ALGOLIVE - CAU ALGORITHM STUDY

DYNAMIC PROGRAMMING

REVIEW
: CONCEPT OF DYANAMIC PROGRAMMING

“동적 프로그래밍은
기억하기 프로그래밍이다”

DYANMIC PROGRAMMING

DYNAMIC PROGRAMMING의 등장

- 큰 문제를 한번에 해결하기 힘들 때 작은 여러 문제로 나누어 푸는 기법
- 작은 문제들이 같은 문제의 반복으로 이루어질 때 = 값들을 기억
- 동적 프로그래밍 = 기억하기 프로그래밍
- 사전에 계산된 값을 재활용

TOP-DOWN
: RECURSION

BOTTOM-UP
: LOOP

SILVER 3. #1463

1로 만들기

```cpp
#include <iostream>

using namespace std;

int N;
int dp[1000001];

int min(int a, int b){
    return a > b ? b : a;
}

int sol(int N){
    dp[1] = 0;
    dp[2] = 1;
    dp[3] = 1;
    for(int i =4 ; i < N+1 ; i++){
        if(i % 2 == 0 && i % 3 == 0){
            dp[i] = min(dp[i/3], dp[i/2])+1;
        }
        else if(i % 3 == 0){
            dp[i] = min(dp[i/3], dp[i-1])+1;
        }else if(i % 2== 0) {
            dp[i] = min(dp[i/2], dp[i-1])+1;
        }else{
            dp[i] = dp[i-1] + 1;
        }
    }
    return dp[N];
}

int main(void){
    cin >> N;
    cout << sol(N);
}
```

#1463 - C++ (나)

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

#define INF             1000001

using namespace std;

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

        int N;
        cin >> N;
        vector<int> dp;
        if (N > 3)
                dp = vector<int>(N+1);
        else
                dp = vector<int>(4);

        dp[1] = 0;
        dp[2] = 1;
        dp[3] = 1;

        for (int i=4; i<=N; ++i) {
                int a, b, c;
                a = b = c = INF;
                if (i%3 == 0)
                        a = dp[i/3];
                if (i%2 == 0)
                        b = dp[i/2];
                c = dp[i-1];
                dp[i] = min(min(a, b), c) + 1;
        }

        cout << dp[N] << '\n';
        return 0;
}
```

#1463 - C++

#1463 - JAVA

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);

        int N = in.nextInt();
        System.out.println(recur(N, 0));
    }

    static int recur(int N, int count) {
        // N이 2 미만인 경우 누적된 count값을 반환
        if (N < 2) {
            return count;
        }
        return Math.min(recur(N / 2, count + 1 + (N % 2)), recur(N / 3, count + 1 + (N % 3)));

    }
}
```

```python
n = int(input())
d = [0] * (n + 1)   ## d에 계산된 값을 저장해둔다. n + 1이라고 한 이유는, 1번째 수는 사실 d[1]이 아니

for i in range(2, n + 1):
## 여기서 왜 if 1빼는 방법, 2 나누기, 3 나누기 동등하게 하지 않고 처음에 1을 빼고 시작하는지 의아해 
## 1을 빼고 시작하는 이유는 다음에 계산할 나누기가 1을 뺀 값보다 작거나 큼에 따라 어차피 교체되기 때
## 즉 셋 다 시도하는 방법이 맞다.

## 여기서 if elif else를 사용하면 안된다. if만 이용해야 세 연산을 다 거칠 수 있다, 가끔 if continue,
    d[i] = d[i - 1] + 1
    if i % 3 == 0:
        d[i] = min(d[i], d[i // 3] + 1) ## 1을 더하는 것은 d는 결과가 아닌 계산한 횟수를 저장하는 것
    if i % 2 == 0:
        d[i] = min(d[i], d[i // 2] + 1)
print(d[n])
```

#1463 - Python

SILVER 1. #2156

포도주 시식

```cpp
#include <iostream>

using namespace std;

int arr[10000];
int dp[10000];
int max(int a, int b){
    return a > b ? a : b;
}
int sol(int num){
    int maxD = 0;
    if(num == 0){ return 0;}
    if(num == 1){ return arr[0];}
    maxD = max(arr[num-1] + dp[num-3], dp[num-2]);
    return maxD;
}
int main(void){
    int N, ans = 0;
    cin >> N;
    for(int i =0 ; i < N ; i++){
        cin >> arr[i];
    }
    for(int i =0 ; i < N ;i++){
        dp[i] = sol(i) + arr[i];
        if(i>0){
            dp[i] = max(dp[i], dp[i-1]);
        }
        ans = dp[i];
    }
    cout << ans;
}
```

#2156 - C++(나)

```cpp
#include <iostream>
#include <algorithm>

#define SIZE    10001

using namespace std;

int cost[SIZE] = { 0, };
int dp[SIZE] = { 0, };

int main() {
        int n;
        cin >> n;

        for (int i = 1; i <= n; ++i)
                cin >> cost[i];

        dp[1] = cost[1];
        dp[2] = cost[1] + cost[2];

        for (int i = 3; i <= n; ++i) {
                dp[i] = max(dp[i - 2] + cost[i], dp[i - 3] + cost[i - 1] + cost[i]);
                dp[i] = max(dp[i - 1], dp[i]);
        }

        cout << dp[n] << '\n';
        return 0;
}
```

#2156 - C++

```java
public static void main(String[] args) {

    Scanner in = new Scanner(System.in);

    int N = in.nextInt();

    int[] arr = new int[N + 1];
    int[] dp = new int[N + 1];

    for (int i = 1; i <= N; i++) {
        arr[i] = in.nextInt();
    }

    dp[1] = arr[1];
    if (N > 1) {
        dp[2] = arr[1] + arr[2];
    }
    for (int i = 3; i <= N; i++) {
        dp[i] = Math.max(dp[i - 1], Math.max(dp[i - 2] + arr[i], dp[i - 3] + arr[i - 1] + arr[i]));

    }
```

#2156 - JAVA

```python
n = int(input())
w = [0]
for i in range(n):
    w.append(int(input()))
dp = [0]
dp.append(w[1])
if n > 1:
    dp.append(w[1] + w[2])
for i in range(3, n + 1):
    dp.append(max(dp[i - 1], dp[i - 3] + w[i - 1] + w[i], dp[i - 2] + w[i]))
print(dp[n])
```

#2156 - Python

SILVER 1. #1149

RGB 거리

```cpp
#include <iostream>
#define INF 1e9

using namespace std;


int arr[1000][3];
int dp[1000][3];
int N;
int min(int a, int b, int c){
    if (a < b) {
        return a < c ? a : c;;
    } else return b < c ? b : c;
}

int min(int a, int b){
    return a > b ? b : a;
}

int sol(){
    dp[0][0] = arr[0][0];
    dp[0][1] = arr[0][1];
    dp[0][2] = arr[0][2];

    for(int i = 1 ; i < N; i++){
        dp[i][0]=min(dp[i-1][1],dp[i-1][2]) + arr[i][0];
        dp[i][1]=min(dp[i-1][0],dp[i-1][2]) + arr[i][1];
        dp[i][2]=min(dp[i-1][0],dp[i-1][1]) + arr[i][2];
    }
    return min(dp[N-1][0], dp[N-1][1], dp[N-1][2]);
}

int main(void){
    ios::sync_with_stdio(false);
    cin.tie(NULL);

    cin >> N;
    for(int i =0 ; i < N ; i++){
        cin >> arr[i][0] >> arr[i][1] >> arr[i][2];
    }
    cout << sol();
}
```

#1149 - C++ (나)

```cpp
#include <iostream>
#include <vector>

#define INF     1001
#define RGB     3

using namespace std;

int N;
vector<vector<int>> costs;
vector<vector<int>> dp;

int paint(int n, int last) {
        if (n >= 0) {
                if (last != INF && dp[n][last]) return dp[n][last];

                int min = INF;
                for (int i = 0; i < RGB; ++i) {
                        if (last == i) continue;
                        int root = N == n ? 0 : costs[n][i];
                        int sub = paint(n - 1, i);
                        int sum = root + sub;
                        if (sum < min || min == INF)
                                min = sum;
                }
                if (last == INF)
                        return min;
                return dp[n][last] = min;
        }
        return 0;
}

int main() {
        cin >> N;
        costs = vector<vector<int>>(N);
        dp = vector<vector<int>>(N+1);
        dp[N] = vector<int>(RGB);

        for (int i = 0; i < N; ++i) {
                costs[i] = vector<int>(RGB);
                dp[i] = vector<int>(RGB);
                for (int j = 0; j < RGB; ++j) {
                        cin >> costs[i][j];
                        dp[i][j] = 0;
                }
        }

        cout << paint(N, INF) << '\n';

        return 0;
}
```

#1149 - C++

```java
public class Main {

    final static int Red = 0;
    final static int Green = 1;
    final static int Blue = 2;

    public static void main(String[] args) throws IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st;

        int N = Integer.parseInt(br.readLine());

        int[][] Cost = new int[N][3];

        for (int i = 0; i < N; i++) {
            st = new StringTokenizer(br.readLine(), " ");

            Cost[i][Red] = Integer.parseInt(st.nextToken());
            Cost[i][Green] = Integer.parseInt(st.nextToken());
            Cost[i][Blue] = Integer.parseInt(st.nextToken());

        }

        // 1부터 N-1까지 각 i별 i-1의 서로 다른 색상 중 최솟값을 누적하여 더한다.
        for (int i = 1; i < N; i++) {
            Cost[i][Red] += Math.min(Cost[i - 1][Green], Cost[i - 1][Blue]);
            Cost[i][Green] += Math.min(Cost[i - 1][Red], Cost[i - 1][Blue]);
            Cost[i][Blue] += Math.min(Cost[i - 1][Red], Cost[i - 1][Green]);
        }

        System.out.println(Math.min(Math.min(Cost[N - 1][Red], Cost[N - 1][Green]), Cost[N - 1][Blue]))
    }

}
```

#1149 - JAVA

```python
n = int(input())
p = []
for i in range(n):
    p.append(list(map(int, input().split())))
for i in range(1, len(p)):
    p[i][0] = min(p[i - 1][1], p[i - 1][2]) + p[i][0]
    p[i][1] = min(p[i - 1][0], p[i - 1][2]) + p[i][1]
    p[i][2] = min(p[i - 1][0], p[i - 1][1]) + p[i][2]
print(min(p[n - 1][0], p[n - 1][1], p[n - 1][2]))
```

#1149 - Python

GOLD 5. #12865

평범한 배낭

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

int N, K;
int DP[101][100001];
int W[101];
int V[101];

// 점화식 max(DP[i-1][j], DP[i-1][j-W[i]])

int main()
{
	cin >> N >> K;

	for (int i = 1; i <= N; i++)
		cin >> W[i] >> V[i];

	for (int i = 1; i <= N; i++)
	{
		for (int j = 1; j <= K; j++)
		{

			if (j - W[i] >= 0) DP[i][j] = max(DP[i - 1][j], DP[i - 1][j - W[i]] + V[i]);
			else DP[i][j] = DP[i - 1][j];
		}
	}

	cout << DP[N][K];

}
```

#12865 - C++

```java
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String[] inputs = br.readLine().split(" ");

        int N = Integer.parseInt(inputs[0]);
        int K = Integer.parseInt(inputs[1]);

        int[][] item = new int[N + 1][2];  // weight, value

        for (int i = 1; i <= N; i++) {
            inputs = br.readLine().split(" ");
            item[i][0] = Integer.parseInt(inputs[0]);
            item[i][1] = Integer.parseInt(inputs[1]);
        }

        int[][] dp = new int[N + 1][K + 1];

        for (int k = 1; k <= K; k++) { // 무게
            for (int i = 1; i <= N; i++) { // item
                dp[i][k] = dp[i - 1][k];
                if (k - item[i][0] >= 0) {
                    dp[i][k] = Math.max(dp[i - 1][k], item[i][1] + dp[i - 1][k - item[i][0
                }
            }
        }


        System.out.println(dp[N][K]);
    }
}
```

#12865 - JAVA

```python
n, k = map(int, input().split())

thing = [[0,0]]
d = [[0]*(k+1) for _ in range(n+1)]

for i in range(n):
    thing.append(list(map(int, input().split())))

for i in range(1, n+1):
    for j in range(1, k+1):
        w = thing[i][0]
        v = thing[i][1]

        if j < w:
            d[i][j] = d[i-1][j]
        else:
            d[i][j] = max(d[i-1][j], d[i-1][j-w]+v)

print(d[n][k])
```

#12865 - Python