Dynamic Programming
Day 5

- Priyansh Agarwal

Solving Homework Problems

Problem 1: Link

```
state:
dp[i][string in last 5 characters] = check if you can
fill question marks in string [i:n] such that there is
no palindrome of length 5 or more in the suffix [i:n]

transition:
dp[i][string in last 5 characters] =
dp[i + 1][string in last 4 characters + '1'] ||
dp[i + 1][string in last 4 characters + '0']

base case:
dp[n][any string so far] = true

final subprolem:
dp[0][""]
```

Problem 2: Link

```
state:
dp[i][j] = maximum score that can be obtained from array [
i...j] for the player whose turn it is

transition:
dp[i][j] = max(
        a[i] + sum[i + 1][j] - dp[i + 1][j],
        a[j] + sum[i][j - 1] - dp[i][j - 1])

base case:
    dp[i][i] = a[i]

final subproblem:
    dp[0][n - 1]
```

Problem 3: Link

```
if((n * (n + 1)) / 2 is odd)
    ans = 0

state:
ways[value][i] = number of ways to create a sum of value from the
first i elements

transition:
ways[value][i] = ways[value][i - 1] + ways[value - arr[i]][i - 1]

base case:
ways[0][0] = 1, ways[0][anything] = 0

final subproblem:
ways[n * (n + 1) / 4][n]
```

Cycling DP states

- What happens when your current state is dependent on itself?
    - dp[i] depending on dp[i] itself

Problem 1

- Given a positive integer $N <= 1e6$, at every step the following 3 things can happen to N with equal probability.
  - $N = N / 2$
  - $N = N - 1$
  - N remains unchanged
- Find expected number of steps it will take to convert for N to become 0

Problem 1

```
state:
dp[n] = expected number of steps to convert n to 0


transition:

dp[n] = 1/3 (1 + dp[n - 1] + 1 + dp[n / 2] + 1 + dp[n])

dp[n] = 1/3 (3 + dp[n - 1] + dp[n / 2] + dp[n])

dp[n] = 1 + dp[n - 1] / 3 + dp[n / 2] / 3 + dp[n] / 3

2/3 * dp[n] = 1 + dp[n - 1] / 3 + dp[n / 2] / 3

dp[n] = 3 / 2 + (dp[n - 1] + dp[n / 2]) / 2

base case:
dp[0] = 0

final subproblem = dp[n]
```

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
```