2022-1 HI-ARC 초급스터디

5주차. DP

이지은 (leeju1013)

목차

1. DP
2. 시간 복잡도

피보나치 수

성공

브론즈 III

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초 (추가 시간 없음)	128 MB	50788	23816	19438	48.230%

문제

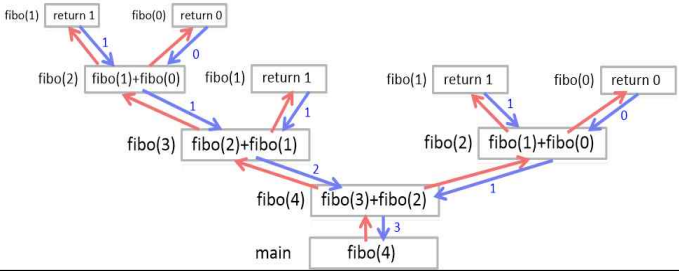
피보나치 수는 0과 1로 시작한다. 0번째 피보나치 수는 0이고, 1번째 피보나치 수는 1이다. 그 다음 2번째 부터는 바로 앞 두 피보나치 수의 합이 된다.

이를 식으로 써보면 $F_n = F_{n-1} + F_{n-2}$ ($n \geq 2$)가 된다.

n=17일때 까지 피보나치 수를 써보면 다음과 같다.

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597

n이 주어졌을 때, n번째 피보나치 수를 구하는 프로그램을 작성하시오.



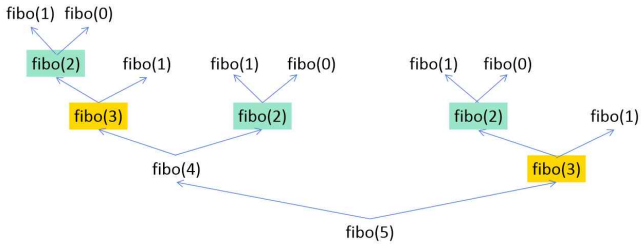
입력

첫째 줄에 n이 주어진다. n은 45보다 작거나 같은 자연수이다.

출력

첫째 줄에 n번째 피보나치 수를 출력한다.

```cpp
#include <iostream>
using namespace std;

int fibo(int n){
    if(n==0) return 0;
    else if(n==1) return 1;
    else return fibo(n-1)+fibo(n-2);
}
int main(){
    ios_base::sync_with_stdio(false);

    int n; cin>>n;
    cout<<fibo(n);
    return 0;
}
```

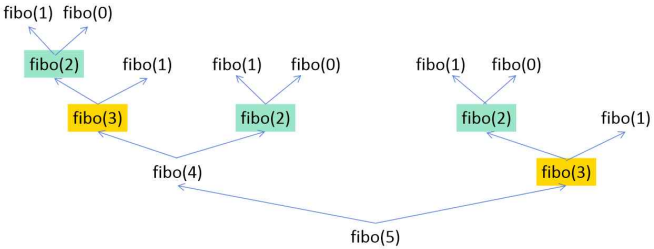


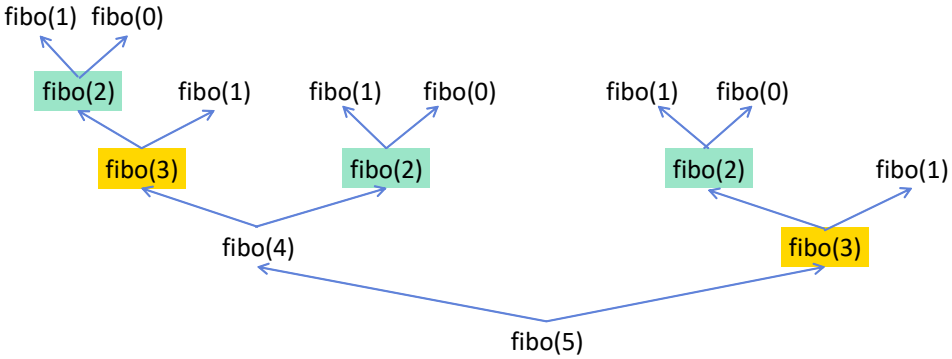
제출 번호	아이디	문제	문제 제목	결과	메모리	시간	언어	코드 길이
40355570	leeju1013	2747	피보나치 수	시간 초과			C++14	362 B

```cpp
#include <iostream>
using namespace std;
int dp[46];
int fibo(int n){
    if(n==0) return 0; //또는 return dp[0];
    if(n==1) return 1; //또는 return dp[1] ;
    if(dp[n]!=-1) return dp[n]; //계산 한 적이 있다면
    return dp[n] = fibo(n-1)+fibo(n-2);
}
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int n; cin>>n;
    fill(dp, dp+n+1, -1); //배열 dp를 -1로 초기화
    dp[0]=0, dp[1]=1;

    cout<<fibo(n);
    return 0;
}
```

idx	0	1	2	3	4	5
dp	0	1	1	2	3	5

idx	0	1	2	3	4	5
dp	0	1	-1	-1	-1	-1




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

```cpp
#include <iostream>
using namespace std;
int dp[46];
int fibo(int n){
    if(n==0) return 0;
    if(n==1) return 1;
    if(dp[n]) return dp[n];
    return dp[n] = fibo(n-1)+fibo(n-2);
}
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int n; cin>>n;
    cout<<fibo(n);
    return 0;
}
```

탑다운(top-down)

: 재귀 호출 사용

```cpp
#include <iostream>
using namespace std;
int dp[46];
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int n; cin>>n;
    dp[0]=0, dp[1]=1;
    for(int i=2; i<=n; i++){
        dp[i] = dp[i-1] + dp[i-2];
    }
    cout<< dp[n];
    return 0;
}
```

바텀업(bottom-up)

: 반복문 사용

1. DP (Dynamic Programming)

- What?
  - 전체 문제를 작은 부분 문제로 나누어 푸는 방법
  - 중복되는 부분 문제는 한 번만 계산

- How?
  - 메모이제이션
  - 탑다운 / 바텀업

1. DP

```cpp
#include <iostream>
using namespace std;
int dp[46];
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int n; cin>>n;
    dp[0]=0, dp[1]=1;
    for(int i=2; i<=n; i++){
        dp[i] = dp[i-1] + dp[i-2];
    }
    cout<< dp[n];
    return 0;
}
```

// DP 테이블 정의. dp[i] = i번째 피보나치 수

// 초기값 설정

// 점화식 찾기★

1로 만들기

실버 III

시간 제한	메모리 제한	제출	정답	맞힌 사람
0.15 초 (하단 참고)	128 MB	190868	61699	39242

문제

정수 X에 사용할 수 있는 연산은 다음과 같이 세 가지 이다.

1. X가 3으로 나누어 떨어지면, 3으로 나눈다.
2. X가 2로 나누어 떨어지면, 2로 나눈다.
3. 1을 뺀다.

정수 N이 주어졌을 때, 위와 같은 연산 세 개를 적절히 사용해서 1을 만들려고 한다. 연산을 사용하는 횟수의 최솟값을 출력하시오.

입력

첫째 줄에 1보다 크거나 같고, 10^6보다 작거나 같은 정수 N이 주어진다.

출력

첫째 줄에 연산을 하는 횟수의 최솟값을 출력한다.

예제 입력 1 복사

```
2
```

예제 출력 1 복사

```
1
```

예제 입력 2 복사

```
10
```

예제 출력 2 복사

```
3
```

힌트

10의 경우에 10 -> 9 -> 3 -> 1 로 3번 만에 만들 수 있다.

정수 X에 사용할 수 있는 연산은 다음과 같이 세 가지 이다.

1. X가 3으로 나누어 떨어지면, 3으로 나눈다.
2. X가 2로 나누어 떨어지면, 2로 나눈다.
3. 1을 뺀다.

정수 N이 주어졌을 때, 위와 같은 연산 세 개를 적절히 사용해서 1을 만들려고 한다. 연산을 사용하는 횟수의 최솟값을 출력하시오.

1. DP 테이블 정의

   dp[x] = x를 1로 만드는 연산 횟수의 최솟값

2. 점화식 찾기

   dp[x] = min(dp[x/3]+1, dp[x/2]+1, dp[x-1]+1)

   x가 3으로 나누어 떨어질 때. x가 2로 나누어 떨어질 때.

3. 초기값 설정

   dp[1] = 0

예제 입력 2 복사

```
10
```

예제 출력 2 복사

```
3
```

힌트

10의 경우에 10 -> 9 -> 3 -> 1 로 3번 만에 만들 수 있다.

```cpp
#include <iostream>
using namespace std;
int dp[1000001];
int func(int x){
    if(x==1) return 0; //base case : dp[1]=0
    if(dp[x]!=-1) return dp[x]; //이미 계산했던거

    int result = func(x-1)+1;
    if(x%3==0) result = min(result, func(x/3)+1);
    if(x%2==0) result = min(result, func(x/2)+1);
    return dp[x]=result;
}
int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);

    int n; cin>>n;
    fill(dp, dp+n+1, -1);
    cout<<func(n);

    return 0;
}
```

탑다운(top-down)
: 재귀 호출 사용

1. DP 테이블 정의

   dp[x] = x를 1로 만드는 연산 횟수의 최솟값

2. 점화식 찾기

   dp[x] = min(dp[x/3]+1, dp[x/2]+1, dp[x-1]+1)

   x가 3으로 나누어 떨어질 때. x가 2로 나누어 떨어질 때.

3. 초기값 설정

   dp[1] = 0

```cpp
#include <iostream>
using namespace std;
int dp[1000001];

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);

    int n; cin>>n;
    dp[1]=0;
    for(int i=2; i<=n; i++){
        dp[i]=dp[i-1]+1;
        if(i%3==0) dp[i]=min(dp[i], dp[i/3]+1);
        if(i%2==0) dp[i]=min(dp[i], dp[i/2]+1);
    }
    cout<<dp[n];
    return 0;
}
```

바텀업(bottom-up)
: 반복문 사용

가장 긴 증가하는 부분 수열

성공

실버 II

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
1 초	256 MB	98771	38738	25401	37.177%

문제

수열 A가 주어졌을 때, 가장 긴 증가하는 부분 수열을 구하는 프로그램을 작성하시오.

예를 들어, 수열 A = {10, 20, 10, 30, 20, 50} 인 경우에 가장 긴 증가하는 부분 수열은 A = {**10**, **20**, 10, **30**, 20, **50**} 이고, 길이는 4이다.

입력

첫째 줄에 수열 A의 크기 N (1 ≤ N ≤ 1,000)이 주어진다.

둘째 줄에는 수열 A를 이루고 있는 A_i가 주어진다. (1 ≤ A_i ≤ 1,000)

출력

첫째 줄에 수열 A의 가장 긴 증가하는 부분 수열의 길이를 출력한다.

예제 입력 1 복사

```
6
10 20 10 30 20 50
```

예제 출력 1 복사

```
4
```

수열 A가 주어졌을 때, 가장 긴 증가하는 부분 수열을 구하는 프로그램을 작성하시오.

예를 들어, 수열 A = {10, 20, 10, 30, 20, 50} 인 경우에 가장 긴 증가하는 부분 수열은 A = {**10**, **20**, 10, **30**, 20, **50**} 이고, 길이는 4이다.

* 가장 긴 증가하는 부분 수열 (Longest Increasing Subsequence)
: 부분 수열(연속하지 않아도 됨) 중 오름차순으로 정렬된 가장 긴 수열

1. DP 테이블 정의
   dp[i] = arr[i]로 끝나는 LIS 길이

idx	0	1	2	3	4	5
arr	**10**	**20**	10	**30**	20	**50**
dp	1	2	1	3	2	4

2. 점화식 찾기
   dp[i] = arr[j] (0<=j<i) 중에서 arr[j]<arr[i]인 수들 중 dp[j]의 최댓값 +1

3. 초기값 설정
   dp[0] = 1

```cpp
#include <iostream>
using namespace std;

int arr[1001]; //수열A 저장할 배열 선언
int dp[1001]; //dp[i]는 arr[i]로 끝나는 LIS길이

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    //변수 n 선언, ans 선언, n 입력받기
    //배열 arr 입력받기

    //dp배열 초기값 설정
    for(int i=1; i<n; i++){ //dp배열 바텀업 방식으로 채우기
        //dp[i] 초기화
        for(int j=0; j<i; j++){ //앞에있는 수들 보기
            //arr[j]<arr[i]이고 dp[i]<dp[j]+1이면 dp[i]값 갱신
        }
        //ans에 dp[i] 최대값 갱신
    }
    //ans 출력
    return 0;
}
```

idx	0	1	2	3	4	5
arr	**10**	**20**	10	**30**	20	**50**
dp	1	2	1	3	2	4

```cpp
#include <iostream>
using namespace std;

int arr[1001]; //수열A 저장
int dp[1001]; //dp[i]는 arr[i]로 끝나는 LIS길이

int main() {
    ios_base::sync_with_stdio(0);
    cin.tie(0); cout.tie(0);

    int n,ans=1; cin>>n;
    for(int i=0; i<n; i++) cin>>arr[i];

    dp[0]=1;
    for(int i=1; i<n; i++){
        dp[i]=1;
        for(int j=0; j<i; j++){
            if(arr[j]<arr[i]) dp[i]=max(dp[i],dp[j]+1);
        }
        ans=max(ans,dp[i]);
    }
    cout<<ans;
    return 0;
}
```

2. 시간 복잡도

3 2747번 | 제출 | 맞힌 사람 | 숏코딩 | 재채점

피보나치 수 성공

3 브론즈 III

시간 제한	메모리 제한
1 초 (추가 시간 없음)	128 MB

제출 번호	아이디	문제	문제 제목	결과	메모리	시간	언어	코드 길이
40355570	leeju1013	2747	피보나치 수	시간 초과			C++14	362 B

2. 시간 복잡도 (Time Complexity)

- What?
  - 특정 알고리즘이 문제를 해결하는데 필요한 시간(연산의 횟수)

- How?
  - 빅-오(Big-O) 표기법
    : 알고리즘 최악의 실행시간 표기

2. 빅-오 표기법

```cpp
#include <iostream>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);

    int n, sum=0; cin>>n;
    for(int i=1; i<=n; i++){
        sum += i;
    }
    cout<< sum;
    return 0;
}
```

O(N)

```cpp
#include <iostream>
using namespace std;

int main(){
    ios_base::sync_with_stdio(false);
    cin.tie(NULL); cout.tie(NULL);

    int n, sum=0; cin>>n;
    sum = n*(n+1)/2;
    cout<< sum;
    return 0;
}
```

O(1)

O(1)

```
int n, ans=0, cin>>n;
ans = 3*n;
cout<< ans;
```

O(logN)

```
bool binary_search(int *arr, int len, int key){
    int l = 0, r = len-1, mid;
    while(l <= r) {
        mid = (l + r) / 2;
        if (arr[mid] == key) return true;
        else if (arr[mid] > key) r = mid - 1;
        else l = mid + 1;
    }
    return false;
}
```

O(N)

```
int n, ans=0, cin>>n;
for(int i=0; i<n; i++){
    ans += 3*n;
}
cout<< ans;
```

O(NlogN)

```
sort(arr, arr+n);
```

$O(N^2)$

```
int n, ans=0, cin>>n;
for(int i=1; i<=n; i++){
    for(int j=1; j<=n; j++){
        ans += i*j;
    }
}
cout<< ans;
```

$O(2^N)$

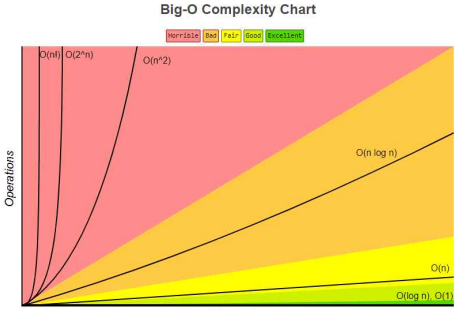
```
int fibo(int n){
    if(n <=1) return n;
    return fibo(n-2) + fibo(n-1);
}
```

2. 빅-오 표기법

- $O(1) < O(\log N) < O(N) < O(N \log N) < O(N^2) < O(2^N) < O(N!)$

  Fast <-- --> Slow

- O(3) -> O(1)
- O(100N + 3) -> O(N)
- $O(N^2 + 2N + 7)$ -> $O(N^2)$
- $O(2^N + 100N^2)$ -> $O(2^N)$
- $O(N! + 2^N + 1000)$ -> O(N!)



https://www.bigocheats

2747번 제출

피보나치 수

브론즈 III

시간 제한

1 초 (추가 시간 없음)

입력

첫째 줄에 n이 주어진다. n은 45보다 작거나 같은 자연수이다.

```cpp
#include <iostream>
using namespace std;

int fibo(int n){
    if(n==0) return 0;
    else if(n==1) return 1;
    else return fibo(n-1)+fibo(n-2);
}
int main(){
    ios_base::sync_with_stdio(false);

    int n; cin>>n;
    cout<<fibo(n);
    return 0;
}
```

대략 $O(2^N)$

```cpp
#include <iostream>
using namespace std;
int dp[46];
int main(){
    ios_base::sync_with_stdio(false); cin.tie(NULL);

    int n; cin>>n;
    dp[0]=0, dp[1]=1;
    for(int i=2; i<=n; i++){
        dp[i] = dp[i-1] + dp[i-2];
    }
    cout<< dp[n];
    return 0;
}
```

$O(N)$

감사합니다

- 필수 문제
  - 2747번 피보나치 수
  - 1463번 1로 만들기
  - 11053번 가장 긴 증가하는 부분 수열
- 연습 문제
  - 2748번 피보나치 수 2
  - 11726번 2×n 타일링
  - 13699번 점화식
  - 1912번 연속합
  - 9461번 파도반 수열
- 5월 10일(화요일) 저녁 8시