다이나믹 프로그래밍 1 (연습)

최백준 choi@startlink.io

1, 2, 3 더하기 3

https://www.acmicpc.net/problem/15988

- 정수 n을 1, 2, 3의 합으로 나타내는 방법의 수를 구하는 문제 ($n \leq 1{,}000{,}000$)
- n = 4
- 1+1+1+1
- 1+1+2
- 1+2+1
- 2+1+1
- 2+2
- 1+3
- 3+1

1, 2, 3 더하기 3

https://www.acmicpc.net/problem/15988

- 소스: http://codeplus.codes/b494f455c9e64ceeb7080ce88a87cfc9

RGB 거리

https://www.acmicpc.net/problem/1149

- RGB거리에 사는 사람들은 집을 빨강, 초록, 파랑중에 하나로 칠하려고 한다
- 또한, 그들은 모든 이웃은 같은 색으로 칠할 수 없다는 규칙도 정했다
- 집 i의 이웃은 집 i-1과 집 i+1이고, 첫 집과 마지막 집은 이웃이 아니다.
- 처음 집과 마지막 집은 이웃이 아니다
- 각 집을 빨강으로 칠할 때 드는 비용, 초록으로 칠할 때 드는 비용, 파랑으로 드는 비용이 주어질 때, 모든 집을 칠하는 비용의 최솟값을 구하는 문제

RGB 거리

https://www.acmicpc.net/problem/1149

- D[i][j] = i번 집을 색 j로 칠했을 때, 1~i번 집을 칠하는 비용의 최소값
  - j = 0 → 빨강
  - j = 1 → 초록
  - j = 2 → 파랑
- D[i][j] = i번 집을 색 j로 칠했을 때, 1~i번 집을 칠하는 비용의 최소값

RGB 거리

https://www.acmicpc.net/problem/1149

- D[i][0] = min(D[i-1][1], D[i-1][2]) + A[i][0]
- D[i][1] = min(D[i-1][0], D[i-1][2]) + A[i][1]
- D[i][2] = min(D[i-1][0], D[i-1][1]) + A[i][2]

RGB 거리

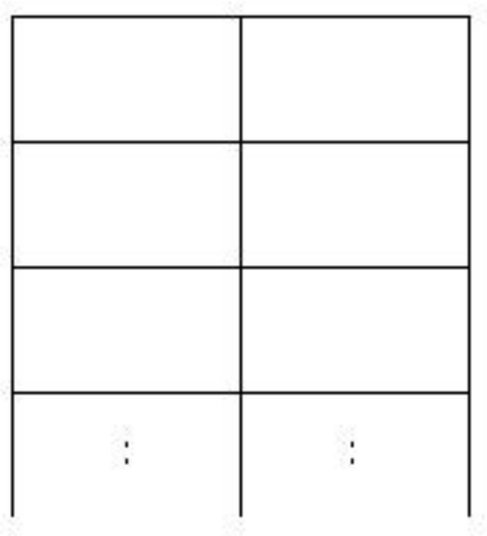
https://www.acmicpc.net/problem/1149

- 소스: http://codeplus.codes/d48b0737233e4b40a3df30d871890c4e

동물원

https://www.acmicpc.net/problem/1309

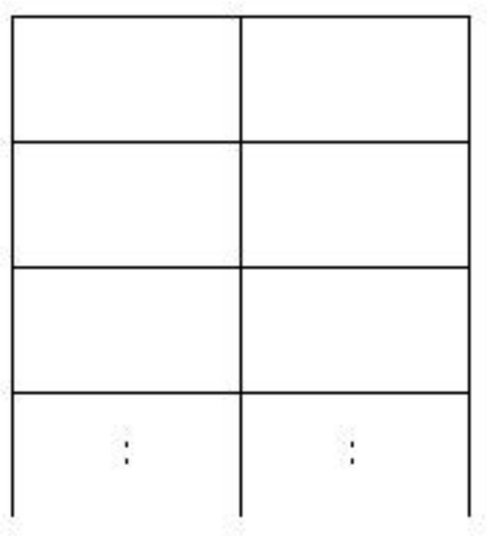
- 가로로 두 칸, 세로로 N 칸인 동물원이 있다
- 가로, 세로로 붙어 있게 배치하면 안된다
- 가능한 배치의 수



동물원

https://www.acmicpc.net/problem/1309

- D[N][0] = N번 줄에 배치하지 않음
- D[N][1] = N번 줄의 왼쪽에만 배치함
- D[N][2] = N번 줄의 오른쪽에만 배치함

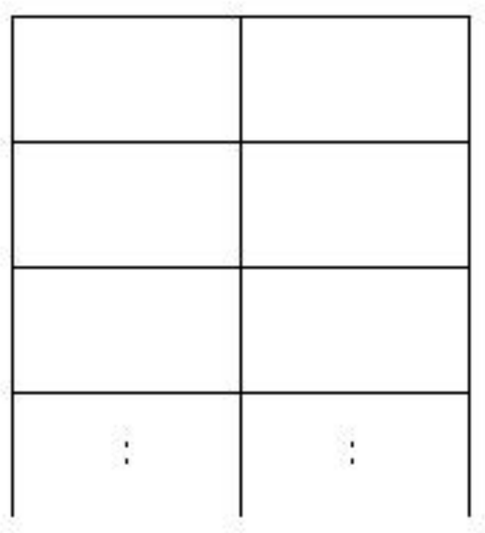


동물원

https://www.acmicpc.net/problem/1309

- D[N][0] = N번 줄에 배치하지 않음
- D[N][1] = N번 줄의 왼쪽에만 배치함
- D[N][2] = N번 줄의 오른쪽에만 배치함

- D[N][0] = D[N-1][0] + D[N-1][1] + D[N-1][2]
- D[N][1] = D[N-1][0] + D[N-1][2]
- D[N][2] = D[N-1][0] + D[N-1][1]



동물원

https://www.acmicpc.net/problem/1309

- 소스: http://codeplus.codes/dcbc1412fd644cd6a68538f4fb748990

오르막 수

https://www.acmicpc.net/problem/11057

- 오르막 수는 수의 자리가 오름차순을 이루는 수를 말한다
- 인접한 수가 같아도 오름차순으로 친다
- 수의 길이 N이 주어졌을 때, 오르막 수의 개수를 구하는 문제
- 수는 0으로 시작할 수 있다
- 예: 1233345, 357, 8888888, 1555999

오르막 수

https://www.acmicpc.net/problem/11057

- D[i][j] = 길이가 i이고 마지막 숫자가 j인 오르막 수의 개수
- D[1][i] = 1
- D[i][j] += D[i-1][k] (0 ≤ k ≤ j)

오르막 수

https://www.acmicpc.net/problem/11057

```
for (int i=0; i<=9; i++) d[1][i] = 1;
for (int i=2; i<=n; i++) {
    for (int j=0; j<=9; j++) {
        for (int k=0; k<=j; k++) {
            d[i][j] += d[i-1][k];
            d[i][j] %= mod;
        }
    }
}
long long ans = 0;
for (int i=0; i<10; i++) ans  += d[n][i];
ans %= mod;
```

오르막 수

https://www.acmicpc.net/problem/11057

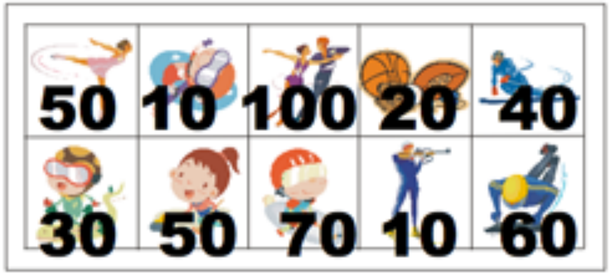
- 소스: http://codeplus.codes/dc186ddd614b4569af3f34313b67bbd5

스티커

https://www.acmicpc.net/problem/9465

- 스티커 2n개가 2×n 모양으로 배치되어 있다
- 스티커 한 장을 떼면 변을 공유하는 스티커는 모두 찢어져서 사용할 수 없다
- 점수의 합을 최대로 만드는 문제

(a)



(b)

스티커

https://www.acmicpc.net/problem/9465

- D[i][j] = 2×i 에서 얻을 수 있는 최대 점수, i번 열에서 뜯는 스티커는 j
- j = 0 -> 뜯지 않음
- j = 1 -> 위쪽 스티커를 뜯음
- j = 2 -> 아래쪽 스티커를 뜯음

스티커

https://www.acmicpc.net/problem/9465

- D[i][j] = 2×i 에서 얻을 수 있는 최대 점수, i번 열에서 뜯는 스티커는 j
- 뜯지 않음 (D[i][0])
  - i-1 열에서 스티커를 어떻게 뜯었는지 상관이 없다
  - max(D[i-1][0], D[i-1][1], D[i-1][2])
- 위쪽 스티커를 뜯음 (D[i][1])
  - i-1열에서 위쪽 스티커는 뜯으면 안된다
  - max(D[i-1][0], D[i-1][2]) + A[i][0]
- 아래쪽 스티커를 뜯음 (D[i][2])
  - i-1열에서 아래쪽 스티커는 뜯으면 안된다
  - max(D[i-1][0], D[i-1][1]) + A[i][1]

스티커

https://www.acmicpc.net/problem/9465

- 소스: http://codeplus.codes/67eeb5961a20490db92bee8b569af74d

포도주 시식

https://www.acmicpc.net/problem/2156

- 포도주가 일렬로 놓여져 있고, 다음과 같은 2가지 규칙을 지키면서 포도주를 최대한 많이 마시려고 한다.

1. 포도주 잔을 선택하면 그 잔에 들어있는 포도주는 모두 마셔야 하고, 마신 후에는 원래 위치에 다시 놓아야 한다.
2. 연속으로 놓여 있는 3잔을 모두 마실 수는 없다.

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양
- i에게 가능한 경우

1. i번째 포도주를 마시는 경우
2. i번째 포도주를 마시지 않는 경우

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양
- i에게 가능한 경우

1. i번째 포도주를 마시는 경우
    - D[i-1] + A[i]
2. i번째 포도주를 마시지 않는 경우
    - D[i-1]

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양
- i에게 가능한 경우

1. i번째 포도주를 마시는 경우
    - D[i-1] + A[i]
2. i번째 포도주를 마시지 않는 경우
    - D[i-1]

- D[i] = max(D[i-1]+A[i], D[i-1])
- 위의 식은 포도주를 연속해서 3잔 마시면 안되는 경우를 처리하지 못한다.

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i][j] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양, A[i]는 j번 연속해서 마신 포도주임
- D[i][0] = 0번 연속해서 마신 포도주 → A[i]를 마시지 않음
- D[i][1] = 1번 연속해서 마신 포도주 -> A[i-1]을 마시지 않았음
- D[i][2] = 2번 연속해서 마신 포도주 → A[i-1]을 마시고, A[i-2]는 마시지 않았어야 함

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i][j] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양, A[i]는 j번 연속해서 마신 포도주임
- D[i][0] = 0번 연속해서 마신 포도주 → A[i]를 마시지 않음
  - max(D[i-1][0], D[i-1][1], D[i-1][2])
- D[i][1] = 1번 연속해서 마신 포도주 -> A[i-1]을 마시지 않았음
  - D[i-1][0] + A[i]
- D[i][2] = 2번 연속해서 마신 포도주 → A[i-1]을 마시고, A[i-2]는 마시지 않았어야 함
  - D[i-1][1] + A[i]

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양
- 0번 연속해서 마신 포도주 → A[i]를 마시지 않음
- 1번 연속해서 마신 포도주 -> A[i-1]을 마시지 않았음
- 2번 연속해서 마신 포도주 → A[i-1]을 마시고, A[i-2]는 마시지 않았어야 함

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양
- 0번 연속해서 마신 포도주 → A[i]를 마시지 않음
    - D[i-1]
- 1번 연속해서 마신 포도주 -> A[i-1]을 마시지 않았음
    - D[i-2] + A[i]
- 2번 연속해서 마신 포도주 → A[i-1]을 마시고, A[i-2]는 마시지 않았어야 함
    - D[i-3] + A[i-1] + A[i]

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양
- 0번 연속해서 마신 포도주 → A[i]를 마시지 않음
    - D[i-1]
- 1번 연속해서 마신 포도주 -> A[i-1]을 마시지 않았음
    - D[i-2] + A[i]
- 2번 연속해서 마신 포도주 → A[i-1]을 마시고, A[i-2]는 마시지 않았어야 함
    - D[i-3] + A[i-1] + A[i]
- D[i] = max(D[i-1], D[i-2]+A[i], D[i-3] + A[i-1] + A[i])

포도주 시식

https://www.acmicpc.net/problem/2156

- D[i] = A[1], …, A[i] 까지 포도주를 마셨을 때, 마실 수 있는 포도주의 최대 양
- D[i] = max(D[i-1], D[i-2]+A[i], D[i-3] + A[i-1] + A[i])
- i-2, i-3 때문에 예외 처리가 예상되기 때문에
- D[1] = A[1]
- D[2] = A[1] + A[2]
- 로 미리 처리를 해두고
- i = 3부터 문제를 푸는 것이 좋다.

포도주 시식

https://www.acmicpc.net/problem/2156

```
d[1] = a[1];
d[2] = a[1]+a[2];
for (int i=3; i<=n; i++) {
    d[i] = d[i-1];
    if (d[i] < d[i-2] + a[i]) {
        d[i] = d[i-2] + a[i];
    }
    if (d[i] < d[i-3] + a[i] + a[i-1]) {
        d[i] = d[i-3] + a[i] + a[i-1];
    }
}
```

포도주 시식

https://www.acmicpc.net/problem/2156

- 소스: http://codeplus.codes/717b6ff705264255b16bb435ff45f666

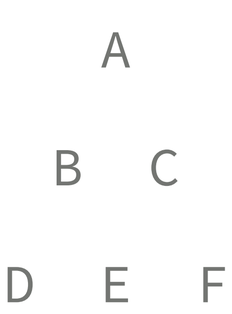
정수 삼각형

https://www.acmicpc.net/problem/1932

정수 삼각형

https://www.acmicpc.net/problem/1932

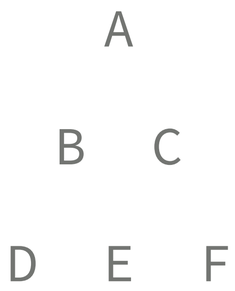
- 아래층으로 내려올 때는 대각선 왼쪽 또는 대각선 오른쪽에 있는 것만 선택할 수 있다.
- A에서 선택할 수 있는 수: B, C
- B에서 선택할 수 있는 수: D, E
- C에서 선택할 수 있는 수: E, F



정수 삼각형

https://www.acmicpc.net/problem/1932

- 반대로 생각해서 어떤 수가 선택되기 전에 선택된 수는 대각선 왼쪽 위, 오른쪽 위에 있는 것이다.
- B에 오기 전: A
- C에 오기 전: C
- D에 오기 전: B
- E에 오기 전: B, C
- F에 오기 전: C



정수 삼각형

https://www.acmicpc.net/problem/1932

- D[i][j] = i행 j열가 선택되었을 때, 최대 합
- (i, j)가 선택되기 전에 선택된 수는 (i-1, j), (i-1, j-1) 중 하나다.
- D[i][j] = Max(D[i-1][j], D[i-1][j-1]) + A[i][j]

정수 삼각형

https://www.acmicpc.net/problem/1932

- 소스: http://codeplus.codes/fbc271224ec54d57b0c225f834435156

가장 큰 증가하는 부분 수열

https://www.acmicpc.net/problem/11055

- 수열 A가 주어졌을 때, 그 수열의 증가 부분 수열 중에서 합이 가장 큰 것을 구하는 문제

가장 큰 증가하는 부분 수열

https://www.acmicpc.net/problem/11055

- D[i] = A[1], …, A[i] 까지 수열이 있을 때, A[i]을 마지막으로 하는 가장 큰 증가하는 부분 수열의 길이

가장 큰 증가하는 부분 수열

https://www.acmicpc.net/problem/11055

```
for (int i=0; i<n; i++) {
    d[i] = 1;
    for (int j=0; j<i; j++) {
        if (a[j] < a[i] && d[i] < d[j]+1) {
            d[i] = d[j]+1;
        }
    }
}
```

가장 큰 증가하는 부분 수열

https://www.acmicpc.net/problem/11055

```
for (int i=0; i<n; i++) {
    d[i] = 1;
    for (int j=0; j<i; j++) {
        if (a[j] < a[i] && d[i] < d[j]+a[i]) {
            d[i] = d[j]+a[i];
        }
    }
}
```

가장 큰 증가하는 부분 수열

https://www.acmicpc.net/problem/11055

```
for (int i=0; i<n; i++) {
    d[i] = a[i];
    for (int j=0; j<i; j++) {
        if (a[j] < a[i] && d[i] < d[j]+a[i]) {
            d[i] = d[j]+a[i];
        }
    }
}
```

가장 큰 증가하는 부분 수열

https://www.acmicpc.net/problem/11055

- 소스: http://codeplus.codes/75bc0900d3b04002bd47b37d68ea5972

가장 긴 감소하는 부분 수열

https://www.acmicpc.net/problem/11722

- 수열 A가 주어졌을 때, 그 수열의 감소하는 부분 수열 중에서 가장 긴 것을 구하는 문제
- 두 가지 방법이 있다
- 입력으로 주어진 수열 A를 뒤집어서 가장 긴 증가하는 부분 수열을 구하는 방법
- 가장 긴 증가하는 부분 수열과 비슷하게 구하는 방법 (뒤에서부터 구해야 한다)

가장 긴 감소하는 부분 수열

https://www.acmicpc.net/problem/11722

- D[i] = A[i]에서 시작하는 가장 긴 감소하는 부분 수열의 길이

가장 긴 감소하는 부분 수열

https://www.acmicpc.net/problem/11722

- D[i] = A[i]에서 시작하는 가장 긴 감소하는 부분 수열의 길이
- D[i] = max(D[j]) + 1 (i < j && A[i] > A[j])

가장 긴 감소하는 부분 수열

https://www.acmicpc.net/problem/11722

- 소스: http://codeplus.codes/4caa16e09461451b973bbfb8cf70e511

가장 긴 감소하는 부분 수열

https://www.acmicpc.net/problem/11722

- D[i] = A[i]에서 끝나는 가장 긴 감소하는 부분 수열의 길이
- D[i] = max(D[j]) + 1 (j < i && A[j] > A[i])

가장 긴 감소하는 부분 수열

https://www.acmicpc.net/problem/11722

- 소스: http://codeplus.codes/e2bd30aac337479c929b575f46dc543c

가장 긴 바이토닉 부분 수열

https://www.acmicpc.net/problem/11054

- 수열 A가 주어졌을 때, 그 수열의 바이토닉 부분 수열 중에서 가장 긴 것을 구하는 문제

가장 긴 바이토닉 부분 수열

https://www.acmicpc.net/problem/11054

- 가장 긴 증가하는 부분 수열(D)과 가장 긴 감소하는 부분 수열(D2)를 구한 다음
- D[i] = i에서 끝나는 가장 긴 증가하는 부분 수열
- D2[i] = i에서 시작하는 가장 긴 감소하는 부분 수열
- D[i] + D2[i] – 1이 가장 큰 값을 찾으면 된다

가장 긴 바이토닉 부분 수열

https://www.acmicpc.net/problem/11054

- 소스: http://codeplus.codes/9aa0833f59ff4bba9fedd03f0ab89ed5

연속합 2

https://www.acmicpc.net/problem/13398

- 수열의 연속합 중 가장 큰 합을 구하는 문제
- 수는 하나 제거할 수 있다. 제거하지 않아도 된다.

연속합 2

https://www.acmicpc.net/problem/13398

- 연속합 문제를 O(N)에 풀 수 있다.
- 연속합 문제를 총 N번 풀면 된다.
- A[0]을 제외하고 구하고 O(N)
- A[1]을 제외하고 구하고 O(N)
- …
- A[N-1]을 제외하고 구하고 O(N)
- 총 $O(N^2)$
- $N \leq 100,000$이기 때문에, 너무 오랜 시간이 걸린다.

연속합 2

https://www.acmicpc.net/problem/13398

- 어떤 수 A[i]를 제거했을 때, 매번 전체를 다 구하는 것은 비효율적이다.
- DL[i] = i번째 수에서 끝나는 최대 연속합
- DR[i] = i번째 수에서 시작하는 최대 연속합
- 이 값을 이용해서 A[i]를 제거했을 때 최대 연속합을 구할 수 있다.
- i번째 수를 제거하면 i-1번째 수와 i+1번째 수가 연속하게 된다.
- 따라서, DL[i-1] + DR[i+1]이 i번째 수를 제거했을 때, i번째 수가 포함되는 최대 연속합이 된다.

연속합 2

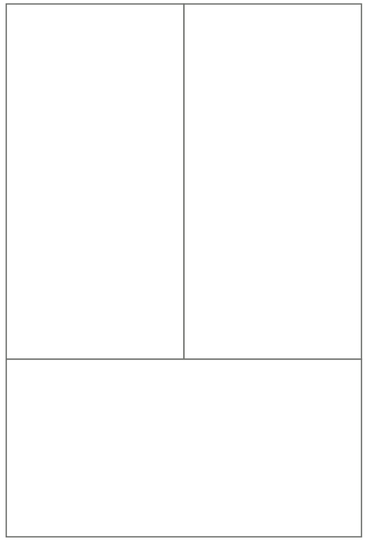
https://www.acmicpc.net/problem/13398

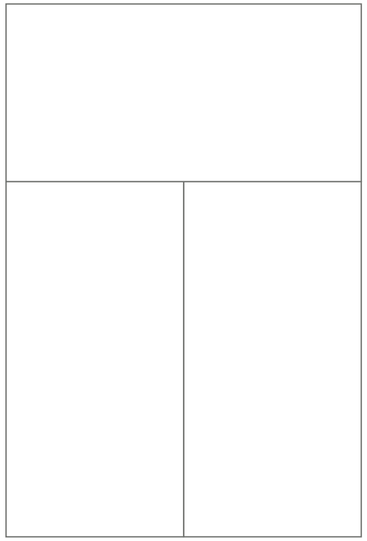
- 소스: http://codeplus.codes/2b8d4fb11a5b4d4193ea76a51550e4cd

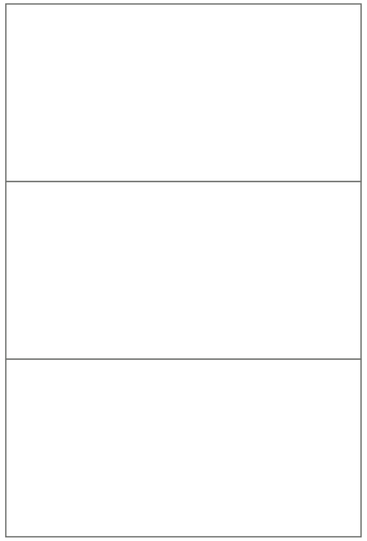
타일 채우기

https://www.acmicpc.net/problem/2133

- 3×N을 1×2, 2×1로 채우는 방법의 수
- D[i] = 3×i를 채우는 방법의 수
- 마지막에 올 수 있는 가능한 경우의 수



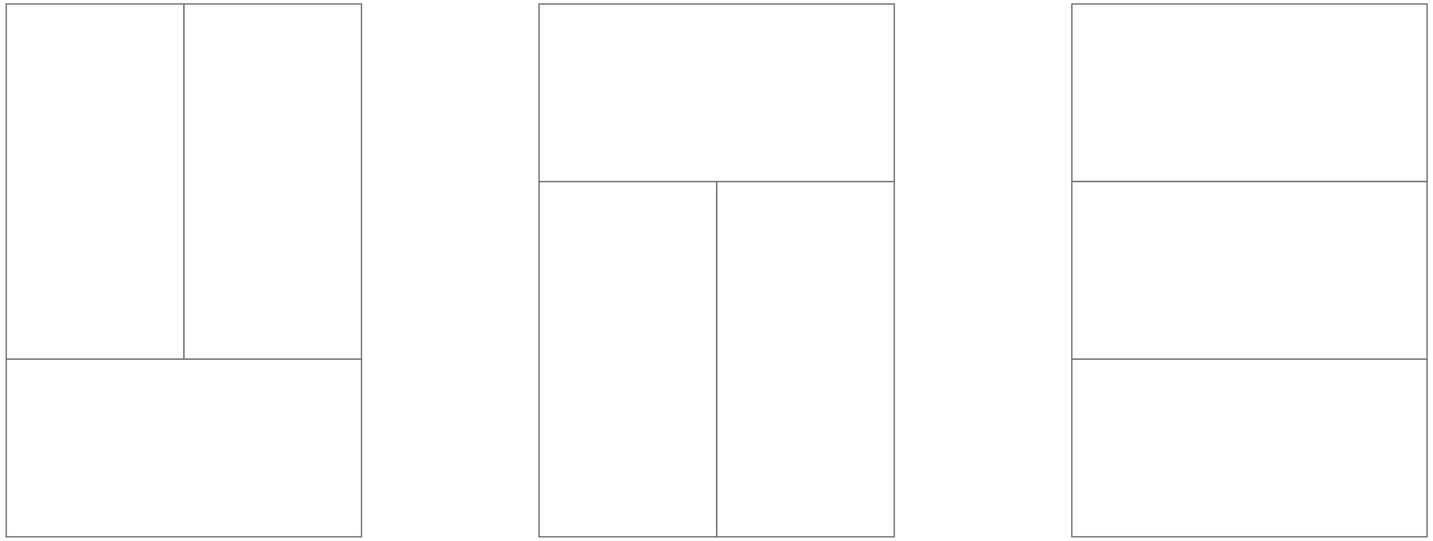




타일 채우기

https://www.acmicpc.net/problem/2133

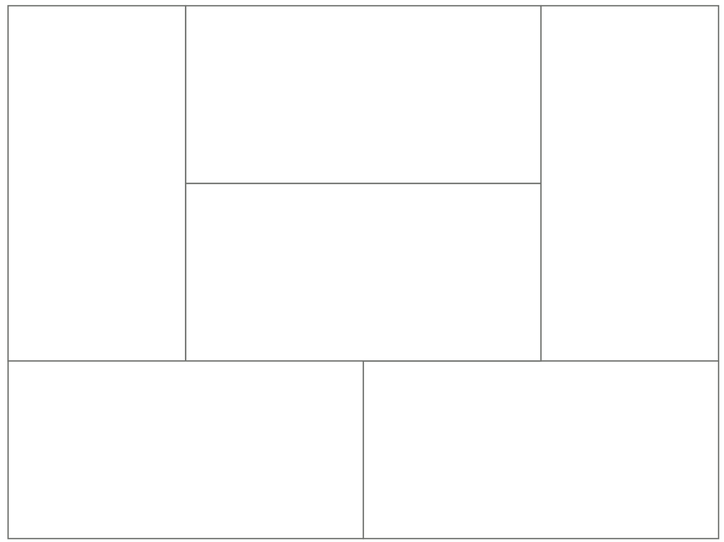
- 3×N을 1×2, 2×1로 채우는 방법의 수
- D[i] = 3×i를 채우는 방법의 수
- D[i] = 3 * D[i-2] (아니다)

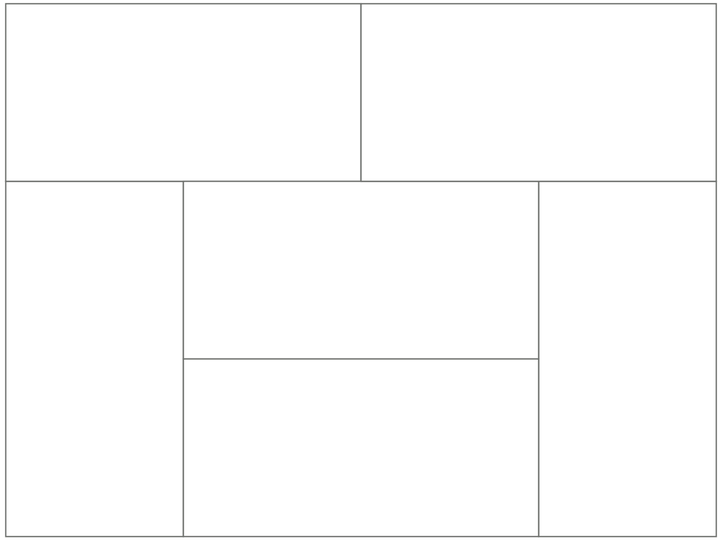


타일 채우기

https://www.acmicpc.net/problem/2133

- 3×N을 1×2, 2×1로 채우는 방법의 수
- D[i] = 3×i를 채우는 방법의 수
- 가능한 경우가 더 있다.

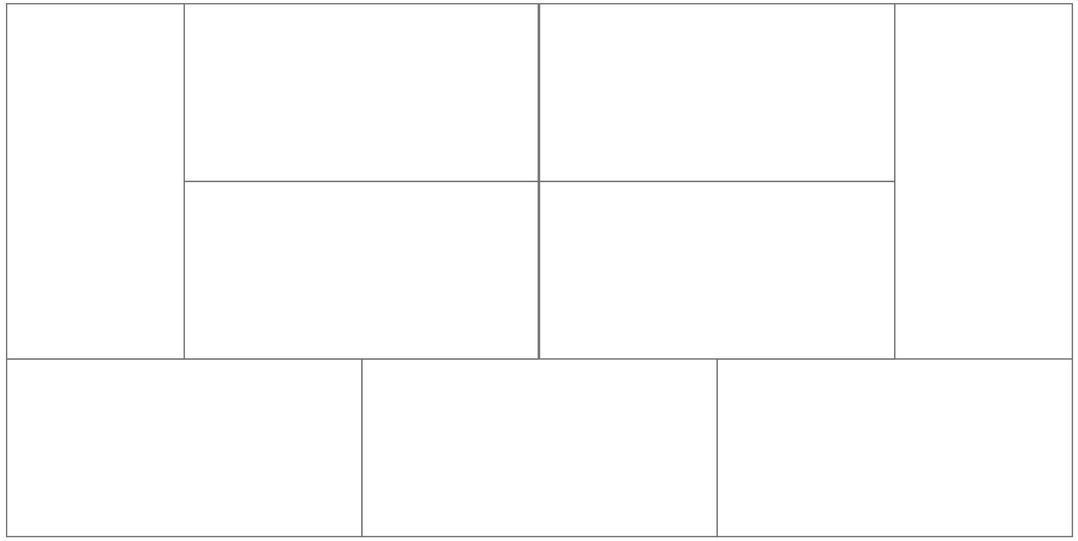


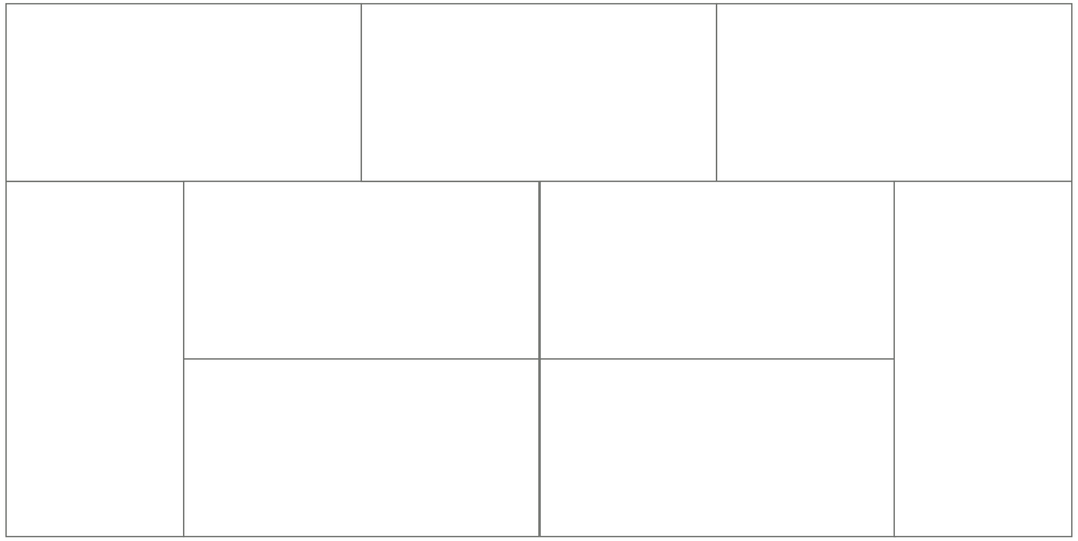


타일 채우기

https://www.acmicpc.net/problem/2133

- 3×N을 1×2, 2×1로 채우는 방법의 수
- D[i] = 3×i를 채우는 방법의 수
- 가능한 경우가 더 있다.





타일 채우기

https://www.acmicpc.net/problem/2133

- 3×N을 1×2, 2×1로 채우는 방법의 수
- D[i] = 3×i를 채우는 방법의 수
- D[i] = 3 * D[i-2] + 2*D[i-4] + 2*D[i-6] + …

타일 채우기

https://www.acmicpc.net/problem/2133

- 소스: http://codeplus.codes/2ff02be500dd442aa9de1955d29d8d23

끝

코드 플러스

https://code.plus

- 슬라이드에 포함된 소스 코드를 보려면 "정보 수정 > 백준 온라인 저지 연동"을 통해 연동한 다음, "백준 온라인 저지"에 로그인해야 합니다.
- 강의 내용에 대한 질문은 코드 플러스의 "질문 게시판"에서 할 수 있습니다.
- 문제와 소스 코드는 슬라이드에 첨부된 링크를 통해서 볼 수 있으며, "백준 온라인 저지"에서 서비스됩니다.
- 슬라이드와 동영상 강의는 코드 플러스 사이트를 통해서만 볼 수 있으며, 동영상 강의의 녹화와 다운로드, 배포와 유통은 저작권법에 의해서 금지되어 있습니다.
- 다른 경로로 이 슬라이드나 동영상 강의를 본 경우에는 codeplus@startlink.io 로 이메일 보내주세요.
- 강의 내용, 동영상 강의, 슬라이드, 첨부되어 있는 소스 코드의 저작권은 스타트링크와 최백준에게 있습니다.