

# 컴퓨터알고리즘과실습 실습11

2020112046 이재원

## 11-1. 실행 결과

```
> cd "/Users/jaewonlee/Desktop/PS/" && g++ -std=c++2a -g tempCodeRunnerFile.cpp -o a.out && "/Users/jaewonlee/Desktop/PS/"a.out
피보나치 20번째 수 (top-down) : 10946
피보나치 10000번째 수 (bottom-up) : 336447648764317832666216120051075433103021484606800639065647699746800814421666623681555955136337340255820653326
80836159373734790483865268263040892463056431887354544369559827491606602998841839338646527313008883026923567361313511757929743785441375213052050434
770160226475831890652789085515436615958298727968298751063120057542878345321551510387081829896979161312785626503319548714021428753269818796246936097
879900350962302291263681314931952756302278376284415403605844025721143349611800239120828746088923962328835461505776583271252546093591128203925285393
434620904245248929403901706233888991085841065183173360437470737908552631764325733993712871937587746897479926305837065742830161637408969178426378624
212835258112820516370298089332099905707920064367426202389783111470054074998459250360633560933883831923386783561364353518921332797329081337326426526
339897639227234078829281779535805709936914917547080893184156146322338217465637321248226383092103297701648054726243842374862411453938122065649140327
510866433945175121615265453613331113140424368548051067658434935238369596534280717687753283482343455573667197313927462736291082106792807847180353291
811767789246598993863545932789452377767440619224033763867440213303432974969020283281459334188268176838930720036347956231171031012919531697946076327
975892535307725523759437884345040677155557795645044316640119462580972216729758615026968443146952346149322911059706762432685159928347098912847067408
528587135162603127190317286094081298321581772820763531866246112782455372085323653057759564300725177443155153960090516860322034916322264088524885243
315805153484962243484829938090507048348244932745373262456775587989187190803662058959474315005240253270974699531877072437682590741993963226598414749
81936092852394503970716544315642132815768890805878318340491743455627052022356484649519611246026831397097506938264870661326450766574611512677522748
521598642530711298441182622661057163515069260298617049454250474913781151541399415506712562711971332527636319396069028956502882686083622410820505624
8070179497617112123306673310059947366875
```

20번째 피보나치 수는 top-down식의 재귀함수로,

10000번째 피보나치 수는 bottom-up 방식의 반복문으로 구현했다.

10000번째 피보나치 수는 굉장히 커서 원시 자료형에 담기지 않기 때문에, int형의 배열을 이용하여 담았다.

## 코드

```
#include <algorithm>
#include <iostream>
#include <string>
#include <vector>

using namespace std;

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

int dp[300][1000000] = {
    0,
};

void bottom_up_fib(int n) {
    dp[0][0] = 1;
    dp[0][1] = 1;

    for (int i = 2; i <= n - 1; i++) {
        // big number addition with int array
    }
}
```

```

    int carry = 0;
    for (int j = 0; j < 300; j++) {
        dp[j][i] = dp[j][i - 1] + dp[j][i - 2] + carry;
        if (dp[j][i - 1] == 0) break; // if previous number is 0, break
        carry = dp[j][i] / 1000000000; // int형의 범위가 2,147,483,647이므로
        // 1,000,000,000으로 나눠준다. (10^9 자리까지만 사용)
        dp[j][i] -= carry * 1000000000;
    }
}

int main() {
    cout << "피보나치 20번째 수 (top-down) : " << fib(20) << endl;
    int N = 10000;
    bottom_up_fib(N);
    // print dp[9999]
    cout << "피보나치 10000번째 수 (bottom-up) : ";
    for (int i = 299; i >= 0; i--) {
        if (dp[i][N - 1] != 0) cout << dp[i][N - 1];
    }
}

```

## 11-2. 실행 결과

```

----- 4x4 예시 행렬에 대한 경로 검색 -----
40
6 13 25 30
11 14 25 43
18 31 28 31
26 36 42 40
----- 랜덤 200x200 행렬에 대한 경로 검색 -----
bottom-up 방식 : 21352
bottom-up 방식 연산 횟수 : 39800
top-down 방식 : 21352
top-down 방식 함수 호출 횟수 : 79999

```

4x4 행렬과 랜덤 200x200행렬에 대해 top-down, bottom-up 방식으로 경로 탐색을 해 보았다.

재귀로 구현한 Top-down 방식의 함수 호출 횟수만 해도 bottom-up 방식의 연산 횟수보다 큰 것을 확인할 수 있는데, 연산 횟수로 비교할 것을 생각하면, top-down 방식이 확연히 느리다는 것을 확인할 수 있다.

## 코드

```

#include <algorithm>
#include <iostream>
#include <random>
#include <vector>

```

```
using namespace std;
```

```
#define N 200
```

```
int matrix[4][4] = {  
    {6, 7, 12, 5},  
    {5, 3, 11, 18},  
    {7, 17, 3, 3},  
    {8, 10, 14, 9},  
};  
int minMatrix[4][4];
```

```
int findMinRoute_exampleMatrix() {  
    minMatrix[0][0] = matrix[0][0];  
    for (int i = 1; i < 4; i++) {  
        minMatrix[0][i] = minMatrix[0][i - 1] + matrix[0][i];  
        minMatrix[i][0] = minMatrix[i - 1][0] + matrix[i][0];  
    }  
    for (int i = 1; i < 4; i++) {  
        for (int j = 1; j < 4; j++) {  
            minMatrix[i][j] = min(minMatrix[i - 1][j], minMatrix[i][j - 1]) +  
matrix[i][j];  
        }  
    }  
    return minMatrix[3][3];  
}
```

```
int BottomUpCount = 0;  
int TopDownCount = 0;
```

```
int minRandomMatrix[N][N];  
int findMinRoute_randomMatrix_bottomUp(int randomMatrix[N][N]) {  
    minRandomMatrix[0][0] = randomMatrix[0][0];  
    for (int i = 1; i < N; i++) {  
        BottomUpCount++;  
        minRandomMatrix[0][i] = minRandomMatrix[0][i - 1] + randomMatrix[0][i];  
        minRandomMatrix[i][0] = minRandomMatrix[i - 1][0] + randomMatrix[i][0];  
    }  
    for (int i = 1; i < N; i++) {  
        for (int j = 1; j < N; j++) {  
            BottomUpCount++;  
            minRandomMatrix[i][j] = min(minRandomMatrix[i - 1][j], minRandomMatrix[i][j  
- 1]) + randomMatrix[i][j];  
        }  
    }  
    return minRandomMatrix[199][199];  
}
```

```
int minRandomMatrix2[N][N];  
int findMinRoute_randomMatrix_topDown(int randomMatrix[N][N], int i, int j) {  
    TopDownCount++;  
    if (i == 0 && j == 0) {  
        return randomMatrix[0][0];  
    }  
}
```

```

    }
    if (i < 0 || j < 0) {
        return 10000;
    }
    if (minRandomMatrix2[i][j] != 0) {
        return minRandomMatrix2[i][j];
    }

    minRandomMatrix2[i][j] = min(findMinRoute_randomMatrix_topDown(randomMatrix, i - 1,
j), findMinRoute_randomMatrix_topDown(randomMatrix, i, j - 1)) + randomMatrix[i][j];
    return minRandomMatrix2[i][j];
}

int main() {
    int minRoute = findMinRoute_exampleMatrix();
    cout << "----- 4x4 예시 행렬에 대한 경로 검색 -----" << endl;
    cout << minRoute << endl;
    // print minMatrix
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            cout << minMatrix[i][j] << " ";
        }
        cout << endl;
    }

    cout << "----- 랜덤 200x200 행렬에 대한 경로 검색 -----" << endl;
    // random 200x200 matrix (1 ~ 200)
    random_device rd;
    int randomMatrix[N][N];
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            randomMatrix[i][j] = rd() % N;
        }
    }

    minRoute = findMinRoute_randomMatrix_bottomUp(randomMatrix);
    cout << "bottom-up 방식: " << minRoute << endl;
    cout << "bottom-up 방식 연산 횟수: " << BottomUpCount << endl;
    minRoute = findMinRoute_randomMatrix_topDown(randomMatrix, N - 1, N - 1);
    cout << "top-down 방식: " << minRoute << endl;
    cout << "top-down 방식 함수 호출 횟수: " << TopDownCount << endl;
}

```