EngineerPro - K01

Backtracking with memoization

Le Vu Nguyen Chuong, 2023.

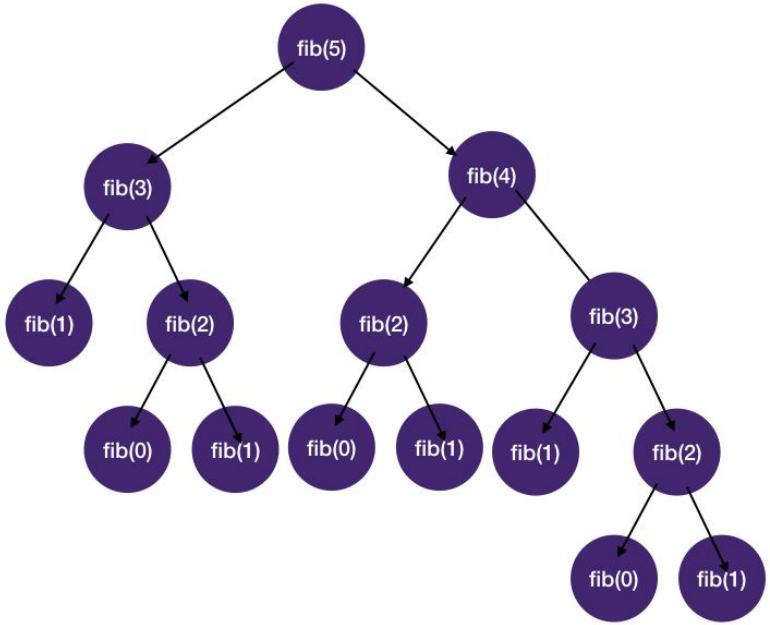
Contents

1. Backtracking with memoization

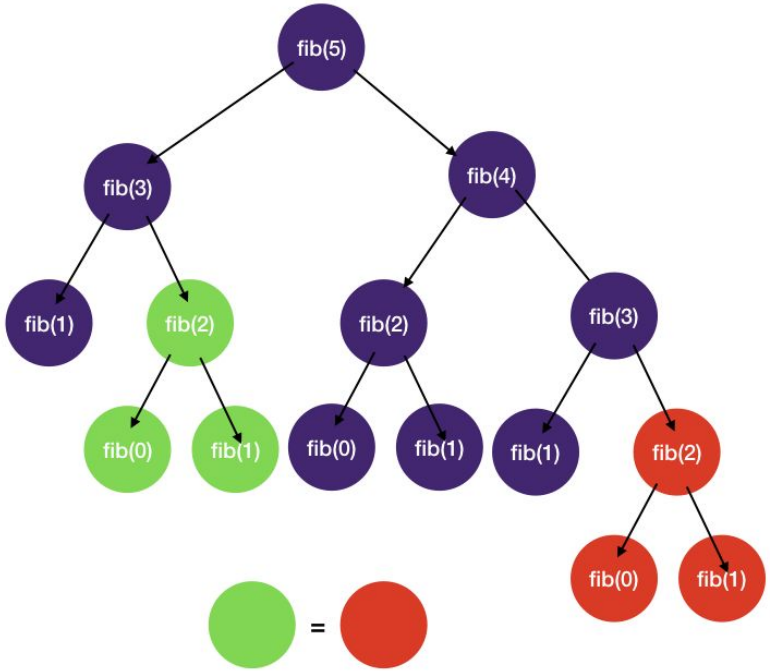
Backtracking with memoization

- When doing backtracking to find solutions for a problem, we may encounter the same recursive function call with same set of inputs.
- We can memorize the result to reuse in a later call.

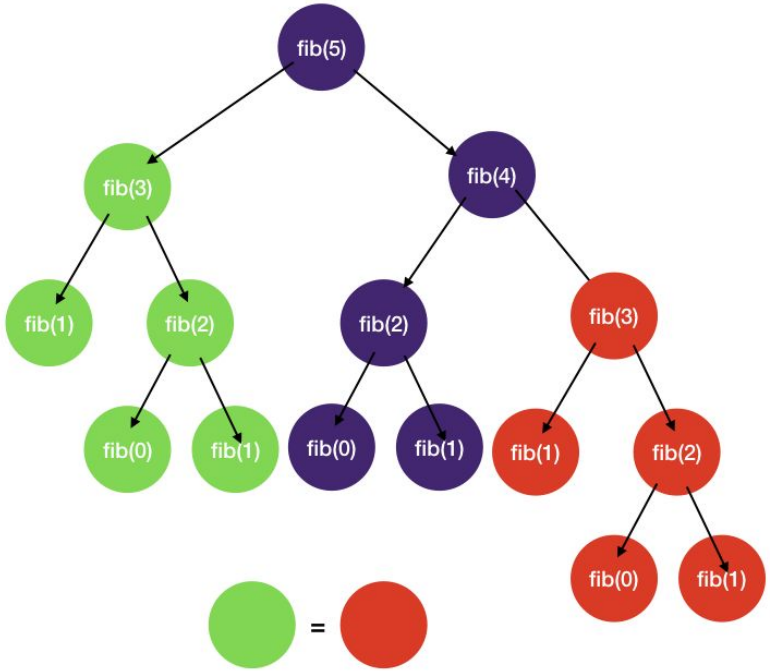
Example: fibonacci calculation



Example: fibonacci calculation



Example: fibonacci calculation



Example: fibonacci calculation

Naive solution:

```
int fib(int n) {
    if (n == 0 || n == 1) {
        return n;
    }
    return fib(n - 1) + fib(n - 2);
}
```

Solution with memoization

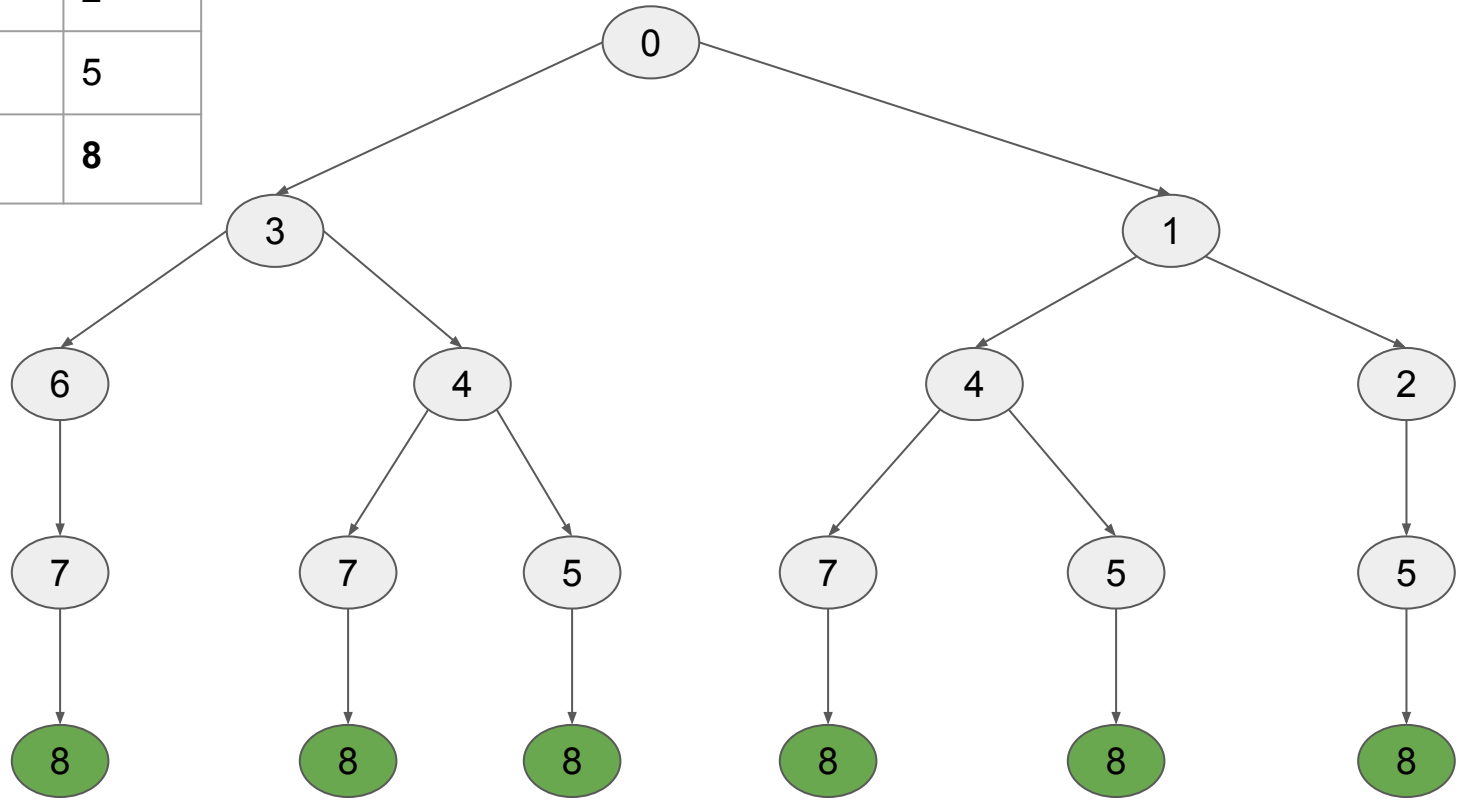
```
int fib(int n, int[] memo) {
    // check in memo, if found, retrieve and return
right away
    if (memo[n] != -1) return memo[n];

    if (n == 0 || n == 1) return n;

    int res = fib(n - 1, memo) + fib(n - 2, memo);

    // save result to memo before returning
    memo[n] = res;
    return res;
}
```

Example: Count numbers of paths

Given an m*n array, we want to count number of paths from top left cell to bottom right cell
Constraint: can only move down or right

0 (Source)	1	2
3	4	5
6	7	**8 (Destination)**

0	1	2
3	4	5
6	7	**8**



Example: Count numbers of paths

Naive solution:

```
public int uniquePaths(int m, int n) {
    return dfs(0, 0, m, n);
}

private int dfs(int x, int y, int m, int n) {
    if (x == m-1 && y == n-1) {
        return 1;
    }
    int val = 0;
    if (x < m-1) {
        val += dfs(x+1, y, m, n);
    }
    if (y < n-1) {
        val += dfs(x, y+1, m, n);
    }
    return val;
}
```

Example: Count numbers of paths

Solution with memoization

```
public int uniquePaths(int m, int n) {
    int[][] dp = new int[m][n];
    for (int[] i : dp) {
        Arrays.fill(i, -1);
    }
    dp[m-1][n-1] = 1;
    return dfs(0, 0, m, n, dp);
}
```

```
private int dfs(int x, int y, int m, int n, int[][] dp) {
    if (dp[x][y] != -1) {
        return dp[x][y];
    }
    int val = 0;
    if (x < m-1) {
        val += dfs(x+1, y, m, n, dp);
    }
    if (y < n-1) {
        val += dfs(x, y+1, m, n, dp);
    }
    dp[x][y] = val;
    return val;
}
```

Time complexity: O(M*N)
Space complexity: O(M*N)

Backtracking with memoization

When to memoize ?

- Identify the possible states and if we see duplication in the states, then we can use memoization

What to memoize ?

- The attribute for each state. For example:
  - In the fibonacci problem, it is the **result** of each **number.**
  - In the count path problem, it is the **number of paths** for each **cell.**

Backtracking with memoization analysis

Time complexity: O(Number of states)

Space complexity: O(Number of states)

2. Live coding

Problem 1

- https://leetcode.com/problems/unique-paths/

Problem 2

- https://leetcode.com/problems/out-of-boundary-paths/

Problem 3

- https://leetcode.com/problems/word-break/description/

Problem 4

- https://leetcode.com/problems/longest-increasing-path-in-a-matrix/

3. Homework

Homework

- https://leetcode.com/problems/target-sum/description/
- https://leetcode.com/problems/cheapest-flights-within-k-stops
- https://leetcode.com/problems/house-robber-iii/description/