A Multidimensional DP

The dimensions of a DP algorithm refer to the number of state variables used to define each state. So far all the algorithms we have looked at required only one state variable - therefore they are one-dimensional. In this section, we're going to talk about problems that require multiple dimensions.

Typically, the more dimensions a DP problem has, the more difficult it is to solve. Two-dimensional problems are common, and sometimes a problem might even require five dimensions. The good news is, the framework works regardless of the number of dimensions.

The following are common things to look out for in DP problems that require a state variable:

- An index along with some input. This is usually used if an input is given as an array or string. This has been the sole state variable for all the problems that we've looked at so far, and it has represented the answer to the problem if the input was considered only up to that index - for example, if the input is

  $\text{nums} = [0, 1, 2, 3, 4, 5, 6]$, then $\text{dp}(4)$ would represent the answer to the problem for the input $\text{nums} = [0, 1, 2, 3, 4]$.

- A second index along with some input. Sometimes, you need two index state variables, say i and j. In some questions, these variables represent the answer to the original problem if you considered the input starting at index i and ending at index j. Using the same example above, $\text{dp}(1, 3)$ would solve the problem for the input $\text{nums} = [1, 2, 3]$, if the original input was $[0, 1, 2, 3, 4, 5, 6]$.

- Explicit numerical constraints given in the problem. For example, "you are only allowed to complete k transactions", or "you are allowed to break up to k obstacles", etc.
- Variables that describe statuses in a given state. For example "true if currently holding a key, false if not", "currently holding k packages" etc.
- Some sort of data like a tuple or bitmask used to indicate things being "visited" or "used". For example, "$bitmask$ is a mask where the i^{th} bit indicates if the i^{th} city has been visited". Note that mutable data structures like arrays cannot be used - typically, only immutable data structures like numbers and strings can be hashed, and therefore memoized.

Multi-dimensional problems make us think harder about deciding what our function or array will represent, as well as what the recurrence relation should look like. In the next article, we'll walk through another example using the framework with a 2D DP problem.