Pragmatic Programming

Session 8 - Dynamic Programming Revisited

Max Nilsson

January 18th

The Beauty of Dynamic Programming in 4 Parts

1. The Longest Common Subsequence

- Given two strings, `s` and `t`, the task is to find the length of the longest common subsequence, i.e. longest subsequence present in both of the strings, which may be not contiguous.

1. The Longest Common Subsequence

- Given two strings, `s` and `t`, the task is to find the length of the longest common subsequence, i.e. longest subsequence present in both of the strings, which may be not contiguous.
- For example `s = "ABCDEF"`, `t = "ACEDF"` has longest common subsequence `"ACDF"`.

1. The Longest Common Subsequence

- Given two strings, `s` and `t`, the task is to find the length of the longest common subsequence, i.e. longest subsequence present in both of the strings, which may be not contiguous.
- For example `s = "ABCDEF"`, `t = "ACEDF"` has longest common subsequence `"ACDF"`.
- But what about `s =` `"hqeerteqwtleqryultirtioryo_ytoyhoryeuretery_wetqperwraeqegyrmaeryuettitiryucro_yppuepyouppiliuireqwe"` and `t =` `"haesgldalhdadojs_fjtashadeadjrhdfejdgh_kjhpk hgrlagxcvbxmjahgjtjihcfhcch_jgpvjhekhlohpcvhclhjevkadxcv"`?

2. Optimal Strategy of a Coin Game

- Given a row of coins, we play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.

2. Optimal Strategy of a Coin Game

- Given a row of coins, we play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.
- For example, if `coins = {1, 2, 3 }` then the answer is 4.

2. Optimal Strategy of a Coin Game

- Given a row of coins, we play a game against an opponent by alternating turns. In each turn, a player selects either the first or last coin from the row, removes it from the row permanently, and receives the value of the coin. Determine the maximum possible amount of money we can definitely win if we move first.
- For example, if `coins = {1, 2, 3 }` then the answer is 4.
- A greedy strategy will not work:
  If `coins = {20, 30, 2, 2, 2, 10}` then the greedy strategy will be worse than the optimal 42.

3. Minimum Insertions to form a Palindrome

- Given a string s, the task is to find the minimum number of characters to be inserted to convert it to a palindrome.

3. Minimum Insertions to form a Palindrome

- Given a string `s`, the task is to find the minimum number of characters to be inserted to convert it to a palindrome.
- For example, if `s = "abcde"` then the answer is 4 and the palindrome is `"edcbabcde"`.

3. Minimum Insertions to form a Palindrome

- Given a string `s`, the task is to find the minimum number of characters to be inserted to convert it to a palindrome.
- For example, if `s = "abcde"` then the answer is 4 and the palindrome is `"edcbabcde"`.
- There is a quadratic solution using left and right pointers such as in **2.**

3. Minimum Insertions to form a Palindrome

- Given a string `s`, the task is to find the minimum number of characters to be inserted to convert it to a palindrome.
- For example, if `s = "abcde"` then the answer is 4 and the palindrome is `"edcbabcde"`.
- There is a quadratic solution using left and right pointers such as in **2.**
- It is also a special case of **1.** (!)

4. Speedrunning Super Mario Bros.

- See Speedrunning.

4. Speedrunning Super Mario Bros.

- See Speedrunning.
- This problem pretty easily be solved without dynamic programming, but in my opinion that solution is less elegant and harder to implement.

This Week

- Solve half of this weeks 10 problems.