Hi, I am Xuanye. In my part, I will talk about how to use Dynamic Programming to solve knapsack with repeat objects problem. In Dynamic programming, we will store the sub-problem solution in memory so that we don’t need to compute it again and again like what we do in recursive, which is a better way and have smaller time complexity. Just now as Joe mention, this time we can use those objects unlimited times. So this time constructing the recursive definition, when we find a wj which is smaller than remaining Capacity, we put this object into it. But this time no need minus 1 objects. So here replace the j-1 with j, which let us can choose repeat object as before. So the equation will be this. Here I use a (C+1\*n+1) 2-D array to do dynamic programming.

Then I will show how we achieve this. And for the first row and first col, we initialize it to 0 since it means no Capacity or no objects. Then we use the bottom-up method starting from capacity 1 to C. Traverse the row (1 to n objects ) for all C. Using this equation. After traverse the whole 2-D array, we can find the max profit at profit[C][n].

And to find the time complexity, we can see here have two for loops so that it is O(nC), which proves that much faster than recursive method when n is large. I will pass the time to Zhao Yu.