<https://github.com/TT00FE39-3001/lecture4>

<https://github.com/seppotk/Datastructures_and_algorithms.git>

**Outline**

**Topics**

* Review
* Recursion
* [Dynamic Programming](https://www.geeksforgeeks.org/dynamic-programming/):
  + Memoization
  + Tabulation

**This Week in Points**

* Group Activities (Max 9 points)
* Homework (Max 9 points)
* Peer reviews (Max 7 points)

**Part 1: Recursion**

* The Top-Down Thought Process
* [Recursion In C++](https://www.softwaretestinghelp.com/recursion-in-cpp/)
* Fibonacci:
  + Visualization: [Link 1](https://www.cs.usfca.edu/~galles/visualization/DPFib.html), [Link 2](https://www.educative.io/courses/grokking-dynamic-programming-a-deep-dive-using-cpp/m2JgzWPw9RR)
  + [Fibonacci number](https://en.wikipedia.org/wiki/Fibonacci_number) & [Fibonacci sequence](https://www.mathsisfun.com/numbers/fibonacci-sequence.html)
* [The Staircase Problem](https://www.geeksforgeeks.org/count-ways-reach-nth-stair/)
* [Activity 1](https://github.com/TT00FE39-3001/lecture5/blob/main/activity1)

**Part 2: Dynamic Programming (Memoization)**

* [Overlapping Sub-problems](https://www.geeksforgeeks.org/overlapping-subproblems-property-in-dynamic-programming-dp-1/)
* Fibonacci Revisited
  + Visualization: [Link 1](https://www.cs.usfca.edu/~galles/visualization/DPFib.html), [Link 2](https://www.educative.io/courses/grokking-dynamic-programming-a-deep-dive-using-cpp/m2JgzWPw9RR)
  + [Fibonacci Revisited](https://www.geeksforgeeks.org/introduction-to-dynamic-programming-data-structures-and-algorithm-tutorials/)
* [The Staircase Problem](https://www.geeksforgeeks.org/count-ways-reach-nth-stair/)
* [The Knapsack Problem](https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/)
* [What is memoization?](https://www.geeksforgeeks.org/what-is-memoization-a-complete-tutorial/)
* [Activity 2](https://github.com/TT00FE39-3001/lecture5/blob/main/activity2)

**Part 3: Dynamic Programming (Tabulation)**

* [Tabulation vs Memoization](https://www.geeksforgeeks.org/tabulation-vs-memoization/)
* Fibonacci Revisited
  + Visualization: [Link 1](https://www.cs.usfca.edu/~galles/visualization/DPFib.html), [Link 2](https://www.educative.io/courses/grokking-dynamic-programming-a-deep-dive-using-cpp/m2JgzWPw9RR)
* [The Staircase Problem](https://www.geeksforgeeks.org/count-ways-reach-nth-stair/)
* [The Knapsack Problem](https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/)
* [Dynamic Programming in the Real-world](https://www.educative.io/courses/grokking-dynamic-programming-a-deep-dive-using-cpp/m2JgzWPw9RR#Real-world-problems)
* [Activity 3](https://github.com/TT00FE39-3001/lecture5/blob/main/activity3)

**Misc**

* [Notes](https://github.com/TT00FE39-3001/lecture5/blob/main/notes.md)

**# Outline**

**## Topics**

- Review

- Recursion

- [Dynamic Programming](https://www.geeksforgeeks.org/dynamic-programming/):

  - Memoization

  - Tabulation

**## This Week in Points**

- Group Activities (Max 9 points)

- Homework (Max 9 points)

- Peer reviews (Max 7 points)

**## Part 1: Recursion**

- The Top-Down Thought Process

- [Recursion In C++](https://www.softwaretestinghelp.com/recursion-in-cpp/)

- Fibonacci:

  - Visualization: [Link 1](https://www.cs.usfca.edu/~galles/visualization/DPFib.html), [Link 2](https://www.educative.io/courses/grokking-dynamic-programming-a-deep-dive-using-cpp/m2JgzWPw9RR)

  - [Fibonacci number](https://en.wikipedia.org/wiki/Fibonacci\_number) & [Fibonacci sequence](https://www.mathsisfun.com/numbers/fibonacci-sequence.html)

- [The Staircase Problem](https://www.geeksforgeeks.org/count-ways-reach-nth-stair/)

- [Activity 1](./activity1)

**## Part 2: Dynamic Programming (Memoization)**

- [Overlapping Sub-problems](https://www.geeksforgeeks.org/overlapping-subproblems-property-in-dynamic-programming-dp-1/)

- Fibonacci Revisited

  - Visualization: [Link 1](https://www.cs.usfca.edu/~galles/visualization/DPFib.html), [Link 2](https://www.educative.io/courses/grokking-dynamic-programming-a-deep-dive-using-cpp/m2JgzWPw9RR)

  - [Fibonacci Revisited](https://www.geeksforgeeks.org/introduction-to-dynamic-programming-data-structures-and-algorithm-tutorials/)

- [The Staircase Problem](https://www.geeksforgeeks.org/count-ways-reach-nth-stair/)

- [The Knapsack Problem](https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/)

- [What is memoization?](https://www.geeksforgeeks.org/what-is-memoization-a-complete-tutorial/)

- [Activity 2](./activity2)

**## Part 3: Dynamic Programming (Tabulation)**

- [Tabulation vs Memoization](https://www.geeksforgeeks.org/tabulation-vs-memoization/)

- Fibonacci Revisited

  - Visualization: [Link 1](https://www.cs.usfca.edu/~galles/visualization/DPFib.html), [Link 2](https://www.educative.io/courses/grokking-dynamic-programming-a-deep-dive-using-cpp/m2JgzWPw9RR)

- [The Staircase Problem](https://www.geeksforgeeks.org/count-ways-reach-nth-stair/)

- [The Knapsack Problem](https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/)

- [Dynamic Programming in the Real-world](https://www.educative.io/courses/grokking-dynamic-programming-a-deep-dive-using-cpp/m2JgzWPw9RR#Real-world-problems)

- [Activity 3](./activity3)

**## Misc**

- [Notes](./notes.md)

Notes

**# Notes**

**## Recursion: The Top-Down Thought Process**

When tackling a top-down problem, it helps to think the following three thoughts:

1. Imagine the function you’re writing has already been implemented by someone else.

2. Identify the subproblem of the problem.

3. See what happens when you call the function on the subproblem and go from there.

**## Dynamic programming**

Dynamic programming is an algorithm design technique with a rather interesting history. It was invented by a prominent U.S. mathematician, Richard Bellman, in the 1950s as a general method for optimizing multistage decision processes. Thus, the word “programming” in the name of this technique stands for **\*\*planning\*\*** and does not refer to computer programming. After proving its worth

as an important tool of applied mathematics, dynamic programming has eventually come to be considered, at least in computer science circles, as a general

algorithm design technique that does not have to be limited to special types of optimization problems. It is from this point of view that we will consider this technique here.

Dynamic programming is a technique for solving problems with overlapping subproblems. Typically, these subproblems arise from a recurrence relating a given problem’s solution to solutions of its smaller subproblems. Rather than solving

overlapping subproblems again and again, dynamic programming suggests solving each of the smaller subproblems only once and recording the results in a table from which a solution to the original problem can then be obtained.

Links

**# Links**

- [Data Structures and Algorithms Interview Course](https://www.enjoyalgorithms.com/data-structures-and-algorithms-course/)

- [Dynamic Programming](https://opendsa-server.cs.vt.edu/OpenDSA/Books/Everything/html/DynamicProgramming.html)

- [Visualization](https://www.cs.usfca.edu/~galles/visualization/DPFib.html)

- [Dynamic Programming: lecture notes](https://courses.csail.mit.edu/6.006/fall09/lecture\_notes/lecture18.pdf)

- [Recommended Playlist](https://www.youtube.com/playlist?list=PLDN4rrl48XKpZkf03iYFl-O29szjTrs\_O)

- https://cpp.sh/

HOMEWORK

**# Homework**

**## Task 1/3:Videos**

- [What Is Dynamic Programming and How To Use It](https://youtu.be/vYquumk4nWw)

- [0/1 Knapsack Problem](https://youtu.be/nLmhmB6NzcM)

**## Task 2/3: Reading**

- [The Staircase Problem](https://www.geeksforgeeks.org/count-ways-reach-nth-stair/)

- [0/1 Knapsack Problem](https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/)

- [Dynamic Programming](https://www.geeksforgeeks.org/dynamic-programming/)

- [Recursion](https://opendsa-server.cs.vt.edu/OpenDSA/Books/Everything/html/RecIntro.html)

**## Task 3/3: Pre-Lecture (Videos)**

- [Trees and heaps](https://youtube.com/watch?v=lhTCSGRAlXI&si=EnSIkaIECMiOmarE)

- [Heaps 1](https://youtube.com/watch?v=BzQGPA\_v-vc&si=EnSIkaIECMiOmarE)