Applied Data Science and Machine Learning Course

Course Instructor

Mohammad Sabik Irbaz
Lead Machine Learning Engineer
Pioneer Alpha Ltd.
sabik@pioneeralpha.com

Powered by Pioneer Alpha Ltd.

Lecture – 01

Review on Basic Programming Techniques

TABLE OF CONTENTS

TABLE OF CONTENTS

Introduction

Basic Programming Skills combine two things:
i) design of algorithms
ii) implementation of algorithms

The design of algorithm requires

(i) problem solving skills
(ii) mathematical thinking

The implementation of algorithm requires

(i) good programming skills

NB: It is not enough that the idea of algorithm is correct but the implementation also has to be correct.

TABLE OF CONTENTS

Time Complexity

Time complexity of an algorithm estimates how much time the designed algorithm will use for any input.

Time Complexity

Time complexity of an algorithm estimates how much time the designed algorithm will use for any input.

=> Time complexity of an algorithm is denoted as **O(f(n))**.
- f(n) represents some function
- ‘n’ denotes the input size

Time Complexity

Time complexity of an algorithm estimates how much time the designed algorithm will use for any input.

=> Time complexity of an algorithm is denoted as **O(f(n))**.
- f(n) represents some function
- ‘n’ denotes the input size

```
for(int i=1;i<=n;i++){
    //code
}
```

Time Complexity

Time complexity of an algorithm estimates how much time the designed algorithm will use for any input.

=> Time complexity of an algorithm is denoted as **O(f(n))**.
- f(n) represents some function
- ‘n’ denotes the input size

```
for(int i=1;i<=n;i++){
    //code
}
```

→ O(n)

```
for(int i=1;i<=n;i++){
    for(int j=1;j<=n;j++){
        //code
}
}
```

→ O(n^2)

TABLE OF CONTENTS

Sorting

Given an array that contains n elements, your task is to sort the elements in ascending or descending order.

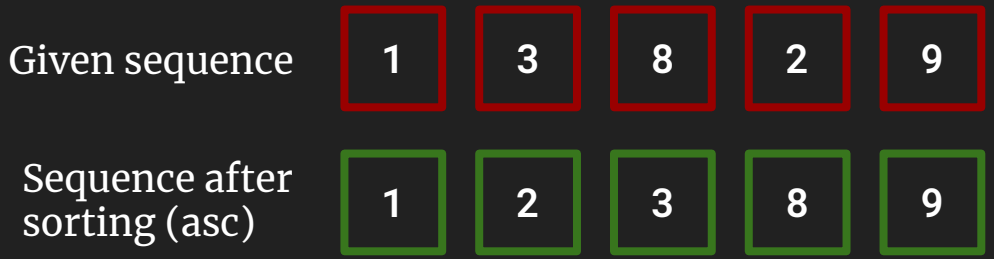
Sorting

Given an array that contains n elements, your task is to sort the elements in ascending or descending order.

Given sequence

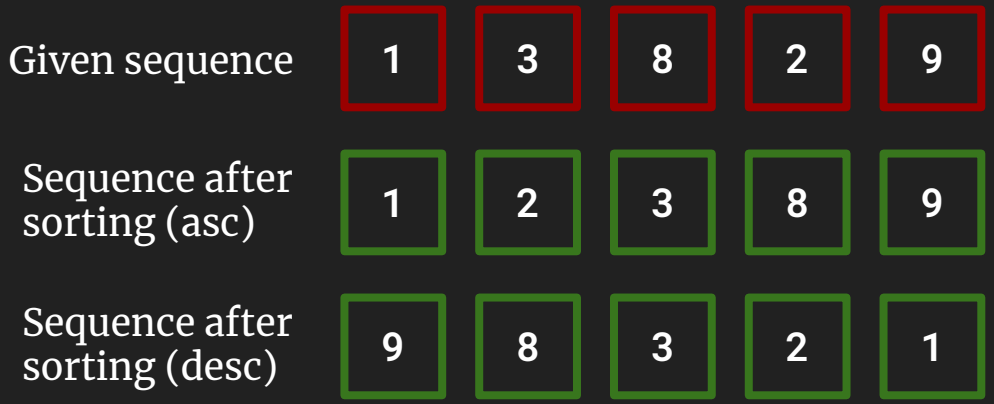
Sorting

Given an array that contains n elements, your task is to sort the elements in ascending or descending order.



Sorting

Given an array that contains n elements, your task is to sort the elements in ascending or descending order.



Types of Sorting

Sorting is considered one of the most important algorithms. Researchers and practitioners have been trying to optimize it as more as they can.

Types of Sorting

Sorting is considered one of the most important algorithms. Researchers and practitioners have been trying to optimize it as more as they can.

Different Types of Sorting:

i) Bubble Sort >>> O(n^2)

ii) Insertion Sort >>> O(n^2)

iii) Merge Sort >>> O(nlogn)

iv) Bucket Sort >>> O(max element in the array)

TABLE OF CONTENTS

Greedy Algorithms

A greedy algorithm constructs a solution to the problem by making a choice that looks best at that moment.

Greedy Algorithms

A greedy algorithm constructs a solution to the problem by making a choice that looks best at that moment.

Some of the common toy problems that are solved or attempted to be solved by greedy algorithms are:

i) Coin Problem

ii) Scheduling

iii) Tasks and deadlines

iv) Minimizing Sums

v) Data Compression

Scheduling

A classic problem:
Given n events with their starting and ending times, find a schedule that includes as many events as possible.

Scheduling

A classic problem:
Given n events with their starting and ending times, find a schedule that includes as many events as possible.

Event	Starting Time	Ending Time
A	1	3
B	2	5
C	3	9
D	6	8

Scheduling(cont.)

Event	Starting Time	Ending Time
A	1	3
B	2	5
C	3	9
D	6	8

Scheduling(cont.)

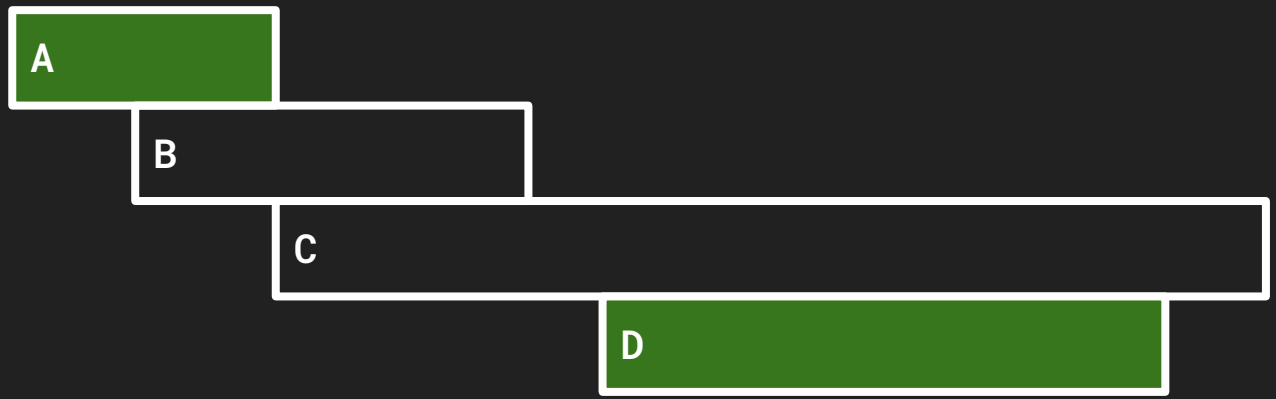
Algorithm - 1

Select as short events as possible

Scheduling(cont.)

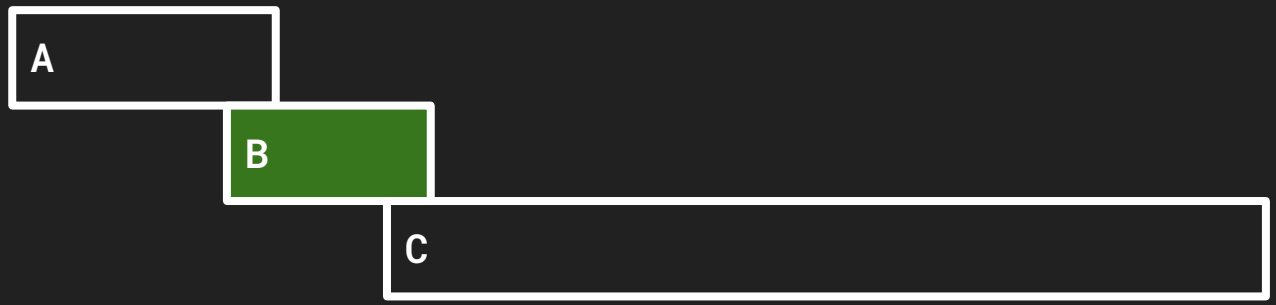
Algorithm - 1

Select as short events as possible



Scheduling(cont.)

Counter Example



Scheduling(cont.)

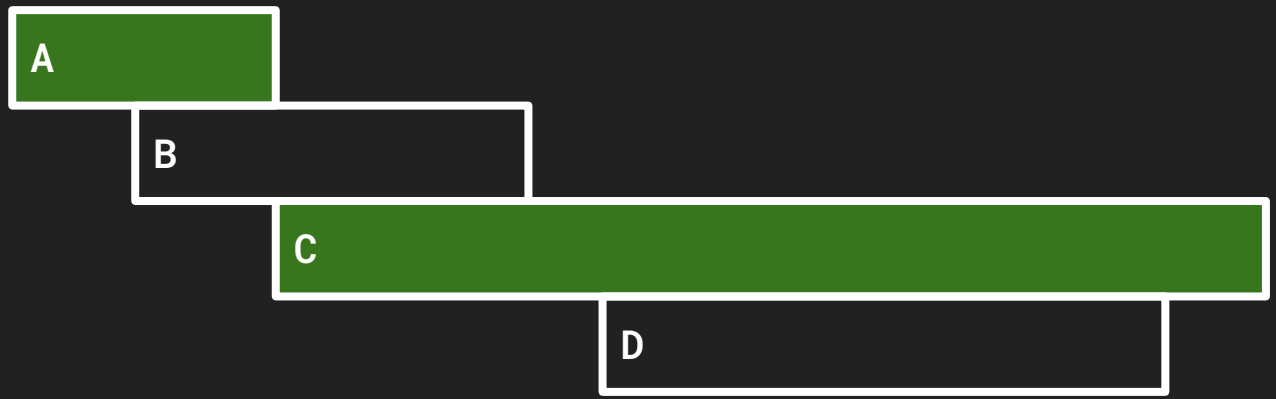
Algorithm - 2

Select the next possible event that begins as early as possible.

Scheduling(cont.)

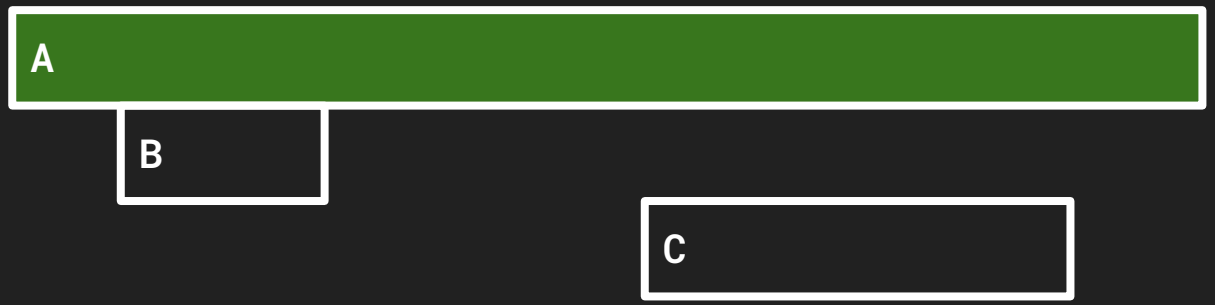
Algorithm - 2

Select the next possible event that begins as early as possible.



Scheduling(cont.)

Counter Example



Scheduling(cont.)

Algorithm - 3

Select the next possible event that ends as early as possible.

Scheduling(cont.)

Algorithm - 3

Select the next possible event that ends as early as possible.

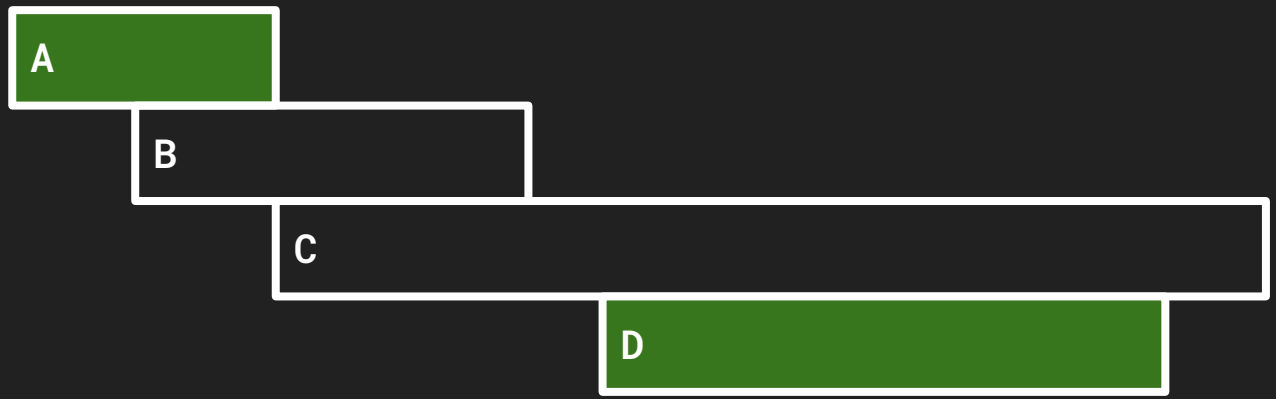


TABLE OF CONTENTS

Recursion

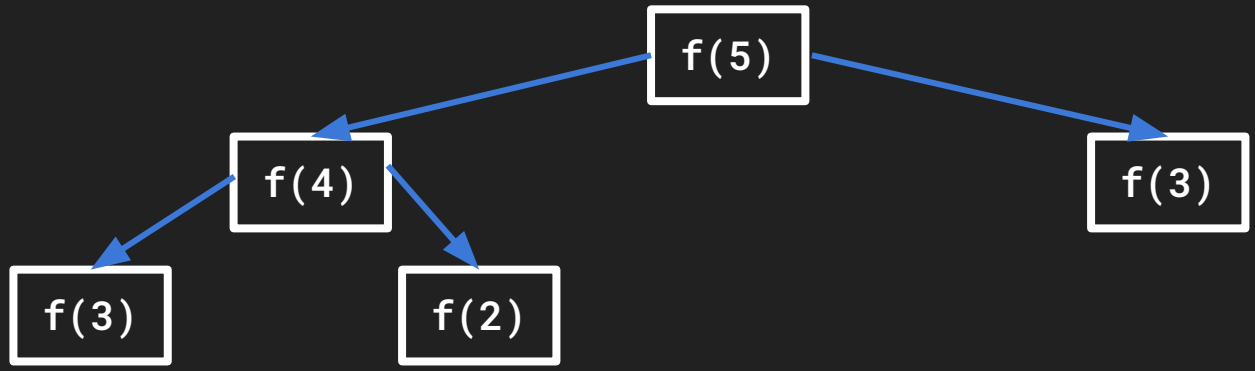
Recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem.

Fibonacci Sequence f(n) = f(n-1) + f(n-2)

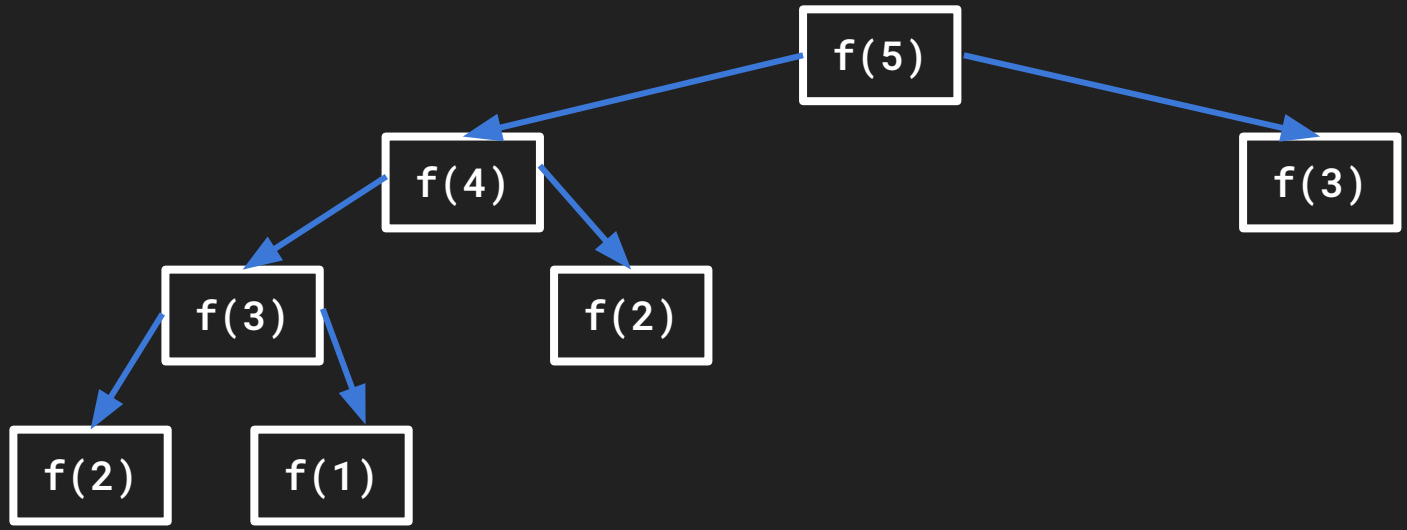
```
int fibonacci(int n){

    if (n<=1) return n

    return fibonacci(n-1) + fibonacci(n-2)
}
```

Recursion(cont.)

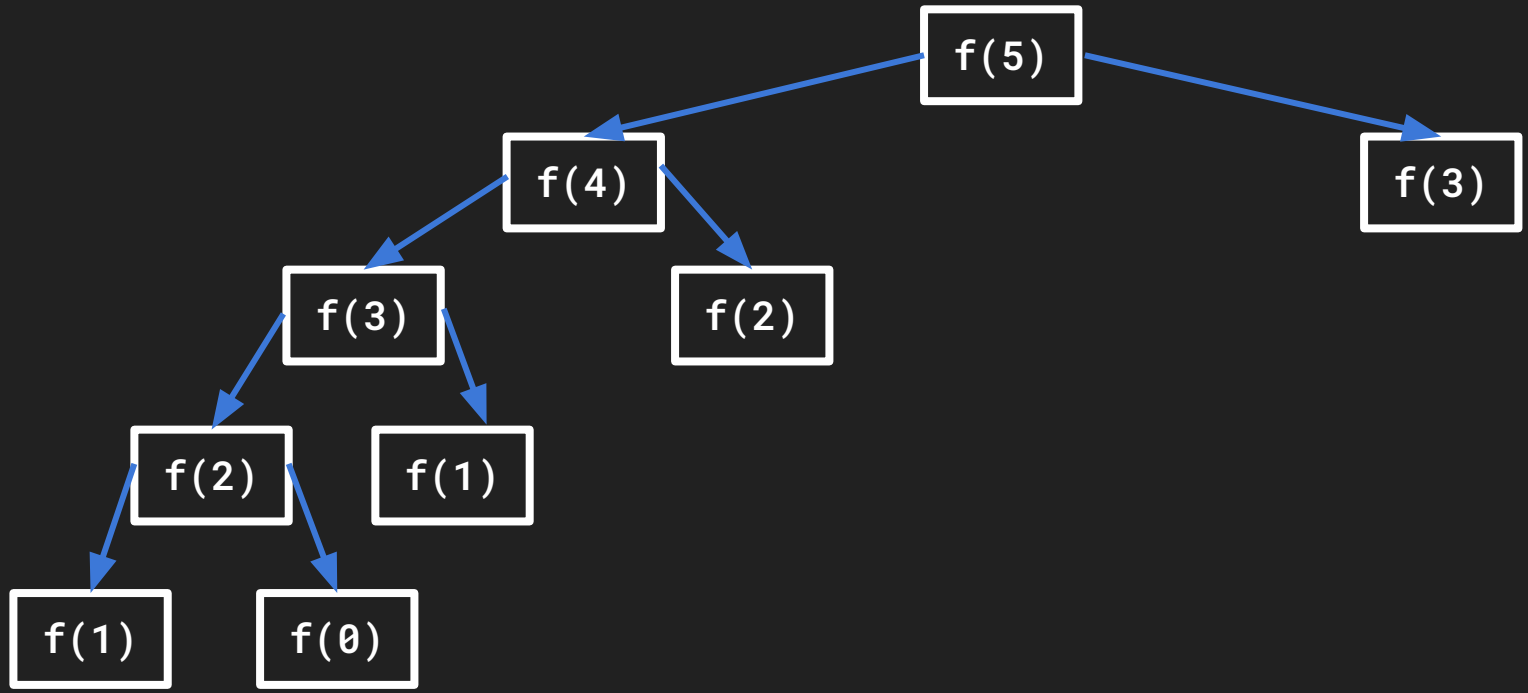
Recursion(cont.)



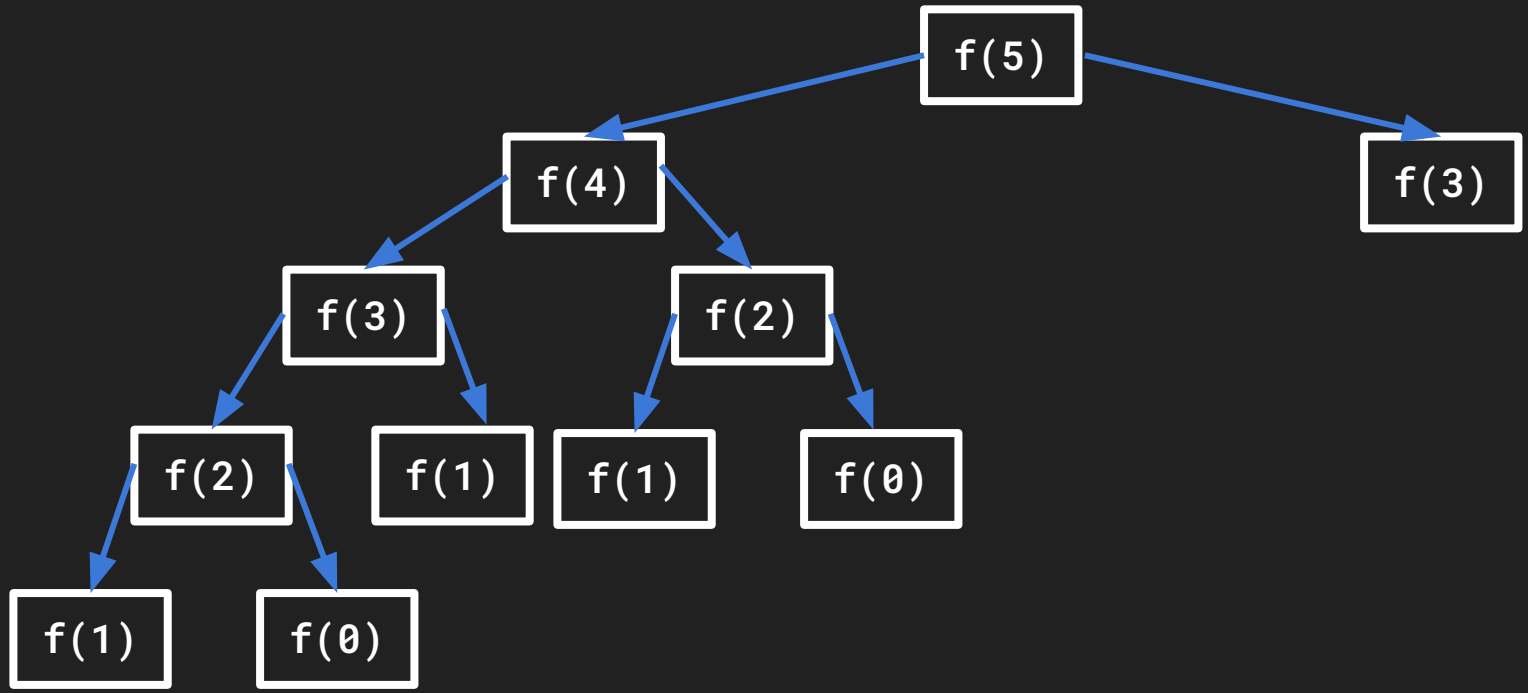
Recursion(cont.)



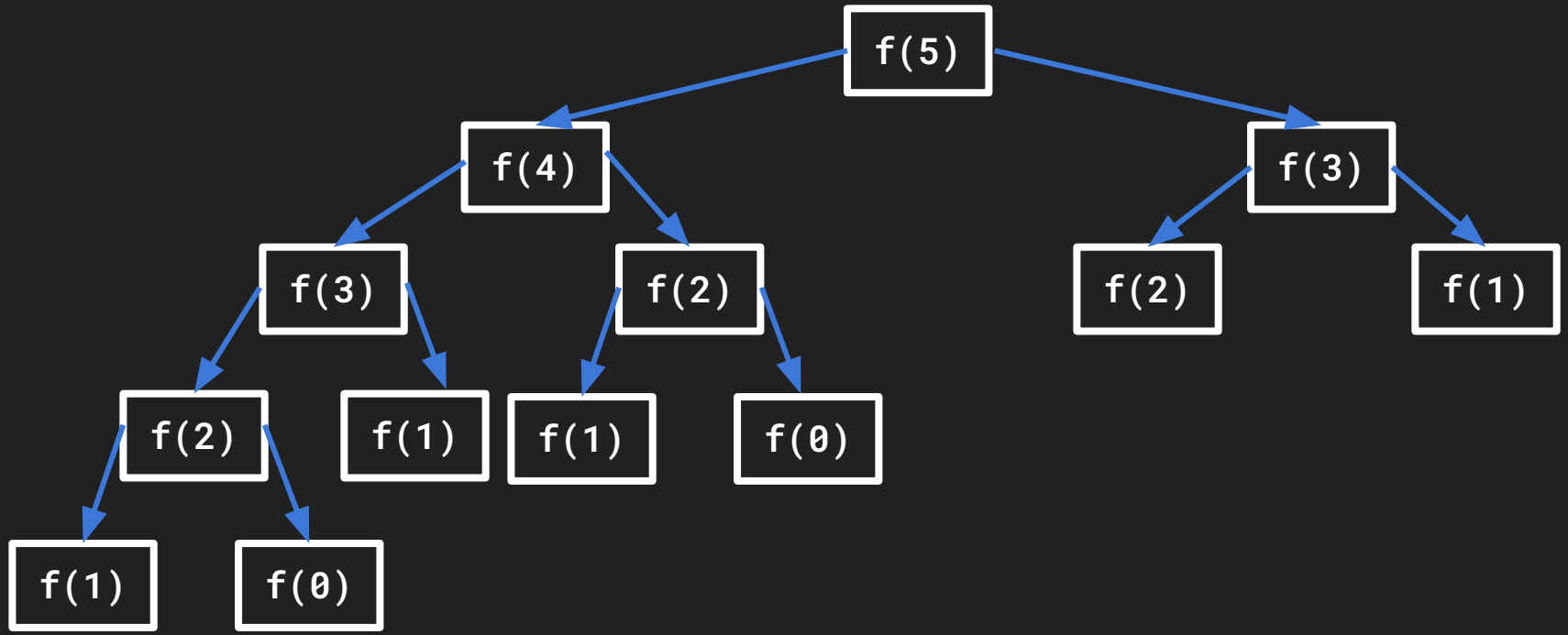
Recursion(cont.)



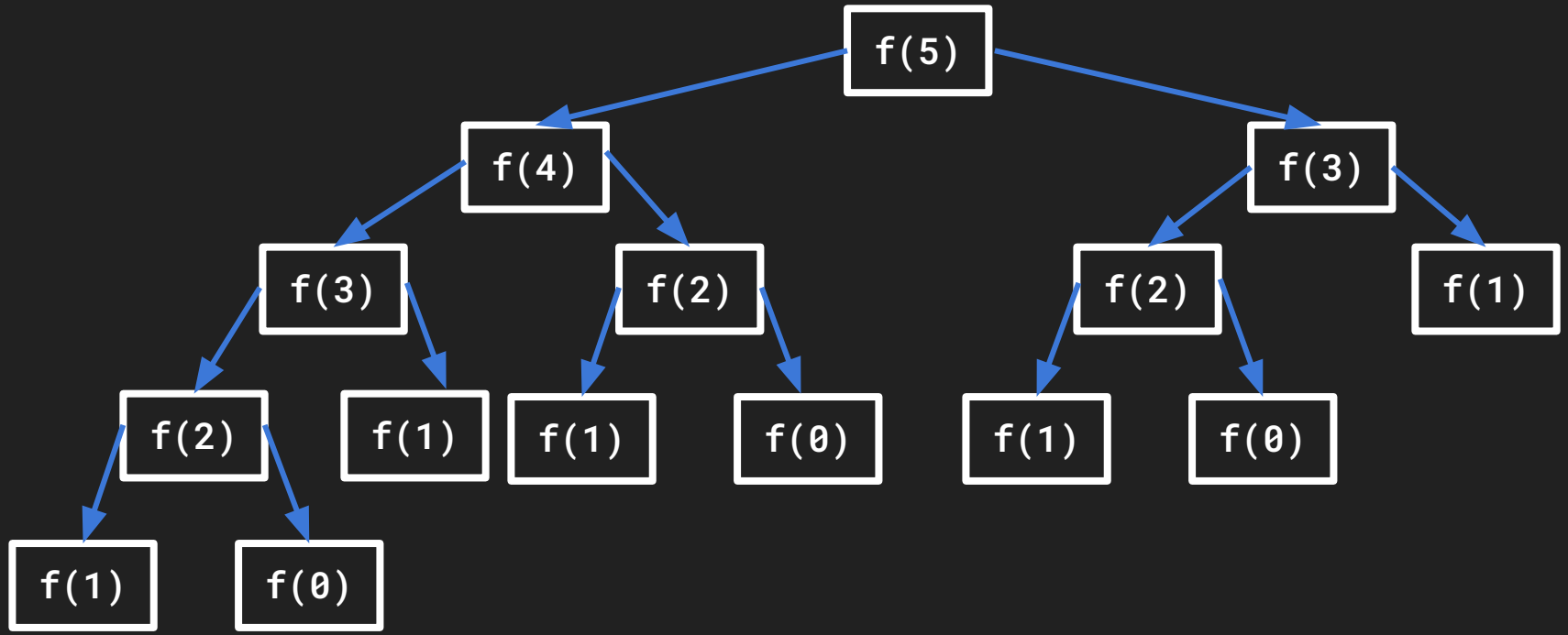
Recursion(cont.)



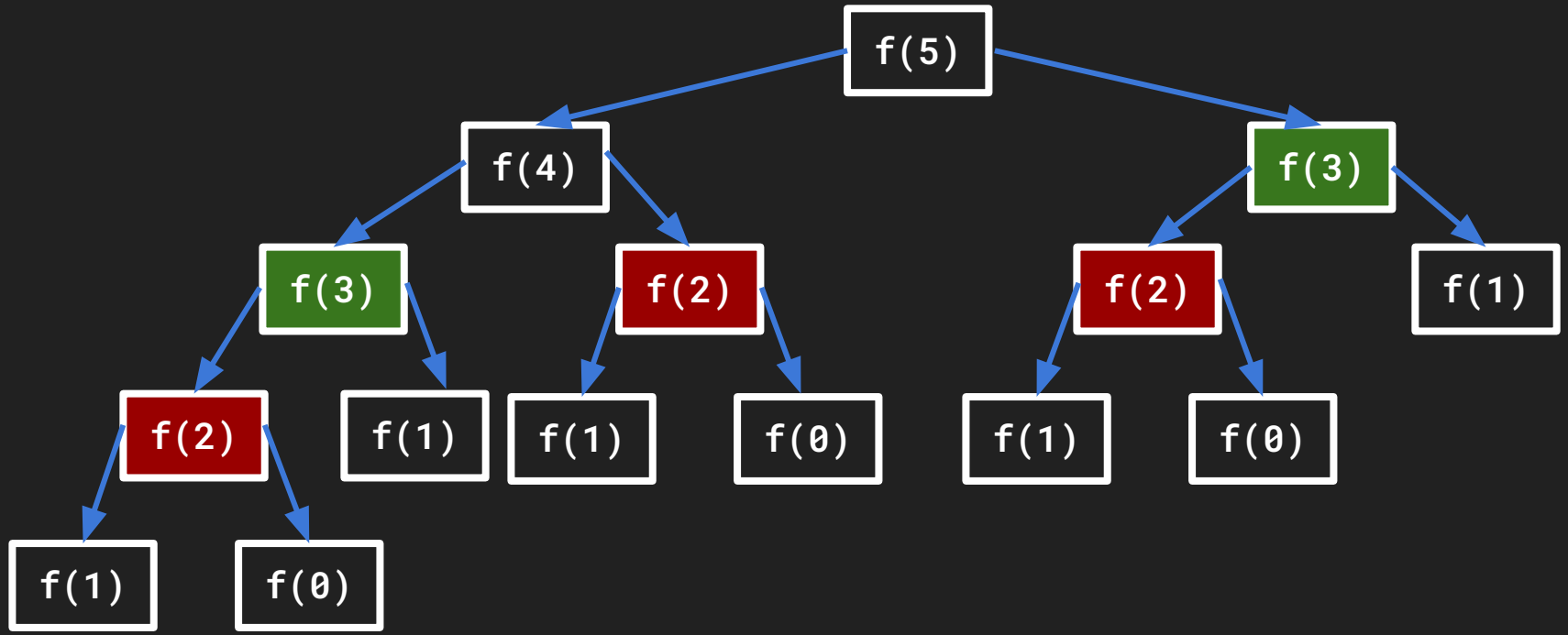
Recursion(cont.)



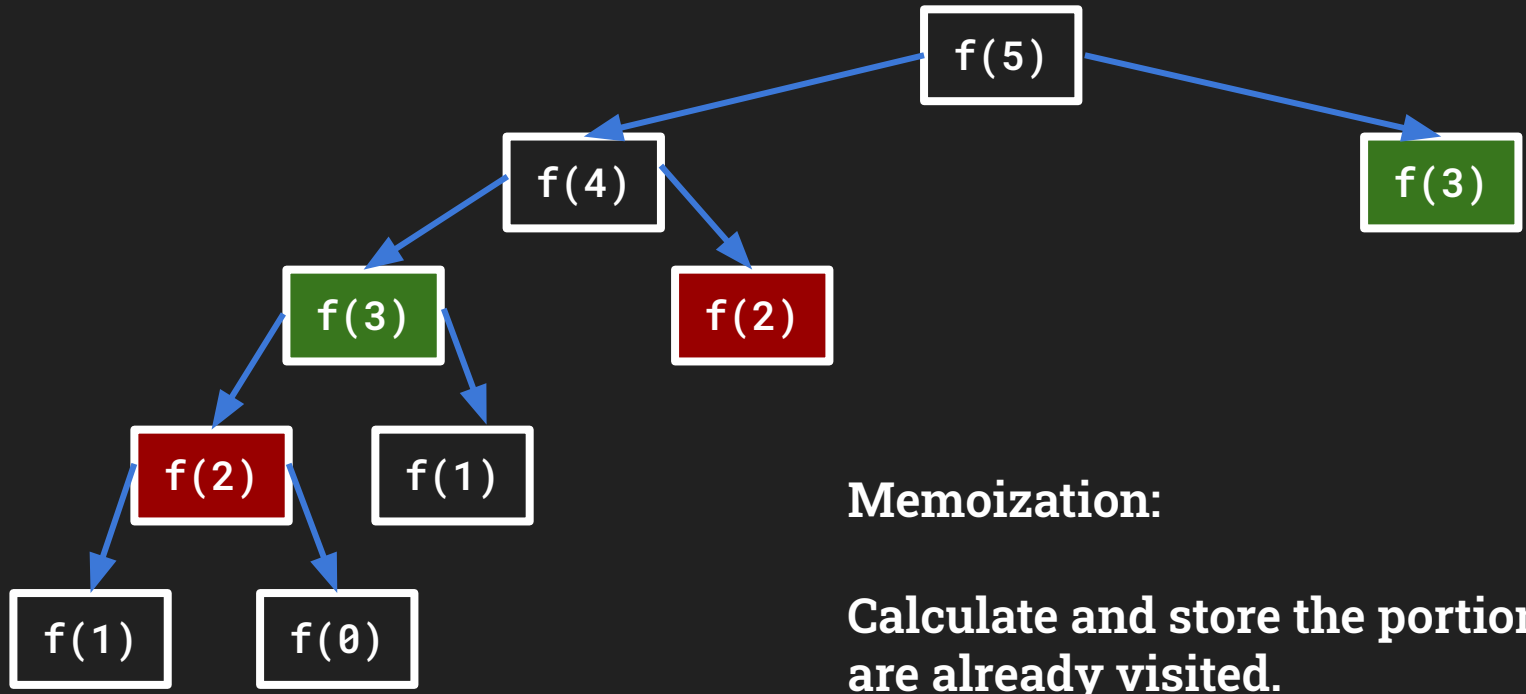
Recursion(cont.)



Recursion(cont.)



Dynamic Programming



Memoization:

Calculate and store the portions that are already visited.

TABLE OF CONTENTS

Setup the Environment

- VSCode
- Jupyter Notebook
- Google Colaboratory
- Kaggle