Technical Behavior of the Algorithms :

1. Recursion :

From the graph, I was able to get the following, as the number of objects increased then the running time is also increased. When the time taken is converted into log to the base 2 scale, then we can observe a linear kind graph telling us that with increase in inputs, the time taken increases but not linearly as graph suggests but more than linear (since y axis is in log )

The slope which I got is less than one. Recursion depends on the inputs, so it is input dependent. We don’t get perfect linear graph since we cannot accurately guess the run time in recursion. One other observation we can make by looking at other two graphs and this is, we cannot go more than 100 – 150 number of objects but in dynamic programming we are able to calculate even for 4096 objects with decent amount of time. This is one of the limitation but recursion is the simple to write and does not require extra space as required by the cache and dynamic programming solutions

Cache Solution:

When we apply log with base 2 to both the x axis (Number of objects) and Y axis (Time taken), we get the above the graph. One of the observation that can be seen clearly is the difference in the time taken when we change it from wide to narrow, even when the capacity and max value of the items size/values remains same. We can say that if we use the narrow outputs (close to max value) it is taking more time to calculate the optimal solution. So cache solution is input dependent. When the number of objects are increased the time taken for narrow inputs is increasing much rapidly than for the wide range.

We take log values here, we can see that it takes (slope)n Worst case we can calculate by the mathematical equations as 2n, which is very close to what we got here. But it is crossing that for narrow input case, I think maybe this is because of other factors. Including the heap and my cpu load at the time.

Dynamic Programming:

When both the x axis (no of objects) and y axis (time taken) is converted into log scale of base 2. The slope of graph is close to 2. We can say that this can be given as worst case of n^2. One of the observations we can make is that there is no much difference between graphs of wide and narrow, we can guess this because in dynamic programming all the values in the 2d array is calculated, so it can take any challenge and do that at almost same time. This seems advantageous when we have complicated things to do and not so when doing simple things (that does not require calculating the whole array)

One thing that is common for all the methods is that time taken increases with the increase of number of objects. This not much of a shock since more number of objects = more calculations = more time.

In Dynamic Programming, as it calculates all the values without the concern of the input we can expect a constant slope (slope gives us the running time, so by slope we can correlate to running times) which we observed in the above graph.

Same running time is observed for both narrow and wide input. Since this is input independent. Worst case here is n2 and the follows through for all inputs