2021 신촌 연합 여름캠프 초급반 5회차

동적 계획법 (DP)

서강대학교 박재형

동적 계획법이란?

- Dynamic Programming (DP)
- 큰 문제를 작은 부분 문제들로 나누어 푸는 방법
- 점화식 세우기
- 메모이제이션 (Memoization) 기법 사용

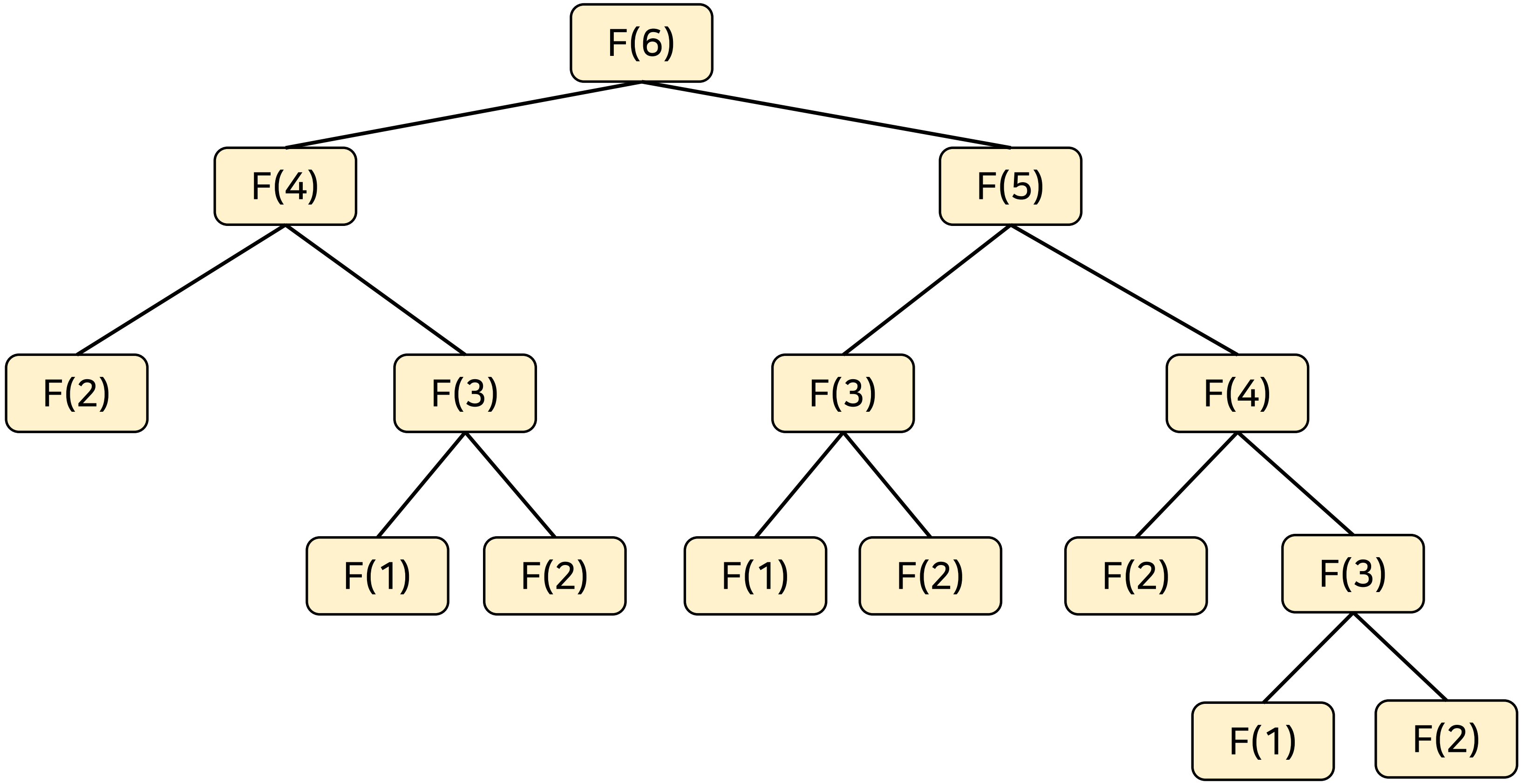
피보나치 수열

[1, 1, 2, 3, 5, 8, 13, 21, 34, …]

$F(n) = F(n-1) + F(n-2)$

```
int fibo(int n) {
    if (n <= 0) return 0;
    else if (n == 1) return 1;
    return fibo(n - 1) + fibo(n - 2);
}
```

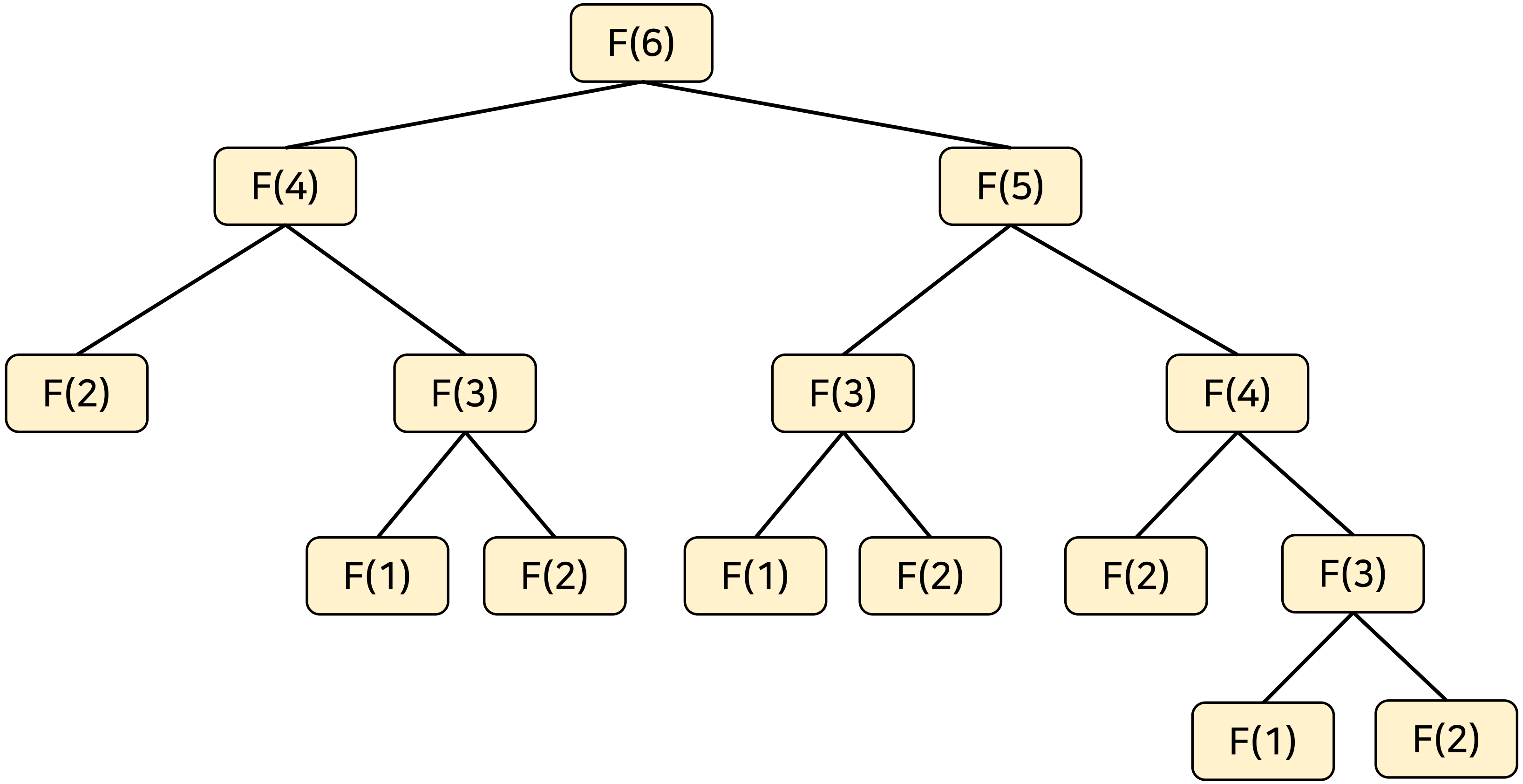
피보나치 수열



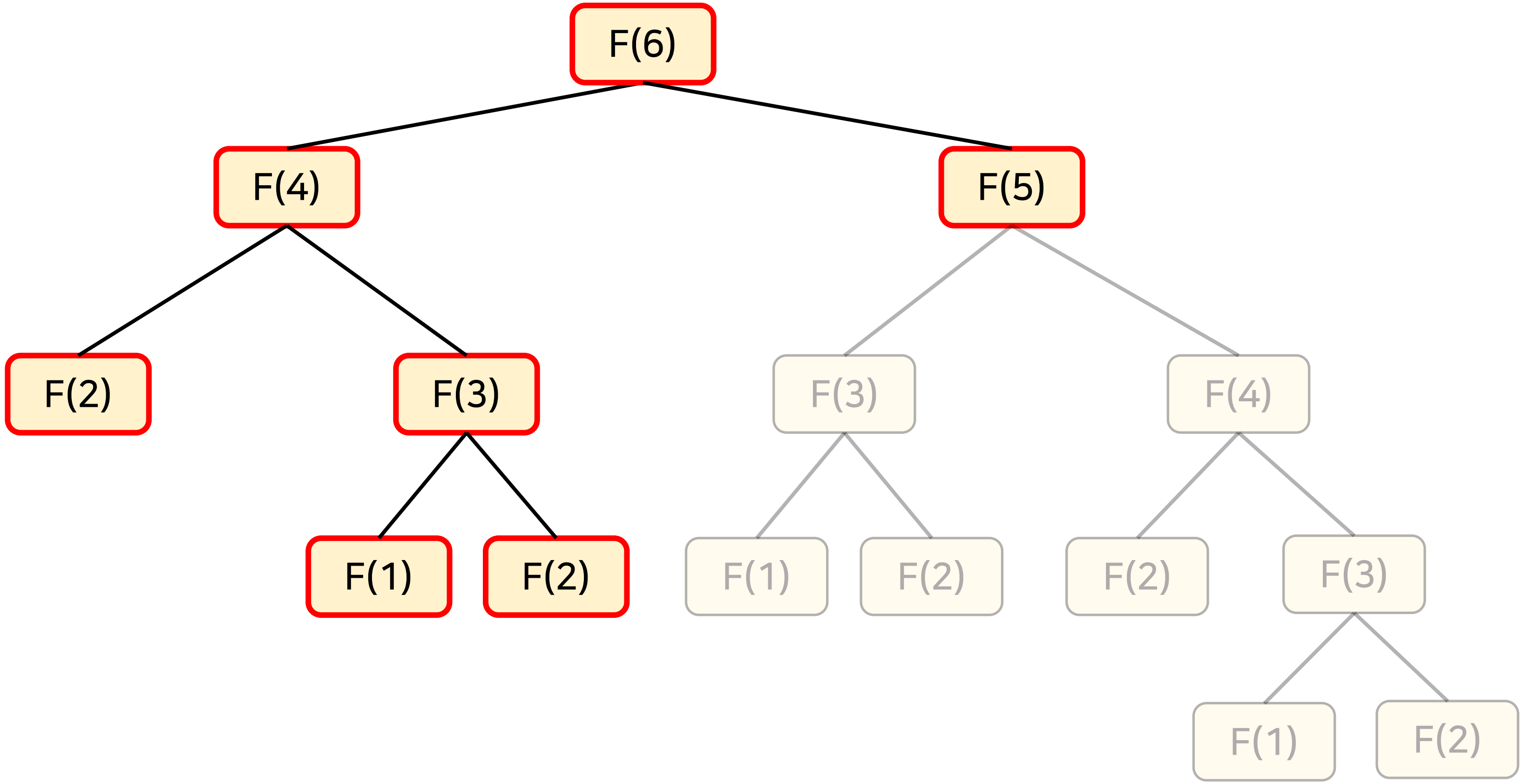
메모이제이션

- Memoization
- 부분문제를 계산한 결과를 메모리에 저장
- 동일한 계산을 할 때 이전에 메모리에 저장한 값을 이용
- 배열 사용

피보나치 수열



피보나치 수열



피보나치 수열

- 초기화

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

int dp[10001];
int main(void) {
    for (int i = 0; i <= 10000; i++) dp[i] = -1;

    memset(dp, -1, sizeof(dp)); //cstring

    fill(dp, dp + 10001, -1); //std::fill
}
```

피보나치 수열

- 코드

```
int fibo(int n) {
    if (dp[n] != -1) return dp[n];
    if (n <= 1) { //base case
        dp[n] = n;
        return dp[n];
    }
    else {
        dp[n] = fibo(n - 1) + fibo(n - 2);
        return dp[n];
    }
}
```

DP 주의할 점

- 순환 구조가 존재하지 않아야 한다

- Base Case 설정

- 최적 부분 구조

  (기본 문제의 최적해가 부분 문제의 최적해를 포함)

Top-Down vs Bottom-Up

<Top-Down>

- 재귀 호출 이용

- 큰 문제에서 필요한 부분 문제들을 호출해나가는 방식

- 상대적으로 DP식을 이해하기 쉽다

<Bottom-Up>

- 반복문 이용

- 제일 작은 부분문제부터 답을 구해가는 방식

- 상대적으로 메모리/시간이 작다

Top-Down vs Bottom-Up

<Top-Down>

```
int fibo(int n) {
    if (dp[n] != -1) return dp[n];
    if (n <= 1) { //base case
        dp[n] = n;
        return dp[n];
    }
    else {
        dp[n] = fibo(n - 1) + fibo(n - 2);
        return dp[n];
    }
}
```

<Bottom-Up>

```
dp[1] = dp[2] = 1;
for (int i = 3; i <= 10000; i++) {
    dp[i] = dp[i - 1] + dp[i - 2];
}
```

```
dp[1] = 1;
for (int i = 1; i <= 10000; i++) {
    dp[i + 1] += dp[i];
    dp[i + 2] += dp[i];
}
```

시간 복잡도

- Bottom-Up인 경우 : 반복문의 연산 횟수

- Top-Down인 경우는?

DP Table의 크기 x 한 칸을 계산하는데 걸리는 시간

ex)

```
int dp[1001];
int sol(int x) {
    if (dp[x] != -1) return dp[x];
    if (x == 1000) return 0;
    for (int i = x + 1; i <= 1000; i++) {
        dp[x] = max(dp[x], sol(i) + i);
    }
    return dp[x];
}
```

DP Tips

- 참조형 변수 (Reference variable)

[자료형]& [참조 변수명] = [변수명]

예시)

```
int dp[10][10][10];
int sol(int a, int b, int c) {
    if (dp[a][b][c] != -1) return dp[a][b][c];
    dp[a][b][c] = max(dp[a][b][c], sol(a + 1, b, c));
    dp[a][b][c] = max(dp[a][b][c], sol(a, b + 1, c));
    dp[a][b][c] = max(dp[a][b][c], sol(a, b, c + 1));
    return dp[a][b][c];
}
```

```
int dp[10][10][10];
int sol(int a, int b, int c) {
    int& ret = dp[a][b][c];
    if (ret != -1) return ret;
    ret = max(ret, sol(a + 1, b, c));
    ret = max(ret, sol(a, b + 1, c));
    ret = max(ret, sol(a, b, c + 1));
    return ret;
}
```

DP Tips

< When use DP? > (예시)

- Naive 시간복잡도

• O(N!) (ex 일렬로 나열하기)

• O(2^N) (ex 모든 부분집합 고려)

- 최대/최소, 경우의 수

- "~~ 1000000007로 나눈 나머지를 구하시오"

DP Tips

< How to use DP? >

1. Naive하게 계산 → 중복이 발생하는가?

2. DP로 해결 가능한 문제인지 파악 (ex 부분 문제로 나뉘어지는가?)

3. DP table로 상태(status) 표현

4. 상태(status)간의 관계식 구하기 (점화식 구하기)

5. Top-down or Bottom-up / 기저 사례(Base case) 설정

BOJ 1463 (1로 만들기)

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.

BOJ 1463 (1로 만들기)

- ✓ Naive하게 계산 → 중복이 발생하는가?
- ✓ 최적 부분 구조를 만족하는가?
- ✓ 상태(status) 표현
- ✓ 점화식
- ✓ 기저 사례 (Base case)

BOJ 1463 (1로 만들기)

[Top-Down]

메모리	시간
21432 KB	20 ms

```cpp
#include<iostream>
using namespace std;
int dp[1000001];
int sol(int x) {
    if (x == 1) return 0;
    int& ret = dp[x];
    if (ret != -1) return ret;
    ret = 10000000;
    if (x % 3 == 0) ret = min(ret, sol(x / 3) + 1);
    if (x % 2 == 0) ret = min(ret, sol(x / 2) + 1);
    ret = min(ret, sol(x - 1) + 1);
    return ret;
}
int main(void) {
    int n; cin >> n;
    fill(dp, dp + n + 1, -1);
    cout << sol(n);
}
```

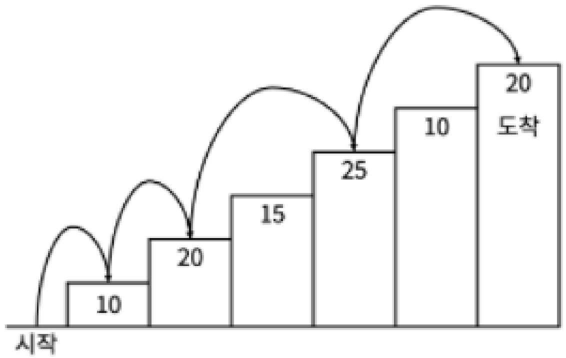
[Bottom-Up]

메모리	시간
5928 KB	4 ms

```cpp
#include<iostream>
using namespace std;
int dp[1000001];
int main(void) {
    int n; cin >> n;
    fill(dp, dp + n + 1, 10000000);
    dp[1] = 0;
    for (int i = 1; i <= n; i++) {
        if (i * 3 <= n) dp[i * 3] = min(dp[i * 3], dp[i] + 1);
        if (i * 2 <= n) dp[i * 2] = min(dp[i * 2], dp[i] + 1);
        if (i + 1 <= n) dp[i + 1] = min(dp[i + 1], dp[i] + 1);
    }
    cout << dp[n];
}
```

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

BOJ 2579 (계단 오르기)



<그림 2>

계단 오르는 데는 다음과 같은 규칙이 있다.

1. 계단은 한 번에 한 계단씩 또는 두 계단씩 오를 수 있다. 즉, 한 계단을 밟으면서 이어서 다음 계단이나, 다음 다음 계단으로 오를 수 있다.
2. 연속된 세 개의 계단을 모두 밟아서는 안 된다. 단, 시작점은 계단에 포함되지 않는다.
3. 마지막 도착 계단은 반드시 밟아야 한다.

따라서 첫 번째 계단을 밟고 이어 두 번째 계단이나, 세 번째 계단으로 오를 수 있다. 하지만, 첫 번째 계단을 밟고 이어 네 번째 계단으로 올라가거나, 첫 번째, 두 번째, 세 번째 계단을 연속해서 모두 밟을 수는 없다.

각 계단에 쓰여 있는 점수가 주어질 때 이 게임에서 얻을 수 있는 총 점수의 최댓값을 구하는 프로그램을 작성하시오.

BOJ 2579 (계단 오르기)

- ✓ Naive하게 계산 → 중복이 발생하는가?
- ✓ 최적 부분 구조를 만족하는가?
- ✓ 상태(status) 표현
- ✓ 점화식
- ✓ 기저 사례 (Base case)

BOJ 2579 (계단 오르기)

[Top-Down]

```cpp
#include<iostream>
#include<string.h>
using namespace std;

int dp[301][3];
int score[301], n;
int sol(int x, int cnt) {
    if (x == n) return 0;
    int& ret = dp[x][cnt];
    if (ret != -1) return ret;
    ret = -100000000;
    if (cnt < 2) ret = max(ret, sol(x + 1, cnt + 1) + score[x + 1]);
    if (x < n - 1) ret = max(ret, sol(x + 2, 1) + score[x + 2]);
    return ret;
}
int main(void) {
    cin >> n;
    for (int i = 1; i <= n; i++) cin >> score[i];
    memset(dp, -1, sizeof(dp));
    cout << sol(0, 0);
}
```

[Bottom-Up]

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

int dp[301][3];
int score[301];
int main(void) {
    int n; cin >> n;
    for (int i = 1; i <= n; i++) cin >> score[i];
    dp[1][1] = score[1];
    for (int i = 2; i <= n; i++) {
        dp[i][1] = max(dp[i - 2][1], dp[i - 2][2]) + score[i];
        dp[i][2] = dp[i - 1][1] + score[i];
    }
    cout << max(dp[n][1], dp[n][2]);
}
```

BOJ 1149 (RGB거리)

문제

RGB거리에는 집이 N개 있다. 거리는 선분으로 나타낼 수 있고, 1번 집부터 N번 집이 순서대로 있다.

집은 빨강, 초록, 파랑 중 하나의 색으로 칠해야 한다. 각각의 집을 빨강, 초록, 파랑으로 칠하는 비용이 주어졌을 때, 아래 규칙을 만족하면서 모든 집을 칠하는 비용의 최솟값을 구해보자.

- 1번 집의 색은 2번 집의 색과 같지 않아야 한다.
- N번 집의 색은 N-1번 집의 색과 같지 않아야 한다.
- i(2 ≤ i ≤ N-1)번 집의 색은 i-1번, i+1번 집의 색과 같지 않아야 한다.

입력

첫째 줄에 집의 수 N(2 ≤ N ≤ 1,000)이 주어진다. 둘째 줄부터 N개의 줄에는 각 집을 빨강, 초록, 파랑으로 칠하는 비용이 1번 집부터 한 줄에 하나씩 주어진다. 집을 칠하는 비용은 1,000보다 작거나 같은 자연수이다.

BOJ 1149 (RGB거리)

- ✓ Naive하게 계산 → 중복이 발생하는가?
- ✓ 최적 부분 구조를 만족하는가?
- ✓ 상태(status) 표현
- ✓ 점화식
- ✓ 기저 사례 (Base case)

BOJ 11053 (가장 긴 증가하는 부분 수열)

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 ($1 \leq N \leq 1{,}000$)이 주어진다.

둘째 줄에는 수열 A를 이루고 있는 A_i가 주어진다. ($1 \leq A_i \leq 1{,}000$)

BOJ 11053 (가장 긴 증가하는 부분 수열)

- Longest Increasing Subsequence (LIS)

- $O(N^2)$ / $O(N\log N)$

10	20	10	30	20	50

BOJ 11053 (가장 긴 증가하는 부분 수열)

- ✓ Naive하게 계산 → 중복이 발생하는가?
- ✓ 최적 부분 구조를 만족하는가? X → 시작점 or 끝점 고정
- ✓ 상태(status) 표현
- ✓ 점화식
- ✓ 기저 사례 (Base case)

누적 합(Prefix Sum)

- i번째 원소부터 j번째 원소까지의 합

- 위 과정을 10만번 한다면?

- psum[i] := 1번째 원소부터 i번째 원소까지의 합

- psum[i] = psum[i-1] + A[i]

- A[i] + A[i+1] + ⋯ + A[j] = psum[j] – psum[i-1]

BOJ 21318 (피아노 체조)

문제

피아노를 사랑하는 시은이는 매일 아침 피아노 체조를 한다. 시은이는 N개의 악보를 가지고 있으며, 1번부터 N번까지의 번호로 부른다. 각 악보는 1 이상 10^9 이하의 정수로 표현되는 난이도를 가지고 있다. 난이도를 나타내는 수가 클수록 어려운 악보이다. $1 \le x \le y \le N$ 을 만족하는 두 정수 x, y를 골라 x번부터 y번까지의 악보를 번호 순서대로 연주하는 것이 피아노 체조이다.

시은이는 피아노 체조를 할 때, 지금 연주하는 악보가 바로 다음에 연주할 악보보다 어렵다면 실수를 한다. 다시 말하자면, $i(x \le i < y)$번 악보의 난이도가 $i + 1$번 악보의 난이도보다 높다면 실수를 한다. 특히, 마지막으로 연주하는 y번 악보에선 절대 실수하지 않는다. 시은이는 오늘도 피아노 체조를 하기 위해 두 정수 x와 y를 골랐고, 문득 궁금한 것이 생겼다. 오늘 할 피아노 체조에서 실수하는 곡은 몇 개나 될까?

입력

첫 번째 줄에 악보의 개수 $N(1 \le N \le 100{,}000)$이 주어진다.

두 번째 줄에 1번 악보부터 N번 악보까지의 난이도가 공백을 구분으로 주어진다.

세 번째 줄에 질문의 개수 $Q(1 \le Q \le 100{,}000)$이 주어진다.

다음 Q개의 줄에 각 줄마다 두 개의 정수 $x, y(1 \le x \le y \le N)$가 주어진다.

BOJ 21318 (피아노 체조)

- i번 악보 난이도 > i+1번 악보 난이도

- x번 악보 ~ y번 악보를 연주

- Q = 100,000

ex) 1 2 3 3 4 1 10 8 1

BOJ 11660 (구간 합 구하기 5)

문제

N×N개의 수가 N×N 크기의 표에 채워져 있다. (x1, y1)부터 (x2, y2)까지 합을 구하는 프로그램을 작성하시오. (x, y)는 x행 y열을 의미한다.

예를 들어, N = 4이고, 표가 아래와 같이 채워져 있는 경우를 살펴보자.

1	2	3	4
2	3	4	5
3	4	5	6
4	5	6	7

여기서 (2, 2)부터 (3, 4)까지 합을 구하면 3+4+5+4+5+6 = 27이고, (4, 4)부터 (4, 4)까지 합을 구하면 7이다.

표에 채워져 있는 수와 합을 구하는 연산이 주어졌을 때, 이를 처리하는 프로그램을 작성하시오.

입력

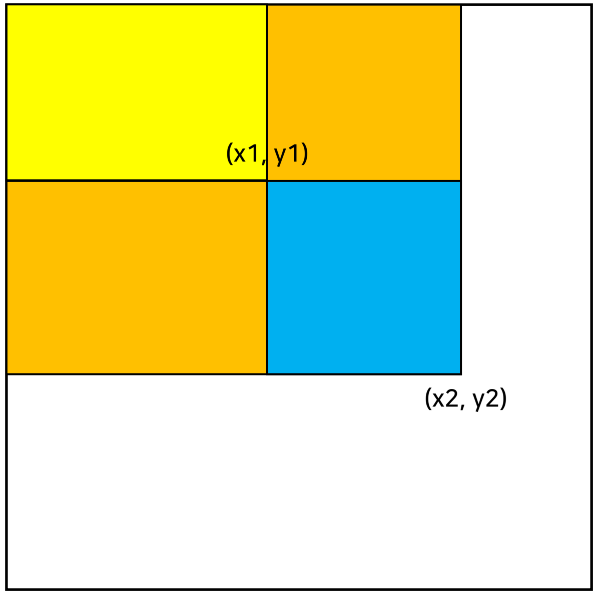
첫째 줄에 표의 크기 N과 합을 구해야 하는 횟수 M이 주어진다. ($1 \le N \le 1024$, $1 \le M \le 100,000$) 둘째 줄부터 N개의 줄에는 표에 채워져 있는 수가 1행부터 차례대로 주어진다. 다음 M개의 줄에는 네 개의 정수 x1, y1, x2, y2 가 주어지며, (x1, y1)부터 (x2, y2)의 합을 구해 출력해야 한다. 표에 채워져 있는 수는 1,000보다 작거나 같은 자연수이다. ($x1 \le x2$, $y1 \le y2$)

BOJ 11660 (구간 합 구하기 5)

- 2차원 누적합?

- psum[x][y] := (1, 1) ~ (x, y)까지의 합

- (x1, y1) ~ (x2, y2) 구간의 합



= psum[x2][y2] – psum[x1-1][y2] – psum[x2][y1-1] + psum[x1-1][y1-1]

필수 / 연습문제

[필수문제]

[BOJ 1932] 정수 삼각형

[BOJ 11048] 이동하기

[BOJ 13302] 리조트

[BOJ 9095] 1, 2, 3 더하기

[BOJ 1451] 직사각형으로 나누기

[BOJ 11568] 민균이의 계략

[연습문제]

[BOJ 2421] 저금통

[BOJ 2293] 동전 1

Challenge [BOJ 2096] 내려가기

★[BOJ 1915] 가장 큰 정사각형

[BOJ 5582] 공통 부분 문자열

Challenge [BOJ 1351] 무한 수열

[BOJ 11726] 2 x n 타일링

[BOJ 2133] 타일 채우기

[BOJ 2565] 전깃줄