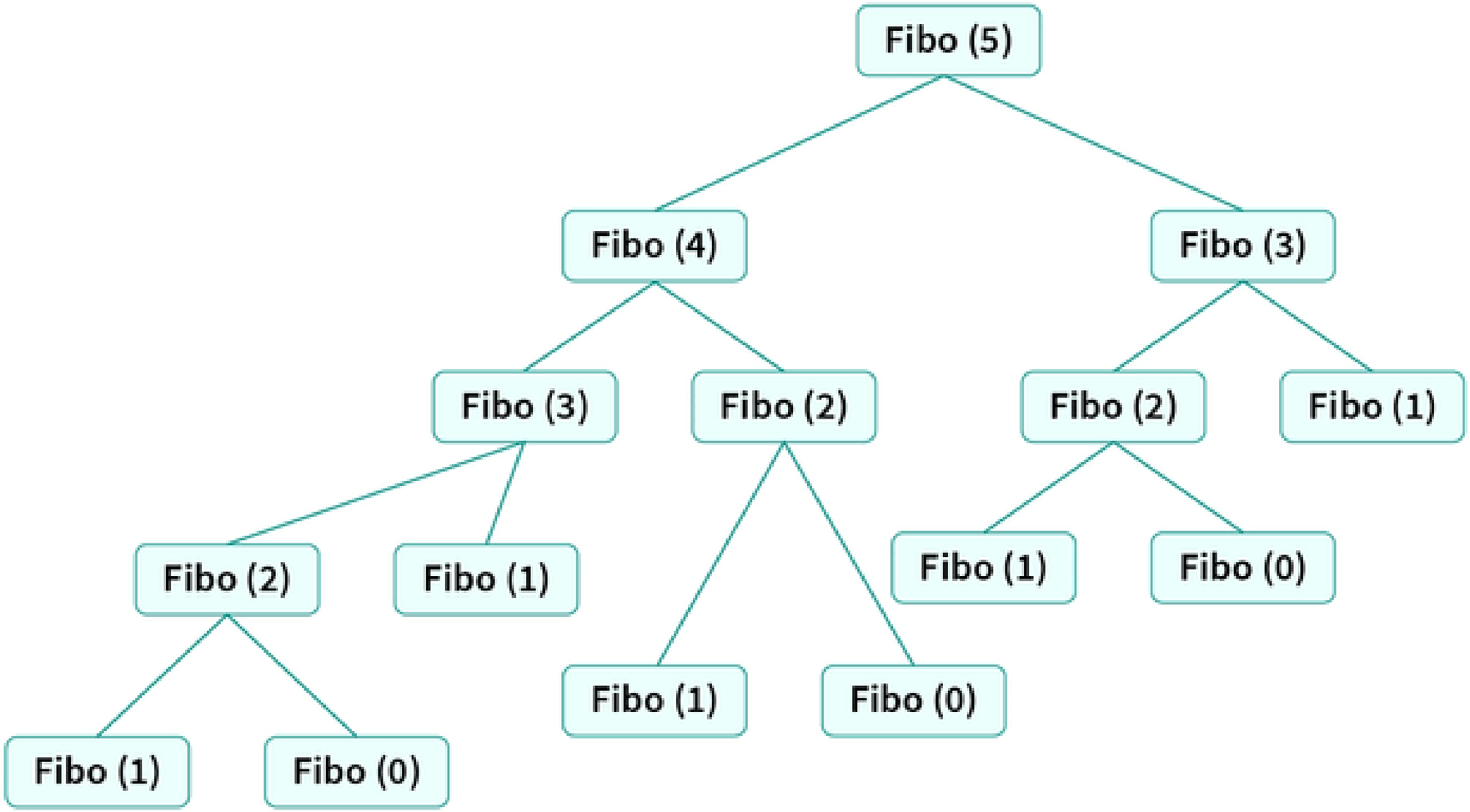
WELCOME

Table of Contents

Dynamic programming

Dynamic programming is a technique that breaks the problems into sub-problems, and saves the result for future purposes so that we do not need to compute the result again.

Example


Fibo (5)
Fibo (4)
Fibo (3)
Fibo (3)
Fibo (2)
Fibo (2)
Fibo (1)
Fibo (2)
Fibo (1)
Fibo (1)
Fibo (0)
Fibo (1)
Fibo (0)
Fibo (1)
Fibo (0)

Example

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

Time complexity : O(2^n)

space complexity : O(n)

How does the dynamic programming approach work?

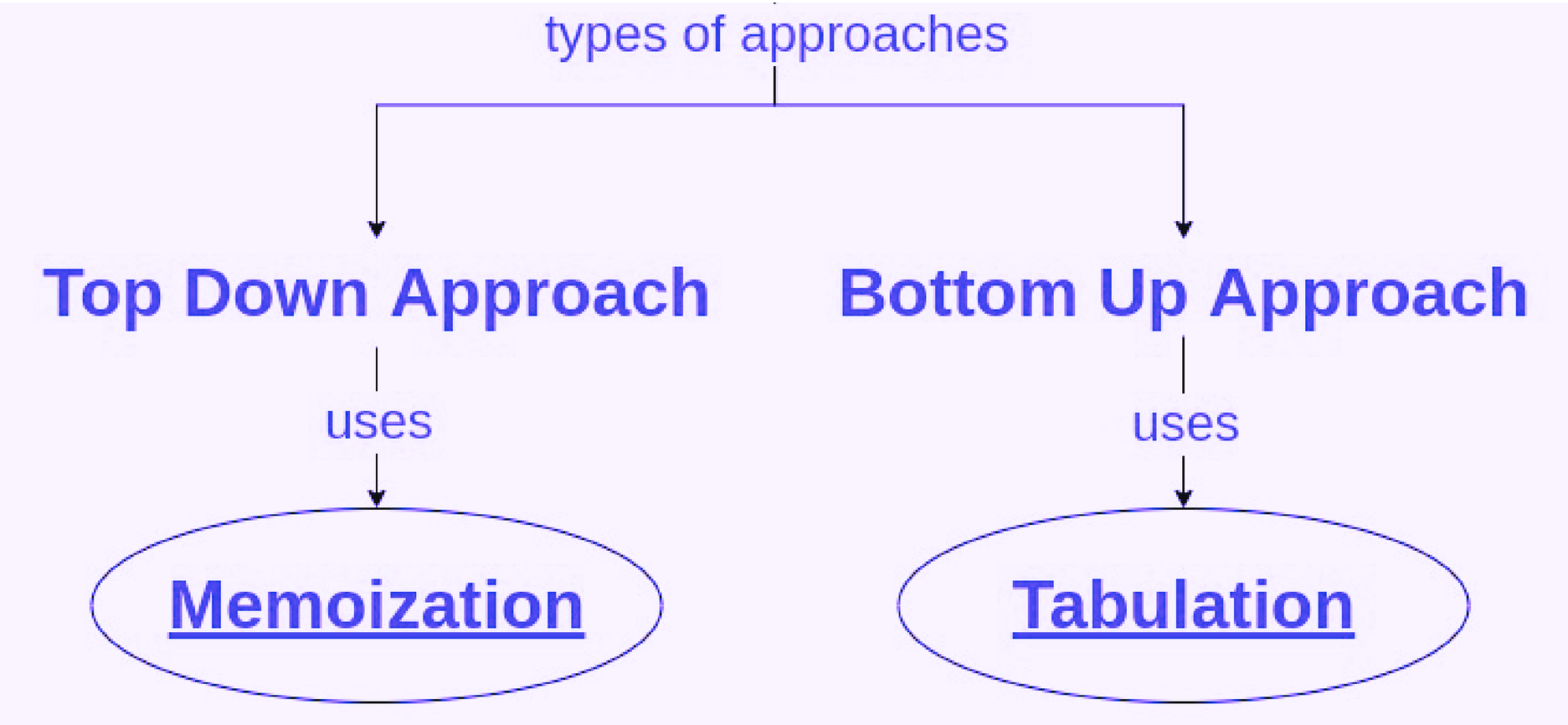
- It breaks down the complex problem into simpler subproblems.
- It finds the optimal solution to these sub-problems.
- It stores the **results of subproblems (memoization).** The process of storing the results of subproblems is known as memorization.
- It reuses them so that same sub-problem is calculated more than once.
- Finally, calculate the result of the complex problem.

Recursion vs Dynamic Programming

Dynamic programming is mostly applied to recursive algorithms. This is not a coincidence, most optimization problems require recursion and dynamic programming is used for optimization.

But not all problems that use recursion can use Dynamic Programming. Unless there is a presence of overlapping subproblems like in the fibonacci sequence problem, a recursion can only reach the solution using a divide and conquer approach.

Approaches of dynamic programming



Bottom UP Approach Code

```
int dp[MAXN];
int dp[0] = 1;
for (int i = 1; i <= n; i++)
{
    dp[i] = dp[i - 1] * i;
}
```

Top Down Approach Code

```
int solve(int x)
{
    if (x == 0)
        return 1;
    if (dp[x] != -1)
        return dp[x];
    return (dp[x] = x * solve(x - 1));
}
```

Characteristics of Dynamic Programming

- Simple subproblems
  - We should be able to break the original problem to **smaller subproblems** that have the **same** structure
- Optimal substructure of the problems
  - The **optimal solution** to the problem contains within **optimal solutions to** its **subproblems**.
- Overlapping sub-problems
  - there exist some places where we solve the **same subproblem** more than **once**.

Where DP can be used

The following computer problems can be solved using dynamic programming approach –

- Fibonacci number series
- Knapsack problem
- Tower of Hanoi
- All pair shortest path by Floyd-Warshall
- Shortest path by Dijkstra
- Project scheduling

Some Resources(To learn)

To know more About DP, follow these articles

1. https://www.shafaetsplanet.com/?p=1022
2. http://www.shafaetsplanet.com/?p=1072
3. http://www.shafaetsplanet.com/?p=1211
4. https://www.topcoder.com/community/competitive-programming/tutorials/dynamic-programming-from-novice-to-advanced/

Some Resources(To Practice)

- https://lightoj.com/problem/hex-a-bonacci
- https://cses.fi/problemset/task/1633
- https://cses.fi/problemset/task/1634
- https://lightoj.com/problem/neighbor-house
- https://lightoj.com/problem/coin-change-i
- https://cses.fi/problemset/task/1635
- https://cses.fi/problemset/task/1636
- https://lightoj.com/problem/rooks
- https://cses.fi/problemset/task/1637

Any Queries

THANK YOU