Dynamic Programming

•••

Week 6 AI Inspire Fall 2019

Introduction

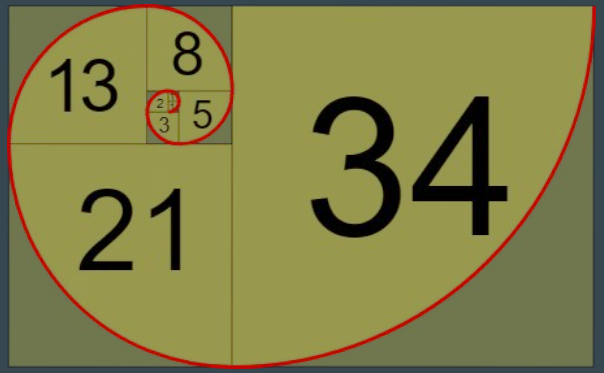
What is Dynamic Programming?

- Mathematical optimization
- Solve problems by splitting into subproblems
- Uses recursion
- Not too many subproblems and subproblems can overlap
- **Very efficient solution - Whenever we repeat a subproblem ⇒ just return the stored value**
  - **Memoization is key component of DP**

Problem 1 - Fibonacci

Fibonacci Example - Without and without Dynamic Programming

1. Runtime and space complexity of algo without dynamic programming?
2. Runtime and space complexity of algo with dynamic programming and memoization?



Solution (with and without memoization)

Pure Recursion vs Memoization -

```
int fib (int n) {
      if (n < 2)
          return 1;
      return fib(n-1) + fib(n-2);
  }
```

```
void fib () {
        fibresult[0] = 1;
        fibresult[1] = 1;
        for (int i = 2; i<n; i++)
            fibresult[i] = fibresult[i-1] + fibresult[i-2];
    }
```

Credits - https://www.hackerearth.com/practice/algorithms/dynamic-programming/introduction-to-dynamic-programming-1/tutorial/

Suggested Homework

Create DP Solution of Count the Paths problem and compare runtime complexities.

https://www.youtube.com/watch?v=P8Xa2BitN3I

Problem 2 - Knapsack

Problem Details

- Thief wants to steal items
- Items each have some sort of weight and values
- Thief has some restraints
  - Can’t hold more than max allowed weight
  - Wants to maximize value

https://www.youtube.com/watch?v=xOlhR_2QCXY

Problem 3- Longest Common Subsequence

Problem Details

- Length of longest common subsequence
- https://www.youtube.com/watch?v=e0CAbRVYAWg

Suggested Homework

- Think of solution to “Count the paths problems”
- Think of bottom up solution to longest common subsequence