A – Decode III

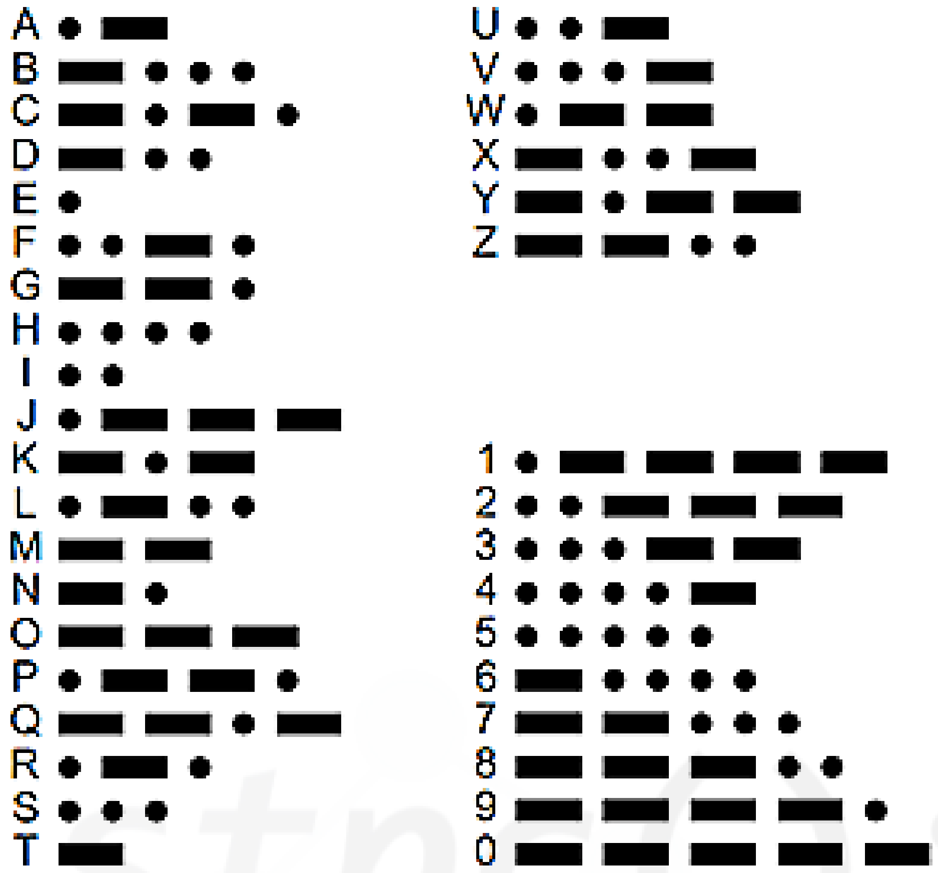
Chin Ka Wang {rina__owo}

2025-05-16

Background

Problem Idea by `rina__owo`

Preparation by `rina__owo`, `pepper1208`



Problem Restatement

Given a string of morse code containing characters and numbers, without splitting of spaces.

Find the total number of possible decoded messages by inserting any number of spaces in the morse code.

Subtasks

- Attempts: 5
- Max: 100
- First solved by **Chan Yu Ham** at **1h 30m 23s**

Subtasks Constraints

	Score	Constraints
1	7	$N \le 20$ s_i is dot for $1 \le i \le N$
2	16	$2 \le N \le 20$ s_i is dot for $1 \le i \le N-2$ s_i is dash for $N-1 \le i \le N$
3	20	$N \le 20$
4	23	$N \le 3000$
5	34	No additional constraints

Subtask 1 (7%): N ≤ 20, all dots

- Note that there exist, “.”, “..”, “...”, “....”, “.....” in the possible codes.
- We can use depth-first search (DFS) to brute force for the answer.
    - For example, define `DFS(x)` where `x` is the remaining length of the code which is not processed yet. To start, we call `DFS(N)`.
    - While searching, iterating `i` from 1 to 5 with `x-i > 0`, search for `DFS(x-i)`, where code with length `i` is partitioned.
    - Base case: For `x = 0`, the search ends as the code has been fully partitioned. We can increase our counter which stores the number of ways to partition the code.
- Expected score: **7** (Culminative Score: **7**)
- Time complexity: ~ $O(1.534^N)$

Subtask 3 (20%): N ≤ 20

- The idea is basically same as subtask 1.
- However, the partitions we made are not guaranteed to be valid.
- We should enter all the morse code into a container such as a map. Then, we can use C++ STL function (e.g. substr()) to extract the actual partition as a string and validate the partition.
- As there are only 36 types of valid partition, the validation will take only O(1) time complexity.
- Expected score: **20** (Culminative Score: **27**)
- Time complexity: ~ $O(1.534^N)$

Subtask 2 (16%): N ≤ 20, dots followed by two dashes

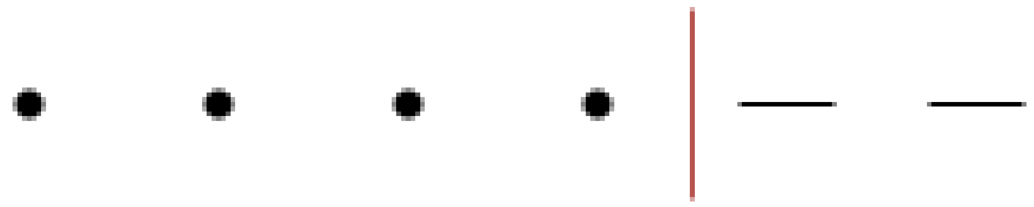
- We can actually use the same algorithm in subtask 3 to achieve subtask 2.
- Easy subtask! ☺
- Expected score: **16** (Culminative Score: **43**)
- Time complexity: ~ $O(1.534^N)$

- However, take a closer look at this subtask.

Subtask 2 (16%): N ≤ 20, dots followed by two dashes

- We can actually get some inspiration for the full solution!
- We know how to deal with all dots, but we don’t know how to deal with dashes yet.
- Why don’t we consider them separately?
- Note that the last two dashes represent a letter “M”. If we consider the last two dashes as a single partition, we can actually use the idea in subtask 1 to find out the way to partition all **previous** dots, which is already **known**.

Subtask 2 (16%): N ≤ 20, dots followed by two dashes



Subtask 2 (16%): N ≤ 20, dots followed by two dashes

- How about we partition the two dashes with a single previous dot?
- If we consider the last two dashes with one previous dot as a single partition, we can actually use the idea in subtask 1 again to find out the way to partition all **previous** dots, which is already **known**.
- The logic can be applied to partition the two dashes with more previous dots.

Subtask 2 (16%): N ≤ 20, dots followed by two dashes

- How about we partition the dashes into two parts?
- If we consider the last dash as a single partition, we can now consider all dots with one dash at the end. Note that we can still handle this to find out the number of ways to partition the **previous** code with a similar fashion, and the method is implicitly **known**.

Subtask 2 (16%): N ≤ 20, dots followed by two dashes

- With these cases handled properly, we can achieve this subtask.
- Expected score: **16** (Culminative Score: **43**)
- Time complexity: ~ $O(1.534^N)$

Subtask 4 (23%): N ≤ 3000

- With the inspiration in subtask 2, we can generalize the idea to achieve subtask 4!
- Notice how we emphasize two key terms: **previous** and **known**.
- What approach they lead us to try? ***Dynamic Programming***!
- We can consider the last 1, 2, 3, …, $N - 1$ characters as a single partition. Then, the only thing we want to find out is the number of ways to partition all **previous** characters.

stpc();

Subtask 4 (23%): N ≤ 3000

- Define the state $dp[i]$ as the number of ways to partition all previous *i* characters. Denote s be the code as a 1-based string.
- Therefore, with $dp[0] = 1$, we can deduce the following transition formula:

$$dp[i] = \sum_{k=0}^{i-1} \left(dp[k] \times \left[s[k+1 \,..\, i] \text{ is a valid partition} \right] \right)$$

- Note: The bracket wrapping s[k+1..i] is a valid partition is called the **Iverson bracket**, which returns 1 if the condition inside the bracket is true, 0 otherwise.
- Expected score: **23** (Culminative Score: **66**)
- Time complexity: O(N^2)

Subtask 5 (34%): No additional constraints

- You may notice some **Time Limit Exceeded** in the judging result.
- Therefore, we need to optimize the transitional formula.
- Notice that the maximum length of the morse code of a single character is **5**. Therefore, it is unnecessary for us to check the partition with the length larger than 5.

Subtask 5 (34%): No additional constraints

- The transitional formula can be optimized as:

$$dp[i] = \sum_{k=i-5}^{i-1} \left(dp[k] \times \left[s[k+1 \text{ .. } i] \text{ is a valid partition} \right] \right)$$

- Expected score: **34** (Culminative Score: **100 AC!**)
- Time complexity: O(N)

Takeaways

- Don’t be afraid to type the whole morse code table. Trust yourself.
- Don’t be obsessed with finding the transitional formula right away. Try to brute force first and carry out more observation when doing dynamic programming problems.
- Be aware of some specific terms like “find number of ways”, “find the minimum”, “find the maximum”, etc. The problems are often related to dynamic programming (or greedy). You should be familiar with identifying the right approach you should use.
- When facing TLE or MLE in a dynamic programming problems, you should know different approaches to optimize the transitional formula, the order of iteration, or the definition of the state.