다이나믹 프로그래밍 3

최백준 choi@startlink.io

문제 풀기

타일 채우기

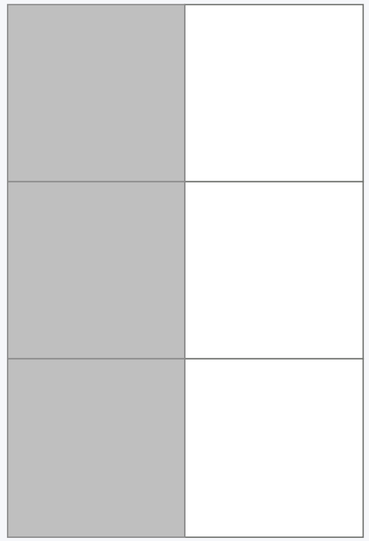
https://www.acmicpc.net/problem/2133

- 3×N을 1×2, 2×1로 채우는 방법의 수
- D[i][j] = 3×i를 채우는 방법의 수, i열의 상태는 j
- 마지막에 올 수 있는 가능한 경우의 수 (회색: 채워져 있는 칸)

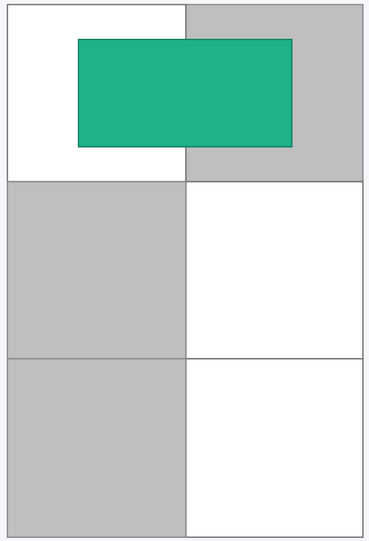
타일 채우기

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

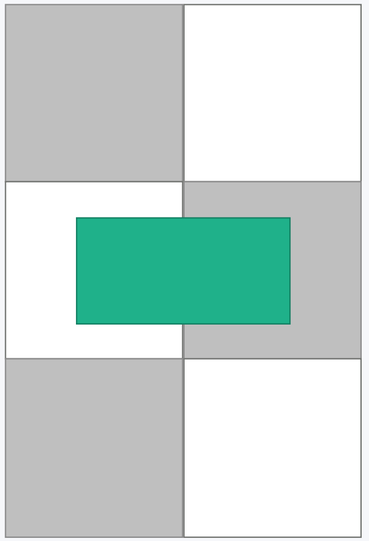
- i열을 채울 때, i-1에 빈 칸이 있으면 안된다



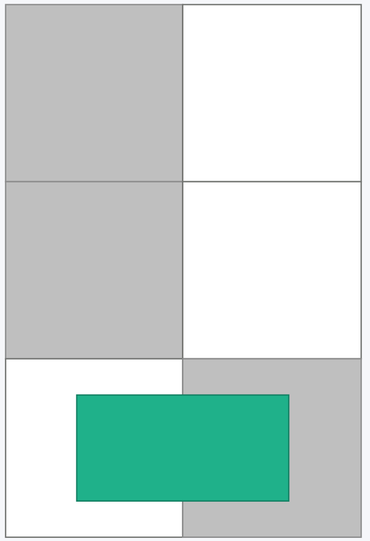
7 0



6 1



5 2

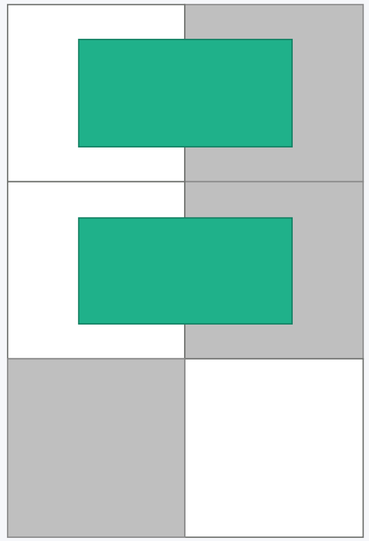


3 4

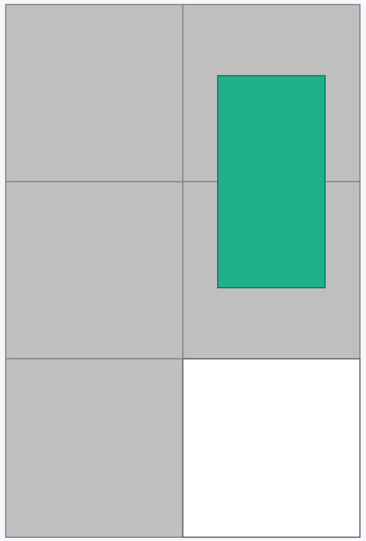
타일 채우기

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

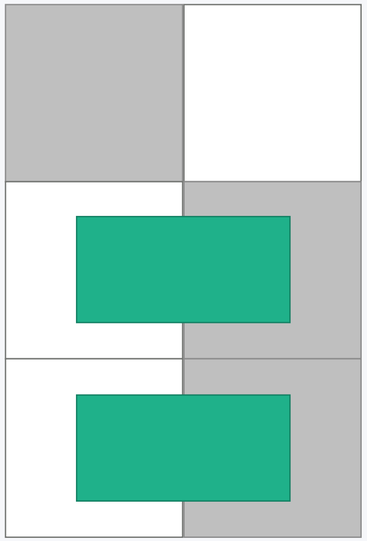
- i열을 채울 때, i-1에 빈 칸이 있으면 안된다



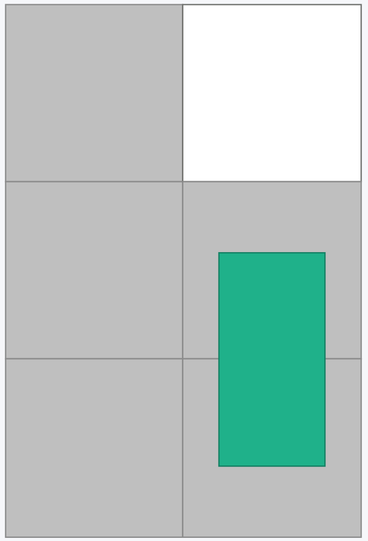
4 3



7 3



1 6

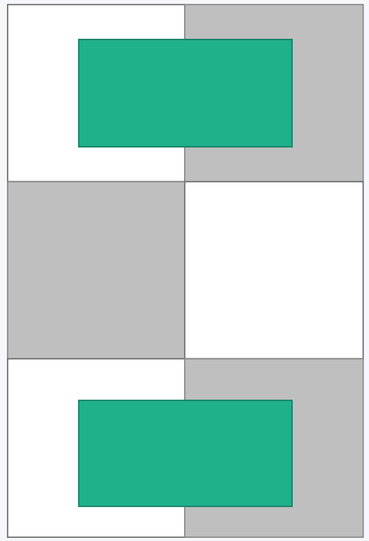


7 6

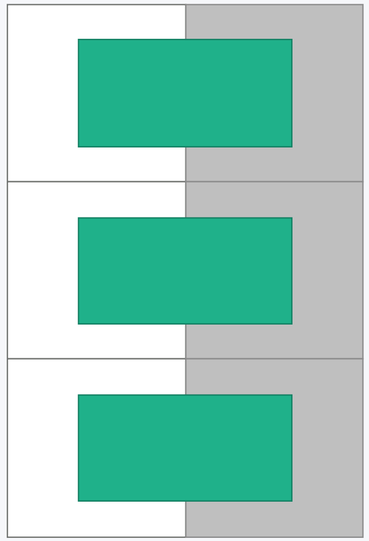
타일 채우기

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

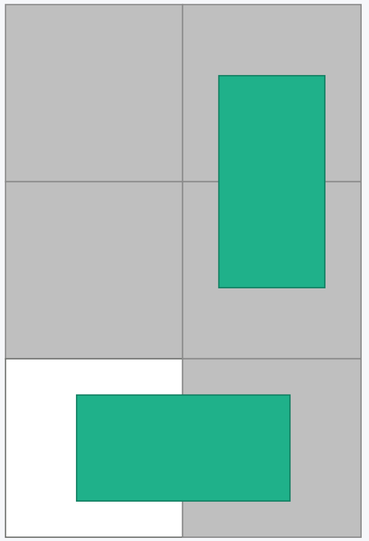
- i열을 채울 때, i-1에 빈 칸이 있으면 안된다



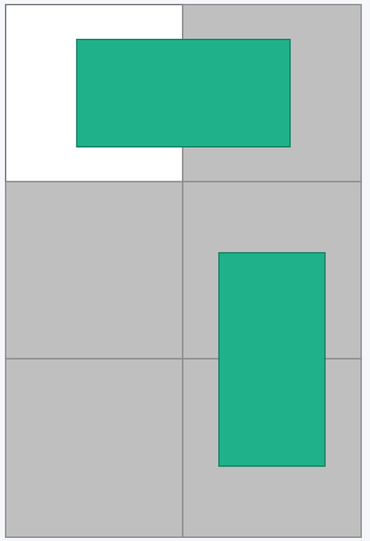
2 5



0 7



3 7



6 7

타일 채우기

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

- D[i][0] = D[i-1][7]
- D[i][1] = D[i-1][6]
- D[i][2] = D[i-1][5]
- D[i][4] = D[i-1][3]
- D[i][3] = D[i-1][4] + D[i-1][7]
- D[i][6] = D[i-1][1] + D[i-1][7]
- D[i][5] = D[i-1][2]
- D[i][7] = D[i-1][0] + D[i-1][3] + D[i-1][6]

타일 채우기

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

```
D[0][7] = 1;
for (int i=1; i<=n; i++) {
    D[i][0] = D[i-1][7];
    D[i][1] = D[i-1][6];
    D[i][2] = D[i-1][5];
    D[i][4] = D[i-1][3];
    D[i][3] = D[i-1][4] + D[i-1][7];
    D[i][6] = D[i-1][1] + D[i-1][7];
    D[i][5] = D[i-1][2];
    D[i][7] = D[i-1][0] + D[i-1][3] + D[i-1][6];
}
```

타일 채우기

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

- C++
  - https://gist.github.com/Baekjoon/fefbc054071fa3fdb80b

외판원 순회

https://www.acmicpc.net/problem/2098

- 도시 1번부터 N번까지 있을 때
- 어느 한 도시에서 출발해서 N개의 도시를 거쳐 다시 원래 도시로 돌아오는 순회 여행 경로
- 비용의 최소값
- $N \leq 16$

외판원 순회

https://www.acmicpc.net/problem/2098

- D[S][i] = 도시를 방문한 상태가 S이고, 현재 있는 위치가 i일때 최소값

- 시작 도시는 1로 고정한다.
- 1 → 2 → 3 → 1
- 2 → 3 → 1 → 2
- 모두 같기 때문

외판원 순회

https://www.acmicpc.net/problem/2098

- 상태를 사용할 수 있는 이유

- 1 → 4까지 정답이
- 1 → 2 → 3 → 4 라고 하자.
- 그러면, 1 → 3 → 2 → 4가 더 크다면, 방문한 도시의 집합은 같기 때문에 필요없는 값이 된다.
- 1 → 5로 갈 때 4에서 가는 경우라면
- 1 → 2 → 3 → 4 → 5가
- 1 → 3 → 2 → 4 → 5보다 무조건 최소값이다

외판원 순회

https://www.acmicpc.net/problem/2098

- D[S][i] = D[S2][j] + A[j][i]
- S2 = S에서 i를 뺀 값
- i는 S에는 포함되어 있어야 하고, S2에는 포함되어 있지 않아야 한다
- j는 S에 포함되어 있어야 한다
- 초기값
  - D[1][0] = 0
  - 나머지 = MAX
- 정답
  - Min(D[(1<<N)-1][i] + A[i][0])

외판원 순회

https://www.acmicpc.net/problem/2098

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

외판원 순회

https://www.acmicpc.net/problem/2098

- C/C++
  - https://gist.github.com/Baekjoon/e9e9d2ce22d08e212aba
- Java
  - https://gist.github.com/Baekjoon/408cb5dff31ed9354568

발전소

https://www.acmicpc.net/problem/1102

- 발전소를 고치는 방법은 간단하다.
- 고장나지 않은 발전소를 이용해서 고장난 발전소를 재시작하면 된다.
- 하지만, 이 때 비용이 발생한다.
- 이 비용은 어떤 발전소에서 어떤 발전소를 재시작하느냐에 따라 다르다.
- 적어도 P개의 발전소가 고장나 있지 않도록, 발전소를 고치는 비용의 최솟값을 구하는 프로그램을 작성하시오.
- $N \leq 16$

발전소

https://www.acmicpc.net/problem/1102

- D[i] = 발전소의 상태를 i로 만들기 위해 필요한 최소 비용
- i = 발전소를 이진수로 나타낸 상태 (1: 켜있음)
- D[i | (1<<k)] = D[i] + A[j][k]
- j는 i에서 켜져있는 발전소
- k는 i에서 꺼져있는 발전소

발전소

https://www.acmicpc.net/problem/1102

- C/C++
    - https://gist.github.com/Baekjoon/294d0d25dc977edae05e
- Java
    - https://gist.github.com/Baekjoon/02407555bf667a39b7bb

계단 수

https://www.acmicpc.net/problem/1562

- N이 주어질 때, 길이가 N이면서 0에서 9가 모두 등장하는 계단 수가 총 몇 개 있는지 구하기

쉬운 계단 수

https://www.acmicpc.net/problem/10844

- 인접한 자리의 차이가 1이 나는 수를 계단 수라고 한다
- 예: 45656
- 길이가 N인 계단 수의 개수를 구하는 문제

쉬운 계단 수

https://www.acmicpc.net/problem/10844

- D[i][j] = 길이가 i이가 마지막 숫자가 j인 계단 수의 개수
- D[i][j] = D[i-1][j-1] + D[i-1][j+1]

계단 수

https://www.acmicpc.net/problem/1562

- D[N][M][S] = 길이가 N이고, M으로 끝나는 계단수, 지금까지 나온 수의 집합: S

- D[N+1][M+1][S | (1<<(M+1))] += D[N][M][S]
- D[N+1][M-1][S | (1<<(M-1))] += D[N][M][S]

계단 수

https://www.acmicpc.net/problem/1562

- C/C++
  - https://gist.github.com/Baekjoon/57766fcb3bb259e0d9da
- Java
  - https://gist.github.com/Baekjoon/eb4cc4fbb5c7de035627

박성원

https://www.acmicpc.net/problem/1086

- 서로 다른 정수로 이루어진 집합이 있다
- 이 집합의 순열을 합치면 큰 정수 하나를 만들 수 있다
- 예를 들어, {5221,40,1,58,9}로 5221401589를 만들 수 있다
- 합친수가 정수 K로 나누어 떨어지는 순열을 구하는 프로그램을 작성하시오
- 그냥 랜덤하게 순열 하나를 정답이라고 출력하려고 한다
- 이 문제에는 정답이 여러 개 있을 수도 있고, 우연히 문제의 정답을 맞출 수도 있다.
- 우연히 정답을 맞출 확률을 분수로 출력하는 프로그램을 작성하시오

박성원

https://www.acmicpc.net/problem/1086

- 서로 다른 정수로 이루어진 집합이 있다
- 이 집합의 순열을 합치면 큰 정수 하나를 만들 수 있다
- 예를 들어, {5221,40,1,58,9}로 5221401589를 만들 수 있다
- 합친수가 정수 K로 나누어 떨어지는 순열을 구하는 프로그램을 작성하시오
- 그냥 랜덤하게 순열 하나를 정답이라고 출력하려고 한다
- 이 문제에는 정답이 여러 개 있을 수도 있고, 우연히 문제의 정답을 맞출 수도 있다.
- 우연히 정답을 맞출 확률을 분수로 출력하는 프로그램을 작성하시오

박성원

https://www.acmicpc.net/problem/1086

- D[S][M] = 사용한 순열에 포함된 수의 집합이 S이고, 나머지가 M인 것의 개수
- S에 포함되어 있지 않은 수를 L이라고 했을 때
- L번째 수 : A[L]
- L번째 수의 길이: Len[L]
- L번째 수가 포함된 경우의 나머지 = (M * 10^(Len[L]) + A[L]) % M
- D[S | (1<<L)][next] += D[S][M]

박성원

https://www.acmicpc.net/problem/1086

- C/C++: https://gist.github.com/Baekjoon/7bd682332aba9ca1273ff83299082f05

격자판 채우기

https://www.acmicpc.net/problem/1648

- N×M 격자판을 2×1 크기의 도미노로 채우는 방법의 수
- $1 \leq N, M \leq 14$

격자판 채우기

https://www.acmicpc.net/problem/1648

- go(N, S) = N번째 칸을 채울 것이고, N번째 칸부터 칸 M개의 상태가 S일 때, 경우의 수
- N번째 칸?
    - Row-major order 순서

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- go(N, S) = N번째 칸을 채울 것이고, N번째 칸부터 칸 M개의 상태가 S일 때, 경우의 수
- N = 7인 경우 상태 S가 나타내는 범위 7~12번 칸 (M개 칸)

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- go(N, S) = N번째 칸을 채울 것이고, N번째 칸부터 칸 M개의 상태가 S일 때, 경우의 수
- N = 7인 경우 상태 S가 나타내는 범위 7~12번 칸 (M개 칸)
- S = 4인 경우: 4는 2진수로 000100(2) 이다. 따라서, 9번 칸만 이미 채워져 있는 상태

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- go(N, S) = N번째 칸을 채울 것이고, N번째 칸부터 칸 M개의 상태가 S일 때, 경우의 수
- N = 7인 경우 상태 S가 나타내는 범위 7~12번 칸 (M개 칸)
- S = 5인 경우: 5는 2진수로 000101(2) 이다. 따라서, 7, 9번 칸이 이미 채워져 있는 상태

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- go(N, S) = N번째 칸을 채울 것이고, N번째 칸부터 칸 M개의 상태가 S일 때, 경우의 수
- N = 7인 경우 상태 S가 나타내는 범위 7~12번 칸 (M개 칸)
- S = 13인 경우: 13은 2진수로 001101(2) 이다. 따라서, 7, 9, 10번 칸이 이미 채워져 있는 상태

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- go(N, S) = N번째 칸을 채울 것이고, N번째 칸부터 칸 M개의 상태가 S일 때, 경우의 수
- N = 7인 경우 상태 S가 나타내는 범위 7~12번 칸 (M개 칸)
- S = 0인 경우: 0은 2진수로 000000(2) 이다. 따라서, 모두 비어있는 상태

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- 왜 M개를 저장할까?
- N번째 칸에 블럭을 놓는 경우에 항상 N번째 칸이 왼쪽 또는 위가 되게 놓기 때문

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- 상태 S에서 이미 1로 되어있는 곳은 아래 그림과 같이 위에서 채웠다는 의미

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- 블럭을 놓을 수 없는 경우
- 7이 이미 채워져 있는 경우
- 상태 S를 이진수로 나타냈을 때, 마지막 비트가 1인 경우 ((S & 1) == 1)
- 이런 경우에는 채울 수 없기 때문에, 다음 칸을 채워야 함
- S를 오른쪽으로 한 비트 shift 해야함 go(N+1, (S>>1))

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- 블럭을 놓을 수 있는 경우 (1x2를 놓는 경우)
- S & 1 과 S & 2 모두 0이어야 함
- 이 때, 가장 오른쪽 칸인지 아닌지 확인하는 절차도 필요
- go(N+2, (S>>2))

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- 블럭을 놓을 수 있는 경우 (2x1를 놓는 경우)
- 현재 칸이 비어있으면 항상 가능함
- go(N+1, (S>>1) | (1 << (M-1))

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17

격자판 채우기

https://www.acmicpc.net/problem/1648

- 올바르게 채웠는지는 어떻게 확인하나?
- N*M번째 칸이 존재한다고 가정
- N*M번째 칸에서 상태가 S이면 올바르게 채운 것!

0	1	2	3	4	5
6	7	8	9	10	11
12	13	14	15	16	17
18					

격자판 채우기

https://www.acmicpc.net/problem/1648

- C/C++: https://gist.github.com/Baekjoon/b575f09e963137fd2b1e
- Java: https://gist.github.com/Baekjoon/1510cabfd96f2fe8c26a00cc61008606

연습시즌

https://www.acmicpc.net/problem/9023

- 첫 점화식
- D[N] = N일만에 연습 시즌을 마칠 수 있는 최소 비용
- 팀이 2갠데 왜 점화식이 일차원일까?

연습시즌

https://www.acmicpc.net/problem/9023

- 두 번째 점화식
- D[x][y] = X팀이 x번째 연습, Y팀이 y번째 연습을 했을 때 최소비용
- 그런데… 중간에 휴식이란 조건을 어떻게 처리해야할지 모르겠다.

연습시즌

https://www.acmicpc.net/problem/9023

- 세 번째 점화식
- D[N][x][y] = N일에 X팀이 x번째 연습까지 했고, Y팀이 y번째 연습까지 했을 때 최소비용
- 문제 조건 중에 연속해서 쉬면 할인해준다는 조건이 있다. 이를 처리하지 못한다.

연습시즌

https://www.acmicpc.net/problem/9023

- 마지막 점화식
- 연속 휴식을 처리하기 위해
- D[N][x][y][sx][sy] = N일에 X팀이 x번째 연습까지 했고, Y팀이 y번째 연습까지 했을 때 최소비용, (sx, sy가 0인 경우, 해당 팀의 첫 휴식, 1인 경우 해당 팀이 연속으로 휴식)

연습시즌

https://www.acmicpc.net/problem/9023

- 각 팀이 할 수 있는 선택은
- 연습을 한다/안한다
- 따라서 2*2 = 4가지 경우가 있다

연습시즌

https://www.acmicpc.net/problem/9023

- 두 팀이 모두 연습을 하는 경우
- 연습을 했기 때문에, 두 팀 모두 다음 연습을 해야 한다.
- 지금 X팀은 x, Y팀은 y 연습을 했기 때문에, 다음 연습은 x+1, y+1
- 휴식이 아니기 때문에 0, 0
- go(N+1, x+1, y+1, 0, 0) + 2*C
  - x와 y가 같은 곳에서 연습을 하는 경우는 +C

연습시즌

https://www.acmicpc.net/problem/9023

- X팀만 연습을 하는 경우 (Y팀은 휴식)
- 지금 X팀은 x연습을 했기 때문에, 다음 연습은 x+1, y
- X는 연습, Y는 휴식이기 때문에, 다음 상태는 0, 1
- go(N+1, x+1, y, 0, 1) + C + d
  - 이 때, sy 가 0인 경우
  - 즉 연속 휴식이 아닌 경우에는 D를 더해줘야 한다
- Y팀만 연습을 하는 경우도 이와 비슷하게 처리

연습시즌

https://www.acmicpc.net/problem/9023

- 두 팀 모두 휴식을 취하는 경우
- 모두 휴식을 취했기 때문에, 다음 연습은 x, y
- X는 휴식, Y는 휴식이기 때문에, 다음 상태는 1, 1
- go(N+1, x, y, 1, 1) + C + d
  - 이 때, sx 또는 sy 가 0인 경우
  - 즉 연속 휴식이 아닌 경우에는 D를 더해줘야 한다

연습시즌

https://www.acmicpc.net/problem/9023

- 문제에 day제한이 없는데 어디까지?
- X의 연습 개수 + Y의 연습 개수이다
- 이게 최대이고, 그보다 더 큰 값은 정답이 될 수 없다
- 이 값을 넘으면 중간에 반드시 두 팀이 휴식을 취하는 일이 추가될 수 밖에 없기 때문

연습시즌

https://www.acmicpc.net/problem/9023

- C/C++
    - https://gist.github.com/Baekjoon/56b1ebd983bfc91d06af

팰린드롬 경로

https://www.acmicpc.net/problem/2172

- D[N][x1][y1][x2][y2] = (x1,y1)에서 시작해서 (x2,y2)에서 도착하는 길이가 N인 팰린드롬 경로의 개수
- 길이가 1인 팰린드롬 경로
- D[N][x][y][x][y] = 1
- 자기 자신은 1이기 때문
- 길이가 2인 팰린드롬 경로
- D[N][x1][y1][x2][y2] = (A[x1][y1] == A[x2][y2])
- 이 때, (x1,y1)에서 (x2,y2)로 이동할 수 있어야 함

팰린드롬 경로

https://www.acmicpc.net/problem/2172

- D[N][x1][y1][x2][y2] += D[N][x3][y3][x4][y4]
  - 필요한 조건
  - A[x1][y1] == A[x2][y2]
  - (x1,y1), (x3,y3)은 인접해야 함
  - (x2,y2), (x4,y4)도 인접해야 함

팰린드롬 경로

https://www.acmicpc.net/problem/2172

- C/C++
    - https://gist.github.com/Baekjoon/71a6bab0151a6d5e5252

여러가지 방법

개근상

https://www.acmicpc.net/problem/1563

- 개근상을 받을 수 없는 사람은 지각을 두 번 이상 했거나, 결석을 세 번 연속으로 한 사람이다.
- 한 학기가 4일이고, O를 출석, L을 지각, A를 결석이라고 했을 때, 개근상을 받을 수 있는 출결정보는

OOOO OOOA OOOL OOAO OOAA OOAL OOLO OOLA OAOO OAOA

OAOL OAAO OAAL OALO OALA OLOO OLOA OLAO OLAA AOOO

AOOA AOOL AOAO AOAA AOAL AOLO AOLA AAOO AAOA AAOL

AALO AALA ALOO ALOA ALAO ALAA LOOO LOOA LOAO LOAA

LAOO LAOA LAAO

개근상 – 방법 1

https://www.acmicpc.net/problem/1563

- D[Day][Now][Prev][Prev2][Late]
- Day일, 지금 상태가 Now이고, 전날 상태가 Prev, 전전날 상태가 Prev2이고, 지각이 Late 번 했을 때
- (상태 : 출석(0), 결석(1), 지각(2))

개근상 – 방법 1

https://www.acmicpc.net/problem/1563

- D[Day][Now][Prev][Prev2][Late]
- Day일, 지금 상태가 Now이고, 전날 상태가 Prev, 전전날 상태가 Prev2이고, 지각이 Late 번 했을 때
- (상태 : 출석(0), 결석(1), 지각(2))
- 경우의 수

1. 오늘 출석을 했을 때
2. 오늘 결석을 했을 때
3. 오늘 지각을 했을 때

개근상 – 방법 1

https://www.acmicpc.net/problem/1563

- D[3][Now][Prev][Prev2][1] = 1
  - Now ==2 || Prev == 2 || Prev2 == 2
- D[3][Now][Prev][Prev2][0] = 1
  - else

개근상 – 방법 1

https://www.acmicpc.net/problem/1563

- 오늘 출석을 했고, (Now = 0), 전날 Prev, 전전날 Prev2이다.
- 이 것은 D[Day][0][Prev][Prev2]와 같다.
- 그럼 Day-1날에는
- D[Day-1][Prev][Prev2][Prev3]와 같을 것이다.
- 따라서, 오늘 출석을 했고, 지각을 0번 했다면
- D[Day][0][Prev][Prev2][0] += D[Day-1][Prev][Prev2][Prev3][0]
- 오늘 출석을 했고, 지각을 1번 했다면
- D[Day][0][Prev][Prev2][1] += D[Day-1][Prev][Prev2][Prev3][1]

개근상 – 방법 1

https://www.acmicpc.net/problem/1563

- 오늘 결석을 했고 지각을 0번, 1번 했다면
- D[Day][1][Prev][Prev2][0] += D[Day-1][Prev][Prev2][Prev3][0]
- D[Day][1][Prev][Prev2][1] += D[Day-1][Prev][Prev2][Prev3][1]
- 오늘 지각을 했으면, 지각을 0번 했을 경우는 없다.
- D[Day][2][Prev][Prev2][1] += D[Day-1][Prev][Prev2][Prev3][0]

개근상 – 방법 1

https://www.acmicpc.net/problem/1563

- 정답은 D[Day][i][j][k][l]에 들어있는 모든 값을 더하면 된다.
- Day=N
- 0<=i<=2
- 0<=j<=2
- 0<=k<=2
- 0<=l<=1

개근상 – 방법 1

https://www.acmicpc.net/problem/1563

- C/C++
    - https://gist.github.com/Baekjoon/a41631e86703c6efd28b

개근상 – 방법 2

https://www.acmicpc.net/problem/1563

- D[Day][i][j][k]
- 수업이 총 Day개이고, 결석을 연속으로 i번, 지각을 j번, Day날의 출결정보(k)가 출석(0), 결석(1), 지각(2) 일 때
- 경우의 수

1. 오늘 출석을 했을 때
2. 오늘 결석을 했을 때
3. 오늘 지각을 했을 때

개근상 – 방법 2

https://www.acmicpc.net/problem/1563

- 첫 날 가능한 경우는 3가지
- 1. 출석을 했을 때
- D[1][0][0][0] = 1
- 2. 결석을 했을 때
- D[1][1][0][1] = 1
- 3. 지각을 했을 때
- D[1][0][1][2] = 1

개근상 – 방법 2

https://www.acmicpc.net/problem/1563

- 오늘 출석을 했다면, 오늘을 포함해서 결석을 연속해서 할 수 없기 때문에 i = 0
- 이제, 지각을 0번 했을 때와 1번 했을 때로 나눌 수 있다.
- 지각을 0번 했을 때는 전날 할 수 있는 출결을 출석과 결석밖에 없기 때문에
- D[i][0][0][0] = D[i-1][0][0][0] + D[i-1][1][0][1] + D[i-1][2][0][1]
- 지각을 1번 했을 때는 위의 경우 + 지각을 하는 경우
- D[i][0][1][0] = D[i-1][0][1][0] + D[i-1][1][1][1] + D[i-1][2][1][1] + D[i-1][0][1][2]

개근상 – 방법 2

https://www.acmicpc.net/problem/1563

- 오늘 결석을 했다면
- 오늘이 첫 번째 결석일 때와 두 번째 연속 결석일 때로 나눌 수 있다
- 각각의 경우에서 출석과 비슷하게 지각을 0번 했을 때, 1번 했을 때로 나눌 수 있다
- 오늘이 첫 번째 결석이고, 지각을 한 번도 한 적이 없으면, 전날은 반드시 출석이어야 한다
- D[i][1][0][1] = D[i-1][0][0][0]
- 오늘이 두 번째 결석이고, 지각을 한 번도 한 적이 없다면, 전날은 반드시 결석이어야 한다
- D[i][2][0][1] = D[i-1][1][0][1]

개근상 – 방법 2

https://www.acmicpc.net/problem/1563

- 오늘 결석을 했다면
- 오늘이 첫 번째 결석일 때와 두 번째 연속 결석일 때로 나눌 수 있다
- 각각의 경우에서 출석과 비슷하게 지각을 0번 했을 때, 1번 했을 때로 나눌 수 있다
- 오늘이 첫 번째 결석이고, 지각을 한 번 한 적이 있으면, 전날은 반드시 출석이거나 지각이다
- D[i][1][1][1] = D[i-1][0][1][0] + D[i-1][0][1][2]
- 오늘이 두 번째 결석이고, 지각을 한 번 한적이 있어도, 전날은 반드시 결석이다
- D[i][2][1][1] = D[i-1][1][1][1]

개근상 – 방법 2

https://www.acmicpc.net/problem/1563

- 오늘 지각을 했다면
- 전날까지는 지각이 있으면 안되고, 전날은 출석이나 결석이어야 한다
- D[i][0][1][2] = D[i-1][0][0][0] + D[i-1][1][0][1] + D[i-1][2][0][1]

개근상 – 방법 2

https://www.acmicpc.net/problem/1563

- 정답은 D[Day][i][j][k]에 들어있는 모든 값을 더하면 된다.
- Day=N
- 0<=i<=2
- 0<=j<=1
- 0<=k<=2

개근상 – 방법 2

https://www.acmicpc.net/problem/1563

- C/C++
  - https://gist.github.com/Baekjoon/0499aaafb64720beb6a3

개근상 – 방법 3

https://www.acmicpc.net/problem/1563

- D[Day][Late][Prev][Now]
- Day일에 지각은 Late번, 전날 상태: Prev, 오늘 상태: Now
- 출석: 0
- 지각: 1
- 결석: 2

개근상 – 방법 3

https://www.acmicpc.net/problem/1563

- 초기화
- D[2][i%2+j%2][i][j] = 1
- 여기서, i%2와 j%2는 지각의 횟수를 의미하게 된다.
- i%2 + j%2는 항상 1보다 작거나 같아야 한다.

개근상 – 방법 3

https://www.acmicpc.net/problem/1563

- D[Day][Late+Now%2][Prev][Now] += D[Day-1][Late][Prev2][Prev]
- $0 \leq Prev \leq 2$
- $0 \leq Now \leq 2$
- $0 \leq Late+Now\%2 \leq 1$
- $0 \leq Prev2 \leq 2$

개근상 – 방법 3

https://www.acmicpc.net/problem/1563

- 지각을 1로 나타내게 되면, %2로 지각인지 아닌지를 쉽게 판별 할 수 있다.
- 결석이 연속으로 3번인 것의 처리는
- Prev + Now + Prev2 == 6 으로 처리할 수 있다.
- 정답은
- D[Day][Late][Prev][Now]에 저장되어 있는 모든 값을 더하면 된다.
- Day = N
- $0 \leq Late \leq 1$
- $0 \leq Prev \leq 2$
- $0 \leq Now \leq 2$

개근상 – 방법 3

https://www.acmicpc.net/problem/1563

- C/C++
    - https://gist.github.com/Baekjoon/5bedb91fe88e1f2c8117

개근상 – 방법 4

https://www.acmicpc.net/problem/1563

- D[i][j][k] = i일, 지각 j번, 연속 결석 k번
- k = 0인 경우에는 출석 또는 지각
- k = 1, 2인 경우는 결석 (연속 결석이 1번, 2번)

개근상 – 방법 4

https://www.acmicpc.net/problem/1563

- D[i][j][k] = i일, 지각 j번, 연속 결석 k번
- D[0][0][0] = 1
- 오늘 출석을 했으면, 전날에 출석, 지각, 결석 중에 하나만 하면 된다
- D[i][j][0] += D[i-1][j][k] (0 ≤ k ≤ 2)
- 오늘 지각을 하려면 j 가 1이어야 한다
- D[i][1][0] += d[i-1][0][k]
- 오늘 결석을 하려면 k < 2이어야 한다
- D[i][j][k+1] += D[i-1][j][k]

개근상 – 방법 4

https://www.acmicpc.net/problem/1563

- D[i][j][k] = i일, 지각 j번, 연속 결석 k번
- 정답은
- i = N
- $0 \le j \le 1$
- $0 \le k \le 2$
- 에 저장되어 있는 모든 값을 합하면 된다.

개근상 – 방법 4

https://www.acmicpc.net/problem/1563

- C++
    - https://gist.github.com/Baekjoon/e3a40182e752d9d5c1f4

개근상 – 방법 5

https://www.acmicpc.net/problem/1563

- D[i][j] = i일, 연속 결석이 j번
- 먼저, 지각은 없다고 가정하고 문제를 푼다
- j = 0이면 출석
- j = 1 or 2 이면 결석

개근상 – 방법 5

https://www.acmicpc.net/problem/1563

- D[i][j] = i일, 연속 결석이 j번
- 오늘 출석을 하는 경우에는 전날 아무거나 해도 상관이 없다.
- D[i][0] += D[i-1][j] ($0 \le j \le 2$)
- 오늘 첫 연속 결석을 하는 경우라면, 전날은 출석이어야 한다.
- D[i][1] = D[i-1][0]
- 오늘 한 결석이 연속 두 번째 결석이라면, 전날은 첫 연속 결석이어야 한다.
- D[i][2] = D[i-1][1]
- 정답은 D[n][0] + D[n][1] + D[n][2] 가 된다

개근상 – 방법 5

https://www.acmicpc.net/problem/1563

- D[i][j] = i일, 연속 결석이 j번
- 이제 지각을 포함하는 경우를 생각해보자
- 지각은 딱 한번만 할 수 있기 때문에 다음과 같이 생각할 수 있다.

1		지각		N

개근상 – 방법 5

https://www.acmicpc.net/problem/1563

- 지각을 i일에 했다고 하면, 지각을 기준으로 왼쪽과 오른쪽으로 나눌 수 있다
- 지각은 1번만 해야 하기 때문에, 왼쪽과 오른쪽에는 지각이 있으면 안된다

1		지각		N

개근상 – 방법 5

https://www.acmicpc.net/problem/1563

- 지각을 i일에 했다고 하면, 지각을 기준으로 왼쪽과 오른쪽으로 나눌 수 있다
- 지각은 1번만 해야 하기 때문에, 왼쪽과 오른쪽에는 지각이 있으면 안된다
- 왼쪽은 총 i-1일, 오른쪽은 N-i일 이다.
- 따라서, D[i-1][j] * D[N-i][k] (0 ≤ j ≤ 2, 0 ≤ k ≤ 2) 가 지각을 i일에 했을 때의 경우의 수이다

i-1	i	N-i

개근상 – 방법 5

https://www.acmicpc.net/problem/1563

- C++
    - https://gist.github.com/Baekjoon/76d2a384dfb3796ade88