Dynamic Programming 3.1

- Problem on cycling dp states
- Tricks to identify DP problems
- Common states & transitions looking at the constraints
- DP with bitmasking
- 2 problems

- Priyansh Agarwal

Problem on Cycling DP States: Link

$N \le 300$, $a_i \le 3$

1	1	2

2	1	1

1	2	1

zeros

ones

twos

threes

$$\left.\begin{array}{lll} \text{zeros} & \longrightarrow & 5 \\ \text{ones} & \longrightarrow & 6 \\ \text{twos} & \longrightarrow & 7 \\ \text{threes} & \longrightarrow & 2 \end{array}\right\} 20$$

$$\begin{tikzcd}
 & (5,6,7,2) \arrow[dl, "6/20"'] \arrow[d, "7/20"] \arrow[ddr, bend left, "5/20"] & \\
(6,5,7,2) & (5,7,6,2) & \\
 & & (5,6,7,2)
\end{tikzcd}$$

$C_0 \quad C_1 \quad C_2 \quad C_3$ $\quad S$

$\text{total} = C_0 + C_1 + C_2 + C_3$

C_0/total

C_1/total

C_2/total

C_3/total

(C_0, C_1, C_2, C_3)

(C_0+1, C_1-1, C_2, C_3) $\quad S_1$

(C_0, C_1+1, C_2-1, C_3) $\quad S_2$

$((C_0, C_1, C_2+1, C_3-1))$ $\quad S_3$

$E[S] \longrightarrow (n, 0, 0, 0)$

$dp[c_0][c_1][c_2][c_3]$ = Expected no. of steps after which array becomes empty such

zeros $= c_0$

ones $= c_1$

twos $= c_2$

three $= c_3$

$$\frac{c_0}{total}\left[1 + dp[c_0][c_1][c_2][c_3]\right]$$

$$dp[c_0][c_1][c_2][c_3] \quad \frac{c_1}{total}\left[1 + dp[c_0+1][c_1-1][c_2][c_3]\right]$$

$c_2 / total \quad \left(1 + dp[c_0][c_1+1][c_2-1][c_3]\right)$

$c_3 / total \quad \left(1 + dp[c_0][c_1][c_2+1][c_3-1]\right)$

$$dp[c_0][c_1][c_2][c_3]$$

$$= 1 + \frac{c_0}{total}\left(dp[c_0)[c_1)(c_2)(c_3)\right) + \frac{c_1}{total}\Big($$

1 | 2 | 2 | 4 | 4

$$\frac{1}{5}(1) + \frac{2}{5}(2) + \frac{2}{5}(4)$$

$$(1 + 2 + 2 + 4 + 4)$$

Trick to identify a DP problem?

Repeating subtasks:

- If I have the answer of state, then why should I calculate it again and waste time

Pro Tips for contests:

- Number of ways problems -> DP, Brute Force or some formula
- Look for small constraints in the problem. (Most probably it would be dp and not greedy)
- Identify states and transition time for each state.
- Calculate time complexity as (number of states * transition time for each state).
- If this number fits into your Time limit (Great), if not, try to see if you can skip some states and still get the right answer.
- Try to reduce the transition time by using some Data Structure or some clever observation if transition time is the bottleneck
- Never try to over optimize. If your current states and transition time fit into your Time Limit, just code it and do not optimize it further.

Common states and transitions with constraints

```
total operations <= 1e8

n <= 125 :

    state: O(n^3), transition: O(1), [n <= 100 O(logn) is possible]
    state: O(n^2), transition: O(n), [n <= 100 O(nlogn) is possible]
    state: O(n), transition: O(n^2)

n <= 5000:

    state: O(n^2), transition: O(1) [n <= 1000 then O(logn) is possible]
    state: O(n), transition: O(n) [n <= 1000 then O(nlogn) is possible]

n <= 1e6:

    state: O(n), transition: O(1), O(logn)

1 second <= operations <= 4 * 1e8
4 second <= operations <= 1e9
```

DP with Bitmasking

- Bitmasks
- Basic operations on Bitmasks
- Limitations on “N”

Problem 1:

Given a list of points on a 2D plane, rearrange these points in any way such that in the final permutation of points, the sum of distances of the adjacent elements is minimized.

Constraints: [N <= 15], [-1e9 <= Xi, Yi <=1e9]

Points -> [{0, 0}], {5, 6}, {1, 2}]

Best permutation -> [{0, 0}, {1, 2}, {5, 6}]]

Ans = Dist(P1, P3) + Dist(P3, P2)

Problem 2: Link