Problem Set 3 - Dynamic Programming

Problem B

I remember Kefa and Dishes being quite a difficult one to wrap my head around - both because of my lack of experience and understanding of Dynamic Programming, but also because of my lack of intuition for bit manipulation. This one was a loong trudge, but got there in the end and learned a lot from it.

Ultimately, the purpose of having the bitset in this question was to track all of the possible different sets of Dishes that Kefa could have taken (of size M for the final answer). I can explain all of this swiftly in retrospect, but actually coming to all of these understandings took a very very long time when first approaching this question - with my level of knowledge in week 3. And then of all of the final states with sets of size m, we would choose the largest value.

The dp state I ended up going with was not purely what subset we chose however. We also needed to know what was chosen last in order to relate it with prior states, as pairings between consecutive dishes eaten can give bonus satisfaction as per the assignment spec. Adding that into the state, we could build a recurrence, where dp[S][i] is the maximum satisfaction with the S dishes eaten, with ‘i’ as the last dish eaten. Then to calculate that value, we would find the max of all the different previous states: dp[S’][j] (where S’ is the prior set of S without i) plus, the satisfaction of eating dish i, plus the additional benefit of eating i after j. The base case in this question would be when the subset S contains only a single dish i, in which case dp[S][i] =a[i], as the satisfaction is simply the value of eating that dish with no prior bonuses.

After that relation was set, coding it was a different story - it’s pretty trivial now, given my accrued experience with bitset in other questions, but back then, finding all subsets of a set, testing if an element i was in a bit set - all of these things I found quite confusing to implement.

Got there in the end though!