Dynamic Programming
동적 계획법

Data Structures and Algorithms

목차

- 동적 계획법
- 피보나치 수열
- 최장 공통 부분 순서

동적 계획법

동적 계획법

- 어떤 문제가 여러 단계의 반복되는 부분 문제로 이루어질 때, 각 단계에 있는 부분 문제의 답을 기반으로 전체 문제의 답을 구함
  - 재귀적으로 해결할 수 있는 문제 → 반복적인 방법으로 해결
  - 같은 문제는 한 번만 푼다 (메모화)

동작 방식

1. 부분 문제로 나누기
2. 가장 작은 부분 문제의 해를 구하기
3. 해를 배열/테이블 등에 저장하기
4. 계산된 해를 이용해 다음 작은 부분 문제 해 구하기
5. 2번 부터 반복

동적 계획법의 적용 전제 조건

- 단, 최적 부분구조를 갖춘 문제라는 전제가 필요함
  - 전체 문제의 최적해가 부문 문제의 최적해로 만들어지는가?
  - 예: 서울, 경기, 부산, 대구, 진주로 도착하는 최단 경로의 길이의 합
    - 각 지역까지 도착하는 최단 경로의 길이를 알아야 전체 최단 경로 길이의 합을 구할 수 있음
    - 이 문제는 최적 부분 구조를 갖음

피보나치 수 구하기

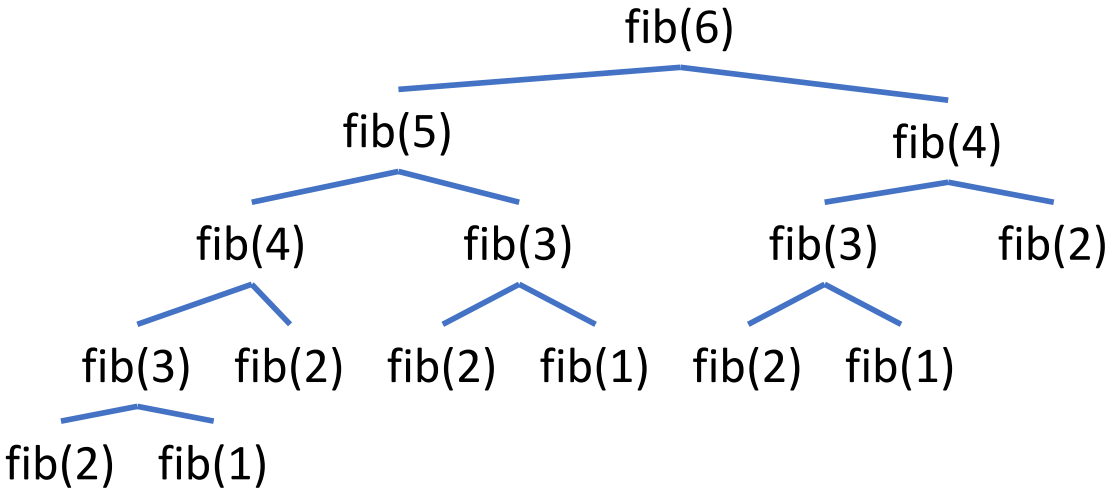
피보나치 수열 → 코드

- 수열: $0,\ 1,\ 1,\ 2,\ 3,\ 5,\ 8,\ 13,\ 21,\ 34,\ \ldots$
- 패턴: $n^{th}\text{수열} = (n-1)^{th}\text{ 수열} + (n-2)^{th}\text{ 수열}$
- 조건:

$$f(n) = \begin{cases} 0 & n = 1 \\ 1 & n = 2 \\ fib(n-1) + fib(n-2) & else \end{cases}$$

피보나치 (재귀적 방법)

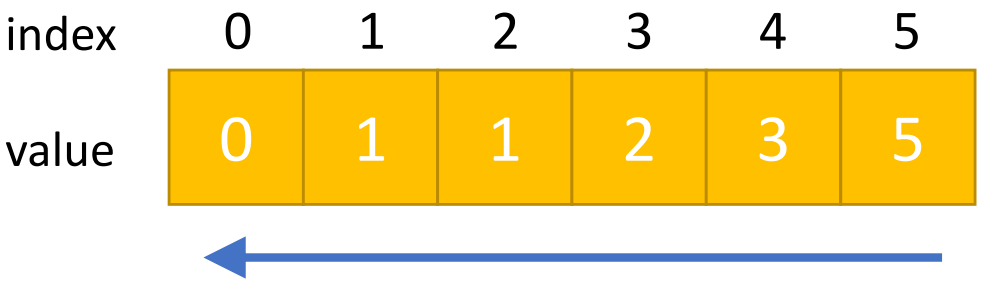
```
#include <stdio.h>

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

int main()
{
    for(int i = 1; i < 15; i++)
        printf( " %d " , fib(i));

    return 0;
}
```



지면 관계상 fib(6)부터 예시

피보나치 (재귀적 메모화 방법)

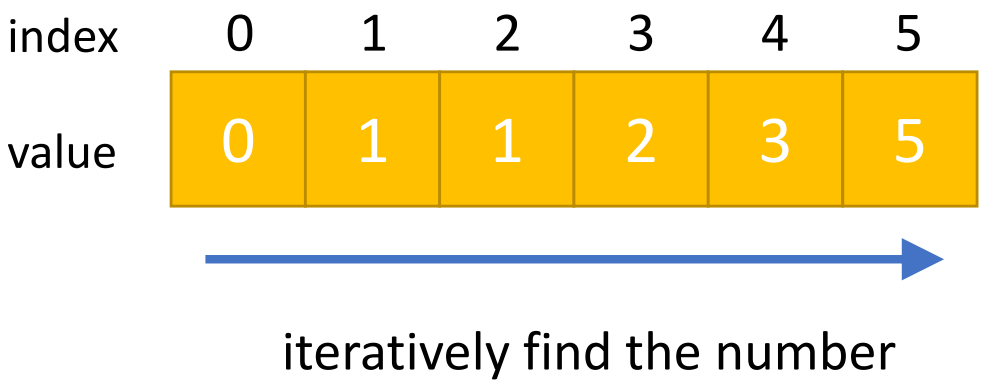
```
int fib(int n, int memo)
{
    int result;
    if (memo[n] != null)
        result = memo[n];
    if(n == 1 || n == 2)
        result 1;
    else
        result fib(n-1) +fib(n);
    memo[n] = result;
    return result;
}
```



Recursively find the number

피보나치 (동적계획법, Bottom up approach)

```
int fib(int n)
{
    int fnum[n+1];
    if(n == 1 || n == 2)
        result 1;
    fnum[1] = fnum[2] = 1;
    for (int i = 3 ; i <= n; i++)
        fnum[i] = fnum[i-1] + fnum[i-2];
    return fnum[n];
}
```



최장 공통 부분 순서

용어 정리: 순서

- 순서(Sequence)
  - 어떤 물건이나 객체의 목록을 가리키는 수학 용어
- 부분 순서(Subsequence)
  - 정렬되어 있는 객체의 목록에서 일부 요소를 제거한 것
  - A B C D E F G → A D E
- 공통 부분 순서
  - 두 순서 사이에 공통적으로 존재하는 부분 순서

    A B C D E F G
    A A B E E C G → A B E G
- 최장 공통 부분 순서 (Longest common subsequence)
  - 여러 개의 공통 부분 순서 중에 가장 긴 것

LCS: 최장 공통 부분 순서

- LCS 문제는 최적 부분구조인가?
- 다음과 같은 문자열이 있음

$$X_i =< x_1, x_2, x_3, \ \ldots\ , x_i >,\ \ Y_j =< y_1, y_2, y_3, \ \ldots\ , y_j >$$

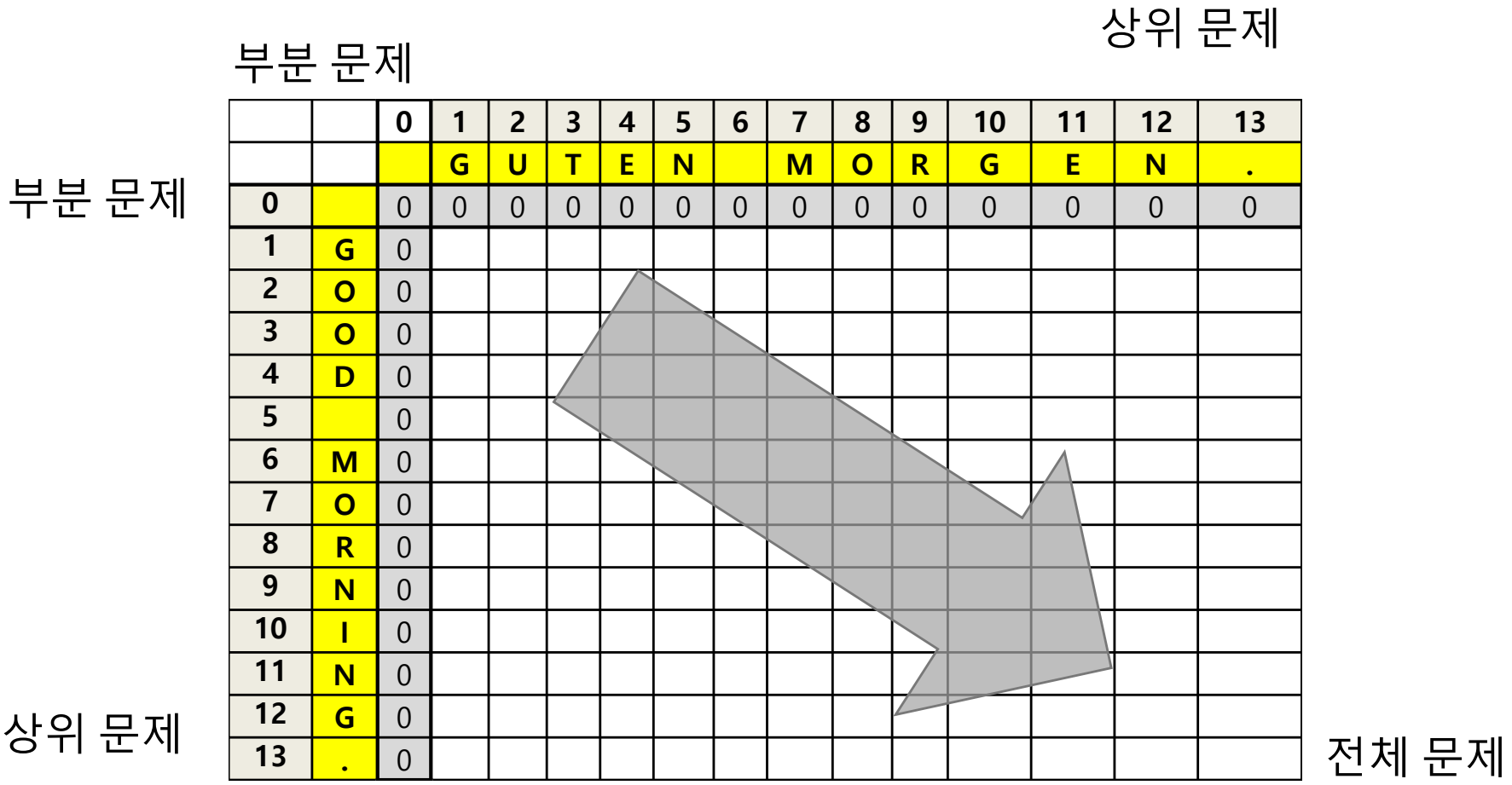
- LCS_Len()은 두 문자열을 매개 변수로 LCS의 길이를 구함

$$LCS_{Len}(X_i, Y_j) = \begin{cases} 0, & i = 0 \text{ or } j = 0 \\ LCS_{Len}(X_{i-1}, Y_{j-1}) + 1, & X_i = Y_j \\ MAX(LCS_{Len}(X_{i-1}, Y_j), LCS_{Len}(X_i, Y_{j-1})), & i,j \geq 0 \text{ and } X_i \neq Y_j \end{cases}$$

- 과거의 계산 결과를 기반으로 문제를 풀 수 있음
  - 동적 계획법 적용 가능

단계 1: 부분 문제로 분할 하기

- 문제 이해와 전체 문제의 답을 구하는 부분 문제 설계가 중요



		0	1	2	3	4	5	6	7	8	9	10	11	12	13
			G	U	T	E	N		M	O	R	G	E	N	.
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0													
2	O	0													
3	O	0													
4	D	0													
5		0													
6	M	0													
7	O	0													
8	R	0													
9	N	0													
10	I	0													
11	N	0													
12	G	0													
13	.	0													

단계 2: 부분 문제 해 구하기

- 부분 문제를 해결, 부분 문제로 상위 문제 해결

		0	1	2	3	4	5	6	7	8	9	10	11	12	13
			G	U	T	E	N		M	O	R	G	E	N	.
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0	1	1	1	1	1	1	1	1	1	1	1	1	1
2	O	0	1	1	1	1	1	1	1	2	2	2	2	2	2
3	O	0	1	1	1	1	1	1	1	2	2	2	2	2	2
4	D	0	1	1	1	1	1	1	1	2	2	2	2	2	2
5		0	1	1	1	1	1	2	2	2	2	2	2	2	2
6	M	0	1	1	1	1	1	2	3	3	3	3	3	3	3
7	O	0	1	1	1	1	1	2	3	4	4	4	4	4	4
8	R	0	1	1	1	1	1	2	3	4	5	5	5	5	5
9	N	0	1	1	1	1	2	2	3	4	5	5	5	6	6
10	I	0	1	1	1	1	2	2	3	4	5	5	5	6	6
11	N	0	1	1	1	1	2	2	3	4	5	5	5	6	6
12	G	0	1	1	1	1	2	2	3	4	5	6	6	6	6
13	.	0	1	1	1	1	2	2	3	4	5	6	6	6	7

단계 3: LCS 추적

- 오른쪽 아래 모서리가 시작 위치
- LCS요소 저장 용 리스트
- 이동 기준
  1. 현재가 좌, 좌상, 상의 위치의 값보다 큰 경우 리스트에 삽입 및 좌상으로 이동
  2. 조건 1이 아니고, 좌측 값과 같고 상보다 큰 경우 좌측 이동
  3. 조건 1, 2가 아닌 경우 상측 이동
  4. i = 0 또는 j = 0이 될 때까지 조건 1부터 반복

단계 3: 예시

		0	1	2	3	4	5	6	7	8	9	10	11	12	13
			G	U	T	E	N		M	O	R	G	E	N	.
0		0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	G	0	↖1	←1	←1	←1	←1	1	1	1	1	1	1	1	1
2	O	0	1	1	1	1	↑1	1	1	2	2	2	2	2	2
3	O	0	1	1	1	1	↑1	1	1	2	2	2	2	2	2
4	D	0	1	1	1	1	↑1	1	1	2	2	2	2	2	2
5		0	1	1	1	1	1	↖2	2	2	2	2	2	2	2
6	M	0	1	1	1	1	1	2	↖3	3	3	3	3	3	3
7	O	0	1	1	1	1	1	2	3	↖4	4	4	4	4	4
8	R	0	1	1	1	1	1	2	3	4	↖5	←5	←5	5	5
9	N	0	1	1	1	1	2	2	3	4	5	5	5	↖6	6
10	I	0	1	1	1	1	2	2	3	4	5	5	5	↑6	6
11	N	0	1	1	1	1	2	2	3	4	5	5	5	↑6	6
12	G	0	1	1	1	1	2	2	3	4	5	6	6	↑6	6
13	.	0	1	1	1	1	2	2	3	4	5	6	6	6	↖7

재귀적 구현

O(2^(m+n))

```
typedef struct _LCSTable
{
    int** data;
} LCSTable;

int LCS(char * X, char* Y, int i, int j, LCSTable* Table)
{
    if (i == 0 || j == 0){
        Table->data[i][j] = 0;
        return Table->data[i][j];
    } else if (X[i-1]] == Y[j-1]) {
        Table->data[i][j] = LCS(X, Y, i-1, j-1, Table)+1;
        return Table->data[i][j]
    } else {
        int A = LCS(X, Y, i-1, j, Table);
        int B = LCS(X, Y, i, j-1, Table);
        if (A > B)
            Table->data[i][j] = A;
        else
            Table->data[i][j] = B;
        return Table->data[i][j];
    }
}
```

동적계획법 구현

```
int LCS(char * X, char* Y, int i, int j, LCSTable* Table)
{
    int m, n;
    for(m = 0, n = 0; m <= i || n <= j; m++, n++)
        Table->data[m][0] = 0;
        Table->data[0][n] = 0;
    for(m = 1; m <=1; m++)
    {
        for(n = 1; n <= j; n++)
        {
            if(X[m-1] == Y[n-1])
                Table->data[m][n] = Table->data[m-1][n-1]+1;
            else
            {
                if(Table->data[m][n-1] >= Table->data[m-1][n])
                    Table->data[m][n] = Table->data[m][n-1];
                else
                    Table->data[m][n] = Table->data[m-1][n];
            }
        }
    }
    return Table->data[i][j];
}
```

O(2^(mn)

추적 구현

```
void trace(char* X, char* Y, int m, int n, LCSTable* Table, char* LCS)
{
    if(m == 0 || n == 0) return; // 종료 조건
    if(Table->data[m][n] > Table->data[m][n-1]
       && Table->data[m][n] > Table->data[m-1][n]
       && Table->data[m][n] > Table->data[m-1][n-1])
    { // 좌, 좌상, 상 보다 큰 경우 좌상 이동
        char temp[100];
        strcpy(temp, LCS);
        sprintf(LCS, "%c%s", X[m-1], temp);
        trace(X, Y, m-1, n-1, Table, LCS);
    } // 상보다 크고 좌와 같은 경우 좌 이동
    else if(Table->data[m][n] > Table->data[m-1][n]
              && Table->data[m][n] == Table->data[m][n-1])
    {
        trace(X, Y, m, n-1, Table, LCS);
    }
    else //그 외, 위로 이동
    {
        trace(X, Y, m-1, n, Table, LCS);
    }
}
```