DYNAMIC PROGRAMMING

INTRODUCTION TO DYNAMIC PROGRAMMING

Goals

To better understand recursive algorithms.

To gain a sense of what dynamic programming is and how to apply it to improve the efficiency of recursive algorithms.

DYNAMIC PROGRAMMING

Concept — a method for solving a problem by breaking it down into simpler subproblems while considering that the optimal solution for the overall problem depends upon finding the optimal solution for each subproblem

DP Problems — problems that are solvable via dynamic programming are made up of overlapping subproblems with an optimal substructure

a problem has overlapping subproblems if the solution involves solving the same subproblem multiple times

a problem has optimal substructure if the optimal solution for the problem can be found from optimal solutions of its subproblems

DYNAMIC PROGRAMMING

Methods	dynamic programming can be implemented via memoization or tabulation
Memoization	recursively compute a solution from the big problem down to smaller subproblems, known as the top-down approach
	resolve overlapping subproblems by caching the results of subproblem computations for future lookup (avoiding repetitive computations)
Tabulation	iteratively compute the solution from subproblems up to the big problem, known as the bottom-up approach
	subproblem solutions are cached and then used to compute bigger problems

RECURSIVE FIBONACCI ALGORITHM

Problem given an input n, return the nth value of the fibonacci sequence

n	0	1	2	3	4	5	6	7	8	9	10
value	0	1	1	2	3	5	8	13	21	34	55

Example if n == 6, return 8
if n == 9, return 34

```
long fibonacci(int n) {
    if(n<2) {   return n; }                          // base case
    return fibonacci(n-1) + fibonacci(n-2);          // recursive task
}
```

RECURSIVE FIBONACCI ALGORITHM

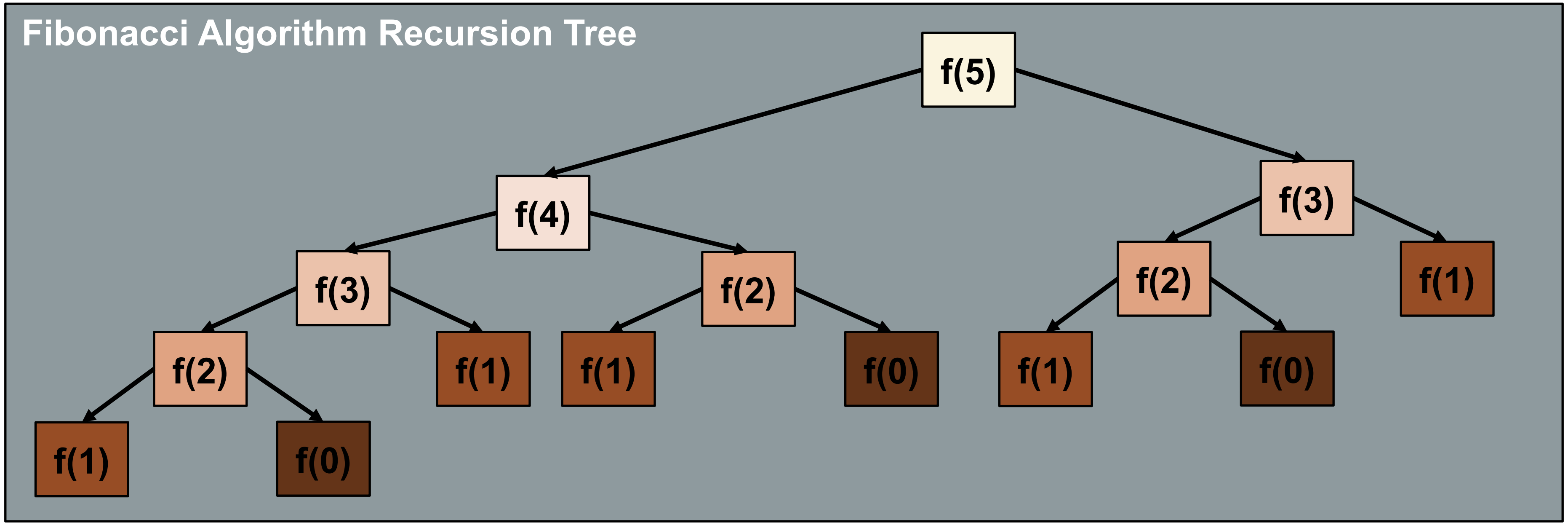
Time Complexity $O(2^n)$

Base case `if(n<2) { return n; }`

Recursive task `return fibonacci(n-1) + fibonacci(n-2);`

note the repetitive sub problem computations

run the example code with values of n greater than 40



ex_2_fibonacci_recursive.cpp

FIBONACCI RECURSION PROBLEM

Question

Is the recursive Fibonacci algorithm a candidate for DP?

1. repetitive function calls illustrate overlapping subproblems

2. the recursive task fibonacci(n-1) + fibonacci(n-2) recursively reduces a problem of size n into problems of size n-1 and n-2 indicating this problem has optimal substructure property

MEMOIZATION APPROACH

Task

recursively compute a fibonacci solution from the big problem down to smaller subproblems, known as the top-down approach

create an array with all values defaulted to -1 to store subproblem solutions

assign 0 and 1 as the first values in the array (first numbers in the sequence)

```
long fibonacci(int n, long *a) {
    if(n<2) return the value at a[n]
    otherwise if a[n] has no value
        compute and store the result of fibo(n-1, a) + fibo(n-2, a)
    otherwise return the value at a[n]
}
```

TABLUATION APPROACH

Task

iteratively compute a fibonacci solution from subproblems up to the big problem, known as the bottom-up approach

```
long fibonacci(int n) {
    create an array to store subproblem answers
    assign 0 and 1 as the first values in the array (first numbers in the sequence)
    iterate from 2 to n
        compute and store the result of n-1 + n-2
    return the value at a[n]
}
```

HOMEWORK

1. Code a memoized fibonacci algorithm using an array with dynamic memory.
    a. Compare this solutions performance to the normal algorithm for values of n up to 46.
    b. Draw the recursion tree for the memoized algorithm of the fibonacci sequence.
    c. What is the time complexity and how does it compare to the original algorithm?

2. Code a tabulation fibonacci algorithm using an array with dynamic memory.
    a. Compare this solutions performance to the normal algorithm for values of n up to 46.
    b. What is the time complexity and how does it compare to the original algorithm?