Mapcode : Recursion Modeling

Nitheesh Chandra
nitheeshchandra.y@research.iiit.ac.in
TA, PoPL

Contributors from PoPL 2025, IIITH
Students, PoPL

2025-11-16

Abstract

Recursion modeling in the `mapcode` framework.

Keywords Mapcode · Recursion · Programming

1 Notations and Conventions

A vector is (zero-)indexed as follows that has size $n+1$:

$$\begin{bmatrix} x_0 \\ \vdots \\ x_n \end{bmatrix}_{n+1}$$

A matrix is also (zero-)indexed as follows that has size $m \times n$:

$$\begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,n-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,n-1} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m-1,0} & x_{m-1,1} & \cdots & x_{m-1,n-1} \end{bmatrix}_{m \times n}$$

All primitives are *strict* meaning they do not allow for undefined values (i.e., $\bot$) to be used in their computations.

2 Examples

2.1 Factorial

Compute the factorial of a non-negative integer n. i.e $n \in \mathbb{N}_0, n \geq 0$

Formal definition:

$$n! = \prod_{k=1}^{n} k$$

Equivalently:

$$0! = 1$$
$$n! = n * (n-1)! \text{ for } n \in \mathbb{N}, n \geq 1$$

Examples:

- fact(0) $\rightarrow$ 1
- fact(5) $\rightarrow$ 120

As mapcode:

primitives: `product`$(*)$ and `subtract`$(-)$ are strict. i.e product and subtract on $\perp$ is undefined.

$$I = n : \mathbb{N} \qquad X_n = [0..n] \rightarrow \mathbb{N}_\perp \qquad A = \mathbb{N}$$
$$\rho(n) = \{i \rightarrow \perp \mid i \in [0..n]\}$$
$$F(x_n) = \begin{cases} 1 & \text{if } n = 0 \\ n * x_{n-1} & \text{if } n > 0 \end{cases}$$
$$\pi(x) = \text{last}(x) = x_{|x|-1}$$

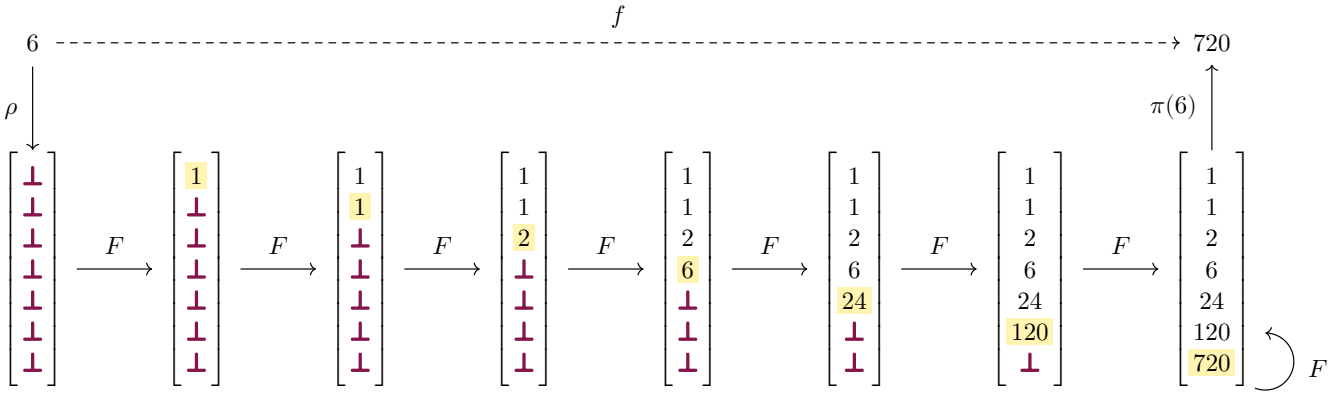

Figure 1: Factorial computation using mapcode for $n = 6$

2.2 Fibonacci

Compute the n-th Fibonacci number, where n is a non-negative integer.

Formal definition:

$$F(0) = 0,$$
$$F(1) = 1,$$
$$F(n) = F(n-1) + F(n-2) \quad \text{for} \quad n \geq 2$$

Examples:

- fib(0) $\rightarrow$ 0
- fib(1) $\rightarrow$ 1
- fib(4) $\rightarrow$ 5
- fib(8) $\rightarrow$ 21

As mapcode:

primitives: `sum`$(+)$ and `minus`$(-)$ are strict. i.e sum and minus on $\perp$ is undefined.

$$I = n : \mathbb{N} \qquad X_n = [0..n] \rightarrow \mathbb{N}_{\perp} \qquad A = \mathbb{N}$$
$$\rho(n) = \{i \rightarrow \perp \mid i \in \{0...n\}\}$$
$$F(x_n) = \begin{cases} 0 & \text{if} \quad n = 0 \\ 1 & \text{if} \quad n = 1 \\ x_{n-1} + x_{n-2} & \text{if} \quad n \geq 2 \end{cases}$$
$$\pi(x) = \text{last}(x) = x_{|x|-1}$$

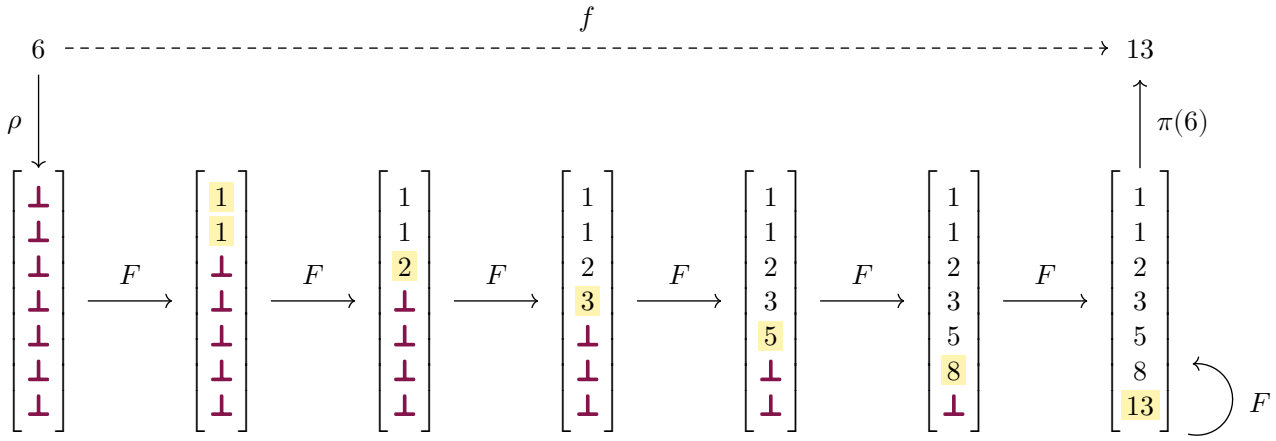

Figure 2: Factorial computation using mapcode for $n = 6$

2.3 Binomial Coefficients

Compute the binomial coefficient $C(n, k)$, which represents the number of ways to choose k elements from a set of n elements.

Formal definition:

$$C(n,k) = \frac{n!}{k!(n-k)!}$$

equivalently, as recursive definition:

$$C(n,k) = \begin{cases} 1 & \text{if } k = 0 \\ 1 & \text{if } k = n \\ C(n-1,k-1) + C(n-1,k) & \text{otherwise} \end{cases}$$

Example:

- C(5, 2) = 10
- C(5, 0) = 1
- C(5, 5) = 1
- C(10, 3) = 120

As mapcode:

primitives: `sum`$(+)$, `sub`$(-)$

$$I = n : \mathbb{N} \times k : [0..n] \qquad X_{n,k} = [0..n] \times [0..k] \to \mathbb{N}_{\perp} \qquad A = \mathbb{N}$$

$$\rho(n,k) = \{(i,j) \to \perp \mid i \in \{0...n\}, j \in \{0...k\}\}$$

$$F(x_{i,j}) = \begin{cases} 1 & \text{if } j = 0 \vee j = i \\ x_{i-1,j-1} + x_{i-1,j} & \text{if } i \geq j \end{cases}$$

$$\pi_{n,k}(x) = x_{n,k}$$

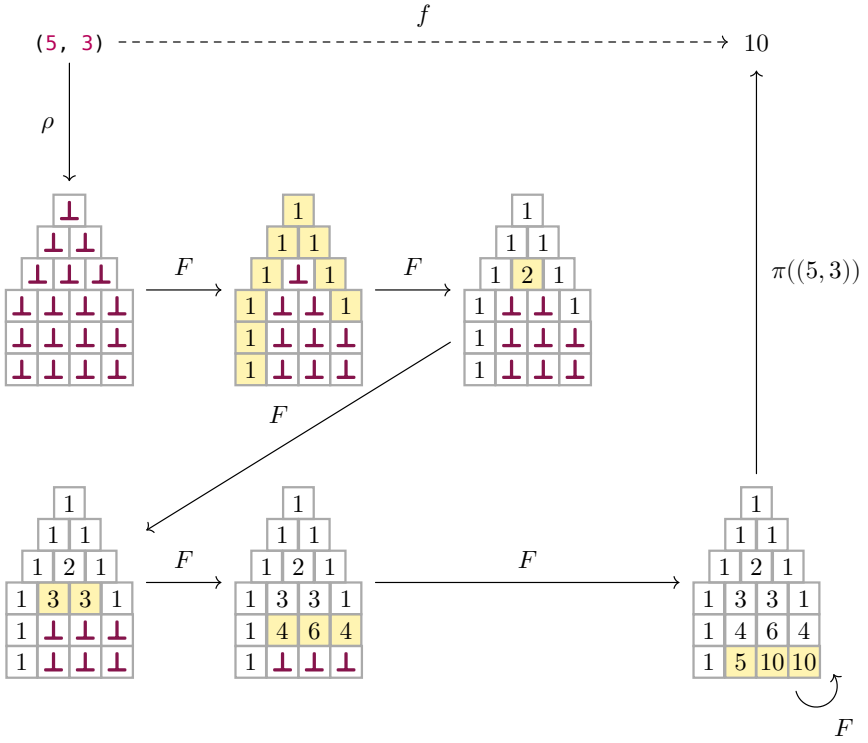

Figure 3: Binomial Coefficient computation using mapcode for $n = 5$ and $k = 3$; A Pascal's Triangle style visualization is used.

2.4 Length of Longest Common Subsequence

Compute the length of the longest common subsequence (LCS) of two sequences S and T. A subsequence is a sequence that can be derived from another sequence by deleting some elements without changing the order of the remaining elements.

Formal definition:

$$\mathrm{LCS}(i,j) = \begin{cases} 0 & \text{if } \ i = 0 \vee j = 0 \\ \mathrm{LCS}(i-1, j-1) + 1 & \text{if } \ S_i = T_j \\ \max(\mathrm{LCS}(i-1,j), \mathrm{LCS}(i,j-1)) & \text{otherwise} \end{cases}$$

Example:

- $S = \mathrm{AGGTAB}, T = \mathrm{GXTXAYB} \rightarrow$ LCS length $= 4(\mathrm{GTAB})$

As mapcode:

primitives: `sum`$(+)$, `max`$(\max)$

$$I = i : [0..m] \times j : [0..n] \qquad X_{i,j} = [0..m] \times [0..n] \to \mathbb{N}_{\perp} \qquad A = \mathbb{N}$$

$$\rho(m,n) = \{(i,j) \to \perp \mid i \in \{0...m\}, j \in \{0...n\}\}$$

$$F_{S,T}(x_{i,j}) = \begin{cases} 0 & \text{if } \ i = 0 \vee j = 0 \\ x_{i-1,j-1} + 1 & \text{if } \ S_i = T_j \\ \max(x_{i-1,j}, x_{i,j-1}) & \text{otherwise} \end{cases}$$

$$\pi_{S,T}(x) = x_{m,n} \qquad \text{where } m = |S|, n = |T|$$

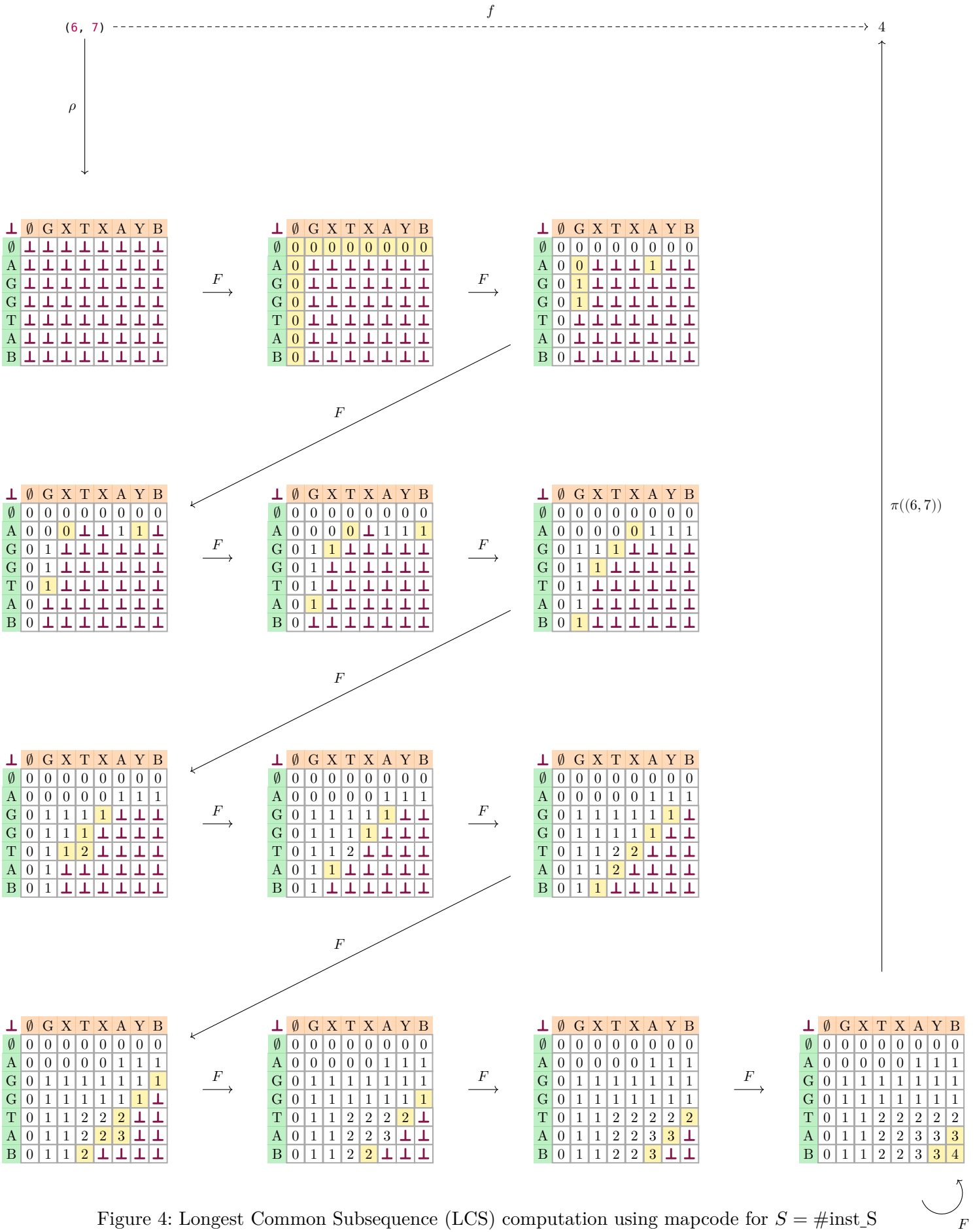

Figure 4: Longest Common Subsequence (LCS) computation using mapcode for $S =$ #inst_S and $T =$ #inst_T; dynamic-programming table visualization.

2.5 Add Two Numbers LeetCode P.2

You are given two non-empty linked lists representing two non-negative integers. The digits are stored in reverse order, and each of their nodes contains a single digit. Add the two numbers and return the sum as a linked list.

You may assume the two numbers do not contain any leading zero, except the number 0 itself.

Example 1:
Input $l_1 = [2, 4, 3], l_2 = [5, 6, 4]$
Output $[7, 0, 8]$
Explanation $342 + 465 = 807$.

Example 2:
Input $l_1 = [0], l_2 = [0]$
Output $[0]$

Example 3:
Input $l_1 = [9, 9, 9, 9, 9, 9, 9], l_2 = [9, 9, 9, 9]$
Output $[8, 9, 9, 9, 0, 0, 0, 1]$

Constraints:

The number of nodes in each linked list is in the range [1, 100]. 0 <= Node.val <= 9 It is guaranteed that the list represents a number that does not have leading zeros.

Mapcode:

primitives: `sum`(+), `head`(head)(to get value at head of linked list), `tail`(tail)(to get next nodes in linked list), `div_euclid` (÷)(integer division), `rem`(mod)(modulus)

let

- next : $[0..9] \to [0..9] \cup \bot$ be a successor function.
- head : $h \in [0..9] \cup \bot$ be a head node.
- $\mid L \mid : l \in [1..100]$ be the length of linked list.

Then the linked list is represented as:

$$\text{List} = (h, \text{next}(h), \text{next}^2(h), \text{next}^3(h), ..., \text{until } \bot)$$

$$m \in [0..100] \quad n \in [0..100]$$

$$I : l_1 : \text{List} \times l_2 : \text{List} \quad |l_1| = m \quad |l_2| = n$$

$$X_{l_1,l_2} : k \in [0..\max(m,n)) \to \left(\text{next}^k(l_1), \text{next}^k(l_2), \text{carry} : [0..1], \text{result} : [0..1] \cup \{\bot\}\right)$$

$$A : \text{List}$$

$$\rho(l_1, l_2) = \{(a, b, \text{res}, \text{carry}) \mid a \in \text{List}, b \in \text{List}, \text{carry} = 0, \text{res} = \bot\}$$

$$F(l_1,l_2)(x_k) = \begin{cases} \begin{bmatrix} a \\ b \\ \text{res} \\ \text{carry} \end{bmatrix} & \text{if } a = \bot \wedge b = \bot \wedge \text{carry} = 0 \\ \begin{bmatrix} a \\ b \\ \text{carry} \\ 0 \end{bmatrix} & \text{if } a = \bot \wedge b = \bot \wedge \text{carry} \neq 0 \\ \begin{bmatrix} a \\ b \\ \text{digit} \\ \text{carry} \end{bmatrix} & \text{otherwise where } \begin{cases} \text{val}_a = \text{if } a \neq \bot \text{ then head}(a) \text{ else } 0 \\ \text{val}_b = \text{if } b \neq \bot \text{ then head}(b) \text{ else } 0 \\ \text{sum} = \text{val}_a + \text{val}_b + x_{k-1}.\text{carry} \\ \text{digit} = \text{sum} \bmod 10 \\ \text{carry} = \text{sum} \div 10 \end{cases} \end{cases}$$

$$\pi(l_1,l_2)(x) = [\text{digit} \mid i \in [0..|x|), (?,?,\text{digit},?) = x_i, \text{digit} \neq \bot] + \begin{cases} \left[x_{|x|-1}.\text{carry}\right] & \text{if } x_{|x|-1}.\text{carry} \neq 0 \\ [] & \text{otherwise} \end{cases}$$

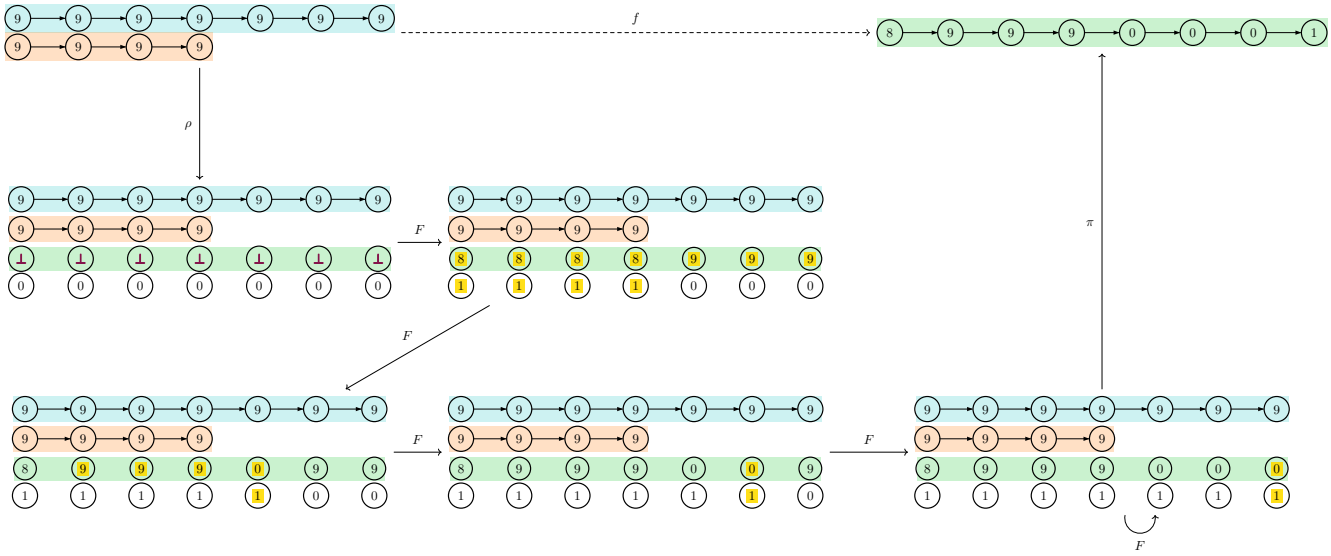

Figure 5: Add Two Numbers computation using mapcode for $l_1 =$ `(9, 9, 9, 9, 9, 9, 9)` and $l_2 =$ `(9, 9, 9, 9)`

2.6 Longest Increasing Subsequence (LIS)

Compute the length of the longest increasing subsequence in an array.

Formal definition:

$$\text{LIS}(i) = \begin{cases} 1 & \text{if } i = 0 \\ 1 + \max_{j<i, A[j]<A[i]} \text{LIS}(j) & \text{if } i > 0 \text{ and valid j exists} \\ 1 & \text{otherwise} \end{cases}$$

Example: A = [10, 9, 2, 5, 3, 7, 101] LIS length = 4 (subsequence: [2, 3, 7, 101] or [2, 5, 7, 101])

As mapcode:

primitives: `max` and `add`$(+)$ are strict. i.e max and add on $\perp$ is undefined.

$$I = (n : \mathbb{N}, A : \text{array}) \qquad X = [0..n-1] \to \mathbb{N}_{\perp} \qquad A = \mathbb{N}$$

$$\rho(n, A) = \{i \to 1 \mid i \in [0..n-1]\}$$

$$F(x)[i] = \begin{cases} 1 & \text{if } i = 0 \\ 1 + \max_{j<i, A[j]<A[i]} \{x[j]\} & \text{if valid j and deps defined} \\ 1 & \text{otherwise} \end{cases}$$

$$\pi(x) = \max_i x[i]$$

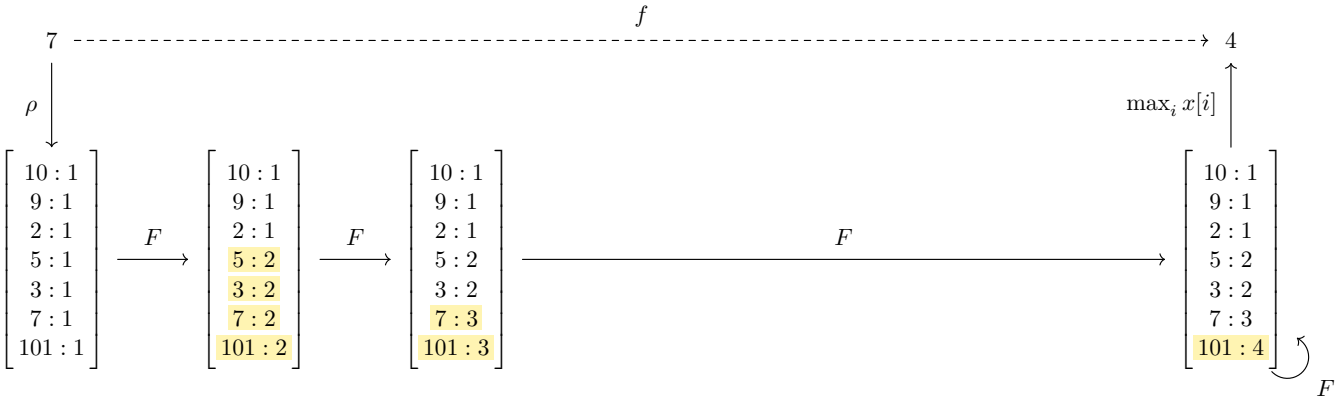

Figure 6: LIS computation using mapcode for array = `(10, 9, 2, 5, 3, 7, 101)`

2.7 Coin Change (Minimum Coins)

Compute the minimum number of coins needed to make a target amount.

Formal definition:

$$\mathrm{CC}(0) = 0$$
$$\mathrm{CC}(n) = \min_{c \in \text{ coins}, c \le n} (1 + \mathrm{CC}(n - c)) \quad \text{for } n > 0$$

Example: coins = [1,3,4], amount = 6 Minimum coins = 2 (using 3+3)

As mapcode:

primitives: `min` and `add`(+) are strict. i.e min and add on $\bot$ is undefined.

$$I = \text{amount} : \mathbb{N} \qquad X = [0..\text{amount}] \to \mathbb{N}_{\bot} \qquad A = \mathbb{N}$$
$$\rho(\text{amount}) = \{i \to \bot \mid i \in [0..\text{amount}], i > 0\} \cup \{0 \to 0\}$$
$$F(x)[i] = \begin{cases} 0 & \text{if } i = 0 \\ \min_{c \in \text{ coins}, c \le i}\{1 + x[i-c]\} & \text{if } i > 0 \text{ and deps defined} \\ \bot & \text{otherwise} \end{cases}$$
$$\pi(x) = x[\text{amount}]$$

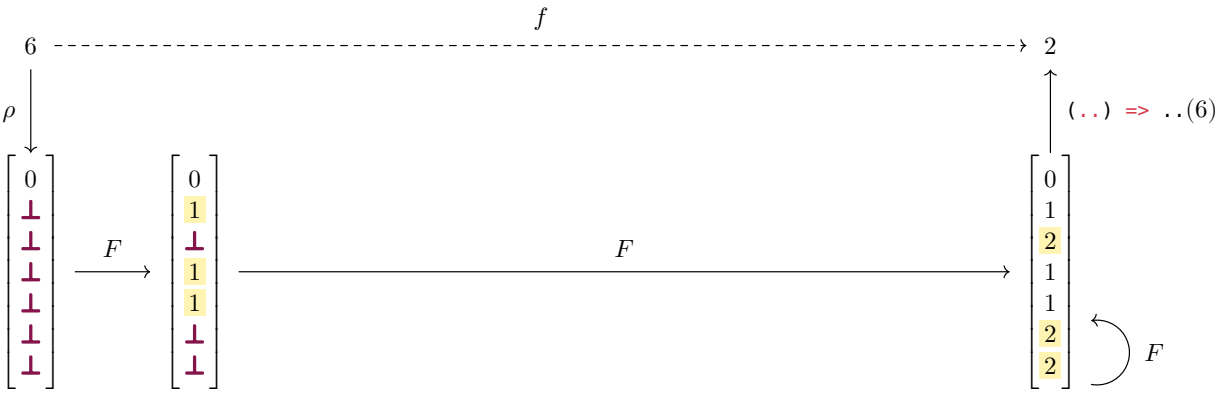

Figure 7: Coin Change computation using mapcode for amount = 6, coins = `(1, 3, 4)`