

ALGOLIVE - CAU ALGORITHM STUDY

DYNAMIC PROGRAMMING

REVIEW : CONCEPT OF DYNAMIC 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 = 값을 기억
- ▶ 동적 프로그래밍 = 기억하기 프로그래밍
- ▶ 사전에 계산된 값을 재활용

**TOP-DOWN
: RECURSION**

**BOTTOM-UP
: LOOP**



SILVER 3. #1463

12 만들기

#1463 - C++ (\vdash)

```
1 #include <iostream>
2
3 using namespace std;
4
5 int N;
6 int dp[1000001];
7
8 int min(int a, int b){
9     return a > b ? b : a;
10 }
11
12 int sol(int N){
13     dp[1] = 0;
14     dp[2] = 1;
15     dp[3] = 1;
16     for(int i =4 ; i < N+1 ; i++){
17         if(i % 2 == 0 && i % 3 == 0){
18             dp[i] = min(dp[i/3], dp[i/2])+1;
19         }
20         else if(i % 3 == 0){
21             dp[i] = min(dp[i/3], dp[i-1])+1;
22         }else if(i % 2== 0) {
23             dp[i] = min(dp[i/2], dp[i-1])+1;
24         }else{
25             dp[i] = dp[i-1] + 1;
26         }
27     }
28     return dp[N];
29 }
30
31 int main(void){
32     cin >> N;
33     cout << sol(N);
34 }
```

#1463 - C++

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4
5 #define INF          1000001
6
7 using namespace std;
8
9 int main() {
10     ios_base::sync_with_stdio(false);
11     cin.tie(NULL);
12
13     int N;
14     cin >> N;
15     vector<int> dp;
16     if (N > 3)
17         dp = vector<int>(N+1);
18     else
19         dp = vector<int>(4);
20
21     dp[1] = 0;
22     dp[2] = 1;
23     dp[3] = 1;
24
25     for (int i=4; i<=N; ++i) {
26         int a, b, c;
27         a = b = c = INF;
28         if (i%3 == 0)
29             a = dp[i/3];
30         if (i%2 == 0)
31             b = dp[i/2];
32         c = dp[i-1];
33         dp[i] = min(min(a, b), c) + 1;
34     }
35
36     cout << dp[N] << '\n';
37     return 0;
38 }
```

#1463 - JAVA

```
1 import java.util.Scanner;
2
3 public class Main {
4     public static void main(String[] args) {
5         Scanner in = new Scanner(System.in);
6
7         int N = in.nextInt();
8         System.out.println(recur(N, 0));
9     }
0
1     static int recur(int N, int count) {
2         // N이 2 미만인 경우 누적된 count값을 반환
3         if (N < 2) {
4             return count;
5         }
6         return Math.min(recur(N / 2, count + 1 + (N % 2)), recur(N / 3, count + 1 + (N % 3)));
7
8     }
9 }
```

```
n = int(input())
d = [0] * (n + 1)    ## d에 계산된 값을 저장해둔다. n + 1이라고 한 이유는, 1번째 수는 사실 d[1]이 아니기 때문
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 때문이다
    ## 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. #11052

카드 구매하기

#11052 - C++ (L)

```
#include <iostream>

using namespace std;

int N;
int arr[1000];
int dp[1000];

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

int sol(){
    int maxD;
    dp[0] = arr[0];
    for(int i = 1 ; i < N ; i++){
        maxD = 0;
        for(int j=0; j < i; j++){
            maxD = max( a: maxD, b: arr[j]+dp[i-j-1]);
            maxD = max( a: maxD, b: arr[i]);
        }
        dp[i] = maxD;
    }
    return dp[N-1];
}

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

#11052 - C++

```
#include <stdio.h>
#include <iostream>
#include <vector>
using namespace std;

vector<int> p(10001);
int dp[1001];
int main(void){

    int n;
    cin >> n;
    // n개 까지의 가격을 입력받는다.
    for(int i=1; i<=n; i++){
        cin >> p[i];
    }
    // n개 만큼의 값을 입력받고 i-j까지의 수를 넘어갈 수 없으므로, max를
    // i-j까지 p값을 더해나가면서 이용하여 최댓값을 구해준다.
    // 결국 최종 n의 값에 최대값이 들어가있는것을 확인 할 수 있다.

    for(int i=1; i<=n; i++){
        for(int j=1; j<=i; j++){
            dp[i] = max(dp[i], dp[i-j]+p[j]);
        }
    }
    cout << dp[n];

    return 0;
}
```

#11052 - JAVA

```
package Dynamic_Programing;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.StringTokenizer;

/*
 * Date : 20. 10. 26
 * Question.No : 백준 11052 – 카드 구매하기
 * Remark :
 */
public class Boj11052_2 {
    public static void main(String[] args) throws IOException {

        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int n = Integer.parseInt(br.readLine());
        int[] cardPack = new int[n+1];
        int[] maxArr = new int[n+1];
        cardPack[0] = 0;
        StringTokenizer st = new StringTokenizer(br.readLine());

        for(int i = 1 ; i <= n ; i++) {
            cardPack[i] = Integer.parseInt(st.nextToken());
        }

        for(int i = 1; i <=n; i++) {
            for(int j = 1; j <=i; j++) {
                maxArr[i] = Math.max(maxArr[i],cardPack[j]+maxArr[i-j]);
            }
        }
        System.out.println(maxArr[n]);
    }
}
```

#11052 - Python

```
N = int(input())
p = [0] + list(map(int,input().split()))
dp = [0 for _ in range(N+1)]

for i in range(1,N+1):
    for k in range(1,i+1):
        dp[i] = max(dp[i], dp[i-k] + p[k])
print(dp[i])
```



SILVER 1. #2156

포토주식

```
1 #include <iostream>
2
3 using namespace std;
4
5 int arr[10000];
6 int dp[10000];
7 int max(int a, int b){
8     return a > b ? a : b;
9 }
10 int sol(int num){
11     int maxD = 0;
12     if(num == 0){ return 0;}
13     if(num == 1){ return arr[0];}
14     maxD = max(arr[num-1] + dp[num-3], dp[num-2]);
15     return maxD;
16 }
17 int main(void){
18     int N, ans = 0;
19     cin >> N;
20     for(int i =0 ; i < N ; i++){
21         cin >> arr[i];
22     }
23     for(int i =0 ; i < N ;i++){
24         dp[i] = sol(i) + arr[i];
25         if(i>0){
26             dp[i] = max(dp[i], dp[i-1]);
27         }
28         ans = dp[i];
29     }
30     cout << ans;
31 }
```

#2156 - C++(L)

#2156 - C++

```
1 #include <iostream>
2 #include <algorithm>
3
4 #define SIZE    10001
5
6 using namespace std;
7
8 int cost[SIZE] = { 0, };
9 int dp[SIZE] = { 0, };
10
11 int main() {
12     int n;
13     cin >> n;
14
15     for (int i = 1; i <= n; ++i)
16         cin >> cost[i];
17
18     dp[1] = cost[1];
19     dp[2] = cost[1] + cost[2];
20
21     for (int i = 3; i <= n; ++i) {
22         dp[i] = max(dp[i - 2] + cost[i], dp[i - 3] + cost[i - 1] + cost[i]);
23         dp[i] = max(dp[i - 1], dp[i]);
24     }
25
26     cout << dp[n] << '\n';
27     return 0;
28 }
```

#2156 - 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]));  
    }  
}
```

```
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

```
1 #include <iostream>
2 #define INF 1e9
3
4 using namespace std;
5
6
7 int arr[1000][3];
8 int dp[1000][3];
9 int N;
10 int min(int a, int b, int c){
11     if (a < b) {
12         return a < c ? a : c;;
13     } else return b < c ? b : c;
14 }
15
16 int min(int a, int b){
17     return a > b ? b : a;
18 }
19
20 int sol(){
21     dp[0][0] = arr[0][0];
22     dp[0][1] = arr[0][1];
23     dp[0][2] = arr[0][2];
24
25     for(int i = 1 ; i < N; i++){
26         dp[i][0]=min(dp[i-1][1],dp[i-1][2]) + arr[i][0];
27         dp[i][1]=min(dp[i-1][0],dp[i-1][2]) + arr[i][1];
28         dp[i][2]=min(dp[i-1][0],dp[i-1][1]) + arr[i][2];
29     }
30     return min(dp[N-1][0], dp[N-1][1], dp[N-1][2]);
31 }
32
33 int main(void){
34     ios::sync_with_stdio(false);
35     cin.tie(NULL);
36
37     cin >> N;
38     for(int i =0 ; i < N ; i++){
39         cin >> arr[i][0] >> arr[i][1] >> arr[i][2];
40     }
41     cout << sol();
42 }
```

#1149 - C++ (\vdash)

#1149 - C++

```
1 #include <iostream>
2 #include <vector>
3
4 #define INF    1001
5 #define RGB    3
6
7 using namespace std;
8
9 int N;
10 vector<vector<int>> costs;
11 vector<vector<int>> dp;
12
13 int paint(int n, int last) {
14     if (n >= 0) {
15         if (last != INF && dp[n][last]) return dp[n][last];
16
17         int min = INF;
18         for (int i = 0; i < RGB; ++i) {
19             if (last == i) continue;
20             int root = N == n ? 0 : costs[n][i];
21             int sub = paint(n - 1, i);
22             int sum = root + sub;
23             if (sum < min || min == INF)
24                 min = sum;
25         }
26         if (last == INF)
27             return min;
28         return dp[n][last] = min;
29     }
30     return 0;
31 }
32
33 int main() {
34     cin >> N;
35     costs = vector<vector<int>>(N);
36     dp = vector<vector<int>>(N+1);
37     dp[N] = vector<int>(RGB);
38
39     for (int i = 0; i < N; ++i) {
40         costs[i] = vector<int>(RGB);
41         dp[i] = vector<int>(RGB);
42         for (int j = 0; j < RGB; ++j) {
43             cin >> costs[i][j];
44             dp[i][j] = 0;
45         }
46     }
47
48     cout << paint(N, INF) << '\n';
49
50     return 0;
51 }
```

```
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

```
n = int(input())
p = []
for i in range(n):
    p.append(list(map(int, input().split())))
for i in range(1, len(p)): #1149 - Python
    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]))
```



(BONUS) GOLD 5. #12865

평범한 배낭

```
#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++

```
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

```
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