will be 4 times slower
* n^3 doubled will be 8 times slower
* 1000n^2 doubled will be 4 times slower
* nlogn doubled will be 2 times slower
* 2^n doubled will be n^2 times slower
* 2^2^n doubled will be n^4 times slower

1b.

* n^2 where n = n+1 will be ((n+1)^2-n^2) slower
* n^3 where n = n+1 will be ((n+1)^3-n^3) slower
* 1000n^2 where n = n+1 will be (1000(n+1)^2-(1000n^2) slower
* nlogn where n = n+1 will be ((n+1)log(n+1)-nlogn) slower
* 2^n where n = n+1 will be (2^(n+1)-(2^(n)) slower
* 2^2^n where n = n+1 will be ((2^2^(n+1))-(2^2^n)) slower

1c.

* n^2, 6million
* n^3, 33019
* 10000n^2, 60k
* nlogn, 3551138 (using Newton’s Method)
* 2^n, 45
* 2^(2^n), 5

2a. Least to greatest: f2,f3,f6,f1,f4,f5

2b. Least to greatest: g2,g1,g4,g5,g3,g7,g6

3. If I’m not mistaken, those songs are simply of order n^2. The nth verse of the song has n parts to go through. So if each nth verse has n parts to it, that makes the song have an order n^2 runtime.

4a. Bubble Sort is order n^2 in worst case since it will have to swap every item in the list (n-1) times due to being in the reverse sorted order . If it does that in a list of n size, then this makes it n^2 order. (n(n-1)/2)

4b. Best case is order n since it simply needs to run through the list once and check if everything is in its proper order, thus it is linear.

4c. The tightest bounds would be the best and worst case of the algorithm. Bubble sort is at most n^2 order and at best n order, but on average it would be n^2 order. Even if the list is minorly unsorted, it starts diving into n^2 order by the nature of the algorithm.

4d. Theta class of Bubble Sort would be n^2. Again this is due to the average case being n^2 order and is the most consistently occurring order.